الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

CONTRIBUTION A LA COMMUNICATION RAINBOW 100-M24 et M24-M24

Proposé par :
Dr. F. BRIKCI

Etudié par :
Mr NOUR A.
Mr DJELOUAH K.

Dirigé par :
Dr. F. BRIKCI

PROMOTION :
Janvier 1986

E.N.P. 10, Avenue Hacen Badi – EL-HARRACH – ALGER

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

CONTRIBUTION A LA COMMUNICATION RAINBOW 100-M24 et M24-M24

Proposé par :
Dr. F. BRIKCI

Etudié par :
Mr NOUR A.
Mr DJELOUAH K.

Dirigé par :
Dr. F. BRIKCI

PROMOTION :
Janvier 1986

E.N.P. 10, Avenue Hacen Badi – EL-HARRACH – ALGER

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

Nous tenons à exprimer nos plus vifs remerciements à notre promoteur monsieur F.BRIKCI pour son aide et ses conseils tout au long de l'élaboration de ce projet de fin d'étude.

Nous remercions aussi Melle KASDI pour ses conseils qui nous ont été très bénéfiques.

Que Mr BENMESSAOUD Hocine trouve ici l'expression de notre profonde gratitude.

Nous tenons aussi à remercier Mme ZERGUINE ,Messieurs HOUHOU et NACER pour tout l'aide qu'ils nous ont fourni pour la réalisation de ce mémoire.

Nous tenons à exprimer notre reconnaissance à tout le corps enseignant pour leur contribution à notre formation.

Qu'il nous soit permis d'exprimer ici nos sincères remerciements à tout le personnel du centre de calcul pour l'acceuil qui nous a été accordé.

D E D I C A C E S

Je dédie,ce modeste travail,à mes parents,grands parents, frères et soeurs,à mon cousin qui m'a tant aidé.

KAMEL

Je dédie,ce travail,à mes parents,grands parents,mon frère et mes soeurs.

ABDERRAHMANE

CHAPITRE 1 : LES MICROPROCESSEURS 16 BITS 8086/8088.

1.1 Etude et description.
1.1.1 Caractéristiques globales du 8086/8088.
1.1.2 Bus du 8086/8088.
1.1.2.1 Multiplexage du bus de données.
1.1.3 Organisation de la mémoire.
1.1.4 Architecture interne.
1.1.5 La segmentation.
1.1.6 Les registres internes.
1.1.6.1 Les registres généraux.
1.1.6.2 Les registres pointeurs(Base et Index).
1.1.6.3 Les registres segments.
1.1.6.4 Le mot d'état.
1.1.7 Les interruptions.
1.1.7.1 Vectorisation.
1.1.7.2 Mécanisme des interruptions.
1.1.8 Différence entre le 8086 et le 8080.
1.2 Le langage assembleur.
1.2.1 Généralités.
1.2.2 Les modes d'adressages.
1.2.3 Quelques instructions de base du 8086/8088.
1.3 L'ASM86.
1.3.1 Principales directives de l'ASM86.
1.4 Le MS GWBASIC.

CHAPITRE 2 : LES SYSTEMES D'EXPLOITATION.

2.1 Concepts fondamentaux des systèmes d'exploitation.
2.1.1 Introduction.
2.1.2 Utilisation des systèmes d'exploitation.
2.1.3 Evolution historique des systèmes d'exploitation.
2.1.4 Construction logicielle des systèmes d'e"xploitation.
2.1.4.1 Noyau du système d'exploitation.
2.1.4.2 Sémaphore.
2.1.4.3 Ordonnanceur
2.1.4.4 Chargeur
2.1.4.5 Bibliothèque
2.1.4.6 Moniteur
2.1.4.7 Gestion des programmes
2.1.4.8 Gestion des fichiers
2.1.4.9 Gestion des E/S
2.2 Le système d'exploitation CP/M
2.2.1 Introduction
2.2.2 Les commandes de CP/M
2.2.3 L'interface de contrôle
2.2.4 Allocation mémoire
2.2.5 Description de CP/M
2.2.5.1 Le système de fichiers
2.2.5.2 Fonctionnement et exécution de CP/M
2.2.5.3 Les fonctions FDOS
2.2.5.3.1 Les fonctions BDOS
2.2.5.3.2 Le BIOS
2.2.6 Le BOOTSTRAP
2.3 Le système d'exploitation MSDOS
2.3.1 Introduction
2.3.2 Manipulation des fichiers
2.3.3 Le traitement par lots
2.3.4 Bref aperçu sur le DOS

CHAPITRE 3:DESCRIPTION DES PRINCIPALES COMPOSANTES DU MODULE SYSTEME

3.1 Description générale
3.2 Description fonctionnelle
3.2.1 Bus d'adresse et de données du module système
3.2.2 Mémoire du module système
3.2.2.1 Les 64 KO de mémoire partagée
3.2.2.2 Les 24 KO de ROM
3.2.2.3 Mémoire non volatile
3.2.2.4 RAM reservée au Z80A
3.3 Interface d'E/S MPSC 7201
3.3.1 Description des principaux pins pour un canal
3.3.2 Description générale
3.3.3 Description fonctionnelle
3.3.4 Le canal de communication du MPSC
3.4 Les décodeurs d'E/S
3.5 Le circuit de détection MHFU
3.6 Connecteurs du module système
3.7 Les interruptions du 8088

CHAPITRE 4:LA TRANSMISSION DE DONNEES

4.1 Nature des informations à échanger entre les terminaux
4.1.1 Codage des informations
4.1.1.1 Le code ASCII
4.1.1.2 Problèmes de transmissions
4.1.1.3 La transmission asynchrone
4.1.1.4 Circuit de données
4.1.1.5 Procédures d'E/S
4.2 La norme RS 232
4.3 Modification des paramétres de communication par le mode de fonctionnement du RAINBOW 100
4.4 Protocoles de communication
4.5 La compatibilité IBM PC

CHAPITRE 5:REALISATION DE LA TRANSMISSION DE FICHIERS OLIVETTI M24 -OLIVETTI M24 ET RAINBOW 100-OLIVETTI M24
INTRODUCTION
5.1 Les possibilités offertes par CP/M
5.2 possibilités offertes par MS-DOS
5.3 Application du MS-GWBASIC dans la mise au point du programme de transfert de fichiers RAINBOW OLIVETTI et OLIVETTI OLIVETTI
5.4 Application de l'ASM 86 et des fonctions BIOS et BDOS dans la mise au point d'un programme de transfert de fichiers sur le RAINBOW 100

INTRODUCTION.

Si l'on peut considérer que la transmission à distance d'informations aplhanumériques est aussi ancienne que le télégraphe,ce n'est que vers les années soixante que s'est vraiment développée la connexion des calculateurs à des lignes de télécommunications,dans le but d'accéder au traitement à partir de périphériques distants appelés terminaux.Limités au début à quelques liaisons isolées,des réseaux de transmission de données complexes se sont développés autour de centres informatiques permettant d'effectuer la saisie et la restitution de l'information plus près des sources et des utilisateurs de cette dernière.

Comme alternative à de telles structures de nature centralisée,apparaît actuellement la tendance de l'informatique répartie,dans laquelle le traitement et le stockage de l'information sont placés plus près des utilisateurs et sont répartis sur plusieurs ordinateurs plus petits ou sur des terminaux pouvant échanger des données entre eux par l'intermédiaire de moyens de télécommunications.

Les développements récents survenus dans le domaine des systèmes d'information,des ordinateurs et des micro-ordinateurs ont fait ressortir la nécessité de la transmission de données.

Les utilisateurs du centre de calcul étants amenés à travailler sur des microordinateurs différents selon leurs disponibilité,la réalisation d'un réseau permettant l'accés aux différentes informations s'est faite ressentir.

C'est dans ce contexte que s'inscrit notre contribution à la réalisation d'une transmission de données entre les micro-ordinateurs Rainbow100 et OLIVETTI M24 ainsi qu'entre deux M24.

L'architecture du Rainbow 100 étant basée autour du microprocesseur 8088;et celle du M24 autour du micro-processeur 8086;nous avons commencé par l'étude de la structure des deux microprocesseurs puis celle du langage assembleur associé.L'absence de documentation technique sur le M24 nous a contraint à ignorer sa structure interne. Nous avons donc approché le problème par une programmation en langage BASIC lequelle on disposait d'une documentation suffisante (l'assembleur n'étant pas disponible sur le M24) (Chapitre 1).

L'étude des systèmes d'exploitation CP/M et MS-DOS gérant respectivement le Rainbow 100 et le M24,nous a permis de connaitre les ressources qu'ils offraient quant à la transmission de données(chapitre 2).

De plus,l'étude du module système qui nous a permis de localiser l'interface d'E/S et d'étudier ses liaisons avec le connecteur de communication (chapitre 3).

Le travail qui nous a été confié consiste en la réalisation d'un transfert de fichiers entre les microordinateurs RAINBOW 100 et OLIVETTI M24.Pour mener à bien cette tâche,il est nécessaire de définir les règles qui régissent les échanges de données et qui assurent l'efficacité et la fiabilité du circuit de données.De même, les signaux de données et les caractéristiques physiques du support de transmission et des interfaces de communication doivent être bien définies.L'ensemble de ces règles constitue les normes et les protocoles de communication.

Dans notre application,nous nous sommes interessés plus particulièrement à la norme RS 232 qui est utilisée par le RAINBOW 100 et le M24(chapitre 4).

Enfin,dans le chapitre 5,nous avons expliqué toute la démarche qui nous a permis d'établir des programmes de transfert de transfert de fichiers entre les deux micro-ordinateurs déjà cités,et tous les résultats obtenus.

CHAPITRE 1: LES MICROPROCESSEURS 16 BITS 8086/8088

1.1-ETUDE ET DESCRIPTION.

1.1.1-Caractéristiques globales du 8086/8088:

Les microprocesseurs 8086(16 bits) et 8088 (pseudo-16 bits) se présentent en boîtier de 40 pins,ce qui ne reflète pas la complexité de l'organisation de la "puce".Ils peuvent en effet adresser 1 M-octets de mémoire ce qui exige 20 lignes pour le bus d'adresse qui est multiplexé avec le bus de données.

Avant d'examiner l'architecture de cette machine dégageons quelques unes de ses principales caractéristiques:

-Espace adressable: .1 M-octets
.64 K d'E/S

-Bus mutipléxé: .Adresses/données/état

-Fonctionnement: .mode minimum:processeur unique sans support multiprocesseur
.Mode maximum:support multiprocesseur local ou distant

-Support logiciel: .Modes d'adressages complexes pour supporter les langages évolués
.Instructions de traitements de châines de caractères
.Multiplication et division cablées
.Entrées/sorties en mode direct

-Interruptions: .256 niveaux externes véctorisés
.1 niveau non masquable
.Interruptions logicielles

1.1.2-Bus du 8086/8088:

Le CPU dispose d'un bus d'adresse de 20 bits,ce qui donne une capacité d'adressage de 1 MO.Le bus de données est un vrai bus de 16 bits (8 bits pour le 8088), c'est-à-dire que les accés sont de 16 bits par cycle machine. L'organisation de la mémoire est donc vue 512 K-mots.L'accés à un octet se fait simplement en éliminant la partie du mot non désirée(octet inférieur ou supérieur).

Les bus sont multiplexés.En effet,le composant ne disposant que de 40 pins,il a été nécessaire d'utiliser les mêmes broches pour y faire circuler les adresses,les états et les données.les informations se succédent dans le bus de manière temporelle,les adresses d'abord puis les données ensuite,impliquant un signal supplémentaire indiquant le type d'information placé sur le bus .
Ce signal ALE (Adress Latch Enable) est employé pour verrouiller les adresses fugitives et produire un bus adresse de 20 bits disponible pendant toute la durée du cycle machine.A la suite des adresses,les 16 bits de poids faible véhiculent les données,les quatre bits de poids fort un état.

*Bus de contrôle:il est composé de 3 lignes:

-IO/M sélectionne mémoire ou entrées/sorties.
-RD lecture
-WR écriture

1.1.2.1-Construction d'un bus d'adresse de 20 bits:

L'extension au-delà de 64 K est conceptuellement simple.On Dsélectionne d'abord un boîtier d'indirection MMU(Memory Management Unit) par une instruction de sortie,puis on se sert des poids forts (5 bits par exemple) de l'adresse qui donnent un numéro de registre dans le boîtier.Ce registre contient une adresse de base pour la page qui,ajoutée au poids faible du bus d'adresses (11 bits),génère une adresse sur 20 bits (fig-1.1.1).

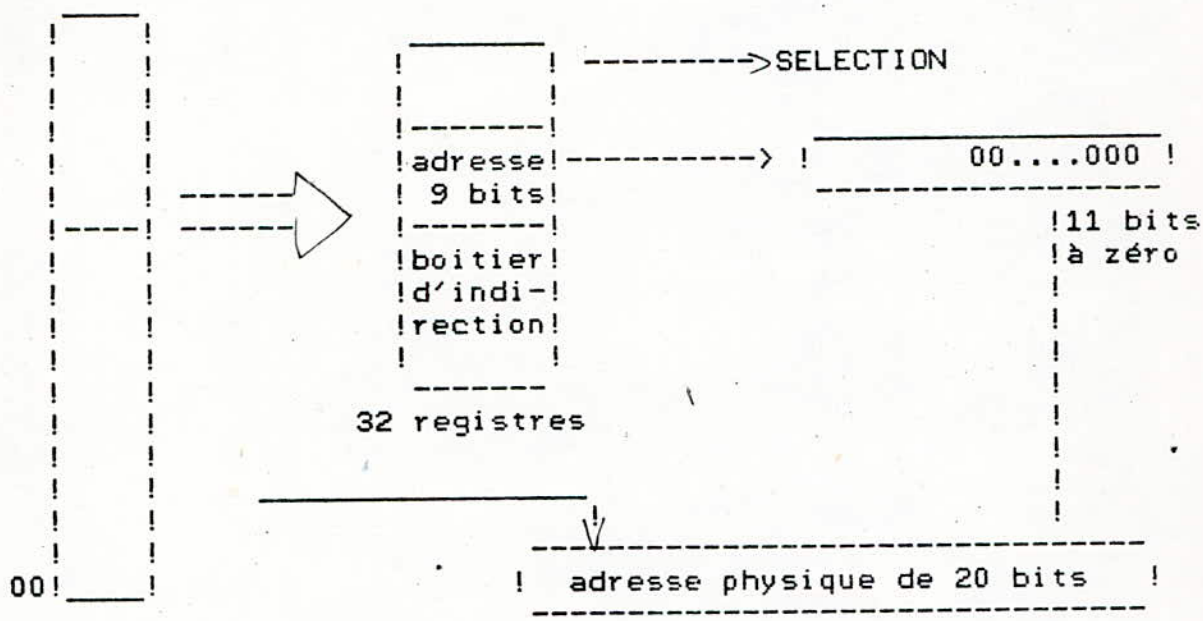


fig-1.1.1-Adressage paginé-

La fabrication d'une adresse valide pendant toute la durée d'un cycle machine est nécessaire.Cette opération est effectuée par un composant spécialisé,le verrou 8282.Ce circuit est transparent il,laisse passer des adresses et les pièges à l'aide du signal ALE.Le bus d'adresse et le signal BHE (dans le cas du 8086) sont ainsi conservés pendant la durée entière d'un cycle.

Outre leur capacité,à verrouiller l'information,ils ont une fonction d'amplification du bus d'adresse.En effet les spécifications des bus du 8086 n'autorisent guère plus de 6 à 8 circuits (mémoire et périphérique) à être connectés sur le bus.

Ces méthodes sont mises en oeuvre soit par du hardware associé,soit par des boîtiers spécialisés(MMU DU MOTOROLA 6809). La multiplication de ces boîtiers donne donc un espace d'adressage important.8 MMU donnent un espace de 8 Méga-Octets.Les logiciels tenant compte de cette

architecture matérielle doivent être spécialement adaptés:compilateurs,éditeurs de lien,outils de tests.

1.1.2.1-Multiplexage du bus de données:

Le microprocesseur pouvant aussi bien recevoir qu'envoyer des données sur le bus de données,il s'agit de multiplexer ces échanges.Pour celà un "contrôleur de bus" est nécessaire , comportant:

-Une porte d'entrée validée par une commande de lecture (READ) générée par le microprcesseur.

-Une porte de sortie validée par une commande d'écriture (WRITE).

-Des buffers d'entrée et de sortie,assurant un stockage momentané (en "tampon") des données transférées.

D'autre part,le contrôleur de bus comporte une logique de contrôle décodant les commandes générées par le microprocesseur, et gérant les opérations de lecture ou d'écriture mémoire(Memory Read ou Memory Write) et d'entrée ou de sortie (I/O Read ou I/O Write). La fig-1.1.3.1 illustre le principe d'un tel contrôleur.

Précisons que certains processeurs nécessitent un module "contrôleur de bus"(bus driver) tel le 8080 d'INTEL et que d'autres plus intégrés (tel le MOTOROLA 6800) réalisent eux-mêmes le multiplexage du bus de données.

Le système d'adressage est basé sur le principe suivant:
Le microprocesseur génère des adresses sur 16 bits.Les 4 bits de poids faible (A0,A1,A2,A3) sont utilisés pour sélectionner le module appelé;en les décodant on peut adresser directement 16 modules différents par 16 signaux SM0,SM1,...,SM15.

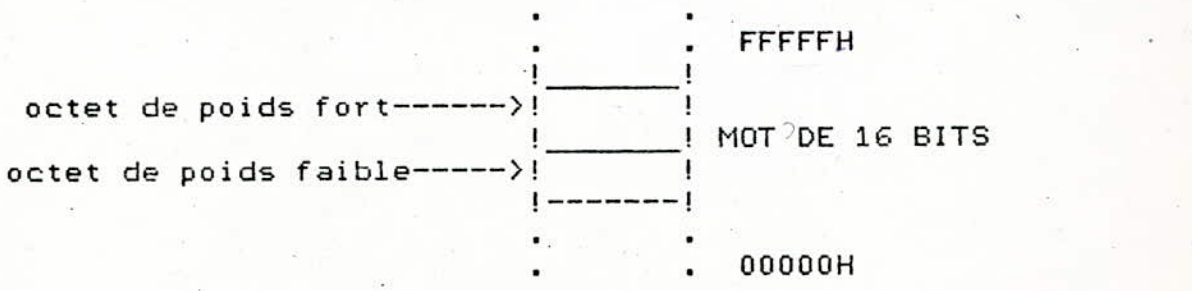
Les 12 bits d'adresses restants (A4-A15) fournissent l'adresse de base pour l'adressage des mots en mémoire.

Il est très important de noter qu'un tel système d'adressage des modules permet au microprocesseur de traiter de la mme façon les échanges avec les mémoires et avec les périphériques.C'est l'avantage essentiel de cette méthode.

1.1.3-Organisation de la mémoire:

Nous n'entrerons pas dans le détail du hardware mais puisque nous aurons à travailler en assembleur,il faut savoir comment le 8086 lit un mot de 16 bits.

Les mots de 16 bits sont rangés avec l'octet de poids fort à l'adresse la plus haute.



Organisée logiquement en 512 K-Mots(1mot=16 bits),la mémoire se divise physiquement en deux bancs de 512 K-octets qui sont accédés par paire (16 bits).Le processeur utilise les adresses A0-A19.Pour la lecture d'un octet le processeur n'utilise que la partie faible ou forte du bus de données.En écriture il faut protéger l'octet non écrit,on emploie pour cela le signal BHE(Bus High Enable) servant à autoriser ou non l'écriture sur la partie forte du bus de données,et le signal A0,pour les poids faibles.La fig-1.1.4.1 donne selon A0 et BHE,les divers accés.

A0	BHE	Type d'accés	bus de données
0	1	Octet d'adresse paire	D0-D7
1	0	Octet d'adresse impaire	D8-D15
0	0	Mot d'adresse paire	D0-D15

fig-1.1.4.1-Types d'accés mémoire-

Un mot rangé rangé à une adresse paire (terminée par 0,2,4,...) est lu en une fois.Un mot rangé à une adresse impaire est lu en deux fois selon le schéma fig-1.1.4.1'.Pour lire un mot de 16 bits,le 8088 qui a un bus de données de 8 bits opère en 2 temps.

1.1.4-Architecture interne:

UNITE D'INTERFACE BUS (UIB)/UNITE D'EXECUTION (UE)

Les processeurs(8086/8088) fonctionnent en mode "pipe-line".Cette expression signifie simplement que leur architecture interne se compose de deux unités dialoguant entre-elles par l'intermédiaire d'un bus d'échange.La première machine regroupe les fonctions de traitement (UAL) ainsi que les registres généraux,elle se compose de plus du décodeur d'instructions.Cette machine dite unité d'exécution (UE),lit les codes opérations dans une file d'attente et les exécute.La file d'attente est alimentée par une autre machine:unité d'interface du bus (UIB).

Le travail principal de l'UIB,qui est de chercher les instructions en mémoire,est rompu dans deux cas particuliers.Le premier est celui ou l'UE sur rencontre d'instruction d'accés mémoire demande à l'UIB de lire ou

d'écrire une valeur en mémoire.Le second est celui ou l'UE rencontre un déroutement de programme(JMP,CALL...),le contenu de la file d'attente n'est plus à jour et l'UE doit demander une réinitialisation à partir de l'adresse de débranchement.La file d'attente des instructions est d'une taille de 6 octets pour le 8086 et 4 pour le 8088,l'accés se fait toujours par mots de 16 bits.Dans le cas particulier ou l'adresse de réinitialisation est impaire,le processeur effectue un accés octet avant de reprendre la recherche du mot.

La fonction primordiale de l'UIB est de générer une adresse physique de 20 bits,ceci est réalisé au moyen de registres de base contenant l'adresse de début de segment,et d'un déplacement sur 16 bits(OFFSET).
La disponibilité des deux unités donne une trés bonne occupation du bus puisqu'à de rares cas prés(instruction longues) les deux machines travaillent en parallèle et le bus est pratiquement toujours actif.

Conséquence:du fait de la séparation de l'UIB et de l'UE,le processeur peut échanger des données avec la mémoire et les périphériques indépendemment des instructions.

1.1.5-La segmentation

Le mécanisme d'adressage du 8088 (8086) utilise une technique de segmentation.La mémoire de 1 MO est découpée logiquement en zones de 64 KO au maximum,dites "segments".Pour accéder à un mot mémoire,il faut disposer de deux quantités de 16 bits:un registre de segment de base(RS) et un déplacement(offset).
L'adresse physique sur 20 bits est alors calculée par: "16*RS+OFFSET"(fig-1.1.6).

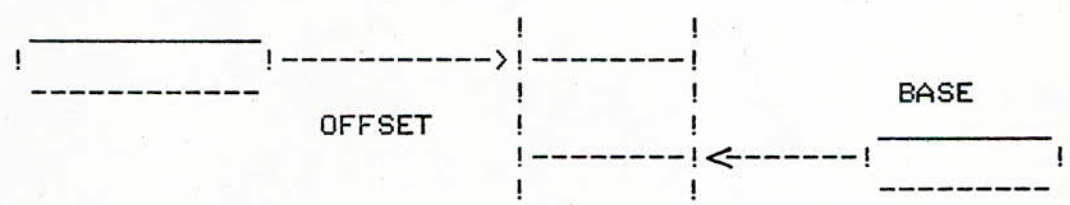


fig-1.1.6-Mécanisme de la segmentation-

Les segments peuvent être,disjoints,partiellement recouverts ou confondus (fig-1.1.6').

mémoire (adresses arbitraires)

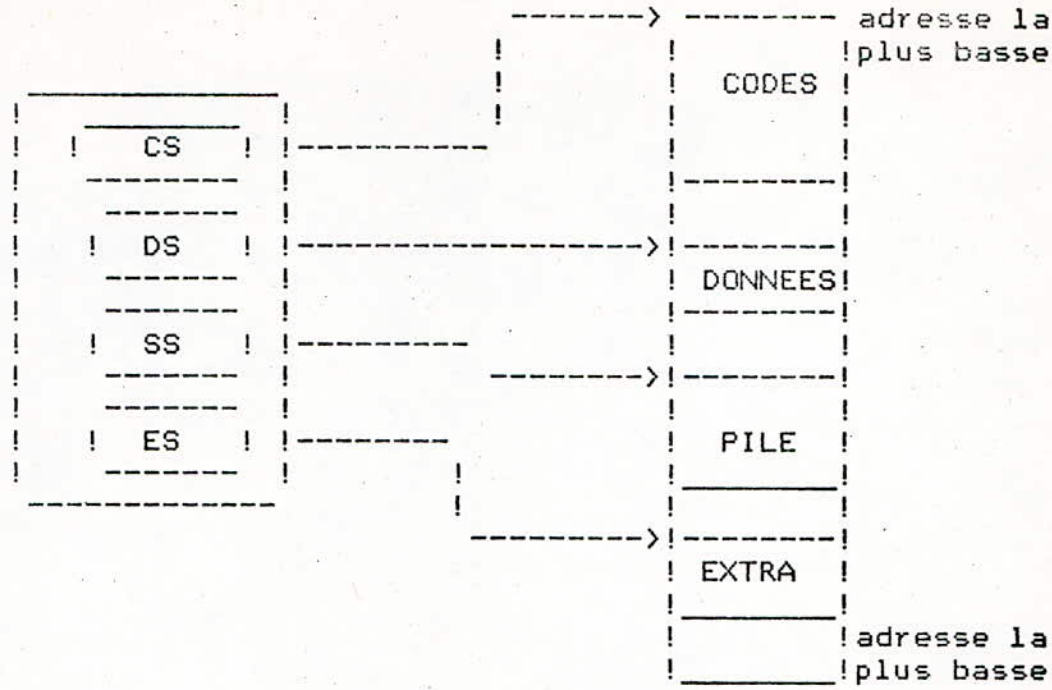


fig-1.1.6'-La segmentation

Ce mécanisme est fortement simplifié car ce calcul est bien entendu effectué par la machine,le programmeur devra en général concentrer son attention sur l'élaboration du seul déplacement.

Le mécanisme de segmentation est étendu aux quatre registres de segment donnant donc une capacité d'adressage réelle de 256 K (4*64K) sans modification des bases(fig-1.1.6').Ces 4 registres sont:

-Un segment de code : adressé par CS:CODE SEGMENT
-Un segment de données: adressé par DS:DATA SEGMENT
-Un segment de pile : adressé par SS:STACK SEGMENT
-Un segment de données supplémentaires: adressé par ES:EXTRA SEGMENT

1.1.6-Registres internes:

Le processeur dispose de 4 types de registres internes:

-Registres généraux
-Registres pointeurs
-Registres de segment
-Registres d'état et compteur ordinal

Le processeur n'est pas une machine orthogonale dans le sens ou il n'est pas possible d'utiliser n'importe quel registre pour n'importe quel instruction,ce n'est pas non plus une machine complétement dédiée car la notion d'accumulateur est fortement étendue.Un compromis a été établi:certains registres sont dédiés pour certaines opérations mais libres pour d'autres.

1.1.6.1-Registres généraux:

Ils sont au nombre de 4 et peuvent travailler par moitié(8 bits).Ils ont pour appellation:

-Accumulateur:AX composé de AH et AL
-Base :BX composé de BH et BL
-Compteur :CX composé de CH et CL
-Données :DX composé de DH et DL

Bien que ces registres soient généraux,ils ont des fonctions très précises en dehors des opérations arithmétiques et logiques classiques comme nous le verrons plus loin:

-BX registre de base pour l'adressage de zones mémoire
-CX et CL servent de compteurs de boucles
-DX permet l'adressage des ports et sert également d'extension à AX (donnés sur 32 bits).

1.1.6.2-Registres pointeurs(Base et Index):

Quatre registres pointeurs de 16 bits(Tableau ci-dessous)
SI(Source Index) ET DI(Destination Index) sont 2 registres d'index qui contiennent les déplacements des données vis-à-vis des registres de base ES et DS.SI et DI impliquent en genéral l'utilisation de DS.Pourtant dans le cas des instructions de chaînes,SI est associé à la zone destination par ES.SP est le pointeur de pile,BP sert de registre de base pour adresser la pile,ces deux derniers registres sont associés à SS.

	Fictif 15 0	
Index "source"	0000	SI
Index "destination"	0000	DI
Poiteur de pile	0000	SP
Base de pile	0000	BP

1.1.6.3-Registres de segment:

Quatre registres de segment de 16 bits (Tableau ci-dessous)

	19 3fictif0	
Segment de CODE	CS	0000
Segment de DATA	DS	0000
Segment de STACK	SS	0000
Segment EXTRA	EX	0000

Ces registres contiennent les 16 bits de poids fort de l'adresse 20 bits du début du segment.Les 4 bits de poids faible sont considérés comme nuls puisqu'un segment débute toujours à une adresse multiple de 16 octets(paragraphe).Une adresse mémoire est donc représentée par deux quantités de 16 bits (base et deplacement).

1.1.6.4-Mot d'état du 8086(flags):

La fig-1.1.6.4 montre la disposition des bits du mot d'état.Les flags sont en principe modifiés par toutes les opérations arithmétiques et logiques mais non par celles de transfert (MOV).(voir document du constructeur).

!	!	!	!	!OF!	DF!	IF!	TF!	SF!	ZF!	!AF!	!PF!	!CF!

*Flags d'état:

CF(Carry Flag):retenue

PF(Parity FLag):parité.Si ce bit est à 1,le résultat de l'opération posséde un nombre pair de 1.Ce flag peut être utilisé dans la détection d'erreurs de transmission.

AF(Auxiliare Flag):retenue auxilliaire. C'est la retenue qui se propage du quartet de poids faible à celui de poids fort.

ZF(zero flag):Indique si la donnée est nulle ou non nulle.

SF(Sign Flag):Indique le signe du nombre.1 pour négatif, 0 pour positif.

OF(Overflow Flag):Indique un dépassement de capacité pour les opérations arithmétiques signées.

*Flags de contrôle:

IF(Interrupt Enable Flag):Autorise ou interdit globalement les interruptions externes.

DF(Direction Flag):Selon que le bit DF vaut 0 ou 1,permet de balayer une suite de données par valeurs croissantes ou décroissantes des adresse.

TF(Trap Flag):Si ce bit est positionné,une interruption Trap
se produit à chaque instruction.Le pas à pas ou Trap permet la mise au point du logiciel sans gestion externe.

1.1.6.4-Compteur ordinal:

Le registre IP,de 16 bits,contient le déplacement de l'instruction courante vis-à-vis du segment de code.L'adresse physique de l'instruction courante est donc :"16*CS+IP".

La fig-1.1.6.4 résume l'architecture interne du 8088.

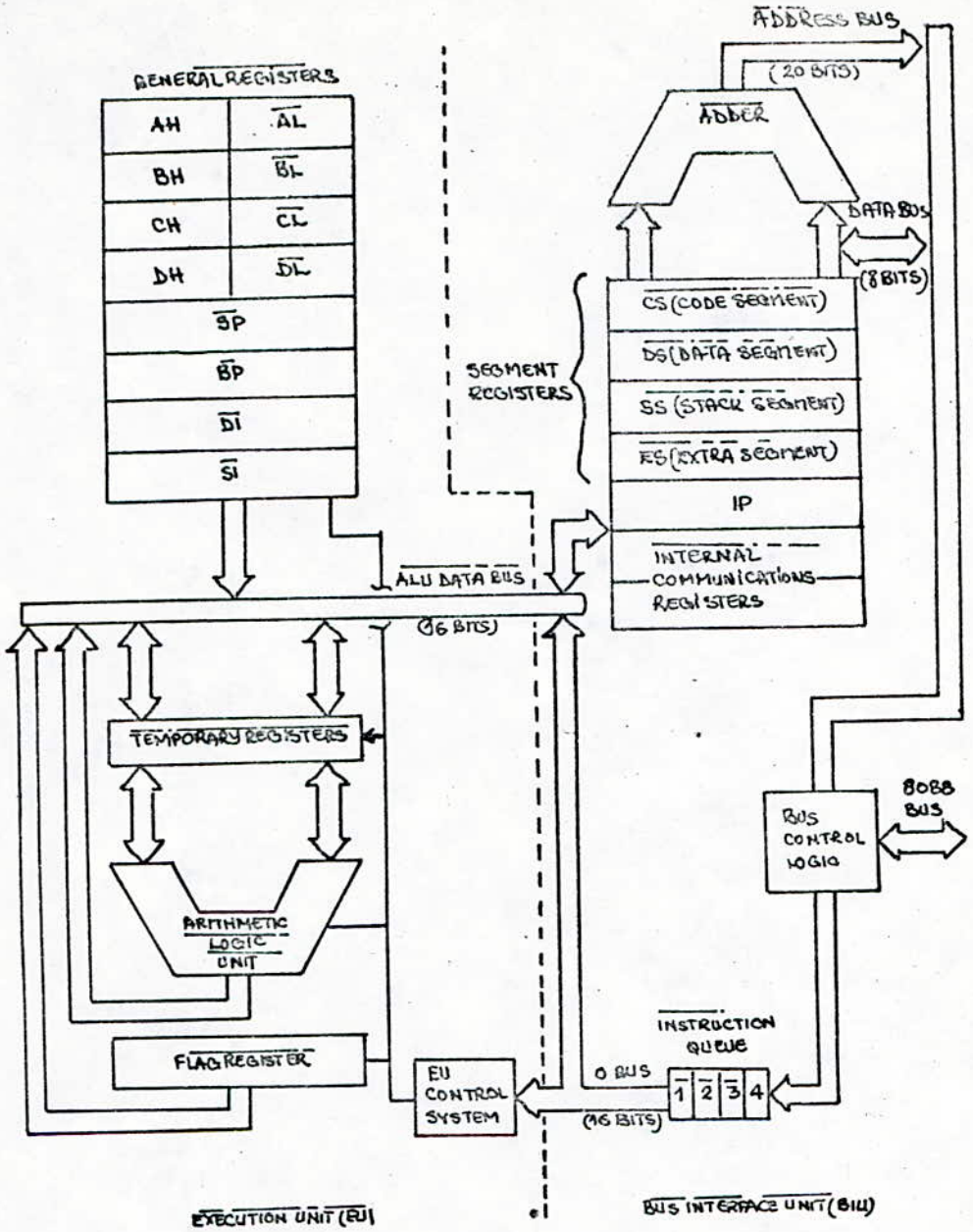

ADDRESS BUS
(20 BITS)
ADDER
DATA BUS
(8 BITS)
GENERAL REGISTERS
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
DI
SI
SEGMENT REGISTERS
CS (CODE SEGMENT)
DS (DATA SEGMENT)
SS (STACK SEGMENT)
ES (EXTRA SEGMENT)
IP
INTERNAL COMMUNICATIONS REGISTERS
ALU DATA BUS
(16 BITS)
TEMPORARY REGISTERS
ARITHMETIC LOGIC UNIT
FLAG REGISTER
EU CONTROL SYSTEM
BUS CONTROL LOGIC
8088 BUS
INSTRUCTION QUEUE
1
2
3
4
(16 BITS)
EXECUTION UNIT (EU)
BUS INTERFACE UNIT (BIU)

FIG - 1.1.6.4

1.1.7-LES INTERRUPTIONS (en mode minimum).

Le processeur supporte 256 types d'interruptions numérotés de 0 à 255.A chaque type correspond un pointeur de 32 bits don- nant l'adresse de la procédure à exécuter.

1.1.7.1-Vectorisation:

Les interruptions sont classées suivant leur nature:interne ou externe.Internes,elles sont provoquées par certains états du processeur ou par instructions spéciales et sont vectorisées automatiquement,c'est à dire que le type d'interruption est figé par l'architecture du processeur. Externes,elles sont provoquées par un périphérique qui doit fournir au processeur un type d'interruption(8 bits).Un boîtier d'encodage de priorité est habituellement employé pour faciliter la génération de ce type (fig-1.1.7.1).

Le type d'interruption sur 8 bits est lu(cas externe) ou fabriqué(cas interne),il est ensuite multiplié par 4,il indique ainsi un délacement dans une table de 1024 octets placés à l'adresse physique 00000H et composé de 256 pointeurs.Un pointeur adresse donc une procédure de traitement située dans l'espace adressable de 1 MO du processeur.(fig-1.1.7.1)

*Déroulement d'une interruption dirigée(cas général)

1)Un périphérique envoit un signal de demande d'interruption, à la fois au microprocesseur et au contrôleur de priorité.

2)A la réception de la demande d'interruption,le microprocesseur débute une procédure d'interruption classique:génère sur son bus d'adresse une adresse N.Mais,dans ce cas,l'adresse N n'est pas celle du programme de gestion de l'E/S.

