Ecole Supérieure d'Ingénieurs de Luminy

Carte Audio

(DSP)

Projet d'Architecture des Ordinateurs

Présenté par :
Jacqueline BOLLET – Thierry GRENIER

Responsable du projet :
Christian OLIVETTO

Année scolaire 1999-2000

Présentation succinte du Projet d'Architecture des Ordinateurs

Notre projet porte sur le DSP (Digital Signal Processor).
Nous avons utilisé le kit d'Analog Devices : EZ-Kit Lit, équipé d'un DSP ADSP- 2181 (Famille des 2100 de DSP chez ce fournisseur).
Notre objectif a été de découvrir, de comprendre son fonctionnement à travers son architecture. Nous avons été amenés à étudier le langage assembleur de la carte fournie grâce aux routines fournies par le constructeur du DSP. Cela pour être en mesure de programmer une routine de saisie et effectuer une compression de la voie sonore. Nous avons conçu une interface de communication avec le DSP à partir d'un PC via RS232 par le logiciel Visual Basic (Microsoft) sur plateforme Windows.

Le Projet de Carte Audio Avec un DSP d'Analog Devices

Table des matières

Le Projet de Carte Audio Avec un DSP d'Analog Devices

Introduction

Le projet d'architecture des ordinateurs se proposait l'étude et l'utilisation d'un kit à base de circuit DSP (Digital Signal Processors) par la saisie de la voie sonore (échantillonnage) et par un traitement de compression.

Le kit choisi a été l'EZ-KIT Lite équipé d'un processeur ADSP-2181 et d'un codec AD-1847 (Kit d'Analog Devices).
Ce kit permet une conversion de la voie humaine en données digitales, récupérées par le PC via la liaison RS232.

Les principaux domaines abordés par le projet sont :
- la Conversion Analogique/Numérique et Numérique/Analogique (réalisé par le Codec).
- les Circuits spécifiques du traitement du signal (type d'échantillonnage, mode audio : stéréo, mono…).
- la Gestion de la liaison série avec le PC (liaison RS232).
- la Création et la manipulation de fichiers sous environnements Windows.
- l'Affichage et la gestion de fenêtres sous environnement Windows.
- la Compression et le traitement de la voie humaine.

Nous avons essayé lors de notre projet de répondre aux demandes faites par les objectifs du projet.
C'est à dire :
- enregistrer la voie sonore
- convertir en fichier données et compresser.
- transférer le fichier vers le PC, visualiser sur l'écran de celui-ci et l'écouter par sa carte son (écriture en format fichier standard).

Mais avant d'aborder le sujet plus en profondeur, voyons ce qu'est un DSP…

Le Projet de Carte Audio Avec un DSP d'Analog Devices

CHAPITRE 2

Qu'est ce qu'un DSP ?

- Le cœur d'un système de traitement numérique du signal
- Différences entre un processeur classique et un DSP
- Types et formats de données manipulées par les DSP

Un DSP est un type particulier de microprocesseur. Il se caractérise par le fait qu'il intègre un ensemble de fonctions spéciales. Ces fonctions sont destinées à le rendre particulièrement performant dans le domaine du traitement numérique du signal.

Comme un microprocesseur classique, un DSP est mis en œuvre en lui associant de la mémoire (RAM, ROM) et des périphériques. Un DSP typique a plutôt vocation à servir dans des systèmes de traitements autonomes. Il se présente donc généralement sous la forme d'un microcontrôleur intégrant, selon les marques et les gammes des constructeurs, de la mémoire, des timers, des ports série synchrones rapides, des contrôleurs DMA, des ports d'E/S divers.

Le cœur d'un système de traitement numérique du signal.

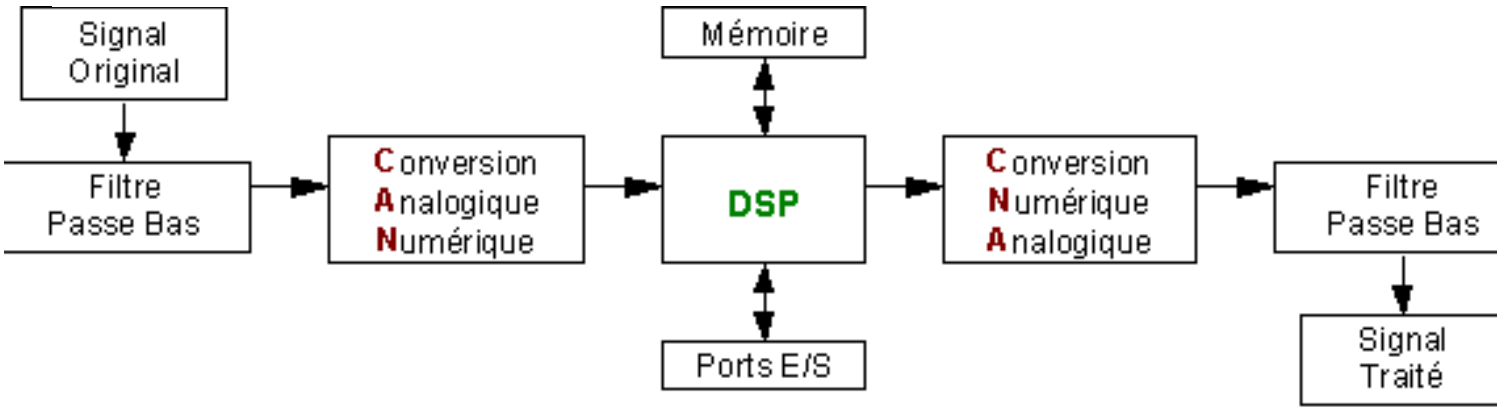


Fig. 1 : Chaîne complète typique d'un système de traitement numérique du signal

Tous les systèmes à bases de DSP bénéficient des avantages suivants :

- Souplesse de la programmation : un DSP est avant tout un processeur exécutant un programme de traitement du signal. Ceci signifie que le système bénéficie donc d'une grande souplesse de développement. De plus, les fonctions de traitements numériques peuvent évoluer en fonction des mises à jour des programmes, et cela pendant toute la durée de vie du produit incluant le système. Ainsi, modifier par exemple tel ou tel paramètre d'un filtre numérique ne nécessite pas un changement matériel. .

- Implémentation d'algorithmes adaptatifs : une autre qualité issue de la souplesse des programmes. Il est possible d'adapter une fonction de traitement numérique en temps réel suivant certains critères d'évolutions du signal (exemple : les filtres adaptatifs).

- Des possibilités propres au système de traitement numérique du signal . Certaines fonctions de traitement du signal sont difficiles à implanter en analogique, voire irréalisables (exemple : un filtre à réponse en phase linéaire). .

- Stabilité : en analogique, les composants sont toujours plus ou moins soumis à des variations de leurs caractéristiques en fonction de la température, de la tension d'alimentation, du vieillissement, etc. Une étude sérieuse doit tenir compte de ces phénomènes, ce qui complique et augmente le temps de développement. Ces inconvénients n'existent pas en numérique. .

Différences entre un processeur classique et un DSP

En quoi un DSP est-il différent d'un microprocesseur ? Pour répondre à cette question, il nous faut prendre en considération les éléments suivants :

L'opération MAC (Multiplier/ Accumulator)

Après avoir été numérisé, le signal se présente sous la forme d'une suite de valeurs numériques discrètes . Cette suite d'échantillons est apte à être stockée et traitée par un système informatique. Par nature, le traitement numérique du signal revient à effectuer essentiellement des opérations arithmétiques de base du type A = (B x C) + D.

Un microprocesseur classique va nécessiter plusieurs cycles d'horloge pour effectuer un tel calcul, par exemple, un 68000 à besoin de :

· 10 cycles d'horloge pour effectuer une addition,
· 70 cycles d'horloge pour effectuer une multiplication.

Soit 80 cycles pour seulement calculer A. Si ce temps est admissible dans des applications informatiques courantes, il n'est pas acceptable pour faire du traitement rapide du signal.

Les DSP sont donc conçus pour optimiser ce temps de calcul, à cet effet, ils disposent de fonctions optimisées permettant de calculer A beaucoup plus rapidement.

Dans la pratique, la plupart des DSP ont un jeu d'instructions spécialisé permettant de lire en mémoire une donnée, d'effectuer une multiplication puis une addition, et enfin d'écrire en mémoire le résultat, le tout en un seul cycle d'horloge. Ce type d'opération est nommé MAC, de l'anglais Multiplier/ACcumulator.

Effectuer une opération MAC en seul cycle n'est malgré tout pas satisfaisant si le cycle d'horloge est trop « long ». Le principal objectif d'évolution des DSP a toujours été d'améliorer le temps de calcul d'un MAC.

