# TP7 : Prise en main du système de développement EVM OMAP137

## 1 Présentation

### Connaissances ciblées

- Réponse fréquentielle d'un filtre.
- Propriétés des systèmes linéaires.
- Théorème de l'échantillonnage.
- Filtrage FIR en temps réel.

### Capacités ciblées

- Prendre en main un IDE et les outils de développement pour DSP.
- Comprendre l'architecture d'une carte dédiée au traitement numérique du signal et programmer des blocs fonctionnels.
- Implanter un filtre FIR en temps réel, en mode échantillon par échantillons en utilisant une ligne à retard.
- Implanter un filtre FIR en temps réel, à l'aide d'un buffer circulaire.
- Vérifier expérimentalement la réponse fréquentielle d'un filtre.

## 2 Préparation

1. Prendre connaissance de l'annexe de ce TP et revoir les pages 110-126 du cours.
2. Quels sont les principaux constituants du Système de Développement OMAP L-137.

## 3 Manipulations

### 3.1 TP7_projet1 (1h30)

Programme FIR avec lecture d'un signal depuis un fichier.

1. Commencer par lire le paragraphe 6 Lancement d'un projet sous CCS.
   Récupérer le zip du TP7 depuis Moodle. Déposer votre archive sur le disque local et non sur le disque distant.
   Lancer Code Composer Studio, spécifier un Workspace sur le disque local (**n'utiliser aucun espace dans les chemins de vos répertoires**).
   Importer les projets de l'archive par le menu : **Project** \ **Import CCS project...**\ **Select archive file** \ **Select all** avec validation de l'option **"Copy project in Workspace"**.
   Sélectionner dans le **Project Explorer** à gauche de la fenêtre d'édition le projet **TP7_projet1** pour le rendre actif.
2. Sans lancer la compilation, ni l'exécution du projet, analyser les programmes `main.c` , `utility.c` et `fir_basic.c` et repérer les différentes fonctions qui y sont implémentées.
   Analyser les fichiers entête `filter_coefficients.h` et `filter_algorithme.h`.
   L'objectif est essentiellement de repérer les différentes fonctions, et comprendre ce qu'elles font.

3. En vous appuyant sur votre analyse, tracer un schéma fonctionnel montrant les différentes étapes de l'exécution de ce projet (fonctions, données échangées, etc.).

4. Quel est le nom du fichier qui contient le signal à traiter et qui est lu par le programme de filtrage. Ce fichier est contenu dans l'archive **TP7_DSP.zip**. Adapter le chemin spécifié par défaut dans le programme.

5. Compiler le projet et lancer le débogueur.
   Exécuter alors le projet en pas à pas (touche F5 instruction par instruction) :

   a. Vérifier à chaque pas les valeurs des variables (indice, nom de fichier, valeur de signal, etc.).

   b. Aller dans la boucle de lecture du fichier, et vérifier que les valeurs écrites en mémoires sont bien les valeurs du fichier contenant le signal. Vous avez la possibilité de visualiser la mémoire où sont stockées les variables.

   c. Vérifier la cohérence de l'affichage obtenu avec ce qui est attendu.

   d. Tracer dans CCS les signaux d'entrées et de sorties et commenter. Pour cela choisir 'Tool/Graph' et donner le type de données et l'adresse de début du signal. Il faut visualiser la mémoire avant que celle-ci ne soit libérée (avant la fin du programme).

   e. Vous pouvez afficher également les signaux d'entrée et de sortie en copiant le contenu de **s.txt** et **s_out.txt** dans l'environnement de Matlab.

   Cette partie est assez longue car vous devez exploiter ici les différentes possibilités offertes par le débogueur.
   L'outil `http://babbage.cs.qc.cuny.edu/IEEE-754/index.xhtml` permet de convertir simplement un code en virgule flottante en sa valeur décimale.
   Penser à placer des points d'arrêt pour sauter les boucles et aller plus vite.

