UNIVERSITE SIDI MOHAMMED BEN ABDELLAH
FACULTE DES SCIENCES ET TECHNIQUES - FES
DEPARTEMENT DE GENIE ELECTRIQUE

LICENCE
Electronique, Télécommunications et Informatique
(ETI)

RAPPORT DE FIN D'ETUDES

Intitulé :

Implémentation sur DSP d'un filtre numérique passe-bas

Réalisé Par :

ADIL MAKHOUKH

Encadré par :

Pr.Hicham GHENNIOUI(FST FES)

Soutenu le vendredi 14 Juin 2013devant le jury

Pr.Abdellah MECHAQRANE	**(FST FES)**
Pr .Farid ABDI	**(FST FES)**
Pr.Hicham GHENNIOUI	**(FST FES)**

RESUME

Un DSP « Digital Signal Processor », qu'on pourrait traduire par « processeur de signal numérique » est un microprocesseur optimisé pour exécuter des applications de traitement numérique du signal (filtrage, extraction de signaux, etc.) le plus rapidement possible [7].

Les DSP sont utilisés dans la plupart des applications du traitement numérique du signal en temps réel. On les trouve dans les modems (modem RTC, modem ADSL), les téléphones mobiles, les appareils multimédia (lecteur MP3), les récepteurs GPS… Ils sont également utilisés dans des systèmes vidéo, les chaînes de traitement de son, partout où l'on reçoit un signal que l'on doit modifier à l'aide du filtrage.

Ce projet de fin d'étude a pour but l'étude de l'architecture des DSP et plus particulièrement celui de TEXAS INSTRUMENT, le TMS320C6713, ainsi que l'implémentation sur la carte d'évaluation TMS320C6713DSK d'un filtre numérique à réponse impulsionnelle finie, afin de pouvoir étudier de façon pratique ce filtre.

Remerciements

En premier lieu, je tiens à remercier très sincèrement mon encadrant, Monsieur, Hicham GHENNIOUI, Professeur à la Faculté des Sciences et Techniques de Fès, pour les conseils judicieux et pour le soutien moral qui m'ont permis de réaliser ce travail.

Par la même occasion, je tiens à remercier les membres de jury pour l'intérêt qu'ils ont porté à ce travail en acceptant de l'évaluer.

Finalement, je remercie mes parents, mes frères, mes amis, ainsi que tous les membres de ma grande famille de près et de loin.

SOMMAIRE

LISTE DES FIGURES

LISTE DES tableaux

Glossaire et abréviations

Terme	Définition
A/D	Analog/Digital
ADC	Analog Digital Converter
ADSL	Asymetric Digital Suscriber Line
CAN	Convertisseur Analogique Numérique
CCS	Code Composer Studio
CNA	Convertisseur Numérique Analogique
CPU	Central Processing Unit
D/A	Digital/Analog
DAC	Digital Analog Converter
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DSK	Digital Starter Kit
DSP	Digital signal Processor
E/S	Entrée/Sortie
EDMA	Enhanced Direct Memory Access
EEG	Electro Encyphalo Graphie
EMG	Electromyographie
EMIF	External Memory Interface
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
GFLOPS	one billion Floating Point Operations Per Second.
GPS	Global Positioning System
HDSL	High-bit-rate Digital Subscriber Line
HPI	Host Port Interface
IIR	Infinite Impulse Response
JTAG	Joint Test Action Group
MAC	Multiply and Accumulat

McBSP	Multi-channel Buffered Serial Port
MFLOPS	one Million Floating Point Operations Per Second.
MIPS	Million d'Instructions par Seconde
PLL	Phase-locked loop
RAM	Random-Access Memory
RMN	Résonance Magnétique Nucléaire
RTC	Réseau Téléphonique Commuté
SRAM	Static Random Access Memory
TFR	Transformé de Fourier Rapide
TNS	Traitement Numérique du Signal
VLIW	Very Long Instruction Word

Introduction générale

L'utilisation des techniques numériques a nettement pris le pas sur les techniques analogiques dans de nombreux domaines, parmi lesquels on peut citer,

- le traitement du signal classique (filtrage, transformée de Fourier rapide, génération de signaux,…),
- les télécommunications (codage-décodage, modulation-démodulation, égalisation adaptative, annulation d'écho, cryptage,…),
- le traitement de la parole (codage-compression, analyse, reconnaissance, synthèse…) et des images (codage-compression, reconnaissance de formes,…),
- le radar (poursuite multimode, traitement antiréverbération, identification de cibles,…),
- les applications médicales : traitement de signaux EEG, EMG,…, imagerie biomédicale dans les RMN…),
- et la commande (industrielle, avionique,…), etc.