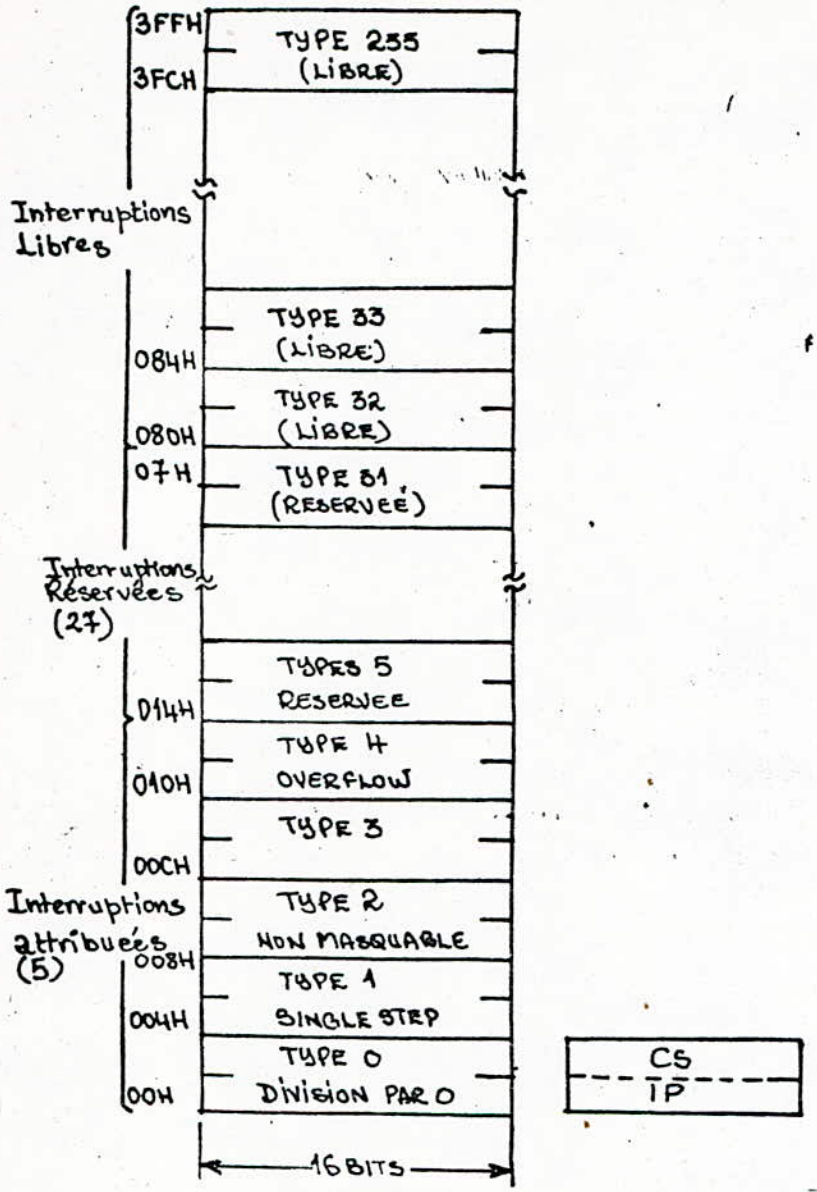
3)Cette adresse sert à valider le codeur de priorité du contrôleur de priorité.Ce codeur,en fonction de la priorité du périphérique demandeur,génère un adresse partielle d'interruption.

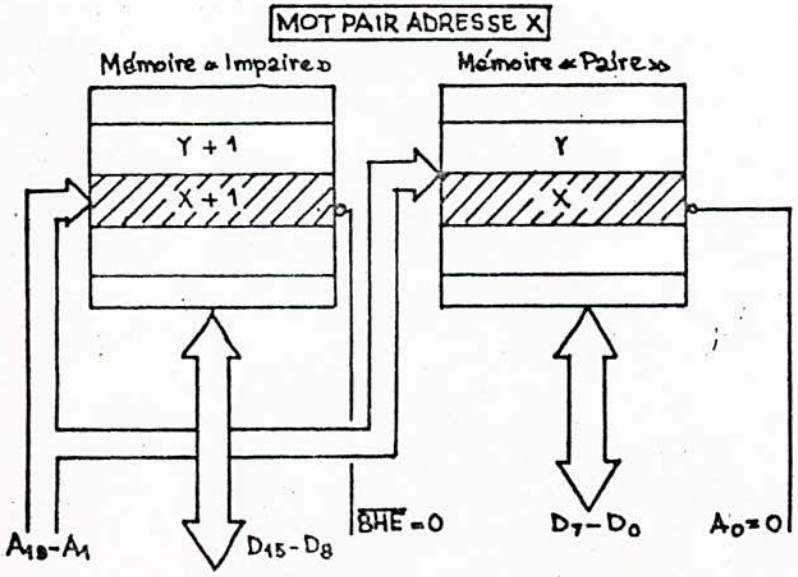
4)Un décodeur d'adresses combine cette adresse partielle à une autre adresse partielle provenant directement du bus d'adresse du microprocesseur.Le microprocesseur génère alors l'adresse complète du vecteur d'interruption.Le microprocesseur charge ce vecteur,via le bus de données,dans son compteur de programme,ce qui provoque le branchement à l'adresse réelle du programme d'interruption cherché.

1.1.7.2-Mécanisme des interruptions externes:

a)Interruptions masquables(INTR):Les interruptions externes sont signalées par la broche INTR.Le processeur termine l'instruction en cours.Il entre ensuite dans 2 cycles de validation INTA(Interrupt Acknowledge).Pendant ces 2 cycles le processeur lit sur son bus le type d'interruption.Le CPU sauve le mot d'état en pile,et utilise le type comme vecteur

* ZONE MEMOIRE RESERVEE POUR LES INTERRUPTIONS DE 8086/8088 *





* LECTURE / ECRITURE D'UN MOT (16 BITS) SELON SON ADRESSE.

d'indirection,il remplace le CS et l'IP courant par ceux trouvés à l'adresse pointée par le type.Les interruptions (bits IF et TF du mot d'état) sont interdites.La procédure d'interruption ainsi activée doit se terminer par une instruction spécifique IRET qui dépile l'adresse de retour et restaure le mot d'état en entier.La sauvegarde du reste du contexte(registres) est effectuée par le logiciel de la procédure d'interruption.

b)Les interruptions non masquables:Le processeur dispose d'une broche NMI qui est une entée d'interruption sensible au front montant du signal NMI.L'activation de cette entée provoque un déroutement de type 2.Cette entrée est non masquable.

1.1.8-DIFFERENCES ENTRE LE 8086 ET LE 8088.

Le 8088 est un 16 bits ou plutôt un pseudo 16 bits dans le sens ou il possède une archutecture entièrement identique à celle du 8086 du point de vue du programmeur.Il possède les mmes capacitées d'adressage,d'E/S,de segmentation.Les logiciels développés pour l'une des machines s'exécutent totalement sur l'autre.La différence majeure réside dans l'accès mémoire.Le 8088 accéde uniquement à des octets.Là ou le 8086 n'avait besion que d'un cycle machine pour acquérir un mot de 16 bits,le 8088 fait donc deux accés séquentiels.Du point de vue externe le bus de données n'a que 8 bits,d'ou l'économie d'un boîtier 8286 (buffer de données),la broche BHE n'existe plus,la broche M/IO est changée en M/IO.

1.2-LE LANGAGE D'ASSEMBLAGE DU 8086/8088.

1.2.1-Généralités:

Au niveau le plus fondamental,un microprocesseur réagit à une liste d'opérations appellées "programme machine".Ce programme est écrit en code binaire, et chaque instruction est représentée par un code binaire qui lui est spécifique.La manipulation fastidieuse du code binaire étant source d'erreur,on a été conduit à la création de l'assembleur puis de langages évolués.
Le langage assembleur est un langage de niveau immédiatement supérieur au langage machine,il permet au programmeur de représenter en code machine une instruction par une abréviation symbolique,une telle abréviation est dite mnémonique d'instruction.
L'assembleur(ou le programme assembleur) permet la traduction du programme source en un programme machine (seul langage compris par le microprocesseur) qui est dit programme exécutable.

Une instruction de programme écrite en assembleur est divisée en 4 champs:l'étiquette,le mnémonique,l'opérande et le commentaire.

L'étiquette est optionnelle.

Le champ mnémonique,contient la mnémonique précis donné par le constructeur,il indique au programme assembleur quelle est l'opération à effectuer.

Le champ opérande comporte des informations sur les registres,les données ou les adresses associées à l'opération.

Outre sa fonction de traduire les mnémoniques et opérendes en langage machine,l'assembleur assigne des cases mémoires consécutives à chaque code opération et opérande.

1.2.2-Les modes d'adressage du 8088:

Les opérations se font entre regisres,registres et mémoire mais jamais entre mémoires sauf pour les chaînes de caractères. Les différents modes d'adressage sont:

a)Adressaage registre:

C'est le plus simple, les opérations se font entre registres sans cependant mêler les valeurs 16 et 8 bits.

```
Exemple:          ORG 100H
-------           MOV AX,10        ;immédiat
                  MOV BX,AX        ;adressage registres
APRES EXECUTION: AX---->000A
                 BX---->000A
```

b)Adressage immédiat:

Traite des données directement définies dans l'instruction.

Ce mode est utile pour manier des constantes de calcul ou d'adresse (voir exemple ci-dessus).

c)Adressage direct:

Le déplacement de la donnée à laquelle on veut accéder est spécifiée directement en donnant l'identificateur.Il est nécessaire que le type de la donnée soit accordé avec le registre utilisé.

```
EXEMPLE:          MOV  AX,TAUX          ;La donnée TAUX est sur
                                        ;16 bits.
```

d)Adressage basé:

Cet adressage utilise des registres dédiés pour contenir le déplacement.L'adressage basé suppose la base dans BX associé au segment DS ou BP associé à SS.La notation se fait par une paire de crochets et indique le contenu de la mémoire.

```
EXEMPLE:          ADD AX,(SI)
```

f)Adressage basé et indexé:

On utilise dans ce cas 2 registres,un registre de base BP ou BX et un registre d'index SI ou DI,il y a donc quatres configuration possibles.La notation est toujours celle des doubles crochets.Le déplacement est calculé par ajout des contenus du registre de base est du registre d'index modulo 16.Le registre de base impose le registre de segment: BX avec DS ou BP avec SS.Cette notation doit être spécifiée par l'opérateur PTR.

EXEMPLE: AND (BX)(SI),12H

1.2.3-Quelques instructions de base du 8086/8088:

La liste complète des instructions sera donnée en annexe.

1.2.3.1-Transfert de données:

Les transferts de données sont effectués par l'instruction "MOV" qui charge la source dans la destination et par "XCHG" qui effectue un transfert croisé.

SYNTAXE: MOV destination,source

```
EXEMPLE:  MOV  CL,AL      ;CL:=AL
          XCHG BX,CX      ;BX:=CX ET CX:=BX
          MOV  AX,12H     ;AX:=12H
```

1.2.3.2-Comparaisons:

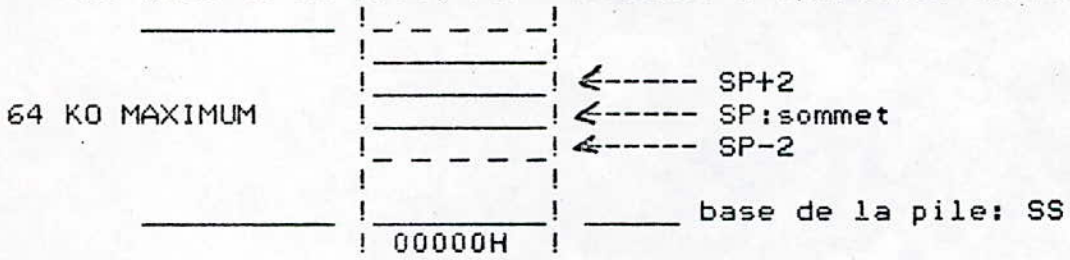
L'instruction CMP compare 2 opérandes (registres, mémoire et registre,ou mémoire/registre et immédiat) et met les flags à jour.Les comparaisons se font par soustraction sans modification des opérandes et l'exploitation du résultat se fait en signé ou non.

SYNTAXE: CMP OP1,OP2

```
EXEMPLE:  MOV AL,12H
          MOV BL,06H
          CMP AL,BL
```

1.2.3.3-Gestion de la pile:

C'est une pile organisée en mots.Le pointeur de pile SP, décroit de deux uniés par empilage et croit d'autant à chaque dépilage (voir figure ci-dessous).L'adressage de la pile se fait explicitement par BP qui contient l'offset dans la pile et peut être employé dans la plupart des instructions.L'instruction "PUSH" empile des registres ou des mots de 16 bits,"POP" effectue l'opération inverse.



*** PILE DU 8086/8088 ***

1.2.3.4-Boucles de programmes:

Les boucles de programme sont efféctuées grâce à l'instruction "LOOP" qui fonctionne avec le registre CX.

L'instruction "LOOP" a deux actions:

a-Décrémenter le registre CX (obligatoirement).

b-Sauter sur l'étiquette spécifiée si CX est non nul.

LOOP	Boucle si CX=0
LOOPNE	Boucle si CX=0 et ZF=0
LOOPE	Boucle si CX=0 et ZF=1

1.2.3.5-Sauts conditionnels:

Les sauts conditionnels sont relatifs relatifs à un segment et ne peuvent aller plus loin que 127 octets ni reculer de plus de 128 octets.

*Sauts arithmétiques:

Ils sont employés aprés les instructions de comparaison et permettent de prendre une décision sur les valeurs relatives des 2 opérendes.C'est l'instruction de saut qui indique le choix.

Sémantique	opérande signés	Opérendes non signés
=	JEQ	JEQ
>	JL	JA
inf	JL	JB
>=	JGE	JAE
inf ou =	JLE	JBE
=/=0	JNE	JNE

1.2.3.6-Sauts inconditonnels intra-segments:

1.2.3.6.1-Sauts proches:

Identiques aux sauts conditionnels,ils effectuent un saut relatif vis-à-vis de l'IP(compteur ordinal),saut dont la taille ne dépasse pas +/-32768 octets.

SYNTAXE: JMP étiquette

1.2.3.6.2-Saut indirect:

Ici,la nouvelle valeur du contenu de IP est le contenu d'une case mémoire ou d'un registre.

```
EXEMPLE:  JMP  AX          ;(IP)=(AX)
          JMP  TABLE(BX)
```

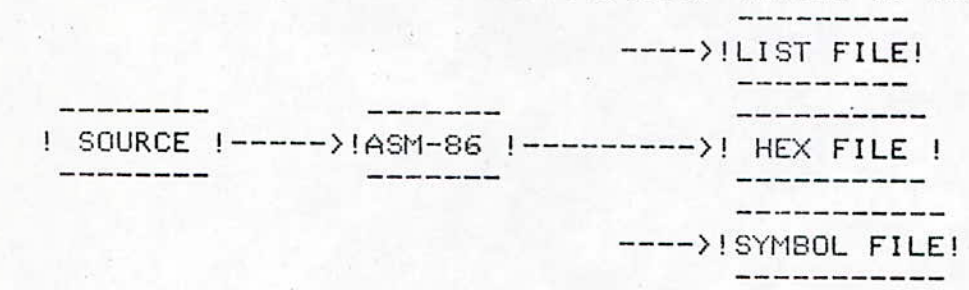
1.3-L'ASM86.

1.3.1-L'opération d'assemblage:

ASM-86 procéde à l'assemblage d'un fichier source en langage assembleur 8086 en 3 phases et fournit 3 fichiers en incluant un fichier en langage machine en format hexadécimal de Digital Research ou en format Intel.

ASM-86 existe sous deux formes:le cross assembleur

8086 destiné à être exécuté sous CP/M avec système basé sur l'INTEL 8088 ou le ZILOG Z80,et l'assembleur 8086 destiné à tourner sous CP/M-86 dans un système basé sur l'INTEL 8086 ou 8088.
ASM-86 produit typiquement 3 fichiers de sortie à partir d'un fichier (source en entrée).Voir schéma ci-dessous.



(filename).A86-contient le source
(filename).LST-contient un listing
(filename).H86-contient le programme assembleur en format héxadécimal.
(filename).SYM-contient tout les symboles définits par l'utilisateur.

fig-1.3.1-Fichiers source et fichiers objets de l'ASM-86

La fig-1.3.1 montre aussi les extensions données aux fichiers crées par l'ASM-86.

ASM-86 accepte un fichier source avec 3 lettres quelconques comme extension(ou type),mais si l'extension est omise en invoquant la commande d'assemblage,l'ASM-86 suppose que le fichier source posséde l'extension .A86.Si le fichier posséde une autre extension que ".A86" ou n'a pas du tout d'extension,ASM-86 retourne un massage d'erreur.Les autres types listés fig-1.3.1 identifient les 3 fichiers objets.

Le fichier".LST"contient le listing en langage assembleur avec les messages d'erreur.

Le ".H86"contient le programme en langage machine soit dans le format HEX de DIGITAL RESEARCH soit dans celui d'INTEL.

Le ".SYM" liste les symboles définis par l'utilisateur.

L'ASM-86 est invoqué en entrant une commande de la forme suivante:

ASM86 (fichier source)

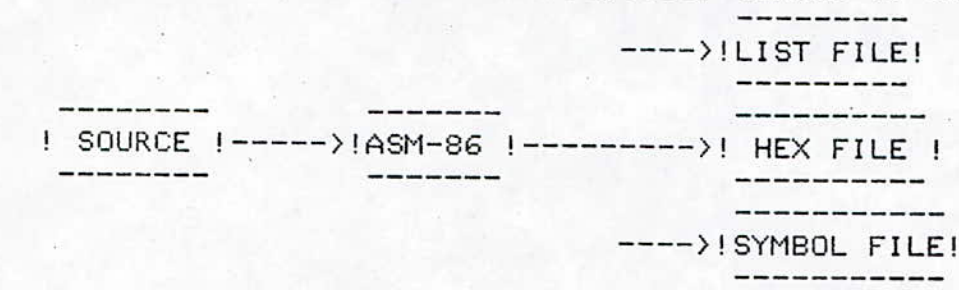
Il faut spécifier le nom du fichier source de la manière suivante:

(unité de disque:)(nom du fichier)(.extension optionnelle)

(unité de disque:):Représente une lettre spécifiant l'unité ou se trouve le fichier.N'est pas nécessaire si le fichier se trouve sur l'unité courante.

(nom du fichier):Composé de 1 à 8 caractères(crée sous CP/M)

8086 destiné à être exécuté sous CP/M avec système basé sur l'INTEL 8088 ou le ZILOG Z80,et l'assembleur 8086 destiné à tourner sous CP/M-86 dans un système basé sur l'INTEL 8086 ou 8088.
ASM-86 produit typiquement 3 fichiers de sortie à partir d'un fichier (source en entrée).Voir schéma ci-dessous.



(filename).A86-contient le source
(filename).LST-contient un listing
(filename).H86-contient le programme assembleur en format héxadécimal.
(filename).SYM-contient tout les symboles définits par l'utilisateur.

fig-1.3.1-Fichiers source et fichiers objets de l'ASM-86

La fig-1.3.1 montre aussi les extensions données aux fichiers crées par l'ASM-86.

ASM-86 accepte un fichier source avec 3 lettres quelconques comme extension(ou type),mais si l'extension est omise en invoquant la commande d'assemblage,l'ASM-86 suppose que le fichier source posséde l'extension .A86.Si le fichier posséde une autre extension que ".A86" ou n'a pas du tout d'extension,ASM-86 retourne un massage d'erreur.Les autres types listés fig-1.3.1 identifient les 3 fichiers objets.

Le fichier".LST"contient le listing en langage assembleur avec les messages d'erreur.

Le ".H86"contient le programme en langage machine soit dans le format HEX de DIGITAL RESEARCH soit dans celui d'INTEL.

Le ".SYM" liste les symboles définis par l'utilisateur.

L'ASM-86 est invoqué en entrant une commande de la forme suivante:

ASM86 (fichier source)

Il faut spécifier le nom du fichier source de la manière suivante:

(unité de disque:)(nom du fichier)(.extension optionnelle)

(unité de disque:):Représente une lettre spécifiant l'unité ou se trouve le fichier.N'est pas nécessaire si le fichier se trouve sur l'unité courante.

(nom du fichier):Composé de 1 à 8 caractères(crée sous CP/M)

(extension optionnelle):comporte de 1 à 3 caractères, d'habitude ".A86".

Voici quelques exemples de commandes de l'ASM-86:
Une fois appelé,l'ASM-86 retourne le message:

CP/M 8086 ASSEMBLER VER 2.0

L'ASM-86 ouvre le fichier source et l'assemble,s'il n'est pas trouvé ou n'a pas l'extension correcte,le message d'erreur suivant est affiché:

NO FILE: nom de fichier

Aprés ouverture du fichier,l'assembleur crée les différents fichiers objets cités.Durant l'assemblage,ASM-86 s'arrête si une condition d'erreur est réalisée par exemple:disque plein.Quand ASM-86 détecte une erreur dans le fichier source, il place une ligne représentant le message d'erreur dans le listing du fichier en face de la ligne contenant l'erreur.Chaque message d'erreur a un numéro et donne une brève explication de l'erreur.Quand l'assemblage se termine,ASM-86 liste le message suivant:

END OF ASSEMBLY . NUMBER OF ERRORS:n USE FACTOR:x%

1.3.1-Principales directives de l'ASM-86:

Ce sont en général,des commandes fournissants des indications au programme assembleur.On rencontre les commandes suivantès:

*Affectation d'étiquettes à des valeurs spécifiques durant l'assemblage.
*Définition de l'espace de stockage des données.
*Spécification de l'adresse de début et de fin du programme.
*Définition de la longueur d'un mot.

Chaque directive est dénotée par une pseudo-instruction.

Les principales pseudo-instructions de l'ASM-86 sont:

*ORG:La déclaration ORG prend la forme :

étiquette ORG expression

L'étiquette est facultative et permet d'identifier le programme.L'expression est un nombre de 16 bits indiquant la position mémoire à partir de laquelle l'assembleur générera le code machine.On notera que la plupart des programmes écrits en ASM-86 débutent par ORG 100H(base de la zone de programme du CP/M).Si l'étiquette est spécifiée dans l'expression ORG,elle prendrait la valeur de l'expression, auquel cas elle pourrait être utilisée dans le champ d'opérande d'une autre expression pour représenter l'expression qui lui a été affectée.

* END: Elle est facultative dans un programme écrit en assembleur,mais si elle existe elle doit être la dernière expression écrite (toutes les expressions suivants END seront ignorées).Les 2 formes de END sont:

```
étiquette END
étiquette END expression
```

L'étiquette est encore facultative.Dans la première forme, l'adresse de début du programme est prise comme 0000.Autrement, l'expression est évaluée et constitue alors l'adresse de début du programme qui prend fin.Ainsi la majorité des programmes en assembleur se teminent par l'expression:

```
END 100H
```

* EQU:Elle est utilisée pour donner des synonymes à des valeurs numériques particulières.Sa forme générale est:

```
étiquette EQU expression
```

Dans ce cas,l'étiquette est obligatoire et ne peut être,dans le programme,celle d'une autre expression. L'assembleur évalue l'expression et assigne cette valeur à l'identificateur donné dans le champ étiquette.
L'identificateur est généralement un nom qui décrit une valeur permettant ainsi d'orienter l'utilisateur.
De plus,la valeur affectée à ce nom sera maintenue tout le long du programme.

* CSEG:

```
CSEG (expression numérique)
CSEG
CSEG$
```

Cette directive indique que les instructions qui suivent appartiennent au segment de code.Toutes les directives sont légales dans le segment de code.On utilise la première forme lorsque la position du segment est connue lors de l'assemblage,dans ce cas le code généré n'est pas relogeable.La deuxième forme est utilisée lorsqu'on ignore la position du segment,le code généré est relogeable.On utilise la troisième forme pour continuer le segment de code lorsqu'il a été arrêté par un DSEG,SSEG ou un ESEG.

* DSEG:

```
DSEG (exp-num)
DSEG
DSEG$
```

Cette directive indique que les expressions qui suivent appartiennent au segment de données.
Explications similaires à celles de CSEG,mais il est très important de noter que les instructions ne sont pas

admises dans ce segment.

* DB(DW):La directive DB(DW) permet au programmeur de définir les espaces de stockage initialisés en précisant le format Byte (ou Word).

étiquette DB e1,e2,...,eN

e1,...,eN sont des expressions qui repésentent des valeurs 8 bits (16 bits) ou des chaînes de caractères ASCII ayant une longueur maximale de 64 caractères.

* RB(RW):La directive RB(RW) alloue des bytes en mémoire sans les initialiser.

étiquette RB (exp-num)

Exp-num donne le nombre de bytes à réserver ainsi que l'attribut byte à l'étiquette.

1.4: LE MS-GWBASIC DE L'OLIVETTI M-24.

Le MS-GWBASIC,ou langage GWBASIC sous le système d'exploitation MS-DOS permet la création et l'accés à deux types de fichiers de données sur disque:

*Les fichiers séquentiels.
*Les fichiers à accés direct.

Un fichier est ouvert avec l'instruction "OPEN",le port de communication est ouvert de la même manière avec l'instruction "OPEN COM".Cette instruction affecte une mémoire tampon pour l'entrée et la sortie des données.

1.4.1-LES FICHIERS SEQUENTIELS:

Les fichiers séquentiels sont plus faciles à créer que les fichiers à accés direct,mais limitent la souplesse et la vitesse de l'accés aux données.Les données écrites dans un fichier séquentiel sont sous la forme de caractères ASCII qui sont chargés et mémorisés,les uns après les autres (séquentiellement), dans l'ordre selon lequel ils sont envoyés.

1.4.1.1-Création d'un fichier séquentiel:

Les séquences du programme suivant sont nécessaires pour créer un fichier séquentiel et pour accéder aux données se trouvant dans le fichier:

1-Ouvrir le fichier DONNEES(nom du fichier) en mode "O"

```
OPEN "O",#1,"DONNEES"
```

Ou "1" est le numéro de fichier affecté au fichier "DONNEES".

2-Ecrire les données dans le fichier en utilisant l'instruction: WRITE#(ou PRINT#)

```
WRITE#1,A$
```

Ou A$ est la chaine de caractères à écrire sur le fichier "1"(DONNEES).

3-Pour accéder aux données se trouvant dans un fichier, on doit fermer (CLOSE) le fichier et le réouvrir (OPEN) en mode "I":

```
CLOSE#1
OPEN "I",#1,"DONNEES"
```

4-Utiliser l'instruction INPUT# pour lire les données à partir du fichier séquentiel vers le programme:

```
INPUT#1,X$
```

Ou X$ est la donnée à lire à partir du fichier "1"

(DONNEES).

La fonction LOC,quand elle est utilisée avec un fichier séquentiel,donne le nombre de secteurs qui ont été écrits dans le fichier ou lus à partir de celui-ci.Par exemple:

```
10 IF LOC(1)>50 THEN STOP
```

termine l'exécution d'un programme si plus de 50 caractères ont été écrits dans le fichier (1) ou lus à partir de celui-ci depuis son ouverture.

Remarque:Dés qu'un fichier séquentiel est ouvert sur le disque en mode "O",son contenu en cours est détruit.Pour ajouter des données supplémentaires au fichier,il est nécessaire d'utiliser l'instruction OPEN avec le mode APPEND.

1.4.2-Les fichiers à accés direct:

La création et l'accés aux fichiers à accés direct nécessitent d'avantage d'instructions de programme que les fichiers séquentiels,mais leurs utilisation offre des avantages.Par exemple,ils nécessitent moins d'espace sur disque,parceque GW-BASIC les mémorise dans un format binaire condensé(un fichier séquentiel est mémorisé comme une série de caractères ASCII).Le plus grand avantage des fichiers à accés direct est que l'utilisateur peut accéder aux données de manière sélective,c.à.d à n'importe quel endroit du disque.Par conséquent,il n'est pas nécessaire de lire toutes les informations qui se trouvent sur le disque contrairement aux fichiers séquentiels.Ceci est possible parceque les informations sont mémorisées et accéssibles en unités distinctes appelées "enregistrement" et parceque chaque enregistrement est numéroté.

1.4.2.1-Création d'un fichier à accés direct:

La création d'un fichier à accés direct nécessite les instructions suivantes:

1-Ouvrir le fichier pour l'accés direct en mode "R".Cet exemple spécifie une longueur d'enregistrement de 32 octets.Si la longueur d'enregistrement est omise,la valeur par défaut est de 128 octets.

```
OPEN "R",#1,"FICHIER",32
```

2-Utiliser l'instruction FIELD pour affecter de l'espace dans la mémoire tampon à accés direct pour les variables qui seront écrites dans le fichier à accés direct:

```
FIELD#1,20 AS N$,4 AS A$,8 AS P$
```

3-Utiliser la commande LSET pour placer les données dans le mémoire tampon.Les valeurs numériques doivent être transformées en chaines quand elles sont placées dans la

mémoire tampon.Pour cela utilisez les fonctions "make":MKI$ transforme la valeur double précision en chaine.

```
LSET N$=X$
LSET A$=MKS$(AMT)
P$=TEL$
```

MKS$ transforme une valeur simple précision en chaine.

4-Ecrire les données à partir de la mémoire tampon dans le disque à l'aide de l'instruction PUT:

```
PUT#1,CODE%
```

La fonction LOC,avec les fichiers à accés direct donne le "numéro" d'enregistrement en cours.Le numéro d'enregistrement en cours est égal au numéro du dernier enregistrement utilisé dans une instruction GET ou PUT augmenté de 1.Par exemple,l'instruction:

```
10 LOC(1)>50 THEN END
```

termine l'exécution du programme si le numéro d'enregistrement en cours dans le fichier #1 est supérieur à 50.

1.4.2.2-Comment accéder à un fichier à accés direct:

La lecture d'un fichier à accés direct nécessite les instructions suivantes:

1-Ouvrir le fichier en mode "R":

```
OPEN "R",#1,"FICHIER",32
```

2-Utiliser l'instruction FIELD pour affecter de l'espace dans le mémoire tampon pour les variables qui seront lues à partir du fichier:

```
FIELD #1,20 AS N$,4 AS  A$,8 AS P$
```

3-Utiliser l'instruction GET pour transférer l'enregistrement désiré dans la mémoire tampon:

```
GET #1,CODE %
```

4-Le programme peut maintenant accéder aux données qui se trouvent dans la mémoire tampon.Les valeurs numériques doivent être converties à nouveau en nombre à l'aide des fonctions "de conversion".CVI convertit les chaines numériques en nombres simple précision,et CVD les chaines numériques en nombres double précision:

```
PRINT N$
PRINT CV$(A$)
```

1.4.2.3-Instructions utilisées avec les fichiers séquentiels:

CLOSE:Cette instruction termine l'E/S d'un fichier ou d'une unité.
Syntaxe: CLOSE ((#)FILENUM(,(#)FILENUM)...)
Ou filenum est le numéro de fichier avec lequel celui-ci a été ouvert.

INPUT#:Cette instruction lit des données à partir d'un fichier séquentiel sur disque et les affecte aux variables du programme.
Syntaxe: INPUT#FILENUM,VARIABLE(,VARIABLE)...

Ou "filenum" est le numéro utilisé quand le fichiera été ouvert.
"Variable" est une variable chaine ou numérique qui recevra une donnée du fichier.

INPUT$:Elle envoie une chaine de caractères lus à partir du clavier ou à partir d'un fichier.

Syntaxe: INPUT$(length (,(#)filenum))

Ou "length" spécifie le nombre de caractères qui seront lus à partir du clavier ou du fichier.

"filenum" est le numéro du fichier utilisé dans l'instruction OPEN.Si filenum est omis,les caractères seront lus à partir du clavier.

LINE INPUT#:Elle lit toutes une ligne à partir d'un fichier séquentiel vers une chaine de variables.
Syntaxe: LINE INPUT#filenum,stringvar

Ou "filenum" est le numéro avec lequel le fichier a été ouvert.
"Stringvar" est la variable chaine à laquelle la ligne sera affectée.

LOC:Cette fonction renvoie la position courante dans le fichier.
Avec les fichiers à accés direct sur disque,LOC renvoie le numéro de l'enregistrement qui vient d'être lu ou écrit à partir de l'instruction GET ou PUT.Avec les fichiers séquentiels,LOC renvoie la position de l'octet courant dans le fichier divisée par 128.
Pour les fichiers de communication,LOC est utilisé pour déterminer s'il y a des caractères dans le mémoire tampon de saisie,qui attendent d'être lus.
Syntaxe: LOC (filenum)

Exemple: 10 IF LOC(2)>50 THEN STOP

LOF:Cette fonction renvoie la longueur du fichier nommé, en octets.
Syntaxe: LOF (filenum)

OPEN:Cette instruction permet les opérations d'E/S dans un fichier ou dans un périphérique.

Syntaxe 1: OPEN (device/filespec) (FOR mode1) as (#) filenum (LEN=record-length)

Syntaxe 2: OPEN mode 2,(#/filenum,filespec(,record-length))

Filespec:spécifie le fichier à ouvrir.Cette chaine peut inclure facultativement un périphérique.

mode1:détermine la position initiale du pointeur de fichier et l'action à entreprendre si le fichier n'existe pas.Les modes et actions sont les suivants:

INPUT:spécifie le mode d'entrée séquentiel.Positionne le pointeur au début d'un fichier existant.Si le fichier n'existe pas,une erreur "FILE NOT FOUND" est sortie.

OUTPUT:spécifie le mode de sortie séquentiel.Positionne le pointeur au début du fichier.Si le fichier n'existe pas,il est créé.
Si la clause "FOR mode1" est omise,la position initiale est au début du fichier.Si le fichier n'est pas trouvé,un fichier est créé.C'est le mode d'accés direct;c-à-d que les enregistrements peuvent être lus ou écrits à n'importe quelle position à l'intérieur du fichier.

filenum:est le numéro (1 à 255) utilisé pour associer une mémoire tampon d'E/S à un fichier sur disquer ou à un dispositif.

record-length:Spécifie la longueur d'enregistrement à utiliser pour les fichiers directs (valeur comprise entre 2 et 32767).

mode2: O:spécifie le mode de sortie séquentiel.
I:spécifie le mode d'entrée séquentiel.
R:spécifie le mode d'E/S à accés direct.
A:spécifie le mode de sortie séquentiel et place le pointeur de fichier à la fin de celui-ci.Un PRINT# ou WRITE# étend le fichier.

Exemple: pour ouvrir l'imprimante,ou l'écran en sortie,on peut utiliser:

```
10 OPEN "LPT1:" FOR OUTPUT AS#1
20 OPEN "SCRN:" FOR OUTPUT AS#2
```

OPEN COM:Cette instruction ouvre un fichier de communication.
Syntaxe:

OPEN"COMn:(speed),(parity)(,(data)(,stop)(,RS)(,CS)(,DS)(,CD)(,BIN)(,ASC)(,LF)))" (FOR mode) AS(#) filenum (LEN=record-length)

Ou:

n:spécifie le numéro du port de communication asynchrone.

speed:spécifie le débit de transmission/réception. Les vitesses possibles sont:
75,110,150,300,600,1200,1800,2400,4800 et 9600 bauds.
La vitesse par défaut est de 300 bauds.

parity:désigne la parité du périphérique à ouvrir:
-E:paire(valeur par défaut).
-O:Impaire
-N:Aucune

data:désigne le nombre de données en bits par octet.Les entrées autorisées sont:5,6,7(par défaut),ou 8.

stop:désigne le bit d'arrêt.Les entrées autorisées sont 1 ou 2.En cas d'omission 75 et 110 Bps,transmettent 2 bits d'arrêt,tous les autres un bit d'arrêt.

*RS:Supprime RTS(demande pour envoyer)
*CS:contrôle CTS (Effacer pour envoyer).
*DS:contrôle DSR (Données affichées prêtes).
*CD:contrôle CD (Détecteur de la porteuse)
*BIN:ouvre le fichier en mode binaire.BIN est choisi

par défaut à moins que ASC ne soit spécifié.
*ASC:ouvre le fichier en mode ASCII.
*LF:spécifie qu'un interligne doit être envoyé après un retour à la ligne.

PRINT# et PRINT#USING:Ces instructions écrivent séquentiellement les données dans un fichier sur disque.
Syntaxe: PRINT#filenum, (USING format-string) list of expressions

filenum:est le numéro utilisé quand le fichier a été ouvert.

format-string:comprend des expressions en chaines ou des expressions numériques qui sont à imprimer.

list-of-expressions:liste des expressions num et/ou chaines à écrire dans le fichier.

WRITE#:Instruction qui permet d'écrire les données dans un fichier séquentiel.
Syntaxe:WRITE#filenum,list-of-expressions

filenum:numéro sous lequel le fichier a été ouvert en mode

OPEN"COMn:(speed),(parity)(,(data)(,stop)(,RS)(,CS)(,DS)(,CD)(,BIN)(,ASC)(,LF)))" (FOR mode) AS(#) filenum (LEN=record-length)

Ou:

n:spécifie le numéro du port de communication asynchrone.

speed:spécifie le débit de transmission/réception. Les vitesses possibles sont:
75,110,150,300,600,1200,1800,2400,4800 et 9600 bauds.
La vitesse par défaut est de 300 bauds.

parity:désigne la parité du périphérique à ouvrir:
-E:paire(valeur par défaut).
-O:Impaire
-N:Aucune

data:désigne le nombre de données en bits par octet.Les entrées autorisées sont:5,6,7(par défaut),ou 8.

stop:désigne le bit d'arrêt.Les entrées autorisées sont 1 ou 2.En cas d'omission 75 et 110 Bps,transmettent 2 bits d'arrêt,tous les autres un bit d'arrêt.

*RS:Supprime RTS(demande pour envoyer)
*CS:contrôle CTS (Effacer pour envoyer).
*DS:contrôle DSR (Données affichées prêtes).
*CD:contrôle CD (Détecteur de la porteuse)
*BIN:ouvre le fichier en mode binaire.BIN est choisi

par défaut à moins que ASC ne soit spécifié.
*ASC:ouvre le fichier en mode ASCII.
*LF:spécifie qu'un interligne doit être envoyé après un retour à la ligne.

PRINT# et PRINT#USING:Ces instructions écrivent séquentiellement les données dans un fichier sur disque.
Syntaxe: PRINT#filenum, (USING format-string) list of expressions

filenum:est le numéro utilisé quand le fichier a été ouvert.

format-string:comprend des expressions en chaines ou des expressions numériques qui sont à imprimer.

list-of-expressions:liste des expressions num et/ou chaines à écrire dans le fichier.