6. Vérifier le bon comportement du filtre à l'aide de Matlab et de Filter Designer (interface remplaçant FDATool, commande **filterDesigner** ). Entrer les coefficients du filtre et visualiser l'amplitude de la réponse fréquentielle en échelle linéaire. Vous devez retrouver la modification d'amplitude apportée au signal d'entrée.
   Conseil : pour retrouver la fréquence normalisée du signal de votre fichier, déterminer le nombre entier de points dans une période de la sinusoïde (soit N ce nombre). La pulsation normalisée sous Matlab sera $\omega = \frac{2}{N}.\ \pi = 2f_{norm}.\pi \quad \text{avec} \quad f_{norm} = \frac{1}{N}$.

7. Changer le fichier du signal d'entrée dans votre programme, avec l'un des fichiers fournis sous Moodle dans la rubrique **Signaux_sinusoidaux_f_norm**.
   Attention, la fréquence apparaissant dans le nom du fichier est la valeur de la fréquence normalisée (que vous pouvez retrouver par la méthode indiquée ci-dessus).
   Vérifier la cohérence de vos résultats en comparant la modification d'amplitude du signal de sortie avec la réponse fréquentielle du filtre.

8. Remplacer le filtre utilisé par défaut par les autres filtres proposés, exécuter de nouveau le programme et commenter les résultats.
   Refaire les mêmes vérifications avec Matlab, tester avec plusieurs fréquences normalisées.

**Dans la suite du TP, nous allons travailler en mode interruption. Ce mode est utilisé pour l'acquisition du signal d'entrée. La conséquence est que nous ne pouvons plus utiliser le débogueur, car lui-même utilise les interruptions. En conséquence, s'il y a besoin d'utiliser Debug, alors il faut placer l'algo à tester dans le projet C6000_filtering_test car celui-ci lit le signal d'entrée à partir d'un fichier et n'utilise pas le mode interruption.**

## 3.2 TP7_projet2 (40 mn)

Programme qui lit un échantillon de l'entrée audio et l'envoie vers la sortie audio en mode interruption.

1. Prendre le temps de jeter un oeil sur les fichiers `evmomapl137_aic3106.c`, `evmomapl137_mcasp.c et evmomapl137_line.c`, afin de voir à quoi ressemblent les fonctions d'initialisation des circuits périphériques qui accompagnent le DSP.
2. Analyser les programmes `main.c, loop_lineInOut.c`.
3. Faire un schéma fonctionnel montrant les différentes étapes de l'exécution de ce programme.
4. Une carte d'interface AREF Electronique a été spécialement conçue pour faciliter les connexions entre les entrées/sorties audio stéreo de la carte EVMOMAPL137 et les différents instruments ou appareils nécessaires aux manipulations.
   Vous pouvez par exemple accéder séparément aux voies gauche et droite en entrée comme en sortie. De nombreux connecteurs permettent de brancher le générateur, l'oscillo, la carte son de votre PC ou la sortie de votre téléphone en entrée de la carte DSP et de récupérer sur l'oscillo ou sur un casque la sortie de la carte DSP après traitement.
   La carte AREF Electronique dispose aussi d'un étage de filtrage analogique nécessaire à la suppression du bruit lorsque l'on traite des signaux de faible amplitude.

   Les connexions sont déjà faites. Le générateur est utilisé pour générer le signal d'entrée de la carte, et l'oscilloscope pour visualiser les signaux d'entrée et de sortie.
5. Compiler le programme, le charger dans le DSP et le lancer. Commenter.
6. Comment vérifier l'existence ou non d'un filtre anti-repliement ? Faire la vérification expérimentale.
7. En modifiant un paramètre d'initialisation de l'AIC dans le `main.c`, faire varier la fréquence d'échantillonnage de l'AIC, puis exécuter le programme pour différents signaux d'entrée. Commenter.

