(2)/ ETUDE SUR LA REALISATION D'UNE
AMELIORATION DE LA QUALITE DE
LA PAROLE TRANSMISE PAR VOCODEUR (LPC)
A PREDICTION LINEAIRE /

Rapport final

Ind
Lib
JUI
JUI

CENTRE DE RECHERCHES SUR LES COMMUNICATIONS

FACULTÉ DES SCIENCES APPLIQUÉES

UNIVERSITÉ DE SHERBROOKE

TÉL.: 819—565-4450

TÉLEX 05-836149

SHERBROOKE, QUÉBEC, CANADA, J1K 2R1

P
91
C655
M15
1985

P
91
C655
M15
1985

CENTRE DE RECHERCHES SUR LES COMMUNICATIONS
Faculté des sciences appliquées
Université de Sherbrooke

ETUDE SUR LA REALISATION D'UNE AMELIORATION DE LA QUALITE DE LA PAROLE TRANSMISE PAR VOCODEUR (LPC) A PREDICTION LINEAIRE

Rapport final

Industry Canada
Library Queen
JUIL JUL 21 1998
Industrie Canada
Bibliothèque Queen

pour le

Ministère des Communications
Ottawa, Ontario
Contrat no. OSV84-00108

COMMUNICATIONS CANADA
JAN 30 1986
LIBRARY – BIBLIOTHÈQUE

Philippe Mabilleau
François Didelot
Jean-Pierre Adoul

Sarto Morissette
Directeur CRCS

25 mars 1985

P
91
C655
M15
1985

DD 6063041
DL 6063216

RESUMÉ

Ce rapport présente les travaux effectués sur l'amélioration et la réalisation d'un vocodeur à 4800 bits/seconde utilisant la quantification vectorielle et la décimation généralisée.

Le passage à une fréquence d'échantillonnage de 10 KHz est envisagé. La qualité de la parole obtenue est évaluée à l'aide d'une série de tests subjectifs.

La réalisation d'un vocodeur en temps réel est étudiée. Du logiciel réalisant la quantification vectorielle sphérique dans E8 et la décimation généralisée est présenté pour le TMS 320 et le 68008.

TABLE DES MATIERES

TABLE DES MATIERES iii

1. INTRODUCTION - PRESENTATION DE L'ETUDE 1

2. ALGORITHMES PROPOSES 4

2.1 Choix d'une méthode de sous-échantillonnage 4

2.1.1 Bref rappel sur la décimation généralisée et le codage multi-impulsionnel 5
2.1.2 Etude du rapport signal à bruit 6
2.1.3 Tests subjectifs 8
2.1.4 Complexité 9
2.1.5 Conclusion sur le sous-échantillonnage 11

2.2 Quantification vectorielle sphérique dans E8 [3,4] 11

2.2.1 Codage 12
2.2.2 Décodage 13
2.2.3 Méthode numérique 14
2.2.4 Logiciels de QVS en temps réel 16

3. SYSTEME DE DEVELOPPEMENT DE VOCODEURS 18

3.1 Procédure de développement de vocodeurs 19
3.2 Le processeur vectoriel SKY 320 22

3.2.1 Matériel 22
3.2.2 Logiciel 24

4. REALISATIONS MATERIELLES EN VUE DE L'IMPLANTATION EN TEMPS REEL. 31

4.0 Implantation temps réel 31
4.1 Carte dictionnaire 31
4.2 Utilisation d'un circuit de synthèse 34

5. RESULTATS 37

5.1 Description des codeurs 37
5.2 Tests subjectifs 41
5.3 Résultats 43
5.4 Interprétation des résultats 46

6. CONCLUSIONS 48

6.1 Amélioration de la qualité du codage par augmentation de la bande passante 48
6.2 Réalisation d'un vocodeur en temps réel 49

Table des matières

BIBLIOGRAPHIE 51

ANNEXE A : Spécifications du circuit de synthèse utilisé 52

ANNEXE B : Problèmes de repliement introduits par la décimation généralisée 63

ANNEXE C : Décimation généralisée sur le TMS 320 66

ANNEXE D : Quantification vectorielle sphérique sur le TMS 320. 77

ANNEXE E : Quantification vectorielle sphérique sur le 68000 .. 88

ANNEXE F : Exemple de logiciel utilisant le circuit de synthèse 96

CHAPITRE 1

INTRODUCTION - PRESENTATION DE L'ETUDE

Ce rapport consigne les résultats des études sur la simulation et la réalisation d'algorithmes de codage de la parole à bas débit numérique (4800 bits/seconde), conduites au Centre de Recherches sur les Communications de l'Université de Sherbrooke (CRCS), pour le compte du Ministère des Communications du Canada, dans le cadre du contrat portant le numéro OSV84.00108.

Les objectifs spécifiques de ce contrat visent l'amélioration de la qualité de la parole codée par le biais d'une augmentation de la cadence d'échantillonnage du signal qui passerait de 8 à 10 Kéch/seconde. L'augmentation du contenu fréquentiel du signal ainsi obtenu devrait permettre une meilleure intelligibilité, notamment en ce qui concerne les sons non-voisés et les voix féminines.

Par contre, au niveau de l'algorithme de codage, cette augmentation de la bande passante, a un impact direct sur la quantité d'information à transmettre, et ceci, à deux niveaux:

1. L'ordre du filtre utilisé pour la prédiction linéaire doit être augmenté pour permettre la représentation d'un nombre supérieur de formants présents dans la bande des fréquences considérées.

2. L'augmentation du nombre d'échantillons à coder pour le signal résiduel nous oblige à modifier l'algorithme utilisé pour coder ce résiduel.

Pour ce qui est du filtre prédicteur, l'utilisation de la quantification vectorielle permet d'augmenter l'ordre des filtres sans modification du débit numérique nécessaire à leur transmission. Par contre, les performances d'un dictionnaire de 1024 filtres à l'ordre 10, ne seront certainement pas les mêmes que les performances d'un dictionnaire, toujours de 1024 filtres, mais à l'ordre 12. C'est un des points qui sera étudié dans ce rapport.

En ce qui concerne le codage du signal résiduel, la méthode utilisée est la décimation généralisée associée avec la quantification vectorielle sphérique utilisant la lattice E8. La quantité plus importante d'échantillons à coder va imposer une augmentation du facteur de décimation (passage de 5 à 6). L'impact de cette contrainte sur la qualité du codage sera aussi étudiée dans ce rapport.

La réalisation d'un vocodeur utilisant un ou deux microprocesseurs TMS 320 est le second volet de cette étude. On se limitera aux points suivants:

1. impact sur la faisabilité du passage à une fréquence d'échantillonnage de 10 KHz, et,

2. codage du signal résiduel par décimation généralisée et quantification vectorielle sphérique.

Les éléments suivants, qui ont fait l'objet d'études et de réalisations spécifiques, sont aussi présentés.

1. réalisation matérielle d'une carte "dictionnaire" permettant l'implantation des algorithmes de quantification vectorielle par dictionnaire sur le TMS 320,

2. l'utilisation d'un circuit spécialisé de synthèse de la parole pour le décodage.

3. l'utilisation d'un microprocesseur 68008 pour la quantification vectorielle sphérique.

Enfin un chapitre est consacré au système de développement et de simulation de vocodeurs que nous avons réalisé autour d'un processeur vectoriel SKY 320. Nous pensons que ce système va permettre une mise au point plus rapide des logiciels de traitement sur TMS 320.

CHAPITRE 2

ALGORITHMES PROPOSES

2.1 Choix d'une méthode de sous-échantillonnage

Les études précédentes ont mis en relief la nécessité d'effectuer un sous-échantillonnage du résidu associé à une quantification vectorielle afin d'atteindre des débits de transmission de la parole de l'ordre de 4800 bits par seconde, voire 2400.

Plusieurs méthodes sont envisageables. Dans le cadre de cette étude, on en a retenues, à priori, quatre:

- codage multi-impulsionnel ou méthode d'Atal,
- décimation généralisée avec centrage,
- décimation généralisée sans centrage,
- décimation "classique" ou bande de base.

Laquelle de ces méthodes se prête le mieux à une réalisation en temps réel? Afin de répondre à cette question, on a retenu trois critères d'évaluation:

- le rapport signal à bruit (RSB),
- le résultat des tests subjectifs,

- la complexité de l'algorithme.

Ces critères ont été utilisés pour évaluer des codeurs travaillant sur des phrases échantillonnées à 8 Kéch/s, par commodité. Toutefois, les résultats obtenus peuvent sans crainte être étendus à des codeurs travaillant à une fréquence d'échantillonnage de 10 KHz.

2.1.1 Bref rappel sur la décimation généralisée et le codage multi-impulsionnel

Ces méthode de sous-échantillonnage spéctralement adaptées, ont été présentées en détail dans le rapport soumis l'an dernier [1,2,5].

La décimation généralisée et le codage multi-impulsionnel sont deux méthodes très liées. Le principe de l'algorithme de codage multi-impulsionnel consiste à rechercher, pour un bloc de signal donné, les impulsions modélisant le signal résiduel les unes après les autres. On est donc en présence d'un algorithme itératif en deux étapes:

1. calculer le résidu,
2. chercher une nouvelle position,
3. calculer l'ensemble des amplitudes associées aux positions connues,
4. aller en 2 si nécessaire.

A l'étape 2 et 3, les résultats sont obtenus en minimisant l'erreur perceptuelle entre la synthèse et l'original: on effectue une pondé-

ration afin de masquer l'erreur par le signal. Les étapes 2 et 3 sont indépendantes, c'est-à-dire, que l'on peut calculer un ensemble d'amplitudes minimisant l'erreur pondérée associées à un ensemble quelconque de positions. La décimation généralisée se fonde sur cette remarque. Dans la plupart des processus d'optimisation itératifs, la première boucle contribue le plus à la minimisation du critère, c'est pourquoi, sur un bloc, il faut retenir la première position donnée par l'algorithme de codage multi-impulsionnel. Par la suite, la décimation généralisée extrait un ensemble de positions régulièrement espacées contenant cette impulsion, puis calcule l'ensemble des amplitudes correspondantes.

Le bloc de résidu est donc modélisé et sous-échantillonné en une seule itération. De plus, l'information relative à la position des impulsions est réduite à la position relative de la première impulsion dans le bloc.

2.1.2 Etude du rapport signal à bruit

Le rapport signal à bruit est une méthode simple pour mesurer la performance d'un codeur, en termes de quantité d'erreur produite. Cette méthode convient tout à fait au codage multi-impulsionnel et à la décimation généralisée qui sont des méthodes d'analyse par synthèse, c'est-à-dire minimisant l'erreur entre le son original et le son synthétisé. Dans le cas du codage en bande de base du résidu, le filtrage passe-bas introduit des déphasages entre l'original et la synthèse. Ceci peut expliquer la faiblesse du RSB. Toutefois le codage en bande de base doit être rapproché de la décimation généralisée non centrée, qui est en fait équivalente à un

sous-échantillonnage "classique" utilisant un filtre anti repliement adaptatif. Dans ce test, comme dans tous les autres, on a procédé à aucune quantification.

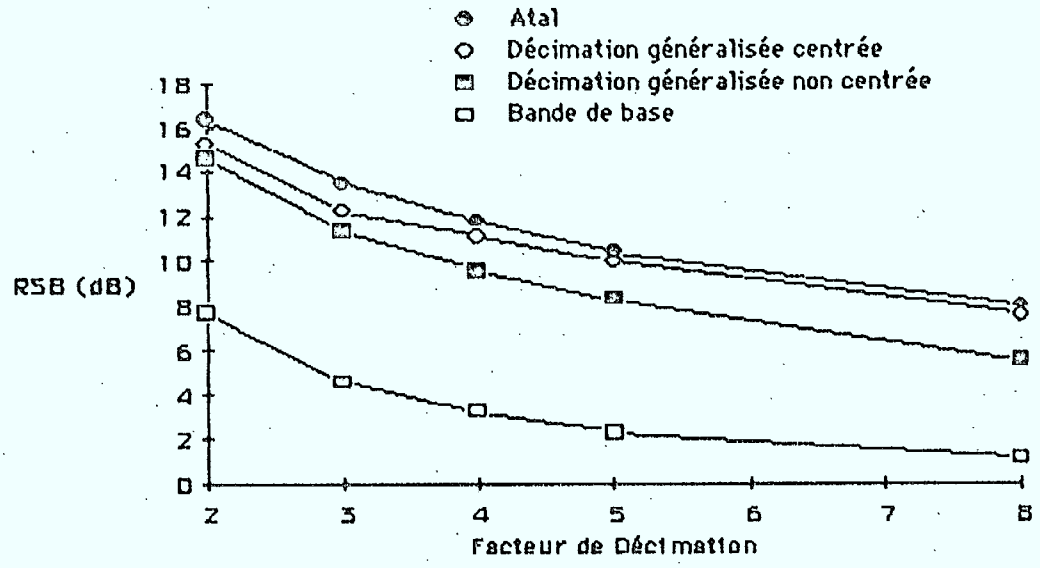


Fig. 2.1 Variations du rapport signal sur bruit en fonction du du facteur de décimation pour les quatre méthodes étudiées.

On peut distinguer deux groupes de codeurs: ceux inspirés du concept de pondération de l'erreur comme la méthode d'Atal ou la décimation généralisée, et le sous-échantillonnage classique. Parmi les codeurs du premier groupe, le comportement de la décimation généralisée avec centrage de la principale impulsion appelle un commentaire. En effet, pour un facteur de décimation important, ce codeur donne des performances comparables à celles du multi-impulsionnel intégral, alors que pour des facteurs de décimation faibles, la contribution de ce centrage diminue.

2.1.3 Tests subjectifs

Ces tests ont théoriquement une valeur supérieure au simple RSB, puisqu'ils permettent de classer les codeurs suivant la préférence d'un auditeur. Toutefois, dans le cas des facteurs de décimation faibles, la qualité de la parole reconstituée est telle que la différence entre les codeurs devient difficilement appréciable. Les résultats de ces tests ne sont donc pas toujours significatifs. Il convient donc de les interpréter avec prudence.

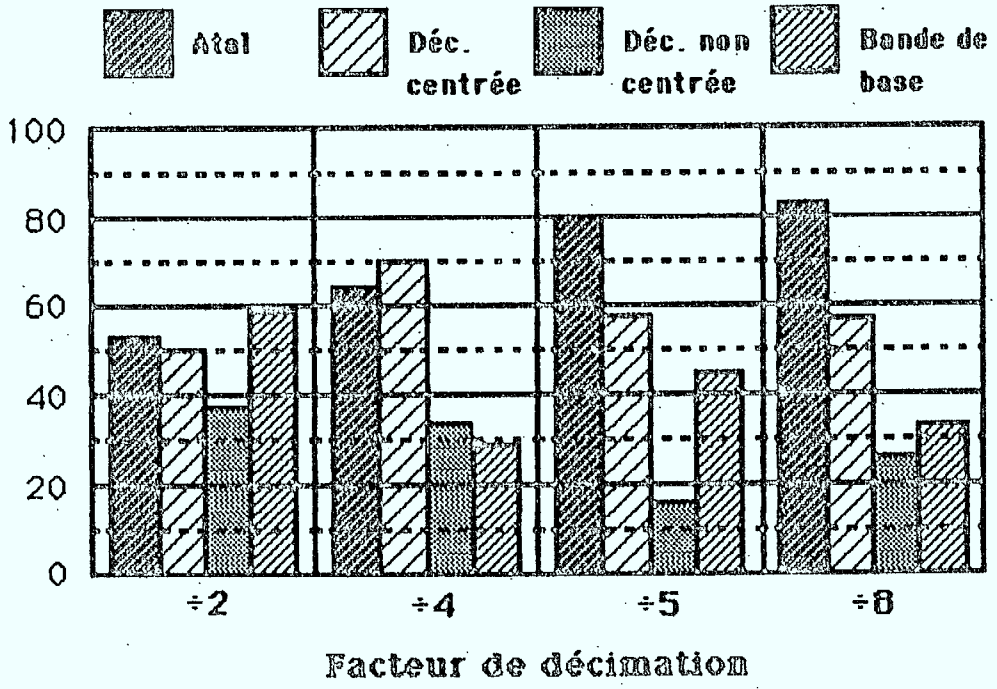


Fig. 2.2 Résultats des tests subjectifs: performance des différentes méthodes pour des facteurs de décimation de 2,4,5 et 8.

Les tests dont les résultats sont présentés ici sont fondés sur des choix effectués par les auditeurs sur des paires de codeurs. La performance d'un codeur est proportionnelle au nombre de fois où il est préféré.

Il est intéressant de constater que la décimation généralisée non centrée est supérieure, en terme de RSB, au codage en bande de base, alors que ce dernier est préféré lors d'écoutes subjectives. La contribution du centrage de la décimation généralisée n'en est que plus évidente, son effet se faisant essentiellement ressentir lors de l'évaluation par l'oreille humaine.

2.1.4 Complexité

La complexité d'un algorithme s'évalue en terme de nombre d'opérations de base (additions, multiplications, etc...) On a choisi, dans le cadre de cette étude, de l'évaluer en pourcentage du temps réel nécessaire au codage d'un bloc de parole sur un processeur TMS 320.

On a présenté dans le rapport de l'an dernier, deux moyens de simplifier l'algorithme de minimisation de l'erreur perceptuelle en réduisant sa complexité [(1) pp. 38,39,85-88]. Le premier moyen consiste à limiter l'ordre du filtre de pondération qui possède théoriquement une réponse impulsionnelle infinie. On a montré que la limitation à un ordre 21 n'affecte pas les résultats. L'autre point complexe dans l'algorithme réside dans la résolution du système linéaire donnant les amplitudes. Dans le cas de la décimation généralisée, l'ordre de ce système peut être élevé. On avait alors introduit une méthode simple pour approximer la matrice inverse, réduisant la résolution à un produit matriciel simple.

L'application de ces simplifications a conduit à l'écriture d'un programme FORTRAN, de décimation généralisée avec calculs en entiers 16

bits. L'égalité aux erreurs d'arrondis près entre les résultats obtenus par simulation sur le PDP-11 et ceux obtenus à partir d'un programme écrit pour le TMS 320, permet de s'assurer de la validité de la transcription en code TMS 320. Les complexités ont été extrapolées de cette transcription.

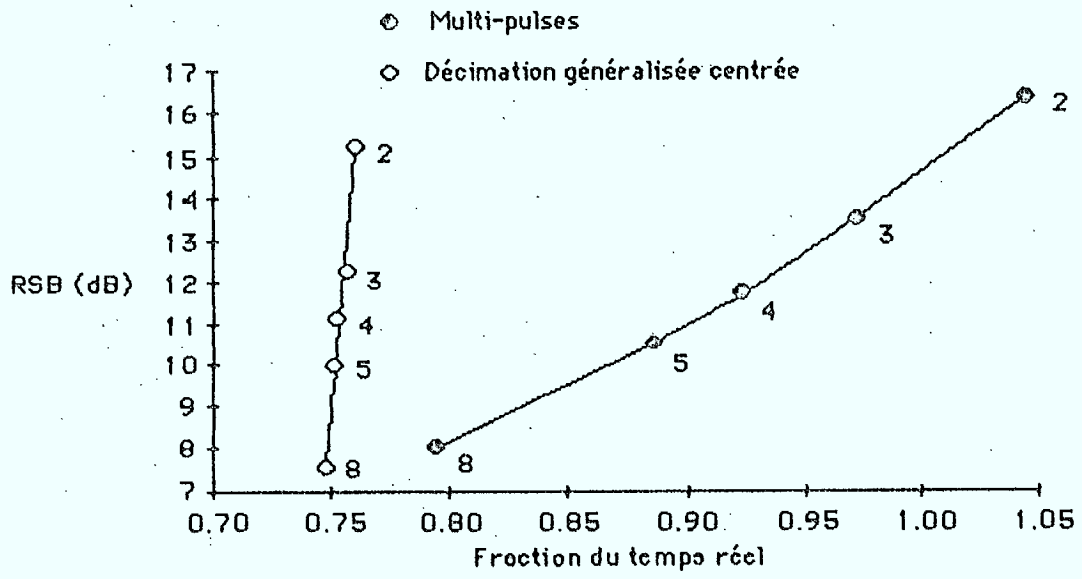


Fig. 2.3 Evaluation de la complexité des méthodes d'Atal et de décimation généralisée centrée.

N.B.: Sur ce diagramme de complexité, il n'a été tenu compte d'aucun codage. Toutefois, dans le cas de la méthode d'Atal, le codage des positions est une étape essentielle. Il convient donc de majorer les résultats qui s'y rattachent, afin d'obtenir un estimé significatif de la complexité.

2.1.5 Conclusion sur le sous-échantillonnage

Il ressort de ces différents tests que la décimation généralisée avec centrage présente un bon compromis entre les performances en termes de RSB et la complexité d'implantation sur un TMS 320.

Cette méthode utilisée conjointement avec la quantification sphérique par les lattices (E8 ou $\Lambda 24$), devrait conduire à la réalisation de codeurs, implantable en temps réel, de qualité à 4800, voire à 2400 bits par seconde.