Les exigences contradictoires de très basse consommation d'énergie et d'accroissement de la fonctionnalité du TNS ainsi que la rapidité de transfert et de traitement des données ont conduit à un certain nombre d'avancées dans les algorithmes, les technologies des semi-conducteurs et les architectures de système qui ont abouti au développement d'un système DSP (Digital Signal Processor) faisant partie de la famille des microprocesseurs, mais dont les jeux d'instructions et l'architecture le rendent performant dans le domaine du traitement numérique du signal.

D'un point de vue économique, on peut être assuré d'une croissance importante du marché des DSP eu égard les équipements dans lesquels ils sont présents,

- les téléphones mobiles sous forme de circuits spécialisés à cœur de DSP,
- les modems sous une forme similaire,
- les terminaux DSL (Digital Subscriber Line), HDSL, ADSL… dans lesquels ils assurent l'égalisation, le brouillage, l'annulation d'écho, la suppression de la télédiaphonie, etc,
- la télévision haute définition (TVHD) (codage-décodage du son et de l'image),
- la radiodiffusion numérique (DAB pour Direct Broadcast Audio),
- les DVD (Digital Video Disks) pour le décodage son Dolby AC-3 et le décodage vidéo MPEG-2,
- le contrôle des moteurs à courant alternatif,
- et le contrôle des disques durs : utilisation de techniques à maximum de vraisemblance (PRML Partial Response, Maximum Likehood) pour augmenter la densité d'enregistrement, etc.

Les objectifs principaux de ce travail se résument aux nombres de deux,

- Découverte en pratique de l'architecture des processeurs de signaux (DSP) et les problèmes liés à leur programmation en implantant des algorithmes simples sur un processeur 32 bits à virgule flottante (TMS320C6713 DSK).
- Complètement des connaissances en traitement numérique du signal en implémentant un algorithme d'un filtre FIR passe-bas sur le DSP, objet de notre étude, ainsi que la vérification en pratique des résultats théoriques.

Le document est organisé de la manière suivante. Nous introduisons dans le premier chapitre des notions théoriques sur les processeurs dédiés au traitement du signal. Le second chapitre est consacré à

la description de la carte TMS320C6713 DSK à base de DSP et qui sera utilisée pour implémenter le filtre numérique. Nous présentons dans le troisième chapitre les deux modèles de filtrage numérique passe-bas que nous avons conçu et développé : le premier est le modèle MATLAB qui sert de référence et le deuxième est le modèle pratique pour DSP. Des résultats sont également présentés au niveau du troisième chapitre. Nous terminons ce document par une conclusion et des perspectives de ce travail.

Chapitre 1 – Concepts de base des DSP

Introduction

Les DSP sont utilisés dans de nombreux domaines d'applications comme cité au niveau de l'introduction générale. La façon dont un DSP s'insère dans une chaîne de traitement du signal est illustrée dans la figure 1.1. Cette chaîne comporte généralement trois étapes,

- la conversion analogique numérique,
- le traitement pour le DSP,
- et la conversion numérique analogique.

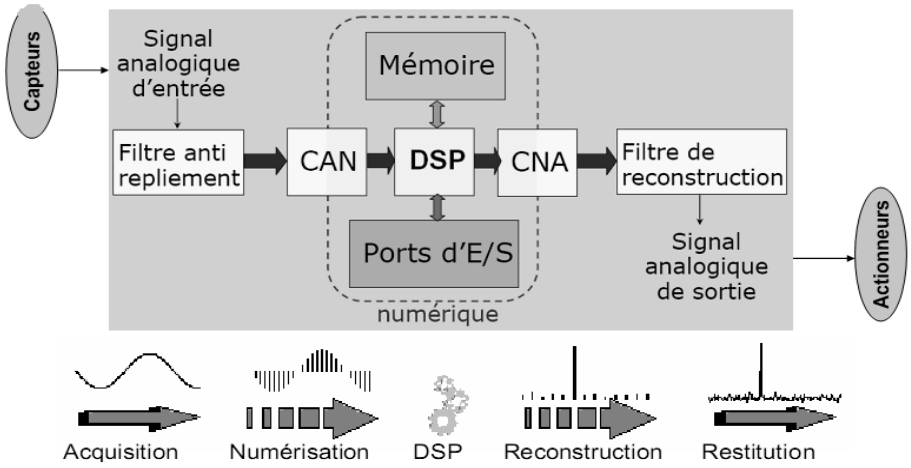


Figure 1.1 - Chaîne complète typique d'un système de traitement numérique de signal.

L'étape de conversion analogique-numérique a pour but de réaliser les opérations d'échantillonnage et de quantification du signal à traiter. Le DSP effectue le traitement désiré : filtrage, détection d'un signal, estimation, régulation, modulation,… Le signal traité est alors de nouveau converti (si nécessaire) en signal analogique lors de l'étape de conversion numérique-analogique.

