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 : D'ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

SIMULATION DE L'ANALYSE DE LA PAROLE A L'AIDE DU MICROPROCESSEUR TMS 32010

Proposé par : N. BENIDDIR

Etudié par : H. OUHAB, D.E. TALBI

Dirigé par : N. BENIDDIR

PROMOTION : JUIN 1987

D E D I C A C E S

Je dédie ce modeste travail :

- A la mémoire de mes grands parents .
- A mon père et ma mère .
- A mes fréres et soeurs .
- A toute ma famille .
- A tous mes amis .

DJAMAL EDDINE

Je dédie ce modeste travail :

- A la mémoire de mes grands parents .
- A mon père et ma mère .
- A mes fréres et soeurs .
- A toute ma famille .
- A tous mes amis .

HACENE

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

Nous tenons à éxprimer nos vifs remerciments à notre promoteur Mr N.BENIDDIR, pour son aide permanente et ses précieux conseils qu'il nous a prodigués tout le long de ce travail .

Nous remercions également, Mr BOUSSEKSOU pour ses précieux conseils .

Que Mr SAADOUN et Mme HAMAMI trouvent en ses lignes l'éxpression de notre profonde gratitude pour l'aide précieuse qu'ils nous ont apporté .

La frappe de se polycopié à été faite par Mr M.TALBI, qu'il trouve ici l'éxpression de notre sincère reconnaissance .

Que tous ceux qui ont contribué à notre formation trouvent ici l'éxpression de notre gratitude .

S O M M A I R E

INTRODUCTION

Généralités :

La possibilité d'utiliser la parole, moyen naturel et privilégié de communication, dans le dialogue homme-machine, ouvre de nombreuses applications parmis lesquelles nous pouvons citer : les commandes de machines outils et robots, les concéptions de systèmes assistés par ordinateurs, les machines d'aide aux handicapés,...

L'analyse acoustique est une partie importante que subit le signal sonore pour pouvoir réaliser un système de reconnaissance de la parole.

Si les progrès réalisés du point de vue algorithmiques (FFT, DTW, classification,...) ont permis un accroissement très important des performances des systèmes de reconnaissance, les progrés de la téchnologie ont contribué à cet accroissement du point de vue vitesse de traitement.

Aussi, la communication parlée doit-elle son regain d'interêt à l'électronique numérique et particulièrement aux microprocesseurs spécialisés en traitement du signal.

Présentation du travail :

Notre projet de fin d'études comprend deux axes principaux :

- Etudier le microprocesseur TMS 320 10 de texas instruments spécifique au traitement du signal.
- Développer des programmes en assembleur pour l'analyse de la parole, à l'aide de la simulation du TMS 320 10 sur OLIVETTI M24.

Pour atteindre ces objectifs, le travail a été mené de la manière suivante :

Le premier chapitre est consacré à la présentation du TMS 320 10 ainsi que la mise en oeuvre de sa simulation sur le micro-ordinateur OLIVETTI M24.

Le second chapitre traite les téchniques d'analyse les plus utilisées en reconnaissance de la parole.

La mise en oeuvre sur le TMS 320 10 de l'analyse par prédiction linéaire, et de la transformée de fourier rapide, fait l'objet du troisième chapitre.

On trouvera enfin, dans le dernier chapitre, des recommandations pour la concrétisation de ce travail en un système complet de reconnaissance de la parole à base du TMS 320 10.

C H A P I T R E I

ETUDE DU T M S 320

A / EXPOSE DES CARACTERISTIQUES

I-I - Déscription
I-2 - Architécture
I-3 - Eléments constituant le Hardware
I 4 - Instructions
I-5 - Modes d'adréssage
I-6 - Interfaçage et controle
I-7 - L'assembleur
I-8 - Systèmes et logiciels de développement

B / SIMULATION

I-I - Mise en oeuvre de la simulation sur le microordinateur OLIVETTI M 24
I-2 - Commandes du simulateur
I-3 - Chargement des fichiers
I-4 - Chargement de la mémoire de données
I-5 - Simulateur des fichiers journaux

A/ EXPOSE DES CARACTERISTIQUES

I-I- DESCRIPTION :

La famille TMS 320 est la première vérsion MOS des microcalculateurs basée sur une architécture parallèle I6/32 bits , elle pérmet l'éxécution de cinq (5) millions d'instructions par seconde.

Le TMS 320 IO fût le premier processeur numérique du signal I6/32 bits de cette famille , taillé pour réaliser des traitements rapides; une puissance remarquable et une souplesse lui ont donné la capacité de pérformer de multiples fonctions , souvant demandées pour une simple application .

Ce processeur couvre un champ trés vaste d'applications , notament le traitement du signal (filtrage numérique, transformée de Fourier rapide corrélation, fenêtrage, etc ...), le traitement de la parole (analyse, synthèse, reconnaissance, vocodeurs, etc ...), l'instrumentation, le traitement numérique, le contrôle haute vitesse, les télécommunications et le traitement de l'image .

Les principales caractéristiques de cette famille sont :

- Temps de cycle d'une instruction de 200 ns ,
- Mémoire RAM intégrée de 288 octets (I44 mots) ,
- Multiplication I6 par I6 bits en 200 ns ,
- Accumulateur et unité arithmétique et logique sur 32 bits ,
- Deux (02) décaleurs à gauche ,
- Huit (08) voies d'entrées et huit (08) voies de sortie ,
- Arithmétique fixe signée en complément à deux ,
- Téchnologie N-MOS (2,7 microns) ,
- Alimentation en monotention (05 V) ,
- Boitier DIP de 40 broches ,
- Dissipation typique de 950 mW ,
- Fréquance d'horloge variant de 6,7 MHZ à 20 MHZ ,

Le premier membre de cette famille se présente sous deux (02) vérsions : Le TMS 320 IO et le TMS 320 M IO , ces deux procésseurs sont identiques , à la seule éxcèption que le TMS 320 M IO utilise une mémoire éxtèrne ,

et une mémoire intèrne , et le TMS 320 IO une mémoire éxtèrne uniquement .
Le schéma fonctionnel de ce premier membre est donné par les figures I et 2 .

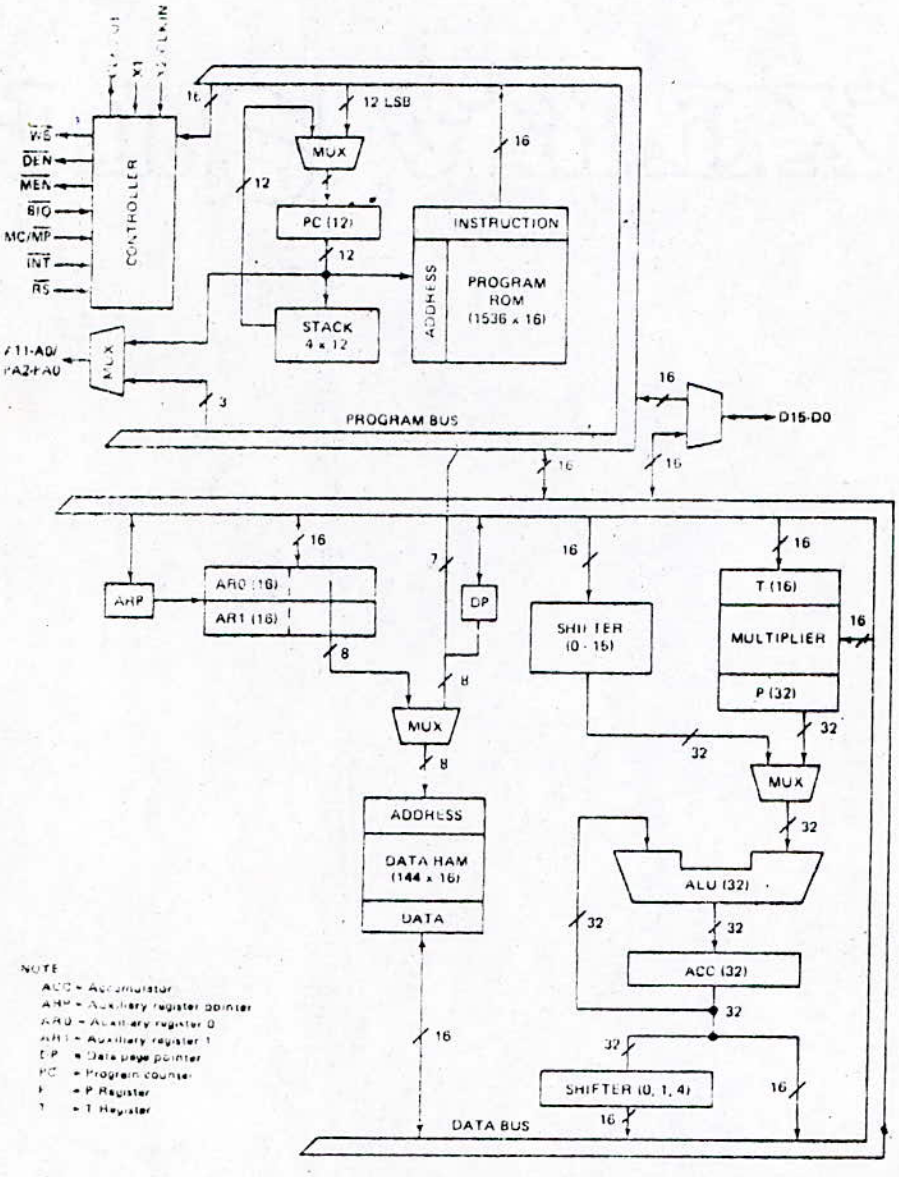


FIGURE 2 - BLOCK DIAGRAM OF TMS320M10

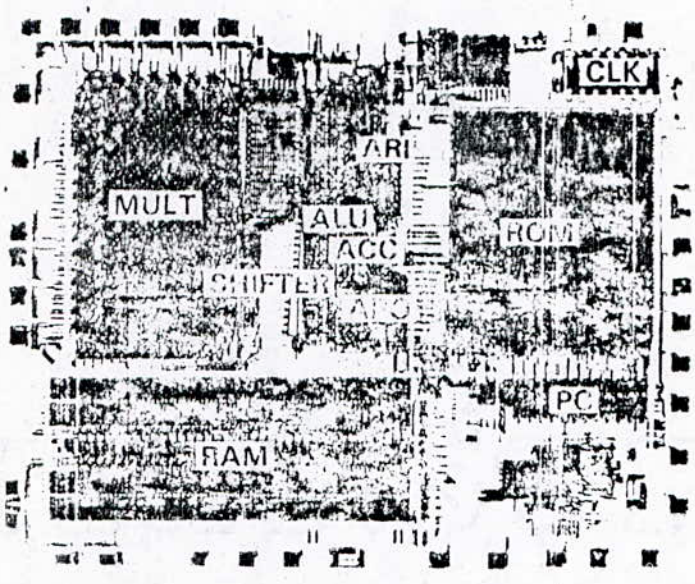


Figure (I) – TMS 320 IO

I-2– ARCHITECTURE :

Cette famille utilise une architecture dite de " HARVARD " modifiée , cette modification lui a donnée l'avantage de pouvoir lire des coéficients stockés en mémoire programme (ROM) pour les transférer dans la mémoire des données (RAM) , et aussi la séparation des bus de données et de programme , ce qui recouvre totalement le temps de l'acquisition et de l'éxécution d'une instruction .

Définition des broches :

- Voir table I çi-jointe , et figure(3).

- Table I : TMS 320 IO Pin définitions

SIGNAL	I/O	DEFINITION
VCC,VSS	IN	Power and ground
XI	IN	Crystal input
X2/CLKIN	IN	Crystal input or external clock input
CLKOUT	OUT	System clock output,I/4 crystal/CLKIN frequency
WE	OUT	Write enable indicates valid data on DI5-DO.
DEN	OUT	Data enable indicates the processer accepting input data on DI5-DO.
MEN	OUT	Memory enable indicates that DI5-DO will accept external memory instruction .
RS	IN	Reset used to initialize the device
INT	IN	Interrupt
BIO	IN	External polling input for bit test and jump operations .
MC/MP	IN	Memory mode select pin . High selects microcomputer mode . Low selects micro-processor mode . (See section,Program Memory and Memory Expansion) .
DI5/DO	I/O	I6 bit data bus
AII-AO	OUT	External address bus . I/O port adress multiplexed
PA2-PAO		Over PA2-PAO

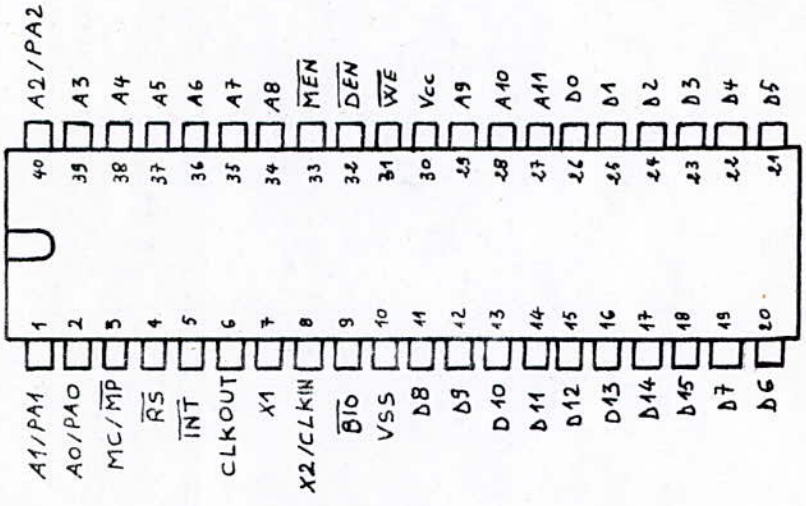


Figure (3) - Brochos du TMS 320 IO

I-3- ELEMENTS CONSTITUANT LE HARDWARE :

Le T M S 320 I0possède quatre (04) éléments arithmétiques de base qui sont : l'unité arithmétique et logique (ALU) , l'accumulateur , le multiplieur et le décaleur .

I°)- L'unité arithmétique et logique (ALU) :

L'ALU manipule les données sur des mots de 32 bits . En plus des opérations classiques , elle éffectue aussi les opérations logiques entre les I6 bits les moins significatifs de l'accumulateur et la valeur de la mémoire des données .

2°)- L'accumulateur :

Il opère également sur des mots de 32 bits . il est séparé en deux (02) parties , une partie supérieure réservée aux I6 bits de poids le plus fort et une partie inférieure pour les I6 bits de poids le plus faible . Il détient aussi l'adresse de branchement de la mémoire programme durant les opérations de branchement et contient aussi le mode " OVERFLOW " .

3°)- Le multiplieur :

La multiplication est pérformée à l'aide d'un multiplieur cablé (I6 par I6 bits) en complément à deux . Elle utilise deux (02) registres pour le multiplicande et le résultat du produit , le multiplieur est un mot de I6 bits de la RAM de données , ou une valeur de I3 bits d'un mot d'instruction en mode immédiat .

4°)- Le décaleur :

Deux (02) décaleurs à gauche sont pérformés par le T M S 320 :
- Un décaleur de 0 à I5 positions , pérmet de manipuler la donnée avant de la stocker en mémoire .
- Un autre décaleur de 0,I ou 04 positions , pérmet le décalage des I6 bits de poids le plus fort de l'accumulateur avant

rangement de la donnée en RAM .

Les autres éléments sont les suivants :

5°)- Mémoire de programme et éxtention :

Le T M S 320 M IO est équipé d'une mémoire programme (ROM) intèrne de I536 mots extensible à l'aide d'une mémoire (ROM) éxtèrne de 2560 mots .

Le T M S 320 M IO peut fonctionner sous deux (02) modes définis par l'état de la broche MC / MP .

- L'état MC désigne le mode microcalculateur , dans ce mode les instructions adressées de 0 à I535 sont dans la mémoire intèrne et celles qui ont pour adresses de I536 à 4095 sont dans la ROM éxtèrne , voir figure (4) .
- L'état MP désigne le mode microprocesseur , dans ce mode les instructions adressées de 0 à 4095 sont toutes dans la ROM éxtèrne .

Le T M S 320 IO est identique au T M S 320 M IO à l'éxcèption que le T M S 320 IO opère en mode microprocesseur (MP) uniquement .

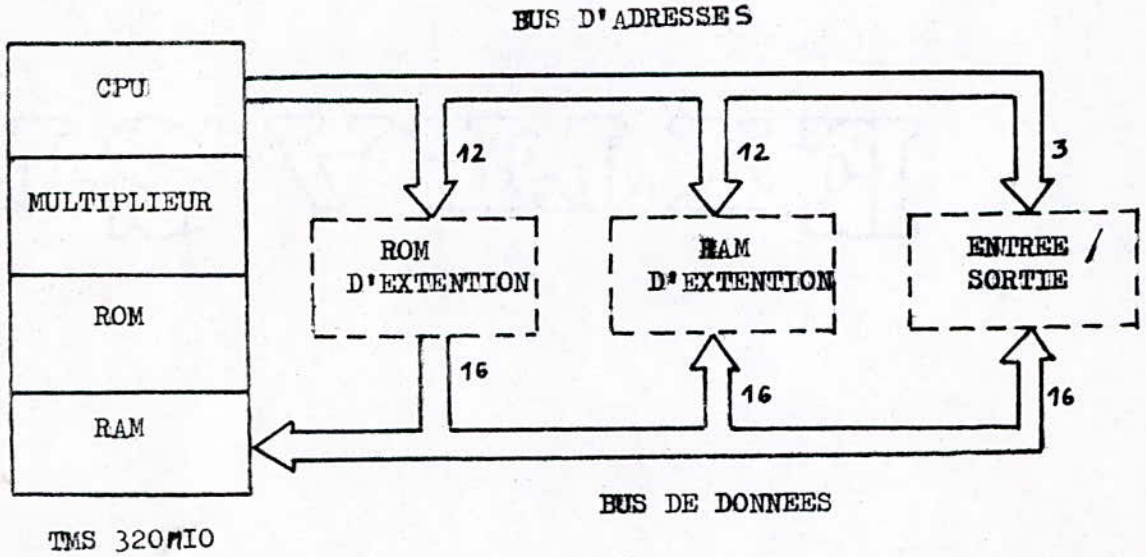


Figure (4) - Extention de la mémoire programme

L'habilité d'utiliser une mémoire éxtèrne fournit les avantages suivants :

- Facilité de manipulation
- Exécution à partir des RAM éxtèrnes
- Facilité de codage

6°)- Mémoire de données :

La mémoire de données est constituée de I44 mots , elle est organisée en deux (02) pages , la page 0 et la page I . Son adressage se fait selon deux (02) modes : le mode direct ou le mode indirect .

Le mode direct pérmet l'adressage d'une seule page suivant qu'on séléctionne l'une ou l'autre .

Quant au mode indirect , il couvre l'adressage de toute la mémoire et se fait à l'aide de l'un des deux (02) registres auxilliaires ARI .

7°)- Compteur programme (PC) :

C'est un registre de I2 bits contenant l'adresse courante de la mémoire programme .