2.2 Quantification vectorielle sphérique dans E8 [3,4]

Le principe de la quantification vectorielle revient à décrire un espace par un nombre réduit de points dispersés, choisis de manière optimale. Dans le cas de la quantification avec les lattices comme E8 l'ensemble des points du quantificateur coincident avec les points de la lattice. L'avantage des lattices est double: tout d'abord, elles permettent de générer des ensembles de points équirépartis possédant une "compacité" proche de l'optimale [1], d'autre part, ces points peuvent être générés mathématiquement, évitant ainsi le stockage d'un dictionnaire. De plus, les points d'une lattice peuvent être regroupés sur des sphères concentriques, facilitant ainsi la conception de quantificateurs sphériques. En effet, dans la quantification vectorielle sphérique (QVS), l'espace est ramené, par normalisation, sur une sphère centrée sur l'origine, découplant ainsi le codage de la norme (ou énergie) et de la direction du vecteur.

2.2.1 Codage

La lattice E8 possède la propriété suivante: tous les points obtenus par permutation des coordonnées d'un élément de E8 sont des éléments de E8. On peut ainsi définir des classes d'équivalence, ou orbites, en regroupant les éléments ayant les mêmes coordonnées à une permutation près. Le représentant dont les coordonnées sont classées par ordre décroissant, constitue le chef d'orbite. L'algorithme présenté plus bas s'appuie sur cette partition afin de simplifier les calculs.

La première étape consiste à ordonner les éléments du vecteur à quantifier du plus grand au plus petit, en gardant trace des permutations effectuées.

La seconde étape consiste à projeter le vecteur ordonné sur les différents chefs d'orbite. La classe du vecteur correspond à la projection maximale.

Le chef d'orbite ainsi sélectionné est ensuite permuté suivant les permutations inverses de la mise dans l'ordre croissant. On obtient ainsi un vecteur indiquant la direction du vecteur quantifié. Reste alors à quantifier la norme.

Vient ensuite le numérotage. Le vecteur est d'abord numéroté dans sa classe, puis on ajoute un biais afin d'obtenir l'indice final. Le principe consiste à comptabiliser toutes les combinaisons "inférieures" au sens lexicographique du terme (on attribue une valeur à un symbole, s'en

suit une relation d'ordre dans les entiers). Tout d'abord, on compte celles qui commencent par un symbole inférieur, puis parmi celles commençant par le même symbole, celles dont le deuxième symbole est inférieur au 2ième symbole du vecteur à numéroter, etc... Ce dénombrage est assez aisé dans la mesure où toutes les permutations possibles de symboles sont permises à l'intérieur d'une classe.

Le numérotage à l'intérieur d'une classe s'effectue donc en 2 étapes:

- dénombrage des éléments différents et détermination de leur poids,
- numérotage proprement dit.

2.2.2 Décodage

Le décodage s'effectue en deux temps:

- Détermination de la classe: on retire les biais à l'indice global reçu jusqu'à trouver la différence minimale non négative. Si on décrit l'ensemble des biais par ordre décroissant, la classe correspond à la première différence non négative.

- Passage de l'indice au vecteur.

Le principe du dénumérotage suit celui du numérotage. On reconstitue l'indice en veillant à chaque position dans le vecteur que la valeur accumulée approche au plus près sans jamais la dépasser, la valeur de l'in-

dice reçu (lequel cas le vecteur trouvé serait "supérieur"), ce qui permet de retrouver les éléments du vecteur. A la fin, la valeur reconstituée est égale à celle initiale de l'indice.

2.2.3 Méthode numérique

Le principe de la numérotation consiste donc à dénombrer les éléments inférieurs. Dans la mesure où toutes les permutations sont autorisées, le rang à l'intérieur d'une classe est donné par:

$$r = \sum_{i=1}^{N} \sum_{j=1}^{q_i-1} \frac{(N-i)!}{(IW_i(j)-1)! \prod_{k=1}^{Nel} IW_i(k)!}$$

N : dimension des vecteurs (8),

Nel : nombre d'éléments constitutifs,

q_i : ordre de l'élément à la ième position dans la liste classée des éléments du vecteur à numéroter (ex: si les éléments constitutifs sont $\{2,0,-1,2\}$ $q_i = 3$ si le ième elt = -1),

$IW_i(j)$: poids du jième élément constitutifs après i itérations, c'est-à-dire le nombre de fois que le jième elt constitutif apparaît dans le vecteur à la ième itération.

Cette formule s'interprête comme suit: à la ième itération, il reste Nel éléments de poids IW_i, le nombre de permutations possibles à ce stade, pour le ième élément arbitrairement fixé, est donné par (N-i)!/(...). Il suffit donc à chaque étape (i = 1,...,N), d'accumuler le

nombre de permutations possibles tant que le ième élément, arbitrairement fixé est "inférieur" en rang à l'élément effectivement présent puis d'accumuler les résultats obtenus à chaque étape.

Présenté ainsi, il semble nécessaire de calculer des produits factoriels donc d'effectuer de nombreuses multiplications. La méthode des racines permet de calculer les éléments avec un nombre réduit de multiplications et de divisions. Il est pour cela nécessaire de tabuler une constante: la racine RAC définie par:

$$RAC = \frac{8!}{\prod_{k=1}^{Nel} IW_1(k)!}$$

RAC est entière, puisque c'est le nombre de toutes les permutations possibles dans la classe. En faisant évoluer RAC suivant la fonction:

$$RAC(i) = \frac{(9-i)!}{\prod_{k=1}^{Nel} IW_i(k)!}$$

la formule s'écrit:

$$\begin{cases} r = \sum_{i=1}^{8} \frac{1}{(9-i)} \sum_{j=1}^{q_i-1} IW_i(j) * RAC(i) \\ RAC(i+1) = RAC(i) * IW_i(q_i)/(9-i) \end{cases}$$

La complexité se réduit donc à 16 divisions et (8+x) multiplications (x dépendant du vecteur). La méthode s'applique aisément au calcul

en entiers puisqu'on est assuré que les divisions tombent justes. On procède à l'inverse pour le dénumérotage

2.2.4 Logiciels de QVS en temps réel

L'algorithme de quantification et la méthode de numérotation par racine, présentés dans les chapitres précédents, se prêtent particulièrement bien à une transcription en langage assembleur. Ces algorithmes ont été transcrits pour le TMS 320 et pour le MC68008, ce processeur pouvant, le cas échéant, être utilisé au décodeur à la place du TMS 320 (cf. chap 4.2). L'utilisation de ces algorithmes nécessite la tabulation des chefs d'orbites, des biais de numérotation et des racines, soit 10 mots par orbite. Les performances et les besoins de stockage sont résumés dans le tableau suivant.

QVS dans E8		1 bit/éch.	2 bit/éch.
Numéro de sphère		1	6
Nbre pts du quanti-ficateur		240	60480
Stockage nécessaire		80 mots	570 mots
Temps exécution:			
- quantification	TMS	0.41 ms max	1.6 ms max
	68008	2.2 ms max	9.1 ms max
- numérotation	TMS	0.26 ms max	0.29 ms max
	68008	0.85 ms max	1.1 ms max
- décodage	TMS	0.32 ms max	0.48 ms max
	68008	1.1 ms max	1.6 ms max
Grand total	TMS	1 ms max	2.4 ms max
	68008	4.2 ms max	12 ms max

Tableau 2.1

CHAPITRE 3

SYSTEME DE DEVELOPPEMENT DE VOCODEURS

Dans ce chapitre nous allons décrire les travaux effectués pour la mise en oeuvre d'un outil de développement logiciel pour le codage de la parole à bas débit. Ce logiciel se situe, à la fois, dans les domaines de la simulation et de la réalisation. En effet, nous avons opté pour un système de développement permettant de simuler les vocodeurs avec un logiciel identique à celui utilisé sur une maquette.

Cette situation, réduisant le nombre des étapes menant à la réalisation matérielle d'un vocodeur, est rendue possible grâce à l'apparition sur le marché d'un "processeur vectoriel" basé sur le microprocesseur spécialisé en traitement du signal que nous avions choisi: le TMS 320. Le processeur vectoriel en question est le SKY 320-Q qui a été annoncé il y a un an environ. Nous sommes un des premiers acheteurs de la version de ce processeur vectoriel pour le Q-bus (il en existe une autre version pour le PC d'IBM).

Le choix du TMS 320 qui a été discuté dans le rapport de l'an dernier reste toujours valable. En effet, aucun autre produit n'est encore venu remplacer ce circuit tant au niveau des performances que du prix.

3.1 Procédure de développement de vocodeurs

Jusqu'à présent, la mise au point de vocodeurs numériques se faisait en deux étapes principales: la mise au point par simulation d'un algorithme et la réalisation d'une maquette où cet algorithme de codage était mis en oeuvre. L'étape de simulation se fait généralement dans un langage évoluée (FORTRAN, PASCAL ou C), sur un mini-ordinateur ou sur une machine plus conséquente.

Cette simulation offre l'avantage d'être très flexible: il est facile de modifier l'algorithme. Par contre elle ne se fait pratiquement jamais en temps réel. L'évaluation de l'algorithme de codage se fait à l'aide de mesures objectives ou de tests subjectifs sur une quantité limitée de parole.

La seconde étape consiste en la construction d'une maquette basée bien souvent sur un ou plusieurs microprocesseurs. Une part importante de cette étape est donc l'écriture du logiciel, réalisant l'algorithme simulé dans la première étape, et qui va tourner sur les processeurs de la maquette.

La procédure de développement que nous avons adoptée, permet d'utiliser en grande partie, le même logiciel pour la simulation et pour la maquette. En effet, l'utilisation du SKY 320, pour une partie de la simulation, permet d'introduire à ce niveau du code qui va rester le même jusqu'au produit final. En fait, ces sections de programme sont celles qui caractérisent le vocodeur comme la quantification vectorielle à l'aide d'un

dictionnaire de filtres, de formes d'onde ou à l'aide d'une lattice ou bien la décimation généralisée. Le reste du code, comme les opérations d'entrée-sortie de parole, n'est pas spécifiques à un vocodeur en particulier, et va varier selon que nous avons affaire au programme de simulation ou à la maquette.

En fait, toute la simulation ne se fera pas sur le SKY 320, une première phase d'exploration des algorithmes va continuer à être réalisée en FORTRAN en conjonction avec le logiciel ILS. Les opérations faisant intervenir une quantité importante de calcul comme la construciton des dictionnaires va, elle aussi, continuer à se faire sur le processeur vectoriel MAP-300.

Par contre, la mise au point final des algorithmes et notamment leur implantation en virgule fixe va systématiquement utiliser le SKY 320.

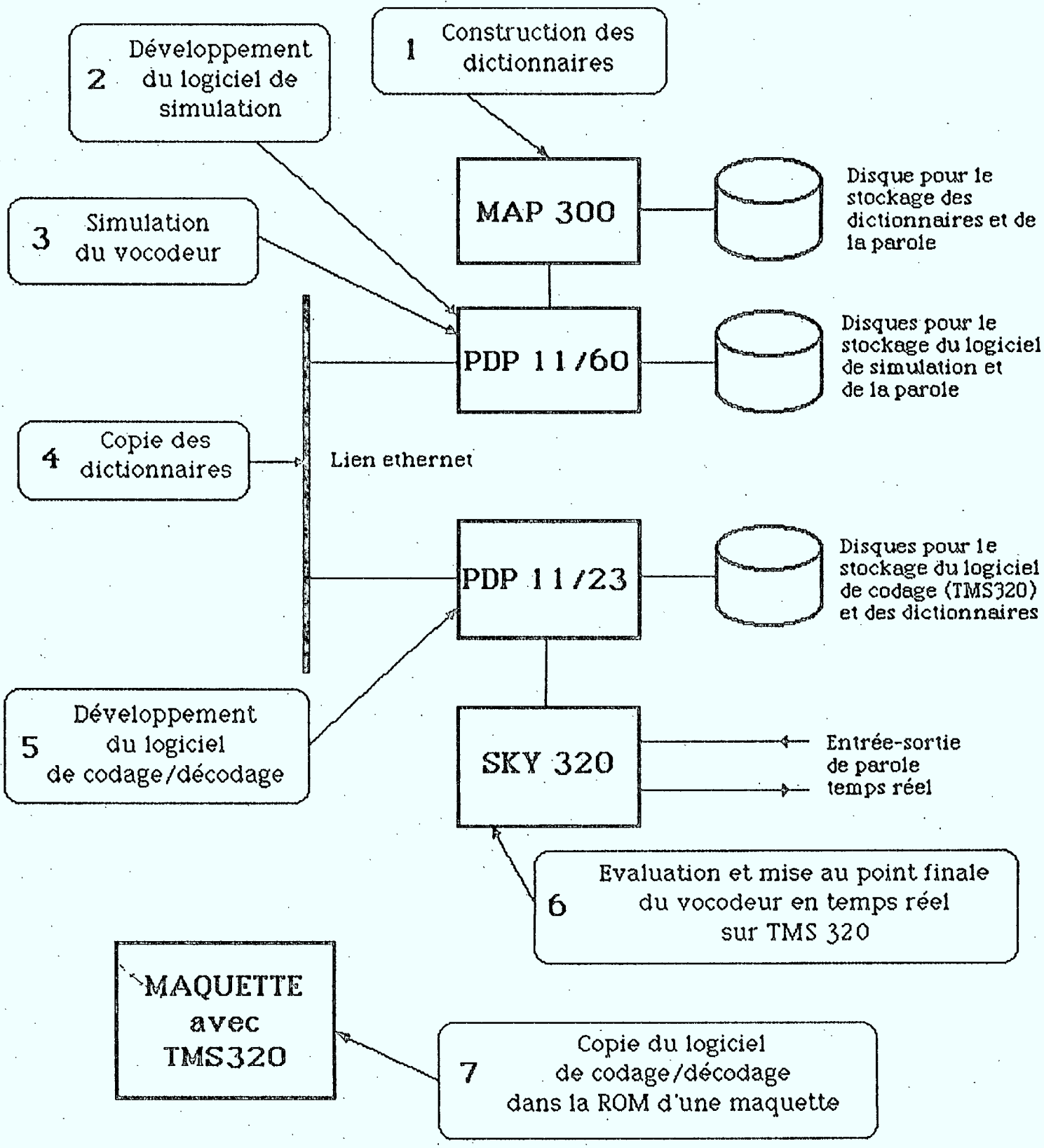


Fig. 3.1 Etapes de la procédure proposée pour la mise au point de vocodeurs.

3.2 Le processeur vectoriel SKY 320

3.2.1 Matériel

Le SKY 320 est construit autour d'un microprocesseur spécialisé pour le traitement du signal TMS 320. Le processeur est basé sur une architecture Harvard avec la mémoire des données interne au circuit et la mémoire programme externe (TMS 32010). Sur le SKY, cette mémoire programme est constituée de 4K mots de 16 bits de mémoire vive accessible aussi bien par le TMS 320 que par la machine hôte qui est une micro PDP-11. Le programme exécuté par le TMS 320 est donc chargé dans cette mémoire par la micro PDP-11 dont on peut utiliser les mémoires de masse (disque dur) pour le stockage du logiciel. L'édition et l'assemblage de ce même logiciel se feront aussi sur cette machine.

En raison de la taille très réduite de la mémoire de données interne du TMS 320 (144 mots de 16 bits), une mémoire externe pour les données est présente sur la carte SKY. Cette mémoire vive comporte 64K mots de 16 bits. Elle est accessible non seulement pour le TMS 320 mais aussi par le Q-Bus (micro PDP-11) et par deux canaux d'accès direct mémoire, un en entrée et l'autre en sortie. Ces deux ports seront utilisés en conjonction avec des convertisseurs analogiques pour l'acquisition et la restitution de parole en temps réel.

La communication entre le TMS 320 et la micro PDP-11 peut s'effectuer selon deux méthodes:

- En utilisant un registre spécial appelé "registre de communication" et qui permet l'échange d'un mot de 16 bits à la fois.
- A l'aide de la mémoire de 64K accessible par la micro PDP-11 et par le TMS 320 en utilisant des instructions IN et OUT.

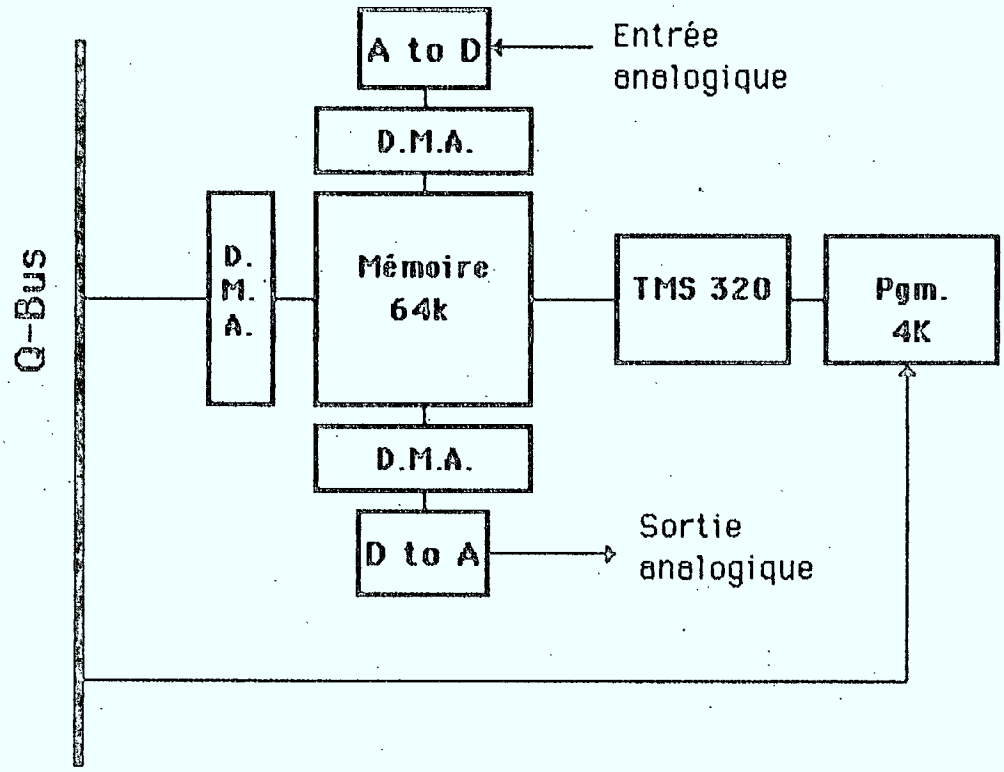


Fig. 3.2 Architecture du processeur vectoriel SKY 320.

La synchronisation de ces échanges est rendue possible grâce à la présence de deux lignes d'interruptions du TMS 320 vers la µPDP et de deux autres lignes de la µPDP vers le SKY 320.

C'est le TMS 320 qui contrôle les canaux de DMA des ports d'entrée-sortie sur le bloc mémoire. Ce contrôle s'effectue en utilisant des instructions IN et OUT comme pour accéder à la mémoire ou au registre de communication avec la µPDP. Cette mémoire de 64K dispose d'une circuiterie de rafraîchissement automatique et de gestion des conflits d'accès qui en rend l'usage très facile. De plus, un système d'auto-incrément par une puissance de 2 de ses registres d'adresse en permet un usage simplifié pour l'accès à des vecteurs.

La mémoire programme est, elle, d'architecture plus simpliste. Elle ne peut être accédée par la µPDP que lorsque le TMS 320 est arrêté. Il n'est donc pas possible de modifier, d'une façon dynamique, depuis la µPDP le logiciel s'exécutant sur le SKY.

3.2.2 Logiciel

Le logiciel fourni avec le SKY 320-Q fonctionne sous le système d'exploitation RT11. Il comprend un programme de test du matériel, un compilateur C et un assembleur pour le TMS 320 et un mini-système d'exploitation s'interfaçant avec des programmes sous RT11 écrits en C.

Nous utilisons la µPDP avec le système d'exploitation RSX11M en raison des possibilités multiples offertes par ce logiciel et pour assurer une compatibilité complète avec les autres machines que nous utilisons (PDP-11/60 et PDP-11/24). En effet, ces trois machines sont interconnectées sur un réseau Ethernet avec lequel nous utilisons un logiciel d'échange de fichiers (développé par nos soins) fonctionnant sous RSX11M.

Toutes ces considérations nous ont amené à écrire notre propre logociel d'utilisation du SKY 320 sous RSX11M. Bien que ce travail représente un effort relativement important, il est indispensable, si l'on veut pouvoir utiliser efficacement le SKY, pour l'usage auquel nous le destinons: la mise en oeuvre de vocodeurs en temps réel.

Le logiciel d'utilisation du SKY 320 peut être divisé en trois parties qui correspondent chacune à un niveau d'accès au processeur vectoriel. Le niveau le plus bas dans cette hiérarchie concerne l'interface entre un programme utilisateur écrit en langage évolué (FORTRAN) et le SKY 320. L'utilisateur de ce premier niveau de logiciel va pouvoir directement contrôler le contenu des régistres du SKY visibles depuis le Q-Bus.

Le second niveau dans la hiérarchie offre au programmeur en langage évolué un certain nombre de services comme le chargement dans le SKY d'un programme pour le TMS 320, le contrôle de l'exécution de ce programme et l'échange de données avec le bloc mémoire de 64K.