Actuellement, un DSP de gamme moyenne effectue une opération MAC sur des données de 16 bits en moins de 25 ns, soit 40 000 000 opérations par seconde. De telles performances sont indispensables pour effectuer des traitements rapides.
Outre le temps d'exécution d'une opération MAC, un autre problème se pose.
L'opération MAC étant une multiplication suivie d'une addition, un débordement (1) de l'accumulateur est toujours possible. Pour contourner ce problème, certains DSP possèdent un accumulateur adapté au MAC. Ces accumulateurs ont un format spécial incorporant des bits supplémentaires (bits de garde) par rapport à la taille des données à manipuler. Les problèmes de débordements sont alors contournés, car un programme de traitement correctement conçu ne devrait pas générer des suites d'opérations MAC telles qu'un résultat excède la capacité élargie de l'accumulateur.

(1) : débordement au sens informatique : l'opération génère un nombre supérieur au plus grand nombre pouvant être manipulé par l'accumulateur.

L'accès à la mémoire

Outre l'opération MAC, une autre caractéristique des DSP est leurs capacités à réaliser plusieurs accès mémoire en un seul cycle. Ceci permet à un DSP de chercher en mémoire une instruction et ses données réalisant un MAC, et simultanément, d'y ranger le résultant du MAC précédent. Le gain de temps est évident. Toutefois, sur certains DSP basiques , ce type d'opération simultané est généralement limité à des instructions spéciales. Ces instructions utilisent un mode d'adressage restreint, c'est à dire ne portant que sur de la mémoire vive intégrée au DSP.

Les modes d'adressages des données sont un point particulier des DSP. Un DSP peut posséder plusieurs unités logiques de génération d'adresse (**DAG1 et DAG2 pour l'AD2181**), travaillant en parallèle avec la logique du cœur du DSP. Une unité logique de génération d'adresse est paramétrée une seule fois via les registres appropriés. Elle génère alors toute seule, en

parallèle avec l'exécution d'une opération arithmétique, les adresses nécessaires à l'accès des données.

Ceci permet non seulement de réaliser les accès mémoires simultanés en seul cycle, comme décrit plus haut, mais également d'incrémenter automatiquement les adresses générées. Ce mode d'adressage particulier, généralement appelé **adressage indirect** par registre avec post (ou pré) incrément, est très utilisé pour effectuer des calculs répétitifs sur une série de données rangées séquentiellement en mémoire. (Instruction : « dm » sur le kit fourni).

Si l'on ajoute la possibilité de générer ces adresses selon un *modulo* paramétrable, ce mode d'adressage devient circulaire, et permet donc de créer des buffers circulaires en mémoire. L'adressage circulaire indirect (parfois également indexé) par registre avec post (ou pré) incrément prends toute son importance quand il est utilisé pour accéder aux opérandes d'un MAC.

Types et formats de données manipulées par les DSP

Un autre point essentiel des DSP est la représentation des nombres (les données) qu'ils peuvent manipuler.
Il est possible de distinguer deux familles :

- Les DSP à **virgule fixe** : les données sont représentées comme étant des nombres fractionnaires à virgule fixe, (compris entre -1.0 et +1.0), ou comme des entiers classiques. La représentation de ces nombres fractionnaires s'appuie la méthode du « complément à deux » (cf tableau ci-dessous). L'avantage de cette représentation (qui n'est qu'une convention des informaticiens) est de permettre facilement l'addition binaire de nombres aussi bien positifs que négatifs.

Par exemple :

MSB			LSB	
-2^3	2^2	2^1	2^0	
-8	4	2	1	
0	1	0	1	= 4+1 = 5
1	1	0	1	= -8+5 = -3
0	0	0	1	= +1 le moins positif
0	1	1	1	= 4+2+1 = 7 le plus positif
1	1	1	1	= -8+7= -1 le moins négatif
1	0	0	0	= -8 le plus négatif

- Les DSP à **virgule flottante** : les données sont représentées en utilisant une mantisse et un exposant. La représentation de ces nombres s'effectue selon la formule suivante : n = *mantisse* x $2^{exposant}$. Généralement, la mantisse est un nombre fractionnaire (-1.0 à +1.0), et l'exposant est un entier indiquant la place de la virgule en base 2 (c'est le même mécanise qu'en base 10).

Le kit que nous utilisons est équipé d'un DSP à virgule fixe. (coût moins important et suffisant pour les applications que nous désirons implémenter).

Le Projet de Carte Audio Avec un DSP d'Analog Devices

Performances du kit

- Qu'est ce que le Ezkit Lite ?
- Le processeur ADSP 2181

Dans cette partie, nous avons voulu présenter les caractéristiques de notre support de travail.

Qu'est ce que le Ezkit Lite ?

C'est un kit complet de développement et de démonstration sur DSP (Digital Signal Processor).
Il comprend un processeur ADSP 2181 cadencé à 33MHz, un AD1847 CODEC (**CO**deur **DEC**odeur) stéréo et une EPROM contenant le code du programme "Monitor" pour pouvoir charger de nouveaux algorithmes sur le DSP au travers d'une connexion RS232.(Figure 1)

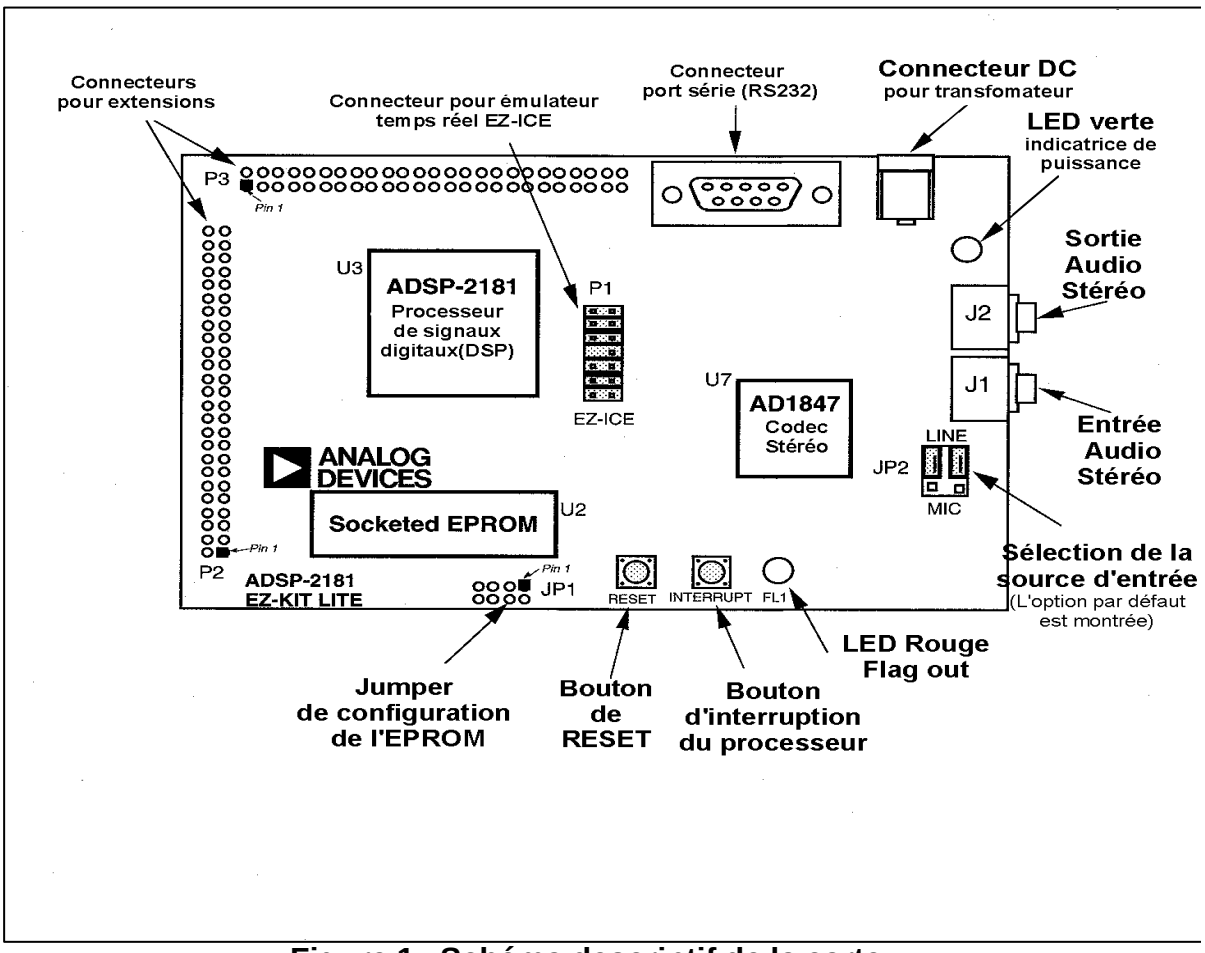


Figure 1 . Schéma descriptif de la carte

Le CODEC AD1847 se charge d'échantillonner le signal entrant dans un format sélectionné, avec ou sans compression. Ces échantillons sont transmis vers le DSP par une liaison série (autre que celle utilisée par le moniteur), caractérisée par deux canaux : SDI (**S**erial **D**ata **I**nput) et SDO (**S**erial **D**ata **O**utput). Cette liaison est bidirectionnelle, c'est à dire que le DSP peut également envoyer des échantillons vers le CODEC pour qu'il les reconvertisse en signaux analogiques.