Les étapes d'entrée et de sortie peuvent elles-mêmes être décomposées en plusieurs étapes. La première étape de conversion analogique-numérique est généralement un filtrage analogique effectué par un filtre antirepliement. Son but est de filtrer le signal qui doit être traité par le DSP de façon à ce que le critère d'échantillonnage de Shannon soit satisfait : le spectre du signal ainsi filtré exhibe donc une amplitude très faible pour les fréquences supérieures à la moitié de la fréquence d'échantillonnage : on évite ainsi le phénomène de recouvrement de spectre. L'échantillonnage est alors effectué en sortie de ce filtre. De façon pratique, il ne peut être réalisé à l'aide d'un Peigne de Dirac. On utilise un échantillonneur constitué par une suite périodique d'impulsions rectangulaires de durée Ω et de période T_e. L'échantillonnage et alors réalisé par un système bloquant son signal d'entrée pendant Ω. Généralement on choisit $\Omega = T_e$ pour des raisons de simplicité et le système s'appelle un échantillonneur

bloqueur. En sortie de l'échantillonneur bloqueur, on place alors un convertisseur analogique-numérique dont le but est d'effectuer l'opération de quantification.

En sortie du DSP, on place un convertisseur numérique-analogique qui fournit en sortie un signal analogique de type fonction en escalier. En effet celui-ci se contente généralement de maintenir pendant une période d'échantillonnage la valeur qu'il a en entrée : on obtient ainsi un bloqueur d'ordre 0. Afin «d'adoucir » ce signal analogique, on place en sortie de ce convertisseur un filtre dit filtre de reconstruction [4].

Avantages des systèmes à base de dsp

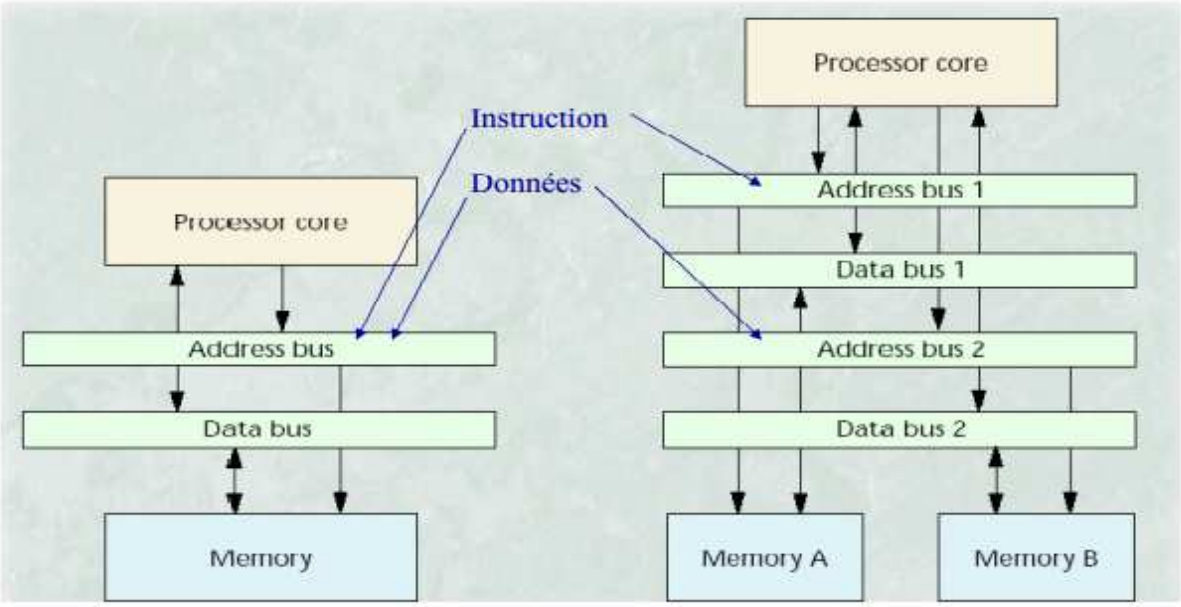
Tous les systèmes à base de DSP bénéficient des avantages suivants,

- **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 traitement numérique peuvent évoluer en fonction des mises à jour des programmes, et cela pendant toute la durée de vie du produit incluant le système. La modification d'un filtre numérique ne nécessite pas un changement matériel.
- **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.
- **Répétitivité, reproductibilité** : les valeurs des composants analogiques sont définies avec une marge de précision plus ou moins grande. Dans ces conditions, aucun montage analogique n'est strictement reproductible à l'identique, il existe toujours des différences qu'il convient de maintenir dans des limites acceptables. Un programme réalisant un traitement numérique est par contre parfaitement reproductible, «à l'infini» [1].