Enfin, le troisième et dernier niveau comporte une série de programmes d'application qui permettent d'opérer un logiciel de traitement sur le TMS 320 avec des données issues et à destination de convertisseurs analogiques ou de fichiers format ILS.

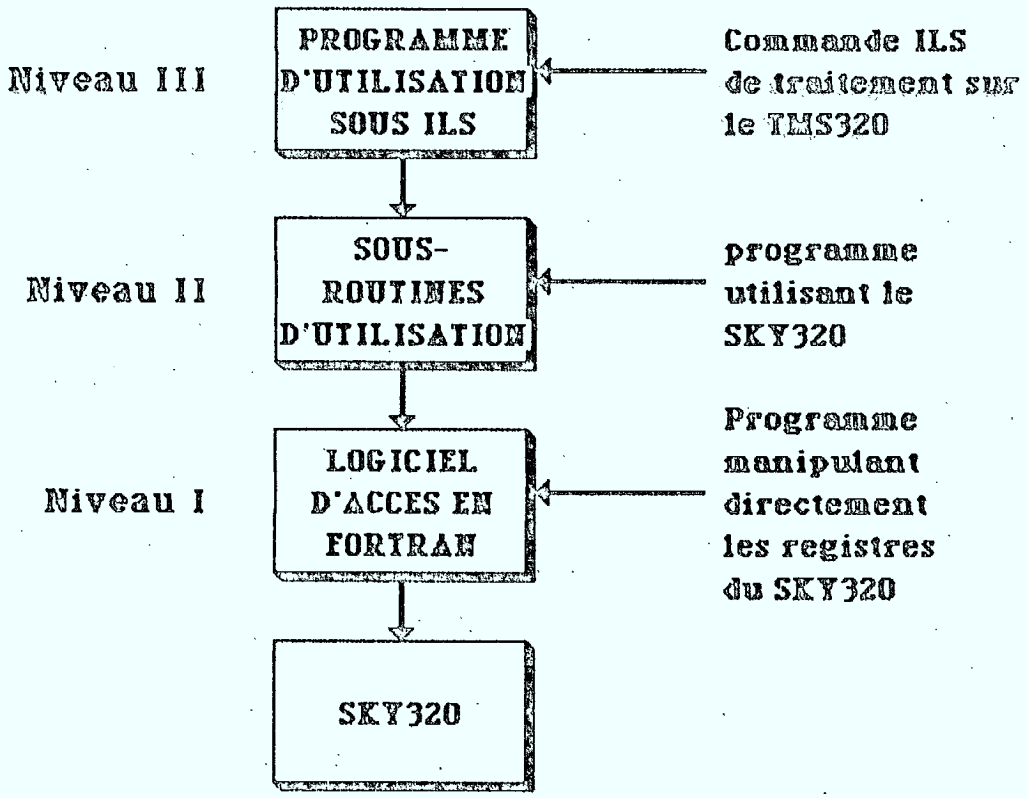


Fig. 3.3 Structure du logiciel d'utilisation du SKY 320.

3.2.2.1 Interface Fortran sous RSX 11M pour le SKY 320

Sous RSX 11M, les tâches utilisateurs n'ont pas accès aux registres des périphériques. Cette restriction permet au système d'exploitation de garder un contrôle absolu sur l'usage des ressources du système puisque toutes les requêtes aux périphériques doivent passer entre ses mains. Le code qui effectue directement le contrôle d'un périphérique est appelé "driver" de ce périphérique. Le driver reçoit les demandes des tâches qui lui sont transmises par l'exécutif et doit y satisfaire avec l'aide de celui-ci. L'utilisation d'un driver n'est pas la seule solution pour uti-

liser un périphérique sous RSX 11M. Elle n'est en fait indispensable que lorsque le périphérique doit être partagé par plusieurs tâches ou bien lorsqu'il comporte une unité de stockage structurée (disque).

Dans le cas du SKY 320, la solution du driver n'a pas été retenue car ce processeur vectoriel ne sera utilisé que par une seule tâche à la fois. De plus, les conditions de travail sur la µPDP (un ou deux utilisateurs maximum), font que la protection du système n'est pas critique.

La méthode d'accès aux registres du SKY qui a été choisie est celle du "device common". Il s'agit d'une partition connue par le système et définie dans la page d'entrée-sortie. Une tâche utilisateur peut accéder à ce device common et manipuler ainsi directement les registres du périphérique. Ces registres deviennent des adresses visibles dans l'espace d'adressage de la tâche.

Le logiciel d'interface FORTRAN pour le SKY 320 est donc constitué d'un ensemble de 7 sous-routines écrites en assembleur. Elles permettent d'aller lire et écrire dans chacun des 4 registres du SKY 320.

En voici une brève description:

- WSTCR(IDATA), RSTCR(IDATA) écriture et lecture du registre de statut/contrôle du SKY 320.

- WCOMR(IDATA), RCOMR(IDATA) écriture et lecture du registre de communication entre la µPDP et le SKY 320.

- WMADDR(IADR) écriture du registre d'adresse de la mémoire programme ou de la mémoire de données du SKY 320. Dans le cas de la mémoire de

données, ce registre est automatiquement incrémenté après chaque lecture ou écriture.

- WMDATA(IDATA), RMDATA(IDATA) écriture et lecture dans la mémoire programme ou la mémoire de données externe du TMS 320.

3.2.2.2 Logiciel d'utilisation du SKY 320 en FORTRAN

A ce second niveau d'accès au SKY, l'utilisateur dispose d'un certain nombre de sous-routines qui vont lui permettre d'exécuter un programme écrit en assembleur TMS 320 en relation avec son programme FORTRAN tournant sur la µPDP. Le programme pour le TMS 320 est assemblé par le MACRO11 en utilisant une série de macros définissant les instructions du TMS 320.

Les opérations réalisables par ce logiciel sont:

- Le chargement dans la mémoire du TMS 320 d'un programme contenu dans un fichier objet (.OBJ) issu du MACRO II.
- Le démarrage et l'arrêt du TMS 320.
- Le transfert d'un vecteur d'entiers définis depuis un tableau FORTRAN vers la mémoire externe de données du SKY 320 et inversement.
- La transmission d'une interruption au TMS 320 depuis la µPDP.

3.2.2.3 Programme de traitement dans le SKY 320 sous ILS

Le troisième niveau du logiciel permet l'utilisation directe de logiciels écrits pour le TMS 320 sur des fichiers de données au standard ILS. Les échantillons issus du fichier primaire ILS sont placés dans la mémoire externe de données du SKY. Le TMS 320 les prélève comme s'il s'agissait d'échantillons venant d'un convertisseur analogique à numérique. Après traitement, ces échantillons sont replacés dans la mémoire où ils sont prélevés par le programme sur la µPDP et placés dans le fichier secondaire ILS.

Un système de double buffer est utilisé pour les échanges dans les deux sens. Le registre de communication sert pour synchroniser l'utilisation des buffers entre les deux processeurs. Un programme permettant le chargement d'un dictionnaire hiérarchique dans la mémoire externe de données est aussi disponible.

En conclusion, un tel logiciel permet de mettre en oeuvre un algorithme sur le TMS 320 dans des conditions identiques à celles d'une maquette fonctionnant avec des convertisseurs analogiques.

3.2.2.4 Traitement par la µPDP des interruptions issus du SKY 320

Une des fonctions du driver d'un périphérique est la prise en charge des interruptions issues de ce périphérique. L'utilisation d'un common device présente l'inconvénient de ne pas permettre à la tâche de

l'utilisateur l'utilisation d'interruptions asynchrones venant du périphérique. Une facilité de RSX 11M permet l'usage des interruptions matérielles pour une tâche privilégiée sans avoir recours à un driver: c'est la directive CINT "connection à un vecteur d'interruption".

Deux sous-routines ont été développées pour permettre à un programme FORTRAN s'exécutant avec le statut de tâche privilégiée (sans accès mémoire à l'exécutif et à la page d'entrée-sortie) de manoeuvrer des drapeaux (event flags) ou d'activer des routines d'interruption asynchrones (AST) en FORTRAN sous le contrôle du TMS 320. Ce dernier élément du logiciel permet un parallélisme complet entre les opérations de la μPDP et du SKY.

CHAPITRE 4

REALISATIONS MATERIELLES EN VUE DE L'IMPLANTATION EN TEMPS REEL

4. Implantation temps réel

Mis à part le processeur SKY320, les outils de développement de matériels incluant le circuit TMS 320 disponibles au laboratoire, se résument à un ensemble carte EVM (module d'évaluation) + carte AIB (interface analogique numérique). La carte AIB [8] a été modifiée afin d'y inclure les fonctions supplémentaires suivantes:

- horloge d'échantillonnage externe,
- préampli audio,
- stockage externe de constantes (carte dictionnaire),
- interfaçage avec le synthétiseur de parole SP1000.

4.1 Carte dictionnaire

Cette carte a été développée en vue d'y stocker des volumineuses tables de constantes tels que les dictionnaires. Le design et l'utilisation s'inspirent du fonctionnement de la mémoire de données de 64 K mots contenue dans la carte SKY.

La configuration de base permet d'obtenir une page de 64K mots avec 16 mémoires 2764; toutefois, en combinant des circuits de capacités supérieures (27128 ou 27256), et en jouant sur la pagination, on peut obtenir toutes les configurations de 1 x 8 Kmots à 4 x 64 Kmots en passant par 1 x 64 Kmots, 2 x 32, etc...

L'espace mémoire est visible du TMS 320 par l'intermédiaire des 2 ports d'entrée-sortie auxiliaires offert par la carte AIB. Un port permet d'écrire l'adresse, que l'on désire lire, dans un compteur programmable. Une lecture d'un des deux ports provoque une lecture de la mémoire avec post incrémentation ou décrémentation du compteur d'adresse dépendant du numéro de port utilisé.

L'utilisation de cette carte est résumée par la figure 4.1. Le plan de la carte est donnée en figure 4.2.

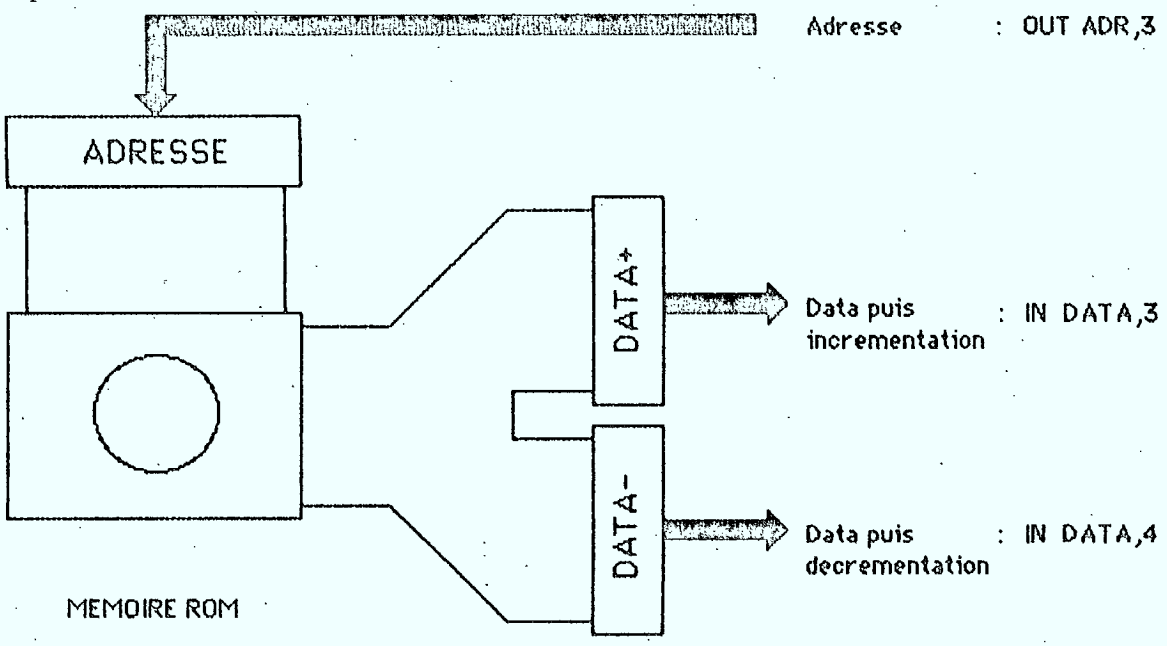


Fig. 4.1 Schéma fonctionnel de la carte dictionnaire.

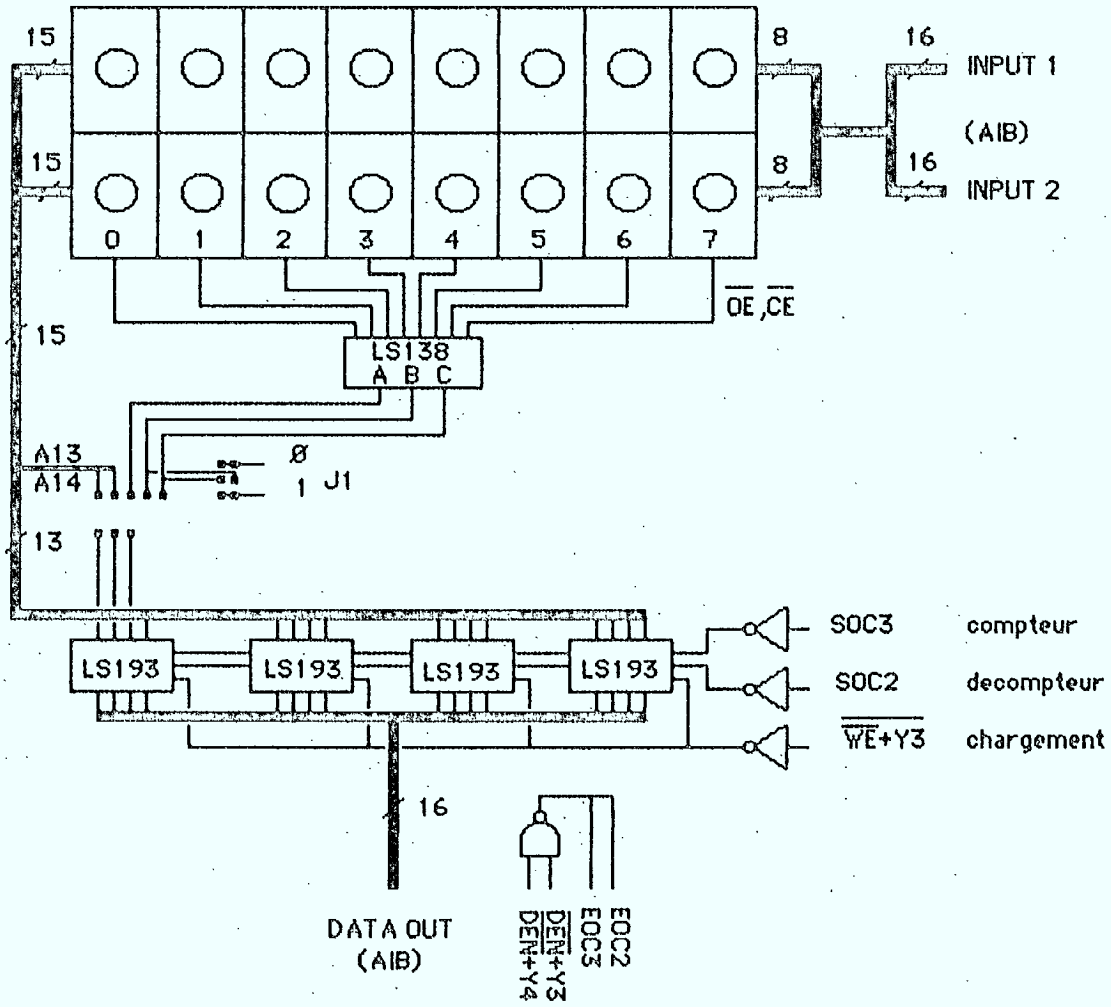


Fig. 4.2 Plan de la carte dictionnaire.

4.2 Utilisation d'un circuit de synthèse

Parallèlement au développement de logiciel de codage de la parole en temps réel sur le TMS 320, il a semblé intéressant d'évaluer les avantages que l'on pourrait retirer de l'utilisation d'un synthétiseur de parole comme le SP 1000 de General Instrument. Le SP1000 est un circuit polyvalent, utilisant la technique LPC. Il se comporte comme un périphérique de micro-processeur, pouvant se connecter facilement sur un bus de 8 bits. Une lecture rapide de la notice (Annexe A), permet de constater la souplesse d'utilisation du circuit. Les principales caractéristiques sont résumées dans le tableau 4.1. Les différentes possiblités offertes ont été testées et se sont révélées fonctionnelles.

Ordre du filtre		10
Codage des coefficients		9 bits/linéaires
Excitation	7 types	non voisée 5 formes d'ondes voisées excitation externe
Codage de l'amplitude		logarithmique, signé
dynamique		96 dB, pas de .75 dB
Précision interne des calculs		16 bits
Cadrage final		48 dB, pas de 6 dB
Résolution D → A		8 bits
Fréquence d'échantillonnage		programmable de 4.0 KHz à 12.7 KHz
Particularité		peut effectuer une analyse LPC à des fins de reconnaissance de la parole

Tableau 4.1

Le matériel développé pour le SP1000 permet de le connecter sur un port auxiliaire du TMS 320 par l'intermédiaire de l'AIB (fig. 4.3). Les données (sur 8 bits) et les différents modes (A0, A1,R/W), sont multiplexés dans le même mot de 16 bits présenté sur le port, c'est pourquoi ce module ne fonctionne seulement qu'en écriture. Le signal WAIT renvoyé sur la broche BIO du TMS permet d'arbitrer les échanges entre le TMS et le synthétiseur.

Cette carte a permis de contrôler la qualité de la parole synthétique restituée par le synthétiseur, à partir d'une analyse pratiquée avec ILS. D'autre part, la réalisation d'une chaîne d'analyse-synthèse LPC, utilisant le TMS 320 et le SP1000 a été réalisée, à l'exception de l'extracteur de fréquence fondamentale. Si nous avions disposé de cet élément, nous aurions pu démontrer la réalisabilité d'un vocodeur LPC réduit à un A/D un SP1000, un TMS 320 et un peu de logique.

Puisqu'il peut être utilisé en mode excitation externe, ce circuit permet de réaliser des décodeurs RELP à bon marché (1 SP1000 = 15$) tant que l'ordre du filtre de synthèse demeure inférieur ou égal à 10. Les avantages d'une telle méthode sont multiples:

- le circuit possède son propre étage D → A,
- le processeur du récepteur n'a plus à s'occuper du filtrage de synthèse, donc on peut utiliser un circuit plus polyvalent comme le 6809 ou le 68008 suivant la complexité des quantificateurs à la place du TMS 320.
- le matériel reste souple à utiliser à cause des possibilités de programmation de synthétiseur.

Toutes ces remarques permettent d'envisager la réalisation économique d'un récepteur de bonne qualité, construit autour du SP1000.

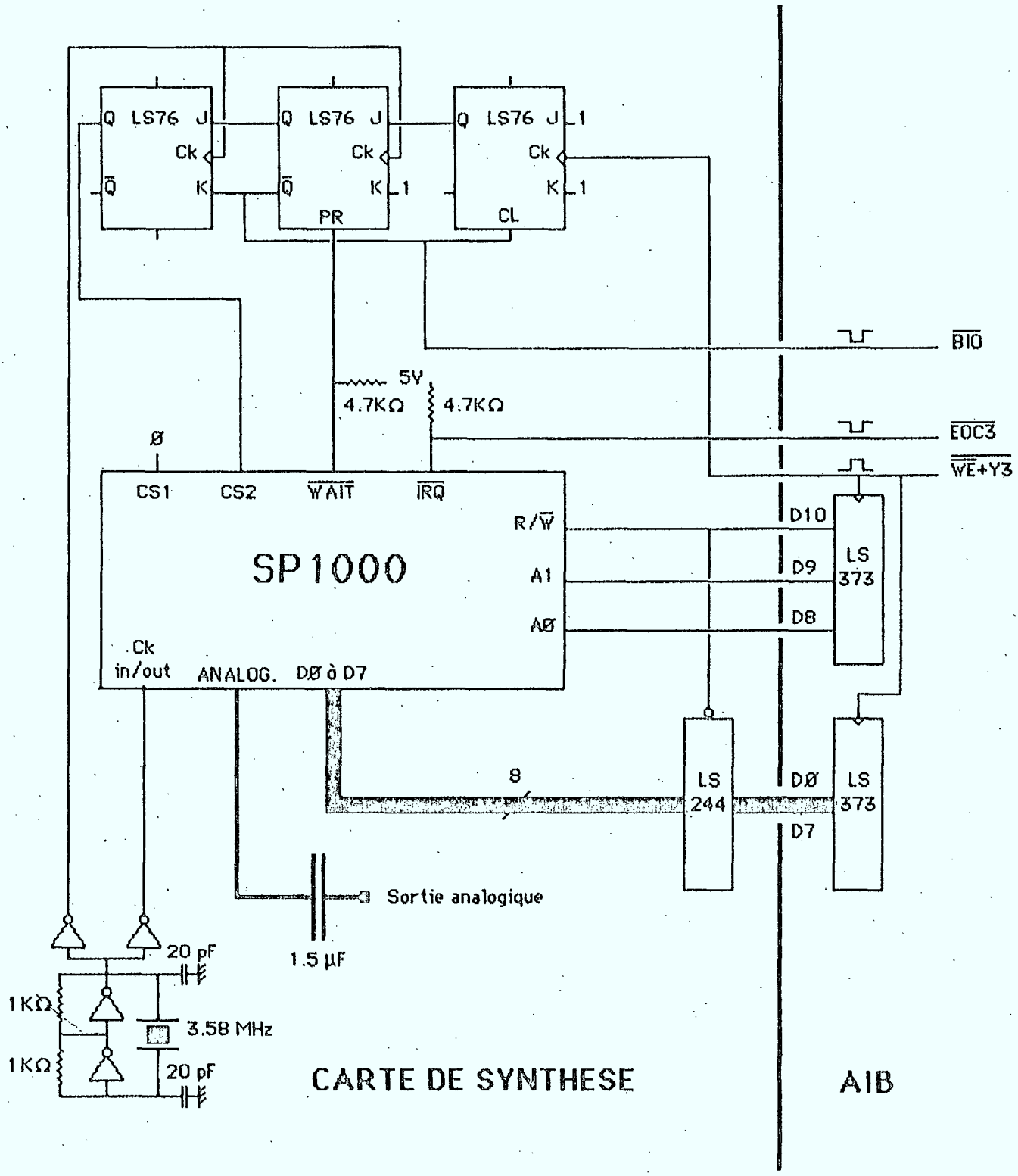


Fig. 4.3 Plan de la carte synthèse.

