TP3- FILTRAGE RIF TEMPS REEL (ASM)

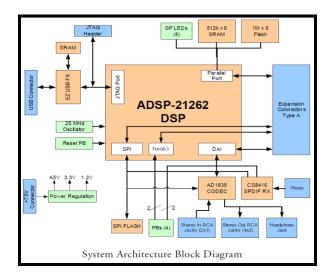
Assembleur, avec E/S

Objectifs du TP	1
Exemple "Talk-Thru"	2
Travail à effectuer	7

Objectifs du TP

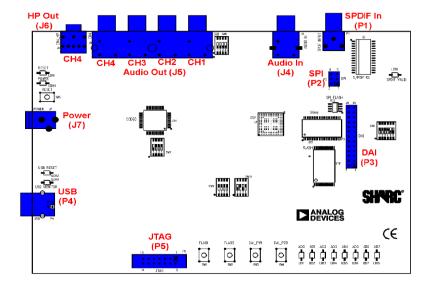
Pour réaliser ce TP, nous avons besoin d'un oscilloscope, d'un générateur de signaux analogiques et d'un module ADSP-21262 EZ-Kit Lite. Ce kit de développement, construit autour du SHARC ADSP-21262, intègre entre autre un CODEC. Un CODEC (COdeur -DECodeur) est une unité périphérique qui échantillonne un signal continu et le transforme en une séquence de données binaires. Pour ce faire, le CODEC possède au moins un convertisseur Analogique/Numérique (A/N).

Le signal numérisé est ensuite traité par le processeur SHARC avant d'être renvoyé vers le CODEC. Le signal numérique est alors converti en signal analogique grâce à un convertisseur Numérique/Analogique (N/A). Le kit d'évaluation ADSP-21262 EZ-Kit Lite est donc très adapté au développement d'applications dédiées au traitement numérique du signal en utilisant des signaux réels (de type audio par exemple).





Dans le cadre de ce TP, nous allons voir comment utiliser le SHARC pour l'acquisition du signal numérique (envoyé par le CODEC), et effectuer le traitement sur le SHARC et renvoyer le résultat au CODEC.



Exemple "Talk-Thru"

Fichiers sources contenus dans le dossier rifes_ASM

21262_IVT.asm : Fichiers des vecteurs d'interruptions ➤ ad1835.h : Registres du CODEC AD1835A

init1835viaSPI.asm : Routines d'initialisation du CODEC en utilisant le protocole SPI

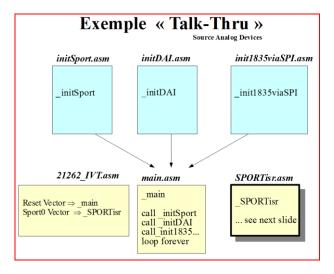
initDAI.asm : Routage des signaux de l'unité SRU

initSPORT.asm : Routines permettant d'initialiser le port série (SPORT 0)

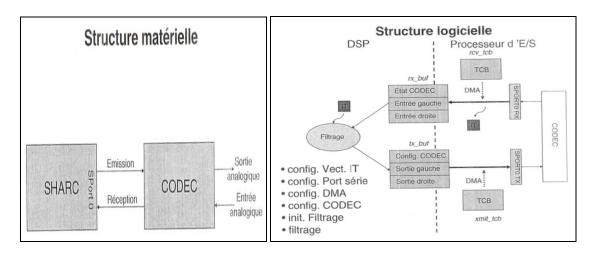
> SPORTisr.asm : Routine de traitement de l'IT du port série (acquisition des échantillons)

main.asm : Programme principal

ADSP-21262 ASM.LDF: Fichier architecture du processeur ADSP-21262



Ces fichiers, écrits en assembleur, correspondent à un programme qui lit des échantillons provenant d'une source analogique (Audio In, J4) et les restitue directement sur les sorties analogiques (Audio Out et HP Out, J6). Les conversions analogiques/numériques et numériques/analogiques sont assurées par un CODEC (AD1835A d'Analog Dvices). Ce dernier est connecté au SHARC par l'intermédiaire d'un port série multicanaux géré par DMA.