8°)- Pile :

Elle est composée de quatre (04) registres de I2 bits pérméttant la sauvegarde du contenu du compteur programme durant un branchement vers un sous-programme ou une intérruption . Elle sert aussi à la lecture / écriture en mémoire éxtèrne .

9°)- Pointeur de page de la mémoire de données (DP) :

Il est formé d'un seul bit qui séléctionne la page de la RAM de données . La première page contient I28 mots et la seconde I6 mots seulement .

10°)- Registres auxilliaires 0 et I (ARI) :

Ils sont composés de 16 bits chacun , les huit (08) bits les moins significatifs sont utilisés pour l'adressage indirect de la mémoire de données ; les neuf (09) bits les moins significatifs peuvent être aussi considérés comme des compteurs bidirectionnels pour le controle d'une boucle avec auto incrémentation ou auto décrémentation , ces registres peuvent être aussi utilisés pour la sauvegarde temporaire des données .

11°)- Pointeur de registre auxilliaire (ARP) :

Il est formé d'un seul bit qui séléctionne le regitre auxilliaire courant .

12°)- Registre T :

C'est un registre de 16 bits qui contient le multiplicande dans une opération de multiplication .

13°)- Registre P :

C'est un registre de 32 bits qui contient le résultat de la multiplication .

14°)- Indicateur d'intérruption (INTF) :

IL contient un seul bit , utilisé pour indiquer une intérruption , il est automatiquement mis à zéro dés que l'intérruption est prise en considération .

15°)- Indicateur de mode d'intérruption (INTM) :

Il contient un seul bit , utilisé pour masquer l'indicateur de l'intérruption . Une fois l'intérruption est prise en considération , ce bit est mis à I par l'instruction DINT , ceci bloque toutes les autres intérruptions qui viennent aprés .

Dés que le mode d'intérruption est términé , ce bit est remis à zéro par l'instruction EINT .

I6°)- Indicateur de dépassement (OV) :

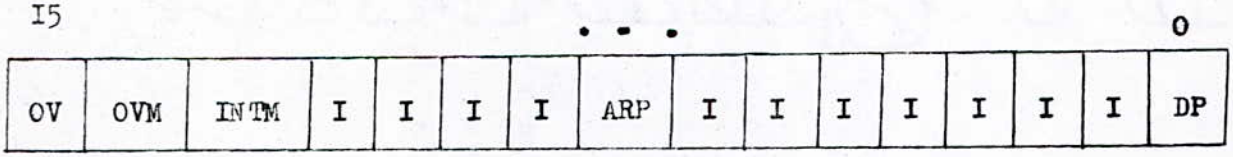
Il contient un seul bit indiquant le dépassement dans les opérations arithmétiques , il est mis à zéro par l'instruction BV (Branch on Overflow) .

I7°)- Mode de dépassement (OVM) :

Il contient un seul bit qui définit si le T M S 320 IO opère en mode de saturation ou de non saturation pendant les opérations arithmétiques . Dans le mode de saturation , un dépassement ou un non dépassement pérmet à l'accumulateur de représenter sa valeur réspéctivement sous une forme étendue ou réduite .
Notons qu'un (OI) logique sur ce bit permet le mode de dépassement et un zéro (0) logique un mode de non dépassement .

I8°)- Registre d'état :

C'est un registre de I6 bits , il contient les indicateurs OV , OVM , INTM , ARP et DP .



I9°)- Horloge :

Le T M S 320 IO peut utiliserson oscillateur intèrne ou un oscillateur éxtèrne comme horloge . L'oscillateur intèrne est mis en service par la connection d'un cristal à travers XI et X2 / CLKIN .
La fréquence de CLKOUT est le quart de la fréquence de CLKIN .

I-4- INSTRUCTIONS :

Le T M S 320 possède un jeu de 60 instructions dont la majorité sont opérées en un mot et un cycle seulement .

On peut les classer selon les catégories suivantes :

- Instructions de l'accumulateur :

Elles permettent l'addition , la soustraction , le chargement et le stockage de l'accumulateur .

- Instructions des registres auxilliaires et de la page de données :

Elles permettent le chargement , le stockage , la modification et la comparaison des registres ARI et ARP .

- Instructions du registre T , du registre P et de la multiplication :

Elles permettent l'éxécution de la multiplication .

- Instructions de controle :

Elles affèctent le mode de dépassement , les intérruptions et stockent certains registres qui ne peuvent être stockés par d'autres instructions (exp : SST , LST) .

- Instructions de branchement :

Il éxiste une grande variété d'instructions de branchement qui permettent de tester les conditions suivantes :

- Dépassement.
- Niveau bas sur la broche BIO.
- Contenu de l'accumulateur , inférieur, égal ou supérieur à zéro.
- Partie non nulle du registre auxilliaire ARI.

- Opérations bouléenne :

Elles pérforment les opérations logiques entre l'accumulateur et la mémoire de données .

- Opérations d'E/S et de mémoire de données :

Elles permettent l'entrée / sortie des données vers les périphériques et le transfert de données entre la mémoire programme et la mémoire de données .

La table -I- donne les différents instructions du T M S 32010 , le nombre de cycles corréspondant ainsi que le code opération (voir Annexe).

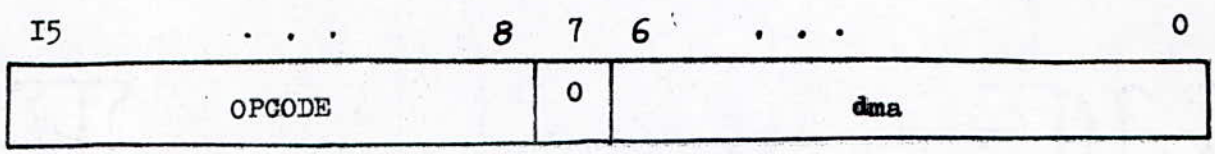
I - 5- MODES D'ADRESSAGE :

Trois modes d'adressage sont utilisés par le T M S 320 IO :

I-5-I - Adressage direct :

Dans ce mode d'adressage , sept (07) bits du mot d'instruction assemblés avec le pointeur de page de données forment l'adresse mémoire de données .

La déscription de ce mode est :



Un zéro "0" sur le bit sept (07) définit le mode d'adressage direct . Les bits de 0 à 6 contiennent les adresses mémoire .
La zone de 0 au 7ème bit de l'adresse mémoire peut adresser jusqu'à I28 mots de la mémoire de données .

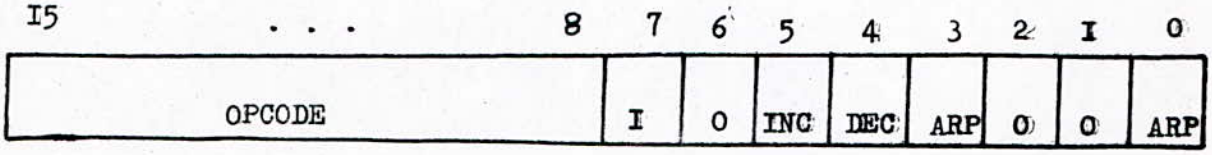
L'utilisation du pointeur de page de mémoire de données permet l'adressage de tous les I44 mots de la mémoire de données , soit donc (I28 + I6) mots .

On peut adresser toutes les instructions avec le mode direct à l'éxcéption des instructions qui s'adressent uniquement avec le mode immédiat . (Voir mnémonique)

I-5-2 - Adressage indirect :

Dans l'adressage indirect , les huit (08) bits les moins significatifs de chacun des deux registres auxilliaires ARO et ARI forment l'adresse mémoire de données . Le pointeur de registre auxilliaire (ARP) séléctionne l'un des deux registres ARI , ces dérniers peuvent être incrémentés et décrémentés automatiquement en parallèle avec l'éxécution de l'instruction adressée en mode indirect .

La déscription de ce mode est :



Un "I" logique sur le bit 7 définit l'adressage en mode indirect .
Le code opération est contenu dans les bits 8 à I5 .
Les bits de 0 à 6 controlent l'adressage indirect .
Les bits de 0 à 3 controlent le pointeur de registres auxilliaires (ARP) .
Un "I" logique sur le bit 0 signifie que l'adresse mémoire est contenue dans le registre auxilliaire ARI , et un "0" logique sur ce bit signifie que l'adresse mémoire est contenue dans le registre auxilliaire ARO .
Si le bit 3 est à l'état logique "0" , le contenu du bit 0 est chargé dans l'ARP aprés éxécution de l'instruction en cours .

Si le bit 3 est à l'état logique "I" , le contenu de l'ARP réste inchangé .

Les bits 4 et 5 controlent les registres auxilliaires ;
Si le bit 5 est à l'état logique "I" , l'ARP définit le registre auxilliaire qui va être incrémenté aprés éxécution .
Si le bit 4 est à l'état logique "I" , l'ARP définit le registre auxilliaire qui va être décrémenté aprés éxécution .

Les bits 6 , 2 et I sont toujours mis à l'état logique "0" .

I-5-3 _ Adressage immédiat :

Le T M S 320 I0 contient cinq (05) instructions adressées en mode immédiat qui sont : LDPK , LARK , MPYX , LACK , LARP .
Dans ces instructions l'opérande est contenue dans le mot d'instruction .

I-6 _ INTERFACAGE ET CONTROLE :

I-6-I _ Entrées / Sorties :

Les dispositifs d'interfaçage vers les périphériques utilisent I28 bits d'entrées et I28 bits de sorties constitués de huit (08) ports d'entrées de I6 bits , multipléxés également .
(Voir figure (5)) .

Le T M S 320 I0 possède un bus d'adresse extérieur de I2 lignes parallèles et un bus de données extérieur de I6 lignes parallèles .
Ce dérnier est utilisé aussi pour pérformer des fonctions d'entrées / sorties à la vitesse de 40 millions de bits par seconde .

Les entrées / sorties sont séparées de la mémoire centrale et permettent d'adresser directement jusqu'à huit (08) périphériques .

Deux instructions , IN et OUT , entrainent l'entrée et la sortie des données vers et depuis le T M S 320 I0 à travers le bus de données . Ces instructions contiennent un port d'adresse de 3 bits lequel est multipléxé sur les trois (03) lignes les

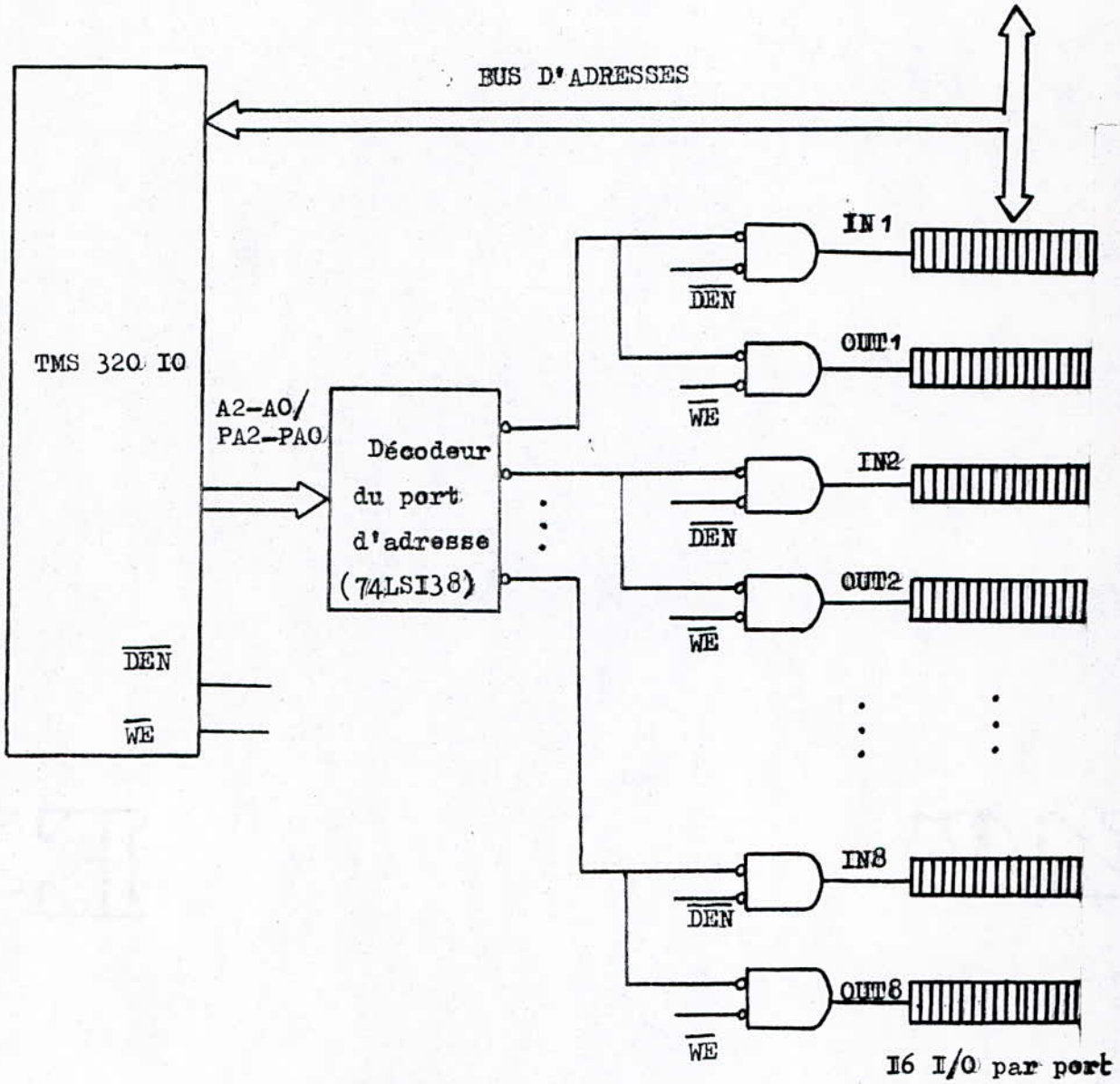


Figure (5) - Interfaçage du TMS 320 IO

moins significatives (A2 / PA2 - A0 / PA0) de l'adresse , le reste des lignes d'adresses sont cependant maintenues à l'état logique "0" .

Les entrées ou les sorties sont distinguées par les broches DEN et WE .

L'éxécution de l'instruction IN génère un créneau sur la broche DEN qui fait entrer les données dans la RAM à travers le bus de données . L'éxécution de l'instruction " OUT " fait sortir les données de la RAM vers le bus de données en générant un créneau sur la broche WE .

En plus , deux (02) instructions , TBLR (table read) et TBLW (table write) , permettent un transfert entre les mémoires de données et de programme . TBLR lit un mot de la mémoire programme et le transfere vers la RAM de données . TBLW copie un mot de la RAM de données dans la mémoire éxtèrne de programme . Dans ces deux (02) instructions , l'adresse de la mémoire de données est l'opérande de l'instruction , et l'adresse de la mémoire programme est contenue dans l'accumulateur .

I-6-2 - Controle de programme :

En mode microcalculateur , le T M S 320 IO peut accéder à 2560 mots de la mémoire programme extèrne et à I536 mots de la mémoire intèrne . Pour faciliter cette habilité , les sorties du compteur programme sont liées à travers un buffer , aux broches AII - A0 , un créneau de sortie (MEN) est généré pour chaque cycle machine afin d'accéder à la mémoire éxtèrne , sauf que lorsque les instructions IN , OUT ou TBLW sont en cours d'éxécution .

Notons que les données de la mémoire éxtèrne sont transferées vers le T M S 320 IO par le bus de données DI5 - D0 et le temps maximum d'accés à la mémoire éxtèrne est de I00 ns .

I-6-3 - Interruptions :

Les interruptions sont gérées avec sauvegarde du contexte . Un front descendant sur la broche INT génère une interruption et met à "I" l'indicateur d'interruption INTF .

Quand l'interruption est prise en considération le T M S 320 IO incrémente le PC dans la pile et se branche à la position 2 de celle-çi .

Les positions O et I sont résérvées pour la remise à zéro (RESET). Le T M S 320 IO possède un bit de mode d'interruption (INTM) qui est mis à " I" par l'instruction DINT et remis à zéro par l'instruction EINT , quand ce bit est à l'état " I " , les interruptions suivantes sont inhibées par le T M S 320 IO .

Une fois l'interruption est prise en considération , l'indicateur d'interruption (INTF) est mis automatiquement à zéro et le bit du mode d'interruption à " I " .
L'instruction EINT permet un stockage et une reprise de tous les registres excépté le registre P .

L'entrée BIO permet aussi une synchronisation logicielle du processeur pour l'attente d'un évenement .
Notons enfin qu'avant qu'une interruption ne soit gérée , le contenu du registre P sera accumulé et l'instruction en cours achevée .

I-7 L'ASSEMBLEUR :

Le programme en langage assembleur , codé par le programmeur , est appelé programme source . Avant qu'il ne soit éxécuté par le calculateur ; ce programme source doit être traité par le programme macro-assembleur pour obtenir un programme en langage machine afin qu'il puisse être éxécuté par le simulateur .
Ce traitement est appelé assemblage , il consiste à assembler les valeurs binaires qui corréspondent au code opération du mnémonique , avec l'information de l'adresse binaire pour former l'instruction en langage machine . [11]

Les directives de l'assembleur controlent le procédé qui permet de faire un programme en langage machine à partir du programme en langage assembleur . Parmis ces directives , ils existent celles qui affectent le compteur position , la sortie

de l'assembleur , l'initialisation des constantes et celles qui permettent la liaison entre programmes .

L'assembleur traite le code source en deux étapes ; dans la première étape il maintient le compteur position qui définit les adresses de la mémoire programme affectées aux mots résultants du code objet , il construit une table de symboles et produit une copie du code source pour le traitement de la seconde étape . Dans cette deuxième étape , il lit la copie de la source et assemble le code objet en utilisant les codes opérations et la table des symboles produite durant la première étape .

Les caractères acceptés par l'assembleur du T M S 320 IO sont des caractères en code ASCII et en code HOLLERITH .

Ils existent cependant cinq (05) types de constantes reconnaissables par l'assembleur , chacune d'elle est maintenue intérieurement comme une quantité de I6 bits . Ces constantes sont les suivantes :

- Constante entière décimale :

Elle est écrite comme une suite de digits décimaux . La plage des valeurs des constantes décimales entière est comprise entre (- 32768) et (+ 65535) .

- Constante entière binaire :

Elle est écrite au maximum sur une suite de I6 digits binaires (0 / I) précédée par le point d'interrogation " ? " .

- Constante entière héxadécimale :

Elle est écrite au maximum sur une suite de 4 digits héxadécimaux (0 à F) précédée par le signe " > " .

- Constante de caractères :

Elle est écrite en I ou 2 caractères seulement , mise entre côtes .

Les caractères sont représentés intérieurement par huit (08) bits en code ASCII .