CHAPITRE 5

RESULTATS

L'objet de la présente étude porte sur l'amélioration de la qualité de la parole transmise en augmentant par exemple la fréquence d'échantillonnage. Parallèlement à l'étude de l'intégration sur le TMS 320, de tels codeurs, il a semblé pertinent de mesurer l'impact de ce changement de fréquence d'échantillonnage. C'est pourquoi les résultats présentés dans ce chapitre portent à la fois sur des codeurs travaillant à 8 KHz ou à 10 KHz de fréquence d'échantillonnage.

5.1 Description des codeurs

Le problème abordé ici est la transmission du signal de parole sous forme numérique à un débit de 4800 bits/seconde. Un tel débit impose l'utilisation de la prédiction linéaire (LPC) et d'une excitation résiduelle (RELP). L'accent est mis ici sur l'amélioration de la qualité de la parole transmise. En effet, si les travaux effectués ces dernières années ont permis une bonne intelligibilité à des débits aussi bas que 2400 bits/seconde, la qualité de la parole obtenue reste médiocre.

Un des facteurs important pour la qualité de la parole obtenue est la bande passante allouée au signal. Dans les vocodeurs utilisés actuellement, le signal est filtré en bande téléphonique, c'est-à-dire à 3.4

KHz. Il semble qu'une extension de cette bande à 4.7 KHz apporterait une amélioration de la qualité et de l'intelligibilité surtout au niveau des sons non voisés. Or un élargissement de la bande passante du signal implique une augmentation du nombre de pôles du modèle prédictif utilisé dans les vocodeurs (LPC) ainsi qu'une augmentation de la cadence d'échantillonnage du signal résiduel.

L'autre facteur important pour la qualité de la parole transmise est le rythme d'adaptation du filtre prédicteur. En effet, la capacité du vocodeur à reproduire fidèlement les transitions est liée à la rapidité avec laquelle le filtre prédicteur pourra changer. On conçoit donc facilement que plus le filtre sera transmis souvent, meilleure sera la qualité de ces transitoires. Dans les vocodeurs actuels, un nouveau filtre est transmis tous les 20 ms environ. Il semble que si cette valeur était amenée à une dizaine de ms, une amélioration sensible serait possible.

Ces deux facteurs aboutissent inévitablement sur une augmentation du débit numérique. Toutefois, la quantification vectorielle, développée depuis quatre ans environ au CRCS, offre la possibilité d'améliorer les performances du codage du filtre prédicteur et du signal résiduel. Si ces nouvelles techniques ont, jusqu'à présent, été utilisées dans le but de réduire le débit numérique, il est également possible de tirer profit de leur efficacité pour obtenir un meilleur signal.

L'utilisation de la quantification vectorielle du filtre prédicteur permet de travailler avec un ordre de prédiction supérieur, 12 au lieu de 10 par exemple, sans augmenter le nombre de bits alloués à la transmis-

sion de chaque filtre. Cette possibilité vient du fait que le nombre de bits nécessaires à la transmission de chaque filtre dépend de la taille du dictionnaire utilisé, et non plus du nombre de paramètres de représentation des filtres comme dans le cas de la quantification scalaire.

Pour le codage du résiduel, les quantifications sphériques avec les lattices ou vectorielles avec dictionnaires sont envisageables. Toutefois, la première méthode ne nécessite pas un stockage aussi important que la seconde. D'autre part, pour maintenir la qualité de la quantification malgré le nombre réduit de bits disponibles, un sous-échantillonnage du résidu est indispensable. On a montré au chapitre 2.1 que la décimation généralisée était la méthode idéale, compte tenu des contraintes de puissance de calcul et de débit, en effet toutes les solutions proposées sont étudiées en vue de leur réalisation à l'aide de micro-processeurs TMS 320 spécialisés en traitement du signal. L'utilisation de micro-processeurs plus universels est aussi envisagée pour certaines étapes de l'algorithme faisant plus appel à des opérations logiques qu'à des calculs. Ceci est particulièrement vrai pour les opérations de numérotation dans le cas de la quantification vectorielle sphérique (2.2).

Les codeurs présentés, qu'ils soient à 8 KHz ou à 10 KHz de fréquence d'échantillonnage, se fondent sur le diagramme de blocs de la figure 5.1.

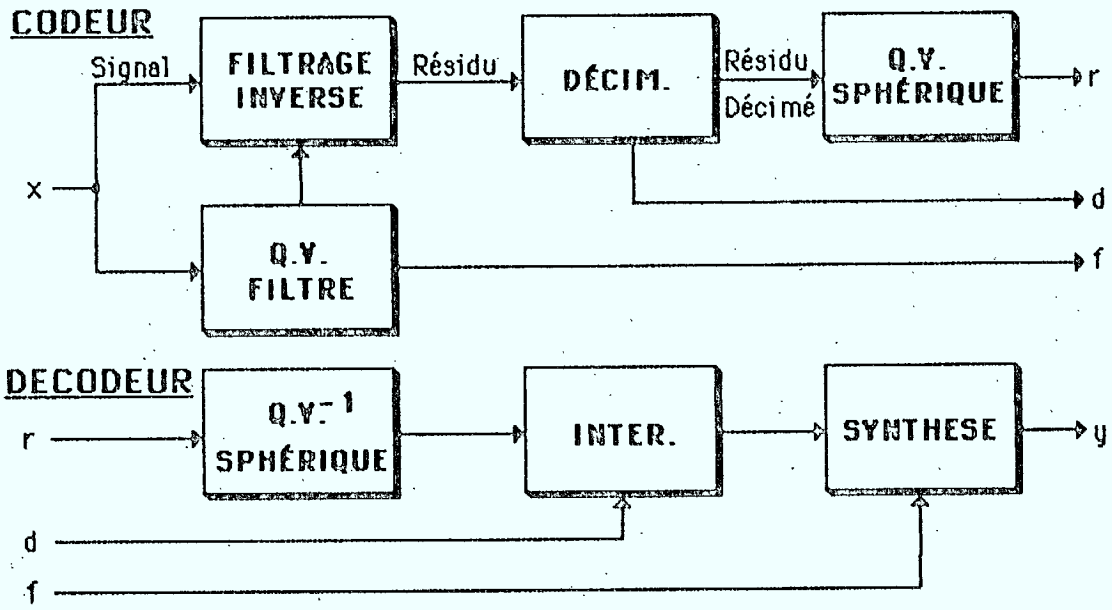


Fig. 5.1 Structure des codeurs étudiés.

Les caractéristiques techniques comparées des codeurs utilisés par la suite sont rassemblées dans le tableau suivant:

Fréquence d'échantillonnage	10 KHz	8 KHz
Largeur des fenêtres d'analyse LPC	144 éch(14.4 ms)	160 éch(20 ms)
Ordre de l'analyse LPC	12	10
Taille du dictionnaire filtre	1024	1024
Largeur des blocs de calcul de la DG.	48 éch(4.8 ms)	40 éch(5 ms)
Facteur de décimation	6	5
Bande de base équivalente	830 Hz	800 Hz
Quantificateur	E8 sphère 6	E8 sphère 6
Transmission du filtre	700 bits/s	500 bits/s
QVS	3360 bits/s	3200 bits/s
Phase initiale	490 bits/s	500 bits/s
Energie	(250) bits/s	(600) bits/s
Débit total	4800 bits/s	4800 bits/s

Tableau 5.1 Caractéristiques des codeurs à 4800 bits/s.

N.B.: Les codeurs utilisés en simulation n'effectue pas le codage de l'énergie pour lequel une méthode "delta" est envisageable. Le débit indiqué représente la différence entre le débit total et le débit utilisé.

5.2 Tests subjectifs

Ces tests ont été effectués afin de mesurer non seulement l'impact de la fréquence d'échantillonnage mais aussi celui des méthodes de calcul et de quantification du résidu décimé sur la qualité de la synthèse. Trois axes ont été retenus:

- influence du facteur de décimation,
- influence de la QVS,
- influence du calcul en entiers.

On a calculé à cette fin une série de phrases synthétiques n'introduisant qu'un des aspects présentés à la fois. L'ensemble de ces trois aspects, correspondant au codage du résidu décimé et quantifié, implantable en temps réel sur le TMS 320. Il faut noter qu'on a toujours utilisé un filtre LPC quantifié, cette étape n'étant pas l'objet de la présente étude. Les 3 séries de tests subjectifs effectués sont décrits dans les tableaux suivants, regroupant les caractéristiques des synthèses confrontées.

1° phrase	f_e = 8 KHz	DG par 5 sur bloc de 40
2° phrase	f_e = 8 KHz	DG par 4 sur bloc de 40
3° phrase	f_e = 10 KHz	DG par 6 sur bloc de 48
4° phrase	f_e = 10 KHz	DG par 4 sur bloc de 48

<u>1er test:</u> Influence du facteur de décimation (tous les calculs effectués en réel, pas de QVS).

1° phrase	f_e = 8 KHz	DG par 5 sur bloc de 40	sans QVS
2° phrase	f_e = 8 KHz	DG par 5 sur bloc de 40	sans QVS
3° phrase	f_e = 10 KHz	DG par 6 sur bloc de 48	sans QVS
4° phrase	f_e = 10 KHz	DG par 6 sur bloc de 48	avec QVS

<u>2ième test:</u> Influence de la quantification vectorielle sphérique (décimation généralisée effectuée en réel).

1° phrase	f_e = 8 KHz	DG par 5 sur bloc de 40	calcul réel
2° phrase	f_e = 8 KHz	DG par 5 sur bloc de 40	calcul entier
3° phrase	f_e = 10 KHz	DG par 6 sur bloc de 48	calcul réel
4° phrase	f_e = 10 KHz	DG par 6 sur bloc de 48	calcul entier

3ième test: Influence du calcul en entier (pas de QVS)

Tableau 5.2 Description des tests subjectifs.

5.3 Résultats

Chaque test a été effectué avec quatre phrases différentes, phonétiquement équilibrées, prononcées par quatre locuteurs différents (deux hommes et deux femmes). Les 72 écoutes comparatives que représente l'ensemble des tests, ont été soumises en une seule fois à chacun des 12 évaluateurs.

Les résultats sont présentés d'une part, de manière graphique, d'autre part sous forme d'un tableau numérique. Le score indiqué représente le nombre de fois où le codeur a été préféré. La note maximale pour chaque groupe de locuteur est 72; elle est de 144 pour le total.

On a introduit une discrimination entre locuteurs et locutrices, mettant en évidence des phénomènes intéressants, (cf. 5.4).

Influence de la décimation	SCORE		
	Homme	Femme	Total
No. 1 8 KHz ÷ 5	37	43	80
No. 2 8 KHz ÷ 4	54	50	104
No. 3 10 KHz ÷ 6	11	8	19
No. 4 10 KHz ÷ 4	42	43	85

Influence de la QVS	SCORE		
	Homme	Femme	Total
No. 1 8 KHz sans QVS	62	64	126
No. 2 8 KHz avec QVS	33	41	74
No. 3 10 KHz sans QVS	37	31	68
No. 4 10 KHz avec QVS	12	8	20

Influence du calcul en entier	SCORE		
	Homme	Femme	Total
No. 1 8 KHz calcul réel	67	60	127
No. 2 8 KHz calcul entier	21	39	60
No. 3 10 KHz calcul réel	35	30	65
No. 4 10 KHz calcul entier	21	15	36

Tableau 5.3 Résultats des tests subjectifs (tableaux numériques).

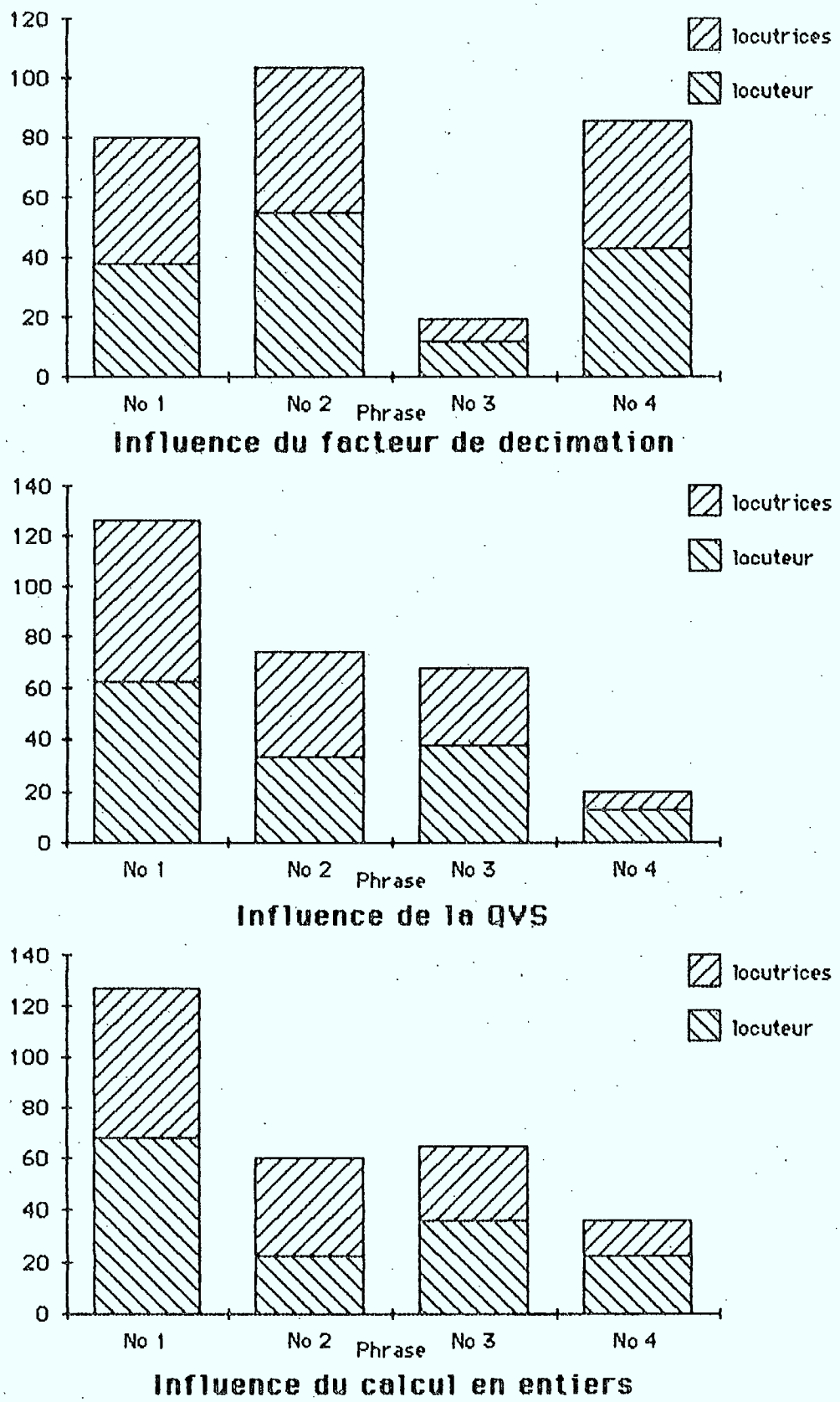


fig 5.2 : Resultats des tests subjectifs (graphiques)

5.4 Interprétation des résultats

La première constatation porte sur le changement de la fréquence d'échantillonnage. Il est clair, à la vue de ces tests, que les évaluateurs préfèrent les réalisations échantillonnées à 8 KHz. Il y a donc un problème de conception dans les codeurs à 10 KHz.

L'étude des préférences dans le cas d'un locuteur masculin, présente des tendances "normales". Bien que les phrases à 8 KHz emportent le plus de suffrages, la hiérarchie entre non codées et codées est sensiblement respectée (2ième et 3ième test), alors que pour les locutrices, toutes les phrases à 10 KHz, semblent rejetées. Il semble donc que la clé de la dégradation se trouve dans la restitution des plus hautes fréquences. Par ailleurs, l'étude des résultats concernant l'influence du facteur de décimation montre que 8 KHz décimé par 5 (bande équivalente de 800 Hz), et 10 KHz décimé par 4 (bande équivalente de 1250 Hz), sont également préférés (80/144 et 85/144 respectivement).

Ces constatations semblent mettre en cause le filtre prédicteur. Il faut en effet se rappeler [1 Annexe A] (reprise en Annexe B), que le filtre de pondération, issu du filtre prédicteur, doit privilégier certaines bandes de fréquence particulières afin d'éviter des problèmes de pollution spectrale.

L'enveloppe spectrale, représentée par le filtre LPC, semble donc trop grossièrement décrite à 10 KHz, par un filtre à l'ordre 12, ou bien

encore si 12 coefficients sont suffisants, leur quantification par un dictionnaire de 1024 formes n'est pas assez précise.

Ces conclusions peuvent être appuyées par une recherche plus systématique de l'ordre du filtre prédicteur optimal et de la taille du dictionnaire correspondant, afin de rétablir la hiérarchie de préférences qu'il devrait exister entre des phrases échantillonnées à 8 KHz et des phrases échantillonnées à 10 KHz. Des premières écoutes informelles d'une synthèse à 10 KHz, utilisant un filtre prédicteur d'ordre 14, viennent appuyer ces constatations.

CHAPITRE 6

CONCLUSIONS

A l'issue de cette étude, deux séries de résultats peuvent être dégagées. La première concerne l'amélioration de la qualité d'un vocodeur travaillant à 4800 bits/seconde et la seconde concerne la réalisation de vocodeurs et plus particulièrement le codage du signal résiduel.

6.1 Amélioration de la qualité du codage par augmentation de la bande passante

Le passage d'une fréquence d'échantillonnage de 8 KHz à une fréquence d'échantillonnage de 10 KHz a été étudié en vue d'une amélioration de la qualité de la parole décodée. Les résultats présentés dans le chapitre 5 montrent que cette augmentation de la bande passante combinée avec la méthode de codage du résidu utilisée est loin de donner la qualité escomptée. Au contraire, les auditeurs préfèrent généralement les codeurs à 8 KHz.

Il existe plusieurs origines possibles pour ce manque de qualité de la parole à 10 KHz. Tout d'abord, le filtre prédicteur et son quantificateur peuvent être mis en cause. Bien que le dictionnaire de filtres utilisé à 10 KHz (ordre 12), se révèle à la construction, d'une qualité équivalente à celui utilisé à 8 KHz (ordre 10), on peut se demander si la

modélisation du spectre qu'il réalise est suffisamment fine surtout dans le haut de la bande. En effet, il faut se souvenir que ce filtre n'est plus seulement utilisé pour produire le signal résiduel mais qu'il intervient aussi dans la pondération du bruit résultant de la méthode de décimation. Le signal étant d'énergie plus faible dans le haut de la bande, c'est dans cette région que cette pondération se doit d'être efficace.

L'utilisation de la quantification vectorielle sphérique, qui génère un signal relativement bruité, accentue cet impact du mauvais filtrage du bruit dans les hautes fréquences. Cette constatation est vérifiée par les tests subjectifs qui montrent que l'influence de la QVS est plus importante à 10 KHz qu'à 8 KHz.

Pour remédier à cette situation, on peut envisager d'adapter la quantification vectorielle du filtre au double rôle qui lui est attribué: prédicteur et pondérateur de bruit. Ceci pourrait, par exemple, se traduire par une prise en compte de ce facteur pondérateur dans le calcul de la distance utilisée pour la construction et l'utilisation du dictionnaire. Il serait aussi possible d'utiliser un dictionnaire spécifique de filtres pondérateurs. L'influence de l'ordre d'analyse (12 ou 14) est de la taille des fenêtres d'analyse seraient des facteurs à étudier.