## 3.3 TP7_projet3 (40 mn)

Programme qui exécute un FIR sur un signal réel en mode interruption.

1. Analyser les programmes contenus dans le projet `main.c, fir_basic.c, fir_coefficients.h`.
2. Modifier une ligne du code afin de bien réaliser un filtrage.
3. Lancer le programme et analyser le bon comportement du filtre. Confronter la réponse fréquentielle obtenue sous Matlab et la réponse en fréquence du filtre obtenue expérimentalement. Attention, les variations d'amplitude ne seront étudiées que relativement au signal de sortie, car le gain initial entre l'entrée et la sortie résulte de la succession de nombreux étages (interface audio de la carte EVMOMAPL137 + carte AREF Electronique)
   Rappeler quel type de signal il faut placer à l'entrée du filtre pour en déterminer la réponse fréquentielle et justifier votre réponse.
4. Faire cette vérification à partir des signaux temporels mais aussi à partir de leur spectre (Menu FFT de l'oscillo + utilisation des curseurs).
5. Observer les effets du filtre pour différentes formes de signal en entrée, en temps et en fréquence. Interpréter vos observations.
6. Refaire l'ensemble de la manipulation pour différents filtres, en vous référant toujours à Matlab pour des comparaisons quantitatives et non seulement qualitatives.

## 3.4 TP7_projet4

Programme FIR avec adressage circulaire (40 mn)

1. Analyser les programmes contenus dans le projet.
2. Compléter le programme `fir_circular.c` pour implémenter la gestion d'un buffer circulaire.
3. Comment peut-on vérifier que le programme fonctionne correctement ? Faire la vérification expérimentale.

# 4 Description du EVMOMPAP L137

Les grandes lignes de ce système ont déjà été présentées en cours p.110 à p.126.
Nous revenons cependant sur quelques éléments. Le synoptique de la carte montre l'existence de l'OMAP L137 qui inclut le DSP et de l'AIC3106, qui est un codec audio, contenant des convertisseurs analogiques numériques, et qui est directement lié aux entrées sorties Line In, Line Out, Mic In, et HP Out.// Ainsi, c'est ce périphérique audio qui fera l'échantillonnage du signal d'entrée, et qui enverra le signal vers la sortie après filtrage.