- Constante de temps d'assemblage :

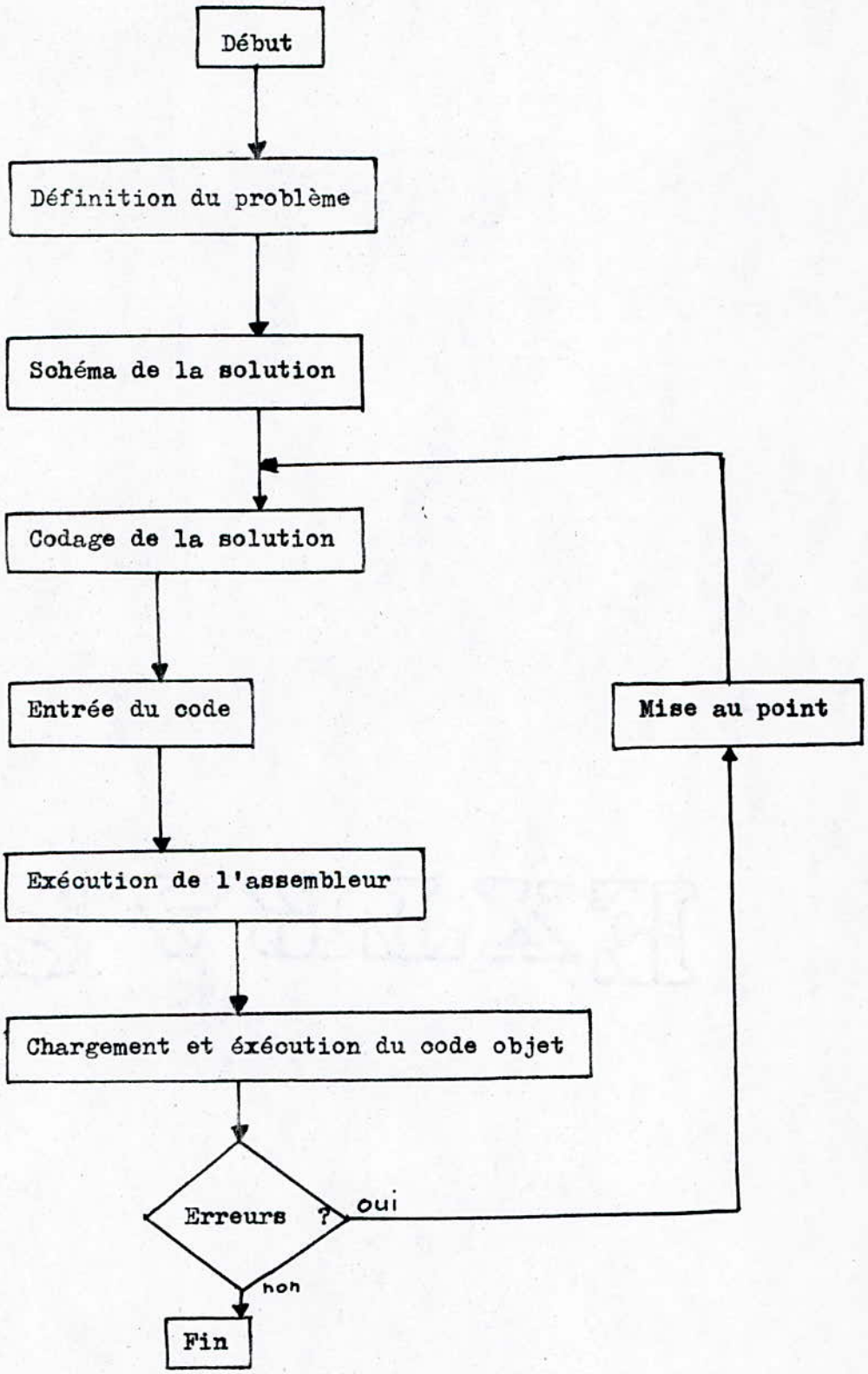
C'est un symbole donnant une valeur par la directive EQU .
La valeur de ce symbole est détérminée au temps de l'assemblage pour être absolue ou bien transférée selon l'expréssion de transfert .

Procédé de developpement d'une carte :

Pour illustrer la place du langage assembleur dans le developpement des programmes , on considère les étapes suivantes (voir figure (6)) :

- Définition du problème .
- Schéma de la solution du problème .
- Codage de la solution en écrivant les instructions en assembleur qui corréspondent aux étapes du schéma de la solution .
- Préparation du programme source en écrivant les instructions .
- Exécution de l'assembleur afin d'assembler le code objet du langage machine corréspondant au programme source ;
- Mise au point de la résolution du code objet en chargeant , éxécutant et en faisant les corréctions conséquentes indiquées .
- Répéter les étapes 5 et 6 jusqu'à ce qu'aucune corréction n'est demandée .

- Figure (6) : Organigramme définissant le procédé de développement .



I-8 - SYSTEMES ET LOGICIELS DE DEVELOPPEMENT :

XDS / 320 est une gamme complette d'outils de developpement pour la fammille T M S 320 . Ces outils sont tout à fait indépendants du type de calculateur utilisé , tant sur le plan matériel que logiciel . Ils peuvent être connéctés à plusieurs calculateurs tels que le TI 990 , IBM , ou VAX / VMS .

La famille T M S 320 dispose de cinq (05) moyens de développement : Une carte EVM 320 , un macro-assembleur / éditeur de liens , un simulateur , et un émulateur . [12]

I-8-1 - Carte EVM 320 (Evaluation modul) :

C'est un outil permettant à l'utilisateur d'évaluer pour une première phase les pérformances dynamiques du T M S 320 , en vue d'une application .

La communication vers l'ordinateur principal , et vers plusieurs périphériques s'éfféctue par la carte EVM . deux ports EIA permettent à cette carte d'être connécter à l'ordinateur principal et au términal , elle peut aussi être connéctée à une imprimante sur un port , tandis que l'autre port est connécté au términal ou à l'ordinateur principal .

I-8-2 - Le Macro-assembleur / Editeur de liens :

Le macro-assembleur traduit le langage assembleur en langage machine éxécutable directement par le simulateur ou l'émulateur . L'assembleur permet au programmeur de travailler avec le mnémonique plutôt que les instructions machines héxadécimales , en donnant des références aux positions mémoires avec des adresses symboliques ; ceci permet à la réalisation du logiciel d'être plus éfficace et sûre .

L'éditeur de liens permet à un programme d'être éxécuté par des modules séparés , et assemblé par la suite pour former un programme complet , cela permet l'utilisation des mêmes modules pour différents problèmes . L'éditeur de liens affécte des valeurs au code établit , en créant un fichier objet qui sera éxécuté par le simulateur ou l'émulateur .

I-8-3 - Emulateur :

Il constitue un système autonome et étendu qui possède tous les moyens nécéssaires pour la mise au point d'une application .
Il est muni d'un moniteur de mise au point ayant un jeu de 65 commandes et un assembleur / assembleur inverse .
Deux liaisons RS 232 lui permettent la connection vers un terminal , un calculateur , une imprimante ou un programmeur .

L'émulateur possède d'autres caractéristiques qui sont :

- Support logiciel de macro-assembleur / éditeur de liens sur un calculateur éxtèrne .
- Dix (IO) points d'arrêts logiciels avec toutes les commandes nécéssaires de positionnement , de visualisation et d'éffacement .
- Deux (02) points d'arrêts matériels dont l'un est sur l'accés mémoire et l'autre sur les entrées / sorties .
- Opération en mode multiprocesseur ; possibilité de connécter jusqu'à neuf (09) émulateurs en série .

I-8-4 - Simulateur :

C'est un programme logiciel permettant de simuler les opérations du T M S 320 et de vérifier grâce à un mode de mise au point , la bonne éxécution d'un programme .

Le simulateur utilise le code objet du T M S 320 IO PRODUIT par le macro-assembleur / éditeur de liens . Il est en général disponible sur des rubans magnétiques pour les systèmes d'éxploitation du VAX / VMS et sur des disquettes pour les systèmes d'éxploitation du TI PC / MC - DOS et IBM PC / PC - DOS . Il permet l'éxécution des programmes à une vitesse de 4000 instructions par seconde pour le VAX / VMS et de 400 instructions par seconde pour les systèmes TI PC / MC - DOS et IBM PC / PC - DOS , et requière 256 K octet de mémoire .

Afin de simuler les dispositifs d'entrées / sorties qui seront connéctés au processeur , les fichiers d'entrée et de sortie peuvent être associés avec le port d'adresses des instructions d'E/S .

Pour simuler une interruption du signal , l'indicateur d'interruption peut être activé périodiquement dont l'intervalle de temps est définit par l'utilisateur . Avant l'initialisation de l'éxécution du programme , les points d'arrêts peuvent être définis .

Durant l'éxécution du programme , les registres intèrnes et les mémoires de simulation sont modifiés selon les instructions éxécutées . Dés que l'éxécution du programme est suspendue , les registres intèrnes et les deux (02) mémoires , programme et données , peuvent être inspéctées et / ou modifiées .

B/ SIMULATION

INTRODUCTION :

Nous disposons d'une disquette contenant trois programmes logiciels principaux exécutables : le simulateur SIM . EXE le macroassembleur XASM3 . EXE et l'éditeur de liens LINKER . EXE . Ces trois programmes nous pérméttent l'éxécution de tous les programmes écrits en langage assembleur propre au TMS 320 IO .

I-I -MISE EN OEUVRE DE LA SIMULATION SUR LE MICRO-ORDINATEUR OLIVETTI M 24 :

Pour la mise en oeuvre de cette simulation , un chargement au préalable des trois programmes de la disquette dans le disque dur est préférable . ceci se fait comme suit :

C > COPY A : XASM3 . EXE C : XASM3 . EXE / V <CR>

C > COPY A : SIM . EXE C : SIM . EXE / V <CR>

C > COPY A : LINKER . EXE C : SIM . EXE / V <CR>

tels que :

XASM3 . EXE	représente le nom du fichier du macroassembleur éxécutable .
SIM . EXE	représente le nom du fichier du simulateur éxécutable .
LINKER . EXE	représente le nom du fichier de l'éditeur de liens éxécutable .
/ V	pour vérifier le bon chargement .
<CR>	commande RETURN du micro ordinateur .

Aprés avoir éffectué le chargement , les étapes à suivre pour l'éxécution d'un programme sont les suivantes:

I-I-I - Ecriture du programme en langage assembleur dans un fichier crée par l'instruction EDIT :

C > EDIT [non du fichier][. ASM] <CR>

Note : Le nom du fichier doit être écrit avec l'éxtention [. ASM] qui signifie langage assembleur .

I-I-2 - Exécution du macroassembleur :

Ce logiciel s'éxécute comme suit :

C > XASM3 <CR>

Le macroassembleur demande ensuite les noms des fichiers source , listing et code objet à créer :

XASM3 [source file . ASM] : <nom du fichier> . ASM

XASM3 [listing file . LST] : <nom du fichier> . LST

XASM3 [objet file . MPO] : <nom du fichier> . MPO

Notons que ASM , LST et MPO signifient réspéctivement assembleur , listing et code objet .

La création de ces fichiers se fait automatiquement dans le disque dur où ils seront sauvegardés .

I-I-3 - Exécution du simulateur :

Pour éxécuter le simulateur du TMS 320 IO , on utilise la commande :

C > SIM <CR>

Le système doit maintenant éxécuter le simulateur , toutes

les positions mémoires de programme et de données sont ainsi initialisées à zéro .

Le simulateur demande ensuite le mode de traitement qui sera simulé , mode microprocesseur (MP) ou mode microcaloulateur (MC).

Dans le cas du TMS 320 IO , on séléctionne le mode microprocesseur (MP) en faisant entrer un zéro (0) ou par la commande RETURN <CR> .

Le chargement de la mémoire de programme par le fichier écrit en code objet et le chargement de la mémoire de données , des registres intèrnes ainsi que l'éxécution , s'éffectue à l'aide des commandes du simulateur .

I-2 — COMMANDES DU SIMULATEUR :

Les commandes du simulateur du TMS 320 IO éxécutent les fonctions spécifiques du simulateur . Les cinq commandes principales sont les suivantes :

- DM : affichage du menu des commandes .
- BH : indique les commandes pour les points d'arrêts .
- MH : pour inspécter ou modifier les mémoires de programme et de données .
- RH : pour inspécter ou modifier les registres et les indicateurs (d'intérruption , overflow , ...)
- IOH : indique les commandes d'E/S .

Notes :

I - Les commandes doivent êtres entrées en majuscule , sinon un message indiquant une erreur aura lieu comme suit :

" INVALID COMMAND "

2 - Chaque commande du simulateur est éxécutée à l'aide de la commande RETURN <CR> du micro ordinateur .

I_3 CHARGEMENT DES FICHIERS :

Les fichiers seront chargés dans le simulateur à l'aide de la commande L (LOAD) comme suit :

C > SIM <CR>

(C) Copyright Texas Instruments incorporated 1984

SIMULATION OF THE TMS 320 IO

VERSION # I.6

0 - MICROPROCESSOR MODE (ADDR 0 - I535 , OFF CHIP)

I - MICROCALCULATEUR MODE (ADDR 0 - I535 , ON CHIP)

0 <CR>

YOU ARE IN THE MICROPROCESSOR MODE (ADDR 0 - I535 , OFF CHIP)

ENTER COMMAND (HELP = <CR>)

L <CR>

ENTER NEW OBJECT FILE

[Nom du fichier] . MPO <CR>

* * * * LOADING PROGRAMME " NO$IDT " * * * *

ENTER COMMAND (HELP = <CR>)

ENTER COMMAND (HELP = <CR>)

Le procédé de chargement de la ROM est le même que celui de la RAM seulement on doit faire entrer la commande ROM au lieu de RAM .

Note :

La commande Q peut aussi être utilisée pour quitter complètement le simulateur :

ENTER COMMAND (HELP = <CR>) :

Q <CR>

Aussitôt que le fichier écrit en code objet sera chargé dans le simulateur , mèttre un arrêt de contrôle en utilisant la commande BIAQ . Cette commande est utilisée pour arréter l'éxécution à la dérnière instruction du programme , sans elle le simulateur continuera l'éxécution jusqu'à la dérnière case mémoire de programme . Aprés ceci on utilise la commande R (RUN) pour éxécuter la simulation .[13]
La procédure est la suivante :

ENTER COMMAND (HELP = <CR>) :

BIAQ <CR>

BREAK ON INSTRUCTION ACQUISITION

ENTER THE ADDRESS (IN HEX)

3 <CR>

ENTER COMMAND (HELP = <CR>) :

R <CR>

I-4 - CHARGEMENT DE LA MEMOIRE DE DONNEES :

ENTER COMMAND (HELP = <CR>)

RAM <CR>

ENTER STARTING ADDRESS (IN HEX)

0 <CR> début de la position mémoire à charger en héxadécimal (0 par exemple) .

0 = 0 valeur courante contenue dans cette position .

7 <CR> changer cette valeur par (exemple:7) .

0 = 7 cette position mémoire contient maintenant la valeur 7 .

+ <CR> passage à la position suivante .

I = 0 la position I contient la valeur 0

5 <CR> modifier cette valeur par (exemple:5).

I = 5 la position I contient maintenant la valeur 5 .

<CR> passage à la position suivante et ainsi de suite .

2 = 0

8 <CR>

2 = 8

<CR>

3 = 0

Q <CR> Fin de chargement .

L'éxécution s'arrète quand le simulateur aurait éxécuté la position (exemple 3) de la mémoire de programme .

I-5 - SIMULATEUR DES FICHIERS JOURNAUX :

Les fichiers journaux peuvent être des outils puissants pour améliorer la qualité de la séssion de simulation . Ils maintiennent l'enregistrement de la séssion de simulation afin qu'il puisse être réexécuté durant une autre séssion . Ils réduisent cependant le temps pour recréer la séssion passée . Ils sont particulièrement utiles quand les mêmes traitements sont répétés plusieurs fois et ils peuvent être utilisés pour faire l'attribution des E / S , placer des séquences de contrôle , etc ... [13]

Création du fichier journal :

Le fichier journal est crée par la commande JF (Select Journal File) . Toutes les commandes qui suivent sa création seront stockées à l'intérieur jusqu'à l'introduction de la commande Q (Quit) .

Exemple :

ENTER COMMAND (HELP = <CR>)

JF <CR>

A JOURNAL FILE NAME HAS NOT BEEN CREATED

ENTER FILE NAME

[Nom du fichier . Extension] <CR>

ENTER COMMAND (HELP = <CR>)

CHAPITRE II

ETUDE DES METHODES D'ANALYSE DE LA PAROLE

II-1 - Analyse cepstrale

II-2 - Analyse par prédiction linéaire

II-3 - Comparaison des deux méthodes d'analyse

Introduction :

Le signal temporel de la parole est trés chaotique et inéxploitable puisque l'introduction des déphasages entre les composantes spéctrales du signal n'affècte pas l'intélligibilité du son (l'oreille est insensible aux variations de phase) et son spéctre corréspondant , mais l'allure temporelle change , donc on utilise le traitement spéctrale du signal de la parole .

L'analyse de la parole pérmet la diminution de redondance du signal vocal et assure l'éxtraction des paramêtres pértinents de celui-çi . Le volume de mémoire nécéssaire au traitement automatique de ce signal dépend du type d'analyseur choisi .

Avant l'application d'une méthode d'analyse , un prétraitement sur le signal de la parole est nécéssaire , ce prétraitement consiste en une préaccentuation afin de rétablir ce signal tel qu'il était avant de déboucher des lèvres , et en un fenêtrage pour limiter l'amplitude des rebonds fréquentiels dûs à la durée limitée des trames .

Les méthodes d'analyse les plus utilisées en vue de la reconnaissance de la parole sont la méthode d'analyse cepstrale et la méthode d'analyse par prédiction linéaire .

II-I - ANALYSE CEPSTRALE :

Introduction :

Une des approches fondamentales de la théorie acoustique est la représentation du signal de la parole comme étant la sortie d'un système linéaire dont les propriétés varies lentement avec le temps (le conduit vocal) , pour un petit ségment du signal de parole , il peut être conséderé comme étant généré par l'éxcitation d'un système linéaire invariant dans le temps .

La source d'éxcitation , qui peut être soit un train d'impulsions périodique (vibration des cordes vocales pour les sons voisés) ou un bruit aléatoire (pour les sons non voisés) , module le conduit vocal , il ya cependant combinaison par convolution de l'éxcitation et de la réponse impulsionnelle du système linéaire

invariant dans le temps .

Le problème de l'analyse de la parole peut donc être considéré comme un problème de séparation des composantes convoluées , qu'on appelle " déconvolution " , c'est à dire la séparation des spèctres du signal et de la source d'éxcitation . Cette séparation est réalisée par des traitements homomorphiques dont le calcul cepstral est une illustration .

II-I-I-Déconvolution :

L'intéret de la déconvolution est d'obtenir le spèctre du signal de parole émis , séparé de la source d'éxcitation qui introduit des ondulations sur le spèctre caractéristique du son émis . [8]

Soient h(n) le signal issu de la source d'éxcitation et x(n) la fonction de transfert du conduit vocal indépendante de la première , ces deux (02) fonctions convoluent et donnent le signal sonore y(n) qui débouche de l'appareil phonatoire dont l'équation est :

$$y(n) = \sum_{k=-\infty}^{+\infty} h(n-k) \cdot x(k) = h(n) * x(n)$$

L'opération (D) de déconvolution donne :

$$\begin{aligned} D\left[y(n)\right] &= D\left[h(n) * x(n)\right] \\ &= D\left[h(n)\right] + D\left[x(n)\right] \\ &= h'(n) + x'(n) \\ &= y'(n) \end{aligned}$$

Où : y'(n) , h'(n) et x'(n) représentent réspéctivement : $D\left[y(n)\right]$, $D\left[h(n)\right]$ et $D\left[x(n)\right]$.