Le programmeur n'a pas à s'inquiéter du séquencage et du protocole de transmission entre le DSP et le CODEC, tout est géré en hardware par le kit.

Deux boutons sont disponibles pour l'utilisateur:
- le bouton RESET sert à sortir des boucles infinies et autres conflits, car son interruption est prioritaire par rapport à toutes les autres. En réalité, il sert surtout à rétablir les communications séries vers le moniteur. (car le programme de l'EPROM est directement chargé après mise en route ou redémarrage)
- le bouton d'interruption du processeur

Pour compléter cette description, il faut savoir que l'EZ-KIT LITE ne nécessite pas de mémoire externe au DSP, grâce à la mémoire comprise sur le DSP lui même. Elle comprend 16384 emplacements de 24 bits pour la mémoire de programme et 16352 emplacements de 16bits pour la mémoire de données
(32 emplacements sont réservés pour les registres de contrôles du système et ne peuvent être utilisés pour d'autres codes). Ces deux mémoires sont des SRAM (**S**tatic **R**andom **A**ccess **M**emory).

A cela s'ajoute l'architecture HARVARD du DSP, permettant entre autre l'accès simultané à la mémoire de données et à la mémoire de programme tout en accomplissant une opération arithmétique. On a également des opérations MAC (**M**ultiply and **AC**cumulate) à notre disposition permettant de réduire le nombre de cycles passés aux divers calculs (comme par exemple les opérations de filtrage).

Le processeur ADSP 2181

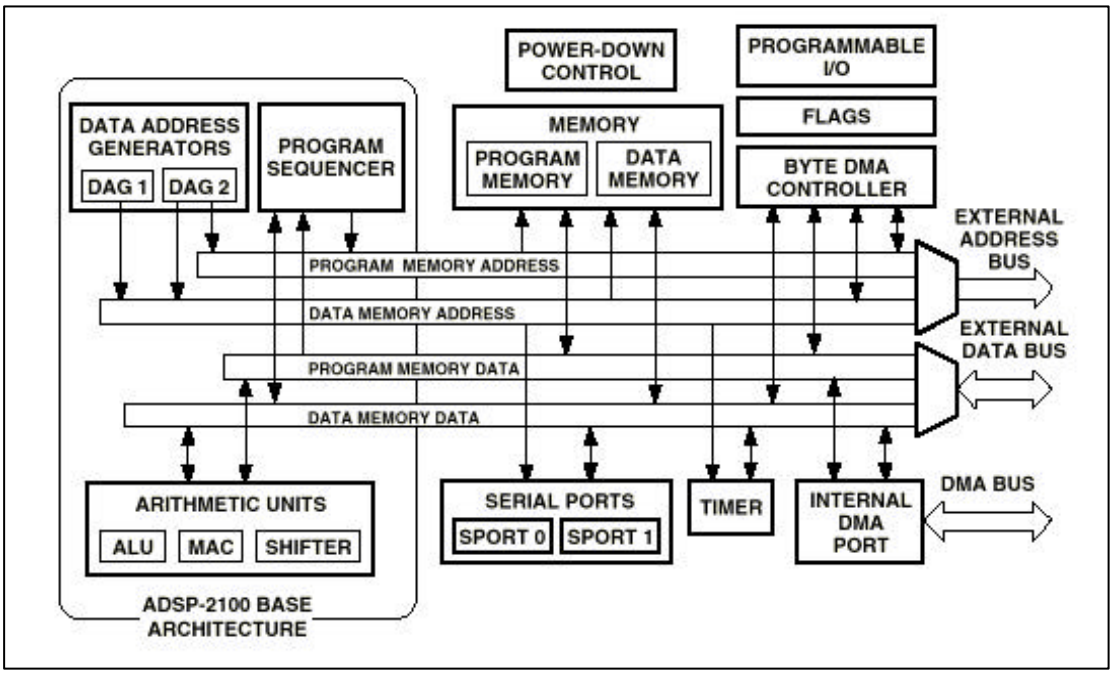


Figure 2 . Description du processeur ADSP 2181

Périphériques intégrés dans l'ADSP-2181 :

L'ADSP 2181 associe l'architecture de base (3 unités de calcul - ALU, MAC et SHIFTER – 2 générateurs d'adresses de données – DAG1 et DAG 2 – et un séquenceur de programme) avec 2 ports série (SPORT 0 et SPORT 1), un port DMA interne (IDMA) 16 bits, un Byte DMA port (BDMA), un timer programmable, des entrées/sorties (I/O) programmables et de la mémoire. (Figure 2)

Détail des composantes du processeur:

- L'ALU (Arithmetic Logic Unit) traite les fonctions arithmétiques (addition, soustraction, négation, incrémentation, décrémentation et valeur absolue) et logiques standards (AND, OR, XOR (Ou exclusif) et NOT). Il fournit un résultat de 16 bits avec deux entrées de 16 bits.

- L'unité MAC (Multiply/accumulator) fournit des traitements rapides de multiplication, de multiplication avec addition cumulative, de multiplication avec soustraction cumulative et des fonctions de remise à O.
  Exemples de fonctions standards :
  X*Y : Multiplie X par Y
  MR+X*Y : Multiplie X par Y et ajoute le résultat au registre MR

  MR-X*Y : Soustrait MR par le résultat de la multiplication de X par Y

- Le SHIFTER effectue des opérations de décalage logiques et arithmétiques.

- DAG1 et DAG2 : Ces générateurs d'adresses de données fournissent les adresses de la mémoire quand la donnée de cette mémoire est transférée vers les registres d'entrée ou de sortie. Autrement dit, ils gardent les adresses de données manipulées mais aussi celles de programmes.

- Le PROGRAM SEQUENCER : le séquenceur de programme assure une utilisation efficace des 3 unités de calculs ALU,MAC et SHIFTER. Il fournit des adresses d'instructions au program memory.

- Un timer programmable qui génère des interruptions périodiques.

- Deux ports série synchrones (SPORT 0 et SPORT 1) permettant la communication série et la communication multiprocesseur. Ils peuvent utiliser une horloge externe ou travailler avec leur propre horloge interne. Différents modes de communication série sont paramétrables :
  - Le mode bidirectionnel : chaque SPORT peut transmettre et recevoir indépendamment.
  - Le mode bufferisé : Chaque SPORT peut transmettre et recevoir en même temps.
  - La compression des données suivant la loi A ou μ.
  - Le multiplexage en temps permettant une communication multiprocesseur.

- IDMA et BDMA : Ces deux ports internes fournissent des transferts de données performants entre le DSP et le système extérieur.

- L'AD2181 intègre 80K octets de mémoire divisée en deux blocs de mémoires, programme et données. Il y a 16Kmots (en 24 bits) de mémoire programme (program memory) et 16Kmots (en 16 bits) de mémoire données (data memory).
- Performances : 1 Quartz de 33 Mhz pour 25 ns de temps de cycle

- Interfaçage externe de la mémoire programme :

L'ADSP2181 peut adresser jusqu'à 16 Kmots (1 mot = 24 bits) de mémoire programme externe.

- Interfaçage externe de la mémoire données locale :

L'ADSP2181 peut adresser jusqu'à 16 Kmots (1 mot = 16 bits) de mémoire données locale externe.

Principaux éléments de l'architecture interne du processeur :

- Multiplier/Accumulator (MAC)
- Unité logique et arithmétique (ALU)
- Unité arithmétique de décalage SHIFTER
- Générateurs d'adresses DAG1 et DAG2
- Program Sequencer

- Boot intégré en ROM interne : 64K octets de mémoire permettent l'initialisation du DSP à la mise sous tension.

Tableau récapitulatif:

Circuit	ADSP-2181
Temps de cycle (ns)	**25**
RAM interne	**16K Data + 16K Program**
ROM interne	**64 Ko (1 octet=8bits)**
Mémoire externe adressable	**16 Kmots (1 mot=16bits) DM** **16 Kmots (1 mot=24bits) PM**
Ports série	**2 synchrones**
Timer	**1**
Fréquence d'horloge (Mhz)	**33**
Performance (MIPS)	**33**
Architecture	**Harvard**
Résolution	**16 bits**
Virgule	**Fixe**

Le Projet de Carte Audio Avec un DSP d'Analog Devices

Notre étude de conception.