6.2 Réalisation d'un vocodeur en temps réel

De sérieux progrès ont été réalisés en vue de la réalisation d'un vocodeur à quantification vectorielle. En plus du logiciel développé pour le TMS 320, des alternatives ont été explorées dans le but de réaliser

certaines fonctions du vocodeur à l'aide d'autres circuits. L'utilisation d'un microprocesseur à usage général tel que le 68008 pour le QVS, permet d'envisager la réalisation d'un vocodeur avec un seul TMS 320 associé avec ce microprocesseur.

La mise au point du système de développement autour du SKY 320 est une autre étape qui doit permettre la mise au point rapide et souple de vocodeurs basés sur le TMS 320. La réalisation de la carte dictionnaire, dont l'accès est compatible avec la mémoire de données du SKY, va permettre le passage rapide des vocodeurs du système de développement vers une maquette.

Dans un autre ordre d'idée, l'utilisation du circuit de synthèse SP 1000 pour le décodage du signal, permet non seulement de réduire la charge des autres circuits (TMS 320 et 68008) du vocodeur, mais aussi, rend possible la réalisation complète d'un décodeur à quantification vectorielle par dictionnaire autour d'un microprocesseur 8 bits (6803). Nous pensons que ce concept présente un intérêt considérable dans les applications où il n'y a qu'un transmetteur pour plusieurs récepteurs.

BIBLIOGRAPHIE

1 "Transmission et traitement de l'information parlée: vocodeur 2400 bits utilisant le codage par dictionnaire et la technique TDHS", Rapport final, contrat no. OSU-83-00134 pour le Ministère des Communications, Ottawa.

2 J.P. Adoul, F. Didelot, Ph. Mabilleau, S. Morissette, - "Generalization of the Multipulse Coding for Low Bit Rate coding Purposes: the Generalized Decimation", ICASSP'85, Tampa.

3 J.P. Adoul, C. Lamblin, A. Leguyader, - "Base Band Coding at 2400 bits/s Using Spherical Vector Quantization", ICASSP'84, pp. 1.12.1 @ 1.12.4.

4 C. Lamblin, - "Vocodeur à excitation vocale pour le codage de la parole à bas débit", Rapport de stage, ENST Bretagne, Juin 1983.

5 B. Atal, J. Remde, - "A New Model of LPC Excitation for Producing Natural Sounding Speech at Low Bit Rate", ICASSP'82, pp. 614-617.

6 S. Singhal, B. Atal, - "Improving Performance of Multipulse LPC Coders at Low Bit Rate", ICASSP'84, pp. 1.3.1 @ 1.3.4.

7 Berouti, Garten, Kabal, Mermelstein, - "Efficient Computation and Encoding of the Multipulse Excitation for LPC", ICASSP'84, pp. 10.1.1 @ 10.1.4.

8 Texas Instrument, - "TMS 32010 Analog Interface Board", Data Manual.

ANNEXE A

SPECIFICATIONS DU CIRCUIT DE SYNTHESE UTILISE

GENERAL INSTRUMENT	SP1000

ANNEXE A

PRELIMINARY INFORMATION

VOICE RECOGNITION/SYNTHESIS DEVICE

FEATURES

- N-channel silicon gate construction
- Single 5V supply
- On chip oscillator
- 0°C to +70°C operation
- User programable sample rate
- TTL compatible
- Standard microprocessor interface including 6809

RECOGNITION FEATURES

- Software controlled sampling frequency from 5.0kHz-15.9kHz
- Automatic gain control outputs for control of external input amplifier
- 8 stage Linear Predictive Coding (LPC) lattice analyzer

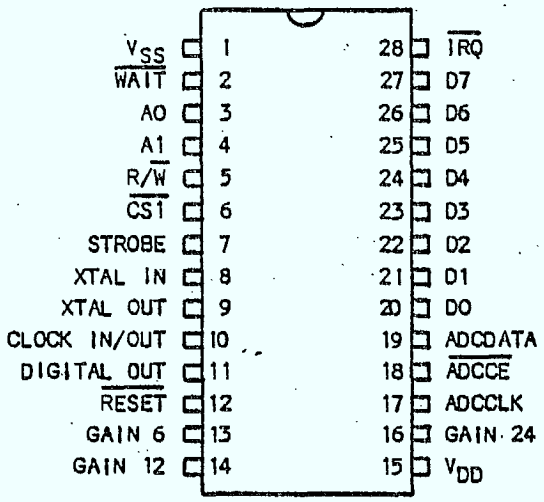
SYNTHESIZER FEATURES

- 10 stage lattice synthesizer
- Software controlled sampling frequency from 4.0kHz-12.7kHz
- Excitation look up ROM for voiced/unvoiced source
- 8 bit on chip digital to analog conversion with PWM digital output

PIN CONFIGURATION

28 LEAD DUAL IN LINE

Top View



DESCRIPTION

In voice recognition applications, the SP1000 forms the system front end by performing LPC feature extraction on an incoming audio signal. It must be supported by an 8 bit microcomputer which would compare the audio signal features with those of templates stored in memory, in order to make a recognition decision.

This results in a system design which leaves definition of key system features in the control of the system designer. Thus, this device can be used in speaker-dependent or speaker-independent systems, recognizing isolated words or connected speech.

The advantage of this approach is that the designer is not locked-in to a recognition algorithm which may or may not suit the design requirements. In addition, since the recognition algorithm is contained in software or firmware it can be easily upgraded to take advantage of advances in recognition techniques, without requiring hardware redesign.

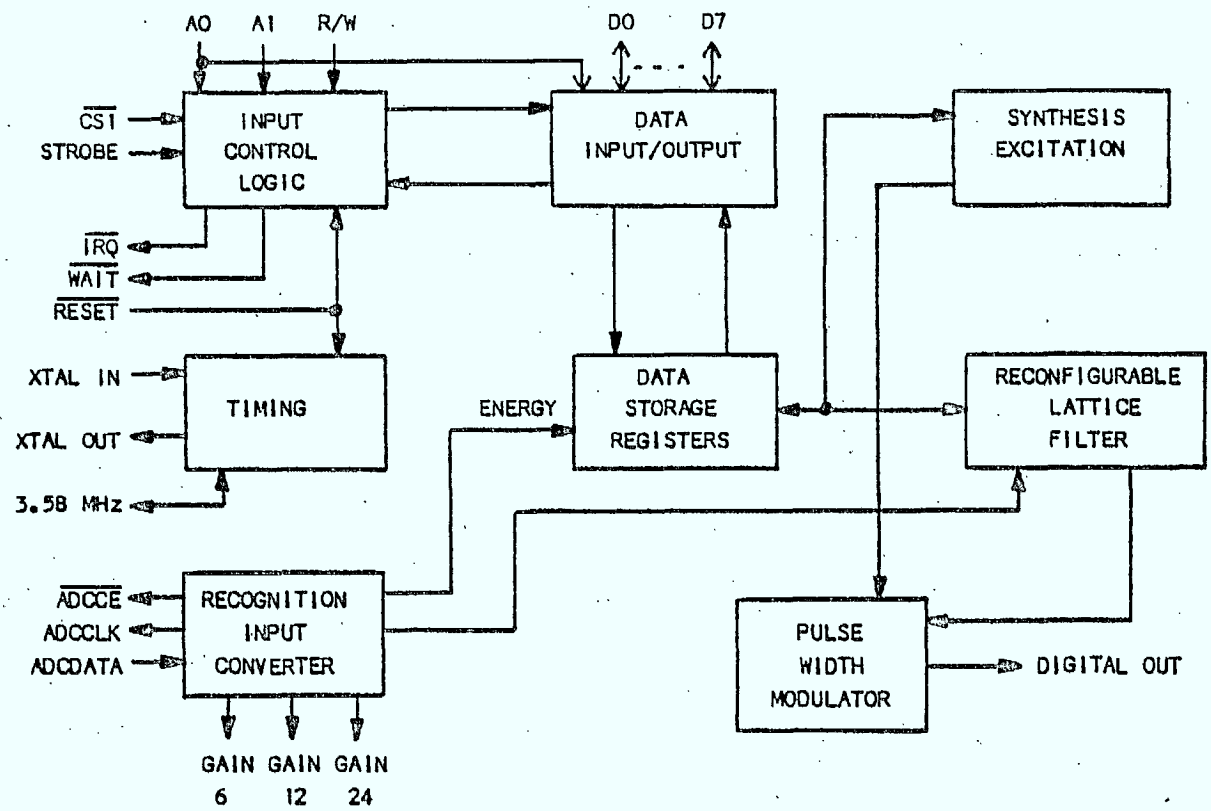
ARCHITECTURAL DESCRIPTION

Processor Interface:

The SP1000 Voice Recognition/Synthesis Device is designed to interface with a standard microprocessor bus, with data lines, address lines, chip select line, and read/write line. Eight bits of data may be read from, or written to the chip by

GENERAL INSTRUMENT	SP1000

SP1000 BLOCK DIAGRAM



VOICE RECOGNITION/SYNTHESIS SYSTEM BLOCK DIAGRAM

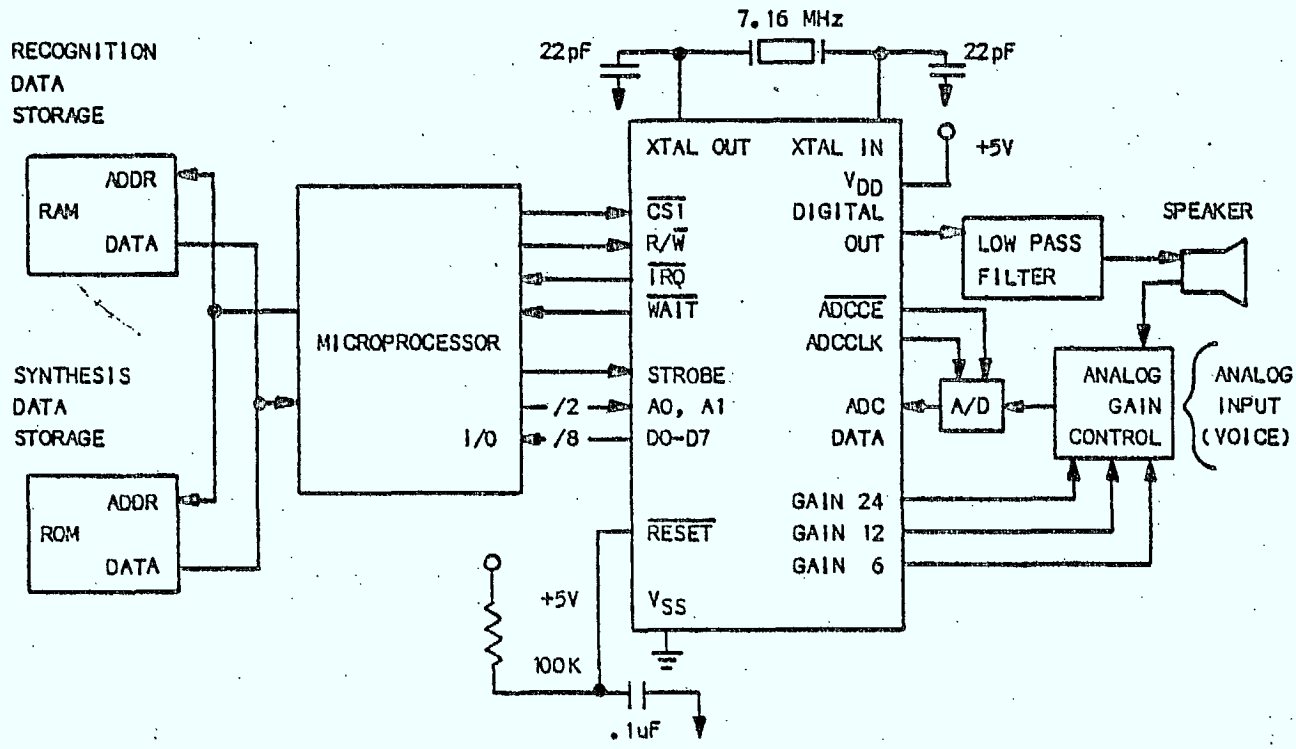


Table 1: OPERATION CODES

STROBE	$\overline{CS1}$	R/$\overline{W}$	A1	A0	Code	Operation
L	X	X	X	X		(Chip not selected)
X	H	X	X	X		
H	L	L	L	L	0	Write to Control Register
H	L	L	L	H	1	Write to Parameter Address Register
H	L	L	H	L	2	Write to Parameter Data Input Register, LSB = 0
H	L	L	H	H	3	Write to Parameter Data Input Register, LSB = 1
H	L	H	L	L	4	Read Status Register
H	L	H	L	H	5	(Not Implemented)
H	L	H	H	L	6	Read Parameter Data Output Register
H	L	H	H	H	7	Same as Code 6, plus initiate Internal Fetch

the processor, following standard peripheral protocol. A nine-bit word may be written to the chip during a coefficient write. Data is transferred via the data lines whenever the chip select line is active. The read/write line determines the direction of the transfer, and the address lines specify a particular register within the chip as source or destination. Another way of looking at the address lines is that they define, together with the read/write line (R/$\overline{W}$), a particular operation to be performed. Table 1 presents the available defined operations. The code column will be used for reference, e.g. the operation of writing to the Parameter Address Register will be referred to as a Code 1 operation.

Before describing the operations in detail, the data organization inside the device will be reviewed. There is a sharp distinction between control/status information and parameter data. Parameter data inside the chip resides in two large recirculating shift registers. The recirculation period is the same as the sample period. Access to a particular parameter can occur only once each recirculation period, when that parameter is passing by a fixed point of access. The access is bit serial, and the processor cannot access parameter data directly. One of the functional blocks inside the chip is a special interface to overcome this problem.

The interface bears some analogy to a serial communication channel for simplex mode operation. Parameter data written by the processor is first stored in the Parameter Data Input Register. A BUSY flag is set to indicate that the register is full. The contents of the Parameter Data Input Register will be written to the destination specified by the Parameter Address Register as soon as that destination becomes accessible. Upon completion, the BUSY flag is reset to signal the processor that the Parameter Data Input Register is empty and ready for another data transfer.

For the processor to read a parameter value, the interface must first perform a fetch operation. The fetch operation is initiated by the processor. The BUSY flag is set to indicate fetch in progress. As soon as the parameter (specified by the Parameter Address Register) becomes accessible, the value (truncated to eight bits if necessary) is transferred to the Parameter Data Output Register, and the BUSY flag is reset. The processor can now read the parameter value from the Parameter Data Output Register.

The reading and writing of parameter data requires a finite amount of time. During that time the interface cannot be disturbed. The processor, therefore, must monitor the state of the BUSY flag in all dealings with the device. The BUSY flag is reflected in the STATUS Register.

GENERAL INSTRUMENT	SP1000

PIN FUNCTIONS

Pin Number	Name	Function
1	V_{SS}	Ground
2	$\overline{WAIT}$	This output pin is normally high, but is taken low if the chip is selected for code 1, 2, 3, or 7 operation while the BUSY flag is set. It remains low until the BUSY flag is reset, or the chip is de-selected. This is an open drain output.
3	A_0	Least significant address line/ninth data line. (Least Significant Bit, input only.)
4	A_1	Most significant address line.
5	R/W	Read/Write. The state of input Pins 3, 4, and 5 determines the operation to be performed, according to Table 1.
6, 7	$\overline{CS1}$, STROBE	Chip select inputs: $\overline{CS1}$ active low, STROBE active high. Both must be active to select the chip.
8	XTAL IN	Crystal In. Input connection for a 7.16 MHz crystal.
9	XTAL OUT	Crystal Out. Output connection for a 7.16 MHz crystal.
10	CLOCK IN/OUT	3.58 MHz test Clock Input/buffered Clock Output. As an input, it allows the chip to be driven by an external clock (one-half the crystal frequency), in which case the crystal oscillator is disabled. As an output, it provides the buffered system clock which is one half of the crystal frequency.
11	DIGITAL OUT	Synthesized speech Digital Output. This is the output of the Pulse Width Modulator (PWM). The PWM, in conjunction with an external low pass filter, forms a digital to analog converter with eight bits of resolution.
12	$\overline{RESET}$	A logic 0 resets the chip. It must be returned to a logic 1 for normal operation.
13, 14, 16	GAIN 6, GAIN 12, GAIN 24	These 3 outputs can control 3 analog switches. The switches determine the gain of the external amplifier which precedes the analog to digital converter.
15	V_{DD}	Power supply pin.

PIN FUNCTIONS (continued)

Pin Number	Name	Function
17	ADCCLK	ADC Clock. This output is the 150 kHz gated clock. It is active only while $\overline{ADCCE}$ is active, otherwise it is held low.
18	$\overline{ADCCE}$	ADC Chip Enable. This output provides chip select for the analog to digital converter. It also controls the sample and hold circuit.
19	ADCDATA	ADC Data In. This input pin receives the serial data from the analog to digital converter.
20-27	D0-D7	Data lines. These 8 pins are the bi-directional data bus for connection to the processor. D7 carries the most significant bit.
28	$\overline{IRQ}$	Interrupt Request. This active-low output provides interrupt of the processor. This pin is an open drain output.

PARAMETER DESCRIPTION

Following is a short description of the parameters as listed in Table 2.

Synthesis

K_{10}-K_1:
These are the ten lattice filter coefficients for synthesis mode. Using the least significant address line as the ninth data line allows these to be specified with nine bits of resolution. K_{10} is the coefficient associated with the first filter section (closest to the excitation).

EXCTYP:
This eight-bit wide field is split in half to contain two parameters; Post Filter Gain and Excitation Type.

a) Post Filter Gain
The three least significant bits determine the post filter gain by determining which consecutive eight bits from the nominal sixteen-bit output of the filter will be sent to the digital to analog converter. A code of zero will choose the eight most significant bits (with proper sign extend) of the filter output. An increase of one will push the choice one bit to the right, such that a code of seven will choose the eight bits following the three most significant bits of the filter output.

Table 2: PARAMETER ADDRESS REGISTER

MSB 7	6	5	4	3	2	1	LSB 0
NOT USED		COEF BUF	NOT USED	INDEX			

Bits 0-3 Specify Parameter Index
Bit 5 Specifies Buffer (=1 for SR in recognition mode; otherwise = 0)

	Index	Parameter	
		Synthesis	Recognition
BIT 5 = 0	0	K_{10}	K_{R1}
	1	K_9	K_{R2}
	2	K_8	K_{R3}
	3	K_7	K_{R4}
	4	K_6	K_{R5}
	5	K_5	K_{R6}
	6	K_4	K_{R7}
	7	K_3	K_{R8}
	8	K_2	ENRGY
	9	K_1	T1
	A	EXCTYP	T2
	B	EXCAMP	SR
	C	T1	-
	D	T2	-
	E	SR	-
	F	-	-
BIT 5 = 1	B	-	SR

b) Excitation Type

The three least significant bits of the most significant half of the EXCTYP field specify one of several possible choices for synthesis excitation. The choices are:

0 Not Used
1 Constant Full Scale (used with external excitation)
2 Unvoiced
3 Voiced 1
4 Voiced 2
5 Voiced 3
6 Voiced 4
7 Voiced 5

	EXCITATION TYPE				POST FILTER GAIN		
EXCTYP		T2	T1	T0		G2	G1	G0

T2	T1	T0	EXCITATION USED	
0	0	0	0	Not Used
0	0	1	1	Constant Full Scale
0	1	0	2	Unvoiced
0	1	1	3	Voiced 1
1	0	0	4	2
1	0	1	5	3
1	1	0	6	4
1	1	1	7	5

G2	G1	G0	LATTICE DATA USED								
									MSB	MSB	MSB
0	0	0	0	S	S	S	S	MSB	-1	-2	-3
0	0	1	1	S	S	S	MSB	-1	-2	-3	-4
0	1	0	2	S	S	MSB	-1	-2	-3	-4	-5
0	1	1	3	S	MSB	-1	-2	-3	-4	-5	-6
1	0	0	4	MSB	-1	-2	-3	-4	-5	-6	-7
1	0	1	5	-1	-2	-3	-4	-5	-6	-7	-8
1	1	0	6	-2	-3	-4	-5	-6	-7	-8	-9
1	1	1	7	-3	-4	-5	-6	-7	-8	-9	-10

	SIGN	EXPONENT			MANTISSA			
EXAMP								

A/D GAINS

Connect the gain ouputs of the chip to the external amplifier preceding the A/D to realize the following table.

GAIN PINS			GAIN REQUIRED
24	12	6	
0	0	0	0 db
0	0	1	6
0	1	0	12
0	1	1	18
1	0	0	24
1	0	1	30
1	1	0	36
1	1	1	42

EXCAMP:
This field controls the amplitude of the selected excitation wave form. It is a gain factor specified as a base 2 logarithm, which is applied to each sample of the internally generated excitation. A special case exists when the excitation type is 1. The internally generated excitation is now a constant value. By writing a new value to the EXCAMP field every sample period, the constant value is modulated and externally generated excitation of the synthesizer is achieved. The gain factor is expressed in two parts: the most significant bit gives the sign of the gain factor (it is actually needed only for externally specified excitation); the seven least significant bits are the base 2 logarithm of its magnitude. These seven bits are interpreted as four exponent bits and three mantissa bits.