Différence entre DSP et microprocesseurs ordinaires

La différence fondamentale se situe sur le simple fait qu'un microprocesseur n'est pas destiné à une application spécifique alors que le DSP est destiné au traitement du signal, ce qui lui permet d'avoir une architecture optimisée pour ces types de traitement. Généralement les principales distinctions se résument au nombre de deux,

- Contrairement aux microprocesseurs classiques, 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, en anglais Multiply and ACcumulate.
- Une autre particularité des DSP par rapport aux microprocesseurs ordinaires, c'est qu'ils ont une architecture Harvard, ceci est en opposition à celle de Von Newman. Dans l'architecture Harvard, la mémoire des données et celle du programme sont



L'architecture de von Neumann ne permet qu'un seul accès mémoire par cycle

Avec une architecture Harvard, le programme et les données sont séparés sur au moins deux bus

physiquement différentes. Ces deux mémoires sont accessibles par l'intermédiaire de bus différents. Cette configuration interne permet d'avoir simultanément accès à la mémoire et d'exécuter une instruction. La vitesse d'exécution d'un programme s'en trouve donc doublé (en théorie). La figure 1.2 illustre la différence entre les deux architectures : Von Newman et Harvard.

Figure 1.2 - Architecture de Von Neumann vs architecture de Havard.

Outre l'opération MAC et l'architecture Havard, la plupart des DSP (dont le TMS320C6713 qui fait objet de notre étude) ont d'autres particularités importantes par rapport aux processeurs généraux (Intel, Cyrix, AMD9…), elles sont en général [1],

- Un adressage circulaire, permettant de ne pas faire des tests de modulo sur les pointeurs de tampons.
- Un mode d'adressage par inversion de bits, servant à réorganiser les échantillons de sortie de Transformée de Fourier Rapide (TFR ou FFT).
- De la mémoire interne minimisant les temps d'accès RAM.
- Des banques de mémoire, permettant de scinder par exemple partie réelle et partie imaginaire.
- Une architecture à pipe-line permettant de paralléliser le maximum d'opérations élémentaires (pré-chargement d'instruction, chargement des données, exécution d'opérations arithmétiques, stockage des résultats).
- Parfois des convertisseurs analogiques numériques et numériques analogiques.
- Certains DSP utilisent l'architecture dite « Structure de Harvard modifiée » (pour réduire le coût de la structure Havard). À l'extérieur, le DSP ne propose qu'un bus de données et un bus d'adresse, comme la structure Von Neumann. Toutefois, à l'intérieur, la puce DSP dispose de deux bus distincts de données et de deux bus distincts d'adresses. Le

transfert des données entre les bus externes et internes est effectué par multiplexage temporel (C'est le cas du TMS320C6713).

Types et formats des données manipulées par les DSP

Un des points essentiels 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 et ceux à virgule flottante.

DSP à virgule fixe

Les données sont représentées comme étant des nombres fractionnaires à virgule fixe, par exemple -1.0 à +1.0, ou comme des entiers classiques. La représentation de ces nombres fractionnaires s'appuie sur la méthode du « complément à deux ». Le tableau 1.1 décrit la représentation à virgule fixe d'un nombre.

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

Tableau 1.1- Représentation à virgule fixe d'un nombre.

L'avantage de cette représentation est de permettre facilement l'addition binaire de nombres aussi bien positifs que négatifs.

La précision des calculs est un point critique des DSP à virgule fixe, car le concepteur de programmes doit rester vigilant à chaque étape d'un calcul. Il doit rechercher la plus grande dynamique possible (c.à.d. exploiter au mieux la gamme des nombres disponibles), pour conserver une bonne précision des calculs, tout en évitant autant que possible les débordements du ou des accumulateurs. Les bits supplémentaires des accumulateurs (les bits de garde) prévus à cet effet permettent de réduire cette contrainte [1].

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).

Poids des bits	MSB			LSB		
Valeurs	-2^0	2^{-1}	2^{-2}	2^{-3}		
	-1	½	¼	1/8		
	0	1	0	1	=	0,5+0,125 = 0,625
	1	1	0	1	=	-1+0,5+0,125 = -0,375
le moins positif	0	0	0	1		= +0,125
le plus positif	0	1	1	1	=	0,5+0,25+0,125 = +0,875
le moins négatif	1	1	1	1	=	-1+0,5+0,25+0,125= -0,125
le plus négatif	1	0	0	0		= -1

Tableau 1.2-

Représentation à virgule flottante d'un nombre.