- Définition des ressources nécessaires
- Notre approche du projet
- Réalisation des objectifs et précisions sur nos choix

Définition des ressources nécessaires

1.Les besoins matériels.

Cette liste comprend les divers éléments indispensables à la bonne marche du projet:

- un ordinateur compatible PC comprenant:
  → un écran, un clavier, une souris
  → au moins 16 Mo de RAM
  → un disque dur de l'ordre de 2 Go
  (la manipulation de fichiers sonores nécessite de la place)
  → un lecteur de disquette et cd-rom
  (afin d'installer les logiciels nécessaires)
  → une carte son (pour écouter les fichiers produits)
  → un port série RS 232 pour communiquer avec la carte
- la carte DSP
- un micro et des hauts parleurs
- un câble jack-jack afin de pouvoir relier l'entrée son de la carte sur une source autre que le microphone

2.Les besoins logiciels.

Ci-dessous sont présentés les différents logiciels dont nous avons eu besoin au cours de ce projet:

- un système d'exploitation (Windows 3.11 ou Supérieur)
- le logiciel Microsoft Visual Basic (Version 4 ou Supérieure) afin de programmer l'interface graphique du projet
- un traitement de texte Microsoft Word (Version 97 ou Supérieure)
- le programme ADSP EZ-KIT Lite (comprenant un compilateur pour le langage assembleur du processeur, un éditeur de liens, un simulateur de fonctionnement du processeur, le EZ-KIT Lite Host Program)
- un programme qui nous a été fourni : Ezkitsde (il facilite la création, la compilation et le linkage de routine assembleur ; il la charge sur la carte et permet d'envoyer des caractères numériques sur le port série).
- un logiciel de manipulation de fichiers sons au format Wave afin de lire nos fichiers sons avec le programme d'un tiers.

Nous avons eu besoin également des sources des codes de compressions en langage assembleur de la carte.

Notre approche du Projet

Pour réaliser notre projet, nous avons vite distingué 2 parties :

- une partie de code assembleur
- une partie d'interface sous Windows.

Pour cette dernière partie, nous avons rapidement choisi Visual Basic qui semblait le mieux adapté.

Ainsi, l'utilisateur grâce à l'interface en Visual Basic, déclenchera une routine en assembleur qui permettra l'acquisition du son en entrée sur la carte.

En effet, la routine assembleur, chargée sur la carte DSP (par l'interface) va réaliser la saisie des données. (échantillons du son)

Ensuite il s'agira de rapatrier ses données saisies vers le PC afin qu'elles puissent être disponibles pour l'utilisateur.

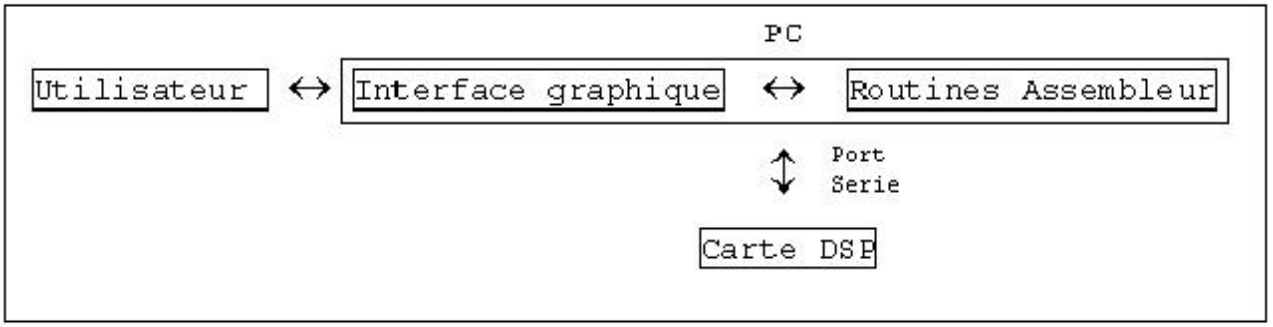


Schéma de fonctionnement

La carte sera pilotée depuis le PC via le port série RS232.

Réalisation des objectifs et précisions sur nos choix

L'objectif était de réaliser un traitement qui consistait à :

L'Enregistrement

1) Enregistrer de la voie sonore

Cette partie est dans le code de la routine assembleur à coder.
On a voulu implémenter les modes stéréo et mono.

Pour les fréquences, nous nous sommes imposé un minimum de 8000 hz.
En effet, un bon échantillonnage se doit de respecter le théorème de Shannon (afin d'éviter le repliement de spectre).

Ce théorème précise qu'il faut :

Fe > 2 Fmax avec :
Fe = Fréquence d'échantillonnage.
Fmax = Fréquence maximale du signal à échantillonner.

Or, la voix humaine a une fréquence maximale de 4000 Hz.
Il suffit donc d'avoir au moins 8000 Hz en échantillonnage.

Nous avons également choisi de faire du 8 ou du 16 bits (obligatoire pour que les lois de compressions soient efficaces comme on le précisera plus tard).

Nous avons eu le souci d'optimiser au cours de ce projet la place en mémoire occupée par la saisie de données.

En effet, on dispose sur la carte d'une mémoire en data de 16 K d'emplacements à 16 bits mais dont environ 15 332 emplacements disponibles pour un programme chargé en sous routine sur la carte (nous ne modifions pas le programme résidant : le monitor).

Aussi, on ne pourra s'étonner de notre choix de 15000 échantillons à 16 bits.
Et de 30000 en 8 bits.

Cela nous a permis d'enregistrer par exemple:
- Sans compression (PCM) : T= 1/8000 (hz) *15000 = 1,875 seconde d'enregistrement en mode mono.
- En compressant, nous pouvons obtenir 1,875*2 = 3,75 seconde en mono.

La conversion des données

2) Convertir les données

Cela se déroule en 3 phases :

- Le traitement et la compression
- Le transfert des données
- La création de fichier en format standardisé

Traitement et compression,

➢ *Le traitement et la compression*

Sur la carte, nous pouvions implémenter 3 types de compression :

- ADPCM
- µlaw
- Alaw

L'ADPCM

L'APCM n'a pas pu être implémenté (se reporter à la section : nos problèmes)

LOI A et LOI µ

La loi µ et la loi A sont de lois logarithmiques de compressions des données. (formats compressés 8 bits, utilisés surtout dans l'industrie des télécommunications; les deux codages sont identiques: le premier est le standard en Europe, le second est le standard aux USA et au Japon; leur principe est le suivant: la formule utilisée est non linéaire et code plus fortement les signaux de faible amplitude et plus faiblement les signaux de forte amplitude; la croissance de la courbe de la fonction utilisée est logarithmique)

La loi µ est Américaine et la loi A est Européenne.

La loi µ est définie telle que :
On dit que µ=255 et
$Y = (\ln(1+\mu x)) / (\ln(1+\mu))$
Et x = Ue/Umax où Ue est la tension à l'échantillon et Umax , la tension maximale sur tous les échantillons.

La loi A est telle que :
On dit que A vaut 87,5 et on a : x = Ue /Umax où Ue est la tension à l'échantillon et Umax , la tension maximale sur tous les échantillons.

$Y = A / (1+\ln A)\ x$ si $0 < x < 1/A$
$Y = (1 + \ln Ax) / (1+\ln A)$ si $1/A < x < 1$

Transfert des données

➢ *Le transfert des données.*

Il s'agit ici de remarquer qu'il faut procéder à 2 types de transfert :
- un chargement du programme (exécutable généré sur le PC) sur la carte DSP.
- Un chargement des données saisies par la carte vers le PC.

Le chargement du programme vers la carte a été réalisé par l'appel du programme Ezld (charge sur la carte, des données dans la Programm Memory) par Visual Basic.

Le chargement des données saisies vers le PC s'est réalisé par un $UD (Upload Data).

➢ *Création de fichier au format standardisé*

Création de fichier

Il a fallu pour cette partie, déterminer les formats récupérés pour les données.

Format 1.15: un bit de signe et 15 bits représentant des valeurs comprises entre –1 et 1 (strictement inférieure).

Le fichier sera crée par notre logiciel qui y placera les en-têtes correspondantes selon la fréquence d'échantillonnage, la taille d'échantillonnage et le nombre de canaux.

Structure d'un fichier au format Wave

Le format Wave est auto-descriptif : il intègre une en-tête qui précise la manière dont sont codées les données sonores dans le fichier (entiers signés ou non, entiers longs ou courts…).