WRITE#:Instruction qui permet d'écrire les données dans un fichier séquentiel.
Syntaxe:WRITE#filenum,list-of-expressions

filenum:numéro sous lequel le fichier a été ouvert en mode

"O".
list-of-expression:liste d'expressions chaines ou numériques. Elles doivent être séparées par des virgules.

Remarque:La différence entre WRITE# et PRINT# est que WRITE# insère des virgules entre les articles lorsqu'ils sont écrits dans le fichier et délimite les chaines avec guillemets.Une séquence CR LF est insérée aprés que le dernier article de la liste ait été écrit dans le fichier.

Exemple:
```
10 A$="VENDREDI":B$="13"
20 WRITE#1,A$,B$
```
L'image suivante est écrite sur le disque:
```
"VENDREDI","13"
```
Si par contre on a:
```
30 a$="VENDREDI":B$="13"
40 PRINT#1,A$;B$
```
L'image suivante est écrite sur disque:
```
VENDREDI13
```

CHAPITRE 2:LES SYSTEMES D'EXPLOITATION

2.1 - CONCEPTS FONDAMENTAUX DES SYSTEMES D'EXPLOITATION

2.1.1 Introduction:

Pour traiter un problème à l'aide d'un ordinateur,on doit lui fournir des ordres sous forme de programme.Mais, pratiquement,pour pouvoir exécuter un programme,l'ordinateur doit,être capable d'effectuer, seul,un certain nombre d'opérations ;entre autres:

-vérifier que les organes périphériques sont en état de fonctionnement.

-savoir ranger en mémoire le programme qu'on lui confie.

Sans cette capacité initiale, l'ordinateur serait inutilisable. Aussi, le constructeur va, à l'origine, écrire des programmes spécifiques qui permettent à l'ordinateur d'effectuer ces differentes tâches.Ces programmes de service,dont l'objectif est de permettre une utilisation plus commode de la machine, constituent son "SYSTEME D'EXPLOITATION".Le système d'exploita- tion est stocké en mémoire principale (situé à l'intérieur de l'unité centrale de l'ordinateur,elle contient le programme en cours d'exécution,les données à traiter immédiatement et les résultats intermédiaires), soit en mémoire morte (ROM) ,soit partiellement en mémoire vive.

Les programmes du système d'exploitation assurent trois fonctions principales:

1- Gérer les travaux confiés à la machine (introduire les programmes en mémoire et contrôler leur exécution).

2- Assurer les opérations d'entrées/sorties

3- Contrôler la bonne utilisation des organes de la machine (vérifier qu'un organe est disponible et l'affecter au programme qui le demande:par exemple disponibilite de l'unité de disquette).

Un système d'exploitation comprend différents programmes dont les noms évoquent la fonction (par exemple CHARGEUR).Un programme particulier ,le superviseur (ou programme directeur, ou encore ordonnanceur) assure la coordination de l'ensemble.Le moniteur de travaux assure l'enchainement des différents programmes soumis à la machine.

CHAPITRE 2:LES SYSTEMES D'EXPLOITATION

2.1 - CONCEPTS FONDAMENTAUX DES SYSTEMES D'EXPLOITATION

2.1.1 Introduction:

Pour traiter un problème à l'aide d'un ordinateur,on doit lui fournir des ordres sous forme de programme.Mais, pratiquement,pour pouvoir exécuter un programme,l'ordinateur doit,être capable d'effectuer, seul,un certain nombre d'opérations ;entre autres:

-vérifier que les organes périphériques sont en état de fonctionnement.

-savoir ranger en mémoire le programme qu'on lui confie.

Sans cette capacité initiale, l'ordinateur serait inutilisable. Aussi, le constructeur va, à l'origine, écrire des programmes spécifiques qui permettent à l'ordinateur d'effectuer ces differentes tâches.Ces programmes de service,dont l'objectif est de permettre une utilisation plus commode de la machine, constituent son "SYSTEME D'EXPLOITATION".Le système d'exploita- tion est stocké en mémoire principale (situé à l'intérieur de l'unité centrale de l'ordinateur,elle contient le programme en cours d'exécution,les données à traiter immédiatement et les résultats intermédiaires), soit en mémoire morte (ROM) ,soit partiellement en mémoire vive.

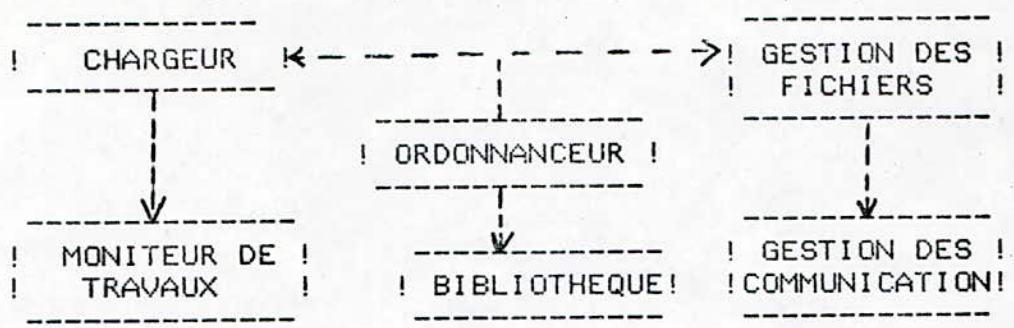
Les programmes du système d'exploitation assurent trois fonctions principales:

1- Gérer les travaux confiés à la machine (introduire les programmes en mémoire et contrôler leur exécution).

2- Assurer les opérations d'entrées/sorties

3- Contrôler la bonne utilisation des organes de la machine (vérifier qu'un organe est disponible et l'affecter au programme qui le demande:par exemple disponibilite de l'unité de disquette).

Un système d'exploitation comprend différents programmes dont les noms évoquent la fonction (par exemple CHARGEUR).Un programme particulier ,le superviseur (ou programme directeur, ou encore ordonnanceur) assure la coordination de l'ensemble.Le moniteur de travaux assure l'enchainement des différents programmes soumis à la machine.

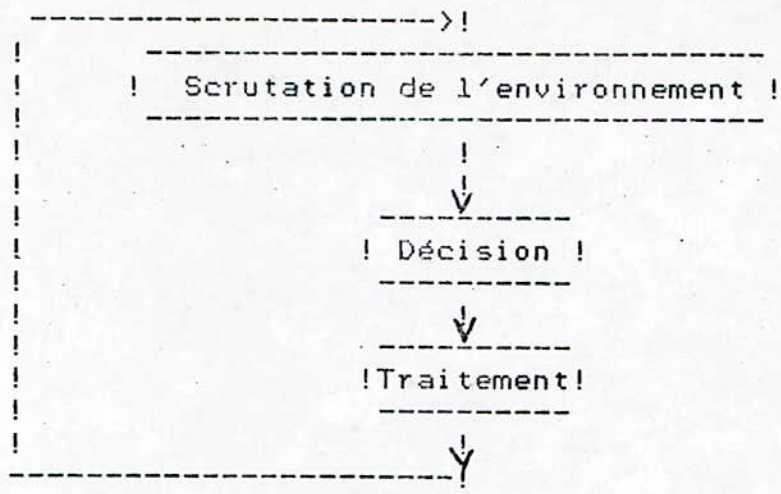


2.1.2-Utilisation des systèmes d'exploitation:

Un système d'exploitation peut être employé pour plusieurs types d'applications: le premier type est orienté vers les processus temps-réels,dans ce cas le système d'exploitation est appelé moniteur temps-réels (MTR) et on utilise les fonctions d'allocation du processeur pour plusieurs tâches simultanées.
Dans un second type, le système d'exploitation sert à faire fonctionner une machine en exploitation par lots.On poura parler de machines en monoprogrammation (un seul programme à la fois),en multiprogrammation ou plusieurs programmes se déroulent en même temps en optimisant la gestion des entrées/sorties.
Un dernier cas est l'utilisation d'un système d'exploitation pour gérer une machine temps-partagé.Dans ce cas,la machine alloue des tranches de temps à plusieurs utilisateurs.Chacun des utilisateurs a l'impression de disposer de la machine entièrement.Le système d'exploitation VMS est un exemple d'un tel système en temps partagé.

2.1.3-Evolution historique des systèmes d'exploitation:

Le premier emploi des microprocesseurs a été un simple remplacement de logique cablée et la machine était employée en tant que contrôleur et régulateur .Elle effectuait donc cycliquement une séquence que l'on peut d'écrire selon le schéma suivant :



Le 1er système d'éxploitation

Cette séquence,dont l'avantage est la simplicité,donne néanmoins une trés mauvaise utilisation du processeur, puisqu'en général la scrutation pendant laquelle le processeur ne fait rien d'autre, consomme beaucoup plus de temps que l'activation des périphériques .Il y a donc sous emploi du temps machine.
Une première amélioration en vue d'augmenter les performances de la machine consiste à employer les interruptions.Dans ce cas il n'y a plus besoin d'effectuer la scrutation et le traitement est interrompu par les évenements externes provoquant un déroutement vers le programme à exécuter.Les logiciels traitants les interruptions sont complexes à mettre en oeuvre et l'utilisation d'un système d'exploitation est utile pour simplifier l'écriture de ces logiciels.C'est cette dernière évolution qui optimise le temps machine et permet de multiplier tâches et périphériques, assurant une utilisation meilleure du temps machine.

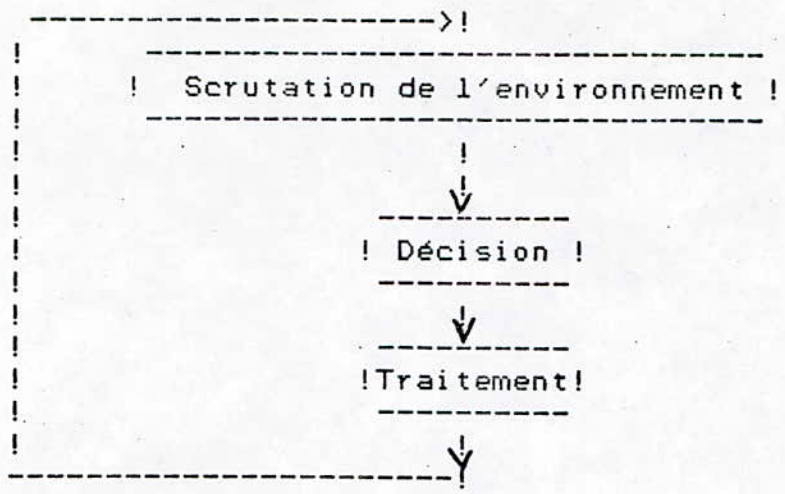
2.1.4-Construction logicielle des systèmes d'exploitation:

Les systèmes d'exploitation sont construits suivant une technique dite de couche (ou de pelure d'oignons).On part des notions les plus fondamentales pour aller vers celles qui sont les plus proches de l'utilisateur.

Avant de décrire ces différentes couches nous allons définir quelques mots clés pour la compréhension de ce qui va suivre.

RESSOURCE:une machine est composée de ressources qui sont réparties en trois groupes.

-Traitement (processeur)
-Périphériques (entrées/sorties)
-Mémoire centrale



Le 1er système d'éxploitation

Cette séquence,dont l'avantage est la simplicité,donne néanmoins une trés mauvaise utilisation du processeur, puisqu'en général la scrutation pendant laquelle le processeur ne fait rien d'autre, consomme beaucoup plus de temps que l'activation des périphériques .Il y a donc sous emploi du temps machine.
Une première amélioration en vue d'augmenter les performances de la machine consiste à employer les interruptions.Dans ce cas il n'y a plus besoin d'effectuer la scrutation et le traitement est interrompu par les évenements externes provoquant un déroutement vers le programme à exécuter.Les logiciels traitants les interruptions sont complexes à mettre en oeuvre et l'utilisation d'un système d'exploitation est utile pour simplifier l'écriture de ces logiciels.C'est cette dernière évolution qui optimise le temps machine et permet de multiplier tâches et périphériques, assurant une utilisation meilleure du temps machine.

2.1.4-Construction logicielle des systèmes d'exploitation:

Les systèmes d'exploitation sont construits suivant une technique dite de couche (ou de pelure d'oignons).On part des notions les plus fondamentales pour aller vers celles qui sont les plus proches de l'utilisateur.

Avant de décrire ces différentes couches nous allons définir quelques mots clés pour la compréhension de ce qui va suivre.
RESSOURCE:une machine est composée de ressources qui sont réparties en trois groupes.

-Traitement (processeur)
-Périphériques (entrées/sorties)
-Mémoire centrale

Donc une ressource est le moyen que met la machine à la disposition de l'utilisateur pour l'exécution d'une tâche.

EVENEMENT:Les systèmes d'exploitation doivent avoir pour caractéristiques particulières de réagir d'une manière optimale (selon un critère de temps ou de ressources) à des événements aléatoires survenants soit du monde externe(interruptions) soit à l'intérieur du système par échange d'information entre les divers fonctions.

TACHE:On appelle tâche une suite d'instructions capable de s'exécuter sur un processeur.Par exemple un programme écrit en langage d'assemblage s'exécutant sur une CPU est une tâche.

BLOCAGE:La notion de blocage étant assez délicate à expliquer,nous allons l'illuster par un petit exemple. Supposons qu'une société emploie deux employés partageant le même bureau,soient obligés de partager un seul bloc-note et un seul crayon.Au même instant les deux employés ont besoin du bloc-note et du crayon.L'employé E1 prend le crayon et E2 le bloc- note.E1 ne peut travailler n'ayant pas toutes les ressources(pas de bloc- note),de même que E2(pas de crayon).On est donc dans une séquence de blocage croisé,E1 et E2 s'attendent mutuellement.L'un attendant la disponibilité du crayon,l'autre celle du bloc-note. Pour résoudre ce type de situation,il faut protéger l'emploi conjoint du crayon et du bloc-note:on dira qu'il s'agit d'une "section critique".

2.1.4.1-Noyau du système d'exploitation:

Le noyau est un ensemble de logiciels fournissant les services de base servant à résoudre les problèmes tel que le blocage.C'est lui aussi qui alloue le processeur à une tâche,gère les différentes tâches d'état(tâches,files d'attente),met à jour les évenements,gère les interruptions, contrôle l'horloge séquençant toutes les temporisations.Le noyau implémente les divers fonctions de base du système d'exploitation.De plus il est en relation directe avec la structure de la machine(interruption) et donne les descriptifs des tâches.

-Exclusion mutuelle:deux (ou plusieurs) tâches accédant à une ressource critique doivent le faire de manière non bloquante,elles ne doivent pas non plus provoquer d'erreurs.

-Communication et synchronisation:deux(ou plusieurs) tâches coopérant pour réaliser une application ont besoin de se synchroniser(l'une attendant un signal de l'autre) ou plus généralement de communiquer(passage d'informations).

2.1.4.2-Sémaphore:

L'exemple des deux employés n'ayant qu'un bloc-note et qu'un crayon à partager montrait une mauvaise protéction de ressources conduisant à un blocage.Pour y remédier chaque ressource critique est associée à une variable Booléenne S prenant la valeur 1 ou 0 suivant que la ressource est respectivement occupée ou non. Le programme exécuté par

chacune des tâches pour aquérir le droit d'accés à la section critique est représenté par le programme suivant:

```
ATTENTE: MOV   AL,01H
         XCHG  S,AL
         CMP   AL,01H
         JEQ   ATTENTE
```

Dans ce petit programme AL est un registre de la machine et S la variable Booléenne protégeant la section critique. L'instruction XCHG permet d'échanger les contenus de S et AL,elle se compose d'une lecture puis d'une écriture.Dans le cas ou l'on dispose d'un système monoprocesseur,il suffit de rendre la séquence du programme précédant ininterruptible. Dans les systèmes multiprocesseurs il suffit que l'instruction XCHG exécute de manière liée les deux étapes d'écriture et de lecture qui la composent.Sinon deux processeurs P1 et P2 pourraient exécuter le programme précédant et concluraient chacun à la disponibilité de S,ce qui engendrerait un blocage.

2.1.4.3-Ordonnanceur:

L'ordonnanceur représente l'ensemble des procédures du noyau qui effectue un changement de contexte.Ces procédures sont donc informées des communications entre tâches,des sémaphores et des files d'attente associées.C'est aussi l'ordonanceur qui décide de l'allocation de telle ou telle tâche à tel ou tel processeur dans le cadre des systèmes mutiprocesseurs.
L'ordonnanceur est donc implémenté par une série de procédures qui gèrent un ou plusieurs processeurs pour l'ensemble des tâches d'application:

-Attribuant un processeur à une tâche.
-Retirant une tâche d'un processeur.

La décision de choix est dictée par une méthode de choix,basée par exemple sur les priorités des tâches.

2.1.4.4-Le chargeur:

Le chargeur charge en mémoire l'ensemble des programmes et sous-programmes à partir d'une adresse donnée par le moniteur,initialise la partie résidente et lance l'exécution au point d'entrée du programme précedant.

2.1.4.5-Bibliothèque:

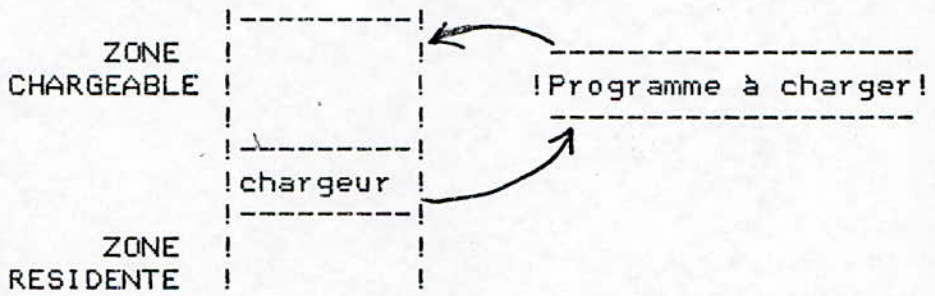
Chaque constructeur d'ordinateurs peut livrer avec sa machine un ensemble de programmes d'usage courant,écrits et mis au point pour son matériel.Cet ensemble est appelé bibliothéque de programmes(programmes de calculs statistiques,programme de comptabilité,...).

2.1.4.6-Moniteur:

Programme ou ensemble de programmes interprétant l'ensemble de base des commandes nécessaires pour utiliser un sytème.Il assure la gestion de toutes les ressources du système.

2.1.4.7-Gestion des programmes:

Dans une machine,le logiciel dont la taille est souvent plus importante que la mémoire physique est contenu sur des mémoires de masse.La machine comporte alors un programme résidant et une zone libre dans laquelle seront chargées selon les besoins les autres programmes d'application.La tâche qui traite cette fonction est dite CHARGEUR.
Les programmes résidants sont contenus soit sur ROM soit sur RAM.Dans le cas de la mémoire vive,on peut également,à l'initialisation du programme,faire charger la partie résidente .
Dans cette optique on emploie un petit logiciel dit CHARGEUR INITIAL (BOOTSTRAP) qui est contenu en ROM et qui est capable de chercher à une adresse fixe de la mémoire de masse,le CHARGEUR qui chargera ensuite le reste de l'application.



2.1.4.7-Gestion des fichiers:

On appele fichier une suite d'octets de taille fixée,disposant éventuellement de caractéristiques d'accés (protection).Le dispositif physique ou cette gestion est la plus simple est le disque ou la disquette.Les noms de fichiers sont regroupés dans un catalogue qui indique pour chaque fichier son adresse physique de début(piste/secteur).
La situation la plus simple est un catalogue ou les fichiers sont organisés en blocs d'informations.
Certaines applications imposent une organisation plus fine du disque,celle-ci est obtenue par la fabrication de catalogues hiérarchiques.A partir d'un catalogue général, les fichiers sont soit des sous-catalogues,soit des fichiers de données.Chacun des fichiers est obtenu par un chemin qui part de la racine et en parcourant les sous catalogues parvient au fichier de données.

2.1.4.9-Gestion des E/S:

Le traitement des entrées/sorties se fait à deux niveaux.Le plus proche de l'application dit "niveau logique" est celui des primitives de haut niveau (CREATE/OPEN/READ /WRITE/CLOSE) en général identique quel que soit le périphérique.Le second,le plus proche du périphérique,est celui des périlogiciels.Ces deux couches échangent entre elles une structure de données complexes dite "Structure d'Entrée/Sorties"(SES).

Cette structure est mise à jour par la couche logique pour indiquer les informations nécessaires à la couche physique.

La SES contient les informations suivantes:

(1)-STATUT du périphérique
(2)-Nombre de caractères à traiter
(3)-Numéro du périphérique
(4)-Fonction à réaliser(lire/écrire)

En conclusion,nous dirons qu'un système d'exploitation est l'interface utilisateur-machine qui offre les ressources nécessaires à l'application de l'utilisateur.

2.2-LE SYSTEME D'EXPLOITATION CP/M:

2.2.1- Introduction:

Le "CONTROL PROGRAM FOR MICROPROCESSOR" ou en abrégé CP/M est un programme de gestion de l'ensemble informatique piloté par un microprocesseur de la famille des 8080 ou Z80. Ensemble qui comprend à la fois l'unité centrale du micro-ordinateur mais aussi tous les périphériques auxquelles il peut être relié: écran,clavier, lecteurs de disquettes, imprimante et aussi un autre ordinateur.

Le système CP/M a été développé en 1975 par GARRY KILDALL, fondateur de DIGITAL RESEARCH et il est rapidement devenu un standard en raison de sa relative simplicité.

Dans ce chapitre,nous allons essayer d'expliquer le fonctionnement de CP/M ainsi que les modules constituant le système.

2.2.2 - LES COMMANDES DE CP/M

Les commandes de CP/M sont divisées en deux groupes,les commandes dites résidentes ou incorporées et les commandes non résidentes.

2.2.2.1-Les commandes résidentes:

Sous CP/M-86,cinq commandes sont incorporées:DIR,DDT, ERA, REN ET TYPE.

La commande DIR affiche le répertoire de la zone utilisateur courante,ou par exemple tous les fichiers ayant un certain type (par exemple DIR *.TXT).

La commande DDT,permet la mise au point de programmes.DDT charge en mémoire les fichiers en vu de les tester et de les mettre au point,et ceci en les exécutant pas à pas.

La commande ERA permet de détruire un fichier ou un groupe de fichiers.Par exemple la commande 'ERA *.TXT'spécifie que tous les fichiers dont le type est TXT seront détruits.

La commande REN permet de changer le nom d'un fichier son format est REN nouveau_nom=ancien_nom.
La commande TYPE permet la visualisation rapide d'un document.

2.2.2.2-Les commandes non résidentes:

Les commandes non résidentes sont celles se trouvant sur la disquette système en tant que fichier et pouvant être exécutées à la demande.Sous CP/M-86 VERSION 2.2 on trouve les commandes suivantes:

ASM86:Assemble les programmes écrits en langage d'assemblage.

RED :C'est un éditeur de texte.RED crée un tampon d'édition et permet à l'utilisateur de modifier le texte de ce tampon. Il est possible d'insérer du texte à partir de fichiers "bibliothéque".

PIP :Programme de transfert Inter-Périphérique,est un ordre qui copie des fichiers d'un périphérique sur un autre.
-copie un ou plusieurs fichiers d'une disquette sur une autre: PIP E:=A:*.*,copie tous les fichiers de 'A' sur 'E'.
-Copie un ou plusieurs fichiers d'une zone utilisateuer à une autre:PIP B:PROG1.TXT(G1)=A:PROG.TXT(G0),copie le fichier PROG.TXT ,se trouvant sur l'unité A en zone user 0, sur l'unité B en zone user 1 tout en lui changeant de nom qui devient PROG1.TXT.
-Envoit un fichier sur une unité(autre qu'une unité de disque),par exemple une imprimante:PIP PRN:=PROG.TXT,envoit le fichier PROG.TXT sur l'unité d'impression.

STAT:Est un ordre qui affiche les informations concernant les fichiers et les unités sur lesquelles,il exerce un certain contrôle.
-STAT (seul) affiche la quantité d'espace libre restant sur une disquette.
-STAT (suivi d'un nom de fichier) affiche ,l'unité dans laquelle se trouve le fichier,le numéro de la zone utilisateur (user),le nombre d'enregistrements utilisés par chaque fichier (RECs),le nombre de K-octets utilisés par chaque fichier (Bytes),le nombre de blocs de contrôle qu'utilise chaque fichier (FCBs), ainsi que les attributs du fichier RO (lecture uniquement) ou RW (lecture/écriture).

Un micro-ordinateur étant amené à communiquer avec des périphériques,le logiciel d'exploitation a besoin de savoir quels périphériques sont connectés au micro pour que ce dernier puisse fonctionner correctement,on doit indiquer au système d'exploitation quels périphériques sont connectés en attribuant des noms logiques aux périphériques physiques. Certains noms logiques sont attribués d'office à certains périphériques,telque:

- CON:=CRT:
- AXI:=PTR:
- LST:=LPT:

Les noms logiques sont nécessaires pour identifier lequel des périphériques est relié au micro pour exécuter une fonction.

L'affectation des noms logiques aux périphériques physiques est faite par l'ordre " S T A T ".

NOM LOGIQUE	FONCTION
CON:	Console utilisateur.Elle communique avec le logiciel d'exploitation,accepte des données entrées au clavier et les affiches sur l'ecran.
AXI:	Ce périphérique reçoit des informations (d'entrées uniquement).
AXO:	Ce périphérique envoit des informations (de sortie uniquement).
LST:	Ce périphérique énumère des informations sur une imprimante par exemple.

NOM PHYSIQUE	PERIPHERIQUE DESIGNE
TTY:	Interface de l'imprimante qui permet les E/S.
CRT:	Terminal d'affichage (écran et clavier).
PTR:	Interface de communication ,en entrée uniquement.
PTP:	Interface de communication ,en sortie uniquement.
LPT:	Interface de l'imprimante ,en sortie uniquement.

L'ordre "STAT VAL:" permet d'afficher les affectations physiques-logiques de périphériques possibles.

```
A>STAT VAL: (return)
  CON:=TTY:CRT:BAT:
  AXI:=TTY:PTR:
  AXO:=TTY:PTP:
  LST:=TTY:CRT:LPT:
```

L'ordre "STAT DEV:" permet d'afficher les affectations en cours.

2.2.3 - L'interface de contrôle:

Le système d'exploitation CP/M est composé de trois modules fonctionnels:

-CCP (Consol Command for Microprocessor);processeur de commande console.
-BIOS (Basic Input/Output System);système d'entrée/ sortie de base.
-BDOS (Basic Disk Operating System);système de base de gestion de disque.

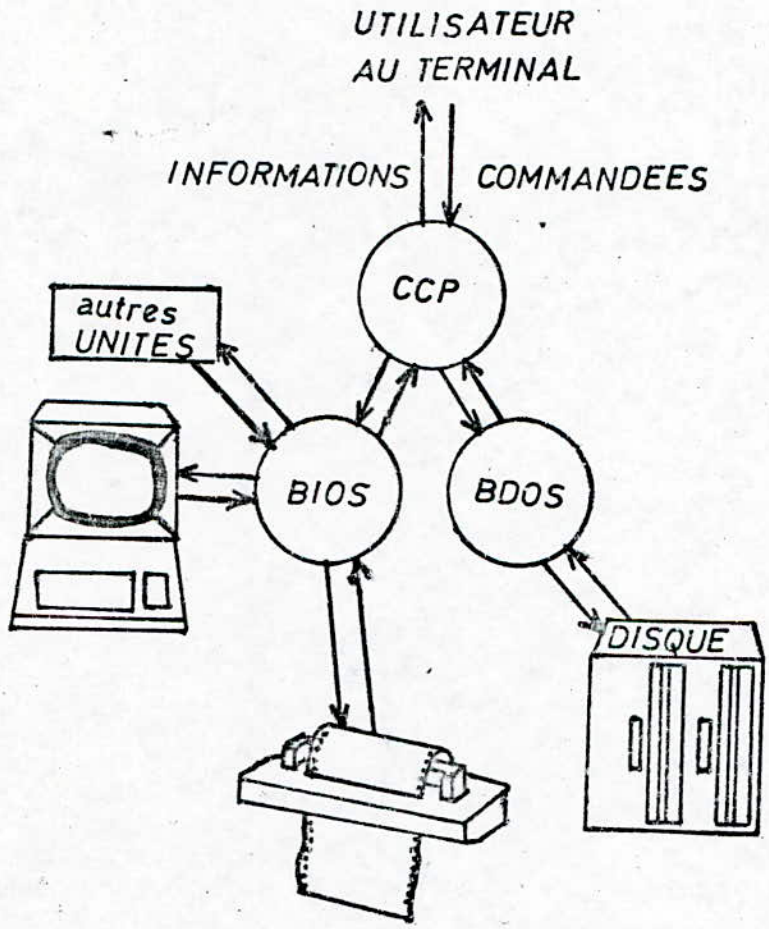
Le rôle de CCP est de communiquer avec l'utilisateur,et d'interpréter les commandes frappées au clavier.CCP est donc essentiellement un interprète de langage de commande. Il utilise les ressources des deux modules BIOS et BDOS. Formellement il peut être vu comme la partie "intelligente" du système,tandisque les autres modules sont des modules de service.

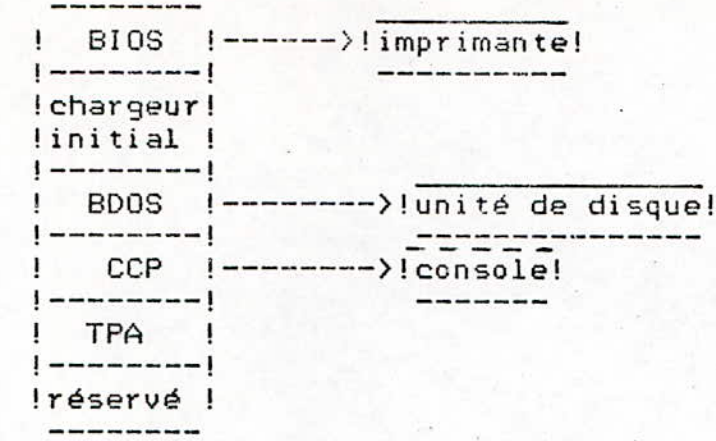
BIOS contient les préposés aux périphériques,c'est à dire les programmes qui communiquent avec les différentes unitées physiques connectées au système.Son rôle est d'envoyer ou de recevoir des informations d'etat ,ou des données entre une unité et CCP.BIOS est appelé par CCP au moyen de paramètres précisant le périphérique désiré.

BDOS s'occupe de la gestion des fichiers sur disque,il contient un certain nombre d'utilitaires,qui réalisent les fonctions désirées.BDOS a pour objectif de rendre la gestion de fichiers transparente à l'utilisateur.Il s'occupe de tout le travail nécessaire à la découverte des différents blocs d'informations répartis sur toute la disquette et alloue ou libère efficacement l'espace de stockage.

2.2.4-Allocation mémoire:

CP/M sépare la mémoire en quatre zones,comme indiqué sur la figure ci-dessous.Le haut de la mémoire est réservé aux program- mes de CP/M proprement dits,c'est-à-dire à CCP,BIOS et BDOS.Quelques emplacements, en bas de la mémoire sont réservés au système:ce sont les 256 premiers mots,autrement dit la page zéro.Enfin la plus grande partie de la mémoire appelée TPA(Transient Program Area,ou zone de programme temporaire) est disponible pour exécuter des programmes.Dans une configuration 16K,la base de CCP appelée Cbase est à l'adresse 2900H.Cette adresse est incrémentée de 4000H par 16K ajoutés à la configuration,étend la TPA de cette même quantité de mémoire.Sous certaines conditions,on peut avoir accés à toute la mémoire en recouvrant CCP ou d'autres zones appartenants au programme de CP/M.Toute fois celà nécessite que le programme ramène CCP en mémoire à la fin de son exécution.


UTILISATEUR
AU TERMINAL
INFORMATIONS
COMMANDEES
CCP
autres
UNITES
BIOS
BDOS
DISQUE



2.2.5-Déscription de CP/M:

2.2.5.1-Système de fichier

Une des fonctions principales de tout système disque (DOS,Disque Operating System) est de permettre une gestion efficace des fichiers disques.Toute gestion de fichiers se fait entre CCP et BDOS.La partie BIOS de CP/M transmet et reçoit essentiellement de simples chaines de données.

Sous CP/M un fichier est une unité logique qui contient du texte,des données ou des programmes.La tâche du système de gestion du disque est de réaliser cette "facilité" logique avec les ressources physiques du support de stockage (disque ou disquette).Comme il n'est pas possible de garder un fichier comme une suite d'enregistrements sur la disquette,il est nécessaire de recourir à des enregistrements répartis sur toute la surface de celle-ci.La liste des enregistrements appartenants à un fichier disque est contenue dans le descripteur ou dans la terminologie CP/M "Bloc de Contrôle du Fichier" (FCB).Le FCB utilise 36 octets contenant toutes les informations relatives au fichier dont le nom et le type.Chaque fois que l'on accède à un fichier,son FCB est amené en mémoire pour faciliter les accés aux divers enregistrements et pour le remettre à jour. Une autre fonction de gestion de fichier est de veiller à l'intégrité des fichiers dans la période oû ils sont accessibles.Sous CP/M les fichiers peuvent être déclarés en lecture seulement,ou en écriture/lecture,ou encore un fichier peut être déclaré "exécutable",mais non lisible.

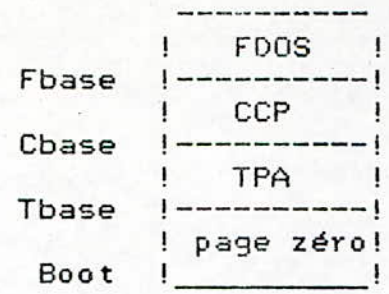
2.2.5.2-Fonctionnement et exécution de CP/M:

Sous le système d'exploitation CP/M,chaque programme en langage machine est chargé dans la TPA.Dans CP/M standard son exécution commence à l'adresse 100H.Dans la terminologie CP/M ce programme est dit non résident,et peut utiliser les ressources du système en exécutant un appel à l'adresse

05H.Cette adresse sert d'interface avec le système d'exploitation.Ceci entraine le transfert du contrôle à CP/M.L'appel au système d'exploitation doit être accompagné d'un paramètre précisant la requête demandée,ce paramétre est passé comme numéro de fonction dans le registre "C" pour accéder aux divers unités d'E/S y compris les fichiers sur disque.
Du point de vue de l'utilisateur de CP/M le système travaille de la manière suivante:CCP (processeur de commande console),affiche le message d'attente du système,et attend une commande.Les transmissions physiques sont gérées par BIOS.Quand CCP reçoit la commande (et les noms de fichiers associés),il l'exécute imméditeme nt s'il s'agit d'une commande résidente.Sinon,CCP suppose qu'il s'agit d'un programme non résident avec un nom de type CMD ou COM (par exemle PIP.CMD).Il demande alors à BDOS de trouver le fichier et d'en lire une copie dans la TPA.CCP construit alors un bloc de contrôle de fichier (FCB) pour le(s) fichier(s) concerné(s) par la commande(ou le programme).BDOS trouve le fichier en cherchant le FCB créé par CCP.Ce FCB est à moitié rempli et ne contient que le nom du fichier. BDOS est amené à comparer le nom du fichier du FCB,créé par CCP,aux noms de fichiers dans les FCB du répertoire.Quand BDOS trouve le fichier,il fournit les informations complémentaires dans le FCB creé par CCP.Chaque fois que le programme accéde à un fichier particulier,BDOS modifie certaines informations du FCB.Quand le programme ferme le fichier,BDOS fait une dernière modification et copie ensuite le FCB de la mémoire centrale (avec les informations des fichiers mis à jour) sur la disquette.Si le premier mot n'est pas une commande résidente ou si CCP ne trouve pas de fichier du type CMD ou COM dont le nom corresponde,CCP affiche le mot tapé suivit d'un point d'interrogation.
Pour communiquer avec les périphériques,le système utilise un "porteur de message",BIOS(Basic Disque Input/Output System).BIOS réalise des opérations simples,comme lire un caractère à partir du clavier ou écrire un caractère sur le port de communication ou sur l'imprimante.

2.2.5.3-Les fonctions FDOS:

La mémoire de CP/M(après chargement dans la mémoire centrale) est montré sur la figure ci-dessous.Le point d'entrée principal de FDOS est à l'adresse "boot+005H",en page "zéro".La page zéro est la zone de mémoire centrale que le système réserve pour y ranger des informations système. L'adresse boot lance les instructions en code machine qui réalisent une relance du système.Les programmes non résidents doivent donc seulement sauter à l'adresse boot pour rappeler CCP et reprendre l'exécution de CP/M.Les valeurs exactes de boot, tbase,cbase et fbase dépendent de la version de CP/M.



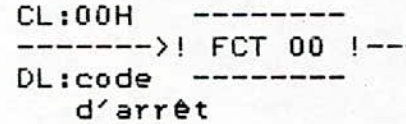
Quand CCP reçoit une ligne de commande et que l'image mémoire du fichier est amené en mémoire(TPA),le programme peut alors accéder aux fonctions FDOS.
FDOS est divisé en deux parties,qui ont déjà été introduites: BIOS qui contrôle les périphériques et les transmissions et BDOS qui explore les disques pour chercher les fichiers,et gère les blocs de contrôle des différents fichiers pour en faciliter l'accés.

L'accés à BIOS et BDOS se fait à travers l'interruption software du microprocesseur 8088 qui est réservée par INTEL pour être utilisé par CP/M-86 :INT 224.