On sait qu'un produit de convolution se transforme en une multiplication par la transformée en Z .

$$y(n) = h(n) * x(n) \Longrightarrow Y(z) = H(z) \cdot X(z) .$$

Il suffit donc d'appliquer le logarithme sur la transformée en Z pour passer d'un produit à une somme , ainsi un produit de convolution de deux (02) signaux peut être transformé en une somme de ces mêmes signaux représentés dans un domaine particulier (quéfrences) où ils peuvent être séparables comme le montre la figure (I) . [10]

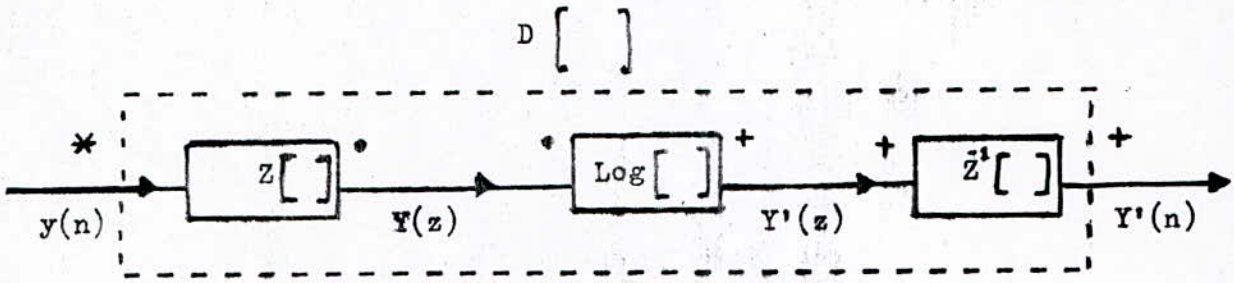


Figure (1) .
Déconvolution homomorphique .

$$Y'(z) = \text{Log}\,(\,Y(z)\,) = \text{Log}\,(\,H(z) \cdot X(z)\,)$$

$$= \text{Log}\,(\,H(z)\,) + \text{Log}\,(\,X(z)\,)$$

$$= H'(z) + X'(z)$$

Remarque : Cette relation n'est pas valide en règle générale puisque la partie imaginaire du logarithme d'un nombre complèxe est indéfinie .

$$Y'(z) = \text{Log}\,(\,Y(z)\,) = \text{Log}\,\left|\,Y(z)\right| + j\,\arg\,(\,Y(z)\,)$$

II-I-2- Cepstre :

Le cepstre complexe est la transformée de Fourier invèrse du logarithme du spectre du signal .

$$Y'(n) = \frac{I}{2\pi} \int_{-\pi}^{+\pi} Y'(e^{jw}) . e^{jwn} dw . \quad (I)$$

Si on considère seulement la partie réelle , on obtient la relation du cepstre C(n) suivante :

$$C(n) = \frac{I}{2\pi} \int_{-\pi}^{+\pi} \text{Log} \left| Y(e^{jw}) \right| . e^{jwn} dw . \quad (2)$$

$$\text{Puisque} \quad C(n) = \frac{Y'(n) + Y'(-n)}{2}$$

donc le cepstre peut être calculé par la transformée en cosinus du logarithme du spéctre complèxe du signal :

$$C(n) = \frac{I}{2\pi} \int_{-\pi}^{+\pi} \text{Log} \left[Y(e^{jw}) \right] . \text{Cos}(wn) dw . \quad (3)$$

II-I-3- Propriétés des coéfficients cepstraux :

La forme générale de Y(z) est :

$$Y(Z) = \frac{A.Z^{r} \prod_{k=1}^{n_1} \left[I-a(k).Z^{-1} \right] . \prod_{k=1}^{n_0} \left[I-b(k).Z \right]}{\prod_{k=1}^{N_1} \left[I-c(k).Z^{-1} \right] . \prod_{k=1}^{N_0} \left[I-d(k)Z \right]}$$

où a(k) et c(k) sont réspectivement les zéros et pôles intérieurs au cercle unité , b(k) et d(k) ceux extérieurs , et tous sont inférieurs à I .

En posant $Z = e^{jw}$ on peut tirer les valeurs des cepstres complèxes puisque $Y'(Z) = \text{Log}\ (\ Y(Z)\ )$ et en négligeant Z^r car $|Z| = I$.

$$Y'(n) = \text{Log } A \qquad \text{Si } n=0$$

$$Y'(n) = \sum_{k=1}^{N1} \frac{c(k)^n}{n} - \sum_{k=1}^{M1} \frac{a(k)^n}{n} \qquad \text{Si } n > 0$$

$$Y'(n) = \sum_{k=1}^{Mo} \frac{b(k)^{-n}}{n} - \sum_{k=1}^{No} \frac{d(k)^{-n}}{n} \qquad \text{Si } n < 0$$

Si $Y'(Z)$ a des pôles et des zéros seulement à l'intérieur du cercle unité , alors :

$$Y'(n) = 0 \qquad \text{pour } n < 0 .$$

De tels signaux sont dits à phase minimale , et le calcul du cepstre complèxe revient à calculer le cepstre normal pour les signaux à phase minimale , donc les relations du cepstre sont :

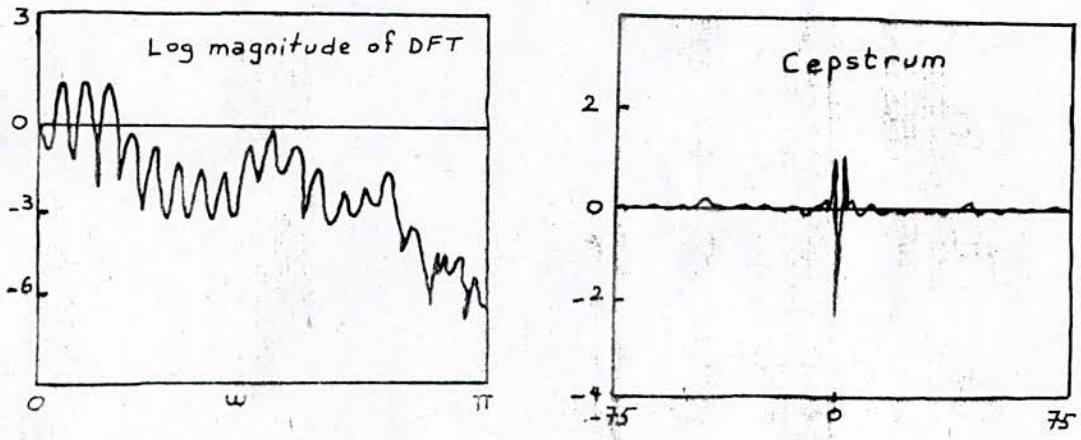
$$C(n) = 0 \qquad \text{Si } n < 0$$

$$C(n) = Y'(n) \qquad \text{Si } n = 0$$

$$C(n) = \frac{Y'(n)}{2} \qquad \text{Si } n > 0$$

Pour n=0 le tèrme Log A définit l'énérgie du spèctre , et pour $n \neq 0$ les coéfficients cepstraux sont de même nature et dépendant que de l'information sur les pôles et les zéros , ils sont donc indépendants del'énérgie , ce qui conduit à omettre le terme Log A .[8]

En pratique on utilise peu de coéfficients cepstraux puisqu'ils décroissent en I/n , où (n) est leur rang .
La figure (2) représente le spèctre d'un signal pour un son voisé , et son cepstre corréspondant .



Spéctre voisé — Cepstre corréspondant

Figure (2)

Sur le spéotre apparaissent des ondulations rapides , dues au fondamental , supérposées à des ondulations moins rapides dues aux forments , et sur le cepstre , les premiers coéfficients cepstraux décrivent la forme des ondulations lentes , donc la nature du son émis . Le fondamental dont les ondulations sont plus rapides que ceux dues à la fonction de transfert du conduit vocal est rejeté loin dans le rang de ces coéfficients .

II-I-4 - Echelle MEL :

Les études phisiologiques et pérséptives de l'oreille semblent indiquer quelle est sensible à une échelle près**q**ue logarithmique de la fréquence , cela veut dire que les informations pértimentes se répartissent dans des zones croissantes de façon exponentielle , en fonction de la hauteur de la fréquence . Pour favoriser ces zones qui ont les plus grandes densités d'informations , on utilise une échelle quasi-logarithmique appelée échelle " MEL " sur laquelle se fait le calcul cepstral , voir figure (3) çi-dessous :

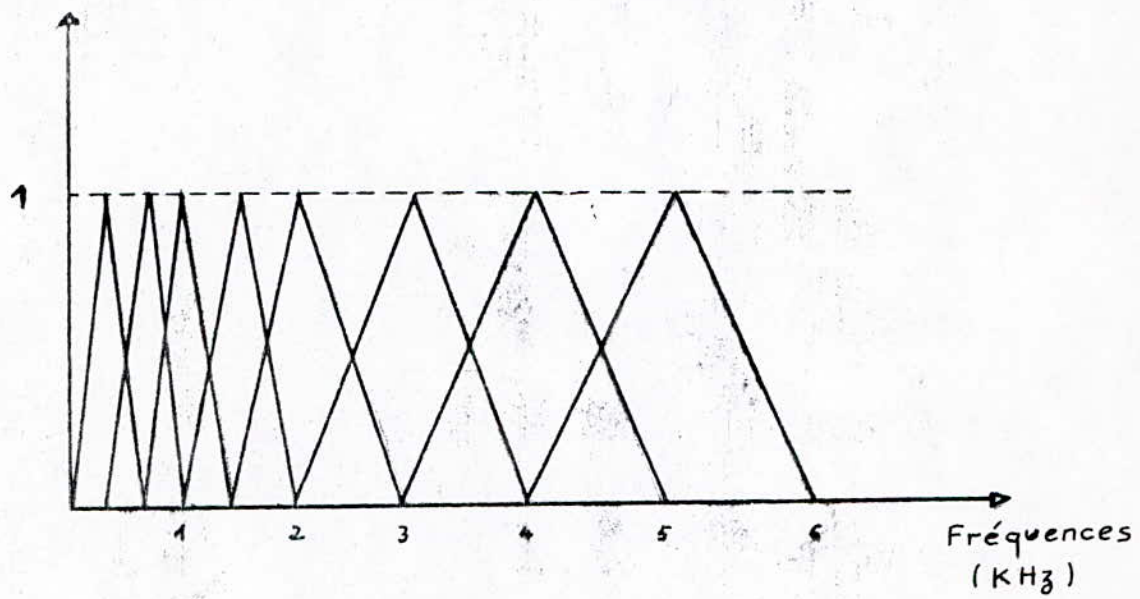


Figure (3) - Filtres MEL

Cette échelle est linéaire jusqu'à I KHZ et logarithmique au delà , en général on utilise un nombre de filtres de l'ordre d'une dizaine , de largeurs uniformes , et de formes triangulaires .
Pour le premier KHZ , ils sont décalés les uns des autres d'une moitié de la largeur des filtres , au delà de I KHZ les décalages entre les filtres forment une suite géométrique de raison supérieur à I (valeur typique I,5) . [2]
Sur cette échelle le cepstre s'écrit comme suit :

$$\Pi FCC(n) = \frac{I}{F} \sum_{k=1}^{F} Log \left[E(k) \right] . Cos \left[n . \left(\frac{k-I}{2} \right) . \frac{2\pi}{F} \right]$$

où (n) est le rang du coéfficient cepstral ,
(k) le numéro du filtre ,
E(k) l'énérgie du filtre ,
et (F) le nombre total des filtres .

II-2 - ANALYSE PAR PREDICTION LINEAIRE :

Introduction :

La méthode d'analyse par prédiction linéaire est l'une des méthodes les plus puissantes d'analyse de la parole .
Elle est fondée sur un modèle simple (filtre numérique récursif) de production de la parole constituant une bonne approximation du système phonatoire .

II-2-I - Idée de base :

L'idée de base de cette méthode repose sur le fait qu'un échantillon S(n) , de la parole peut être prédit approximativement à partir d'une combinaison linéaire des " p " échantillons le précédant immédiatement , compte tenu de l'éxcitation U(n) .

Les échantillons sont cependant pris sur une fenêtre temporelle de IO à 20 ms du signal S(t) de la parole .
La taille N de ces échantillons est généralement entre IOO et 250 .

Notons que cette méthode se base essentiellement sur deux (02) hypothèses :

- Les variations du conduit vocal peuvent être approchées par une succéssion de configurations stationnaires dont la durée est de l'ordre de IO à 25 ms appelée " trame " .
- Pendant chaque trame , la source d'éxcitation est constante .

II-2-2 - Illustration de la méthode :

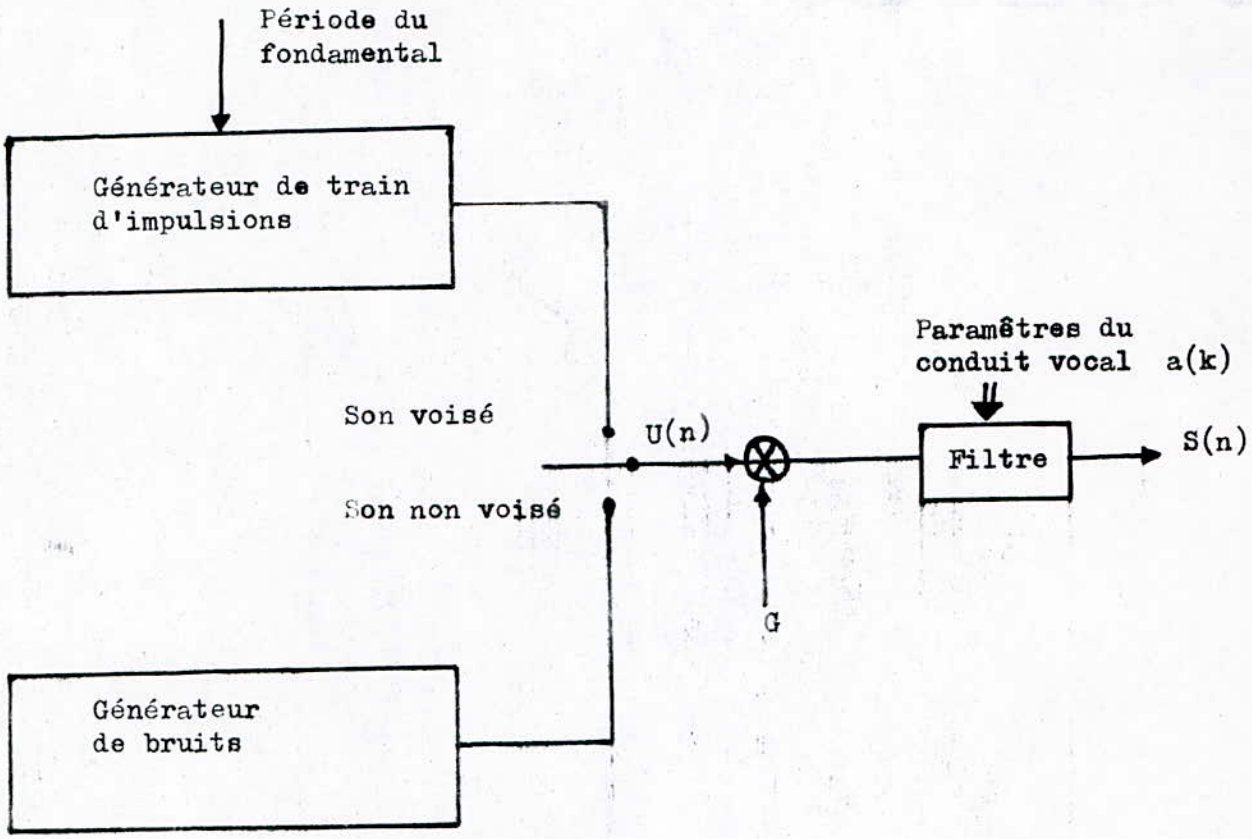


Fig 4 - Modèle de production de la parole -

La figure çi-dessus illustre un modèle de production de la parole pour chaque fenêtre de I0 à 20 ms du signal de la parole .

Soit : $$S(n) = \sum_{k=1}^{p} a(k) S(n-k) + G \cdot U(n) \quad (1) \qquad n \in [0, N-1]$$

Où:

- G : le gain
- U(n) : éxcitation du système
- S(n) : échantillon de la parole
- a(k) : paramètres du conduit vocal appelés " coéfficients de prédiction linéaire " .

La valeur prédite de l'échantillon S(n) est :

$$\hat{S}(n) = \sum_{k=1}^{p} a(k)\, S(n-k) \qquad n \in [0, N-1]$$

L'erreur de prédiction linéaire est par conséquent :

$$e(n) = S(n) - \hat{S}(n)$$

$$e(n) = S(n) - \sum_{k=1}^{p} a(k)\, S(n-k) = G \cdot U(n)$$

Le passage à la transformée en Z donne :

$$E(Z) = S(Z) - \sum_{k=1}^{p} a(k)\, Z^{-k}\, S(Z) = G \cdot U(Z)$$

$$E(Z) = S(Z) \left[I - \sum_{k=1}^{p} a(k)\, Z^{-k} \right] = G \cdot U(Z)$$

On pose :

$$A(Z) = I - \sum_{k=1}^{p} a(k) \cdot Z^{-k} \quad \text{et} \quad H(Z) = \frac{G}{A(Z)}$$

d'où :

$$E(Z) = S(Z) \cdot A(Z)$$

$$S(Z) = \frac{G}{A(Z)} \cdot U(Z) = H(Z) \cdot U(Z)$$

Nous constatons que l'érreur de prédiction linéaire est la sortie d'un système ayant comme fonction de transfert A(Z) appelé filtre invèrse , et S(Z) la sortie d'un filtre récursif ayant comme fonction de transfert H(Z) simulant le conduit vocal . De même nous remarquons que H(Z) possède que des pôles , ces pôles sont les coéfficients a(k) de prédiction linéaire . Ces dérniers nous renseignent donc le mieux possible sur le comportement du conduit vocal .[10]

En général l'ordre " p " de prédiction linéaire est de I2 . Divers méthodes sont utilisées pour la détermination des coéfficients a(k) de prédiction linéaire .