The reading of a particular, randomly selected parameter requires the following steps:

1. Load the Parameter Address Register by executing a write instruction, Operation Code 1. (See Table 1.)

2. Initiate a fetch operation (internal to the chip), by executing a read instruction with Operation Code 7. This will cause the BUSY flag (bit 7 of the STATUS Register) to become set. The fact that a read instruction is used will cause the current contents of the Parameter Data Output Register to be overwritten.

3. Wait for the BUSY flag to become reset, signaling the completion of the fetch sequence. The STATUS Register is read by a read instruction, Operation Code 4.

4. Read the desired parameter value by executing a read instruction, Operation Code 6.

Upon completion of a fetch (or store) operation, the Parameter Address Register is automatically updated to specify the next parameter in sequence. The sequence for synthesis is: K_{10}, $K_9 \ldots K_1$, EXCTYP, EXCAMP, T1, T2, SR. For recognition the sequence is K_{R1}, $K_{R2} \ldots K_{R8}$, ENRGY, T1, T2, SR. This automatic update simplifies sequential reading of parameters. Sequential reading of parameters requires the following steps:

1. Load the address of the first parameter to be read, as in step 1, above.

2. Initiate the first fetch operation, as in step 2, above.

3. Wait for the BUSY flag to become reset, as in step 3, above.

4. Execute a read instruction with Operation Code 7. This will read the parameter value and also initiate the fetch of the next parameter in the sequence.

5. Go to step 3.

Synthesis and Recognition

T1 and T2:
These two identical fields control the two programable timers. The timers are actually sample time counters, and the fields contain the value to be counted down for each.

SR (Sample Rate):
The low order six bits of this field indirectly determine the Sample Rate by specifying how many clock cycles of dead time should be inserted between the processing of each filter stage. The formula for Sample Rate is:

$$f_s = f_c/[(28+n) * m] \quad \text{where}$$

f_s is Sample Rate.
f_c is system clock rate (3.579545 MHz).
n is the integer represented by the six low order bits of the SR field.
m is the number of filter stages, 10 for synthesis and 8 for recognition.

The two most significant bits of the field are used for a different parameter pertaining to recognition only.

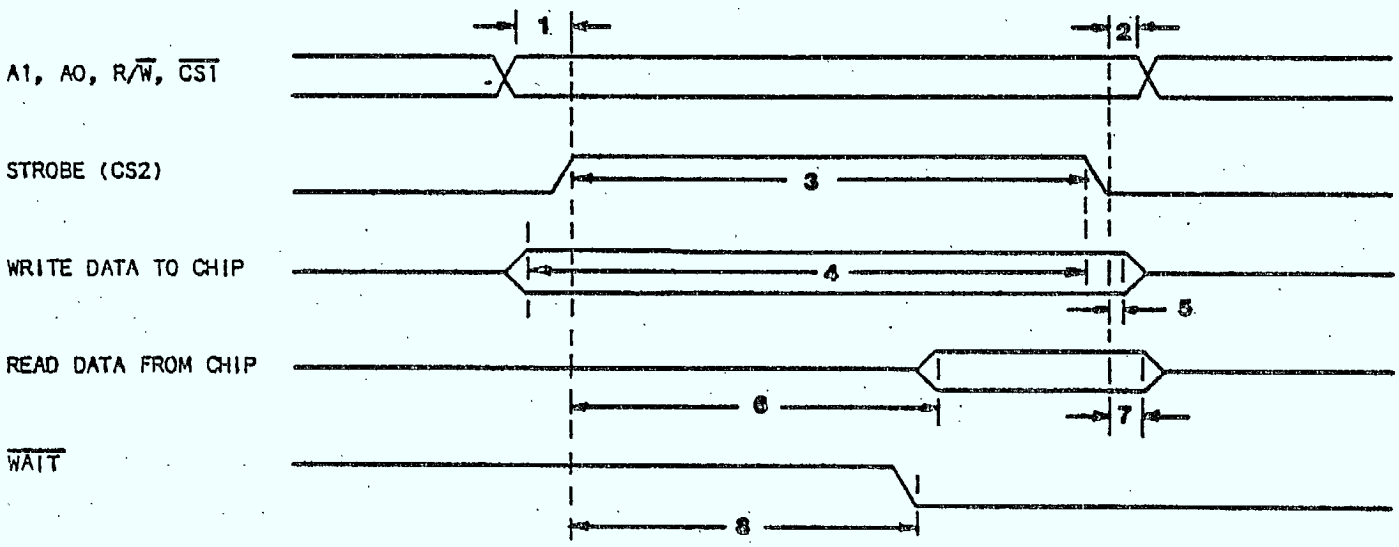
Recognition

K_{R1}-K_{R8}:
These are eight bit coefficients produced by the adaptive lattice filter in recognition mode. Two different representations of these are available to the processor. Averaged values are read from the first circular buffer (BUF=0 in the Parameter Address Register) and instantaneous values may be read from the other buffer (BUF=1).

ENRGY:
This eight bit field contains information generated by the chip. It is a measure of the magnitude of incoming speech samples averaged over time, and is expressed on a logarithmic scale.

SR (two high order bits only)
This 2-bit field specifies the set point for gain control.

PROCESSOR BUS TIMING



	MIN	MAX
1. Address Set Up Time	70 ns	-
2. Address Hold Time	20 ns	-
3. Strobe Pulse Width	210 ns	-
4. Write Data Set Up	280 ns	-
5. Write Data Hold	30 ns	-
6. Read Data Access		220 ns
7. Read Data Hold	10 ns	
8. WAIT Access		140 ns

GENERAL INSTRUMENT	SP1100

ELECTRICAL CHARACTERISTICS

Maximum Ratings*

V_{DD}..... -0.3V to +8V
Storage Temperature..... -25°C to +125°C

Clock

Crystal Frequency..... 7.15909 MHz
Standard Conditions (unless otherwise stated)
Clock In (Optional)..... 3.579545 MHz

DC CHARACTERISTICS

Operating Temperature T_A = 0°C to +70°C
Standard Conditions (unless otherwise stated)
V_{DD} = 5V ± 5%
V_{SS} = GND

*Exceeding these ratings could cause permanent damage to the device. This is a stress rating only and functional operation of this device at these conditions is not implied - operating ranges are specified in Standard Conditions. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Characteristics	Sym	Min	Max	Units	Conditions
DC Characteristics					
Supply Voltage	V_{DD}	4.5	5.5	V	
Supply Current	I_{DD}	-	100	mA	25°C, No output loads.
Inputs (ADCDATA, STROBE, $\overline{CS1}$, R/$\overline{W}$, A1, A0, $\overline{RESET}$)					
Input Logic 0	V_{IL}	0.0	0.8	V	
Input Logic 1	V_{IH}	2.0	V_{DD}	V	
Input Leakage	I_L	-	10	µA	V_{PIN} = 5.5V and 0.0V V_{DD} = 5.5V V_{SS} = 0.0V
Input/Outputs (D0-D7 CLOCK I/O)					
Input Logic 0	V_{IL}	0.0	0.8	V	
Input Logic 1	V_{IH}	2.0	V_{DD}	V	
Output Logic 0	V_{OL}	-	0.6	V	I_{OL} = 1.60 mA 50 pf (4 LS TTL Loads)
Output Logic 1	V_{OH}	2.4	-	V	I_{OH} = 100 uA
Output Leakage	I_L	-	10	µA	V_{PIN} = 5.5V and 0.0V V_{DD}, CS1 = 5.5V V_{SS} = 0.0V
Open Drain Outputs (IRQ, MRDY)					
Output Logic 0	V_{OL}	-	0.6	V	I_{OL} = 1.60mA 50 pf (4 LS TTL Loads)
Output Leakage	I_L	-	10	µA	V_{PIN} = 5.5V and 0.0V V_{DD} = 5.5V V_{SS} = 0.0V
Outputs (DIGITAL OUT, GAIN 6, 12, 24, ADCCLK, $\overline{ADCCE}$)					
Output Logic 0	V_{OL}	-	0.6	V	I_{OL} = 1.60mA 50 pf (4LS TTL Loads)
Output Logic 1	V_{OH}	2.4	-	V	I_{OH} = -100 µA

SP1000	GENERAL INSTRUMENT

REGISTERS

CONTROL REGISTER:

MSB 7	6	5	4	3	2	1	LSB 0
NU	SYN/ REC	ENB T1	ENB T2	ADD/ REPL	TIMER 1 TIMER 2	TEST 2	TEST 1

Bit 7 - NU — Not Used

6 - SYN/$\overline{\text{REC}}$ — 1 = Synthesis, 0 = Recognition

5 - ENB T1 — Set to One to Enable Timer 1 Interrupt to IRQ pin 28

4 - ENB T2 — Set to One to Enable Timer 2 Interrupt to IRQ pin 28

3 - ADD/$\overline{\text{REPL}}$ — 1 = Add to, 0 = Replace Parameter Value in First Buffer ((BUF) = 0)

2 - TIMER 1 — 1 = Timer 1, 0 = Timer 2, Causing Implementation of New Parameter Set

1 - TEST INTERNAL DATA — 1 = Test Mode. Internal data on Pin 11
0 = Normal Mode. D/A PWM on Pin 11

0 - TEST EXTERNAL CLOCK — 1 = Test Mode. Oscillator not used. Pin 10 is input. Use 3.58MHz
0 = Normal Mode. Pin 10 is output of 3.58MHz. lator is used.

STATUS REGISTER:

MSB 7	6	5	4	3	2	1	LSB 0
BUSY	S/R	T1	T2	COINC	NU	NU	NU

Bit 7 - BUSY — Processor Interface Busy

6 - S/$\overline{\text{R}}$ — High for Synthesis, Low for Recognition

5 - T1 OUT — Timer 1: Time-Out Flag (Latched)* (See note 1)

4 - T2 OUT — Timer 2: Time-Out Flag (Latched)* (See note 1)

3 - COINC — Current Slot Number Equals Parameter Address Register Slot Number

2 - NU — Not Used

1 - NU — Not Used

0 - NU — Not Used

Note 1:
*The timer flags are cleared by trailing edge of READ STATUS command.

PARAMETER ADDRESS REGISTER:

MSB 7	6	5	4	3	2	1	LSB 0
NOT USED		COEF BUF	NOT USED	INDEX			

Bits 0-3 — Specify Parameter

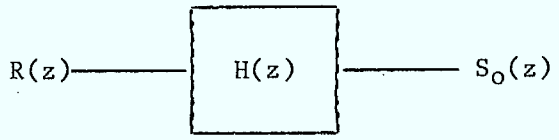
Bit 5 — Specifies Buffer

Bits 4, 6, 7 — Not Used

ANNEXE B

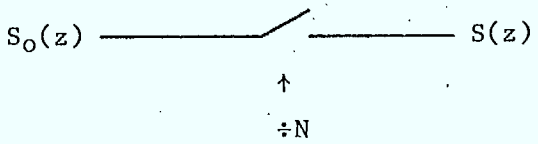
PROBLEMES DE REPLIEMENT INTRODUITS PAR LA DECIMATION GENERALISEE

On peut modéliser le filtre de pondération H par un filtre passe-bande, de support 2B centré sur $\pm\omega_o$.



$\rightarrow$ $S_o(z) = R(z)\,H(z)$ $\qquad$ $S_o(z)$ est à bande limitée 2B centré sur $\pm\omega_o$

la décimation par un facteur N donne S(z)



et on a:

$$S(z) = \frac{1}{N} \sum_{i=1}^{N} S_o(\lambda_i z)$$

où λ_i est la ième racine Nième de l'unité.

Dans le domaine des fréquences:

$$S(e^{j\omega}) = \frac{1}{N} \sum_{i=1}^{N} S_o\left(e^{j(\omega + \frac{2\pi i}{N})}\right)$$

En décomposant $S(e^{j\omega})$ suivant les fréquences négatives et positives: $S_o(e^{j\omega}) = S_1(e^{j\omega}) + S_1(e^{-j\omega})$ d'où:

$$S(e^{j\omega}) = \frac{1}{N} \sum \left(S_1\left(e^{j(\omega + \frac{2\pi i}{N})}\right) + S_1\left(e^{j(\frac{2\pi i}{N} - \omega)}\right)\right)$$

soit: $\omega_i \equiv \omega_o \left(\frac{2\pi}{N}\right)$, première fréquence image de la fréquence centrale ω_o.

Les conditions de non-recouvrement sont:

$$\omega_i + B \leqslant \frac{2\pi}{N} - \omega_j - B$$

$$\omega_i + B \leqslant \frac{2\pi}{N} + \omega_j - B$$

d'où

$$\boxed{\omega_i + B \leqslant \frac{\pi}{N}}$$

Ce qui signifie que la bande "utile" du signal dans une décimation généralisée ne doit pas excéder $2\pi/N$ et ne doit pas contenir de fréquences multiples de fe/2N. Cette condition est illustrée par la figure B.1. Dans le cas de la parole, la dernière condition est réalisée la plupart du temps pour N de l'ordre de 4 ou 5, le recalcul des amplitudes permettant quant à lui de minimiser les effets du recouvrement.