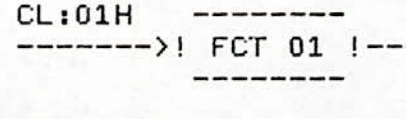
2.2.5.3.1-Les fonctions BDOS

Une opération BDOS est demandée en passant un numéro de fonction dans le registre "C" et une information ou une adresse dans le registre "D".
Les fonctions BDOS que nous utiliserons sont les suivantes:

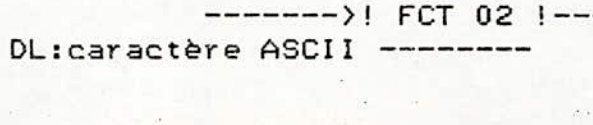
FONCTION "0":Fonction system reset,réinitialisation du système rend le contrôle au système CP/M au niveau de la commande de CCP.Le code d'arrêt passé dans DL peut avoir deux valeurs;si DL=00H,le programme courant est terminé et le contrôle est rendu à CCP.Si DL=01H,le programme reste en mémoire et l'état de l'allocation mémoire reste inchangé.



FONCTION "1":Consol input,lecture console,retourne un caractère ASCII dans AL.



FONCTION "2":Consol output,écriture console,le caractère ASCII dans DL est envoyé vers le terminal.



FONCTION "3": Reader input, retourne un caractère ASCII en provenance de l'unité de lecture (port de communication) dans AL.

```
        CL:03H   --------
        ------->! FCT 03 !------> retourne
                 --------         AL:caractère ASCII
```

FONCTION "4":Punch output,écrit sur le port de communication,envoie un caractère ASCII sur le port.

```
        CL:04H   --------
        ------->! FCT 04 !------>retourne
   DL:caractère  --------
      ASCII
```

FONCTION "5":List output,écriture sur l'unité d'impression. Envoie un caractère ASCII chargé dans DL vers l'unité d'impression.

```
        CL:05H   --------
        ------->! FCT 05 !------>retourne
   DL:caractère  --------
      ASCII
```

FONCTION "6":Direct consol I/O,E/S console direct.DL contient le code de la requête,ou le caractère ASCII à envoyer vers le terminal.

- -DL=FFH:indique une demande d'écriture console.
- -DL=FEH:indique une demande d'état console.
- -DL=CARACTERE ASCII:contient le caractère ASCII à écrire sur l'écran.

```
        CL:06H
        DL:FFH   --------
        ------->! FCT 06 !------>retourne
        DL:FEH   --------         AL:00 pas de caractère prêt
        DL:caractère ASCII        AL:FF sinon
```

FONCTION "9":Print string,impression d'un tampon.Imprime une chaîne d'octets complète commençant à l'adresse donnée par DX et se terminant à '$'.

```
        CL:09H   --------
        ------->! FCT 09 !------>retourne
       DX:offset --------
          de la chaine
```

FONCTION "10":Read consol buffer,lecture du buffer de la console du buffer.Envoie l'adresse d'un tampon de lecture et retourne avec le tampon rempli.La lecture est terminée soit quand le buffer est plein ou quand un RETURN est entré.

```
        CL:0AH   --------
        -------->! FCT 10 !------>retourne
      DX:offset  --------
         du buffer
```

Les fonctions qui suivent permettent l'accés et la manipulation de fichiers sous CP/M-86.Dans un grand nombre de ces opérations,DX contient l'offset du bloc de contrôle de fichier (FCB).Ci-dessous est donné le format du FCB ainsi que la définition de chaque champ de ce dernier.

```
-------------------------------------------------------
!dr!f1!f2!...!f8!t1!t2!t3!ex!s1!s2!rc!d0!...!dn!cr!r0!r1!r2!
-------------------------------------------------------
```

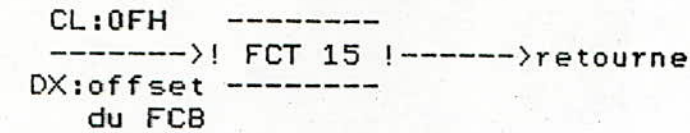
OU:-dr:code de l'unité magnétique(0 à 16)
0:unité par défaut
1:selection automatique de l'unité A
2:selection automatique de l'unité B
3,4,...,16:unités C,D,...,O,P pour les systèmes ayant 15 unités magnétiques.
-f1...f8:contient le nom du fichier en ASCII(8 caractères au maximum)
-t1...t3:contient le type du fichier en ASCII(3 caractères au maximum)
-ex:normalement mis à zéro par l'utilisateur.Mais prend des valeurs comprises entre 0 et 31 lors d'une opération d'E/S de fichiers.
-s1,s2:reservés à l'utilisation du système.
-rc:contient le nombre d'enregistrements,variant de 0 à 128.
-d0...dn:réservé pour l'utilisation du système,et contient la table d'allocation disque.
-cr:contient le numéro du prochain enregistrement à lire ou à écrire lors d'une opération de fichier sequentiel,ou est mis à zéro dans d'autres cas.
-r0,r1,r2:optionnellement contiennent le numéro d'enregistrement direct.Cette valeur est comprise entre 0 et 65535,avec un overflow dans r2.R0 et r1 constituent une valeur 16 bits dont r0 est l'octet de poids faible.

FONCTION "14":Select disk,selection d'un disque.L'unité de disque sélectionnée est passée dans DL:0 pour A,1 pour B,...etc...

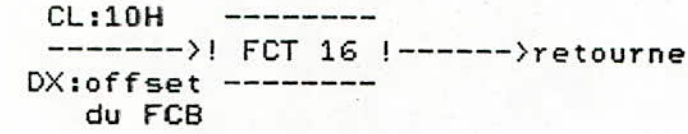
```
        CL:0EH   --------
        -------->! FCT 14 !------>retourne
        DL:unité --------
           choisie
```

FONCTION "15":Open file,ouverture d'un fichier. En donnant une adresse de bloc de contrôle de fichier, BDOS cherche un FCB compatible dans la zone répertoire du disque;il retournera un code répertoire correct indiquant que toutes les informations ont été copiées dans le FCB.Ceci rend possible un accés ultérieur au fichier.Le code répertoire

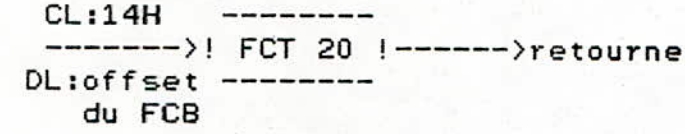
retourne une valeur entre 0 et 3 si l'ouverture du fichier a été faite,et une valeur 0FFH si le fichier n'a pu être trouvé.



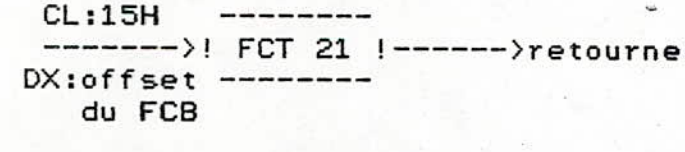
FONCTION "16":Close file,fermeture d'un fichier.On donne l'adresse d'un bloc de contrôle de fichier,et BDOS enregistrera ce nouvel FCB dans le répertoire du disque (l'inverse de l'opération d'ouverture d'un fichier avec les mêmes codes de retour).



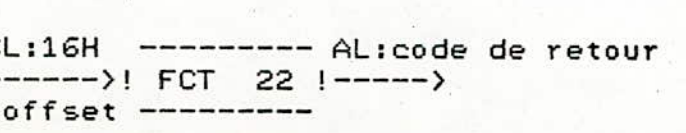
FONCTION "20":Read sequential,lecture sequentielle.Lit les 128 octets(un enregistrement) suivants dans la mémoire,à l'adresse DMA courante.La valeur "00H" est retournée dans AL si la lecture a eu lieu,sinon la valeur "01H" est retournée s'il n'y a pas de données à la position d'enregistrement suivante du fichier.



FONCTION "21":Write sequential,écriture séquentielle.Si le fichier a été ouvert ou nouvellement créé,cette fonction écrira 128 octets commençants à l'adresse DMA courante,sur le fichier désigné par le FCB.



FONCTION "22":Make file,création d'un fichier.Similaire à la fonction d'ouverture,cette fonction crée un nouveau fichier en même temps qu'elle l'ouvre.En donnant l'adresse d'un FCB muni d'un nouveau nom de fichier:cette fonction créera le fichier et initialisera son FCB (sur disque aussi bien qu'en mémoire) comme étant celui d'un fichier vide.On doit s'assurer qu'on ne crée pas un fichier dont le nom est déjà utilisé sur le même disque,rendant ainsi les deux fichiers inaccessibles.



du FCB

FONCTION "26":Set DMA Address,positionnement de l'adresse DMA.DMA (accés direct mémoire) est souvent utilisé dans la communi- cation avec des contrôleurs de disque qui accédent directement à la mémoire de l'ordinateur pour transferer des données de et vers le sous système disque.L'adresse DMA,dans CP/M signifie l'adresse à laquelle les 128 octets de données enregistrées résident avant une écriture disque et aprés une lecture disque.Sous CP/M-86,le positionnement de la fonction DMA est utilisé pour spécifier le déplacement (offset)du buffer d'écriture ou de lecture vis-à-vis de la base DMA courante.Par conséquent pour spécifier l'adresse DMA,l'appel des deux fonctions 26 et 51 est recquis.Ainsi, l'adresse DMA devient la valeur spécifiée par DX plus la valeur de base de DMA jusqu'à ce qu'elle soit changée par un autre positionnement DMA ou une fonction de base ultérieure.

```
      CL:1BH   --------  BX:offset de l'allocation
      ------->! FCT 26 !------>
               --------  ES:base de segment
```

FONCTION "33":Read random,lecture directe.Cette fonction se sert du champ r du FCB pour sélectionner un numéro d'enregistrement,et le lire.Au retour,l'adresse DMA pointe sur l'enregistrement désiré.Le numéro d'enregistrement n'est pas incrémenté,comme dans les opérations de lecture séquentielle.

code d'erreur:

AL=01:quand une opération de lecture dirécte accéde à un bloc de données qui n'a pas été écrit antérieurement.
AL=02:non retourné par une lecture directe.
AL=03:causé par une lecture directe sur un FCB qui n'a pas été ouvert.
AL=04:équivalente à l'erreur "01"
AL=05:non retourné par une lecture directe.
AL=06:retourné quand l'octet r2 du FCB est non nul.

```
      CL:21H   --------
      ------->! FCT 33 !------>retourne
   DX:offset   --------
      du FCB
```

FONCTION "34":Write random,écriture directe.Cette fonction est initialisée de la même manière que l'opération de lecture directe,sauf qu'elle écrit des données sur le disque à partir de l'adresse DMA courante.Si l'espace fichier n'est pas encore alloué,la fonction réalise cette opération avant d'écrire.Le numéro d'enregistrement n'est pas incrémenté. Même code que précedemment.

```
      CL:22H -------- AL:code de retour
      ----->! FCT  34!----->
   DX:offset --------
      FCB
```

2.2.5.3.2-Le BIOS (Basic I/O system)

Toutes les opérations simples d'E/S de caractères ASCII sont assurées par le BIOS.Nous avons déjà dit que les assignations unités logiques et périphériques physiques sont faites par la commande STAT.En fait,STAT fait appel à BIOS pour effectuer ces assignations.BIOS utilise un octet appelé "IOBYTE" (octet d'E/S). L'IOBYTE est divisé en quatre champs utilisant chacun 2 bits dont la valeur permet l'assignation.

```
          ----------------------
IOBYTE    ! LST ! AXO ! AXI ! CON !
          ----------------------
bits:     7  6 5  4 3  2 1  0
```

Les différentes assignations possibles et la valeur de chaque champ sont:

CON:(bits 0,1)
    0- CON est assignée à TTY
    1- CON est assignée à CRT
    2- CON est assignée à BAT.Le mode BATCH (BAT) permet une double assignation:AXI devient écran et LST clavier de sorte que les commandes ne sont plus prises au clavier mais arrivent par le port AXI.
    3- non utilisé

AXI:(bits 2,3)
    0-AXI est assigné à TTY
    1-AXI est assigné à PTR
    2-non utilisé
    3-non utilisé

AXO:(bits 4,5)
    0-AXO est assigné à TTY
    1-AXO est assigné à PTP
    2-non utilisé
    3-non utilisé

LST:(bits 6,7)
    0-LST est assigné à TTY
    1-LST est assigné à CRT
    2-LST est assigné à LPT
    3-non utilisé

2.2.6 - Le Bootstrap:

Cette partie décrit les composants du disque CP/M-86 standard.CP/M-86 est distribué sur une disquette IBM simple densité,utilisant un format de fichier compatible avec tous les précédents systèmes d'exploitations CP/M-80.
Les deux premières pistes de la disquette sont reservées

pour le système d'exploitation et les programmes BOOTSTRAP, tandis que le reste contient l'information directory qui charge les programmes et les fichiers.
Les principales composantes du système sont les suivantes:
-BOOTSTRAP ROM (BOOT ROM).
-Le chargeur pour le démarrage à froid (LOADER).
-Le système CP/M-86 (CPM.SYS).

Quand elle est innstallée,la BOOT ROM devient une partie de l'espace mémoire adressable,elle commence à la location mémoire 0FF000H,et reçoit le contrôle lorsque le bouton de réinitialisation du système est pressé.Dans un environnement non standard,la BOOT ROM est remplacée par un chargeur intial équivalent,c'est pourquoi la ROM elle même n'est pas incluse avec le CP/M-86.La BOOT ROM peut être obtenue de chez DIGITAL RESEARCH ou programmable à partir du listing de la documentation de DIGITAL,ou encore obtenue directement à partir du fichier source qui est inclus dans les disquettes distribuées sous le nom de "BOOT.A86". La BOOT ROM lit le CHARGEUR à partir des deux premières pistes du système dans la mémoire et passe le contrôle au chargeur.

Le programme chargeur contient les fichiers nécessaires et suffisants au "CPM.SYS" pour lire la disquette système en mémoire.Quand le chargeur termine son opération,le programme CPM.SYS reçoit le contrôle et procède à l'exécution de la commande entrée au clavier.Le fichier CPM.SYS est constitué par CCP,BIOS et BDOS,tous les trois dans un format CMD.Le fichier CPM.SYS,une fois chargé,passe le contrôle au point d'entrée de "INIT" à l'adresse d'offset 2500H.Toute initialisation additionnelle,non effectuée par le chargeur, prend place dans le sous programmes INIT,et la fin,INIT exécute un saut à l'adresse zéro (JMP 00H) pour commencer l'exécution de CCP.

2.3-LE SYSTEME D'EXPLOITATION MS-DOS.

2.3.1-INTRODUCTION:

Le Microsoft Disk Operating System(MS-DOS) est l'interface entre le matériel,l'utilisateur et les autres logiciels.Le MS-DOS est un ensemble de programmes permettant la manipulation des informations stockées sur disque rigide ou disquette.Au travers du MS-DOS nous communiquons avc l'unité centrale(CPU),l'écran vidéo,l'imprimante et les autres périphériques à l'aide de l'interface RS_232.

Le MS-DOS possède une bibliothèque comprenant plus de 40 commandes permettant la manipulation de fichiers, l'exécution et le développement de programmes.Il possède aussi des utilitaires tels que l'éditeur de lien "LINK", l'éditeur de ligne "EDLIN", l'utilitaire de mise au point des programmes exécutables "DEBUG".

2.3.2-Manipulation des fichiers:

Les commandes de manipulation de fichiers,permettent non seulement de copier ou de détruire des fichiers,etc..., mais aussi de regrouper des fichiers sous forme de répertoire.Par exemple chaque utilisateur peut posséder son propre répertoire.De plus, MS-DOS permet la création de répertoires à l'intérieur d'autres, créant ainsi une structure hiérarchisée.Pour ce faire,MS-DOS dispose de deux types de commandes:les commandes internes et les commandes externes.A l'initialisation du système,les commandes internes sont chargées en mémoire et y restent.Les commandes externes restent sur disque(ou disquette),et sont chargées en mémoire et exécutées en cas de besoin.Une fois exécutées,elles sont retirées de la mémoire permettant ainsi une utilisation optimum de celle-ci.

a)les commandes internes:

CHDIR(CD):cette commande modifie le répertoire courant, ou affiche le nom du répertoire courant.

CD NOUR permet l'accés au répertoire NOUR.
CD .. permet de revenir au répertoire source.

COPY:permet de faire une copie ou une concaténation de fichiers.

CLS:efface l'écran.

CTTY:permet de modifier le périphérique duquel les commandes sont envoyées.Sa syntaxe est: CTTY devicename
Ou devicename est un nom réservé de périphériques.Ces noms sont:
CON (console utilisateur),AUX (port de communication),PRN (imprimante).

Par exemple "CTTY AUX": la commande d'E/S passe au périphérique relié au port de communication.

DEL:efface le ou les fichiers spécifiés.
DIR:liste les fichiers du répertoire courant ou spécifié.

MKDIR:permet le création d'un nouveau répertoire.

RMDIR:supprime un répertoire dans une structure hiérarchisée.

TYPE:permet la visualisation rapide d'un fichier.

PAUSE:suspend l'éxecution d'un fichier BATCH ou se trouve la commande.

b)Les commandes externes:

ASSIGN:permet de router toutes les demandes pour une unité de disque sur une autre.
Par exemple ASSIGN B=C,cette commande route toutes les demandes pour l'unité B vers l'unité C.

CHKDSK:analyse le contenu d'une disquette de l'unité spécifié.Elle produit un rapport sur l'état du contenu de la disquette.En ajoutant la clé /F,CHDSK tente de corriger toutes les erreurs découvertes notemment elle classe les pistes défectueuses comme indésirables et celles-ci ne seront plus adressées.La clé /V affiche les messages au cours de son passage et donne aussi la liste des fichiers masqués.

COMP:permet de comparer le contenu de 2 disquettes.

DISKCOPY:permet de copier tout le contenu de la disquette.

FORMAT:permet le formattage.

TREE:permet la visualisation des noms des répertoires et sous-répertoires existants ainsi que leurs chemins d'accés.

2.3.3-Le traitement par lots:

Le MS-DOS permet de traiter une séquence de commandes souvent utilisées pour une même tâche.Cette séquence est placée dans un fichier spécial appelé fichier de traitement par lots (BATCH).La séquence est alors effectuée en entrant le nom du fichier BATCH.Ce fichier doit avoir l'extension BAT.
Par exemple le fichier BATCH suivant permet le formatage ainsi que la vérification d'une disquette.Il a pour nom "FORMATB.BAT".

```
REM FICHIER SERVANT A VERIFIER LES DISQUETTES.
PAUSE INSEREZ UNE DISQUETTE DANS L'UNITE B
FORMAT B:
CHKDSK B/F/V
```

2.3.4-Bref aperçu sur le DOS (système de gestion de disques)

Comme CP/M,le DOS est constitué de 3 modules:le système d'E/S,le processeur de commande console et les programmes utilitaires.

Le système d'E/S,gère les E/S clavier-écran,le système de fichiers,l'imprimante,ainsi que tout échange avec le monde externe à travers le port de communication.Pour communiquer avec un périphérique,un ordinateur doit au préalable avoir un logiciel permettant d'adresser le périphérique choisi.Le logiciel accomplissant cette fonction, est le BIOS.Le BIOS posséde des routines toutes faites qui contrôlent et gèrent toutes les E/S.
La description du BIOS a été faite dans l'étude de CP/M.

Le rôle du processeur de commande est d'interpréter les commandes entrées au clavier,il communique avec les autres modules pour l'exécution de ces commandes.

Les programmes utilitaires sont déstinés à faciliter la tâche de l'utilisateur.On pourra classer dans cette catégorie le formatage,la copie des fichiers,le changement de nom...etc...

CHAPITRE 3:DESCRIPTION DES PRINCIPALES COMPOSANTES DU MODULE SYSTEME.

3.1-DESCRIPTION GENERALE

Le module système possède une architecture double processeur utilisant le 8088 de 16 bits couplé au Z80A de 8 bits.

Chaque microprocesseur gère une partie des fonctions du système en plus de l'exécution des programmes utilisateurs ou d'application sur 8 ou 16 bits.

La fig-3.1 est un schéma de bloc montrant la liaison entre les microprocesseurs et leur logique respective.

Les microprocesseurs opèrent à partir d'un bloc de mémoire partagé de 62K octets de RAM et transfèrent les données à travers ce même bloc pour contrôler et gérer les fonctions du système.Chaque microprocesseur possède,en plus du bloc mémoire partagée,2 K-octets de RAM et des circuits de liaison avec les périphériques.

Le 8088 contrôle le moniteur,le clavier,le connecteur de communication ainsi que toutes les options ajoutées au système. Le 8088 communique avec ses supports logiques et avec les différentes options installées via un bus d'adresse unidirectionnel de 20 bits et un bus de données de 8 bits bidirectionnel.

Le Z80A transmet les données ou les adresses et les signaux de contrôle au contrôleur RX50.Le RX50 utilise ces signaux pour effectuer une lecture ou une écriture sur les deux unités de disquettes.

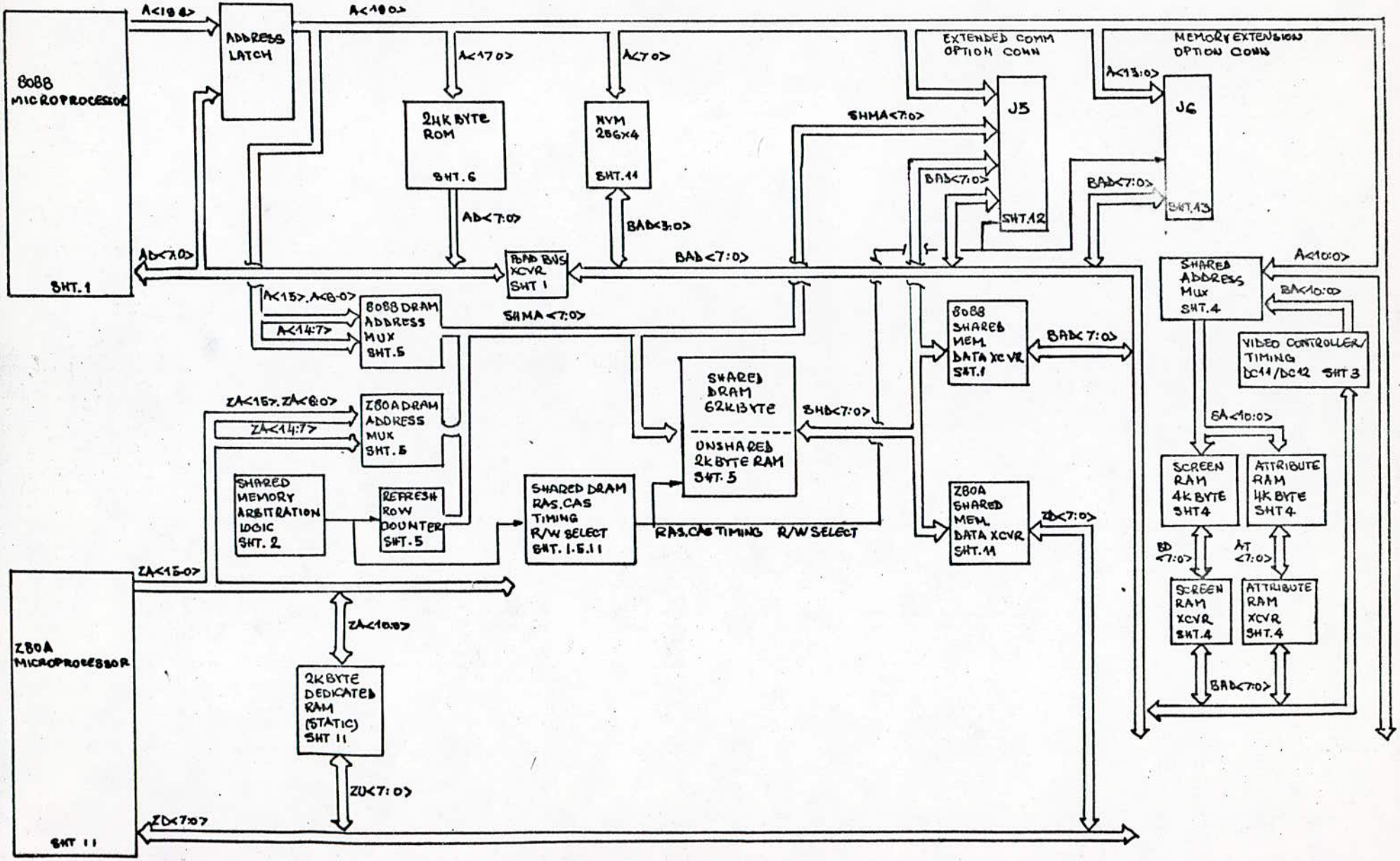
Le Z80A communique avec sa logique et avec sa logique et avec le RX50 via un bus d'adresse unidirectionnel de 16 bits et un bus de données bidirectionnel de 8 bits.

Le module système entretient aussi 2 générateurs de vitesse programmables (transmission/réception).Ce générateur fournit les signaux d'horloges de transmission et de réception au canal de communication du MPSC (Multi Protocol Serial Controller).Ces horloges peuvent être programmées indépendemment.

Le générateur de vitesse de transmission de l'imprimante fournit les signaux d'horloges du transmetteur et du receveur du MPSC pour le canal de l'imprimante et pour le PUSART(Programable Universal Synchronous/Asynchronous Transmitter Receiver) du clavier.

Les horloges du transmetteur et du receveur du MPSC de l'imprimante ne peuvent pas être programmées indépendemment;celles du PUSART du clavier sont fournies par le générateur de vitesse de l'imprimante,celle-ci est fixée à 4,8 KBauds.

Le circuit d'horloge du module système fournit 3 groupes d'impulsions d'horloge dérivant de l'horloge mère pilotée par un maître oscillateur à quartz.Un de ces groupes est utilisé par le 8088 et sa logique.Le deuxième est utilisé par le Z80A et sa logique.Le dernier est utilisé par


8088 MICROPROCESSOR
SHT. 1
A<19:8>
ADDRESS LATCH
A<19:0>
A<17:0>
A<7:0>
2HK BYTE ROM
SHT. 6
NVM 256x4
SHT. 11
AD<7:0>
BAD<3:0>
BAD BUS XCVR SHT 1
BAD<7:0>
EXTENDED COMM OPTION CONN
J5
SHT.12
SHMA<7:0>
BAD<7:0>
MEMORY EXTENSION OPTION CONN
A<13:0>
J6
SHT.13
SHARED ADDRESS MUX SHT.4
A<10:0>
BA<10:0>
VIDEO CONTROLLER/ TIMING 6C11/6C12 SHT 3
A<15>, A<8:0>
A<14:7>
8088 DRAM ADDRESS MUX SHT.5
SHMA<7:0>
8088 SHARED MEM. DATA XCVR SHT.1
BADC 7:0>
ZA<15>, ZA<8:0>
ZA<14:7>
Z80A DRAM ADDRESS MUX SHT. 5
SHARED DRAM 62KBYTE
UNSHARED 2K BYTE RAM SHT. 5
SHDC<7:0>
SA<10:0>
SCREEN RAM 4K BYTE SHT4
ATTRIBUTE RAM 4K BYTE SHT4
SHARED MEMORY ARBITRATION LOGIC SHT. 2
REFRESH ROW COUNTER SHT. 5
SHARED DRAM RAS, CAS TIMING R/W SELECT SHT. 1.5.11
RAS, CAS TIMING R/W SELECT
Z80A SHARED MEM. DATA XCVR SHT. 11
ZDC<7:0>
BD <7:0>
AT <7:0>
SCREEN RAM XCVR SHT.4
ATTRIBUTE RAM XCVR SHT.4
BADC<7:0>
ZA<15:0>
Z80A MICROPROCESSOR
SHT 11
ZA<10:0>
2KBYTE DEDICATED RAM (STATIC) SHT 11
ZD<7:0>
ZDC<7:0>

la logique du contrôleur RX50.

Le module système inclut les composantes suivantes:

-Microprocesseur 8088,
-Microprocesseur Z80A,
-64 KO de mémoire partagée,
-2 KO de RAM dédiée au au Z80A,
-24 KO de ROM,
-256*4 bits de NVM (None Volatil Memory)
-Processeur vidéo:DC011,DC012,
-4KO de mémoire écran (RAM),
-un port de communication à 2 modes,Synchrone et Asynchrone,
-Le port de l'imprimante,
-L'interface clavier,
-L'interface du contrôleur RX50,
-Option expansion des capacités:
*Communications étendues,
*Graphisme
*Extension mémoire (64K ou 192K).

La (fig-3.1') représente le schéma de la répartition physique des différentes composantes du module système.

3.2-Description fonctionnelle:

3.2.1-Bus d'adresse et de données du module système:

a- Bus de données et d'adresse du 8088:

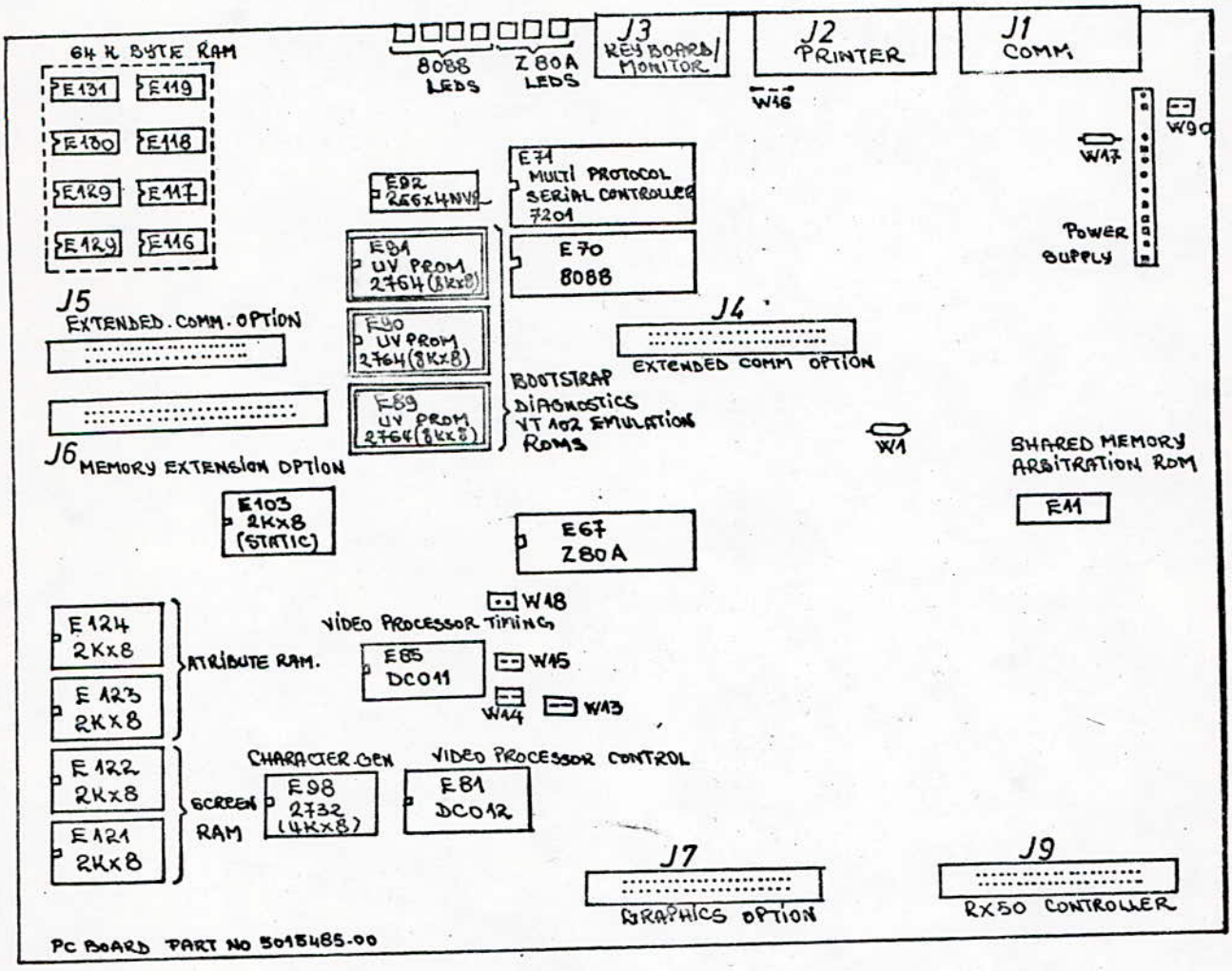
Les bus d'adresse A(19:0) et de données BAD(7:0) entretiennent le moniteur,le clavier,le connecteur de communication ,les options,le vidéo processeur,la logique de la mémoire partagée et la logique de contrôle du 8088.
Le bus d'adresse et de données AD(7:0) connectant le 8088 avec l'adresse latch,le buffer de sortie des données,les 24 KO de ROM et le bus de données est bidirectionnel pour le transfert de données et unidirectionnel pour les cycles d'adresses.

b-Les bus d'adresses et de données du Z80A:

Le bus d'adresses ZA(15:0) et de données ZD(7:0) entretiennent le contrôleur RX50 et la logique de contrôle du Z80A.Le bus d'adresse est unidirectionnel et à 16 bits accédant aux 64 KO de mémoire partagée.Le bus de données ZD(7:0) est un bus bidirectinnel qui transfère les données de ou vers les 64 K de mémoire partagée à travers le transmetteur de données issues de la mémoire partagée du Z80A.

c-Bus de données partagées:

Le bus de données partagées SHD(7:0) est un bus bidirectionnel utilisé par le 8088,le Z80A et l'option de communications étendues pour transférer les données de et vers les 64KO de mémoire partagée.La direction,le passage des bits de données à travers le transmetteur du 8088 et du Z80A est déterminé par des signaux de contrôle issus du 8088



SYSTEM MODULE PHYSICAL LAYOUT

FIG 1.3'

et du Z80A.

3.2.2-Mémoire du module système:

La carte mémoire du 8088 est décrite (fig-3.3.1).La mémoire du module système disponible au 8088 et au Z80A est composée de ROM de RAM et de mémoire RAM non volatile.Les types de mémoire et leur tailles sont les suivantes:

-64KO de mémoire partagée RAM(dynamique)
-256*4 bits de NVM(None Volatil Memory)
-2KO de RAM dédiée au Z80A(statique)
-4KO de RAM d'écran (statique)
-4KO de RAM à attribut (statique)
-Une mémoire additinnelle est disponible pour le 8088 et le Z80A quand un module optionnel d'extension mémoire est installé dans le module système,pouvant être de 64 KO ou 192 KO.

3.2.2.1-Les 64K de mémoire partagée:

Le 8088,le Z80A ou la logique de raffraîchissement peuvent accéder aux 64 KO de mémoire partagée.La logique de rafrîchissement le fait via le bus d'adresse de la mémoire partagée.Le 8088 peut accéder à la totalité des 64K de mémoire,il utilise les deux premiers KO pour stocker les vecteurs d'interruption et d'autres informations qui ne sont pas accessibles au Z80A.Par conséquent,le Z80A ne doit pas accéder aux deux premiers KO de la mémoire partagée.Les adresse du Z80A du premier rang accéderont aux deux KO de la RAM dédiée au Z80A.Le 8088 a pratiquement la même priorité d'accès à la mémoire partagée que le Z80A sauf que si les deux processeurs demandent simultanément accès,auquel cas la logique d'arbitrage donnera la priorité au Z80A.

3.2.2.2-Les 24KO de ROM:

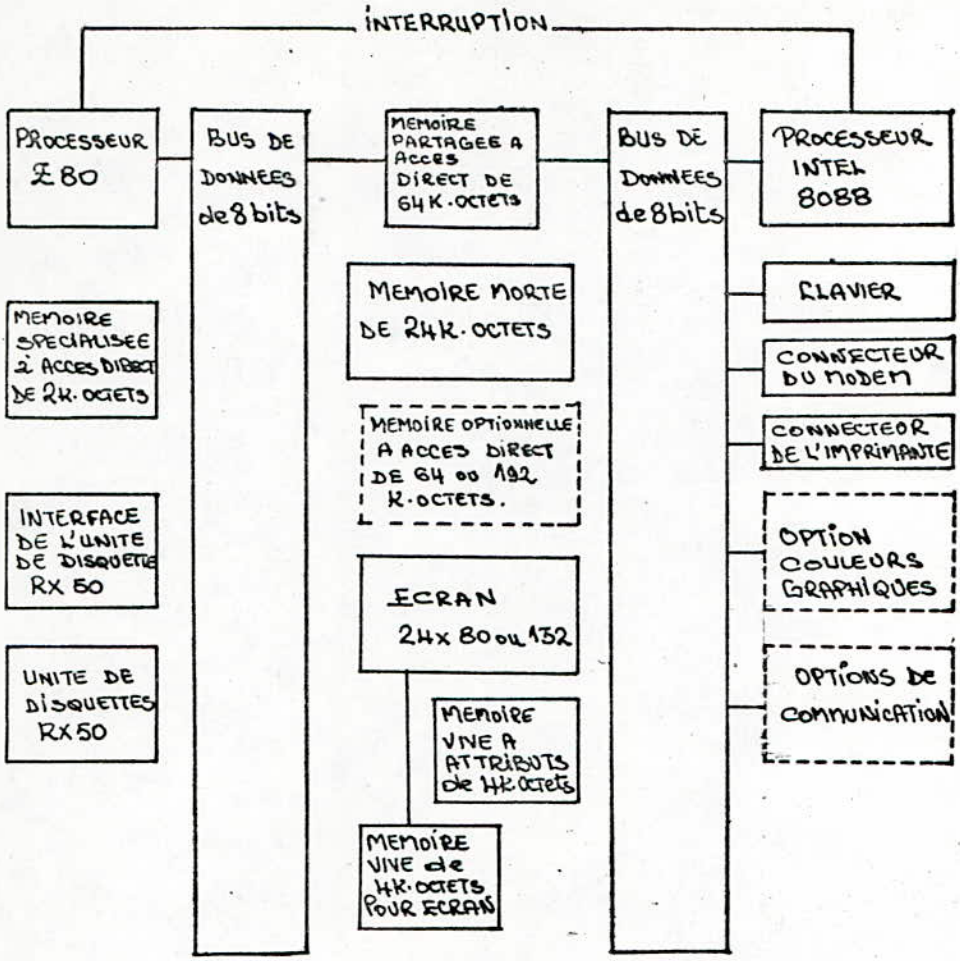
La ROM est partagée en trois zones de 8K et chacune possède une étiquette ROM 0,ROM 1 et ROM 2.Ces ROM contiennent le firmware du système du RAINBOW-100.Le firmware est constitué entre autres du code des processeurs 8088 et Z80A pour le diagnostic,la mise au point et les programmes bootstrap.