Un fichier Wave est une composé de plusieurs parties.

Deux sont indispensables:
- celle qui contient les paramètres importants décrivant la forme d'onde (fréquence, taux d'échantillonnage…)
- celle qui contient les données constituant la forme d'onde .

A titre d'exemple, voici le détail des codes hexadécimaux de tous les segments de données contenus dans l'entête d'un fichier au format Wave dont les caractéristiques sont
une fréquence de 8000 Hertz, un échantillonnage de 16 bits, et un enregistrement monophonique.

Un exemple d'en-tête de fichier Wave

5249 4646 1 5741 5645 666D 7420 RIFF....WAVEfmt
1200 0000 0100 0100 401F 0000 803E 0000 .....@....>..
0200 1000 0000 6661 6374 0400 0000 0000 ......fact.....
0100 6461 7461 2 0000 0000 0000 ..data.....

Les trois zones grisées contiennent:
1 – la longueur du fichier
2 – la longueur du fichier – la longueur de l'entête
On peut noter, surlignée en jaune la fréquence d'échantillonnage, en bleu le nombre de bits par échantillon,
en vert le nombre de canaux.

Les trois mots clés "WAVEfmt" (Wave format), "fact" et "data",
délimitent le début et la fin de l'entête et de la zone de données.

La visualisation

3) Visualisation des données et écoute par le magnétophone.

Cette partie est uniquement réalisé par l'interface Visual Basic.
L'on a créé une fenêtre graphique qui permet la visualisation des données en mono comme en stéréo.

L'écoute se fait tout d'abord par la création d'un fichier Wave puis on récupère le fichier Wave avec le magnétophone de Windows. On peut alors l'écouter.
Mais il est aussi intéressant de sélectionner « propriétés » lors des options proposées lors du click droit sur le fichier ainsi créé.
On peut alors voir les caractéristiques du fichier (compression ou non, laquelle si oui, mode stéréo ou mono, nombre de bits et fréquence d'échantillonnage, longueur du fichier…).

Le Projet de Carte Audio Avec un DSP d'Analog Devices

L'organisation du Travail

Remarque: Nous avons pensé que cinq binômes groupés (soit un total de dix personnes) seraient difficiles à organiser pour ce projet.
Aussi avons-nous décidé de scinder le groupe initial en deux afin de rendre sa gestion plus simple.
Cela ne signifiera pas pour autant qu'il n'y aura pas de communication entre les deux sous groupes.

Nous avons distingué 2 parties composées elles même de deux sous parties et nous y avons affecté des responsables:

Répartition des tâches

Binômes	Parties	Sous-parties	Responsables
Barkats & Colonna	Documentation & Algorithmique	Recherches sur: formats 1.15,Wave,ADSP…	Barkats
		Algorithmique: conversion de formats, VB…	Colonna
Bollet & Grenier	Implémentation	Assembleur	Bollet
		Visual Basic	Grenier

Ce découpage des tâches n'a pas empêché la mise en commun permanente de notre travail et la redistribution des rôles selon l'état d'avancement du projet.

Par ailleurs, nous nous sommes chacun organisé par binôme ensuite.

Pour notre part, cela ne nous a pas beaucoup changé :

Répartition interne au binôme.

BOLLET :
implémentation des algorithmes de compression et des routines en assembleur.

GRENIER:
implémentation des entête de fichiers (Wave) et de l'interface Visual Basic.

Le Projet de Carte Audio Avec un DSP d'Analog Devices

L'avancée du projet

Nous avons considéré très vite qu'il fallait nous répartir les tâches.
Si certains se sont spécialisés plutôt, ils n'en ont pas pour le moins suivi ce qui se passait autre part que dans leur domaine.
Ainsi, nous avons à peu prés chronologiquement :

1ère étape

1) Fait des recherches sur le net : Nous nous sommes informés sur la carte, ce qui avait éventuellement déjà fait aussi.
Nous avons étudié les caractéristiques de notre kit : mémoire, mode d'adressage, imask, …

2ème étape

2) étudié le programme echo.dsp.
Observer comment il fonctionne et essayer de comprendre ce qu'il convient d'utiliser pour notre programme.

3ème étape

3) Créé et testé le premier programme : PCM
Grâce à Ezsde, on a conçu le code et compilé la source (« .dsp »).

4ème étape

4) Commencé à créer l'interface Visual Basic
On a essayé de faire charger l'exécutable généré par l'interface.

5ème étape

5) Implémenter les codes µlaw et Alaw.
On a récupéré les sources de ces codes dans un manuel qui nous a été alors prêté.

Au début on remplissait un emplacement de 16 bits par 8 bits.
Ensuite, grâce à une opération de Shift Logique, nous avons pu remplir la mémoire par des paquets de 8b+ 8b (2 échantillons écrits à chaque fois).

6ème étape

6) On a essayé d'implémenter le code ADPCM..
peine perdue : le fichier réalisé est trop gros pour être compilé.

7ème étape

7) On a mis en place la récupération des données avec écriture d'un fichier Wave.
On a généré un code qui permet de convertir du format 1.15 en format décimal et déterminé les entête à inscrire.
On a eu alors de nombreux problèmes, notamment dans les tailles des données (entête du fichier) et dans la gestion du port série.

8ème étape

8) On a réglé un petit problème qui résidait dans un flag de l'assembleur, changé par inadvertance (on enregistré en 16000hz alors qu'on déclarait du 8000HZ => ralentissement de la voix).
Implémentation d'autres fréquences.

Le Projet de Carte Audio Avec un DSP d'Analog Devices

CHAPITRE 7

Les Problèmes rencontrés.

1er problème

- Notre premier problème a été d'ordre matériel.

En effet, le Pc sur lequel nous devions travailler en salle d'architecture, n'a d'abord pas voulu reconnaître le lecteur CD.
Puis le PC a complètement « planté ».
Il nous a donné beaucoup de fil à retordre….mais finalement nous avons du abandonner et nous avons travaillé sur nos PC propres.

2ème problème

- Un second problème a été de récupérer les sources des lois de compression.

Finalement, elles nous ont été fournies car nous n'avions pu les récupérer sur Internet (donc source : professeur tuteur).
Mais, si nous avons pu assez facilement gérer les lois A et µ (quoique avec un petit problème dans la gestion des décalages de bit, problème mineur vite résolu), il n'en a pas été de même pour le codage ADPCM (problème irrésolu).En effet ,le code source fourni ne fonctionne pas (à la compilation, erreur dans les registres, qui passe plus ou moins mais qui finalement lorsque l'on rajoute la partie enregistrement des données, génère un code source de programme trop lourd à gérer par le compilateur (fichier source DSP trop gros). Bien que l'on ai essayé de bien trier la routine ADPCM pour ne la réduire qu'a sa partie utilisée (pas de « expand » : suppression de la routine de décompression fournie dans le code source), on n'a pas pu réduire plus le code qu'au 137k (incompilable).

3ème problème

- Approcher les fonctionnalité de Visual Basic (dans notre groupe, personne ne connaissait VB –version6-).

Cela nous a donné somme toute pas mal de problèmes, assez épars.
(Mais bon, on ne gagne rien sans «lutte » parfois).
Par exemple, dans la gestion des paramètres du port série : options que l'on avait oubliées de sélectionner.

4ème problème

- Charger les données.

Le « $UD » se doit d'être suivi de l'adresse de début (hi start et lo start) puis de la longueur à récupérer en données (hi lenght et lo lenght).
En l'occurrence, nous avons en start : 0000 et en length : 15000 soit 3A98
Mais ces données doivent alors être transformées en caractères ascii. (ce que nous n'avions pas remarqué tout de suite)

5ème problème

- Créer un algorithme pour transformer dans le format 1.15.

On a eu quelques problèmes de compréhension dans la gestion des données et encore ce fameux problème de la conversion du code ascii.

problèmes mineurs

- Quelques problèmes avec les sensibilités des micros.
- Quelques problèmes avec la gestion des PC (pas de disponibilité à l'école) et nombreux problèmes avec VB et Windows (nécessité des rebooter de nombreuses fois le PC).

Le Projet de Carte Audio Avec un DSP d'Analog Devices

Le fonctionnement général de notre application

Résumé du fonctionnement apparent

Nous avons réaliser une interface en Visual Basic qui agit tel que :

1) dès que l'on a pu précisé des paramètres d'enregistrement, charge le bon programme correspondant. Il propose :
   a) le mono comme le stéréo.
   b) en 8 bits ou 16 bits.
   c) en 8000 hz, 11025 hz ou 22050hz.
   d) le mode PCM (Pulse Code Modulation) qui est le mode « normal » ou la compression en loi µ / loi A (on impose alors un échantillonnage en 16 bits avec résultat sous 8 bits et en fréquence de 8000 Hz).