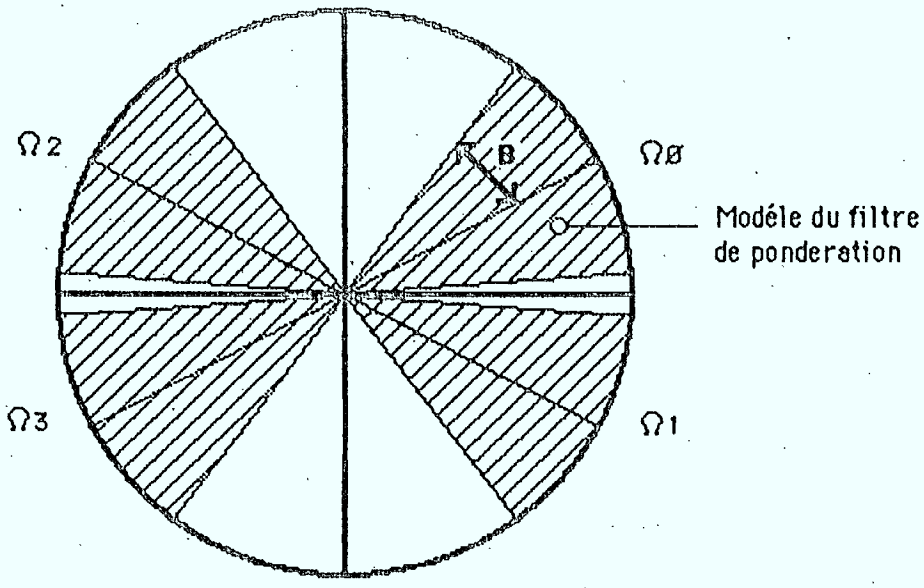


**Illustration de la modelisation de la ponderation
et des effets de la decimation
sans recouvrement spectral**

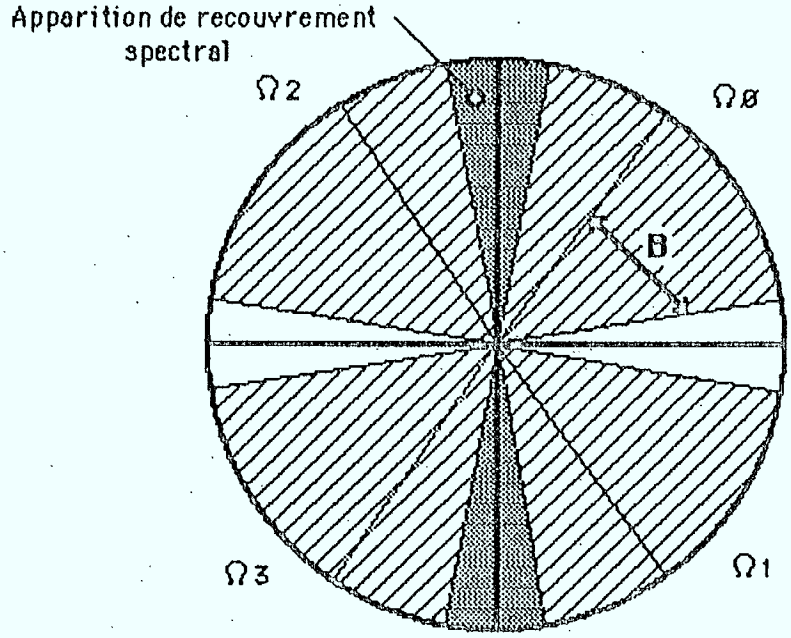


**Illustration de la modelisation de la ponderation
et des effets de la decimation
avec recouvrement spectral**

Fig. : B.1

ANNEXE C

DECIMATION GENERALISEE SUR LE TMS 320

Programme de décimation généralisée écrit pour le TMS 320. Ce programme reprend les simplifications exposées au chapitre 2. Les paramètres entrées sont le filtre LPC, le filtre de pondération, le signal; les paramètres en sortie sont la phase initiale et les amplitudes du résidu décimé. Les parties de code en ligne sont écrites pour une décimation par quatre.

```
*************************************************************
*
*       CRCS Universite de Sherbrooke
*       calcul du residu LPC et decimation generalisee
*
*************************************************************
*
*       constantes definissant le calcul
*
DECIM   EQU     4       facteur de decimation
LBLOC   EQU     39      longueur du bloc de recalcul
NIMP    EQU     9       nombre d'impulsions -1 NIMP+1=(LBLOC+1)/DECIM
MFILT   EQU     9       ordre du filtre de synthese - 1
MRIF    EQU     9       1/2 ordre du filtrage de ponderation - 1
MRIF1   EQU     MRIF+1
*
*       pointeurs de tableaux
*
LONTAB  EQU     LBLOC+MRIF1
MEMRC   EQU     LONTAB+MRIF1+2
MEMRC1  EQU     MEMRC+1
*
*       filtre RIF pas plus de 10 coefficients
*       c'est a dire MRIF1<=10
*       le programme de filtrage utilise l'adressage direct.
*
TP      EQU     LONTAB+1        tampon initial
TAMP1   EQU     TP+1
TAMP2   EQU     TP+2
TAMP3   EQU     TP+3    memoire du passe du filtre
TAMP4   EQU     TP+4    RIF effectuant la ponderation
TAMP5   EQU     TP+5
TAMP6   EQU     TP+6
TAMP7   EQU     TP+7
TAMP8   EQU     TP+8
TAMP9   EQU     TP+9
TAMP10  EQU     TP+10
*
A1      EQU     MEMRC+2*(MFILT+1)
A2      EQU     A1+1
A3      EQU     A1+2
A4      EQU     A1+3
A5      EQU     A1+4
A6      EQU     A1+5
A7      EQU     A1+6
A8      EQU     A1+7
A9      EQU     A1+8
A10     EQU     A1+9
*
*       pas plus de 10 valeurs apres decimation!
*
*       emplacement des valeurs decimees de FL0
*
DEC0    EQU     A10+1
DEC1    EQU     DEC0+1
```

```
DEC2    EQU     DEC0+2
DEC3    EQU     DEC0+3
DEC4    EQU     DEC0+4
DEC5    EQU     DEC0+5
DEC6    EQU     DEC0+6
DEC7    EQU     DEC0+7
DEC8    EQU     DEC0+8
DEC9    EQU     DEC0+9
*
*       emplacement des valeurs recalculees
*       (i.e. valeurs finales apres decimation generalisee)
*
CAL0    EQU     DEC9+1
CAL1    EQU     CAL0+1
CAL2    EQU     CAL0+2
CAL3    EQU     CAL0+3
CAL4    EQU     CAL0+4
CAL5    EQU     CAL0+5
CAL6    EQU     CAL0+6
CAL7    EQU     CAL0+7
CAL8    EQU     CAL0+8
CAL9    EQU     CAL0+9
*
*       page 1
*
SAR0    EQU     11      buffer pour AR0
SAR1    EQU     10      buffer pour AR1
UN      EQU      9      contient 1
PT      EQU      8      pointeur des echantillons en PRAM
MAX     EQU      7      max. en valeur abs. de FL0
PHI0    EQU      6      phase initiale
*
STOKRC  EQU   1024      debut des DATAs
*
**********************************************
*
        AORG     0
*
        B       DEBUT
*
DEBUT   LDPK    1       adressage direct en page 1
        LACK    1
        SACL    UN
        LT      UN
        MPYK    STOKRC  pointeur initial des tableaux
        PAC             pointe l'etat du filtre
        ADD     UN,0
*
        LARK    0,MFILT
        LARK    1,MEMRC1
*
MODELE  LARP    1       chargement du filtre inverse:
        TBLR    *+      coefficients de reflexion uniquement
        MAR     *+,0    les Ki et l'etat du filtre en echelle
        ADD     UN,1    sont imbriques => ki de 2 en 2
```

```
        BANZ    MODELE
*
        SUB     UN,0        ACC pointe maintenant sur la premiere
        LARP    1           valeur du tableau suivant en PRAM
        MAR     *-
        LARK    0,MRIF
        LARK    1,A1
*
CRIF    LARP    1           chargement coefficients du filtre RIF
        TBLR    *+,0
        ADD     UN,0
        BANZ    CRIF
*
        LARK    1,TAMP1
        LARK    0,MRIF
*
PRIF    LARP    1           chargement du passe du filtre RIF
        TBLR    *+,0
        ADD     UN,0
        BANZ    PRIF
*
        SACL    PT
        LACK    MRIF1       incremente le pointeur du
        ADD     PT,0        tableau en PRAM de MRIF1
        SACL    PT          (cf. debut de RES)
*
*       calcul du residu: 10860 micro-instructions
*                         2.18 ms
*
*
ITP2    EQU     15        double tampon pour
ITP1    EQU     14        calcul du residu(FIFO)
ISIH    EQU     13        accumulation 2xprecision
ISIL    EQU     12        du filtre en echelle
*
RES     LACK    MRIF1       debut du calcul du residu
        SUB     PT,0        pointeur en avance de MRIF1
        ABS
*
        LARP    0
        LARK    0,LONTAB    chargement des echantillons
LDECH   TBLR    *
        ADD     UN,0
        BANZ    LDECH
*
        SACL    PT          le pointeur est en avance de MRIF1 pour
*                           la prochaine iteration de RES
        LT      UN
        MPYK    STOKRC
        PAC
        LARK    0,MFILT
        LARK    1,MEMRC   chargement de l'etat du filtre:
EMBARK  LARP    1           etat uniquement (cf. + haut)
        TBLR    *+
        MAR     *+,0
```

```
        ADD     UN,1
        BANZ    EMBARK
*
        LARK    0,LONTAB position de l'echantillon en memoire
        LARK    1,LBLOC  nonbre d'iterations
*
FLTR    LARP    0         debut du processus de filtrage:
        ZALH    *,1        liere partie
        SACH    ITP1,0
        SACL    ISIL      initialisation de l'accumulation
        SACH    ISIH,0
        SAR     0,SAR0   sauvegarde des ARs du lier niveau
        SAR     1,SAR1
        LARK    0,MFILT nombre d'iterations (2ieme niveau d'imbrication)
        LARK    1,MEMRC position des parametres du filtre en memoire
*
TREILL  LARP    1
        LT      *+       maillon du treillis
        MPY     *
        PAC
        LT      ISIH
        ADDH    ISIH     accumulation sur 32 bits
        ADDS    ISIL
        SACH    ISIH
        SACL    ISIL
        MPY     *-
        PAC              Q14 dans ACC H
        ADD     *,14     Q16 dans le tableau d'etat
        DMOV    ITP1
        SACH    ITP1,1   Q15 dans ITP1
        LAC     ITP2,1
        SACL    *+       Q16 dans ACC L
        MAR     *+,0
        BANZ    TREILL
*
        LAR     0,SAR0
        LAR     1,SAR1
        LAC     ISIH,0
        SACL    *-,0,1
        BANZ    FLTR
*
        LT      UN
        MPYK    STOKRC
        PAC
        LARK    0,MEMRC
        LARK    1,MFILT
*
DEBARK  LARP    0        debarquement de l'etat du filtre:
        TBLW    *+        le prochain calcul du residu reprendra
        MAR     *+,1      a cet endroit
        ADD     UN,1
        BANZ    DEBARK
*
        LARP    0
        LARK    0,MRIF
```

```
*
FLTR1   ZALH    *,1     suite du filtrage:
        SACH    ITP1,0          le filtrage donnant la fonction
        SACL    ISIL            de localisation necessite la
        SACH    ISIH,0          connaissance de 10 echantillons
        SAR     0,SAR0          a venir(connaissnace-->N+10)
        LARK    0,MFILT
        LARK    1,MEMRC
*
TREIL1  LARP    1
        LT      *+
        MPY     *
        PAC
        LT      ISIH
        ADDH    ISIH
        ADDS    ISIL
        SACH    ISIH
        SACL    ISIL
        MPY     *-
        PAC
        ADD     *,14
        DMOV    ITP1
        SACH    ITP1,1
        LAC     ITP2,1
        SACL    *+
        MAR     *+,0
        BANZ    TREIL1
*
        LAR     0,SAR0
        LAC     ISIH,0
        SACL    *
        BANZ    FLTR1
*
*       calcul de la fonction de localisation FL0
*               filtrage RIF
*               coeff. de A1 a A10
*               passe TAMP1 a TAMP10 +TP
*               AR0 compteur de boucle
*               AR1 pointeur tableau
*               accumulation a partir des extremites
*
        LDPK    0       adressage direct des param. du filtre en page 0
        LARK    0,LBLOC
        LARK    1,LONTAB
*
RIF     LARP    1
        LAC     *,15
        SACH    TP,1
        LT      TAMP10
        SAR     0,TAMP10
        LAR     1,TAMP10
*
        MPY     A10
        LTA     *+
        MPY     A10
```

```
*
        LTD     TAMP9
        MPY     A9
        LTA     *+
        MPY     A9
*
        LTD     TAMP8
        MPY     A8
        LTA     *+
        MPY     A8
*
        LTD     TAMP7
        MPY     A7
        LTA     *+
        MPY     A7
*
        LTD     TAMP6
        MPY     A6
        LTA     *+
        MPY     A6
*
        LTD     TAMP5
        MPY     A5
        LTA     *+
        MPY     A5
*
        LTD     TAMP4
        MPY     A4
        LTA     *+
        MPY     A4
*
        LTD     TAMP3
        MPY     A3
        LTA     *+
        MPY     A3
*
        LTD     TAMP2
        MPY     A2
        LTA     *+
        MPY     A2
*
        LTD     TAMP1
        MPY     A1
        LTA     *+
        MPY     A1
*
        LTD     TP
        SACH    *-,1,0
        BANZ    RIF
*
        LDPK    1
        ZAC
        SUB     UN,0
        SACL    MAX
        LARK    0,LBLOC
```

```
        LARK    1,LONTAB
*
POSIT   LARP    1         recherche du maximum
        LAC     *-,0      en valeur absolue
        ABS
        SUB     MAX
        BLEZ    SUITE
        MAR     *+
        SAR     1,SAR1
        LAC     *-,0
        ABS
        SACL    MAX
SUITE   LARP    0
        BANZ    POSIT
*
*       calcul de la phase initiale PHI0
*
        LACK    LONTAB
        SUB     SAR1,0
        SACL    SAR1
*
PAS     EQU     ITP1      denominateur, contient DECIM
SAR11   EQU     ITP2      buffer pour AR1
*
        LACK    DECIM
        SACL    PAS
        LARK    0,15      16 iterations pour 1 division
        LAC     SAR1,0    SAR1 contient actuellement la position du max de FL0
DIV     SUBC    PAS
        BANZ    DIV
        SACH    PHI0,0    ACCH contient le reste de la division
*
*       sous echantillonnage de FL0
*
        LACK    LONTAB
        SUB     PHI0,0
        SACL    SAR1      pointeur des positions retenues dans FL0
        LACK    DEC0
        SACL    SAR11     pointeur du tableau DECi
        LARK    0,NIMP    AR0 compteur d'iteration
*
SSECH   LARP    1
        LAR     1,SAR1
        LAC     *,0       on pointe sur la valeur a extraire de FL0
        LAR     1,SAR11
        SACL    *+,0,0    on pointe sur la destination de cette valeur
        SAR     1,SAR11
        LAC     SAR1,0
        SUB     PAS       increment de SAR1 du pas de sous echantillonnage
        SACL    SAR1
        BANZ    SSECH
*
        LDPK    0
*
*       recalcul des amplitudes
```

```
*       tableau en adressage direct
*
        LAC     DEC0,15
        LT      A4
        MPY     DEC1
        SPAC
        LT      A8
        MPY     DEC2
        SPAC
        SACH    CAL0,1
*
        LAC     DEC1,15
        LT      A4
        MPY     DEC0
        SPAC
        MPY     DEC2
        SPAC
        LT      A8
        MPY     DEC3
        SPAC
        SACH    CAL1,1
*
        LAC     DEC2,15
        LT      A4
        MPY     DEC1
        SPAC
        MPY     DEC3
        SPAC
        LT      A8
        MPY     DEC0
        SPAC
        MPY     DEC4
        SPAC
        SACH    CAL2,1
*
        LAC     DEC3,15
        LT      A4
        MPY     DEC2
        SPAC
        MPY     DEC4
        SPAC
        LT      A8
        MPY     DEC1
        SPAC
        MPY     DEC5
        SPAC
        SACH    CAL3,1
*
        LAC     DEC4,15
        LT      A4
        MPY     DEC3
        SPAC
        MPY     DEC5
        SPAC
        LT      A8
```

```
        MPY     DEC2
        SPAC
        MPY     DEC6
        SPAC
        SACH    CAL4,1
*
        LAC     DEC5,15
        LT      A4
        MPY     DEC4
        SPAC
        MPY     DEC6
        SPAC
        LT      A8
        MPY     DEC3
        SPAC
        MPY     DEC7
        SPAC
        SACH    CAL5,1
*
        LAC     DEC6,15
        LT      A4
        MPY     DEC5
        SPAC
        MPY     DEC7
        SPAC
        LT      A8
        MPY     DEC4
        SPAC
        MPY     DEC8
        SPAC
        SACH    CAL6,1
*
        LAC     DEC7,15
        LT      A4
        MPY     DEC6
        SPAC
        MPY     DEC8
        SPAC
        LT      A8
        MPY     DEC5
        SPAC
        MPY     DEC9
        SPAC
        SACH    CAL7,1
*
        LAC     DEC8,15
        LT      A4
        MPY     DEC7
        SPAC
        MPY     DEC9
        SPAC
        LT      A8
        MPY     DEC6
        SPAC
        SACH    CAL8,1
```

```
*
        LAC     DEC9,15
        LT      A4
        MPY     DEC8
        SPAC
        LT      A8
        MPY     DEC7
        SPAC
        SACH    CAL9,1
*
*       soustraction des impulsions obtenues
*       au passe du filtre RIF
*
        LDPK    1
        LACK    TP
        ADD     PAS
        SUB     PHI0,0
        SACL    SAR1            AR1 pointe sur les
        LACK    CAL9            valeurs a modifier dans TAMPn
        SACL    SAR0            AR0 pointe dans tab.des impulsions
*
        LARP    1
ACTAMP  LAR     1,SAR1
        LAC     *,0,0           passe dans ACC
        SUB     *-,0,1          -impulsion correspondante
        SACL    *,0,0           resultat dans passe
        LAC     SAR1,0
        ADD     PAS
        SACL    SAR1
        LACK    TAMP10
        SUB     SAR1
        BGEZ    ACTAMP
*
*
        NOP
        B       RES             fin provisoire
*
        AORG    STOKRC
```

ANNEXE D

QUANTIFICATION VECTORIELLE SPHERIQUE SUR LE TMS 320

Programme de QVS dans E8 fonctionnant sur TMS 320. Transcription de la méthode de projection Adoul/Lamblin et de numérotation par racine.

```
>
*************************************************************************
*       GOSSET.E8:
*       quantification spherique par lattice de Gosset E8
*       tableau des leaders implante en PROM lectures par TBLR
*       La version finale les lira sur une carte dictionnaire
*
*************************************************************************
*
*       pointeurs de tableaux
*       tableau intermediaire contenant
*       valeurs decimee de FL0 (RESDOC) puis valeurs ordonnees
*
DEC0    EQU     16
*
*       emplacement des valeurs recalculees
*       (i.e. valeurs finales apres decimation generalisee)
*       = valeurs a quantifier= valeurs quantifiees apres execution
*
CAL0    EQU     26
*
TAB     EQU     128     table des elts et de leur poids (page 1)
*
*       variables de fonctionnement
*
SAR0    EQU     0       buffer pour AR0
SAR1    EQU     1       buffer pour AR1
UN      EQU     2       contient 1
*
PTLEAD  EQU     4       pointeur dans les leaders
ICLAS   EQU     5       numero de la classe du vect quantifie
INDFIN  EQU     6       valeur finale de l'indice apres quantification
*!!!!!! la variable suivante peut etre a l'origine de sacs de noeuds !!!!!!!!*
MAX     EQU     7       buffer pour maximum et variable temporaire
*
TMP0    EQU     8       variables temporaires
TMP1    EQU     9
*
*       variables de controle
*
PTLD0   EQU     10      debut tableau leader
PTRAC0  EQU     11      fin tableau de racines
PTBIAI  EQU     12      fin tableau des biais
NBLEAD  EQU     13      nombre de leaders
NIMP    EQU     14      dimension des vecteurs
*^^
DBUG    EQU     15
*********************************************
*
        AORG     0
        B       DEBUT
        RET
*
BLOCD   BSS     5
*
```

```
********
*
DEBUT   LDPK    0
        LACK    1
        SACL    UN      UN = 1
*
        LARK    0,4     5 constantes
        LARK    1,PTLD0 en DRAM a partir de PTLD0
        LACK    BLOCD   en PROM a partir de BLOCD
LOD     LARP    1
        TBLR    *+,0    chargement du bloc de controle
        ADD     UN,0
        BANZ    LOD
*
********
*
*       mise en ordre de CAL dans DEC
*
PTMAX   EQU     TMP0
*
        LAR     0,NIMP  AR0 pointe le nb d'iterations
        LARK    1,DEC0  AR1 pointe dans le tableau des valeurs classees
        LAC     UN,15
        SACL    MAX     MAX=-32000
*
ORDRE   LARP    1       "DO I=1,NIMP"
        SAR     0,SAR0  imbrication d'une deuxieme boucle
        SAR     1,SAR1
        LAR     0,NIMP  AR0 --> nb d'iterations
        LARK    1,CAL0  AR1 -->tableau des valeurs a classer
*
MAXI    LARP    1       "DO J=1,NIMP"
        LAC     *+      recherche du maximum dans CAL
        SUB     MAX,0
        BLEZ    PASMAX  comparaison du maximum courant et de CAL courant
        MAR     *-
        SAR     1,PTMAX PTMAX pointe sur le max courant de CAL
        LAC     *+      actualisation de MAX courant
        SACL    MAX
PASMAX  LARP    0
        BANZ    MAXI
*
        LAR     0,PTMAX PTMAX pointe sur le max absolue a cet ordre
        LAR     1,SAR1
        LAC     *,0,1
        SACL    *+,0,0  max (CAL(J)) = DEC(I)
        LAC     UN,15
        SACL    MAX     MAX=-32000
        ADD     SAR1,0
        SACL    *       max(CAL(J)) pointe sur DEC (I) a -32000 pres
        LAR     0,SAR0
        BANZ    ORDRE
*
********
*
```

```
*       projection sur les leaders
*
LEAD    EQU     TMP0    point de passage des elements des leaders
ACC     EQU     TMP1    distance accumulee entre vecteur ordonne et leader
*
        LAC     PTLD0,0
        SACL    PTLEAD
        LAC     UN,15
        SACL    MAX     initialisation de MAX a -32000
*
        LAR     0,NBLEAD
*
PRO0    SAR     0,SAR0  "DO I=1,NBLEAD"
        LAR     0,NIMP
        LARK    1,DEC0
        ZAC
*
PRO1    LARP    1       "DO J=1,NBIMP" toute cette partie
        SACL    ACC     depend du hardware utilise pour
        LAC     PTLEAD,0
        TBLR    LEAD    stocker les leaders.Ici ils
        ADD     UN,0    sont stockes en PROM.Quand ils seront
        SACL    PTLEAD  sur carte dictionnaire un IN suffira
        LAC     ACC
*
        LT      LEAD    calcul de la projection du leader sur
        MPY     *+,0    le vecteur a coder
        APAC
        BANZ    PRO1
*
        SACL    ACC
        SUB     MAX     test si projection maximum
        BLEZ    PRO2
*
        LAC     ACC,0
        SACL    MAX     si oui MAX = projection
        LAC     SAR0,0
        SACL    ICLAS   indice par rapport a la fin
*
PRO2    LAR     0,SAR0
        BANZ    PRO0
*
        CALL    LEADER  lecture du leader correspondant a ICLAS
*
********
*
*       deballage du vecteur ordonne
*       var locale TMP0
*
        LAR     0,NIMP  AR0 -->nb d'iterations
        LARK    1,CAL0  AR1 -->pointeur dans CAL
*
DEBAL   LARP    1       "DO I=1,NIMP"
        SAR     0,SAR0
        LAC     *,0,0
```

```
        SUB     UN,15   on enleve le biais
        SACL    TMP0
        LAR     0,TMP0  AR0 pointe maintenant dans DEC
        LAC     *,0,1
        SACL    *+,0,0
        LAR     0,SAR0
        BANZ    DEBAL
*
********
*
*       numerotage par algorithme de SCHALKWICK
*       pour doc voir rapport LAMBLIN
*       principe de base:
*       on compte les elements qui precede dans l'ordre lexicographique
*
****
RACIND  EQU     TMP0
INDICE  EQU     TMP1
*
*       liere partie exploration du chef d'orbite
*       afin de determiner
*               a) les elements constitutifs
*               b) leur poids respectif
*
        CALL    EXPLOR
*
****
*
        ZAC
        SACL    INDICE  mise a zero de l'indice dans la classe
        LAC     PTRAC0   table rangee dans l'ordre inverse des classes
        ADD     ICLAS   ICLAS = indice par rapport a la fin
        TBLR    RACIND  lecture de la racine de l'indice
*
*       RACIND = (8*)7!/{*[POIDS(j)]!} #la *[.] porte sur le nb d'elts diff.#
*
        LAR     0,NIMP
        LARK    1,CAL0  pointe sur le vecteur quantifie
*
NUM     SAR     0,SAR0
*
        LAC     SAR0,0
        ADD     UN,0
        SACL    SAR1,0
        LT      SAR1
        MPY     INDICE
        PAC
        SACL    INDICE
*
        LARK    0,TAB
        LT      RACIND  RACIND premultiplie par 8 pour etre entier!
*
ACCUM   LAC     *+,0,1  ACC= jeme element constitutif
        SUB     *,0,0   ACC - ieme element
        BZ      ACTU    teste si element inferieur (lexico.)
```

```
*
        ZALS    INDICE  INDICE > 0 pouvant sur sph6 atteindre dyn. de 16 bits
        MPY     *+
        APAC
        SACL    INDICE
        B       ACCUM
*
ACTU    MPY     *
        PAC
        SACL    RACIND   RACIND premultiplie par 8 pour etre entier
        LAC     *
        SUB     UN,0
        SACL    *,0,1
        MAR     *+,0
*
        LAC     RACIND
        LARK    0,15
DIV0    SUBC    SAR1
        BANZ    DIV0
        SACL    RACIND  resultat de la division.
*
        ZALS    INDICE
        LARK    0,15
DIV01   SUBC    SAR1
        BANZ    DIV01
        SACL    INDICE,0
*
        LAR     0,SAR0
        BANZ    NUM
*
        LAC     PTBIAI,0 table rangee dans l'ordre inverse des classes
        ADD     ICLAS,0 ICLAS = indice par rapport a la fin
        TBLR    INDFIN  indice final = BIAIS
        LAC     INDICE,0
        ADD     INDFIN,0 addition du biais
        SACL    INDFIN,0
*
********
*
*       decodage indice --> vecteur
*
*****
*
*       classe
*
PTMP    EQU     TMP0
BIAIS   EQU     TMP1
IXIND   EQU     MAX
*
        LARK    0,0
        LAC     PTBIAI,0
        SACL    PTMP
*
QUCLAS  TBLR    BIAIS   lecture du biais dans la table
        ADD     UN,0
```

```
        SACL    PTMP
        LAC     INDFIN,0
        SUB     BIAIS   lier indice relatif >0 donne la classe
        BGEZ    CLASS
        MAR     *+
        LAC     PTMP
        B       QUCLAS
*
CLASS   SACL    INDFIN  INDFIN contient maintenant indice dansla classe
        SAR     0,ICLAS ICLAS numero de la classe par rapport a la fin
********
        CALL    LEADER  va lire le chef d'orbite
        CALL    EXPLOR  l'explore (elts et leur poids)
********
BUFFAC  EQU     RACIND
*
        ZAC
        SACL    INDICE  mise a zero de l'indice reconstitue
        LAC     PTRAC0   table rangee dans l'ordre inverse des classes
        ADD     ICLAS   ICLAS = indice par rapport a la fin
        TBLR    RACIND  lecture de la racine de l'indice
*
*       RACIND = (8*)7!/{*[POIDS(j)]!} #la *[.] porte sur le nb d'elts diff.#
*
        LAR     0,NIMP
        LARK    1,CAL0  pointe sur le vecteur quantifie
*
DENUM   SAR     0,SAR0
        LARK    0,TAB
*
        LAC     SAR0,0
        ADD     UN,0
        SACL    SAR1,0
        LT      SAR1
        MPY     INDFIN
        SACL    IXIND,0
        MPY     INDICE
        PAC     SACL,0
        SACL    INDICE,0
        LT      RACIND
*
RECONS  MAR     *+
        MPY     *+
        PAC
        ADDS    INDICE
        SACL    BUFFAC
        SUBS    IXIND    Contient indice relatif
        BGZ     ACTU1
*
        DMOV    BUFFAC  la case INDICE suit immediatement BUFFAC
        B       RECONS
*
ACTU1   MAR     *-
        MAR     *-      aller pointer sur element trouve
        LAC     *+,0,1
```

```
        SACL    *+,0,0  ecriture dans vecteur decode
*
        MPY     *
        PAC
        SACL    RACIND  premultiplication
        LAC     *
        SUB     UN,0
        SACL    *       modification de poids
*
        LAC     RACIND
        LARK    0,15
DIV1    SUBC    SAR1
        BANZ    DIV0
        SACL    RACIND  resultat de la division.
*
        ZALS    INDICE
        LARK    0,15
DIV11   SUBC    SAR1
        BANZ    DIV01
        SACL    INDICE
*
        LAR     0,SAR0
        BANZ    DENUM
*
********
        NOP
        NOP
        B       DEBUT
*
********
*       ROUTINE
*       lecture du leader de la classe du vecteur code
*
LEADER  LAR     0,NIMP
        LARK    1,DEC0  ecriture en DEC0
        LAC     NBLEAD,3 *8 car vecteur de dimension 8
        ADD     PTLD0
        SUB     ICLAS,3  ACC pointe maintenant sur debut du leader
*
PRO3    LARP    1       "DO I=1,NIMP"
        TBLR    *+,0
        ADD     UN,0
        BANZ    PRO3
*
        RET
*
********
*       ROUTINE
*       exploration du chef d'orbite en DEC0
*       afin de determiner
*               a) les elements constitutifs
*               b) leur poids respectif
*
EXPLOR  LAR     0,NIMP
        LARK    1,TAB   pointe sur le tableau des resultats
```

```
        MAR     *-,1    7 iterations
        LACK    DEC0
        SACL    SAR1
        LAC     DEC0    1er element du chef d'orbite
        SACL    *+      on est en ARP = 1
        LAC     UN,0
        SACL    *,0,0   offset de 12 bits de l'unite:
*                       pour le cadrage.On peut ainsi par l'instruction
*                       SACH XX,Y (Y=1 ou 4) effectuer une post-multiplication
*                       par 1 ou 8
*
EXPLO   SAR     0,SAR0
        LAR     0,SAR1  AR0 leader,AR1 tableau
        LAC     *+      comparaison de 2 elements consecutifs
        SUB     *,0     du leader
        BNZ     ELSE    branche si different
*
        LARP    1
        LAC     UN,0
        ADD     *,0     sinon incremente le poids d'une unite
        SACL    *,0,0
        B       ENDIF
*
ELSE    LAC     *,0,1   incremente pointeur dans TAB
        MAR     *+
        SACL    *+      ecriture element constitutif dans TAB
        LAC     UN,0
        SACL    *,0,0   de poids 1 pour l'instant
*
ENDIF   SAR     0,SAR1
        LAR     0,SAR0
        BANZ    EXPLO
*
        RET
*
********
*
KLEAD0  EQU     64
KLEAD1  EQU     8
KLEAD2  EQU     7
KLD0    BSS     KLEAD0
KRAC    BSS     KLEAD1
KBIAI   BSS     KLEAD1
*
        AORG    BLOCD
        DATA    KLD0,KRAC,KBIAI,KLEAD2,7
*
        AORG    KLD0
*
*       leader numero 1
*
    DATA    2
    DATA    2
    DATA    0
    DATA    0
```

```
        DATA      0
        DATA      0
        DATA      0
        DATA      0
*
*          leader numero  2
*
        DATA      2
        DATA      0
        DATA      0
        DATA      0
        DATA      0
        DATA      0
        DATA      0
        DATA     -2
*
*          leader numero  3
*
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA      1
*
*          leader numero  4
*
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA     -1
        DATA     -1
*
*          leader numero  5
*
        DATA      1
        DATA      1
        DATA      1
        DATA      1
        DATA     -1
        DATA     -1
        DATA     -1
        DATA     -1
*
*          leader numero  6
*
        DATA      1
        DATA      1
        DATA     -1
        DATA     -1
```

```
      DATA      -1
      DATA      -1
      DATA      -1
      DATA      -1
*
*        leader numero  7
*
      DATA       0
      DATA       0
      DATA       0
      DATA       0
      DATA       0
      DATA       0
      DATA      -2
      DATA      -2
*
*        leader numero  8
*
      DATA      -1
      DATA      -1
      DATA      -1
      DATA      -1
      DATA      -1
      DATA      -1
      DATA      -1
      DATA      -1
*
*       table des racines RACIND
*
      DATA       1
      DATA      28
      DATA      28
      DATA      70
      DATA      28
      DATA       1
      DATA      56
      DATA      28
*
*       table des  biais BIAIS
*
      DATA     239
      DATA     211
      DATA     183
      DATA     113
      DATA      85
      DATA      84
      DATA      28
      DATA       0
*
******
        END
<
```