Le 8088 accéde aux 24 KO de la ROM via les bits d'adresses A(12:0).La donnée est lue en dehors de la ROM via le bus d'adresse et de données.

3.2.2.3-Mémoire non volatile MNV:

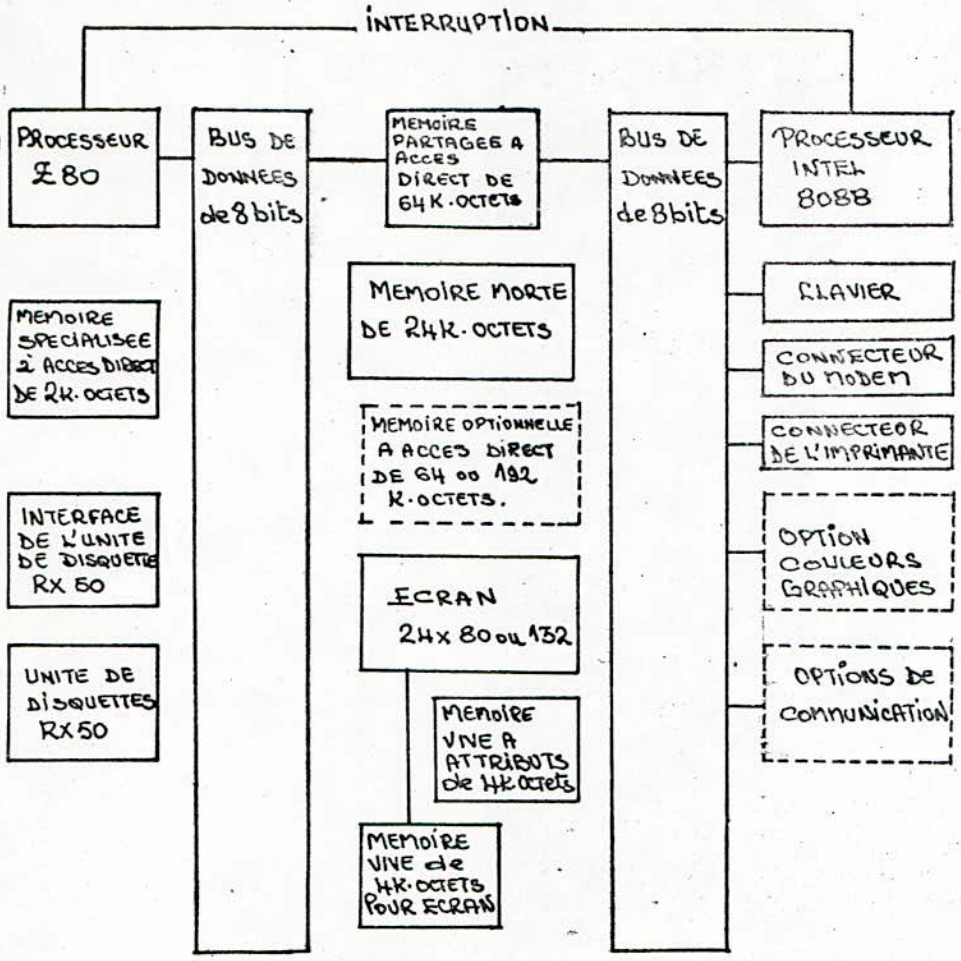
La MNV représente 25*4 bits de mémoire non volatile.Elle est utilisé pour stocker les informations d'établissement du système quand le RAINBOW-100 est mis sous tension.

Les paramètres d'établissement peuvent être changés par écriture de nouveaux pramétres dans la RAM statique,on



SCHEMA FONCTIONNEL DU RAINBOW 100

FIG 3.3.1



SCHEMA FONCTIONNEL DU RAINBOW 100

FIG 3.3.1

stockera alors ces informations dans la NVM via les bits d'adresses A(7:0).

3.2.2.4-RAM réservée au Z80A(2 KO):

C'est une RAM statique accessible par Z80A via les bits d'adresse ZA(10:0).La donnée est écrite dans ou lue de la mémoire via les bits de données ZD(7:0).

3.3.2.5-Les 4KO de RAM d'écran et les 4KO de RAM à attribut:

Ils sont accessibles au 8088 et au processeur vidéo.Le 8088 utilise ces mémoires pour stocker temporairement le caractère et la donnée attribut qui vont être visualisés sur le moniteur.Le processeur vidéo accéde directement à ces mémoire (par DMA) via ses bus d'adresse pour retrouver le caractère ou la donnée attribut,il convertit alors la donnée en un signal vidéo utilisable par le moniteur.

3.3-INTERFACE D'E/S MPSC 7201:

Le MPSC(Multi Protocol Serial Controler) est un circuit intégré de 40 pins dont le schéma bloc est donné (fig-3.4.1).

3.3.1-description des principaux pins pour un canal:

TxD	cette ligne transmet les données en série au canal de communication.
TxC	horloge de transmision de doonées.contrôle! la sortie de la donée sur la pin TxC.
CD	signale que la ligne de transmission est ouverte.
CTS	signale que le module est prêt à recevoir la donnée.
RTS	signale que le canal est prêt à transmettre!
DTR	terminal de données prêt.Ce signal activé,! le MPSC est prêt àrecevoir des données
RxD	cette reçoit la donnée du canal de communication.
RXC	horloge du receveur.Elle gère l'entrée de la donnée sur la pin RxD.

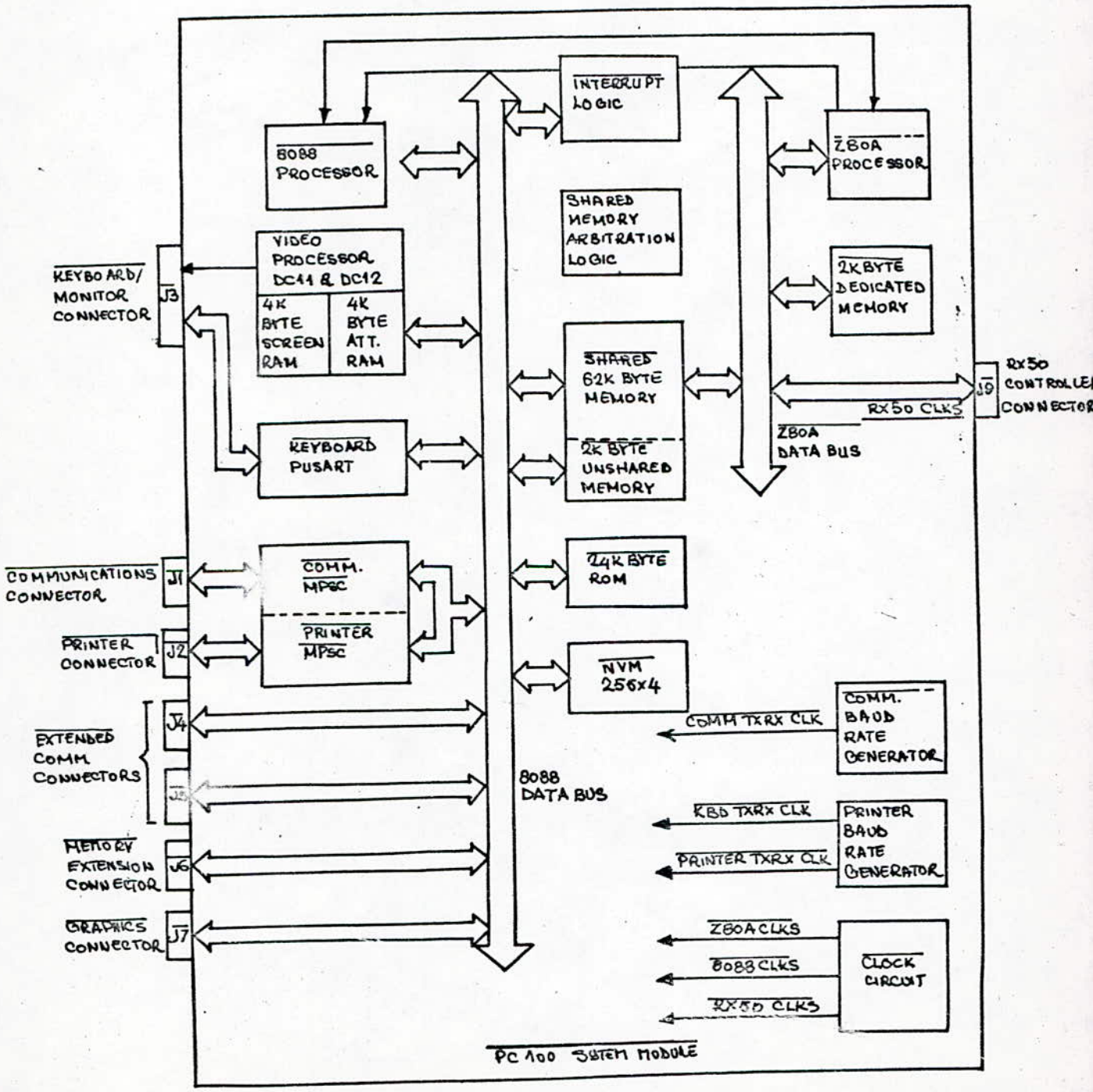

INTERRUPT LOGIC
8088 PROCESSOR
Z80A PROCESSOR
SHARED MEMORY ARBITRATION LOGIC
VIDEO PROCESSOR DC11 & DC12
4K BYTE SCREEN RAM
4K BYTE ATT. RAM
2K BYTE DEDICATED MEMORY
KEYBOARD/ MONITOR CONNECTOR
J3
SHARED 62K BYTE MEMORY
2K BYTE UNSHARED MEMORY
RX50 CONTROLLER CONNECTOR
J9
RX50 CLKS
Z80A DATA BUS
KEYBOARD PUSART
COMMUNICATIONS CONNECTOR
J1
COMM. MPSC
PRINTER MPSC
PRINTER CONNECTOR
J2
14K BYTE ROM
NVM 256x4
EXTENDED COMM CONNECTORS
J4
J5
COMM TXRX CLK
COMM. BAUD RATE GENERATOR
8088 DATA BUS
KBD TXRX CLK
PRINTER TXRX CLK
PRINTER BAUD RATE GENERATOR
MEMORY EXTENSION CONNECTOR
J6
GRAPHICS CONNECTOR
J7
Z80A CLKS
8088 CLKS
RX50 CLKS
CLOCK CIRCUIT
PC 100 SYSTEM MODULE

Fig-3.4-1

3.3.2-Description générale:

Le MPSC est un dispositif de liaison d'un micro-ordinateur avec des périphériques,supportants les protocoles suivants:

-Asynchrone(START/STOP)
-Octet synchrone (Monosynchrone,IBM Bisynchrone)
-bit Synchrone (HLDC d'ISO,SLDC d'IBM)

Il peut être facilement adapté à l'un de ces modes par programmation(SOFT) ou par manipulations matérielles (HARD).
Il accepte également différentes options d'interfaçage avec le microprocesseur:

-Scrutation (Polled mode)
-Attente (WAIT)
-Dirigé par interruption(Interrupt driven)
-Dirigé par DMA.

3.3.3-Description fonctionnelle:

Le MPSC 7201 est composé de deux canaux indépendants(A et B) comportants chacun un transmetteur et un receveur.L'un des deux canaux est utilisé par le RAINBOW-100 pour communiquer avec un autre micrroordinateur (Canal A) avec ou sans MODEM,le canal B pour communiquer avec l'imprimante à travers un connecteur.
En plus des transmetteurs et receveurs,chaque canal comprend des registres à écriture et à lecture qui sont utilisés pour configurer le MPSC,ainsi qu'une logique de contrôle.
Le MPSC,comme il est utilisé dans le RAINBOW-100,utilise deux modes de transfert de données:scrutation et contrôlé par interruption.
Dans le mode scrutation,le 8088 lit périodiquement le registre d'état du MPSC pour déterminer,d'une part le moment ou le caractère est reçu ou transmis,et d'autre part pour détecter les erreurs de transfert de données.Dans le mode contrôlé par interruption,le MPSC provoque l'interruption du 8088 quand un caractère a été reçu ou bien quand il y a un caractère à transmettre,et quand des erreurs de transmission sont détectées.La (fig-3.3.3) est le diagramme de bloc du MPSC.
Le MPSC est relié au 8088 par le bus BAD(7:0).La logique de contrôle de l'interface du système faisant partie du MPSC utilise les sigaux A(1:0),COMM/PTR SEL,RD88 L et WR88 L, d'entrée,issus du 8088 pour communiquer avec les registres internes du MPSC.Chaque canal d'E/S répond à deux adresses d'E/S.
Les informations de commande,de paramétres et d'état sont stockées dans 22 registres internes au MPSC (8 WO,3 RO pour chaque canal).Ces registres sont tous accéssibles par le biais des ports de commande et d'état de chaque canal.Un registre pointeur interne sélectionne les registres de

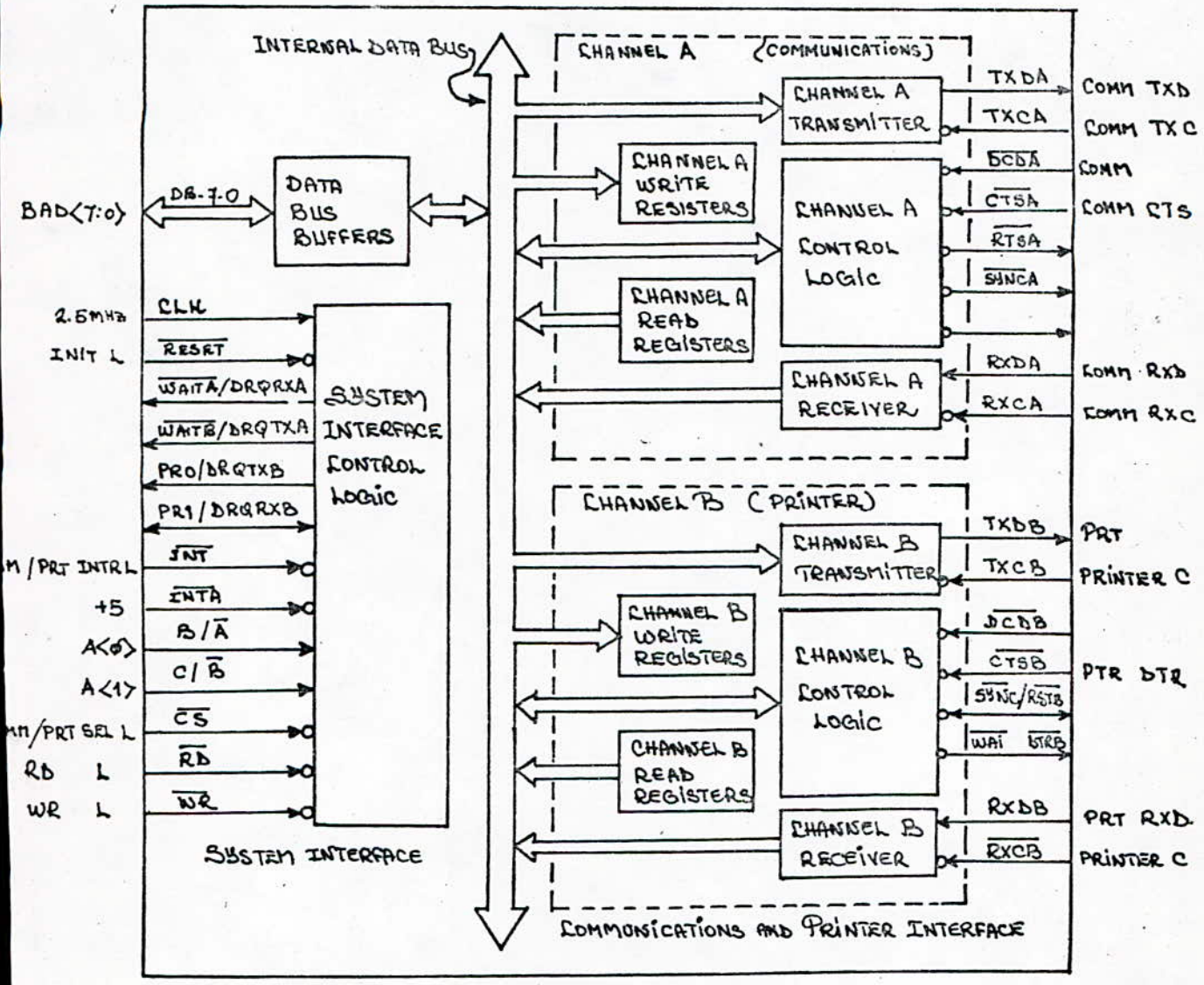

INTERNAL DATA BUS
CHANNEL A (COMMUNICATIONS)
CHANNEL A TRANSMITTER
TXDA
TXCA
COMM TXD
COMM TXC
DCDA
COMM
CTSA
COMM CTS
RTSA
SYNCA
CHANNEL A WRITE REGISTERS
CHANNEL A CONTROL LOGIC
CHANNEL A READ REGISTERS
RXDA
RXCA
COMM RXD
COMM RXC
CHANNEL A RECEIVER
BAD<7:0>
DB-7-0
DATA BUS BUFFERS
2.5MHz
INIT L
CLK
RESET
WAITA/DRQRXA
WAITB/DRQTXA
PRO/DRQTXB
PR1/DRQRXB
SYSTEM INTERFACE CONTROL LOGIC
INT
INTA
B/A
C/B
CS
RD
WR
M/PRT INTRL
+5
A<0>
A<1>
M/PRT SEL L
RD L
WR L
SYSTEM INTERFACE
CHANNEL B (PRINTER)
CHANNEL B TRANSMITTER
TXDB
TXCB
PRT PRINTER C
DCDB
CTSB
SYNC/RSTB
WAI RTRB
PTR DTR
CHANNEL B WRITE REGISTERS
CHANNEL B CONTROL LOGIC
CHANNEL B READ REGISTERS
RXDB
RXCB
PRT RXD
PRINTER C
CHANNEL B RECEIVER
COMMUNICATIONS AND PRINTER INTERFACE

FIG- 3.3.3

commandes ou les registres d'état qui vont être lus durant l'accés au canal du MPSC pour une commande ou une demande d'état.Le schéma bloc (fig-3.4.3) montre l'architecture des registres de commande et d'état pour chaque canal.
Les registres d'état et de contrôle pour chaque canal sont:

- 8 registres de 8 bits (WR0,WR1,...,WR7) à écriture seulement.
- 3 registres (RR0,RR1,RR2) à lecture seulement.

Les 3 bits les moins significatifs de WR0 sont automatiquement chargés dans le registre pointeur à chaque fois qu'on écrit dans WR0.Aprés réinitialisation,WR0 est mis à zéro de manière à ce que la première écriture sur un registre de commande provoque le chargement de la donnée dans WR0.Aprés écriture de WR0,la lecture ou écriture suivante accède au registre sélectionné par le pointeur. Celui-ci est réinitialisé aprés la fin de l'opération d'écriture ou de lecture.Ainsi,toute lecture ou écriture dans un registre quelconque d'un canal du MPSC nécessite deux opérations:

-La première opération est toujours une commande d'écriture pour initialiser le pointeur.
-La seconde peut être une commande d'écriture ou de lecture.

Le registre pointeur préalablement initialisé assure que c'est bien le registre interne fixé par l'opérateur qui est écrit ou lu.Aprés la seconde opération le pointeur est automatiquement mis à zéro (réinitialisé).

Notez qu'écrire dans WR0 ou lire RR0 ne necessite une préinitialisation du registre pointeur.
NOTE:les registres WR6 et WR7 ne sont pas utilisés dans le mode asynchrone.

Le tableau suivant donne les opérations exécutées par le MPSC por les différentes combinaisons des bits d'adresses A(1:0)et des signaux de contôle.

COMM/PTR SEL L	A(1)	A(0)	Lecture RD88 L	Ecriture WD88 L
0	0	0	Lecture d'une donnée du canal A	Ecriture d'une donnée sur le canal A
0	1	0	Lecture de l'état du canal A	Ecriture d'une commande et de paramétres sur le canal A
1	0	1	lecture d'une donnée du canal B	Ecriture d'une donnée sur le canal B
1	1	1	lecture de de l'état du canal B	Ecriture de paramétres sur le canal B
1	x	X	haute impédance	haute impédance

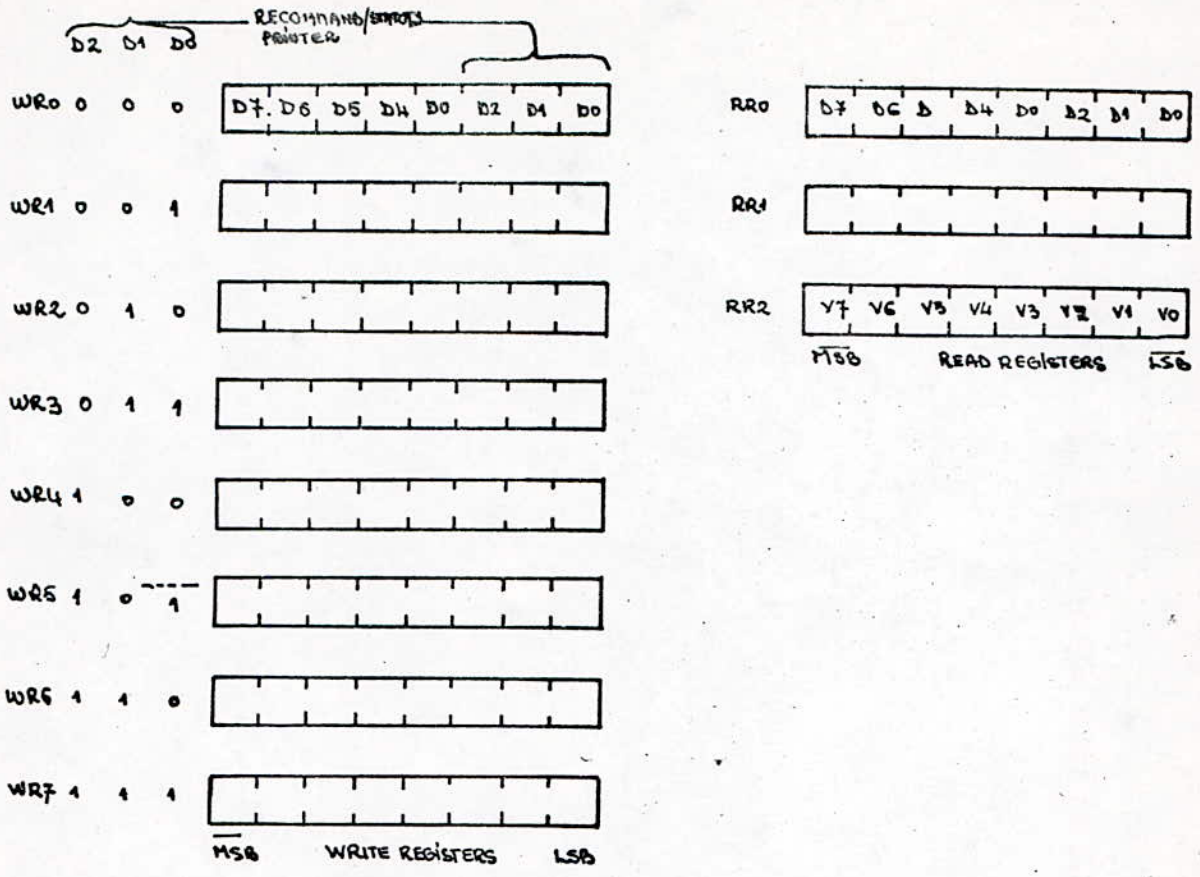

D2 D1 D0
COMMAND/STATUS POINTER
WR0 0 0 0 D7 D6 D5 D4 D3 D2 D1 D0
WR1 0 0 1
WR2 0 1 0
WR3 0 1 1
WR4 1 0 0
WR5 1 0 1
WR6 1 1 0
WR7 1 1 1
MSB WRITE REGISTERS LSB
RR0 D7 D6 D5 D4 D3 D2 D1 D0
RR1
RR2 V7 V6 V5 V4 V3 V2 V1 V0
MSB READ REGISTERS LSB

FIG - 3.4.3

3.3.4-Le canal de communication du MPSC:

Le canal de communication du MPSC est utilisé pour communiquer avec un autre ordinateur à travers le connecteur J2.Ce port peut fonctionner aussi bien en bisynchrone avec un interface physique RS-423 conforme au CCITT.

Les vitesses de transfert en Bauds supportés par ce port sont:

.50	.300	.3600
.75	.600	.4800
.110	.120	.9600
.134,5	.1800	.19200
.150	.2000	
.200	.2400	

Ces différentes vitesses sont obtenues en écrivant un mot de 8 bits dans le registre de vitesse à l'adresse d'E/S: 06H.Les 4 bits les moins significatifs sélectionnent le bit de vitesse de l'horloge de réception (COMM RxC).
Les 4 autres bits sélectionnent le bit de vitesse de l'horloge de transmission(COMM TxC).(Voir-fig ci-dessous).

Les bits de vitesse de transmission et de réception peuvent être sélectionnés par programme d'aprés le tableau suivant:

D(7:4)-D(3:0) en HEXA	Vitesse	D(7:4)-D(3:0) en HEXA	Vitesse
0	50	8	1200
1	75	9	1800
2	110	A	2000
3	134,5	B	2400
4	150	C	3600
5	200	D	4800
6	300	E	9600
7	600	F	19200

3.4-LES DECODEURS D'entrées/sorties:

Le module système contient deux groupes de décodeurs d'E/S.Ceux-ci sont utilisés par le 8088 et le Z80A pour sélectionner et contrôler le transfert de données à travers les décodeurs d'E/S.

3.4.1-Les décodeurs d'E/S du 8088:

Trois décodeurs sont utilisés par le 8088 pour fournir les différents signaux de contrôle de lecture et d'écriture aux différents registres et sélectionner les signaux de contrôle (ou de commande) du contrôleur des ports séries de communication et de l'imprimante,le PUSART du clavier,

l'option graphique et l'option de communications étendues. Les trois décodeurs sont:le décodeur d'écriture (Memory write),le décodeur de lecture (Memory read) et le decodeur de sélection du port.

Les décodeurs d'E/S fournissent les signaux de contrôle aux registres et dispositifs suivants:

-Registre d'état des communications
-Registre de contrôle des communications
-Registre d'écriture et de diagnostic
-Registre d'écriture DC11
-Registre d'écriture DC12
-La logique de détection de défaillances matérielles (MHFU)
-Générateur de vitesse de communications
-Contrôleur série des ports communications/imprimante
-Le PUSART du clavier
-L'option de communications étendues

La figure-3.5.1 est un schéma bloc qui montre les liaisons entre les décodeurs d'E/S,le 8088,les registres et les dispositifs d'E/S.Les décodeurs utilisent les bits d'adresses A(7:1),les signaux de lecture et d'écriture et le signal d'E/S mémoire du 8088 pour contrôler leur fonctionnement.

* Le décodeur de sélection du port d'E/S du 8088 décode les bits d'adresses A(6:4) quand il est validé par le bit d'adresse A(7) et le signal d'E/S mémoire(I/OM).I/OM devient actif(à l'état haut) quand le 8088 accompli un cycle de lecture ou d'écriture d'E/S.Quand les bits d'adresse A(6:4) sont tous actifs (à l'état bas),le signal de validation du décodeur d'E/S devient bas pour activer une des entrées de validation des décodeurs d'écriture et de lecture.Les différentes combinaisons des bits d'adresses A(6:4) déterminent la sortie active sélectionnée par le décodeur de sélection du port d'E/S.

*Le décodeur d'écriture du 8088 décode les bits d'adresse A(3:1) quand il est validé par le signal d'écriture (WR88 L) et le signal de validation du décodeur de sélection du port d'E/S. Les sorties du décodeur d'écriture sont utilisées pour l'écriture de données issues du 8088 dans un registre particulier,ou le générateur de vitesse (voir fig-3.5.1).Quand les bits d'adresees A(3:1) sont actifs (et à l'état bas) le signal (88INTZ) devient bas et réinitialise le flip-flop INT Z80A pour interrompre le Z80A.

*Le décodeur de lecture décode les bits d'adresses A(3:1) quand il est validé par le signal de lecture (RD88 L)et le signal de validation du décodeur d'E/S issu du décodeur de sélection du port d'E/S.Deux sorties du décodeur de lecture sont sollicités par le registre d'état des communications et le registre de lecture de diagnostic pour lire le contenu de ces registres sur le bus BAD(7:0) du 8088.

La troisième sortie CLR88 L est active (à l'état bas) quand tous les d'adresses A(3:1) sont à l'état bas.Le signal

CLR88 L désactive le flip-flop d'interruption du 8088 pour empêcher le Z80A d'interrompre le 8088.

L'information:donnée de contôle ou d'état est lue ou écrite dans les divers registres sous contôle logiciel.

3.5-LE CIRCUIT DE DETECTION DE MHFU (pannes)

Le circuit de détection d'importantes défaillances matérielles(Massiv Hardware Full-Up) placé dans le module système est utilisé pour détécter des situations dans lesquelles le 8088 a perdu la plupart de ses fonctions.

Si le 8088 n'accuse pas réception d'une interruption déclenchée par le processeur vidéo en moins de 108ms,le circuit de détection de MHFU appliquera un signal de réinitialisation au 8088 pendant 108ms.

3.6-CONNECTEURS DU MODULE SYSTEME

Le module système posséde 9 connecteurs de deux types différents.Trois de type-D permettant les connexions externes à des périphériques,à l'imprimante et au clavier-moniteur.Un connecteur permet l'alimentation en courant continu (DC).Les cinq autres connecteurs sont utilisés pour connecter directement le module système au module du contrôleur RX50,au module d'extension mémoire(optionnel),au module de graphisme et à l'option de communicatins.

3.7-LES INTERRUPTIONS DU 8088 (en mode maximum)

Les interruptions du 8088 peuvent être provoquées par logiciel ou par hardware.Les interruptions logicielles ont pour origine l'exécution d'un programme.Les interruptions matérielles proviennent d'une logique externe et classées masquables et non masquables.

Toutes les interruptions ont pour effet le transfert du contrôle à un programme d'interruption.

Les sept interruptions matérielles masquables proviennent des sources suivantes:

-Interruption interprocesseurs du Z80A
-Clavier
-MPSC:canal imprimante ou de communications
-Contrôleur vidéo (DC12)
-Module graphique optionnel
-Module optionnel de communications étendues

L'interruption matérielle non masquable (NMI) est provoquée par le module d'extension mémoire s'il détecte une erreur de parité.La NMI provoquera l'apparition d'un message d'erreur sur l'écran et l'arrêt du 8088.Le 8088 ne pourra ensuite refonctionner qu'en réinitialisant le système.

Les sept interruptions matérielles sont envoyées vers un décodeur de priorité (8 lignes-3 lignes).L'encodeur de priorité envoie un code de niveau de priorité sur 3 bits à

l'encodeur type d'interruption pour un stockage temporaire et en même temps activer le signal INTR H pour interrompre le 8088.Quand le 8088 accépte la demande d'interruption,il active le signal d'accusé de reception de l'interruption (INTA L) pour désactiver la bascule d'interruption et introduire le type d'interruption dont le numéro a été placé sur le bus BAD(7:0) par l'encodeur de type d'interruption.Les bits du type d'interruption sont utilisés par le 8088 comme pointeur d'adresse du vecteur d'interruption.

CHAPITRE 4: LA TRANSMISSION DE DONNEES

4.1-NATURE DES INFORMATIONS A ECHANGER ENTRE LES TERMINAUX.

Pour être acheminée,toute information doit être mise sous forme de symboles.La signification précise de ces symboles est évidemment fondamentale,mais est une pure affaire de conventions entre l'émeteur du message et le destinataire.Toute fois,dans la pratique,chaque organe impliqué dans une telle transmission est susceptible d'être mis en relation entre plusieurs autres; d'autre part,la traduction de l'information devant nécessiter une opération "physique" (même si elle est réalisée par logiciel),on aura tout intérêt à ce que la table de correspondance qui définit la signification des symboles ait une portée générale.

L'information est émise ou reçue par un équipement terminal de traitement de données (ETTD),souvent appelé terminal,mais qui peut être un ordinateur (ce qui est notre cas) ou un terminal ce dernier comportant ou non des fonctions de traitement.

Dans un ETTD,conformément à la figure 6.1 nous distinguons en fait 2 parties qui réalisent des fonctions différentes:la machine de traitement qui peut être source ou collecteur de données,et le contrôleur de communication qui regroupe les organes chargés des fonctions de communication. Ce dernier réalise en particulier la protection contre les erreurs et introduit les éléments (caractères) de service permettant le dialogue entre les deux terminaux.Le contrôleur de communication peut,ou non,constituer un sous ensemble physiquement dissociable des organes de traitement proprement dits.

L'équipement de terminaison du circuit de données (ETCD) est l'organe chargé,en particulier,d'adapter le signal électrique délivré par le terminal au support de transmission.Cette fonction est réalisée le plus souvent par modulation-démodulation d'un signal auxilliaire porteur dans un MODEM.L'ETCD assure de plus des fonctions d'établissement et de libération du circuit.

4.1.1-Codage des informations:

Par définition,le code est la loi de correspondance entre les informations à représenter et les configurations binaires associées,chaque information correspondant à une et une seule configuration binaire.Le codage est l'opération matérielle qui réalise la correspondance.

4.1.1.1-Le code ASCII:

Les insuffisances pour certaines applications des codes à 6 moments (comme le DCB) ont plaidé en faveur d'un code plus riche permettant par exemple le double jeu

majuscule/miniscule.

En 1963,une première version était définie aux états unis et connue sous le nom de code ASCII (American Standard Code for Inforamation Interchange).Les organisations internationales de normalisation se préoccupaient de définir un code universel accépté par toutes les machines et assurant la compatibilité des supports et la possibilité des échanges,devaient en publier une version améliorée qui est maintenant connue sous le nom de code ISO à 7 (ou 8) bits ou alphabet international n°5.

Le code comporte 7 bits utils complétés éventuellement par un élément de parité qui permet la mise en oeuvre de 128 caractères.

4.1.1.2-Problémes de transmission:

Les signaux binaires utilisés par les ordinateurs le sont par des circuits intégrés logiques des divers familles TTL,CMOS selon la rapidité de la machine utilisée.Considérons le cas des circuits TTL qui peut être appliqué aux autres familles logiques:

De tels circuits utilisent une tension d'alimentation unique de 5V et représentent le 1 logique par toute tension comprise entre 2.4V et 5V,et le zéro logique par toute tension comprise entre 0v et 0.8V.Entre 0.8V et 2.4V c'est la "zone interdite",c'est à dire que toute tension se trouvant entre ces limites correspond à un niveau logique indetérminé.Ce peut être un zéro ou un 1 ou une oscillation permanente entre les deux.Pour cette raison,et pour éviter cette incertitude,le passage d'un niveau logique à un autre se fait rapidement (de 20 à 100 ns) afin que le signal concérné ne puisse rester trop longtemps dans la "zone interdite".

Ces contraintes interdisent de véhiculer de tels signaux sur de longues distances car les capacités parasites inévitables dans ce cas,dégradent les fronts de montée et de descente de ces signaux;ils risquent alors de rester trop longtemps dans la fameuse zone interdite ce qui conduit à fonctionnement erratique.

Tant que l'on reste à l'intérieur d'un calculateur,le fait de devoir véhiculer des octets pour transmettre un caractère n'est pas gênant car cela reste sur de courtes distances,et le nombre de fils nécessaires qui se monte à 11 au minimum (1 fil par bit,1 fil de masse et au minimum 2 fils de dialogue) n'est pas un handicap.En utilisant ce procédé,il faudra 11 fils pour relier un ordinateur à n'importe quel terminal,cela devient alors gênant,donc:il faut trouver un autre procédé de transmission: c'est la transmission série.

Principe d'une transmission série:

Le principe de transmission exposé ci-avant,est nommé transmission paralléle.En transmettant les bits d'information les uns aprés les autres,un seul et unique fil est nécessaire:c'est le principe de la transmission série. Voyons maintenant comment est mis en oeuvre ce dernier

principe en étudiant la figure ci-dessous.