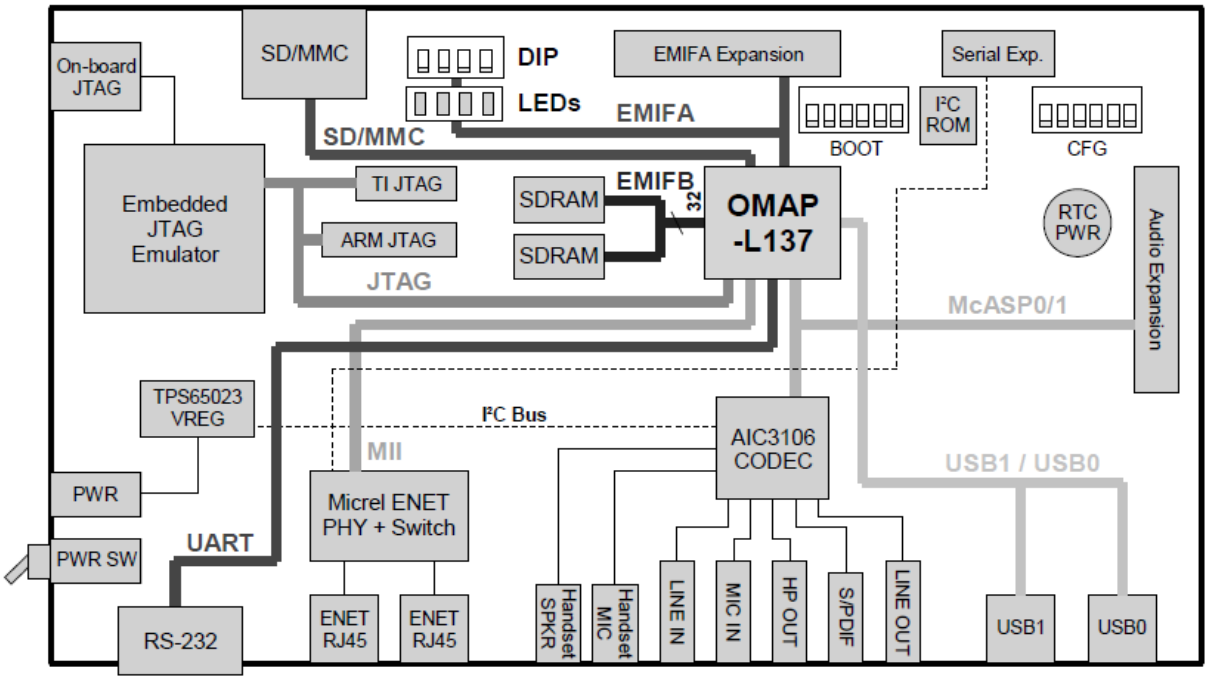


FIGURE 1 – Schéma fonctionnel du kit de développement EVMOMAP L137 EVMOMAPL137

Le programme de configuration de l'AIC3106 `evmomapl137_aic_3106_init.c` est fourni par le constructeur Spectrum Digital, et l'unique paramètre que l'on sera amené à modifier dans ce TP, est la fréquence d'échantillonnage. Comme dit précédemment, l'OMAP contient un DSP, et aussi une interface audio qui permet le transfert des signaux entre le DSP et l'AIC3106. Là encore, le programme de configuration du McASP `evmomapl137_mcasp_init.c` est fourni par le constructeur. On jettera un ?il à ces 2 fichiers mais uniquement pour comprendre comment ils sont structurés, et repérer où changer la fréquence d'échantillonnage de l'AIC.

# 5 Les fenêtres de l'interface graphique de CCS

- Fenêtre 1 : Explorateur de projet (*ouvrir / fermer / activer / etc.* ).
- Fenêtre 2 : Débogueur (*lance / arrêter / exécuter pas à pas avec F5 ou F6 / etc.*).
- Fenêtre 3 : Editeur de fichier (*voir et modifier les sources / suivre l'exécution pas à pas / etc*).
- Fenêtre 4 : Visualisation des registres, variables, expressions, etc.
- Fenêtre 5 : Contenu de la mémoire, code assembleur correspondant au programme.

D'autres fenêtres peuvent être utilisées, notamment celle qui permet de tracer un signal, son spectre, etc ?