La présence d'un échantillon nouveau dans le registre de donnée du port série déclenche une **interruption**, nommée **SPOI**. On vous demandera, le moment venu, de configurer cette interruption pour provoquer l'exécution de la routine **_talkThroughISR** contenue dans le fichier **SPORTisr.asm**

```
_talkThroughISR:

r10 = dm(RXSP0A); // Lecture de l'ADC de l'échantillon gauche

dm(TXSP2A) = r10; // Écriture dans DAC3

dm(TXSP2B) = r10; // Écriture dans DAC4

rti;
_talkThroughISR.end:
```

L'objectif de ce TP est de vous familiariser avec la gestion des interruptions en assembleur. Pour effectuer vos tests, vous avez besoin d'un générateur de signaux analogiques et d'un oscilloscope pour observer les signaux à traiter et le résultat du traitement. <u>Une analyse approfondie des fichiers sources fournis est indispensable pour comprendre la gestion des interruptions en assembleur</u>. <u>Des corrections sont à effectuer afin d'obtenir un code compilable.</u>

Remarque importante

Avant de connecter le générateur de signaux analogiques au kit, vérifier soigneusement que l'amplitude du signal de sortie ne dépasse pas 1Volt crête à crête. Une tension supérieure risquerait d'endommager les étages analogiques du kit de développement.

Travail à effectuer

- T1. Mettre en place de l'application "talkthru"
 - I. Après vérification de l'amplitude, connecter la sortie d'un générateur de signaux analogiques au connecteur J4 (Audio In) du kit et la sortie J6 (HP Out) du kit à une des voies de l'oscilloscope.
 - II. En se référant, si nécessaire, au scénario exécuté lors du premier TP, **TP1_rif**, créer un nouveau projet **VisualDSP++** nommé **rifes_ASM**. Ajouter au projet les fichiers présents dans le dossier fourni **rifes_ASM**.
 - III. Analyser le fichier architecture " *ADSP-21262_ASM.LDF* ". Vérifier qu'il est bien adapté pour un projet Assembleur. Qu'en est-il dans le cas d'un projet entièrement écrit en langage C ?
 - IV. Compiler le projet et corriger les erreurs signalées ainsi que les avertissements (warnings). Après correction des erreurs, vérifier que l'exécution n'est pas concluante (aucun signal en sortie sur les deux voies). Pourquoi?
 - **V.** Mettez un point d'arrêt sur la routine **_talkThroughISR** et vérifier qu'elle n'est jamais appelée. Corriger le problème et vérifier que l'acquisition et la restitution du signal d'entrée sont bien réalisées.
 - 🖔 Quelle est la fréquence de coupure du signal de sortie ? Justifier sa valeur.
 - ♥ Sur combien de bits sont codés les échantillons reçus du CODEC ?
- **T2.** Modifier le code de la routine _talkThroughISR pour provoquer une atténuation du signal de sortie.
- **T3.** Tester le filtrage RIF du TP1 en intégrant d'abord la version séquentielle (fichier *rif_SEQ.asm* contenu dans le dossier *RIF*) puis la version optimisée (fichier *rif_OPT.asm* contenu dans le dossier *RIF*). Pour appeler individuellement les deux fonctions *rif_SEQ.asm* et *rif_OPT.asm*, inspirez-vous de l'exemple du TP1 (voir le fichier *riftest.asm* contenant les instructions assembleur ci-dessous).

```
/* Echantillon d'entrée dans F0, sortie renvoyée dans F0 */
call rif (DB);
r1 = TAPS;
r0 = dm(i1,1);
```

<u>Remarque</u>: Pensez à vérifier que les buffers circulaires sont activés.

🗳 Assurez-vous, en validant ce travail auprès du professeur, que le résultat du filtre est correct.