II-2-3 - Méthode d'autocorrélation :

Soit En l'érreur quadratique totale de prédiction linéaire :

$$En = \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1+p} \left[S(n) - \hat{S}(n) \right]^2$$

$$En = \sum_{n=0}^{N-1+p} \left[S(n) - \sum_{k=1}^{p} a(k)\, S(n-k) \right]^2$$

Pour une bonne éstimation des échantillons $\hat{S}(n)$, l'érreur quadratique totale En doit être la plus petite possible .
Il s'agit donc de trouver les valeurs optimales des coéfficients a(k) qui minimisent le mieux possible cette érreur . Cependant on fait appel aux formules mathématiques des dérivées partielles :

$$\frac{\delta En}{\delta a(i)} = \sum_{n=0}^{N-1+p} a \left[S(n) - \sum_{k=1}^{p} a(k) \cdot S(n-k) \right] \cdot S(n-i) = 0 \quad (1)$$

d'où :

$$\sum_{n=0}^{N-1+p} S(n-i) \cdot S(n) = \sum_{k=1}^{p} a(k) \sum_{n=0}^{N-1+p} S(n-i) \cdot S(n-k) \quad (2)$$

soit :

$$\phi(i,k) = \sum_{n=0}^{N-1+p} S(n-i) \cdot S(n-k) \quad (3) \qquad \begin{matrix} I \leqslant i \leqslant p \\ I \leqslant k \leqslant p \end{matrix}$$

par changement de variable on obtient :

$$\phi(i,k) = \sum_{n=0}^{N-1-(i-k)} S(n) \cdot S(n+i-k) \qquad \begin{matrix} I \leqslant i \leqslant p \\ I \leqslant k \leqslant p \end{matrix}$$

Les bornes suppérieure et inférieure de la sommation sont limitées réspéctivement à $[N - I + (i - k)]$ et zéro (0) car $\phi(i,k)$ est nulle pour $n \notin [0 , N - I + (i - k)]$

La fonction d'autocorrélation R(k) est définie comme suit :

$$R(k) = \sum_{n=0}^{N-1-k} S(n) \cdot S(n + k)$$

$$R(k) = R(-k)$$

d'où :

$$\phi(i , k) = R(|i - k|)$$

l'équation (2) s'écrira donc :

$$\sum_{k=1}^{p} a(k) \cdot R(|i - k|) = R(i) \qquad I \leqslant i \leqslant p$$

son écriture sous forme matricielle est la suivante :

$$\begin{bmatrix} R(0) & \cdots & R(p-I) \\ \vdots & \ddots & \vdots \\ R(p-I) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a(I) \\ \vdots \\ a(p) \end{bmatrix} = \begin{bmatrix} R(I) \\ \vdots \\ R(p) \end{bmatrix}$$

La matrice carré p x p des valeurs d'autocorrélation est appelée matrice de Toeplitz , elle est symétrique et les éléments constituants sa diagonale sont tous égaux . [10]

II-2-4 -Méthode de covariance :

La seconde méthode consistant de déterminer les coéfficients a(k) de prédiction linéaire est la méthode de covariance . Cette méthode suppose que le signal est non stationnaire à l'intérieur de l'intérvalle à analyser . Elle se base éssentiellement sur les hypothèses suivantes :

- Le signal est défini pour " N + P " échantillons conséoutifs avec N et P entiers .

- L'érreur quadratique totale est minimisée pour l'ensemble des " N " échantillons consécutifs .

Soit : $$En = \sum_{n=0}^{N-1} e^{2}(n)$$

L'équation (2) s'écrira donc comme suit :

$$\sum_{n=0}^{N-1} S(n-i) \cdot S(n) = \sum_{k=1}^{p} a(k) \sum_{n=0}^{N-1} S(n-i) \cdot S(n-k) \qquad (4) \quad I \leq i \leq p$$

On pose :

$$\phi(i,k) = \sum_{n=0}^{N-1} S(n-i) \cdot S(n-k) = \phi(k,i) \qquad (5) \quad I \leq i \leq p$$

Par substitution de l'équation (5) dans l'équation (4) on obtient :

$$\phi(i,0) = \sum_{k=1}^{p} ak \quad \phi(i,k) \qquad (6) \qquad I \leq i \leq p$$

L'écriture de l'équation (6) sous forme matricielle est la suivante :

$$\begin{bmatrix} \phi(I,I) & \cdots & \phi(I,p) \\ \vdots & & \vdots \\ \phi(p,I) & \cdots & \phi(p,p) \end{bmatrix} \begin{bmatrix} a(I) \\ \vdots \\ a(p) \end{bmatrix} = \begin{bmatrix} \phi(I,0) \\ \vdots \\ \phi(p,0) \end{bmatrix}$$

On remarque que la matrice carré P x P composée des éléments $\phi(i,k)$ est symétrique car : $\phi(i,k) = \phi(k,i)$.

II-2-5 - Calcul de Gain :

Le gain G du signal d'éxcitation caractérise l'énérgie des échantillons prédits . Son calcul peut se faire à partir

de l'érreur $e(n)$ de prédiction linéaire .

Soit :

$$e(n) = S(n) - \hat{S}(n) \qquad n \in [0 , N-I]$$

$$e(n) = S(n) - \sum_{K=1}^{\rho} a(k) \cdot S(n-k) = G \cdot U(n) \qquad (7)$$

On remarque que le signal d'éxcitation $U(n)$ est proportionnel au signal de l'érreur $e(n)$ avec une constante de proportionnalité G .

En général , il est impossible de déduire directement à partir de cette relation la valeur du gain G . Pour cela on suppose que l'énérgie du signal de l'érreur est égale à l'énérgie du signal d'éxcitation .

Soit :

$$Eh = \sum_{n=0}^{N-1} e^{2}(n) = G^{2} \sum_{n=0}^{N-1} U^{2}(n)$$

Pour les sons non voisés , l'éxcitation $U(n)$ est assimilée à un bruit blanc , stationnaire , sa valeur est prèsque nulle .

Pour les sons voisés , l'éxcitation $U(n)$ est assimilée à une impulsion unitaire $S(n)$ tel que :

$$S(n) = \begin{cases} I & n = 0 \\ 0 & n \neq 0 \end{cases}$$

d'où :

$$En = \sum_{n=0}^{N-1} \left[h(n) - \hat{h}(n) \right]^{2} = G^{2} \sum S^{2}(n)$$

$h(n)$ étant la sortie du filtre récursif éxcité par l'impulsion $S(n)$ et $\hat{h}(n)$ sa valeur prédite .

$$En = \sum_{n=0}^{N-1} \left[h(n) - \sum_{K=1}^{\rho} a(k) \; h(n-k) \right]^{2}$$

$$En = \sum_{n=0}^{N-1} h^{2}(n) - 2 \sum_{K=1}^{\rho} a(k) \sum_{n=0}^{N-1} h(n) \; h(n-k) + \sum_{K=1}^{\rho} a(k) \sum_{i=1}^{\rho} a(i) \sum_{n=0}^{N-1} h(n-k) . h(n-i) \qquad (8)$$

Le passage aux dérivées partielles $\partial En / \partial a(i)$ donne :

$$\frac{\partial En}{\partial a(i)} = 2 \sum_{n=0}^{N-1} \left[h(n) - \sum_{k=1}^{p} a(k) \cdot h(n-k) \right] \cdot h(n-i) = 0 \qquad I \leq i \leq p$$

$$\sum_{n=0}^{N-1} h(n-i)\, h(n) = \sum_{k=1}^{p} a(k) \sum_{n=0}^{N-1} h(n-i)\, h(n-k) \qquad I \leq i \leq p$$

Par changement de variable de l'équation çi-dessus on obtient :

$$\sum_{n=0}^{N-1} h(n-k) \cdot h(n) = \sum_{i=0}^{p} a(i) \sum_{n=0}^{N-1} h(n-k) \cdot h(n-i) \qquad (9) \qquad I \leq k \leq p$$

D'aprés les équations (8) et (9) , l'érreur quadratique totale minimisée est :

$$En = \sum_{n=0}^{N-1} h^2(n) - \sum_{k=1}^{p} a(k) \sum_{n=0}^{N-1} h(n) \cdot h(n-k) \qquad (10)$$

La fonction d'autocorrélation de $h(n)$ est définie comme suit :

$$R(k) = \sum_{n=0}^{\infty} h(n) \cdot h(n+k) \; ; \; R(k) = R(-k) \qquad I \leq k \leq p$$

d'où :

$$En = R(0) - \sum_{k=1}^{p} a(k) \cdot R(k)$$

$$En = G^2 \sum_{n=0}^{N-1} S^2(n) = G^2 S^2(0) = G^2$$

Par conséquant :

$$En = R(0) - \sum_{k=1}^{p} a(k) \cdot R(k) = G^2$$

On remarque que le carré du gain G représente l'érreur quadratique totale de prédiction linéaire .

II-2-6 - Calcul des coéfficients de prédiction linéaire :

Soit à résoudre le système d'équation suivant :

$$\sum_{k=1}^{p} a(k) \; R(|i-k|) = R(i) \qquad I \leq i \leq p$$

obtenu par la méthode d'autocorrélation .

On calcul les coéfficients a(k) de prédiction linéaire pour ce système à l'aide de la méthode récursive de Durbin dont l'algorithme s'illustre comme suit :

Soit :

$$E^{(o)} = R(0)$$

$$ki = \left[R(i) - \sum_{j=1}^{i-1} aj \; R(i-j)\right] / E^{(i-1)} \qquad (I)$$

$$a_i^{(i)} = ki \qquad I \leq i \leq p \qquad I \leq j \leq i - I$$

$$a_j^{(i)} = aj - ki \; a_{i-j}^{(i-1)}$$

$$E^{(i)} = (I - ki^2) \; E^{(i-1)} \qquad (2)$$

La résolution des équations (I) et (2) se fait d'une manière récursive et la solution finale sera donnée ainsi :

$$aj = aj^{(p)} \qquad I \leq j \leq p$$

L'avantage de cette méthode c'est qu'elle utilise P(P + I) opérations et utilise donc une place mémoire réduite par rapport à une autre méthode .

II-3 -COMPARAISON DES DEUX METHODES D'ANALYSE : (LPC et Cepstrale)

- Méthode d'analyse par prédiction linéaire :

Elle souffre des mêmes limitations que les méthodes classiques d'analyse numérique , liées au caractère non stationnaire et pseudo-périodique du signal de la parole . Sa supériorité provient du fait qu'elle est fondée sur un modèle simple de production de la parole constituant une bonne approximation du système phonatoire , mais elle utilise en général douze (I2) coéfficients de prédiction linéaire pour représenter la fonction de transfert du conduit vocal et les caractéristiques du signal source .

- Méthode cepstrale :

Cette méthode peut utilisér plus de huit (8) coéfficients pour l'identification du locuteur . Néanmoins en reconnaissance de la parole (multilocuteurs) huit coéfficients suffisent pour obtenir un taux de reconnaissance satisfaisant , mais son algorithme est complèxe (calcul du logarithme) , et la méthode nécéssite le calcul des filtres " MEL " ainsi que le lissage , ce qui influe sur le temps de reconnaissance de la parole .

On remarque que les deux méthodes d'analyse de la parole , en vue de la reconnaissance , sont équivalentes et peuvent être mises en application sur le TMS 320 IO .
Cependant pourles systèmes de reconnaissance multilocuteurs utilisant un grand nombre de mots à reconnaître , on utilise de préférence la méthode cepstrale car elle utilise moins de cases mémoires , pour le stockage d'un mot (8 coéfficients) , mais le temps de reconnaissance sera long par rapport à celui obtenu en utilisant la méthode d'analyse par prédictionlinéaire (LPC) .

Pour les systèmes utilisant un vocabulaire réduit à reconnaitre et éxigeant un temps de reconnaissance rapide , la méthode LPC est recommandée .

C H A P I T R E III

MISE EN OEUVRE DE L'ANALYSE A L'AIDE DU T M S 320 IO

III-1 - Mise en oeuvre de l'analyse L P C .

III-2 - Mise en oeuvre de la F F T .

III-I - MISE EN OEUVRE DE LA METHODE L P C

Introduction :

La place mèmoire disponible sur le T M S 320 IO ne permet pas de réaliser une analyse sur plus de I28 échantillons sans utiliser de mémoires extérieures.

Les programmes que nous avons developpé comprennent toutes les étapes de l'analyse de la parole, la mémoire de donneés du T M S 32 0 IO aura été entièrement utilisée: I28 échantillons codés sur I6 bits sont chargés dans la page 0, ce qui correspond à IO ms de parole pour une fréquence d'échatillonnage de I2,8 KHZ. Quant à la page I, elle est résérvéeaux indices et aux résultats intérmédiaires nécessaires aux calculs.

Les étapes de cette analyse sont :

- Préaccentuation des échantillons
- Fenêtrage
- Calcul des coefficients d'autocorrelation
- Calcul des coefficients de prédiction linèaire

III-I-I - Préaccentuation des échantillons :

Cette opération se réalise aisement en numérique par la différentiation d'échantillons successifs :

$$S_1(n) = S(n) - S(n-1)$$

où N : est le nombre d'échantillons
n : le rang de l'échantillons
S (n) : l'échantillon de rang n
S_1 (n) : l'échantillon préaccentué.

Aprés éxécution du programme correspondant à cette étape, les échantillons préaccentués S_1 (n) écrasent les échantillons S (n) chargés précedemment dans la mémoire de donneés.

III-I-2-Fenêtrage :

La fenêtre de Hamming a été adoptée dans cette étape, pour ses caractéristiques :

- 99,96 % d'énergie concentrée dans le lobe principal.
- Niveau du I^{er} lobe secondaire à 43,9 dB au dessous du niveau du lobe principal.

L' expression de cette fenêtre est donneé par :

$$H(n) = 0{,}54 - 0{,}46 \cos\left((2\cdot\pi\cdot n)/(N - 1)\right)$$

où N est la taille des échantillons.

Pratiquement, cette étape est réaliseé en multipliant la valeur de chaque échantillon préaccentué par la valeur de la fenêtre de Hamming correspondant à son rang.

Soit donc : $S_2(n) = S_1(n) \times H(n)$

où H (n): est la fenêtre de Hamming

$S_1(n)$: l'échantillon préaccentué

$S_2(n)$: l'échantillon obtenu après fenêtrage.

Les échantillons $S_2(n)$ écrasent les échantillons $S_1(n)$ calculés préoedemment.

III-I-3 - Calcul des coefficients d'autocorrelation :

La fonction d'autocorrelation est donneé par :

$$R(I) = \sum_{J=0}^{N-1-I} S(J).S(J+I)$$

où P : est l'ordre de prédiction linèaire

I : le rang du coefficient d'autocorrelation

Ces coefficients sont stockés à partir de la position mémoire (2 P + 1) à 3 P .

III-I-4 - Calcul des coefficients de prédiction linèaire :

Nous avons utilisé pour ce calcul l'algorithme de Durbin illustré par l'organigramme de la méthode L P C .

Ces coefficients seront codés sur I6 bits et stockés dans la page 0 à partir de la position I à p .

III-I-5 - Temps d'éxécution :

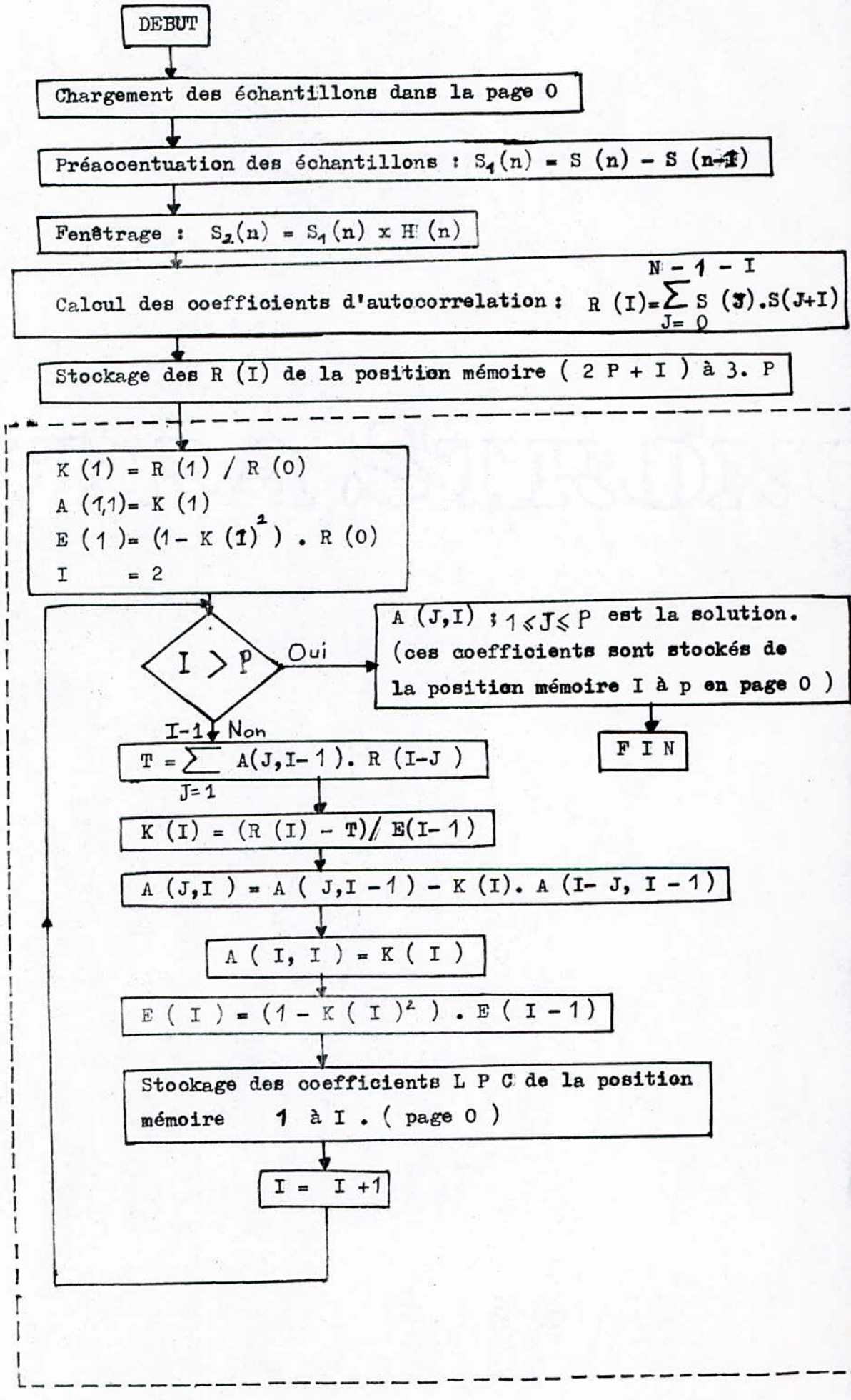
Le temps d'éxécution de chaque programme est donné par :

- Préaccentuation ---------- 202 µs
- Fenêtrage ---------- 308 µs
- Calcul des coefficients d'autocorrelation ----- 4418 µs
- Calcul des coefficients L P C ---------- 1109 µs

Le temps total est par conséquent : ---------- 6037 µs

Note : Les chargements des constantes et des donnéés sont compris dans le temps d'éxécution du programme. Ce programme nécessite une mémoire de programme de 323 mots de 16 bits.

- Organigramme de la méthode L P C



III-2 - MISE EN OEUVRE DE LA F F T

Introduction :

Le T M S 320 IO peut calculer la F F T (algorithme RADIX 2) sur 64 points complexes (sans R A M d'extension) .