La très grande dynamique proposée par les DSP à virgule flottante (comme le TMS320C6713) permet virtuellement de ne pas se soucier des limites des résultats calculés lors de la conception d'un programme. Cet avantage a cependant un prix, à savoir qu'un système basé sur un DSP à virgule flottante a un coût de fabrication supérieur par rapport à un système basé sur DSP à virgule fixe. La puce d'un DSP à virgule flottante nécessite à la fois une surface de silicium plus importante (cœur plus complexe), et un nombre de broches supérieur, car la mémoire externe est elle aussi au format 32 bits. Le système revient donc plus cher (exemple : 2 x 32 broches juste pour les bus de données externes avec une architecture Harvard de base).

Sélection du DSP le plus adapté

La sélection d'un DSP se base avant tout sur la puissance de traitement nécessaire. Toutefois, la performance du DSP n'est pas le seul critère à prendre en compte, il faut également tenir compte des impératifs suivants,

- Le type de DSP à utiliser (virgule fixe ou flottante) en fonction du domaine d'application.
- Les ressources mémoires utilisés.
- Les besoins d'un ou de plusieurs timers internes, de ports série synchrones ou asynchrone, etc.
- La nécessité éventuelle d'exécuter un système temps réel, qui s'avérera plus facile à implanter sur certains DSP.
- Le coût du DSP, son rapport « performance/prix » en fonction du volume de production envisagé.

D'autres éléments non négligeables interviennent dans le choix d'un DSP, il s'agit des moyens disponibles pour mener le développement en un temps donné, comme,

- La qualité de la documentation (de préférence claire et abondante).
- La disponibilité de notes d'applications, d'un support technique.
- La qualité du système de développement utilisé.
- La possibilité d'utiliser un langage de haut niveau (Langage C).
- La présence de librairies (du constructeur ou de tierces parties).
- La possibilité de réaliser facilement des prototypes et à faible coût.

Le choix n'est pas toujours simple et certains critères peuvent être contradictoires, certaines règles de choix se dégagent quand même. Ainsi pour des applications destinées à faire un fort volume de production, le critère déterminant est sans conteste le prix du DSP. Pour des applications à faible volume de production, le prix du DSP importe peu, le critère est alors la facilité de développement.

Dans tous les cas, la présence d'un bon support technique est un facteur à ne pas négliger, car un DSP est quand même plus complexe à mettre en œuvre qu'un microprocesseur classique [6].

produits de texas instruments

La famille de processeurs la plus répandue actuellement est sans conteste celle des DSP de Texas Instruments qui détient environ 70 % du marché, les 30 % restant étant partagés entre Motorola, Analog Devices, Lucent Technologies, Nec et Oki.

Les familles les plus récentes des DSP de Texas instrument sont,

- TMS320C54x, DSP format fixe ;

- TMS320C20x, DSP format fixe;

- TMS320C24x, DSP format fixe;

- TMS320C62x, DSP format fixe à architecture VLIW ;

- TMS320C67x, DSP format flottant à architecture VLIW;

- TMS320C64x, DSP format fixe;

Ces nouvelles familles sont regroupées en trois classes appelées plates-formes. Ces trois classes sont appelées:

-TMS320C6000, formée des familles C62x, C67x et C64x;

-TMS320C5000, formée de la famille C54x et des C54xx;

-TMS320C2000, formée des familles C20x et C24x;

Le tableau suivant résume les principales caractéristiques de ces trois classes,

Application		Type de DSP	Caractéristique
TMS30C6000 DSP hautes performances			
C64x	Application au traitement numérique de signal	32 bits virgule fixe	24000 MIPS -3,0 GHz
C62x	Applications exigeantes en vitesse: stations de base des réseaux de communications mobiles, équipement de radiodiffusion, réseaux informatiques	16 bits virgule fixe architecture VLIW	1200-2400 MIPS

C67x	Applications exigeantes en précision, dynamique et vitesse: Antennes adaptatives des stations de base, imagerie médicale, reconnaissance de parole, graphisme…	32 bits virgule flottante architecture VLIW	600MFLOPS-1GFLOPS
TMS320C5000 DSP optimisés en consommation			
C54x	Applications de télécommunications exigeantes en coût, consommation, vitesse: terminaux mobile, voix sur IP, alphapages…	16 bits virgule fixe	0.54 mWLMIPS 30-200 MIPS
TMS320C2000 DSP optimisés pour les applications de contrôle			
C20x	Applications de grand volume en téléphonie, électronique grand public, appareils photos numériques ou contrôleurs de disques durs…	16 bits virgule fixe	PLL, UART, timers, mémoire flash intégrée 20-40MIPS
C24x	Applications de contrôle moteur, automatisation, robotique, contrôle d'appareils électroménagers…Bon compromis prix/performance	16 bits virgule fixe	port série SCI, SPI et CAN, convertisseur Analogique/numérique 20MIPS

Tableau 1.3- Caractéristiques des trois classes des DSP de Texas Instruments.