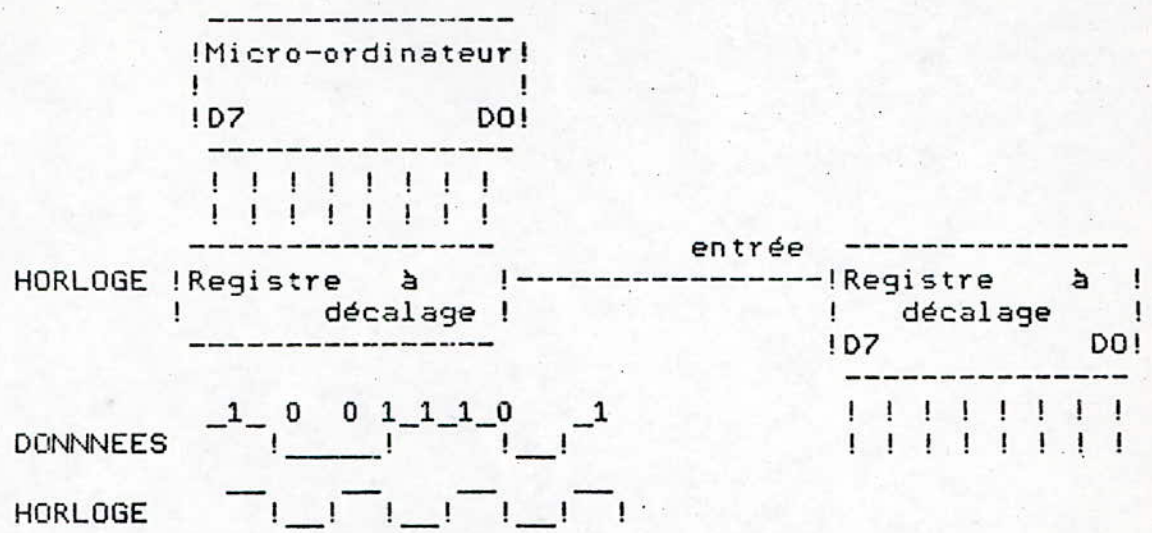


****Schéma de principe d'une transmission série***

Nous y voyons un emetteur de données qui est,par exemple,un ordinateur duquel sortent en paralléle les 8 bits à transmettre.Ceux-ci sont appliqués à un registre à décalage qui reçoit sur son entrée de décalage un signal rectangulaire de fréquence parfaitement stable et connue.Ce signal s'appelle horloge de transmission puisque,du fait du principe même d'un registre à décalage,c'est lui qui fixe la durée d'un bit de sortie du registre et par conséquent,la vitesse de transmission de celui-ci.Nous avons donc par ce moyen convertit nos 8 fils paralléles en une succession de 8 bits voyageant sur un seul fil.

A l'autre extrémité de la liaison,constituée par exemple par un autre ordinateur,un registre à décalage symétrique du précédant reçoit les bits sous forme série et reçoit également un signal rectangulaire de la fréquence rigoureusement identique à celle utilisée à l'emission;ce signal est nommé horloge de réception.On conçoit aisément que si l'horloge d'émission et de réception sont synchronisées,il va être possible,au registre à décalage de réception,de reconstituer l'information paralléle à partir des bits reçus en série.

Pour que le principe que nous venons de décrire puisse fonctinner,il faut assurer un synchronisme rigoureux entre les horloges d'emission et de réception;en effet il suffit d'un petit décalage de l'une par rapport à l'autre pour qu'à la réception on se "trompe" d'un bit et que,de ce fait,le mot de 8 bits reçu soit complètement erroné.Comme il est impossible de réaliser un tel synchronisme parfait,l'horloge est souvent transmise sur un fil séparé ou est mélangé au signal utile avec un codeur spécial dans ce qu'on appele alors les transmissions série synchrones.
Comme cette nécessité conduit à avoir un fil ou de la circuiterie supplémentaire pour mélanger les données et l'horloge,un autre système a été dérivé du précédent:la

principe en étudiant la figure ci-dessous.

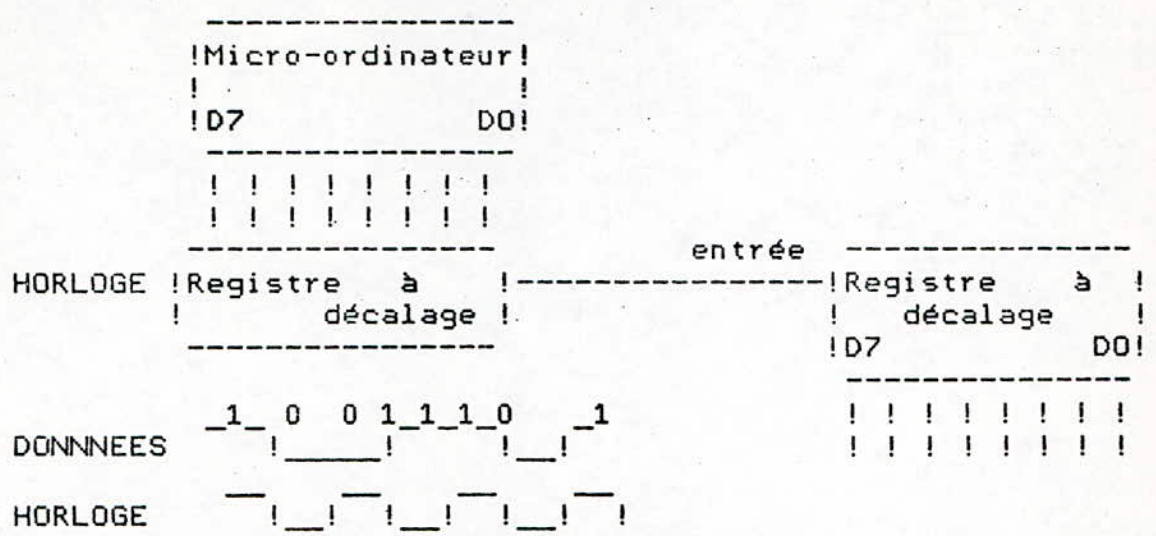


****Schéma de principe d'une transmission série***

Nous y voyons un emetteur de données qui est,par exemple,un ordinateur duquel sortent en paralléle les 8 bits à transmettre.Ceux-ci sont appliqués à un registre à décalage qui reçoit sur son entrée de décalage un signal rectangulaire de fréquence parfaitement stable et connue.Ce signal s'appelle horloge de transmission puisque,du fait du principe même d'un registre à décalage,c'est lui qui fixe la durée d'un bit de sortie du registre et par conséquent,la vitesse de transmission de celui-ci.Nous avons donc par ce moyen convertit nos 8 fils paralléles en une succession de 8 bits voyageant sur un seul fil.

A l'autre extrémité de la liaison,constituée par exemple par un autre ordinateur,un registre à décalage symétrique du précédant reçoit les bits sous forme série et reçoit également un signal rectangulaire de la fréquence rigoureusement identique à celle utilisée à l'emission;ce signal est nommé horloge de réception.On conçoit aisément que si l'horloge d'émission et de réception sont synchronisées,il va être possible,au registre à décalage de récéption,de reconstituer l'information paralléle à partir des bits reçus en série.

Pour que le principe que nous venons de décrire puisse fonctinner,il faut assurer un synchronisme rigoureux entre les horloges d'emission et de réception;en effet il suffit d'un petit décalage de l'une par rapport à l'autre pour qu'à la réception on se "trompe" d'un bit et que,de ce fait,le mot de 8 bits reçu soit complètement erroné.Comme il est impossible de réaliser un tel synchronisme parfait,l'horloge est souvent transmise sur un fil séparé ou est mélangé au signal utile avec un codeur spécial dans ce qu'on appele alors les transmissions série synchrones.
Comme cette nécessité conduit à avoir un fil ou de la circuiterie supplémentaire pour mélanger les données et l'horloge,un autre système a été dérivé du précédent:la

transmission asynchrone.

4.1.1.3-La transmission asynchrone:

Le principe de la transmission asynchrone est plutôt que de synchroniser en permanence les horloges,on synchronise celle-ci pour chaque caractère reçu au moyen d'un bit particulier.

Alors que l'on ne sait pas techniquement réaliser deux horloges parfaitement synchrones,on sait tout de même,à très peu de frais,réaliser deux horloges fonctinnant à la même fréquence avec seulement quelques % d'erreurs.

Lorsque la source de données produit des caractères à des instants aléatoires,il est plus simple de transmettre ces caractères au moment ou la source les délivre sans tenir compte des caractères précédants ou suivants.On a alors une succession de trains de symboles binaires séparés par des intervalles quelconques:ce type de séquence de données est dit asynchrone.Il rend nécessaire l'adjonction,à chaque caractère,d'éléments de repérage permettant la reconnaissance du début et de la fin du caractère et sont souvent désignés par leur appellation anglo-saxone bit de START,bit de STOP.

La durée du START est fixée à celle d'un bit du caractère,celle du STOP n'est définie que quant à sa valeur minimale qui peut être de 1,1.5 ou 2 bits selon le cas.Une transmission de ce type est qualifiée d'arythmique ou START-STOP,mais l'usage est de l'appeler simplement asynchrone.

Les données asynchrones sont typiquement issus de terminaux
lents à des débits inférieur à 1200 bits/seconde.

4.1.4-Circuit de données:

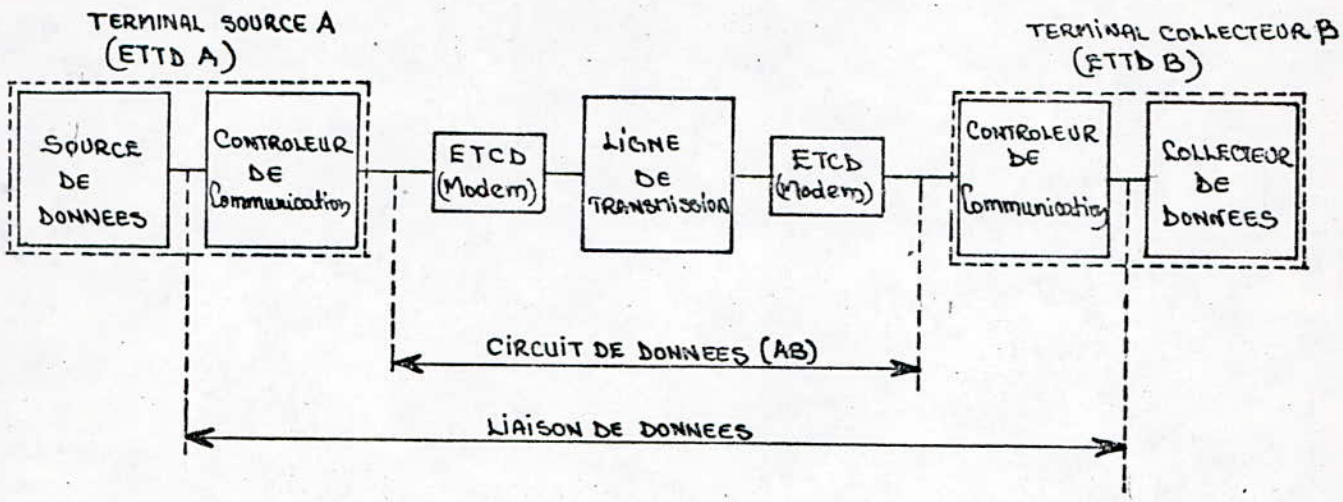
La transmission de données consiste en la transmission éléctronique d'informations codées entre 2 points.Considérons 2 terminaux A et B susceptibles d'être simultanément source et collecteur (fig-1.2.8).Le circuit de données (AB) est l'ensemble des moyens qui permettent l'échange de données entre les terminaux A et B.Un tel circuit comporte la ligne de transmission et les deux ETCD associés.

Dans l'échange d'information on distingue aussi la voie principale qui achemine les données utiles de A vers B par exemple,et la voie secondaire utile pour l'exploitation qui permet de transférer les inforamtions de service (le plus souvent dans le sens inverse B-A) en particulier pour les besoins de correction d'erreur.

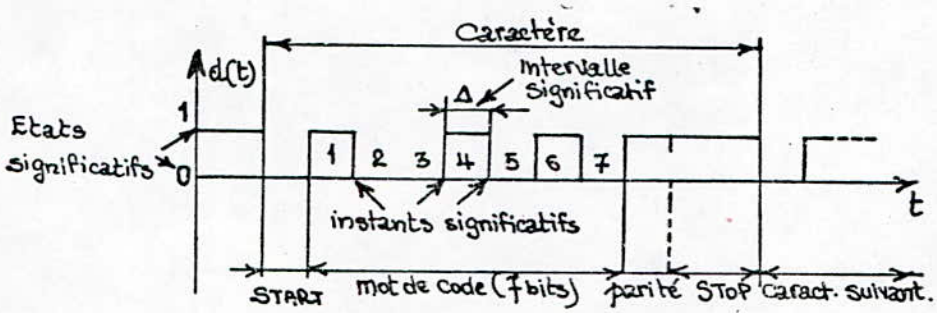
Selon que le circuit est utilisable dans un seul sens,dans les deux sens à l'alternat ou dans les deux sens simultanément,on pourra parler de circuits simplex,semi-duplex ou duplex à l'alternat(halph-duplex),ou enfin duplex intégral(full-duplex).

La figure 1.2.8 schématise les différents modes de transmission sur un circuit de données.

- TRANSMISSION DE DONNEES DE A vers B -



- MESSAGE DE DONNEES ASYNCHRONE -



- DISTORSION BIAISE ET DISTORSION ISOCHRONE -

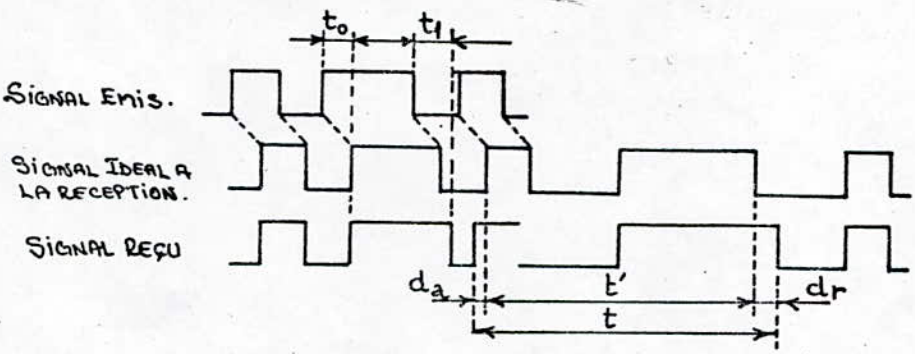


FIG - 1.2.8-C

FIG - 1.2.8

*Caractéristiques d'un circuit de données:

a)Synchronisme:
Dans la transmission asynchrone,la synchronisation des caractères se fait par reconnaissance des signaux de départ(START) et des signaux d'arrêt(STOP) qui ne portent pas d'information et délimitent chaque caractère.La synchronisation des bits est immédiate car on dispose à l'émission et à la réception d'horloges locales de même fréquence nominale.Dans le recepteur,le signal de départ(START) déclenche au début de chaque caractère un générateur de rythme local qui permet l'échantillon- age des symboles binaires contenus dans chaque caractère.

b)Rapidité de modulation.Cadence de transfert:
La rapidité de modulation exprimée en Bauds est notée R.Si DELTA représente la durée exprimée en secondes de l'intervalle significatif séparant 2 instants significatifs successifs,alors (voir fig-1.2.8):

R=1/DELTA (Bauds)

On fait,au moins en partie,abstraction de l'incertitude qui existe sur la donnée entre caractères(STOP).

c)Qualité de la transmission:
Dans le cas hypothétique d'une transmission parfaite,les signaux de données délivrés par l'ETCD à la réception sont identiques au décalage près constitué par le temps de propagation à ceux émis par le terminal source.Dans la pratique,cet idéal est loin d'être atteint.Plusieurs critères sont utilisés pour pouvoir mesurer la qualité d'une liaison numérique.

Le taux d'erreur sur les bits caractérise la qualité de transmissions synchrones c'est à dire,lorsque le terminal délivre par unité de temps un nombre constant de bits.C'est le nombre de bits érronés,reçus pendant un intervalle de temps déterminé rapporté au nombre total de bits transmis pendant cet intervalle de temps.La notion de taux d'erreur s'applique également aux systèmes de transmission asynchrone à condition d'effectuer la mesure avec des signaux de référence synchrones.Cependant,elle ne suffit pas à caractériser la qualité d'une transmission qui dans le cas général doit évaluer avec fidélité le recepteur, reconstitue les transitions du signal asynchrone délivré par la source à des instants quelconques.

On est alors conduit à définir différentes catégories de distorsions pour évaluer les imperfections du signal reçu.

-La distorsion biaise caractérise un défaut relatif à des transitions 0-1 et 1-0 qui seraient affectés de temps de propagations différents.

-La distorsion isochrone est le rapport à l'intevalle T du maximum des valeurs mesurées prise en valeur absolue,entre les intevalles réels qui séparent 2 instants significatifs quelconques et les intervalles théoriques

correspondants.

Cela revient à dire que da et dr,étant les valeurs maximales des distorsions individuelles avance et retard-respectivement négative et positive- la distorsion isochrone @=dr-da=dr+da (voir fig 1.2.8-c)

$$@=\frac{t-t'}{T}=da+dr$$

4.1.5-Procédures d'E/S:

Nous allons étudier dans ce paragraphe les 3 sortes de procédures possibles:

*E/S par DMA
*contrôlées par interruptions
*par demande d'interruption

1)E/S par DMA:c'est une procédure spéciale qui permet de transférer directement des données d'un périphérique vers une mémoire sans passer par l'intermédiaire du microprocesseur.Cette procédure est interessante lorsque de grandes quantités de données doivent être transférées dans des mémoires à accés rapide.

Considérons un microprocesseur effectuant un traitement de données en temps réel,sur un nombre N de mesures relevées périodiquement pendant une durée Tn,le microprocesseur traite les N données acquises pendant le temps Tn-1,mais en même temps arrivent les N mesures suivantes,qu'il faut charger en mémoire en attendant leur traitement au temps Tn+1.

voyons comment une procédure de DMA assure la mise en mémoire des mesures reçues.

a)au lieu d'interrompre le microprocesseur pour chaque nouvelle mesure,on utilise un buffer d'interface ou viennent s'accumuler provisoirement les mesures,ceci,sans intervention du microprocesseur qui traite pendant ce temps les N données précédantes.

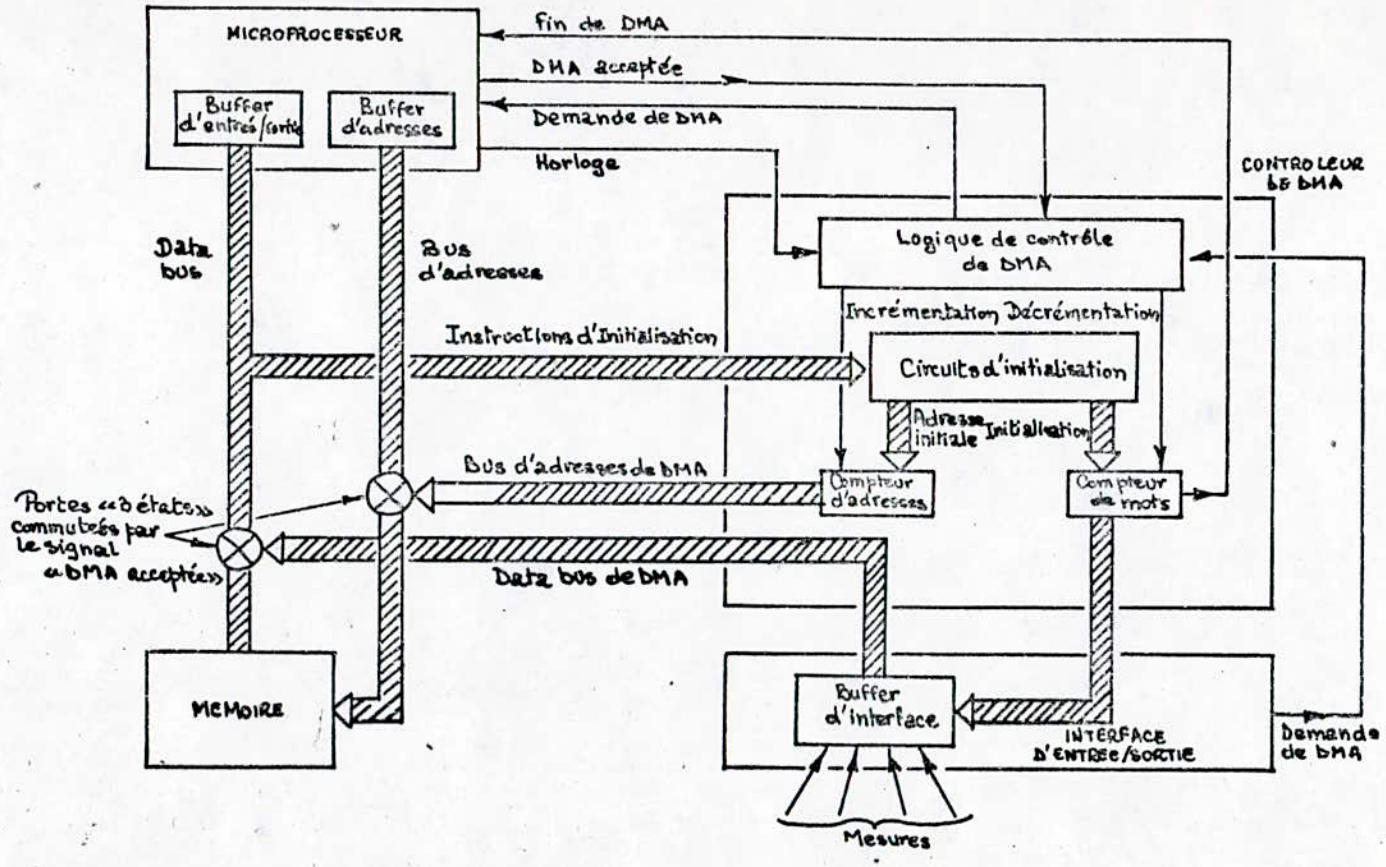
b)lorsque le buffer est plein,l'interface du périphérique envoie un signal de demande de DMA au contrôleur de DMA,qui transmet la demande au microprocesseur.

c)le microprocesseur termine l'instruction en cours.Il s'agit ensuite d'effectuer une double initialisation pour préparer le transfert par DMA.

-initialisation d'un compteur d'adresse de DMA ou est chargée l'adresse initiale à laquelle sera rangé en mémoire le premier mot transféré.

-initialisation d'un compteur de mots de DMA,ou est chargé le nombre de mots à transférer.

Sur la figure 6.1.5 nous avons considéré le cas ou C'est le microprocesseur qui effectue cette



SCHEMA DE PRINCIPE d'un DISPOSITIF de DMA (Accès direct Mémoire)

initialisation.Ensuite le microprocesseur envoie le signal "DMA acceptée":il est alors déconnecté du bus de données et du bus d'adresse,par exemple en commutant des portes à 3 états qui placent le bus en état haute impédance.Le contrôle du bus passe alors au contrôleur de DMA.

d)le contrôleur de DMA gère alors le transfert de données en mémoire:le premier est rangé à l'adresse initiale contenue dans le compteur d'adresses.Puis celui-ci est incrémenté d'une unité, et le contrôleur de mot décrémenté d'une unité.La même opération recommence,et l'itération continue jusqu'à ce que le compteur de mots soit à zéro;celà provoque l'envoi au microprocesseur du signal de fin de DMA.Le microprocesseur reprend alors le contrôle.

2)E/S par programme:

Cette procédure très simple est assez rarement employée,car elle suppose que le périphérique concerné est toujours prêt au moment ou on l'appelle.Le microprocesseur effectue simplement une instruction d'entrée (INPUTm) ou de sortie (OUTPUTm) dans le programme en cours.L'instruction OUTPUT,par exemple,charge le contenu de l'accumulateur dans le buffer d'interface n°m ceci est valable uniquement pour un périphérique simple.Il existe une variante de cette méthode,l'E/S conditionnel:le microprocesseur, avant d'effectuer une opération d'E/S,test si le périphérique est prêt à effectuer un transfert;tant que le périphérique n'est pas prêt le microprocesseur effectue une boucle d'attente.On voit que cette procédure pénalise la vitesse du microprocesseur puisqu'elle l'oblige à "attendre" le périphérique.

3)E/S par demande d'interruption:

Pour effectuer correctement un échange avec un périphérique, le microprocesseur doit savoir si ce périphérique est disponible. On voit que si le microprocesseur effectue systématiquement ce test par programme,il est concidérablement ralenti.Dés lors il apparaît plus rationnel que le périphérique lui-même signale au microprocesseur est disponible:c'est le principe de la demande d'interruption.

Un microprocesseur ne dispose que d'une entrée INTERRUPT.Or les microordinateurs comportent généralement plusieurs périphériques,les voies de demande d'interruption des différents périphériques doivent être "OU-câblés" sur cette entrée INTERRUPT.Il se pose alors un double problème:

-Le microprocesseur doit pouvoir identifier le périphérique demandeur.

-Dans le cas de plusieurs demandes d'interruption simultanées, il doit savoir dans quelle ordre de priorité il faut desservir les périphériques.

Deux méthodes sont couremment employées dans ce but:

a)Scrutation des périphériques:

Plusieurs périphériques pouvant demander simultanément une interruption,il est nécessaire de pouvoir mémoriser ces demandes tant que le microprocesseur ne les a pas toutes acquitées.Dans ce but,les interfaces des périphériques disposent de bascules d'état (flags) validées lors d'une interruption (fig-3-a).La méthode de scrutation des périphériques consiste en le branchement du microprocesseur,dés qu'il reçoit le signal INTERRUPT,à un programme de scrutation qui teste successivement les bascules d'état des périphériques.Une fois trouvé le périphérique demandeur,il effectue le programme spécifique de gestion de ce périphérique.

Cette procédure est donc d'un emploi assez simple;elle est couremment utilisée.Remarquons néanmoins qu'elle est assez lourde et qu'elle peut faire perdre beaucoup de temps:en effet dans le cas le plus défavorable,le microprocesseur doit tester tout les périphériques avant de détecter le périphérique demandeur.

b)Interruptions dirigées ou véctorisées ("Vectored interrupts")

Voir chapitre 1.1.3.1

4.2-LA NORME RS 232:

La norme RS-232 est apparue aux USA dans les années soixante.Après plusieurs années de travail,l'EIA (Electronic Industrie Association) publie en 1969 la norme RS 232C.

RS :Recommanded Standard.
232:Le numéro de la norme.
C :3éme version de la norme.

La norme définit les caractéristiques physiques d'un connecteur,le nombre de fils(25 fils numérotés de 1 à 25),les fonctions de chacun de ces fils,et la nature des signaux éléctriques.En microinformatique on n'utilise que 8 ou 9 fils.

Le moyen le plus simple de réaliser une transmission de données entre deux ordinateurs disposants chacun d'un interface série (RS 232) est d'utiliser un câble à 2 fils,l'un pour transmettre des données et l'autre représentant la référence éléctrique (masse).Il est bien entendu que ce transfert est fait sous contrôle d'un logiciel de communication.

La norme RS 232 a défini le fil n°2 comme étant le fil de transmission et le n°7 comme la masse.

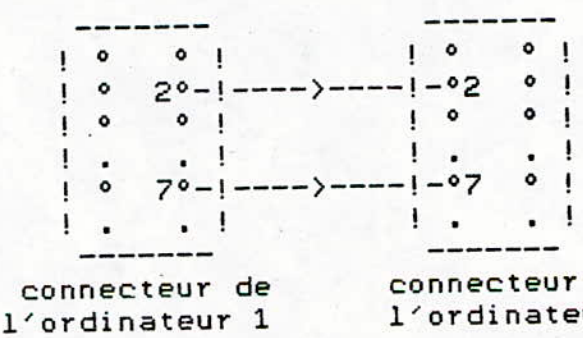


fig-a-"Câble théorique de transmission".

Dans ce cas la transmission est simplex c'est à dire que les données sont transférées dans un seul sens.

Si l'on veut transmettre des données dans les 2 sens (transmission halph ou full-duplex) on doit définir un fil de transmission et un autre de réception.En halph-duplex,la transmission se fait dans les 2 sens mais alternativement.La norme RS 232 définit le fil n°3 comme fil de "réception de données".

on verra le rôle des autres broches dans la partie "signaux du connecteur de communication".

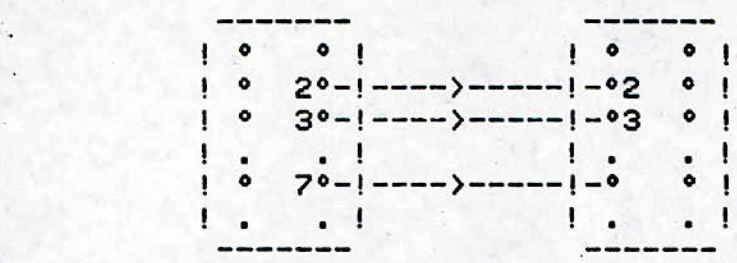
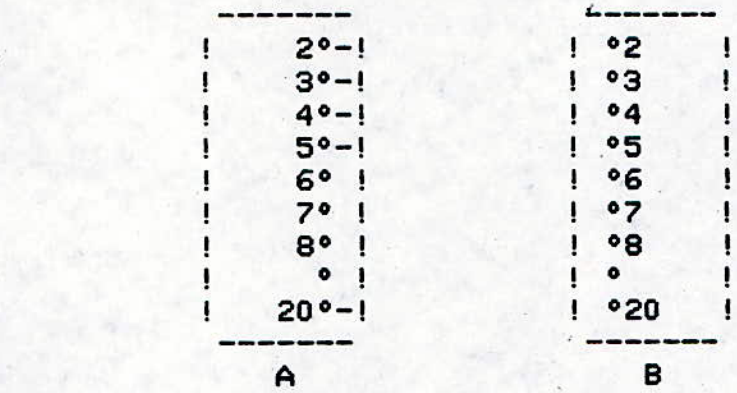


fig-b-Câble théorique de transmission full-dup.

Pour connecter 2 ordinateurs sans MODEM,on utilise des câbles disposés comme suit;



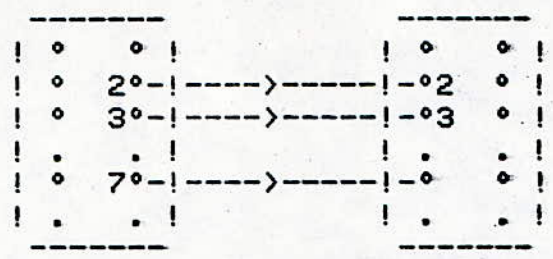
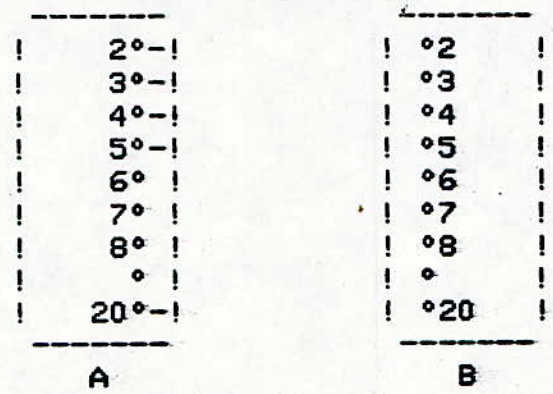


fig-b-Câble théorique de transmission full-dup.

Pour connecter 2 ordinateurs sans MODEM,on utilise des câbles disposés comme suit:



4.3-MODIFICATION DES PARAMETRES DE COMMUNICATION PAR LE MODE DE FONCTIONNEMENT DU RAINBOW 100.

A)Modification des bits de données et de la parité:

Ce choix détermine deux caractéristiques de communication séparés,mais néanmoins liés:les bits de données et de parité.Les informations transmises sur la ligne de communication sont conformes à une configuration de données.Celle-ci commence par un bit de début,utilise 7 bits de données (les états unis) ou 8 bits de données (en Europe),ajoute 1 bit de parité et termine par 1 ou 2 bits d'arrêt.Ce paramétre détermine si l'ordinateur doit utiliser 7 ou 8 bits de données pour chaque caractère,ainsi que le type de parité.

CARACTERES AFFICHES	BITS DE DONNEES PAR CARACTERE	TYPE DE PARITE PENDANT LA TRANSMISSION	ACTION SUR LA PARITE A LA RECEPTION
7O	7	impaire	vérifiée
7E	7	paire	vérifiée
7N	7	pas de parité	ignorée
8O	8	impaire	vérifiée
8E	8	paire	vérifiée
8N	8	pas de parité	ignorée

B)Modification de la vitesse de transmission et de réception:

La modification de la vitesse de transmission et de réception est faite en passant dans le mode de fonctionnement.

Le réglage de la vitesse de transmission doit correspondre à la vitesse de réception de l'autre ordinateur.

Les vitesses disponibles sont:
50;75;110;135,5;150;200;300;1200;1800;2400;3600;4800;9600;19200

4.4-PROTOCOLES DE COMMUNICATION.

Le RAINBOW 100 peut utiliser un MODEM bidirectionnel simultané (full-duplex).Les régles de communication,les signaux employés et leur intéprétaion constituent le protocole de communication.
Trois différents protocoles sont disponibles sur le RAINBOW 100: FDXA,FDXB et FDXC.

Nous avons utilisé le FDXA qui permet au RAINBOW 100 de communiquer sans MODEM.L'ordinateur est prêt à transmettre et à recevoir dés qu'il est sous tension ou en ligne.Une

fois sous tension,le RAINBOW 100 active "terminal de données prêt" et demande pour emettre.

FDXA:communication en bidirectionnel simultané avec liaison directe avec un ordinateur à distance ou un MODEM n'utilisant pas de signaux de contrôle.

Signaux du connecteur de communication du RAINBOW 100

Broche	Désignation	Description
1	terre de protection (PROT GND)	Chassis à la terre
2	emission de données ED (TXD)	données transmises par le Rainbow 100; en position de repos (état haut) quand il n'y a pas de transmission
3	réception de données RD (RXD)	caractères reçus de l'ordinateur à distance
4	demande pour emettre DPE (RTS)	activé si le Rainbow 100 est en ligne,et désactivé s'il est déconnecté
5	prêt à emettre PAE (CTS)	indique que le MODEM est prêt à transmettre
6	poste de données PDP (DSR)	indique que le MODEM est en mode de données
7	terre de signalisa- TS (SGND)	terre commune aux circuits internes (sauf terre de protection)
8	terminal de données TDP (DTR)	ce signal activé,le Rainbow 100 est prêt à recevoir des données ; sinon le MODEM se déconnecte et ne répond plus aux appels

CHAPITRE 5: REALISATION DE LA TRANSMISSION OLIVETTI(M24) =-=-=-=-=-=- OLIVETTI(M24) ET RAINBOW_100-OLIVETTI(M24)

INTRODUCTION.

Au début de l'ère de l'informatique,les problèmes de communications n'étaient pas abordés,dans la mesure ou toutes les données étaient acquises sur place.Puis au fûr et à mesure que la quantité d'informations traitées par l'ordinateur augmentait,et que le temps devenait un élément prépondérant,et dans un soucis de décentralisation des réseaux de transmission de données de plus en plus complexes ont vu le jour.Ces réseaux, constitués aussi bien d'ordinateurs que de terminaux,permettent un accés aisé à toutes les informations disponibles dans le réseau.

Notre but était de contribuer à la réalisation d'un mini-réseau de microordinateurs mais vu les moyens dont dispose le centre de calcul,on a été contraint à envisager des transferts de fichiers entre deux microordinateurs.En effet,chaque micro- ordinateur ne disposant que d'un seul connecteur d'E/S série, il est impossible de concevoir un réseau composé de plus de deux microordinateurs.

Notre choix s'est porté sur la réalisation d'une liaison M24-RAINBOW 100 et M24-M24.

Deux manières d'aborder ce problème se présentèrent à nous.Programmer le port d'E/S (MPSC) directrment,ou bien utiliser les ressouces du système d'exploitation qui se chargerait de la programmation de ce même port.On a opté pour la deuxième méthode qui nous paru plus simple.

5.1-POSSIBILITES OFFERTES PAR CP/M.

Une fois notre choix fait,on s'est mis à chercher les facilités ou les ressources qu'offrait CP/M pour les E/S.

Deux commandes particulières ont attiré notre attention: PIP et STAT (déjà décrites dans le chapitre consacré à CP/M).

PIP,programme interpériphérique,permet la manipulation des fichiers sur disque (concaténations,copies),permet aussi l'envoi de fichiers sur les différentes unités physiques tels que l'imprimante et le port d'E/S.

STAT,outre sa fonction d'afficher des informations concernant les fichiers sur disques,permet l'assignation d'unités.STAT DEV affiche les différentes assignations possibles pour chaque unité et STAT VAL permet d'afficher les assignations en cours.L'ordre STAT permet de modifier ces assignations.Ces modifications restent valables jusqu'à ce que d'autres assignations occurent ou après réinitialisation du système.En effet, le système adopte alors des assignations par défaut.

La combinaison de PIP et STAT permet l'envoi d'un fichier sur disque vers n'importe quel unité physique du

système.PIP dans ce cas a pour rôle de lire une copie du fichier dans la TPA, puis ce dernier est acheminé vers l'unité spécifiée dans la ligne de commande tout en tenant compte des assignations courantes effectués par STAT.Par exemple:

```
A> STAT AXO:=LPT: (RETURN)
A> PIP  AXO:=B:FICH.TXT (RETURN)
```

La première ligne de commande assigne AXO(port de communication, en sortie uniquement) à LPT(interface de l'imprimante, en sortie uniquement).La deuxième ligne envoie le fichier FICH.TXT vers le port de communication, donc vers l'imprimante(grâce à l'assignation effectué par STAT).On peut également effectuer un transfert de fichiers entre deux RAINBOWS-100 en utilisant uniquement la commande PIP et les noms logiques spécifiant que le port de communication est en entrée sur l'un des microordina- teurs, en sortie sur l'autre:AXI et AXO respectivement.

Sur l'un des RAINBOWS on entrera la commande:

```
A> PIP E:PROG1.TXT=AXI: (RETURN)
```

Après avoir entré la ligne de commande ci-dessus,le RAINBOW se met en réception et va lire l'unité AXI pour introduire les données envoyées par l'autre micro.Le programme PIP prend fin dés la réception du caractère de fin de fichier (EOF=CTRL-Z).
Une fois EOF reçu,le fichier PROG1.TXT est fermé et enregistré sur l'unité spécifiée(E).