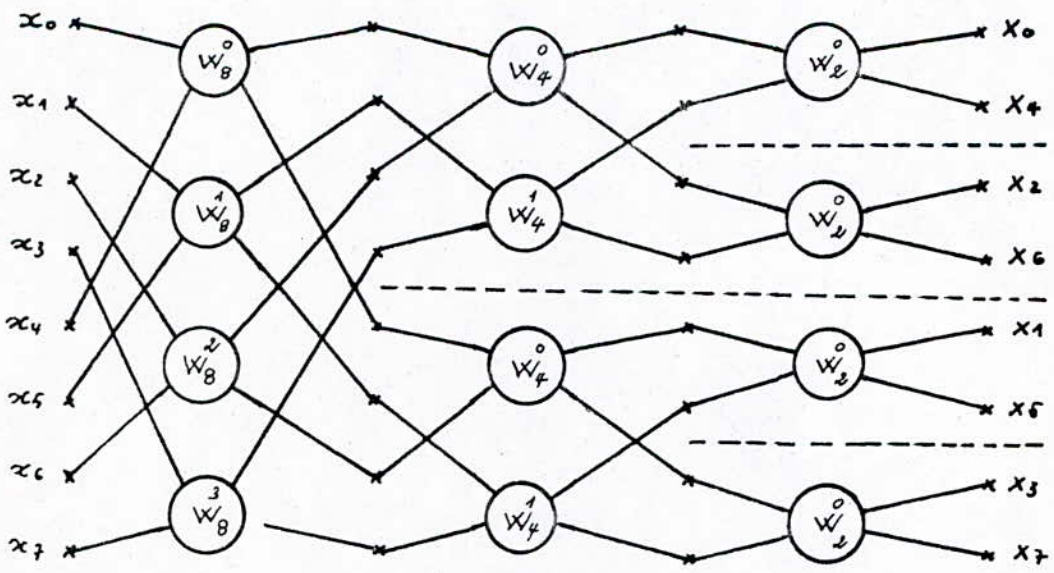
Les programmes contenus dans la disquette de simulation calculent la F F T sur 64 points complexes, avec entrelacement fréquenciels (graphe de fluence F F T) en utilisant une mémoire extèrne afin de stocker les résultats intérmédiaires correspondant à chaque niveau de calcul. Cette mémoire est adresseé par le port O et les donneés sont lues par l'intérmédiaire du port I .

Comme nous disposons d'un logiciel qui simule seulement le T M S 320 IO sans R A M extèrne, nous étions donc ramenés à calculer la F F T sur 64 points complexes en utilisant uniquement la R A M intèrne du T M S 320 IO , soit donc la capacité mémoire intèrne maximale de ce microprocesseur.

Calcul de la F F T avec entrelacement fréquentiel :

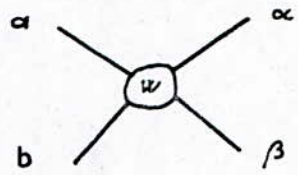
Dans cette méthode, les donneés sont chargeés dans l'ordre normal et les résultats seront obtenus dans l'ordre du " bit- revèrse". [6]

Pour " 08 " points, la graphe de fluence est le suivant :



- Graphe de Fluence F F T -

Le papillon



correspond au calcul $\begin{cases} \alpha = a + b \\ \beta = W(a-b) \end{cases} ; \quad W = \exp\left[-j\frac{2.\pi}{N}\right]$

Le programme que nous avons réalisé calcul la F F T sur 64 points complexes codés sur I6 BITS puis chargés dans la page 0 comme valeur réelle suivie d'une valeur immaginaire, les résultats seront récu.perés dans la même page et dans le même ordre. La page I est réserveé aux indices et aux calculs intérmédiaires.

Ce programme necessite une mémoire de programme de I7I mots de I6 bits et s'éxécute dans I.7 ms, avec temps de chargement et de lécture compris.

Ce programme F F T a été developpé dans le but de mettre au point un autre logiciel d'analyse de la parole sur T M S 320 I0 en l'occurence l'analyse cepstrale.

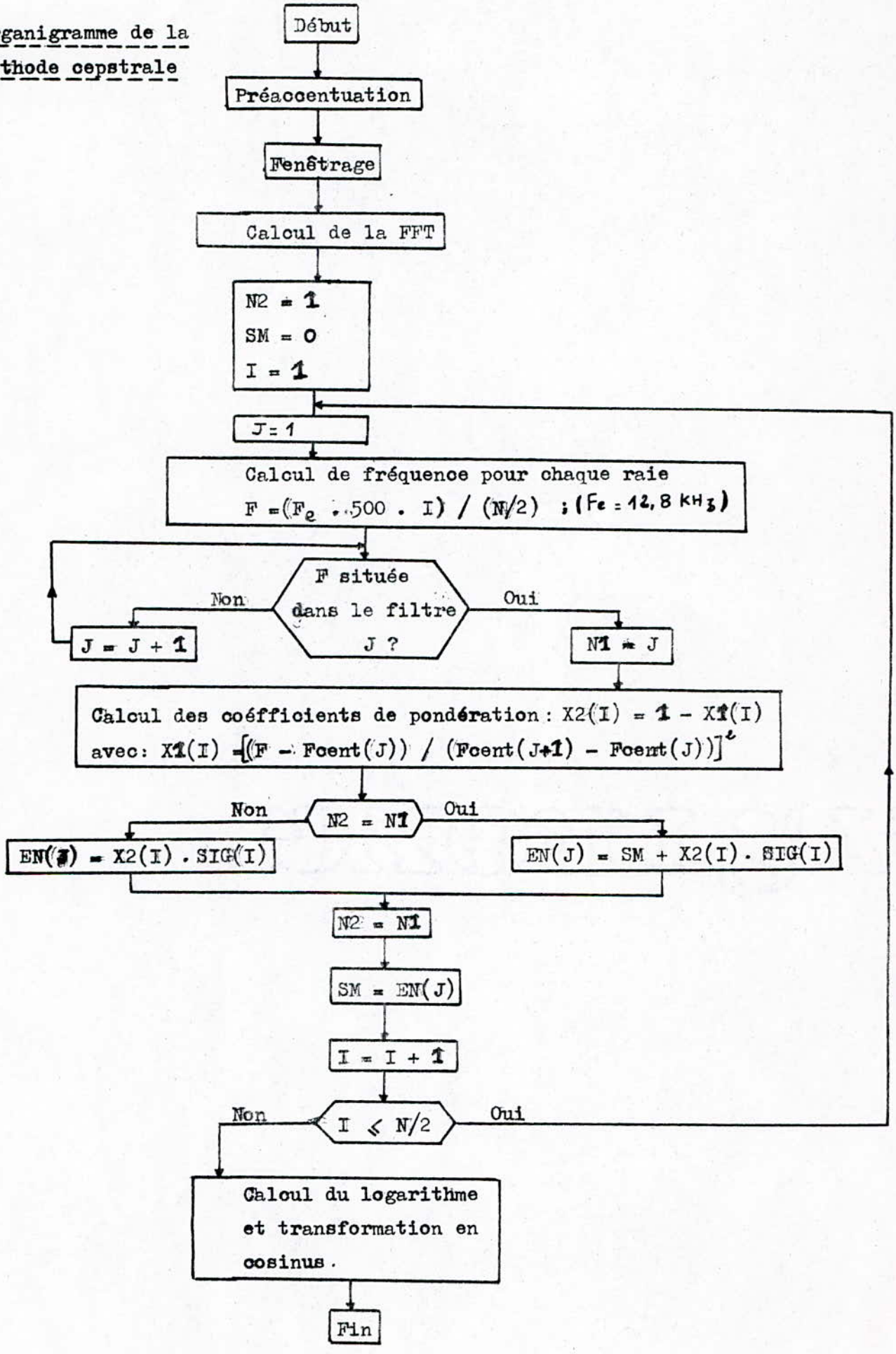
En effet , cette technique peut se résumer dans les étapes suivantes :

- Préaccentuation
- Fenêtrage
- F F T
- Calcul des coefficients cepstraux sur l'échelle M E L

pour ceci , nous proposons l'organigramme ci - dessous : tel que :

N : est la taille de la FFT.
I : détermine le rang de la raie.
J : détermine le numéro du filtre.
X2(I) : est le coéfficient de ponderation.
SIG(I) : est l'énergie de la raie.
F : est la fréquence de la raie.
Fe : est la fréquence d'échantillonnage.
EN(J) : énergie du filtre J.
SM , N1 , et N2 servent au calcul des énergies des filtres MEL .

Organigramme de la méthode cepstrale



C H A P I T R E I V

RECOMMANDATIONS EN VUE DE LA REALISATION D'UN SYSTEME DE RECONNAISSANCE DE LA PAROLE EN UTILISANT LE T M S 320 IO

IV-I - Reconnaissance de la parole

IV-2 - Mise en application sur le T M S 320 IO

IV-I - RECONNAISSANCE DE LA PAROLE

Introduction :

On peut classer les travaux dans ce domaine selon deux critères indépendants :

- Le premier critère est le caractère global ou analytique de l'approche utilisée . Une approche globale consiste à tenter de reconnaitre le mot ou la phrase prononcée comme une entité unique , sans chercher à la scinder en ses composants phonétiques .

L'approche analytique , au contraire , tire partie de la structure de la parole et des résultats de la phonétique pour isoler et identifier les constituants élémentaires (phonèmes , diphonèmes , etc ...) du méssage parlé .[2]

- Le second critère concèrne le type d'élocution accépté par le système : mots isolés , ou parole continue .

Les systèmes de reconnaissance dits " globaux " :

Dans un système global , l'unité minimale est le mot (ou groupe de mots) enregistré comme référence acoustique .

La reconnaissance globale de mots isolés consiste à identifier un mot prononcé par comparaison à tous les mots du vocabulaire . La réponse est alors la classe dont le mot reconnu se rapproche le plus .

L'établissement du " dictionnaire " contenant tous les mots du vocabulaire est éffectué au cours d'une phase d'apprentissage (enregistrement des mots du vocabulaire) . Pour les systèmes monolocuteurs , chaque utilisateur doit éffectuer une phase d'apprentissage , tandis que pour les systèmes multilocuteurs l'apprentissage est éffectué une fois pour toute suivant un algorithme de classification choisi .

La figure (I) donne le schéma général d'un système global de reconnaissance . Les systèmes globaux nécéssitent de stocker pour chaque mot du vocabulaire au moins une forme de référence , ce qui représente une grosse capaçité de mémoire , donc il faut utiliser un codage qui réduit le débit , tout en gardant les paramêtres pértinents pour la reconnaissance , l'éxtraction de ces paramêtres est assurée par l'analyseur acoustique .

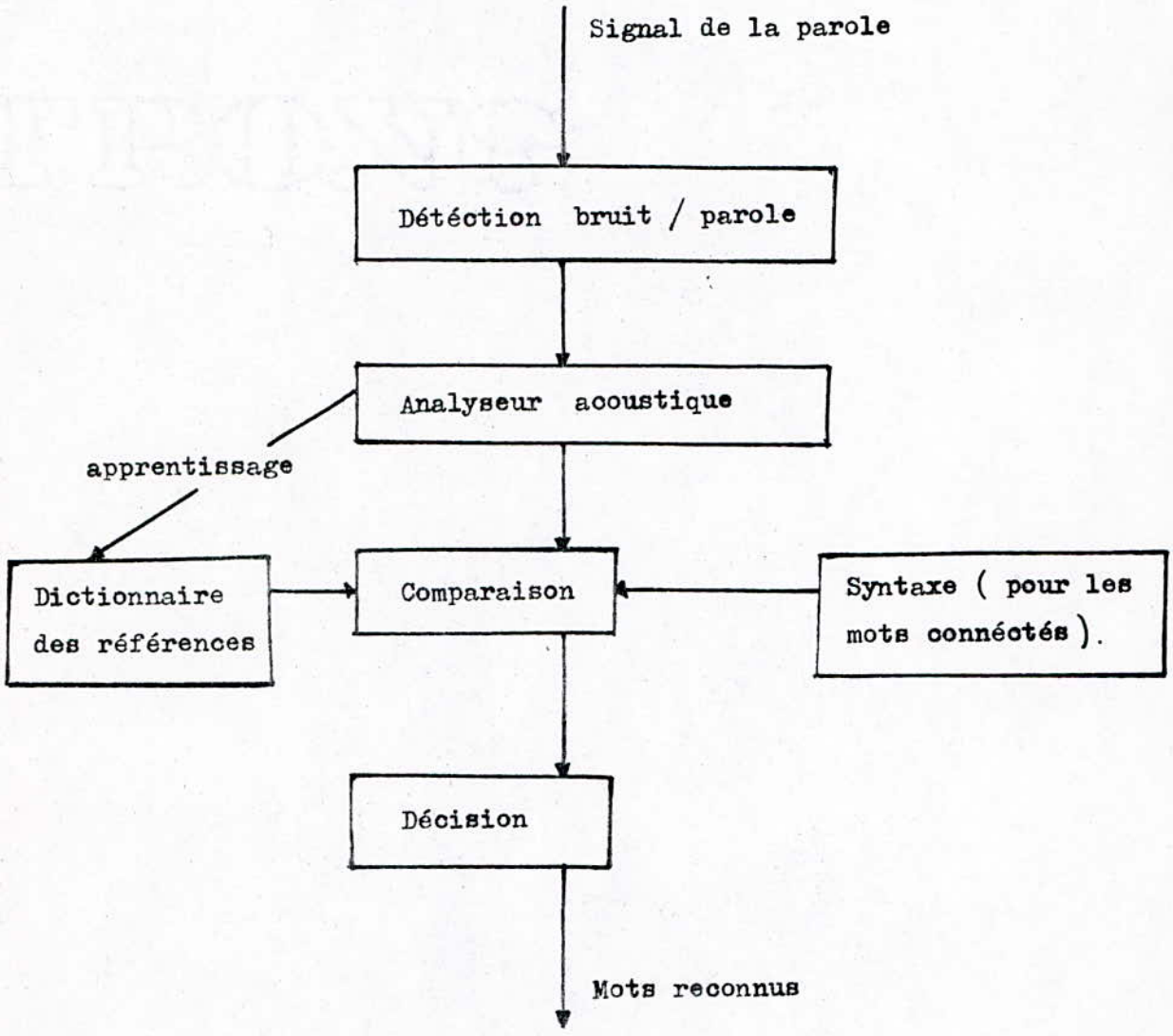


Figure (I) : Système globale de reconnaissance

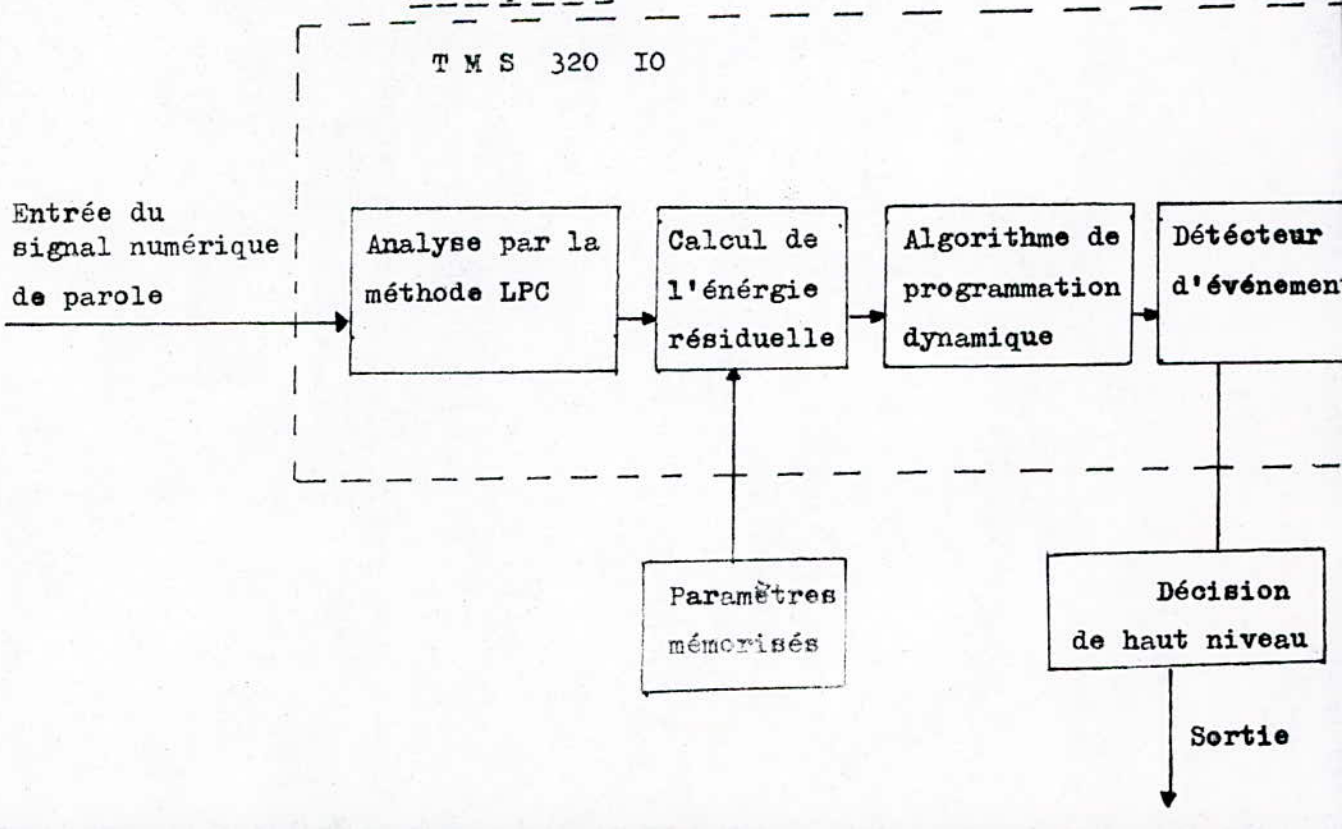
IV-2 - MISE EN APPLICATION SUR LE T M S 320 IO

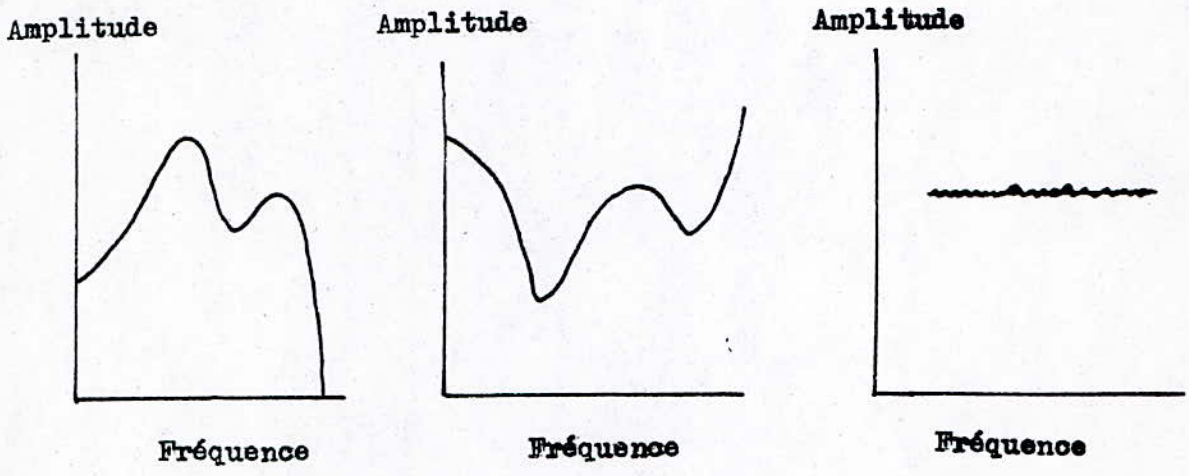
La méthode de la reconnaissance de la parole pouvant être mise en application par le T M S 320 IO est illustrée par la figure (2) . L'analyseur acoustique par la méthode LPC extrait les paramètres pértinents du mot lesquels passeront ensuite à travers un filtre invèrse formé par des paramètres de référence . Quand ces paramètres sont identiques à ceux stockés dans la mémoire , la sortie du filtre invèrse (énérgie résiduelle) est à peu prés nulle . Voir figure (3) .