ANNEXE E

QUANTIFICATION VECTORIELLE SPHERIQUE SUR LE 68000

Ensemble de modules écrit en code 68000 effectuant la QVS dans E8. Transcription de la méthode Adoul/Lamblin (PROJ.SUB) et de numérotation par racine (NUM et DENUM).

```
*       PROJ.S B:

*         antification spherique par projection sur les leaders
*       arguments:
*               TABDC(LONG) vecteur a quantifier (en entree)
*                           vecteur quantifie (en sortie)
*               ICLAS       classe du vecteur de sortie

*
*       mise en ordre
*               A0      (TABO)
*               A1      (TABDC)
*               A2      (MAX)
*               A3      sauve (TABDC(1))
*               D0      valeur intermediaire
*               D1      indice ordre
*               D2      indice recherche maximum
*               D3      valeur du maximum
*               D4      sauvegarde de LONG-1
*               D7      constante $8000

        LEA     TABO,A0         pointeur de TABO tableau ordonne
        LEA     TABDC,A3        -------- -- TABDC ----- "debale"et classement
        MOVE.W  #$8000,D7       constante tres negative !
        MOVE.W  #(LONG-1),D4    D4 = LONG-1
        MOVE.W  D4,D1           longueur des tableaux

DBORDRE MOVE.W  D4,D2           reinitialise les registres de la boucle
        LEA     (A3),A1
        MOVE.W  D7,D3       initialisation du maximum courant

ORDOO   CMP.W   (A1)+,D3
        BGT.S   DOJ             teste par rapport au maximum courant

        LEA     -2(A1),A2       sauve le pointeur du maximum courant dans A3
        MOVE.W  (A2),D3         actualise le maximum courant

DOJ     DBF     D2,ORDOO        fin de DO D2=...

        MOVE.W  D3,(A0)+        place max courant dans TABO
        MOVE    D1,D0
        ADD.W   D7,D0
        MOVE.W  D0,(A2)         remplace max courant par ordre - 32000

        DBF     D1,DBORDRE      fin de DO D1=...

*       projection sur les leaders de E8
*               A0      (TABO)
*               A1      (leaders)
*               A3      (TABO(1))
*               A4      pointeur en memoire du vecteur retenu
*               D0      indice des classes
*               D1      indice de la classe de la projection maximum
*               D2      accumulation de la projection
*               D3      projection maximum
```

```
*               D4      sauvegarde de LONG-1 (deja initialise)
*               D5      indice des elements dans le vecteur
*               D6      valeur intermediaire de l'accumulation D2
*               D7      constante $8000 (deja initialisee)

        MOVE.W  #KLEAD,D0       LONG est encore dans D4
        LEA     TABO,A3
        LEA     KLD0,A1
        MOVE.W  D7,D3
        SWAP    D3              initialisation du maximum a tres negatif
        BRA.S   INPROJ

PROJEC  MOVE.L  #0,D2           mise a 0 de l'accumulation de la projction
        LEA     (A3),A0         recharge les registres de la boucle "vecteur"
        MOVE.W  D4,D5           id.

BOUVECT MOVE.W  (A1)+,D6        D6 = LEADER(i,j)
        EXT.W   D6              leader = octet inferieur, superieur reserve
        MULS    (A0)+,D6        D6 = TABO(i)*LEADER(i,j)
        ADD.L   D6,D2           D2 = somme sur i [TABO(i)*LEADER(i,j)]
        DBF     D5,BOUVECT

        CMP.L   D2,D3           CC modifie par D3-D2
        BGT.S   INPROJ          teste si D2 > D3 sinon branche

        MOVE.L  D2,D3           mise a jour du max courant
        MOVE.W  D0,D1           trace de la classe correspondante
        LEA     (A1),A4         trace du pointeur de la classe

INPROJ  DBF     D0,PROJEC

        ASL.W   #1,D1           *2 parce que doit servir a pointer des mots
        MOVE.W  D1,ICLAS        garde trace dans ICLAS

*       restitution de l'ordre original
*               A1      (TABDC)
*               A4      pointeur du leader ordonne(deja initialise)
*               D1      indice des vecteurs
*               D0      valeur intermediaire
*               D4      sauvegarde de LONG-1(deja initialise)
*               D7      constante $8000 (deja initialisee)

        LEA     TABDC,A1        --------- -- TABDC ----- "debale"et classement
        MOVE.W  D4,D1           longueur d

DEBAL   MOVE.W  (A1),D0         lecture du pointeur dans TABDC
        SUB.W   D7,D0           suppression du biais
        NEG.W   D0              le biais donne ordre / fin
        ASL.W   #1,D0           multipication par 2 du biais
        MOVE.W  -2(A4,D0),D0
        EXT.W   D0
        MOVE.W  D0,(A1)+        transfert du leader vers TABDC

        DBF     D1,DEBAL
```

```
*****************************************************************************
*       NUM.SUB : numerotation
*       arguments en entree :
*               TABDC   vecteur quantifie a numeroter
*               ICLAS   sa classe d'equivalence (orbite)
*                 en sortie :
*               INDICE  indice absolu sur la sphere

*       exploration du leader retenu :
*               -elements constitutifs
*               -poids

*               A0      pointe sur le leader
* routine (R)   A1      pointe sur le leader avec une avance d'un mot sur A0
*               A2      pointe sur le tableau des poids
*               A3      pointe sur le tableau des elements constitutifs
*               A4      trace du leader retenu(pointe sur le debut du suivant)
*               D0      nombre d'iterations - 1
* R             D1      accumulation dupoids d'un element

        LEA     -(LONG*2)(A4),A0        positionne A0 sur debut du leader
        LEA     POIDS,A2
        LEA     ELTS,A3
        MOVE.L  #(LONG-1),D0    initialisation du nombre d'iterations

        BSR     EXPLOR

*       numerotation

*               A0      vecteur quantifie TABDC
*               A1      elements constitutifs ELTS
*               A2      poids correspondants POIDS
*               A5      trace classe sous forme de pointeur relatif
*               D0      indice de la boucle
*               D1      accumulation du numero du vecteur dans sa classe
*               D2      racine de numerotation
*               D3      buffer de l'element (A0)+
*               D5      valeur intermediaire
*               D6      largeur du domaine des permutations possibles (D0+1)

        MOVE.L  #(LONG-1),D0    initialisation de la longueur de la boucle
        MOVE.W  D0,D6
        ADD.W   #1,D6           initialisation de la largeur du domaine ...
        MOVE.L  #0,D1           initialise a 0 la valeur accumulee de l'indice
        MOVE.W  ICLAS,A5        classe sous forme de pointeur relatif
        MOVE.W  KRAC(A5),D2     D2 = RACIND
        LEA     TABDC,A0        initialisation definitive de A0 sur TABDC

NUMERO  LEA     ELTS,A1
        LEA     POIDS,A2

        MULU    D6,D1           mise a l'echelle de D1 pour resul. entiers
        MOVE.W  (A0)+,D3        D3 = element de rang D6 de TABDC

ACCUM   CMP.W   (A1)+,D3
```

```
        BEQ.S   ACTU            accumule tant que elment du vecteur quantifie
*                               est different de l'elment constitutif de ELTS
        MOVE.W  D2,D5           passage par valeur intermediaire
        MULU    (A2)+,D5        calcul du biais a accumuler
        ADD.L   D5,D1           accumulation
        BRA.S   ACCUM           one more time !

ACTU    MULU    (A2),D2        pre multiplication
        SUB.W   #1,(A2)         on a place un element:son poids diminue

        DIVU    D6,D2           actualisation de la racine (reste = 0)
        DIVU    D6,D1           restitution de l'echelle initiale de l'indice

        MOVE.W  D0,D6           decremente D6 de 1
        DBF     D0,NUMERO       fin de la boucle

        ADD.W   KBIAI(A5),D1    addition du biais
        MOVE.W  D1,INDICE       sauvegarde en memoire
*****************************************************************************

*       restitution du vecteur quantifie a partir de son indice

*       recherche de la classe a partir de l'indice

*               A0      pointeur dans le tableau des biais
*               A4      pointeur relatif table des leaders
*               A5      pointeur relatif tableaux RAC et BIAIS
*               D0      compte des iterations avant de trouver la classe ICLAS
*               D1      variable intermediaire pour le biais
*               D2      indice recu

        LEA     KBIAI,A0
        MOVE.L  #0,D1           en particulier mise a 0 des LSB
        MOVE.W  #(KLEAD-1),D0   on effectue un decompte / fin

SUBIAI  MOVE.W  (A0)+,D1        passage du biais en "long mot"
        CMP.L   D1,D2           compare les biais par ordre decroissant a D2
        DBGE    D0,SUBIAI       branche et decremente tant que biais > indice

        LEA   -(2+KBIAI)(A0),A5 stocke pointeur relatif tableaux RAC & BIAIS
        SUB.W   KBIAI(A5),D2    suppression du biais D1 = indice dans classe
        MULU    #(2*LONG),D0    calcul pointeur relatif dans table des leaders
        MOVE.L  D0,A4           le stocke dans A4

*       exploration du leader de la classe "recue"

*               -elements constitutifs
*               -poids

*               A0      pointe sur le leader
* routine (R)   A1      pointe sur le leader avec une avance d'un mot sur A0
*               A2      pointe sur le tableau des poids
*               A3      pointe sur le tableau des elements constitutifs
*               A4      trace du leader retenu(pointeur relatif)
* preserver (P) A5      trace classe retenue sous forme de pointeur relatif
```

```
*                D0      nombre d'iterations -1
* R              D1      accumulation du poids d'un element
*                D2      indice relatif

        LEA      KLD0(A4),A0     positionne A0 sur debut du leader
        LEA      POIDSR,A2
        LEA      ELTSR,A3
        MOVE.L   #(LONG-1),D0    initialisation du nombre d'iterations

        BSR      EXPLOR

*       denumerotage

*                A0      vecteur decode TABR
*                A1      elements constitutifs ELTSR
*                A2      poids correspndants POIDSR
*                A5      pointeur relatif dans RAC
*                D0      indice de boucle
*                D1      largeur domaine permutations possibles
*                D2      indice relatif recu INDICE
*                D3      indice accumule reconstitue INDACC
*                D4      trace de LONG - 1
*                D5      racine de numerotage RACINE
*                D6      indice relatif "a l'echelle" (IREF)
*                D7      variable intermediaire ITP

        MOVE.W   KRAC(A5),D5     lecture de la racine dans la table
        LEA      TABR,A0

        MOVE.W   D4,D0           initialise pointeur de boucle
        MOVE.W   D0,D1
        ADD.W    #1,D1           initialisation de D1 par rapport a D0
        MOVE.W   D2,D6
        MULU     D1,D6           initialisation de IREF a D1*indice
        MOVE.L   #0,D3           mise a 0 de INDACC

DENUM   MULU     D1,D3   mise a l'echelle de l'indice accumule
        LEA      (ELTSR-2),A1    initialisation des pointeurs
        LEA      POIDSR,A2

RESTI   MOVE     (A2)+,D7        ITP = POIDSR(j)
        MULU     D5,D7           ITP = POIDSR(j)*RACINE
        ADD.W    D3,D7           ITP = INDACC + POIDSR(j)*RACINE
        ADD      #2,A1           incremente d'un mot le pointeur dans ELTSR

        CMP.L    D7,D6           CC selon IREF - ITP
        BLT.S    ACTUR           branche quand l'indice accumule est trop grand

        MOVE.W   D7,D3           INDACC = ITP
        BRA.S    RESTI

ACTUR   MOVE     (A1),(A0)+      TABDR(i) = ELTSR(j)
        MULU     -2(A2),D5       RACINE = RACINE*POIDSR(j) premultiplication
        SUB.W    #1,-2(A2)       POIDSR(j) = POIDSR(j) -1
        DIVU     D1,D5           RACINE / "largeur des permutations"
```

```
        DIVU    D1,D3           INDACC / "------- --- ------------"
        SUB.W   D2,D6           IREF = IREF -INDICE
        MOVE.W  D0,D1           revient a decrementer D1 d'une unite

        DBF     D0,DENUM
*****************************************************************************

*       routines


*       exploration du leader retenu :
*               -elements constitutifs
*               -poids

* argument (A)  A0      pointe sur le leader
* A             A2      pointe sur le tableau des poids (AaI)
* A             A3      pointe sur le tableau des elements constitutifs (AaI)
* P             A5      trace classe retenue sous forme de pointeur relatif
* A             D0      indice de la boucle
*               D1      variable intermediaire

EXPLOR  MOVE.W  (A0),D1
        EXT.W   D1              demultiplexage poids leader
        MOVE.W  D1,(A3)+        ecriture elt
        MOVE.L  #1,D1
        ADD.B   (A0),D1
        MOVE.W  D1,(A2)+        POIDS = POIDSmux +1

EXPLO   CMP.W   #255,(A0)+      CC suivant LEADmux - 255
        DBLE    D0,EXPLO        boucle tant que la fin de classe non atteiente
        DBRA    D0,EXPLOR       va ecrire element suivant s'il existe

        RTS
```

ANNEXE F

EXEMPLE DE LOGICIEL UTILISANT LE CIRCUIT DE SYNTHESE

Exemple de programme utilisant le SP1000 à partir de l'EVM et de l'AIB. Ce programme charge le synthétiseur à partir d'un segment de parole codé sur la PDP-11 et stocké dans sa mémoire programme.

```
>
**
*
*       programme effectuant la synthese sur SP1000 de
*       donnes d'analyse ILS formattee par le programme
*       SP1000.ASM
*       ce programme emule le hardware bati autour de la
*       carte AIB
*
**
UN      EQU     16
PRIMEL  EQU     17
NBEL    EQU     18
CNTLSP  EQU     19
CNTLAD  EQU     20
REARM   EQU     21
SRAD    EQU     22
SRSP    EQU     23
GAINS   EQU     24
PTMAX   EQU     25
PTRAM   EQU     26
TMP     EQU     27
SAR0    EQU     28
SAR1    EQU     29
ISPCPK  EQU     30
*
*
        B       DEBUT
*
*       interruption
*
        OUT     REARM,3         !rearmement des IT du sp
        NOP
        OUT     REARM,3
        LAC     PRIMEL
        ADD     UN,8
        SACL    TMP
A1      BIOZ    A1
        OUT     TMP,3   !emmission de l'adresse de debut
*
        LAR     0,NBEL
        LAR     1,PRIMEL
SP      LARP    1
A2      BIOZ    A2
        OUT     *+,3,0
        BANZ    SP
*
        B       GET
*
BLOCD0  DATA    0,14,81,7,1024,0,529,6,3042
BLOCD1  DATA    256
*
*
DEBUT   LDPK    0
        LACK    1
```

```
        SACL    UN
        LACK    BLOCD0
        LARK    0,8
        LARK    1,PRIMEL
*
D0      LARP    1               lecture du bloc de datas BLOCD0
        TBLR    *+,0
        ADD     UN,0
        BANZ    D0
*
        OUT     CNTLAD,0        controle de l'A/D
A3      BIOZ    A3
        OUT     CNTLSP,3        controle du synthetiseur
        LAC     SRSP,0
        SACL    >E              position de SR dans la trame d'analyse
*
INISOF  LACK    BLOCD1          lecture du debut de datas d'analyse
        TBLR    PTRAM
*
GET     LAC     PTRAM,0         test de fin de phrase
        SUB     PTMAX,0
        BGEZ    INISOF
*
        LARK    0,6             lecture d'une trame compacte
        LARK    1,ISPCPK
        LAC     PTRAM,0
SPCPK   LARP    1
        TBLR    *+,0
        ADD     UN,0
        BANZ    SPCPK
*
        SACL    PTRAM,0
*
        LARK    0,6             conditionnement en vue de l'emmission
        LARK    1,ISPCPK
        ZAC
        SACL    SAR1,0
*
DECPK   LARP    1
        SAR     0,SAR0
        LAR     0,SAR1
        LAC     *,8,0
        SACH    TMP,0           separation LSB/MSB
        LACK    255
        AND     TMP
        ADD     UN,9            mode ecriture
        SACL    *+,0,1
        LACK    255
        AND     *+,0
        ADD     UN,9
        SACL    *+
        SAR     0,SAR1
        LAR     0,SAR0
        BANZ    DECPK
*
```

```
        LAC     >A,0
        ADD     GAINS,0
        SACL    >A              programmation du gain de sortie
*
        EINT
*
ATTENT  B       ATTENT
*
```

CACC / CCAC
86126

MABILLEAU, PHILIPPE.
--Etude sur la réalisation...

P
91
C655
M15
1985

DATE DUE
DATE DE RETOUR

LOWE-MARTIN No. 1137