2) On peut alors lancer l'enregistrement et la récupération des échantillons. (elle succède automatiquement).

3) On visualise alors une courbe (fenêtre graphique) des données saisies.

4) On enregistre dans un fichier Wave qui peut être lu par tous les lecteurs Wave standard.

5) Une fois le fichier écrit, on peut charger le fichier généré pour l'écouter et le visualiser (courbe réelle dans la fenêtre graphique).

Quelques petits détails …

Le points sur lesquels nous nous sommes intéressés ont du être :
le paramétrage du port série, l'écriture des routines en assembleur, et la création des fichiers au format wave sont entièrement pris en charge.

a) Gestion du port série.
Elle concerne plusieurs aspects, dont voici la liste:
premièrement, l'affectation d'un numéro de port valide; si l'utilisateur se trompe en le sélectionnant, le logiciel
le lui signale et attend une modification de sa part.
La vitesse du port peut être paramétrée de 1200 à 115200 bps.
Seule la vitesse de 9600 bps nous intéresse, puisque c'est celle à laquelle la carte communique avec le port série.

b) Ecriture, assemblage et édition des liens.
La présence d'un éditeur de texte qui assemble et réalise l'édition de liens des fichiers sources en assembleur facilite le développement des programmes destinés à être chargés dans la mémoire de la carte.

c) Création et manipulation des fichiers au format Wave.

Les fichiers au format Wave sont d'usage très fréquent dans le monde du PC; ils autorisent des fréquences d'enregistrement très variées, de 8000 à 44100 Hertz, aussi bien en 8 qu'en 16 bits.

Le Projet de Carte Audio Avec un DSP d'Analog Devices

Assembleur AD2181
d'Analog Device

- Structure globale d'un programme en assembleur.
- Approche de l'assembleur en analysant echo.dsp
- Réalisation de nos routines en assembleur.

Structure d'un programme assembleur pour ADSP2181

Structure d'un programme assembleur pour ADSP 2181.

Un programme développé pour le kit Ez-Lite doit toujours avoir la structure suivante :

Nom du programme.

Déclaration de constantes
Adresses mémoire des registres de contrôle.
Constantes du programme.

Déclaration des variables
Initialisations / Section principale
Initialisation de registres.
Remplissage des vecteurs d'interruption.
Initialisation des registres internes du ADSP-2181.
Initialisation des modes d'accès à la mémoire.
Initialisation de l'horloge (si on utilise l'interruption TIMER).
Initialisation du port sériel SPORT0.
Initialisation du CODEC AD1847.

Programme utilisateur
Initialisations.
Masque d'interruption.
Point de stationnement.

Routines d'interruptions.
Gestionnaire d'interruption du SPORT0(TX).
Gestionnaire d'interruption du SPORT0(RX).
Routines de traitement utilisateur.
Gestionnaire d'interruption IRQE.
Gestionnaire d'interruption TIMER.

Fin du programme.

Pour débuter avec l'assembleur de l'AD2181, nous nous sommes basés sur le programme echo.dsp qui s'avérait être le plus simple de tous ceux qui étaient fournis.

Approche de l'assembleur en analysant echo.dsp

Comme convenu, nous analysons ce programme afin de déterminer les parties qui nous serait utiles pour plus tard.

- Déclaration des variables : nous utiliserons par exemple des buffers circulaires. Le fait que les buffers soient circulaires permet de réécrire les nouvelles informations sur les anciennes.
- Initialisation de la vitesse du CODEC avec le tableau init_cmds et du port le reliant au DSP.

Voici par exemple la 8 ème valeur du tableau init_cmds dans laquelle on lit la configuration du CODEC. Par défaut, celui-ci est réglé à 8 Khz de vitesse d'échantillonnage, formaté en 16 bits signed linear PCM et en stéréo.

```
0xc850,     { INITIALISATION DE LA VITESSE DU CODEC
                soit : 1100 1000 0101 0000
                  data format register
                  b7  : res
                  b5-6: 0=8-bit unsigned linear PCM
                        1=8-bit u-law companded
                        2=16-bit signed linear PCM
                        3=8-bit A-law companded
                  b4  : 0=mono, 1=stereo
                  b0-3: 0=  8.
                        1=  5.5125
                        2= 16.
                        3= 11.025
                        4= 27.42857
                        5= 18.9
                        6= 32.
                        7= 22.05
                        8=   .
                        9= 37.8
                        a=   .
                        b= 44.1
                        c= 48.
                        d= 33.075
                        e=  9.6
                        f=  6.615
                  (b0) : 0=XTAL1 24.576 MHz; 1=XTAL2 16.9344 MHz
               }
```

- La table des vecteurs d'interruption
- Initialisation du DSP lui même et du CODEC
  Le timer, le port série qui le relie au CODEC (sport0), sa gestion de mémoire.
- La boucle principale gérant les différents sauts aux étiquettes suivant l'application à produire.

Une partie particulièrement intéressante de echo.dsp est sa boucle principale de commande (command loop) qui est exécutée en permanence et qui permet d'établir un menu de commandes adressées au DSP par le PC. Elle permet notamment de pouvoir sortir du programme « proprement ».

L'écoute des requêtes utilisateurs :

Cette boucle principale est composé par l'attente d'un ordre du PC transféré par le RS232 (sport1). On scanne le port en utilisant le fonction « get_char » chargée en mémoire par le monitor. On dispose du pointeur PTR_TO_GET_CHAR que l'on appelle si besoin. Tout ceci est réalisé dans la boucle appelée again :

```
again:      { any thing from host ?}
     ar = dm (CHAR_WAITING_FLAG);       {attente d'un caractère}
     none = pass ar;
     if ne jump again;

          { has something }
     i4 = dm (PTR_TO_GET_CHAR);         {l'ordinateur vient de
                                         transmettre une valeur
                                         au DSP via le RS232}
     call (i4);
     if lt jump again; { time out }

     ay0 = 78;   { 'N' }
     none = ax1 - ay0;
     if eq jump move_disp;          {un N fait un saut à move_disp}

     ay0 = 90;   { 'Z' }
     none = ax1 - ay0;
     if eq rts;                     {un Z fait retourner au monitor}
     ar = ax1 + 1;
     ax1 = ar;
     iSend:  i4 = dm (PTR_TO_OUT_CHAR);         {Si une autre
                                                 valeur est envoyée,
                                         on la renvoie à l'ordinateur}
     call (i4);
```

Remarque : On pourra remarquer que nous nous sommes servi de cette forme pour implémenter nos programmes d'enregistrement.
Ainsi nous pouvons indiquer à notre routine si nous voulons faire du mono ou du stéréo.

Nous avons alors utilisé des caractères numériques pour indiquer les commandes.

Les pointeurs sur des fonctions comme PTR_TO_GET_CHAR sont définies dans le fichier system.k.
Si le caractère reçu ne correspond pas à une commande prévue, on utilise la fonction « Out_Char » pour renvoyer le caractère via RS232.

Algorithme de réalisation de l'écho :
- Les échantillons numérisés par le codec sont stockés dans le buffer circulaire écho[l].
- On diminue l'amplitude de chaque échantillon en lui appliquant un gain.
- On décale (retarde) ces échantillons dans le temps et on met à jour le buffer écho[l] en sommant le signal d'origine et le signal retardé atténué.

Nous avons juste adapté cette routine pour nos besoins et supprimé les algorithmes de filtrage (fir), modifié les commandes, supprimé l'écho.

Réalisation de nos routines en assembleur

1) Routine PCM

C'est la routine de base à l'enregistrement de la voie.
Elle enregistre en 8000 hz, 11025 hz ou 22050 hz.
Elle échantillonne en 8 ou 16 bits.
Elle peut enregistrer du mono comme du stéréo.

Pour cela, après avoir charger le programme en Program Memory de la carte (PM), on envoie un 0 pour enregistrer en mono et un 2 pour enregistrer en stéréo. (Utilité du ms_flag : indique lorsque l'on enregistre, si c'est en mono ou en stéréo).

Lorsque le programme est chargé, il attend une de ses 2 commandes pour pouvoir enregistrer. (Utilité du rec_flag : indique que l'on enregistre - à moins que le buffer ne soit plein).

On enregistre dans la Data Memory avec la commande :
dm (« registre de pointeur sur le buffer», « registre du déplacement »).
registre de pointeur sur le buffer : on a utilisé i0.
registre du déplacement : on a utilisé m1.

Sinon, tant que le programme est chargé en mémoire, le codec saisit des échantillons et les ressort sur les speakers. (talkthru)

Pour stocker les valeurs, on a déclaré un buffer de 15000 emplacements. (emplacements de 16 bits).
On l'a déclaré avec ABS=0 pour le forcer à prendre pour adresse de début, l'adresse 0.