Chapitre 2 - Le processeur TMS320C6713

introduction

La plateforme TMS320C6000 des processeurs de signaux numériques fait partie de la famille TMS320 de TEXAS INSTRUMENTS. Elle comporte les processeurs TMS320C62x à arithmétique fixe et TMS320C67x à arithmétique flottante. Le TMS320C6713 est considéré comme le membre le plus performant de la catégorie C67x, son architecture VLIW lui permet le traitement par paquets de huit instructions en parallèle par huit unités fonctionnelles. La discussion dans ce chapitre se focalisera sur le processeur TMS320C6713. L'architecture et les périphériques qui lui sont associés seront également discutés.

L'évolution de la famille TMS300 est indiquée dans la figure suivante :

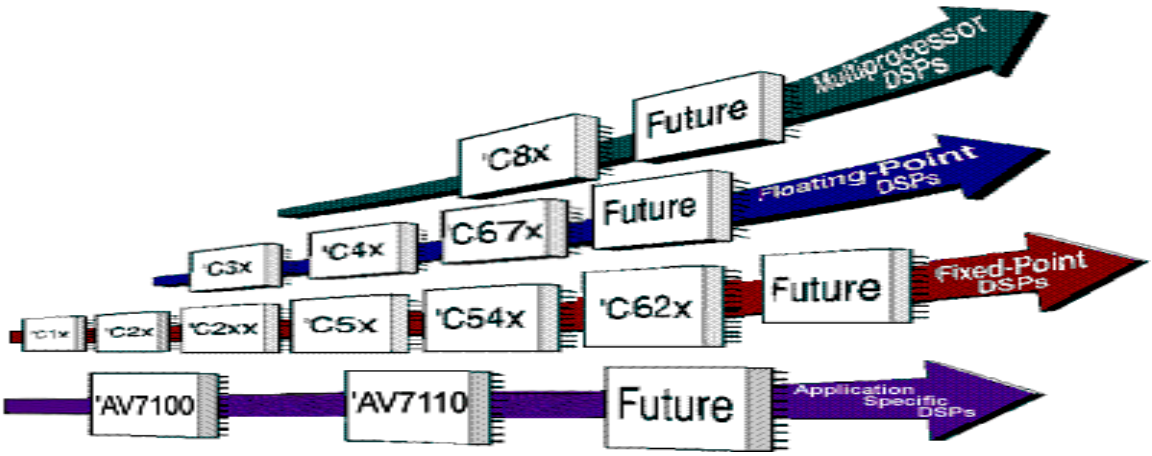


Figure 2.1 Évolution de la famille TMS320.

L'architecture du Tms320c6713

Le TMS320C6713 est constitué de trois parties principales (figure suivante),

- l'unité centrale de traitement CPU,
- les périphériques,
- et la mémoire.

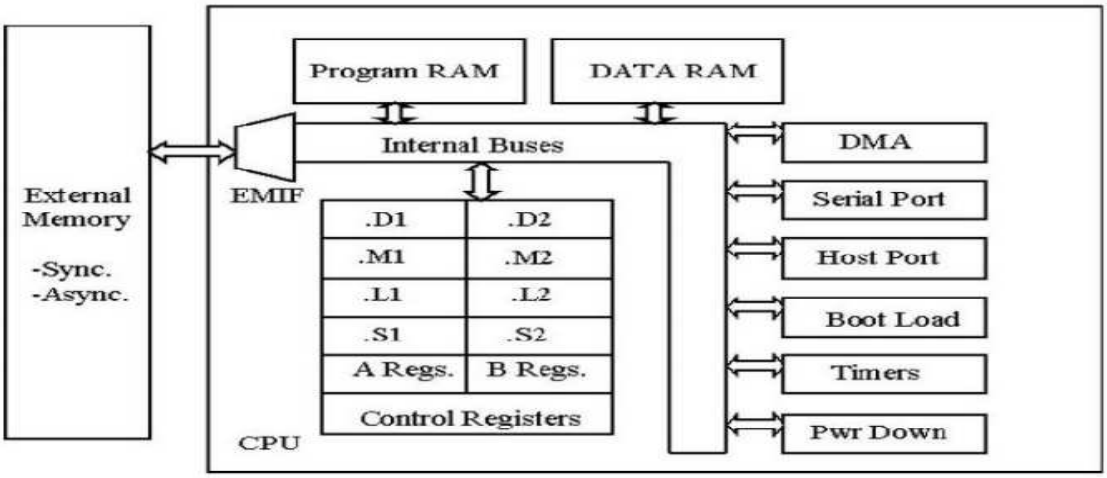


Figure 2.2 - Bloc diagramme simplifié de la famille TMS320C67xx.

Unité centrale de traitement (CPU)

Le CPU est constitué d'une unité de contrôle de programme, de deux unités fonctionnelles, de deux blocs de 16 registres de 32 bits, de contrôleurs d'interruptions et d'autres éléments.