Sur l'autre micro on introduira:

```
A> PIP AXO:=PROG.TXT (RETURN)
```

Le fichier PROG.TXT est envoyé vers le port de communication à travers le câble RS-232.

Exemple récapitulatif :

TRANSFERT DE FICHIERS ENTRE 2 RAINBOW 100
****

*Sur le micro receveur on lance la commande:

```
A> PIP AXI:=E:FILENAME
```

*Sur le micro transmetteur on exécute le programme:

```
CSEG
ORG 100H
MOV CL,04H  ;appel de la fonction d'envoi vers
            ;le port de communication
MOV DL,1AH  ;caractère à envoyer CTRL-Z=1AH
CALL BDOS   ;appel de BDOS
MOV CL,00H  ;appel de la fonction Retour au
            ;système
```

```
        CALL BDOS
BDOS:   INT 224
        RET
```

On prendra soin de vérifier que les paramétres de communication (vitesse,parité,nbre de bits de stop,nbre de bits par caractère) sont identiques sur les deux micros.

Donc,on peut dire d'emblée,que le problème de transfert de fichiers ASCII entre deux RAINBOWS-100 est résolu.

Voyons maintenant ce que peut nous offrir MS-DOS.

5.2-POSSIBILITES OFFERTES PAR MS-DOS.

Sous MS-DOS, la commande COPY offre, apparemment, les mêmes possibilités que PIP sous CP/M.En effet,avec COPY on peut effectuer des opérations similaires à celles possibles avec PIP.
Par exemple : " A> COPY CON: FILENAME " permet de créer le fichier filename à partir des caractères entrés au clavier. En terminant le texte avec un CTRL-Z suivit d'un RETURN,le fichier sera fermé puis stocké sur dique.
Une autre forme de la commande COPY a attiré notre attention:

```
A> COPY filename PRN
```

Cette commande permet l'impression du fichier filename.
En suivant cette logique,il nous part possible d'atteindre le port de communication (AUX) à l'aide de la commande COPY.
Pour ce faire,deux étapes sont nécessaires:

- -initialisation du port de communication.
- -envoi du fichier vers le port.

La commande permettant d'initialiser le port est "MODE", sa syntaxe est:"MODE COM1:vitesse,parité,nbre de bits par caractère, nbre de bits de stop".
La commande permettant l'envoi sur le port est COPY.
En principe,deux lignes de commandes sont suffisantes pour l'acomplissement d'untransfert de fichiers.Ceux-sont:

```
A> MODE COM1:9600,E,7,1 (RETURN)
   COM1:9600,E,7,1
A> COPY filename AUX: (RETURN)
```

Mais le système retourne alors, le message d'erreur suivant:

```
Périphérique AUX non prêt en écriture.
```

Ce message signifie que le périphérique AUX n'est pas disponible pour l'opération d'écriture demandée.
Après plusieurs essais vains,l'idée de concevoir un

programme s'imposait.

5.3-APPLICATION DU MS-GWBASIC DANS LA MISE AU POINT DU DU PROGRAMME DE TRANSFERT DE FICHIERS RAINBOW-OLIVETTI ET OLIVETTI-RAINBOW.

L'assembleur sur l'olivetti n'étant pas disponible au centre de calcul,on opta pour l'utilisation du language BASIC.
Le MS-GWBASIC (sur le M24) permet une programmation aisée du port de communication.

a)Transfert de fichiers ASCII par le procédé séquentiel:

Avant de s'attaquer directement au problème de la communication, on a d'abord essayer de simuler un transfert de fichiers en créant par programme un fichier source, puis accéder de manière sélective aux données de ce fichier, et les transférer dans un fichier destination qu'on aura ainsi créé.Une fois passée cette étape,on a essayé d'envoyer les données auquels on a accédé vers l'imprimante.La dernière étape fut d'envoyer un fichier vers le port de communication.
Les programmes suivants illustrent cette progression.
Le programme PRG1 permet la création du fichier"DONNEES" (ligne 10),les lignes 20,40 et 50 permettent d'introduire le nom,le département et la date d'embauche de chaque ouvrier d'une certaine entreprise.La ligne 60 permet d'écrire ces informations dans le fichier "DONNEES".

PROGRAMME PRG1
****

```
10 OPEN "O",#1,"DONNEES"
20  INPUT "NOM";N$
30 IF N$="FIN" THEN END
40 INPUT "DEPARTEMENT";D$
50 INPUT "DATE D'EMBAUCHE";H$
60 PRINT #1,N$;",";D$;",";H$
```

Le programme PRG3 accède de manière sélective au fichier "DONNEES".Chaque fois qu'il trouve un employé appartenant au département "A",il retire son nom ainsi que les diveres informations le concernant et les place dans le fichier "DEPT-A".Quand le programme sera terminé, on aura créé le fichier "DEPT-A" à partir du fichier "DONNEES",regroupant tout les noms d'employés appartenants au département "A" ainsi que les informations les concernant.

PROGRAMME PRG3
****

```
10 OPEN "I",#1,"DONNEES"
20 OPEN "A",#2,"DEPT-A"
```

```
30 IF EOF(1) THEN END
40 INPUT#1,N$,D$,H$
50 IF D$="A" THEN PRINT#2,N$;",";H$
60 GOTO 20
```

Le programme PRG4 permet l'impression du fichier "DONNEES".

```
                PROGRAMME PRG4
                **************
10 OPEN "I",#1,"DONNEES"
20 OPEN "LPT1:" FOR OUTPUT AS#2
30 WHILE NOT EOF(1)
40 INPUT#1,N$,D$,H$
50 IF D$="A" THEN PRINT#2,N$,D$,H$
60 WEND
```

Le programme PRG5 permet d'envoyer le fichier "DONNEES" vers le port de communication.

```
        PROGRAMME PRG5
        **************
 5 REM "ouverture du fichier DONNEES"
10 OPEN "I",#1,"DONNEES"
15 REM "ouverture en sortie du port de communication en"
16 REM "mode ASCII"
20 OPEN "COM1:1200,E,7,1,RS,CS,DS,LF,ASC" FOR OUTPUT AS#2
25 REM "test de la fin de fichier"
30 WHILE NOT EOF(1)
35 REM  "Lecture des données N$,D$ et H$ du fichier"
36 REM  "DONNEES"
40 INPUT#1,N$,D$,H$
45 REM "Envoi des données vers le port de communication"
50 IF D$="A" THEN PRINT#2,N$,D$,H$
60 WEND
```

Pour tester le bon fonctionnement de ce dernier programme,nous avons relié le M-24 au RAINBOW 100.Sur ce dernier, nous avons lancé la commande "PIP CON:=AXI:" qui permet d'envoyer tout caractère présent sur le port de communication vers l'écran. Sur le M-24 nous avons lancé PRG5.Ceci étant fait,le fichier "DONNEES" commençait à s'imprimer caractère par caractère sur l'écran du RAINBOW 100.

Cette étape fut le moyen de vérifier que les données étaient bien envoyées.L'étape suivente fut d'écrire un programme de réception sur le M-24(soit PRG6).

Le programme "PRG6" permet la réception des données transmises et leur stockage dans un fichier de données:

```
        PROGRAMME PRG6
        **************
 5 REM "ouverture du port de communication en entrée  et
 6 REM "en mode ASCII"
10 OPEN "COM1:1200,E,7,1,RS,CS,DS,LF,ASC" FOR INPUT AS#2
15 REM"Ouverture du fichier de données"
20 OPEN "A",#1,"Nom de fichier"
25 REM"Test du caractère de fin de fichier"
```

```
30 WHILE NOT A$=CHR$(26)
35  REM"Lecture  d'un caractère présent sur le  port  de
36 REM "communication"
40 A$=INPUT$(1,2)
45  REM  "écriture  du  caractère  dans  le  fichier  de
50  REM "données"
60 PRINT#1,A$
70 WEND
```

Ces deux programmes permettent donc l'émission et la réception de fichiers d'un microordinateur vers l'autre. Ces deux programmes,constitueront les deux sous programmes utilisés par le programme final dans les modes emission et réception.

Dans le but de rendre la tâche plus facile,nous avons introduit un troisième mode:"le mode obéir".De ce fait,une petite modification du "mode réception" est nécessaire.Cette modification nous a conduit à changer le "mode réception" en "mode acquisition" (ou appel).

Avant ces modifications,pour effectuer un transfert de fichiers,il était de sélectionner le "mode émission" sur le microordinateur émetteur et le "mode réception" sur le microordinateur récepteur.

Mais maintenant,en sélectionnant le mode obéir sur l'un des microordinateurs,on pourra effectuer des transferts sans avoir à entrer les noms des fichiers sur les deux machines à la fois.

Le microordinateur sur lequelle on sélectionne le mode obéir,sera appelé "SERVEUR" ou "ESCLAVE".Le serveur effectue une scrutation de son port de communication pour recevoir les commandes et les executent aussitôt.

Pour transmettre un fichier au serveur,il faut tout d'abord lui envoyer un caractère spécifique lui indiquant de se placer en réception en plus du nom du fichier qu'il doit recevoir.Quand le serveur reçoit le signal,il lit le nom du fichier sur le port,ouvre l'ouvre et se met en attente du caractère "XON" (CHR$(17)).XON permet au serveur d'entammer la récupération des données et stockage dans le fichier qu'il aura déjà ouvert.Le fichier de données restera ouvert jusqu'à réception du caractère de fin de fichier(EOF CTRL-Z)

Une fois reçu ce caractère,le fichier est fermé,la réception est achevée et la scrutation du port d'E/S reprend en vue de satisfaire une autre demande.

Pour acquérir un fichier à partir du serveur,il faut sélectionner le "mode appel" sur le micro demandeur.En sélectionnant ce mode,la requête est signalée au serveur à l'aide du caractère STX$ suivie du nom du fichier.Après quoi,le demandeur se met en réception.Quand le serveur reçoit STX$,la scrutation du port cesse.Le serveur lit alors le nom du fichier sur le port.Une fois ce dernier acquis il le cherche dans le répertoire:s'il ne le trouve pas,le caractère CHR$(1) est envoyé au demandeur spécifiant que le fichier n'existe pas sur l'unité indiquée.Si le fichier est trouvé,il est envoyé au demandeur.Quand l'envoi est terminé, le serveur se remet à scruter le port.

Le demandeur qui était alors en réception,s'il reçoit CHR$(1),le massage "FICHIER NON TROUVE" est affiché sur l'écran du microordinateur demandeur.Si par contre,il reçoit XON,les données seront introduites dans le fichier jusqu'à réception de EOF.Quand EOF est reçu,le fichier est fermé et sauvegardé sur disquette.

Enfin nous avons réunis ces programmes en un seul en donnant à l'utilisateur le choix entre les trois modes:transmission, appel et obéir.

b)Transfert de fichiers binaires:

Ici aussi on a utilisé une simulation de transfert consistant en la création d'une copie d'un fichier binaire par programme.Nous n'avions,jusqu'à présent,manipulé que des fichiers ASCII (en mode séquentiel).L'accèes aux fichiers binaires, n'est possible qu'en utilisant les procédures des fichiers à accès direct.De même que pour la création d'un fichier binaire,il faut utiliser une procédure cette même procédure.En effet,les fichiers à accés direct sont stockés sous forme binaire condensé (sur l'olivetti).

Le caractère de fin de fichier(EOF) dans le cas de l'ASCII n'étant pas le même que celui du binaire,un problème se posait: le test de EOF n'est plus applicable comme critère de fin de fichier.En effet,si le caractère de fin de fichier en ASCII est bien connu et égal à "1AH" (soit CTRL-Z),il nous ait demeuré inconnu en binaire condensé.Pour cette raison,nous n'arrivions pas à faire une copie complète d'un fichier binaire.

On utilisa alors,la taille du fichier comme condition d'achèvement de la copie.Il suffisait de compter le nombre d'enregistrements effectivement copiés et de les comparer à la taille (réelle) du fichier.

Nous essayames alors,d'élaborer un programme de transmission et de réception par paquets de 255 octets.Mais si la connaissance de la taille d'un fichier est très aisée lorsqu'on opère sur un même microordinateur(cas de la recopie de fichiers),ceci l'est moins lorsqu'on doit faire un transfert entre deux microordinateurs.

Il s'imposait alors,d'envoyer en plus du nom du fichier et des signaux nécessaires au sous programme de réception,la taille de ce même fichier(paramètre indispensable à la fermeture du fichier à la réception).

Le programme EMETJ permet l'envoi d'un fichier binaire vers le port de communication.Le nom du fichier est introduit au clavier ensuite il est ouvert et un numéro lui est affécté.On définit aussi la taille d'un enregistrement.

On ouvre ensuite le port de communiaction en mode ASCII pour envoyer au hôte les informations suivantes:

-un signal permettant au receveur d'entammer la lecture de son port de communication

-la taille du fichier et un signal de fin du bloc "longueur du fichier".

-puis en fin le nom du fichier

Le port de communication est ensuite ouvert en mode binaire et l'envoi du fichier commence.

Le micro emetteur commence à décrémenter le nombre d'enregistrements et arrête l'envoi dés que ce nombre devient nul.

Le micro receveur compare le nombre d'enregistrements reçus (I) à la taille du fichier (K) (qui a été au préalable envoyée par le micro emetteur).Ce nombre est incrémenté à chaque lecture d'un enregistrement.La lecture du port s'arrête lorsque I=K.

Le programme RECEPTJ représente le programe de réception de fichiers binaires.

Le programme COPYX permet la recopie d'un fichier binaire. Le programme OLICOM permet le transfert de fichiers aussi bien ASCII que binaires(entre 2 M-24) par paquets de 255 octets et en mode accés direct.

5.4-APPLICATION DU L'ASM86 ET DES FONCTIONS BIOS ET BDOS DANS LA MISE AU POINT D'UN PROGRAMME DE TRANSFERT DE FICHIERS SUR LE RAINBOW-100.

Il s'agit maintenant d'élaborer un programme de communication du RAINBOW 100 avec le M-24.

L'assembleur étant disponible sur le RAINBOW,aucun doute ne subsistait quant au choix du langage à utiliser.Le système d'exploitation CP/M,et plus particulièrement les fonctions FDOS (BIOS et BDOS) permettent d'effectuer les opérations nécessaires pour le maniement des fichiers sur disque,ainsi que les transferts de ou vers le RAINBOW.

Le programme "TRANS.A86" a rendu l'émission du RAINBOW vers le M-24 aussi bien que la réception,possibles.Ce programme est constitué de deux sous programmes:l'un pour l'émission,l'autre pour la réception.

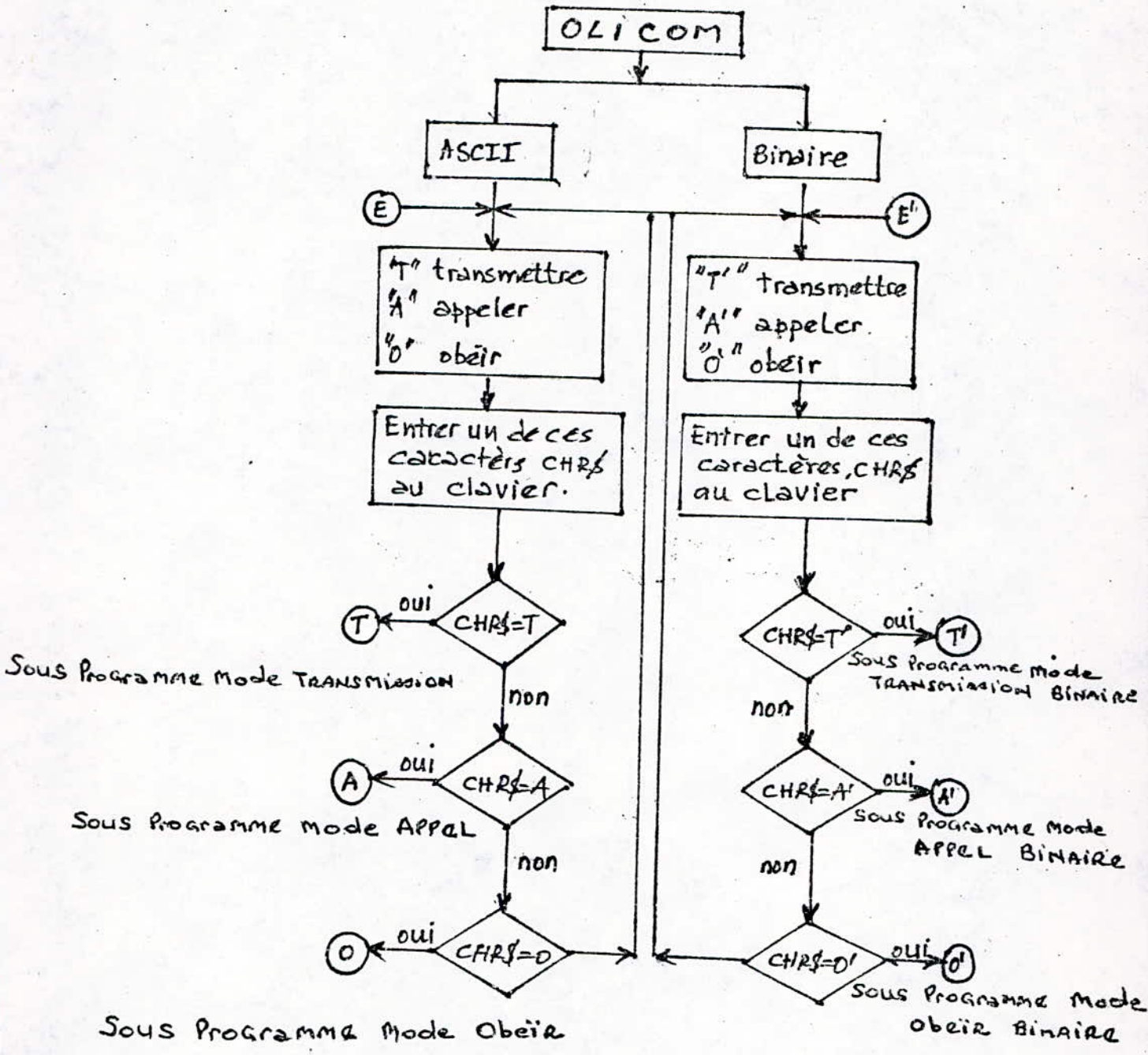
Application des fonctions BDOS dans le programme "TRANS.A86":

A L'EMISSION:

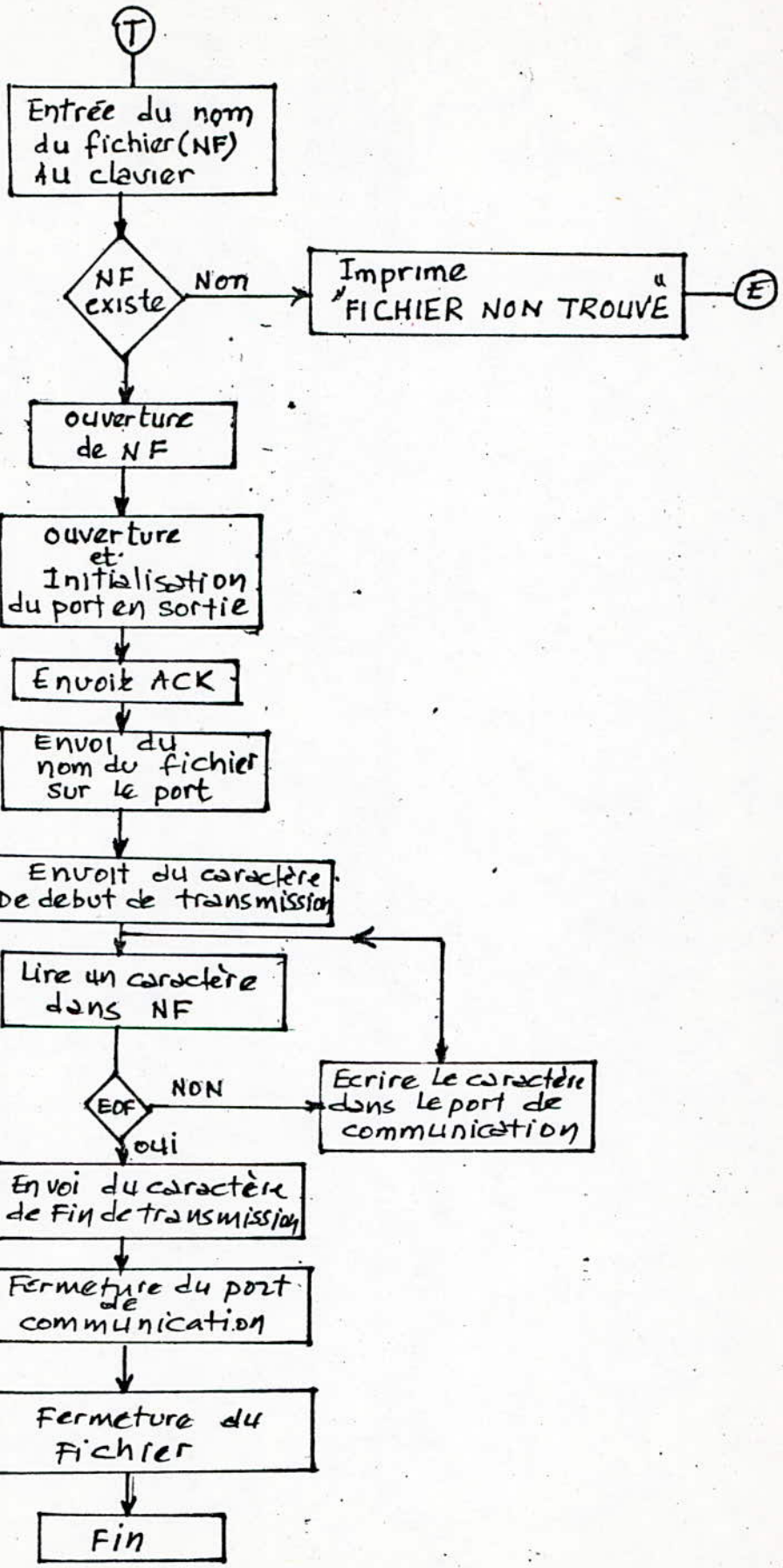
-La fonction 15 ouvre le fichier:pour ouvrir un fichier,les instructions suivantes sont nécessaires:

```
MOV CL,0FH
MOV DX,OFFSET FCB
INT 224
```

Le fichier est ouvert en passant le numéro de fonction (15) dans le registre CL.Dans DX est passé l'offset du FCB du fichier à ouvrir.Le nom du fichier doit être déjà positionner sur le FCB.
L'interruption 224 réalise cette fonction.

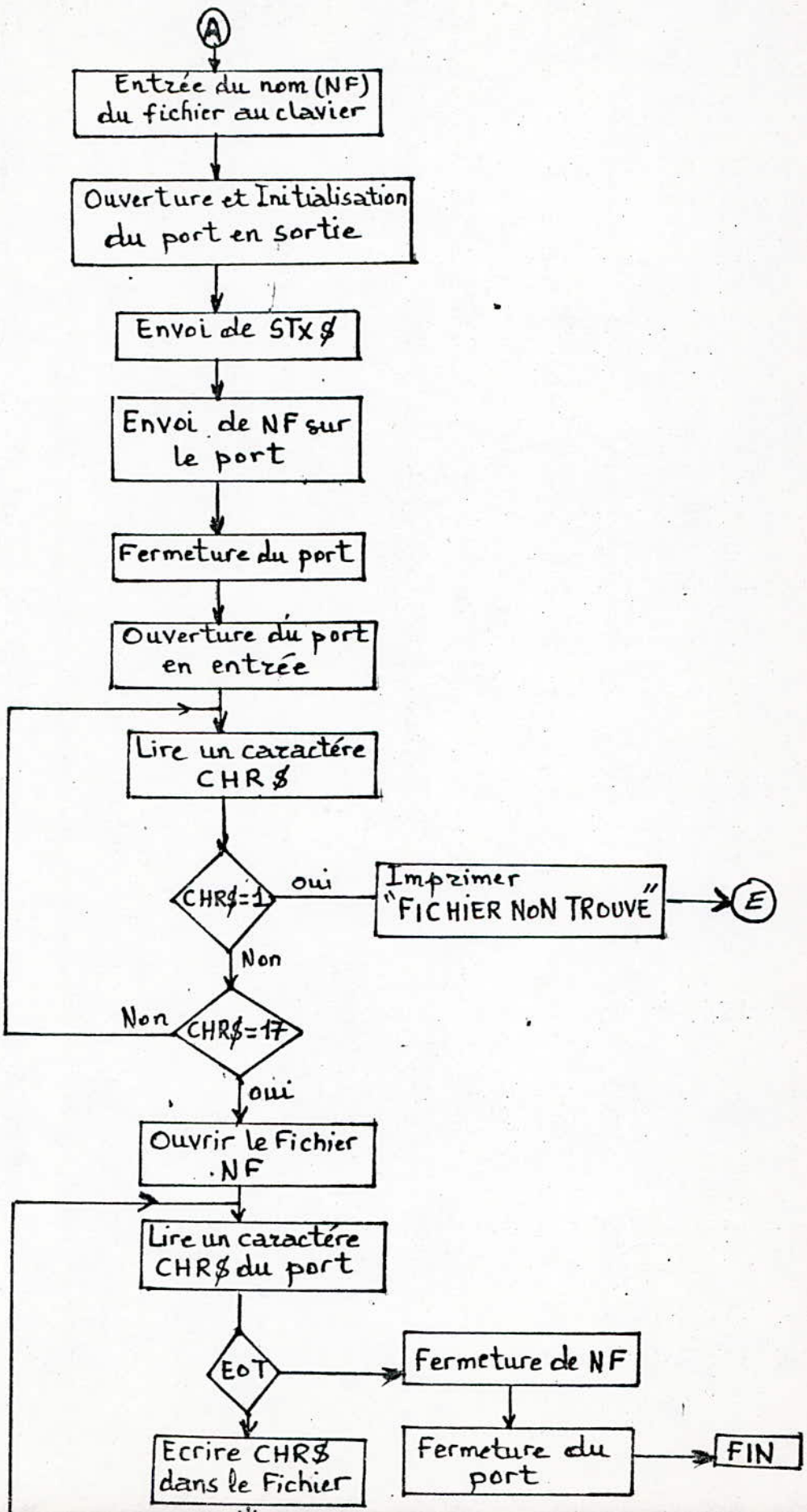


ORGANIGRAMME DU PROGRAMME OLICOM

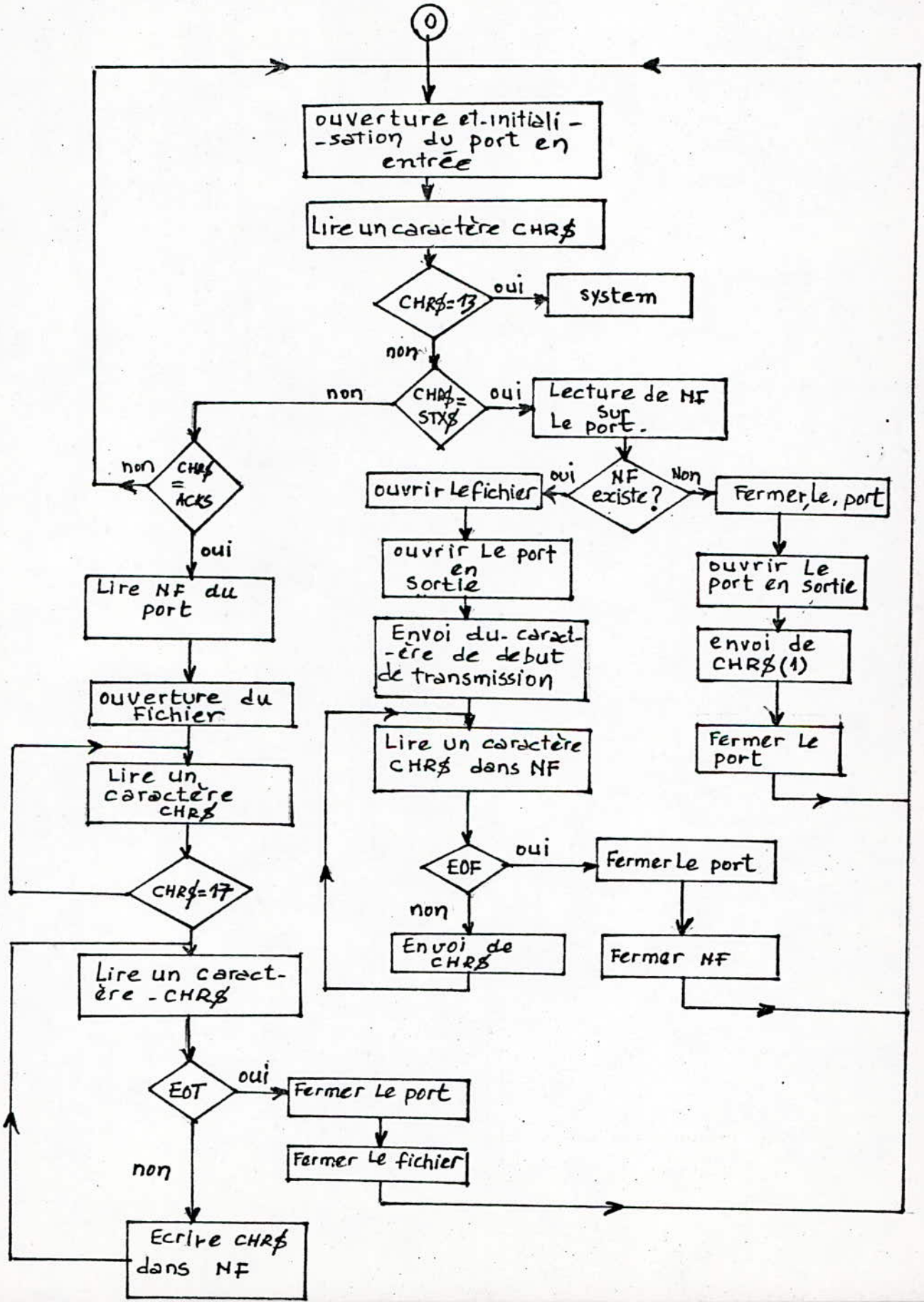


Sous-programme d'émission "OLICOM"

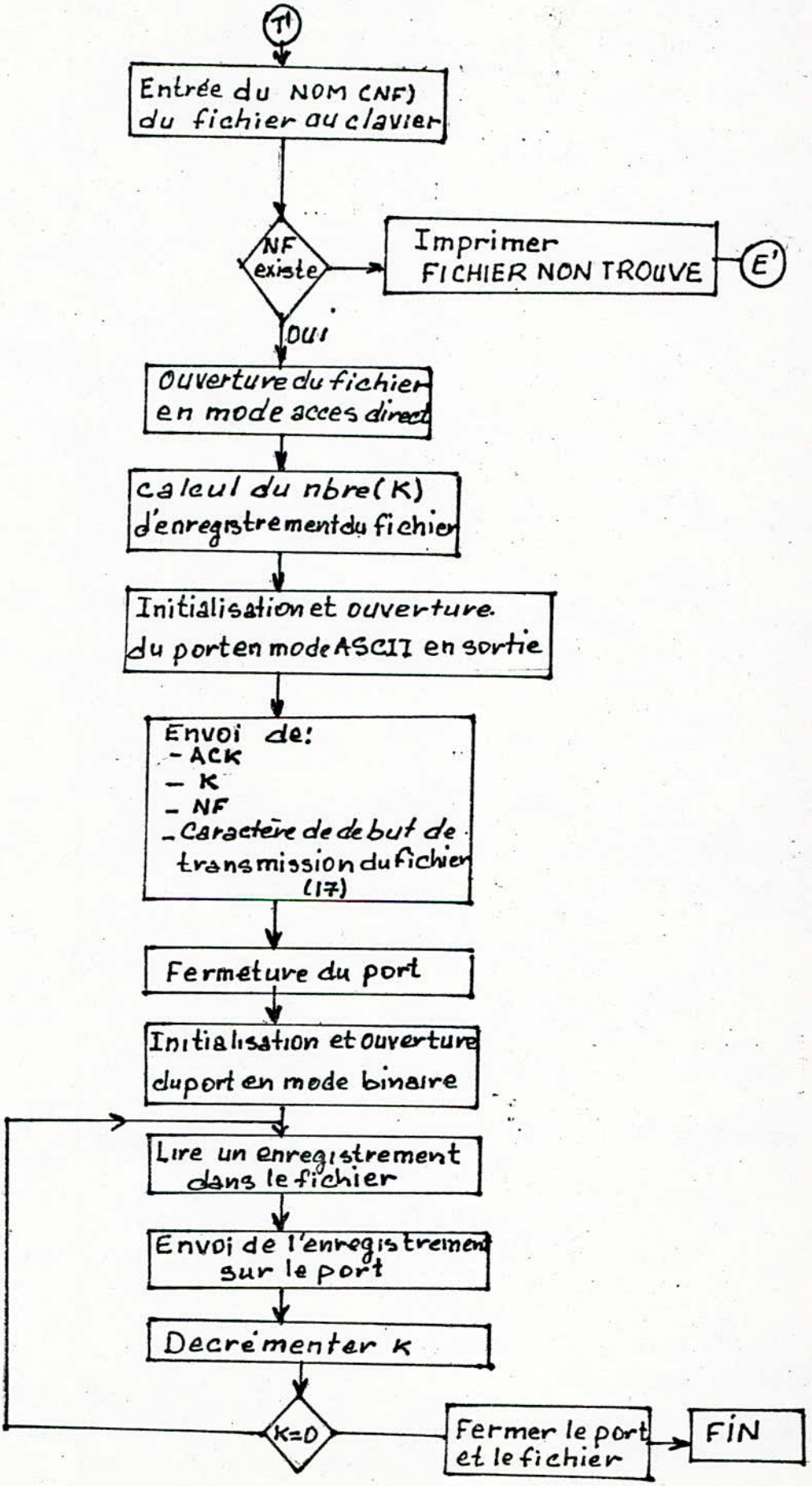
ORGANIGRAMME DU MODE APPEL OLICOM

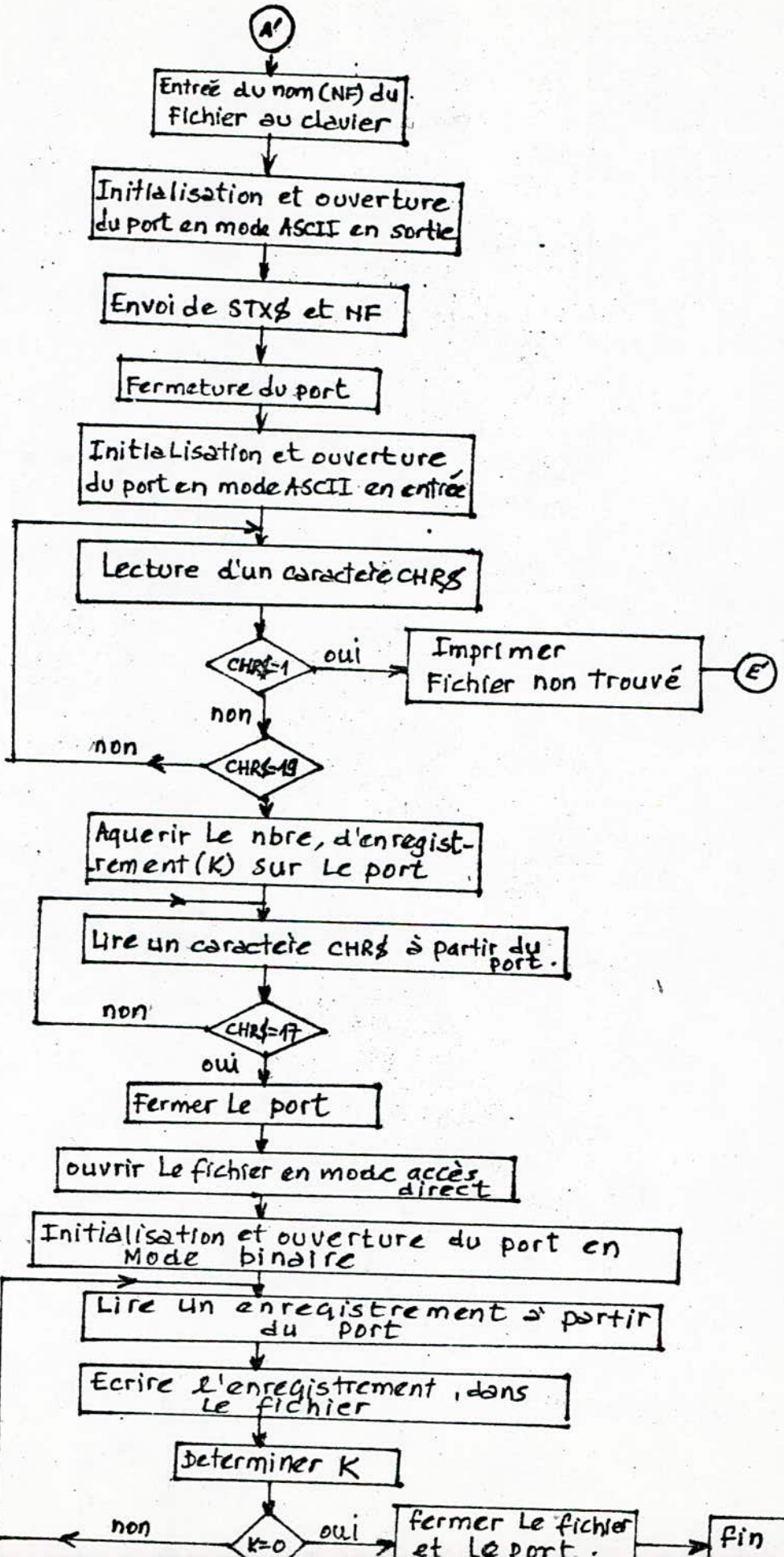


ORGANIGRAMME DU Mode Obéir oLicom

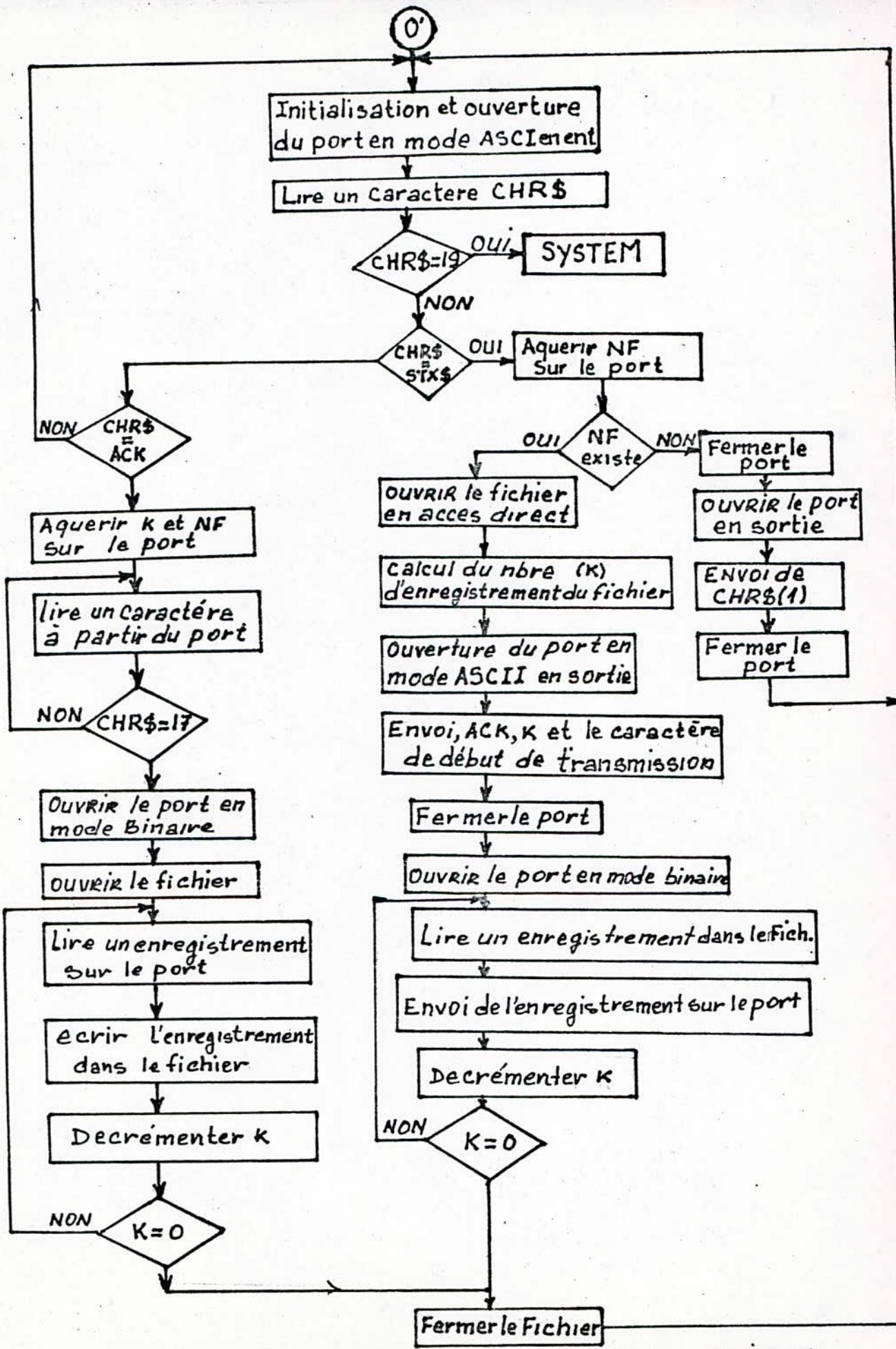


ORGANIGRAMME DU Mode D'amissioN oLicom Binaire





Organigramme du mode appel binaire "OLICOM"



ORGANIGramme du Mode Obeir Binaire OLicom.