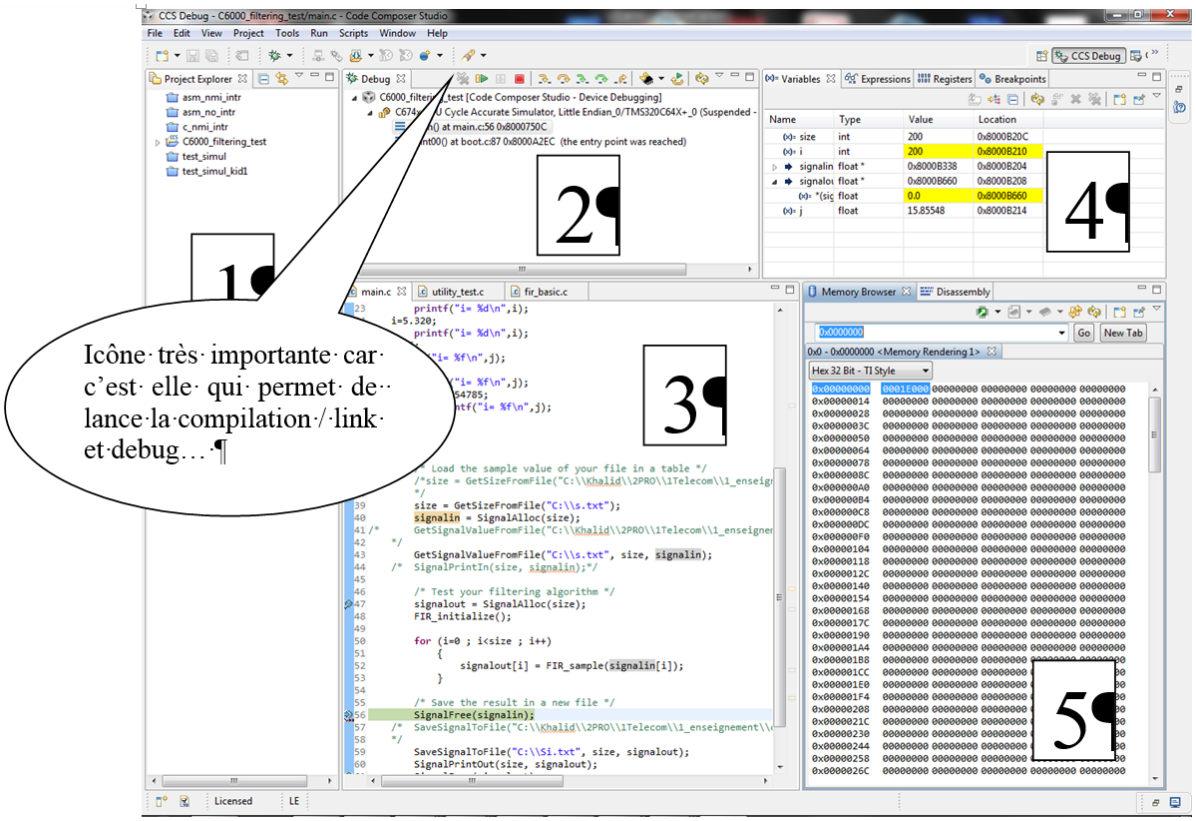


FIGURE 2 – Interface graphique de CCS.

# 6 Lancement d'un projet sous CCS

Commencer par aller sous Moodle pour récupérer le ZIP contenant les projets pour ces TP. Ils seront sauvegardés par défaut dans `c:\utilisateur\Download`. Lancer CCS. Cliquer sur projet/import/ et aller dans le répertoire `c:\utilisateur\Download` pour choisir le projet que vous voulez importer.
CSS le mettra alors dans `c:\utilisateur\Workspace`.
Vous verrez alors apparaître le projet dans la fenêtre 1. En développant l'arborescence, vous verrez les différents fichiers, et en double cliquant dessous vous les ouvrez pour en voir le contenu.
Vous pouvez, en cliquant sur l'icône du debug, lancer la compilation, l'édition de lien et le débogueur.
Ainsi vous pouvez commencer l'exécution du programme soit en pas à pas (touche F5), soit en lui demandant de s'exécuter jusqu'au prochain point d'arrêt (si vous en avez créé, en vous plaçant là ou voulez en mettre un, puis bouton droit de la souris et choisir Toogle Breakpoint).
Dans la fenêtre 3, vous voyez votre programme, et vous voyez l'exécution en pas à pas. En vous positionnant sur une variable, vous pouvez en voir la valeur.
Dans la fenêtre 4, vous pouvez visualiser le contenu des registres, les valeurs des variables, etc., Dans notre cas, c'est là qu'il est possible de récupérer les adresses des variables, et notamment des signaux d'entrées et de sorties pour pouvoir les visualiser grâce à l'outil Graph.
Dans la fenêtre 5, vous pouvez voir le contenu de la mémoire. Il suffit pour cela de donner l'adresse à partir de laquelle on veut visualiser la mémoire.

**Attention** : en TD, il a déjà été montré comment les données sont rangées en mémoire, et plus exactement, la manière avec laquelle la mémoire est affichée par le CCS.

# Table des matières