A. Unité de contrôle de programme

Elle est constituée des éléments suivants,

- Unité "fetch" programme : Elle a pour rôle récupérer les programmes. Cette opération se déroule en quatre phases :

- Phase PG: l'adresse du code est générée.
- Phase PS : l'adresse est envoyée à la mémoire.
- Phase PW: l'attente de lecture du code de la mémoire.
- Phases PR : la lecture du code.

- Unité "dispatche" de l'instruction: le code récupéré de la mémoire est affecté à 1'unité fonctionnelle associée.
- Unité de décodage de l'instruction: elle a pour rôle de décoder l'instruction.

B. Unités fonctionnelles

Le CPU contient huit unités: fonctionnelles divisées en deux parties 1 et 2. Leurs fonctions sont les suivantes:

- Unités .Ml et .M2 : ces unités sont dédiées à la multiplication.
- Unités .L1 et .L2 : ces unités sont dédiées à l'arithmétique et la logique.
- Unités .Dl et .D2 : ces unités sont dédiées au chargement, la sauvegarde et calcul d'adresse.
- Unités .S1 et .S2: ces unités sont dédiées pour le décalage de bit, l'arithmétique, la logique et le branchement [5].

Functional Unit	Description
.L unit (.L1, .L2)	32/40-bit arithmetic and compare operations Left most 1, 0, bit counting for 32 bits Normalization count for 32 and 40 bits 32 bit logical operations 32/64-bit IEEE floating-point arithmetic Floating-point/fixed-point conversions
.S unit (.S1, .S2)	32-bit arithmetic operations 32/40 bit shifts and 32-bit bit-field operations 32 bit logical operations Branching Constant generation Register transfers to/from the control register file 32/64-bit IEEE floating-point compare operations 32/64-bit IEEE floating-point reciprocal and square root reciprocal approximation
.M unit (.M1, .M2)	16 x 16 bit multiplies 32 x 32-bit multiplies Single-precision (32-bit) floating-point IEEE multiplies Double-precision (64-bit) floating-point IEEE multiplies
.D unit (.D1, .D2)	32-bit add, subtract, linear and circular address calculation

Tableau 2.1- Description détaillée unités fonctionnelles du TMS320C6713

C. Registres

Le CPU contient 32 registres de 32 bits divisés en deux blocs égaux : registre fichier A (AO-A15) et registre fichier B (BO-B15), leurs fonctions sont réparties comme suit:

- Les registres A1-A2 et BO-B1-B2 : ils sont utilisés comme registres conditionnels.
- Les registres A4-A7 et B4-B7: ils sont utilisés pour adressage circulaire.

- Les registres AO-A9, BO-B2 et B4-B9 : ils sont utilisés comme registres temporaires.
- Les registres A10-A15 et B10-B15 : ils sont utilisés pour la sauvegarde et la restitution de données d'un sous-programme.

À ces 32 registres s'ajoutent les registres de contrôles et d'interruptions.

Les périphériques du TMS320C6713

Le TMS320C6713 a plusieurs périphériques qui sont :

- Le contrôleur DMA : Il permet sans l'aide du CPU de transférer des données entre les espaces mémoire (interne, externe et des périphériques). Il a quatre canaux programmables et un autre canal auxiliaire.
- Le contrôleur EDMA : Il permet le transfert des données entre les espaces mémoire comme le DMA. Il a 16 canaux programmables.
- L'interface port hôte HPI. Il donne au processeur hôte un contrôle total pour un accès direct de l'espace mémoire du CPU et à la cartographie de la mémoire des périphériques du DSP.
- Deux McBSP qui sont des ports séries multi-canaux protégés. Ils permettent la communication avec les périphériques externes. Ils ont la même structure. Ils supportent une communication full-duplex.
- L'interface de mémoire externe EMIF : Il permet l'interface avec plusieurs éléments (mémoires) externes.
- Les compteurs : Le DSP possède deux compteurs qui peuvent être synchronisés par une source interne ou externe et ils sont utilisés comme générateurs de pulsations, compteurs d'événements externes, interrupteurs du CPU après l'exécution de tâches et déclencheur du DMA/EDMA.
- Les interruptions : l'ensemble des périphériques contient jusqu'à 32 sources d'interruptions.

La structure de la mémoire

Le TMS320C6713 basé sur l'architecture de Havard modifiée utilise une mémoire externe et une mémoire interne (Figure 2.3).

- La mémoire externe occupe les espaces CEO, CE1, CE2, CE3.
- La mémoire interne a une taille de 260 KB qui est décomposée en deux niveaux :
  - Le niveau (L1) est constitué de deux mémoires caches de 4 KB chacune, (L1P) qui est utilisée pour les programmes et (L1D) qui est utilisée pour les données.
  - Le Niveau (L2) est composé de 256 KB de mémoire partagée entre mémoire des données et mémoire de programmes.