Lors d'un enregistrement en 8 bits, on décale les données (label « decale ») pour une optimisation de la place.

2) Routine Alaw

On se sert de la routine PCM en base.

La différence réside dans le fait qu'on a été obliger de modifier certains registres pour pouvoir réaliser la compression.

Ici, la routine va compresser les échantillons de 16 bits (envoyé par le codec) en 8 bits.
On a donc un gain de place : en effet, en réalisant un shift logique de 8 bit et en compulsant, on parvient à stocker un 1er échantillon en valeur Hi et un 2ème échantillon en valeur Lo.

Juste après avoir reçu l'échantillon, on appelle la procédure de compression qui nous donne immédiatement la valeur « compressée ».

3) Routine µlaw.

De même, la compression en loi µ se comporte.
On a juste placé une sous routine différente pour la compression, mais le corps du programme est le même que celui de PCM et plus encore ressemblant à celui de A law.

NB : ADPCM
Pour la compression ADPCM, nous rappelons que nous n'avons pas pu l'implémenter (fichier de compression trop volumineurx et source du codage ADPCM comportant une erreur de registre –comme le précise le compilateur.

Les principaux labels (présents dans tous les programmes) :

Again : scrute sans cesse les port sérié en quête des renseignements envoyés.

Avec « 0 » envoyé, on fait un enregistrement en mono (flags mis à jour : ms_flag =0 et rec_flag à 1). On débute alors l'enregistrement (saut au label start_record_mono).

Avec « 2 » envoyé, on fait un enregistrement en stéréo (flags mis à jour : ms_flag =1 et rec_flag à 1). On débute alors l'enregistrement. (saut au label start_record_stereo).

Input samples : récupère les échantillons du codec.

Talkthru : envoie les échantillons reçus du codec vers les speakers.

Record : effectue l'enregistrement en mémoire via d'autre saut à différents labels (voir les 2 labels suivant).En fait ce label contient le test pour savoir si l'on est en fin de mémoire.

Stereo : effectue l'enregistrement en mémoire en alternant les 2 canaux.

Mono : effectue l'enregistrement en mémoire.

Zut : fin de l'enregistrement

Si l'on se trouve en 8 bits en récupération (compression aussi), on a le label :
Decale : effectue un décalage de la valeur tous les 2 échantillonnages saisis.
La varible p_val stocke une première valeur. La valeur suivante est décalée :
- dans le cas de la compression , on a un décalage vers le haut.
(par une instruction de Logical Shift : Lshift si by 8 (lo) : on récupère le résultat dans sr0).=> la donnée intéressante se trouve dans la partie basse.
- Dans le cas du PCM 8 bits : on fait alors un décalge vers le bas.
(par une instruction de Logical Shift : Lshift si by -8 (lo) : on récupère le résultat dans sr0).=> la donnée intéressante se trouve dans la partie haute.

Le Projet de Carte Audio Avec un DSP d'Analog Devices

CHAPITRE 10

Interfacage avec Visual Basic

Page de présentation du logiciel

Page de présentation du logiciel

Au lancement du logiciel, on arrive sur une fenêtre qui possède trois menus qui correspondent aux fonctionnalités proposées :

- Menu Fichier :

Il permet le lancement de la fenêtre du Lecteur – enregistreur de fichiers wave (Lecteur Wave : voir plus loin). Il permet aussi de quitter l'application (quitter).

- Menu Carte DSP :

Il permet de se servir directement de la carte. Pour ce faire, il permet en premier lieu de fermer d'ouvrir ou de configurer le port série (voir plus loin pour la configuration). Ensuite, on peut envoyer un beep ou une commande à la carte, charger un programme assembleur. On remarquera que l'état du port série est rappelé en permanence dans le coin inférieur gauche de la fenêtre.

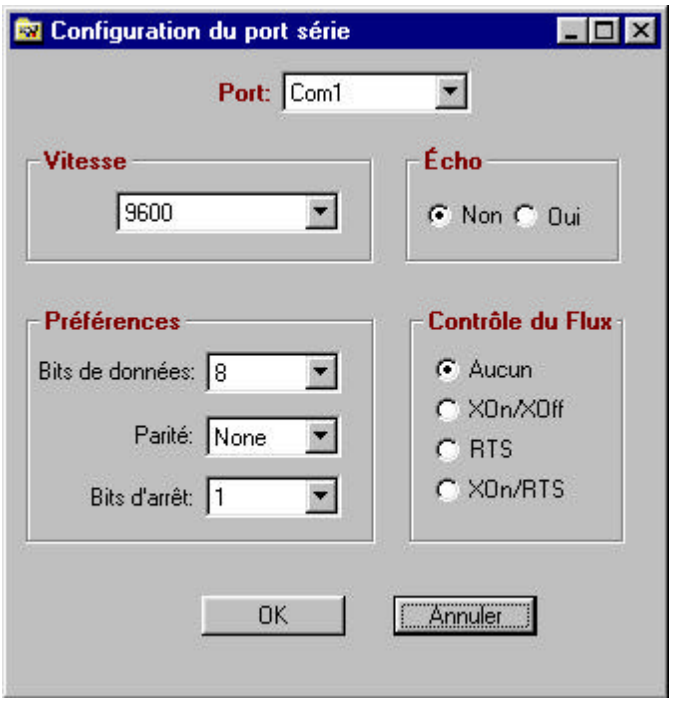
Important : Pour lancer le lecteur Wave, il faut que le port série soit fermé.

- Menu Aide :

Il contient les références des auteurs et permet un appel aux informations sur le système utilisé.

Fenêtre de configuration du port série.

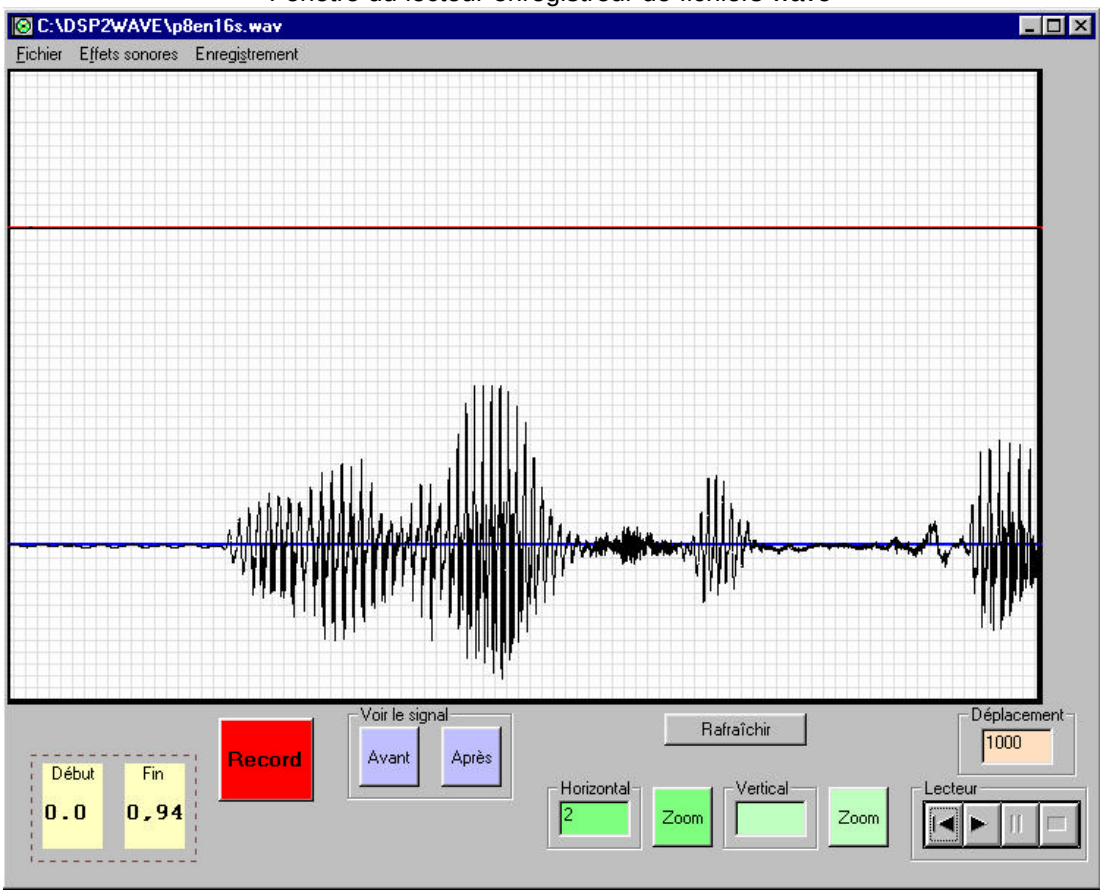
Fenêtre de configuration du port série



Accessible depuis la page de la page de présentation, cette fenêtre permet de configurer toutes les options du port série. Par défaut, les valeurs sont celles qui permettent un bon fonctionnement de la carte DSP. Seul le numéro du port peut être à changer. Cependant, grâce à cette fenêtre et à la précédente, on peut se servir d'une autre carte qui se branche sur un port série.