L'algorithme de programmation dynamique pérmet une adaptation à chaque fois que le mot est éxprimé à une vitèsse différente que celle du mot stocké dans la mémoire , donc il réalise une normalisation temporelle .

Le détécteur d'événement séléctionne les adaptations possibles des mots stockés et les passera ensuite à travers la fonction de décision de haut niveau qui donnera le mot séléctionné parmis toutes ces possibilités .[12]

Figure (2) : Méthode approchée à la reconnaissance de la parole.





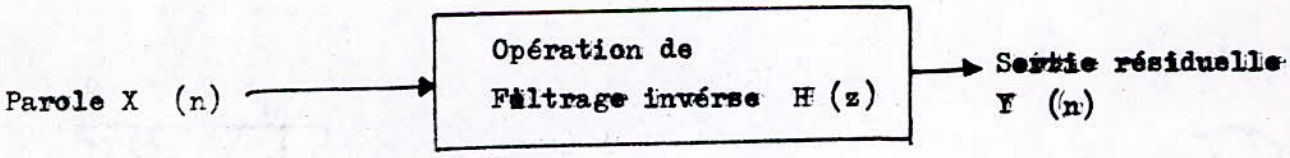


Figure (3) : Filtrage invèrse .

- PROGRAMMATION -

---------o---------

```
*       ##########################################################
*       #                                                        #
*       # CALCUL DES COEFFICIENTS DE PREDICTION LINEAIRE         #
*       #                                                        #
*       ##########################################################
*
*
*      Ce programme traite l'analyse par prédiction linéaire
* sur 128 échantillons codés sur 16 bits et chargés dans la
* page 0 . Les résultats sont récuperés dans la meme page
* de la position mémoire 1 à P .
*      La page 1 est réservées aux indices et aux résultats
* intermédiaires nécessaires aux calculs .
*
*
* Liste des constantes
************************
*
N   EQU 127          * Taille des échantillons
P   EQU 12           * Ordre de prédiction
NI  EQU 126          * NI=N-1
PI  EQU 11           * PI=P-1
X   EQU 25           * X=(2*P)+1
Y   EQU 36           * Y=3*P
Z   EQU 63           * Z=(N/2)-1
*****************************************************************
*
* Positions mémoire de la page 1
*********************************
*
I   EQU 0          * I,J,I1,NJ,JI,et K servent de compteurs
J   EQU 1          * de boucles et d'indexes
I1  EQU 2
NJ  EQU 3
JI  EQU 4
K   EQU 5
KI  EQU 6          * Valeur de K(I)
EI  EQU 7          * Valeur de E(I)
T   EQU 8          * Valeur de T
AJ  EQU 9          * Valeur de A(J,I)
NUMERA EQU 10      * Numérateur
DENOM EQU 11       * Dénominateur
QUOT EQU 12        * Résultat sur la division
FRAC EQU 13        * Définit la précision de la division
SIGNE EQU 14       * Signe de la division
UN EQU 15          * Contient la valeur 32767
*****************************************************************
*
* Préaccentuation des échantillons
***********************************
*
            AORG 0
START       LDPK 1
            LARP 0
            LARK AR0,NI
BCLP        LAC  *
            MAR  *-
            SUBS *+
```

```
            SACL *-                 * S1(n)=S(n)-S(n-1)
            SAR  AR0,I
            LAC  I
            BGZ  BCLP               * I variant de 1 à N
*
* Fenétrage
**********
*
            LACK FEN
            SACL I1
            LARK AR0,0
BCLF        SAR  AR0,I
            LAC  I1
            TBLR NJ                 * Valeur courante de H(n)
            LT   NJ
            MPY  *
            PAC                     * S2(n)=S1(n)*H(n)
            SACH *,0,1              * n variant de 0 à (N/2)-1
            LACK NI
            SUBS I
            SACL JI
            LAR  AR1,JI
            MPY  *
            PAC                     * S2(n)=S1(n)*H(n)
            SACH *,0,0              * n variant de N/2 à N-1
            LACK 1
            ADDS I1
            SACL I1
            LACK Z                  * Z=(N/2)-1
            SUBS I
            MAR  *+
            BGZ  BCLF               * I variant de 0 à (N/2)-1
*
* Calcul des coefficients d'autocorrelation
*******************************************
*
            LACK 1
            SACL I1
            LT   I1
            MPYK ADR
            PAC
            SACL I1
            LARK AR0,PI
BCLR        ZAC
            SACL NJ                 * Initialisation de R(I)
            SAR  AR0,I
            LACK NI
            SUBS I
            SACL J
            LAR  AR1,J
BCLS        LARP 1
            SAR  AR1,J
            LT   *,0
            LAC  J
            ADDS I
            SACL JI
            LAR  AR0,JI
            MPY  *,1
```

```
            PAC
            SACH K
            LAC  NJ
            ADDS K
            SACL NJ
            BANZ BCLS            * J variant de 0 à N-1-I
            LAC  I1
            TBLW NJ                      N-1-I
            LACK 1                       ----
            ADDS I1                      \
            SACL I1              * R(I)= /___ S(J)*S(J+I)
            LARP 0                       J=0
            LAR  AR0,I
            BANZ BCLR            * I variant de 1 à P
*
* Transfert des R(I) de la ROM vers la RAM
*******************************************
*
            LARK AR0,Y
            LACK 1
            SACL I1
            LT   I1
            MPYK ADR
            PAC
            SACL I1
BCLTR       SAR  AR0,JI
            LAC  I1
            TBLR *
            LACK 1
            ADDS I1
            SACL I1
            LACK X
            SUBS JI
            MAR  *-
            BLZ  BCLTR
*
*Calcul des coefficients  de prédiction lineaire
*************************************************
*
            LARK AR0,N
            LAC  *
            SACL DENOM           * Chargement du diviseur (R(0))
            MAR  *+
            LAC  *
            SACL NUMERA          * Chargement du dividande (R(1))
            CALL DIV             * Exécution de la division
            LAC  QUOT
            SACL KI              * K(1)=R(1)/R(0)
            LARK AR0,1           * A(1,1)=K(1)
            SACL *
            LT   KI
            MPY  KI
            PAC
            SACH EI              * E(1)=K(1)^2
            LACK ONE
            TBLR UN
            LAC  UN
            SUBS EI
```

```
                SACL EI              * E(1)=1-K(1)^2
                LARK AR0,N
                LT   *
                MPY  EI
                PAC
                SACH EI              * E(1)=(1-K(1)^2)*E(0)
                LARK AR0,1           * (E(0)=R(0))
BCLI            ZAC
                SACL T               * Initialisation de T
                MAR  *+
                SAR  AR0,I
                MAR  *-,1
                SAR  AR0,I1
                LARK AR1,0
BCLJ            MAR  *+
                SAR  AR1,J
                LACK N
                ADDS I
                SUBS J
                SACL NJ              * Addresse de R(I-J)
                LAR  AR1,NJ
                LT   *
                LAR  AR1,J
                MPY  *               * A(J,I-1)*R(I-J)
                PAC
                SACH JI                   I-1
                LAC  JI                   ---
                ADDS T                    \
                SACL T               * T= /__  A(J,I-1)*R(I-J)
                LAC  I1                   J=1
                SUBS J
                BGZ  BCLJ            * J variant de 1 à I-1
                LACK N
                ADDS I
                SACL JI              * Addresse des R(I)
                LAR  AR1,JI
                LAC  *,0,0
                SUBS T
                SACL NUMERA          * Chargement du dividande
                LAC  EI              * (R(I)-T)
                SACL DENOM           * Chargement du diviseur (E(I-1))
                CALL DIV             * Execution de la division
                LAC  QUOT
                SACL KI              * K(I)=(R(I)-T)/E(I-1)
                LARK AR0,0
BCLA            MAR  *+
                SAR  AR0,J
                LAC  I
                SUBS J
                SACL JI              * Addresse de A(I-J,I-1)
                LAR  AR0,JI
                LT   *
                MPY KI
                PAC
                SACH AJ              * A(J,I)=K(I)*A(I-J,I-1)
                LAR  AR0,J
                LAC  *
                SUBS AJ
```

```
                SACL  AJ              * A(J,I)=A(J,I-1)-K(I)*A(I-J,I-1)
                LACK  P
                ADDS  J
                SACL  JI
                LAR   AR0,JI
                LAC   AJ
                SACL  *
                LAR   AR0,J
                LAC   I1
                SUBS  J
                BGZ   BCLA            * J variant de 1 à I-1
                LACK  P
                ADDS  I
                SACL  JI              * Addresse de A(I,I)
                LAR   AR0,JI
                LAC   KI
                SACL  *               * A(I,I)=K(I)
                LT    KI
                MPY   KI
                PAC
                SACH  K               * K=K(I)^2
                LACK  ONE
                TBLR  UN
                LAC   UN
                SUBS  K
                SACL  K               * K=1-K(I)^2
                LT    K
                MPY   EI
                PAC
                SACH  EI              * E(I)=(1-K(I)^2)*E(I-1)
                LARK  AR1,P
                LARK  AR0,0
BCLST           MAR   *+,1
                SAR   AR0,J
                MAR   *+
                LAC   *,0,0
                SACL  *               * Stockage des coefficients LPC
                LAC   I
                SUBS  J
                BGZ   BCLST
                LAR   AR0,I
                LACK  P
                SUBS  I
                BGZ   BCLI            * I variant de 1 à P
FIN             B     FIN
ONE  EQU $
                DATA  32767
*
```

```
*Sous programme de la division
**********************************
*
DIV         LACK 15
            SACL FRAC
            LT   NUMERA
            MPY  DENOM
            PAC
            SACH SIGNE     * Sauvegarde du signe de la division
            LAC  DENOM
            ABS
            SACL DENOM     * Rendre le dénominateur positif
            LACK 15
            ADD  FRAC
            SACL FRAC      * sért pour la précision
            LAC  NUMERA
            ABS            * Rendre le numérateur positif
            LAR  AR0,FRAC
BCLD        SUBC DENOM         * boucle de division
            BANZ BCLD
            SACL QUOT
            LAC  SIGNE
            BGEZ FDIV      * Brancher vers FDIV si le signe
            ZAC            * est positif
            SUB  QUOT
            SACL QUOT      * Quotient négatif
FDIV        RET
*
* Valeurs de la fenétre de Hamming
**************************************
*
FEN EQU $
            DATA 2621    * Ces valeurs sont calculées à partir
            DATA 2640    * de l'expression suivante :
            DATA 2695    * H(n)=0.54 - 0.46*Cos((2*  *k)/127)
            DATA 2787    * avec 0 <= k <= 63 en raison de
            DATA 2916    * symétrie  . Elle sont ensuite codées
            DATA 3080    * sur 16 bits , tel que la valeur de
            DATA 3281    * 32767 correspond à H(n)=1 .
            DATA 3516
            DATA 3787
            DATA 4091
            DATA 4429
            DATA 4799
            DATA 5201
            DATA 5633
            DATA 6095
            DATA 6585
            DATA 7102
            DATA 7645
            DATA 8213
            DATA 8804
            DATA 9417
            DATA 10050
            DATA 10702
            DATA 11371
            DATA 12055
            DATA 12754
            DATA 13464
            DATA 14185
            DATA 14914
            DATA 15650
            DATA 16391
            DATA 17135
            DATA 17881
            DATA 18626
```

```
                DATA 19369
                DATA 20107
                DATA 20840
                DATA 21565
                DATA 22281
                DATA 22985
                DATA 23677
                DATA 24354
                DATA 25014
                DATA 25657
                DATA 26280
                DATA 26882
                DATA 27462
                DATA 28018
                DATA 28548
                DATA 29052
                DATA 29528
                DATA 29975
                DATA 30393
                DATA 30779
                DATA 31133
                DATA 31454
                DATA 31741
                DATA 31994
                DATA 32212
                DATA 32395
                DATA 32542
                DATA 32652
                DATA 32726
                DATA 32762
ADR EQU $
                DATA ADR
                END
```

```
*          ##################################################
*          #                                                #
*          #  CALCUL DE LA FFT 64 POINTS COMPLEXES          #
*          #                                                #
*          ##################################################
*
*          Le TMS 320 10 peut calculer une FFT (radix 2)
*   sur 64 points complexes en utilisant seulement la RAM
*   interne .
*          Les données sont dans la page 0 , organisées
*   comme valeur réelle suivit d'une valeur imaginaire ,
*   pour chaque point complexe .
*                              -nk
*          Les coéfficients (W   ) sont chargés dans la
*   mémoire de programme .
*
* List des constantes
**********************
*
* N est la taille de la transformeé .  N = 2**M.
N   EQU 64
M   EQU 6
M1  EQU 5                      * M1=M-1
N3  EQU 62                     * N3=N-2
*************************************************************
*
* positions memoires de la page 1
* ********************************
*
XT    EQU 0                    * Valeur intermédiaire X(I)
YT    EQU 1                    * Valeur intermédiaire Y(I)
I   EQU 2                      * Premier index
L   EQU 3                      * Deuxième index
COS EQU 4                      * Partie réelle du coefficient
SIN EQU 5                      * et sa partie imaginaire.
IA  EQU 6                      * Index de coefficients .
IE  EQU 7                      * Pour incrementer IA .
HOLDN  EQU 8                   * Contient la valeure N .
QUARTN    EQU 9                * Contient la valeure N/4 .
N1    EQU 10                   * Incrémente I.
N2    EQU 11                   * Séparation de I et L .
J     EQU 12                   * compteur de boucle .
ONE   EQU 13                   * Contient la valeur 1 .
TABLE    EQU 14    * position de la table de coéfficients.
A  EQU 15          * Sert pour le cucul du "bit reverse" .
*************************************************************
```

```
* debut de programme
*
        AORG 0
START     LDPK 1
          LACK 1
          SACL ONE          * Initialisation de IE = 1
          SACL IE
          LT   ONE
          MPYK   SINE
          PAC
          SACL TABLE        * Table contenant la position
          MPYK N            * du coefficient.
          PAC
          SACL HOLDN        * Holdn = N
          SACL N2           * Initialisation de N2 = N
          LAC  HOLDN,14
          SACH QUARTN       * Quartn = N/4
          LARK AR0,M1       * AR0 contient K compteurs
KLOOP     SAR  AR0,A
          LACK  RA0
          TBLW  A
          LARP 1
          LAC  N2,15
          SACH N1,1         * N1 = N2
          SACH N2           * N2 = N2/2
          ZAC
          SACL IA
          SACL J
          LAR  AR1,N2       * AR1 contient la valeur J
          MAR  *-           * commencer à N2-1
JLOOP     SAR  AR1,A
          LACK RA1
          TBLW  A
          LAC  TABLE
          ADD  IA
          TBLR SIN          * lire la paire de coéfficients
          ADD  QUARTN
          TBLR COS
          LAC  IA
          ADD  IE
          SACL IA           * IA =  IA + IE
          LAC  J,1
          SACL I      * I = J  (la donneé est organiseé
ILOOP     LAC  I      * comme une valeur réelle suivie
          ADD  N2,1   * d'une valeur imaginaire .
          SACL L            * L = I + N2
*
* Calcul du papillon
**********************
*
          LARP 0
          LAR  AR0,I
          LAR  AR1,L
          LAC  *,0,1
          SUB  *
          SACL XT           * XT = XI - XL
          ADD  *+,1,0
          SACL *+           * XI = XI + XL
          LAC  *,0,1
          SUB  *
          SACL YT           * YT = YI - YL
          ADD  *,1,0
          SACL *,0,1        * YI = YI + YL
```

```
            LT    COS
            MPY   YT
            PAC
            LT    SIN
            MPY   XT
            SPAC
            SACH  *-,1        * YL = COS*YT - SIN*XT
            MPY   XT
            PAC
            LT    COS
            MPY   XT
            APAC
            SACH  *,1,1       * XL = COS*XT + SIN*XT
*
* Incrémentation pour la boucle suivante .
*
            LAC   I
            ADD   N1,1         * I = I + N1
            SACL  I
            SUB   HOLDN,1      * jusqu'à I < N
            BLZ   ILOOP
            LAC   J
            ADD   ONE          * J = J + 1
            SACL  J
            LACK  RA1
            TBLR  A
            LAR   AR1,A
            BANZ  JLOOP
            LAC   IE,1
            SACL  IE           * IE = 2 * IE
            LARP  0
            LACK   RA0
            TBLR  A
            LAR   AR0,A
            BANZ  KLOOP
* compteur de "bit-reverse".
*
DEBUT       ZAC
            SACL  L
            SACL  I
            LACK  N3
            SACL  A
LOOP1       LAC   I        * si I >= L, brancher vers LOOP2
            SUB   L
            BGEZ  LOOP2
            LARP  0
            LAR   AR0,I
            LAR   AR1,L
            LAC   *+
            SACL  XT
            LAC   *-,0,1
            SACL  YT
            LAC   *-,0,0
            SACL  *-,0,1
            LAC   *-,0,0
            SACL  *,0,1
            LAC   XT
            SACL  *+
```

```
                LAC   YT
                SACL  *
LOOP2           LAC   HOLDN
                SACL  J             * J = N
LOOP3           LAC   L
                SUB   J             * si L >= J alors
                BLZ   LOOP4
                SACL  L             * L = L - J
                LAC   J,15
                SACH  J             * J = J/2.
                B     LOOP3
LOOP4           ADD   J,1
                SACL  L
                LAC   I
                ADD   ONE,1
                SACL  I             * Incrémenter I
                LAC   A
                SUB   ONE
                SACL  A
                BLZ   LOOP1
FIN             B     FIN
RA1 EQU $
                DATA  RA1
RA0 EQU $
                DATA  RA0
*
* table de coefficients (la taille de la table est 3n/4).
*****************************************************************
SINE  EQU $
COSINE EQU $
                DATA  32767     * Pour le codage des coefficients
                DATA  32609     * on ecrit la valeur maximale de
                DATA  32137     * sinus ou cosinus (1)sur 16 bits
                DATA  31356     * signés  (7FFF = 1 , FFFF = -1)
                DATA  30272     * et on déduit les autres valeurs.
                DATA  28897
                DATA  27244
                DATA  25329
                DATA  23169
                DATA  20787
                DATA  18204
                DATA  15446
                DATA  12539
                DATA  9511
                DATA  6392
                DATA  3211
                DATA  0
                DATA  -3211
                DATA  -6392
                DATA  -9511
                DATA  -12539
                DATA  -15446
                DATA  -18204
                DATA  -20787
                DATA  -23169
                DATA  -25329
                DATA  -27244
                DATA  -28897
                DATA  -30272
                DATA  -31356
                DATA  -32137
                DATA  -32609
                END
```

```
SINE EQU $
   DATA 0
   DATA 3211
   DATA 6392
   DATA 9511
   DATA 12539
   DATA 15446
   DATA 18204
   DATA 20787
   DATA 23169
   DATA 25329
   DATA 27244
   DATA 28857
   DATA 30272
   DATA 31356
   DATA 32137
   DATA 32609
```