-La fonction 20 lit un enregistrement à partir du fichier et le positionne à l'adresse DMA courente(celle-ci, si elle n'est pas spécifiée,est prise par défaut 80H;il suffira de réserver 128 octets à partir de l'adresse 80H).

```
MOV CL,14H
MOV DX,OFFSET FCB
CALL BDOS
```

-L'écriture des caractères sur le port de communication se fait grâce à la fonction 4.

```
MOV CL,4H
MOV DL,'CAR'  ;CAR:cractère à envoyé.
INT 224
```

La donnée à envoyer est passé dans le registre DL.

A LA RECEPTION:

-La fonction 22 permet la création du fichier à la réception.

```
MOV CL,16H
MOV DX,OFFSET FCB
INT 224
```

-La fonction 3 permet l'acquisition des caractères à partir du port de communication et les positionne à l'adresse DMA courante.

```
MOV CL,03H
INT 224
```

La donnée aquise à partir du port est mise dans le registre AL.Ensuite,la donnée est positionné à l'adresse DMA courante.

-La fonction 21 permet d'écrire un enregistrement à partir de l'adresse DMA sur le fichier nouvellement créé.

```
MOV CL,15H
MOV DX,OFFSET FCB
INT 224
```

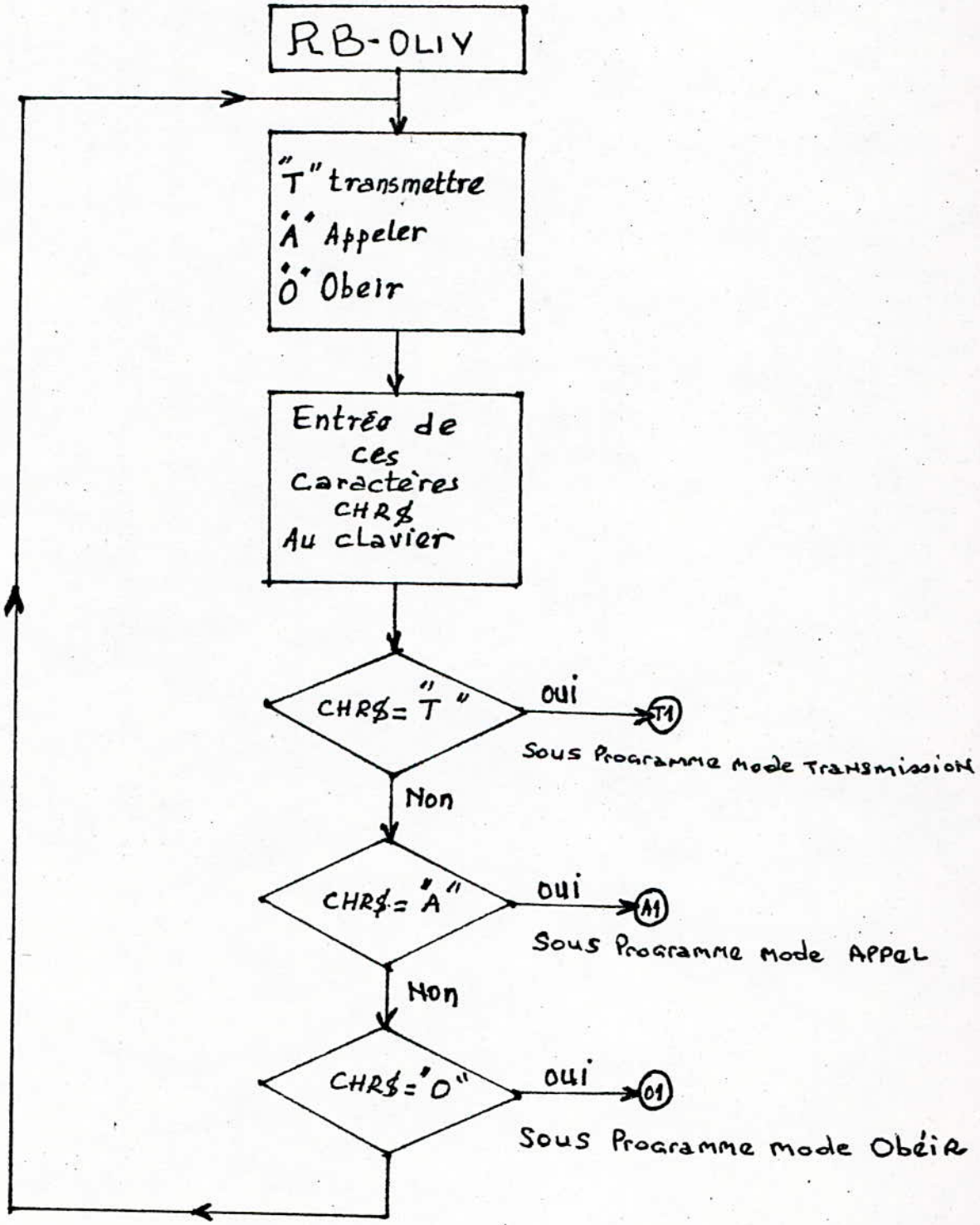
Dans les deux cas de l'émission et la transmision le fichier doit être fermé.La fonction 16 ferme le fichier.

```
MOV CL,10H
MOV DX,OFFSET FCB
INT 224
```

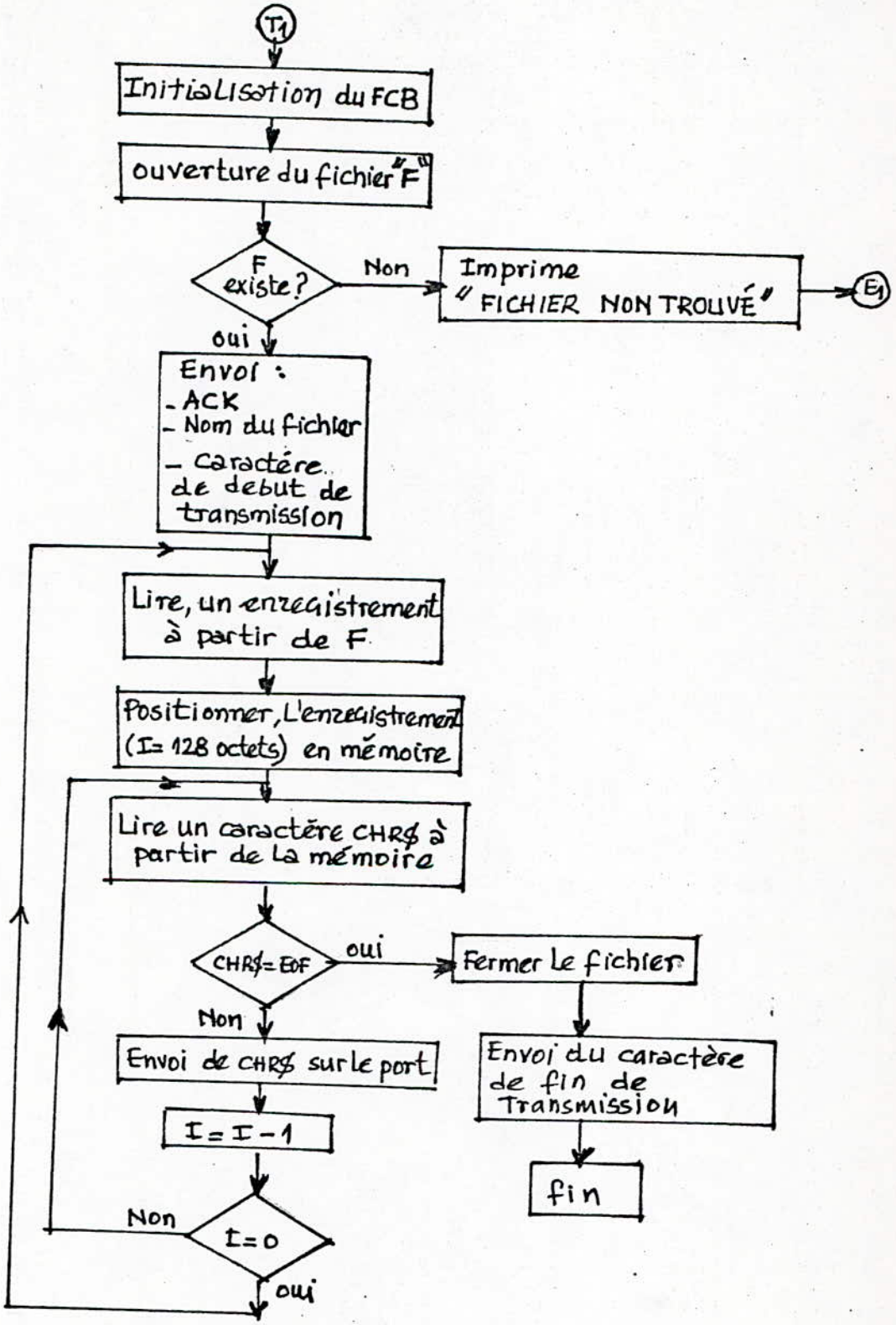
Avant de procéder aux opérations d'émission et de réception, on doit au préalable remplir le bloc de contrôle fichier(FCB).
Dans le FCB doivent être positionnés l'unité dans laquelle se trouve le fichier ainsi que son nom,en plus des autres bits dont on a vu le rôle dans le chapitre de l'étude de CP/M.

*En résumé,à chaque fois que l'on désire effectuer un transfert de fichiers,on doit faire appel à 2 programmes:l'un sur le RAINBOW 100 écrit en langage assembleur et l'autre sur le M-24 écrit en BASIC.Il est à chaque fois nécessaire préciser les noms de fichiers à transmettre et à recevoir respectivement sur le micro transmetteur et sur le receveur.L'idée nous vint alors d'éviter les déplacements d'un micro vers l'autre(en tenant compte du fait que les 2 micros peuvent être assez éloignés, jusqu'à une distance de 600m).Nous avons ensuite amélioré ces deux programmes en y insérant un sous programme "SERVEUR".Ce sous programme ,selon le caractères de contrôle qu'il reçoit,permet de placer le micro hôte en émission ou en réception.Voyons maintenant le principe de ce programme.Il reçoit les signaux de contrôle par le biais du port de communication.Ce sous programme serveur ne doit être lancé que sur l'un des micros qui devient alors "esclave" de l'autre dans la mesure ou il exécutera les demandes formulées à partir du local.
Si l'on désire envoyer un fichier,on sélectionne son nom; puis le programme va d'abord envoyer un caractère spécifique au programme "SERVEUR" pour brancher le hôte en réception.Ensuite,le local envoie le nom du fichier qu'il transmettra.
Le programme final sur le RAINBOW 100 est appelé RB-OLIV.
Son organigramme est donné à la page suivante.

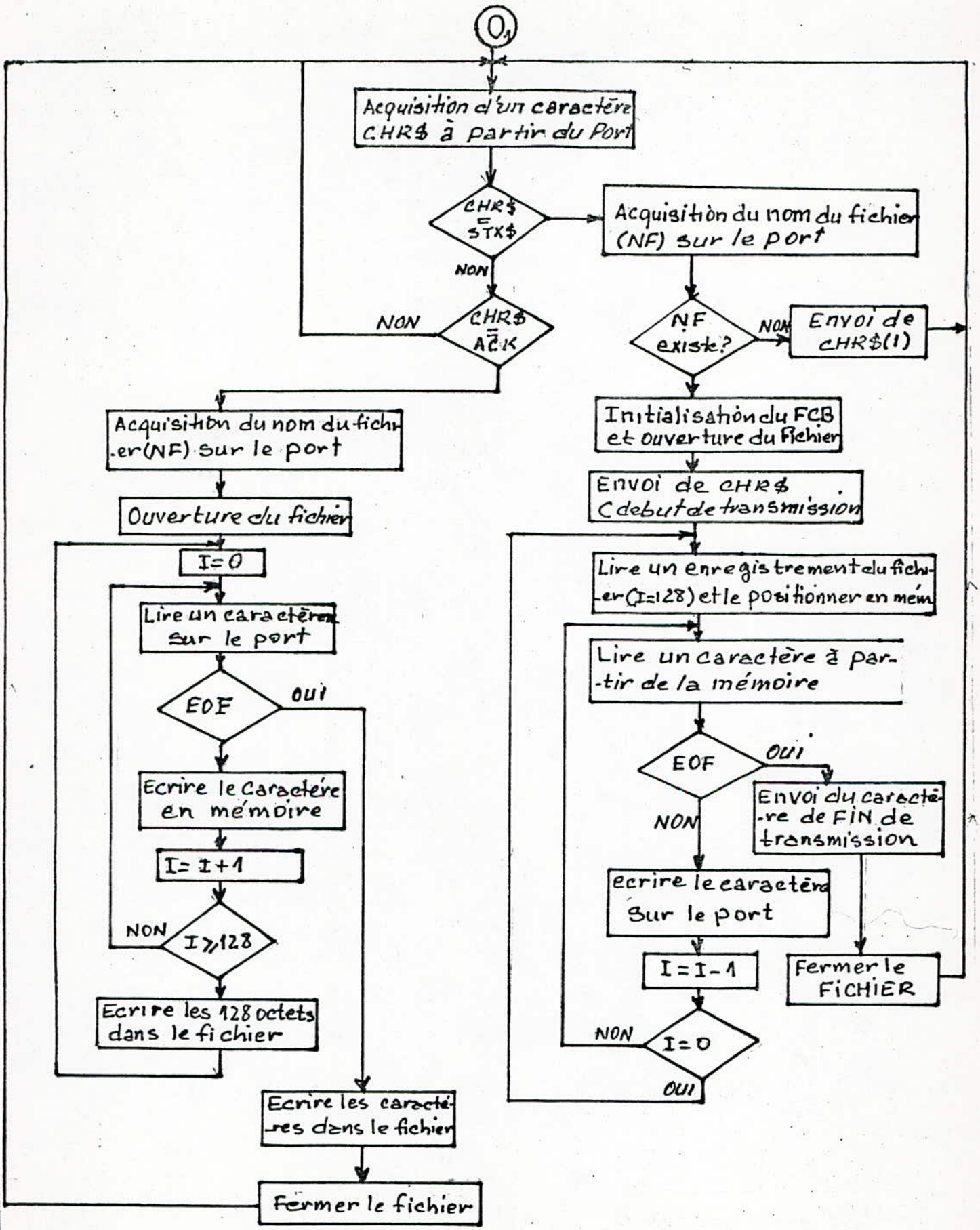
Les programmes que nous avons conçus, permettent les transferts entre deux microordinateurs à une vitesse de 9600 bauds.Ceci est dû au fait que le M24 ne peut transmettre à des vitesses supérieur à 9600 bauds.



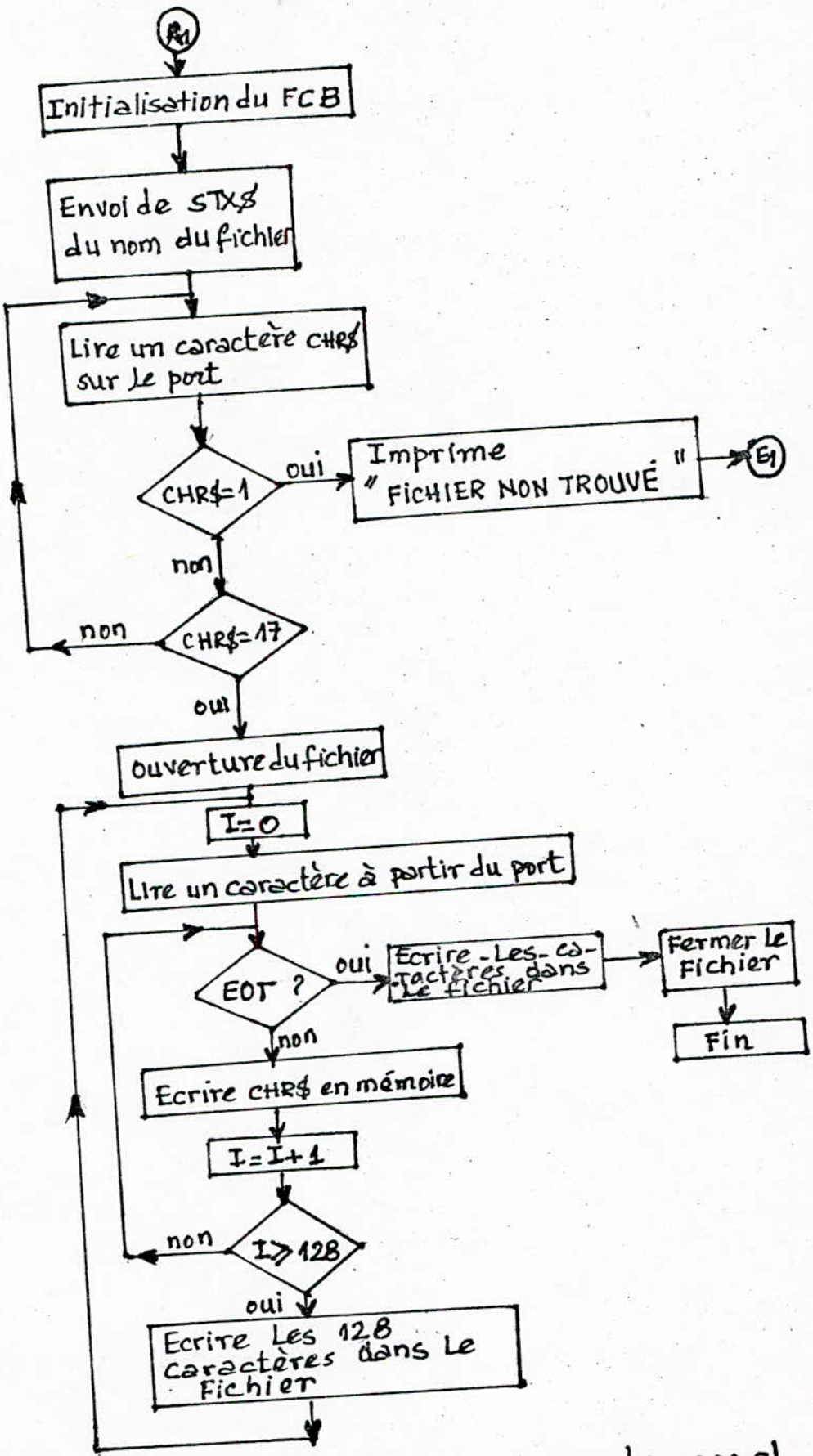
ORGANIGRAMME DU Programme RB-OLIV.



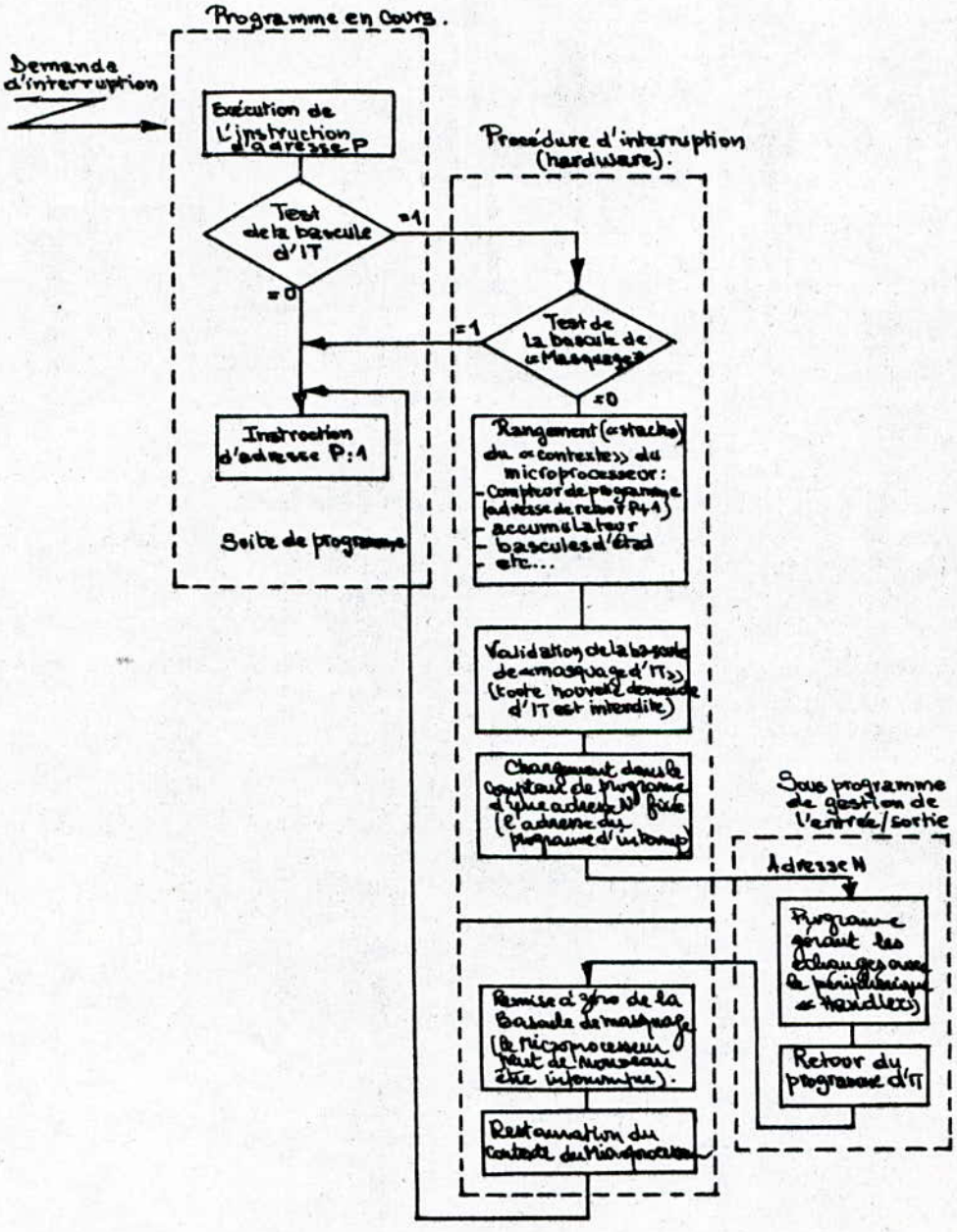
Organigramme du sous_programme d'émission (RB_OLIV)

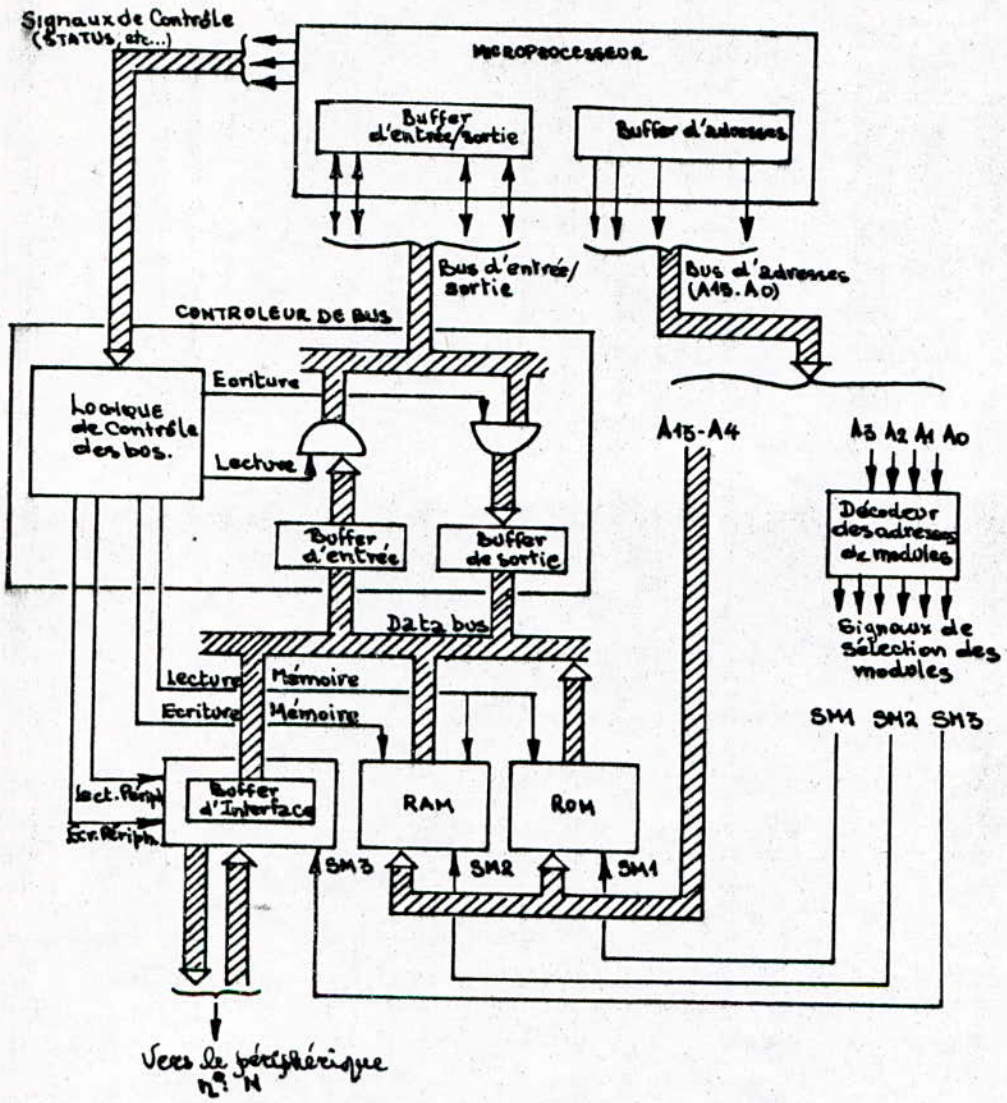


ORGANIGRAMME DU mode Obeïr RB-OLIV



Organigramme du sous-programme du mode appel


Programme en cours.
Demande d'interruption
Exécution de l'instruction d'adresse P
Test de la bascule d'IT
=1
=0
Test de la bascule de «Masquage»
=1
=0
Instruction d'adresse P:1
Suite de programme
Procédure d'interruption (hardware).
Rangement («stacks») du «contexte» du microprocesseur:
- Compteur de programme (adresse de retour P+1)
- accumulateur
- bascules d'état
- etc...
Validation de la bascule de «masquage d'IT» (toute nouvelle demande d'IT est interdite)
Chargement dans le compteur de programme d'une adresse N fixée (l'adresse du programme d'interruption)
Sous programme de gestion de l'entrée/sortie
Adresse N
Programme gérant les échanges avec le périphérique («handler»)
Retour du programme d'IT
Remise à zéro de la bascule de masquage (le microprocesseur peut de nouveau être interrompu).
Restauration du contexte du Microprocesseur



Architecture de principe d'un microordinateur à bus d'entrée/sortie unique.

CONCLUSION:
****

De plus en plus,on cherche à relier des ordinateurs entre-eux,pour leur permettre d'échanger des informations,de partager des programmes...Ces liaisons par lignes constituent des réseaux.D'autre part,au lieu d'acheter plusieurs gros ordinateurs,il est plus économique d'avoir des terminaux ou des microordinateurs et de les brancher à un ordinateur central par le biais d'un réseau.Mais pour utiliser à distance un ordinateur à l'aide d'un terminal (ou pour relier des ordinateurs entre-eux),il est nécessaire de "transporter" les informations entre les divers organes.Ce transport des informations constitue la transmission de données qui nécessite un logiciel de communication.

C'est dans ce but que nous avons écrit nos programmes de transferts de fichiers.Ils peuvent être utilisés dans un logiciel de communication permettant la gestion des échanges de données dans un réseau.Ils permettent aussi de palier les limites techniques inhérentes à chaque microordinateur (mémoire limitée,périphériques plus ou moins performants..) en donnant la possibilité à chacun d'utiliser les ressources de l'autre (mémoire,imprimante).Par exemple,le RAINBOW 100 ne disposant pas du mode graphique,on peut utiliser celui disponible sur l'OLIVETTI M24 en transférant le fichier et en le traitant sur ce dernier.

Notre travail permet de surmonter des problèmes courants tels que l'impossibilité d'un microordinateur de lire une disquette qui a été écrite sur un autre microodinateur.Cela est dû,dans le cas de deux microordinateurs différents,au fait que chaque machine possède un système d'exploitation qui lui est propre.Et bien que soient faits des efforts de standardisation,ce problème est loin d'être résolu.Là encore l'incompatibilité est hélas la règle.Dans le cas de deux microordinateurs de même type (par exemple 2 M24),le problème de lecture de disquettes se pose aussi.

Entre 2 olivettis,nous avons pu réaliser le transfert de fichiers ASCII et de fichiers binaires.

Mais,hélas,entre RAINBOW 100 et M24 les programmes mis au point ne permettent que le transfert de fichiers ASCII. Le problème que nous avons eut dans le cas des fichiers binaires est dû au fait suivant:nous avons utilisé les fonctions FDOS de CP/M dans la mise au point du programme de transfert sur le RAINBOW 100.Or toutes les données manipulées par les fonctions FDOS sont du type ASCII (à 7 bits) et un caractère binaire condensé est constitué de 8 bits,donc:il est impossible de réussir un transfert de fichiers binaires à partir du RAINBOW 100 en utilisant les fonctions FDOS.Il faut passer par la programmation du MPSC

à 8 bits par caractère et utiliser les instructions IN et OUT pour les E/S.

Les informations données sur les microprocesseurs 8086/8088 ,sur le système d'exploitation et sur le langage assembleur peuvent servir de manuel de base.La démarche que nous avons suivit peut être appliquée dans toute liaison de microordinateurs compatibles avec le RAINBOW 100 et le M24.

Nous tenons à souligner que l'expérience que nous avons vécue dans le cadre de l'élaboration de ce projet nous a été très bénéfique,et va nous permettre d'aborder,dorénavant,des applications encore plus délicates et plus complexes.

à 8 bits par caractère et utiliser les instructions IN et OUT pour les E/S.

Les informations données sur les microprocesseurs 8086/8088 ,sur le système d'exploitation et sur le langage assembleur peuvent servir de manuel de base.La démarche que nous avons suivit peut être appliquée dans toute liaison de microordinateurs compatibles avec le RAINBOW 100 et le M24.

Nous tenons à souligner que l'expérience que nous avons vécue dans le cadre de l'élaboration de ce projet nous a été très bénéfique,et va nous permettre d'aborder,dorénavant,des applications encore plus délicates et plus complexes.

BIBLIOGRAPHIE
****

*"8086-8088 PROGRAMMATION EN LANGAGE ASSEMBLEUR"
par B.GEOFFRION Editions Radio (1984)

* LE MICROPROCESSEUR 16 BITS
8086/8088
Matériel.Logiciel Système d'exploitation
par A.B.FONTAINE Edition MASSON (1984)

* "L'INFORMATIQUE PROFESSIONNELLE" N°14 JUIN-JUILLET
(1984)

*"MICROPROCESSEURS ET MICROORDINATEURS"
par R.LYON-CAEN & J.M.CROZET Edition MASS

*"TELEINFORMATIQUE" C.MACCHI,J.F.GUILBERT
Edition DUNOD (1983)

*"CP/M operating system guide" INTEL (1981)

*"ASM-86 programmers guide" INTEL

*"MS-DOS user's guide" CHRIS DE VORNEY
Microsoft MS-DOS operating system

*"PROGRAMMATION EN LANGAGE BINAIRE ET SYMBOLIQUE" M.THORIN

*"PC-100 System module,chapter 4"

*"Programming the 8086/8088" James W.Coffron

*"8274 Multi Protocol Serial Controller MPSC"
INTEL corporation (1983)

*"MS-GWBASIC INTERPRETE ,Guide de l'utilisateur"
OLIVETTI (1984)

*"SYSTEME D'EXPLOITATION MS-DOS ,Guide de l'utilisateur"
OLIVETTI (1984)

*"RAINBOW 100 ,Guide d'utilisation" DIGITAL

BIBLIOGRAPHIE
****

*"8086-8088 PROGRAMMATION EN LANGAGE ASSEMBLEUR"
par B.GEOFFRION Editions Radio (1984)

* LE MICROPROCESSEUR 16 BITS
8086/8088
Matériel.Logiciel Système d'exploitation
par A.B.FONTAINE Edition MASSON (1984)

* "L'INFORMATIQUE PROFESSIONNELLE" N°14 JUIN-JUILLET
(1984)

*"MICROPROCESSEURS ET MICROORDINATEURS"
par R.LYON-CAEN & J.M.CROZET Edition MASS

*"TELEINFORMATIQUE" C.MACCHI,J.F.GUILBERT
Edition DUNOD (1983)

*"CP/M operating system guide" INTEL (1981)

*"ASM-86 programmers guide" INTEL

*"MS-DOS user's guide" CHRIS DE VORNEY
Microsoft MS-DOS operating system

*"PROGRAMMATION EN LANGAGE BINAIRE ET SYMBOLIQUE" M.THORIN

*"PC-100 System module,chapter 4"

*"Programming the 8086/8088" James W.Coffron

*"8274 Multi Protocol Serial Controller MPSC"
INTEL corporation (1983)

*"MS-GWBASIC INTERPRETE ,Guide de l'utilisateur"
OLIVETTI (1984)

*"SYSTEME D'EXPLOITATION MS-DOS ,Guide de l'utilisateur"
OLIVETTI (1984)

*"RAINBOW 100 ,Guide d'utilisation" DIGITAL