Fenêtre du lecteur de fichiers wave.

Fenêtre du lecteur enregistreur de fichiers wave



Il s'agit de la fenêtre qui correspond le plus à ce que nous voulions. Elle comporte de nombreux boutons et de nombreux menus :

➢ Menu Fichier :

Il permet d'ouvrir un fichier wave, d'enregistrer le fichier courant ou de revenir à la fenêtre de présentation (Quitter).

➢ Menu Effets Sonores :

Il permet de modifier le son chargé : on peut alors inverser le son, l'amplifier ou le réduire.

Attention : Les modifications ainsi effectuées ne sont prises en compte que pour l'affichage. Pour entendre les modifications, il faut enregistrer le fichier sous un autre nom, puis ouvrir ce nouveau fichier.

➢ Menu Enregistrement :

Il permet de basculer sur la fenêtre d'options pour l'enregistrement (voir plus loin).

Les boutons

Voici une description des boutons présents en bas de fenêtre de gauche à droite :

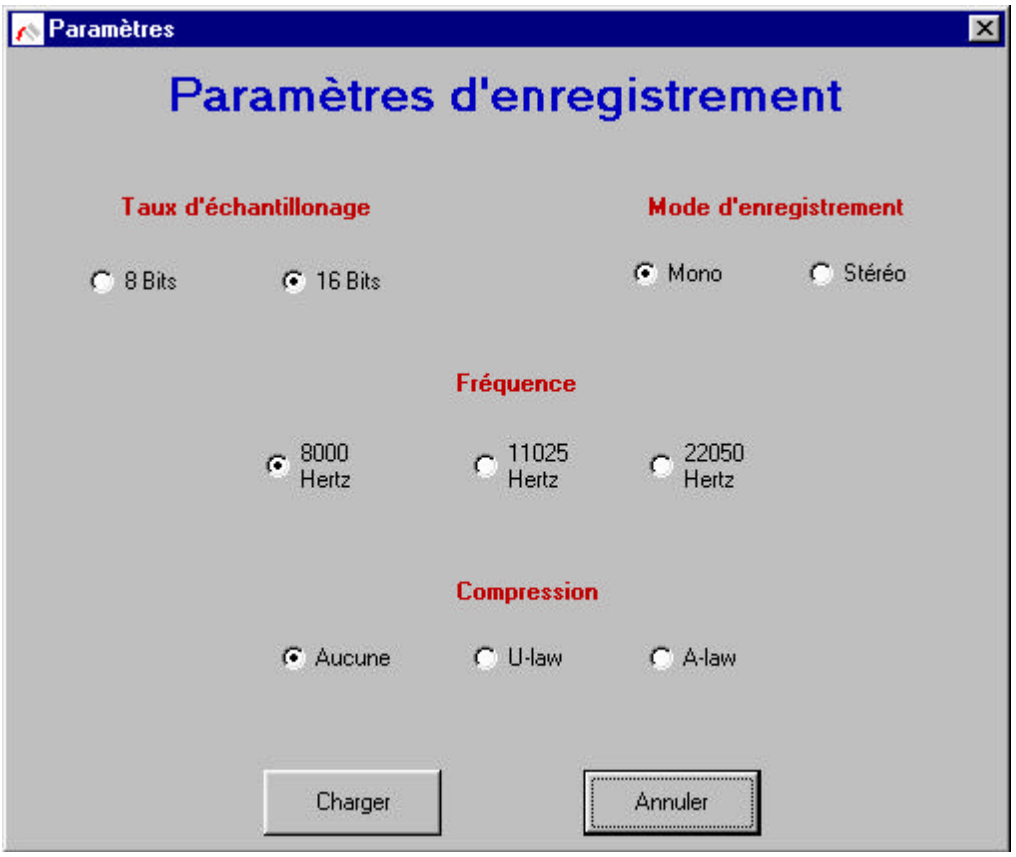
Début et Fin : indiquent la date du début et de la fin de l'enregistrement.

Record : permet de lancer l'enregistrement. Il faut commencer à parler dans le micro dès qu'on appuie sur ce bouton. Le début de l'enregistrement est caractérisé par le changement de la souris (ainsi que par l'allumage de la diode de la carte DSP). A la fin, la souris reprend son aspect normal.
La carte envoie alors un beep indiquant que le programme en assembleur s'arrête. La souris change à nouveau de forme et les données sont immédiatement chargées dans le logiciel. Quand la souris reprend son aspect initial, le son est près à être enregistré. Une fenêtre d'enregistrement s'ouvre alors pour demander le nom du fichier à créer. Une fois le nom entré, la souris change d'aspect pour signaler l'écriture du fichier. Enfin, la dernière fenêtre se ferme, la souris reprend son aspect normal et une courbe est affichée à l'écran.
Attention : cette courbe ne représente pas forcément le son enregistré. Elle permet simplement de signaler que le travaille a été correctement effectué. Il faut ouvrir le fichier pour visualiser la vraie courbe et entendre le son produit.

Voir le signal : Ces deux boutons permettent de se déplacer en avant ou en arrière du signal. Le pas utilisé est celui du bouton déplacement. Le pas par défaut est de 1000.

Zoom horizontal et vertical : Une fois une valeur entrée dans les cases correspondantes, un clic sur le bouton zoom permet d'effectuer un zoom sur la courbe de la valeur voulue.
Rafraîchir : Ce bouton permet de réafficher la courbe dans le cas où elle aurait été effacée (par exemple si vous ouvrez une fenêtre).
Lecteur : Les boutons sont ceux habituels de tout lecteur wave : un bouton pour revenir au début de l'enregistrement, un autre pour lire, un troisième pour faire une pause et un dernier pour stopper la lecture.

La fenêtre des paramètres



C'est cette fenêtre qui permet d'utiliser toutes les méthodes d'enregistrement implémentées.
Elle permet de basculer entre les modes 8bits et 16bits, mono ou stéréo, 8000Hz, 11025 Hz ou 22050Hz, et enfin PCM, Ulaw et Alaw.

Toutes les combinaisons ne sont pas possibles. C'est pourquoi le logiciel ne vous permettra pas de cliquer dans certaines cases si d'autres sont activéees.
Par exemple, si la case Ulaw est active, il n'est pas possible de demander du 16bits, puisqu'en fait Ulaw fait déjà du 16bits caché (par la compression).

Le bouton Charger permet de charger le programme assembleur correspondant à vos options dans la carte DSP. La diode s'éteint alors.

Remarque : une fois un programme chargé, il n'est possible d'en charger un autre directement sans enregistrer entre temps. Si vous voulez quand même en charger un autre, il faut revenir dans la fenêtre de présentation, ouvrir le port série, et envoyer un « 9 » pour arrêter le programme. Vous pourrez alors revenir au lecteur wave et charger les bonnes options.

Conclusion

Notre projet d'architecture des ordinateurs nous a été très bénéfique.
En effet, programmer un DSP fait appel à un large champ de connaissances techniques : électronique, informatique (langage assembleur), traitement du signal et comme ce fut notre cas, il a fallu s'adapter au VB. Ce qui nous a été d'une bonne part dans notre enseignement. .

Ce projet nous a permis de découvrir les DSP et leur particularité par rapport aux processeurs classiques. La difficulté réside dans la programmation en assembleur et en VB. Mais grâce à nos travaux, nous avons atteint nos objectifs.

Ce projet nous a permis de travailler en groupe et de gérer les tâches et le temps. Notions qu'un bon ingénieur se doit de savoir maîtriser.

Pour notre part (notre binôme), nous avons pu équitablement et très pratiquement séparer le projet en 2 section par les 2 langages du projet et en sommes assez satisfaits.

Nous pouvons donc dresser un bilan très positif de l'ensemble.

Remerciements : Toutes les personnes qui ont pu nous aider. Dont notamment notre tuteur de projet, M. OLIVETTO qui nous a bien aidé (Ezsde, les sources des compressions…) ainsi que tous les autres binômes du groupe « Carte Audio DSP ».

On remercie aussi tous les auteurs des sites sur le net, qui nous ont documentés. (ils sont très nombreux).

On remercie également le service de support des produits d'Analog Devices sur le net.