Echantillons :

```
ENTER COMMAND (HELP=<CR>):
RAMI

ENTER  0  TO DISPLAY ADDRESSES >0  - >47
       1  TO DISPLAY ADDRESSES >48 - >89
0

>>ADDRESS
    0    306   2184   3039   2628   2457   3023   2799   4014
    8   4059   3023  -8225   1363   3547     67     51  13364
   10  13106  21573  26486 -30856 -26472  17219    803  12850
   18   4626  -8225 -17220   8621   -545    -35  21588  25942
   20 -30600 -27258   3021   2783    221    255    197      0
   28    187   1349  21573  21573   1622     86   3039   4013
   30    171   3018    203 -27288     52     67     86    120
   38    137      9    135    118    103   3530    101     86
   40     84   4013   2731   3005      4     69     84     67

ENTER COMMAND (HELP=<CR>):
RAMI

ENTER  0  TO DISPLAY ADDRESSES >0  - >47
       1  TO DISPLAY ADDRESSES >48 - >89
1

>>ADDRESS
   48     67     50     50      2     33     18     33     84
   50    188    188    188     11    203     33     67     33
   58     86      6     84     67   3021    163   1277     52
   60     50    529     33    541    539    538    540    173
   68     52   1347    101  17355    188   3023   3018   3018
   70   3279     13    253     10     33     50   1023      3
   78     52     67    203     12 -13448    203   2198 -26512
   80      0      0      0      0      0      0      0      0
   88      0      0      0      0      0      0      0      0
```

Résultats de la LPC (a(k)) :

```
ENTER COMMAND (HELP=<CR>):
RAMI

ENTER  0  TO DISPLAY ADDRESSES >0  - >47
       1  TO DISPLAY ADDRESSES >48 - >89
0

>>ADDRESS
    0     12 [-6691   8965   3368   1870 -15847   7123  -5843
    8    867  12714  29463   9890 -26065] -6691   8965   3368
   10   1870 -15847   7123  -5843    867  12714  29463   9890
   18 -26065  10197  -2296  -1577    941    879   -663   -572
   20   2303  -1021  -1784   -407    487     11    -20    -70
   28     67    431   7719      0  -8001   -631   1237    416
   30  -1674   1262  -1269 -12574  12679      7      9     16
   38      8    -63     61     -9     -8   1707  -1713     -4
   40     -1   1957   -637    135  -1476     31      7     -9
```

Résultats de la FFT (radix 2) :

```
ENTER COMMAND (HELP=<CR>):
RAMI

ENTER  0  TO DISPLAY ADDRESSES >0  - >47
       1  TO DISPLAY ADDRESSES >48 - >89


>>ADDRESS
    0 -28569   1911  28920 -28921  24291 -24292  24290 -24291
    8     59 -19298  20112 -20113 -22682  22681 -22682  22681
   10 -30888   -598 -25266  25265   -197    196   -197    196
   18  17373 -24061  19886 -19887  18629 -18630  18628 -18629
   20 -26248  30719  20411 -20412  30591 -30592  30590 -30591
   28  11739  20982 -13963  13962  31655 -31656  31654 -31655
   30   3345   -784   8876  -8877  -9771   9770  -9771   9770
   38 -29643 -24733  25980 -25981  30936 -30937  30935 -30936
   40   5225  15979  -1165   1164 -18416  18415 -18416  18415

ENTER COMMAND (HELP=<CR>):
RAMI

ENTER  0  TO DISPLAY ADDRESSES >0  - >47
       1  TO DISPLAY ADDRESSES >48 - >89
1

>>ADDRESS
   48 -22019  30617  -8639   8638 -15329  15328 -15329  15328
   50  28786  31079   5695  -5696  -9221   9220  -9221   9220
   58  27431  -1510 -19413  19412   4008  -4009   4007  -4008
   60   2341 -25350   9228  -9229  -8091   8090  -8091   8090
   68   7414 -13371  -4230   4229   1591  -1592   1590  -1591
   70 -19181 -13124 -30849  30848  25452 -25453  25451 -25452
   78   8574 -15730  -8138   8137    217   -218    216   -217
   80    217   -218      2     64      0  32767     32     64
   88     64     16      2      1     64      1    140     61
```

Ces résultats sont codés sur I6 bits, pour les décoder il suffit de les diviser par 32767 .

On constate que les coéfficients de prédiction linéaire sont compris entre -I et +I ce qui justifie la bonne éxécution du programme d'analyse .

C O N C L U S I O N

Le developpement de microprocesseurs spécialisés et l'affinage des algorithmes de traitement du signal ouvrent, en permanence, de nouvelles applications dans le domaine de la communication parlée.

L'étude du microprocesseur TMS 320 10 de TEXAS INSTRUMENTS, nous a permis de connaitre ses capacités et ses performances, afin de mettre en oeuvre une application d'analyse en vue de la reconnaissance de la parole.

Cette modeste contribution à la reconnaissance s'est matérialisée principalement par deux programmes : le premier developpe la téchnique d'analyse par prédiction linéaire et le second, la transformée de fourier rapide.

Nous pouvons affirmer que l'objectif visé, dans le cadre de notre projet de fin d'étude a été atteint. En effet, d'une part ce travail nous a permis de completer notre formation dans un sujet d'actualité et d'autre part les temps d'éxécution(6 ms pour l'analyse) et les performances du TMS 320 10 nous laissent optimiste à la réalisation de systèmes de reconnaissance de la parole en temps réel.

ANNEXE

Instruction Set Summary

The instruction set summary in the following table consists primarily of single-cycle single-word instructions. Only infrequently used branch and I/O instructions are multicycle.

TABLE II — INSTRUCTION SET SUMMARY

ACCUMULATOR INSTRUCTIONS																		
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER														
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS	Absolute value of accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0
ADD	Add to accumulator with shift	1	1	0	0	0	0	←	S	→		I	←			D			→
ADDH	Add to high-order accumulator bits	1	1	0	1	1	0	0	0	0	0	I	←			D			→
ADDS	Add to accumulator with no sign extension	1	1	0	1	1	0	0	0	0	1	I	←			D			→
AND	AND with accumulator	1	1	0	1	1	1	1	0	0	1	I	←			D			→
LAC	Load accumulator with shift	1	1	0	0	1	0	←	S	→		I	←			D			→
LACK	Load accumulator immediate	1	1	0	1	1	1	1	1	1	0	←				K			→
OR	OR with accumulator	1	1	0	1	1	1	1	0	1	0	I	←			D			→
SACH	Store high-order accumulator bits with shift	1	1	0	1	0	1	1	←	X	→	I	←			D			→
SACL	Store low-order accumulator bits	1	1	0	1	0	1	0	0	0	0	I	←			D			→
SUB	Subtract from accumulator with shift	1	1	0	0	0	1	←	S	→		I	←			D			→
SUBC	Conditional subtract (for divide)	1	1	0	1	1	0	0	1	0	0	I	←			D			→
SUBH	Subtract from high-order accumulator bits	1	1	0	1	1	0	0	0	1	0	I	←			D			→
SUBS	Subtract from accumulator with no sign extension	1	1	0	1	1	0	0	0	1	1	I	←			D			→
XOR	Exclusive OR with accumulator	1	1	0	1	1	1	1	0	0	0	I	←			D			→
ZAC	Zero accumulator	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1
ZALH	Zero accumulator and load high-order bits	1	1	0	1	1	0	0	1	0	1	I	←			D			→
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0	1	1	0	0	1	1	0	I	←			D			→

TABLE II – INSTRUCTION SET SUMMARY (CONTINUED)

AUXILIARY REGISTER AND DATA PAGE POINTER INSTRUCTIONS																		
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER														
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR	Load auxiliary register	1	1	0	0	1	1	1	0	0	R	I	←			D			→
LARK	Load auxiliary register immediate	1	1	0	1	1	1	0	0	0	R	←				K			→
LARP	Load auxiliary register pointer immediate	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	K
LDP	Load data memory page pointer	1	1	0	1	1	0	1	1	1	1	I	←			D			→
LDPK	Load data memory page pointer immediate	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	K
MAR	Modify auxiliary register and pointer	1	1	0	1	1	0	1	0	0	0	I	←			D			→
SAR	Store auxiliary register	1	1	0	0	1	1	0	0	0	R	I	←			D			→

BRANCH INSTRUCTIONS																			
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	Branch unconditionally	2	2	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BANZ	Branch on auxiliary register not zero	2	2	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BGEZ	Branch if accumulator ≥ 0	2	2	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BGZ	Branch if accumulator > 0	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BIOZ	Branch on $\overline{BIO}$ = 0	2	2	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BLEZ	Branch if accumulator ≤ 0	2	2	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BLZ	Branch if accumulator < 0	2	2	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BNZ	Branch if accumulator ≠ 0	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BV	Branch on overflow	2	2	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
BZ	Branch if accumulator = 0	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
CALA	Call subroutine from accumulator	2	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0
CALL	Call subroutine immediately	2	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
				0	0	0	0	←			BRANCH ADDRESS								→
RET	Return from subroutine	2	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	1

TABLE I – INSTRUCTION SET SUMMARY (CONCLUDED)

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS				
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER
				15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
APAC	Add P register to accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
LT	Load T register	1	1	0 1 1 0 1 0 1 0 I ← D →
LTA	LTA combines LT and APAC into one instruction	1	1	0 1 1 0 1 1 0 0 I ← D →
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0 1 1 0 1 0 1 1 I ← D →
MPY	Multiply with T register; store product in P register	1	1	0 1 1 0 1 1 0 1 I ← D →
MPYK	Multiply T register with immediate operand; store product in P register	1	1	1 0 0 ← K →
PAC	Load accumulator from P register	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0
SPAC	Subtract P register from accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0

CONTROL INSTRUCTIONS				
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER
				15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DINT	Disable interrupt	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1
EINT	Enable interrupt	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0
LST	Load status register	1	1	0 1 1 1 1 0 1 1 I ← D →
NOP	No operation	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
POP	Pop stack to accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1
PUSH	Push stack from accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0
ROVM	Reset overflow mode	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0
SOVM	Set overflow mode	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1
SST	Store status register	1	1	0 1 1 1 1 1 0 0 I ← D →

I/O AND DATA MEMORY OPERATIONS				
MNEMONIC	DESCRIPTION	NO. CYCLES	NO. WORDS	OPCODE INSTRUCTION REGISTER
				15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DMOV	Copy contents of data memory location into next location	1	1	0 1 1 0 1 0 0 1 I ← D →
IN	Input data from port	2	1	0 1 0 0 0 ← PA → I ← D →
OUT	Output data to port	2	1	0 1 0 0 1 ← PA → I ← D →
TBLR	Table read from program memory to data RAM	3	1	0 1 1 0 0 1 1 1 I ← D →
TBLW	Table write from data RAM to program memory	3	1	0 1 1 1 1 1 0 1 I ← D →

TABLE DES COMMANDES DU SIMULATEUR DU TMS 320 10
****

MAIN MENU COMMANDS

Command	Function
BH	Breakpoint Help
C	Continue Simulation
DM or <CR>	Display Main Menu
DT	Display Trace Buffer
EX	Execut Commands From Given File
IOH	Input Output Help Menu
JF	Select Journal File
L	Load New Object File
MH	Modify/Inspect Memory Help Menu
NS	Set Number Of Instructions Until Break
Q	Quit Simulation
R	Run Simulation
RH	Modify/Inspect Registers/Flags Help Menu
RS	Reset Simulator
SS	Perform Single-Step Execution
ST	Display Register Status
STR	Save Trace Buffer
TR	Toggle Trace
Z	Zero Clock Counter

SUITE DE LA TABLE
****

MODIFY/INSPECT REGISTERS/FLAGS COMMANDS

Command	Function
ACC	Modify/Inspect Accumulator
AR	Modify/Inpect Auxiliary Registers
BIO	Modify/Inspect I/O Branch Control
CC	Modify/Inspect Clock Counter
INTF	Modify/Inspect Interrupt Flag Register
INTM	Modify/Inspect Interrupt Mode Register
P	Modify/Inpect P Register
PC	Modify/Inspect Program Counter
SK	Modify/Inspect Stack
T	Modify/Inspect T Register

MODIFY/INSPECT STATUS REGISTERS/PINS COMMANDS

Command	Function
ARP	Modify/Inspect Auxiliary Register Pointer
DP	Modify/Inspect Data Memory Page Pointer
INTM	Modify/Inspect Interrupt Mode Register
OV	Modify/Inspect Overflow Flag
OVM	Modify/Inspect Overflow Mode Register

SUITE DE LA TABLE
****

BREAKPOINT COMMANDS

Command	Function
BDP	Breakpoint on Data Pattern When Read/Write from/to Data RAM
BDR	Breakpoint on Data RAM Read
BDRW	Breakpoint on Data RAM Read and Write
BDW	Breakpoint on Data RAM Write
BER	Breakpoint on an Error Condition
BIAQ	Brealpoint on Instruction Acquisition
BPP	Breakpoint on Data Pattern When Read from Program ROM
BPR	Breakpoint on Program ROM Read
DB	Display Braekpoints
RB	Remove Breakpoint

MODIFY/INSPECT MEMORY COMMANDS

Command	Function
RAM	Modify/Inspect Individual Data RAM
RAMH	Display Data RAM in Hex
RAMI	Display Data RAM in Integer
ROM	Modify/Inspect Individual Program ROM
ROMH	Display Program ROM in Hex
ROMI	Display Program ROM in Integer

SUITE DE LA TABLE
****

INPUT/OUTPUT T COMMANDS	
Command	Function
LF	List Files Assigned to Ports
RSI	Reset Selected Input Port File
SI	Select Input Port File
SO	Select Output Port File

INTERRUPT/TIMING COMMANDS	
Command	Function
TIC	Specify Number of Clock Tics till Interrupt
ZTIC	Disable the Commands

Unité de traitement numérique du signal :	S 2811 A.M.I.	2920 Intel	7720 Nec	DSP Bell	TMS 320 M 10 Texas Instr.	MB 8764 Fujitsu
Technologie	V-Mos	N-Mos	N-Mos	N-Mos	N-Mos	C-Mos
Mémoire programme	250 mots de 17 bits (ROM)	192 mots de 24 bits (E-PROM)	512 mots de 23 bits (ROM)	1 024 mots de 16 bits (ROM)	1 536 mots de 16 bits (ROM)	1 024 mots de 24 bits (ROM)
Mémoire des données (RAM)	128 mots de 16 bits	40 mots de 25 bits	128 mots de 16 bits	128 mots de 20 bits	144 mots de 16 bits	deux fois 128 mots de 16 bits
Mémoire des coefficients (ROM)	128 mots de 16 bits	—	512 mots de 13 bits	—	—	—
Durée d'une instruction	300 ns (multiplication addition et stockage)	400 ns (décalage, addition et stockage)	250 ns (multiplication seule)	800 ns (multiplication et accumulation)	400 ns (multiplication et accumulation)	100 ns (74 ns multiplication seule)
Nombre de sections de filtre du 2ème ordre pour un échantillonnage à 8 kHz	50	19	55	39	62	Note 1

Principaux microprocesseurs spécialisés dans le traitement du signal .

Type de fenêtre	Forme de la fenêtre temporelle et équation correspondante	Allure du spectre en fréquence	Amplitude du 1er lobe secondaire par rapport au lobe central (dB)	Largeur du lobe central (à - 3dB)	Largeur équivalente au bruit	Pente d'atténuation des lobes secondaires (dB/oct)
Rectangle (fenêtre naturelle)	$D = 1/\Delta f$; 1		- 13,2	0,88 Δf	Δf	6
Demi sinusoïde	$\sin \frac{\pi t}{D}$		- 22,4	1,15 Δf	1,26 Δf	12
Triangle (Bartlett)	$\frac{2t}{D}$, de 0 à $\frac{D}{2}$; $2 - \frac{2t}{D}$, de $\frac{D}{2}$ à D		26,6	1,28 Δf	1,33 Δf	12
Cosinus carré (Hann)	$\frac{1}{2}\left(1 + \cos \frac{\pi t}{D}\right)$; $= \cos^2 \left(\frac{\pi t}{2D}\right)$		31,6	1,39 Δf	1,5 Δf	18
Cosinus carré et décalage 8 % (Hamming)	$0,08 + 0,46\left(1 + \cos \frac{\pi t}{D}\right)$; $= 0,54 + 0,46 \cos \frac{\pi t}{D}$		- 43,9	1,26 Δf	1,36 Δf	6 (à partir de 5 Δf)

Fig. 6-26 -

Dans le choix d'une fenêtre de correction, il faut tenir compte de la nature du signal analysé (périodique ou bruit, par exemple) et viser le meilleur compromis. La durée de fenêtre est de $D = NT$.

BIBLIOGRAPHIE

I - M - AUMIAUX
" L'emploi des microprocesseurs " — Edition MASSON I982

2 - M - COUVRAT
" Reconnaissance de la parole continue par des méthodes globales " — CNET I982

3 - C - DARDANNE
" Le microprocesseur 6809 " — Edition EYROLLES I982

4 - M - FERRITTI , F - CINARE
" Synthèse , reconnaissance de la parole " — Edition TESTS I983

5 - D - JOUVET
" Etude de la faisabilité de l'analyse cepstrale sur le processeur de traitement de signal du CNET" — CNET I982

6 - M - KUNT
" Traitement numérique des signaux " — MASSON I980

7 - J - LIFERMANN
" Les méthodes rapides de transformation du signal " — MASSON I980

8 - A - MENACER
" Reconnaissance de la parole en mode multilocuteur par des méthodes globales (mots isolés) " — Thèse Docteur - Ingénieur I985

9 - R - MIQUEL
" Le filtrage numérique par microprocesseur " — Edition TESTS I985

I0 - L.R - RABINER R.W - SCHAFER
" Digital processing of speech signals " by bell laboratories incorporated I978

II - TEXAS INSTRUMENTS
" T M S 320 I0 Assembly langage programmer's guide " Revue TI I983

I2 - TEXAS INSTRUMENTS
" Digital signal processor " Revue TI I983

I3 - TEXAS INSTRUMENTS
" Simulator " Revue TI I983