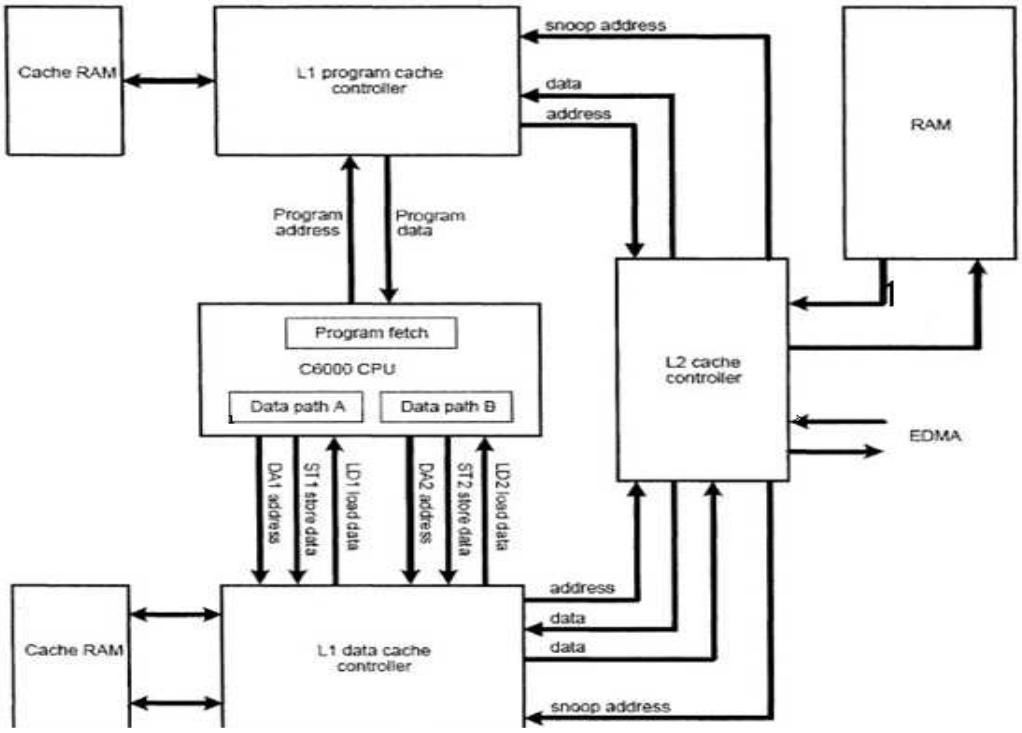


Figure 2.3- Organisation de la mémoire interne du TMS320C6713.

TMS320C6713 DSK (C6713 DSK)

La carte d'évaluation DSK est puissante, relativement peu coûteuse, avec les outils de support matériels et logiciels nécessaires pour le traitement du signal en temps réel.

Il s'agit d'un système DSP complet. Cette carte comprend le processeur de traitement numérique du signal, le TMS320C6713 à virgule flottante et un codec stéréo TLV320AIC23 (AIC23) de 32 bits pour l'entrée et la sortie (Figure 2.4). Le codec AIC23 « utilisant une technologie sigma-delta » fonctionne comme un CAN pour les entrées analogiques et comme un CNA pour les sorties numériques du DSP. Il se connecte à une horloge système de 12 MHz. Le taux d'échantillonnage variable de 8 à 96 kHz et peut être réglé facilement. La carte comporte un emplacement libre pour l'ajout d'un périphérique ou d'une carte additionnelle ("daughter card"), un emplacement pour ajouter de la mémoire, quatre LED ("Light-Emitting Diodes") et quatre DIP switchs ("Dual In-line Pin") programmables.

La carte DSK comprend 16 MB de mémoire synchrone dynamique à accès aléatoire (DRAM) et 256 kB de mémoire flash. Le DSK fonctionne à 225 MHz, il intègre un régulateur de tension qui fournit 1,26 V pour le noyau C6713 et 3,3 V pour la mémoire et les périphériques.

Le DSK dispose également de quatre prises audio jacks de 3,5 mm, deux pour les entrées: microphone (mono) et "line in" (stéréo), et deux pour les sorties : "speaker" (stéréo) et "line out" (stéréo). En fait les deux entrées (respectivement les deux sorties) renvoient les signaux au même port physique, c'est-à-dire au même signal d'entrée (respectivement de sortie). La seule différence entre microphone et "line in" (respectivement "speaker" et "line out") réside dans les impédances des ports. Autrement dit, on a quatre prises audio, mais une seule entrée et une seule sortie, chacune disponible avec deux impédances différentes [2].

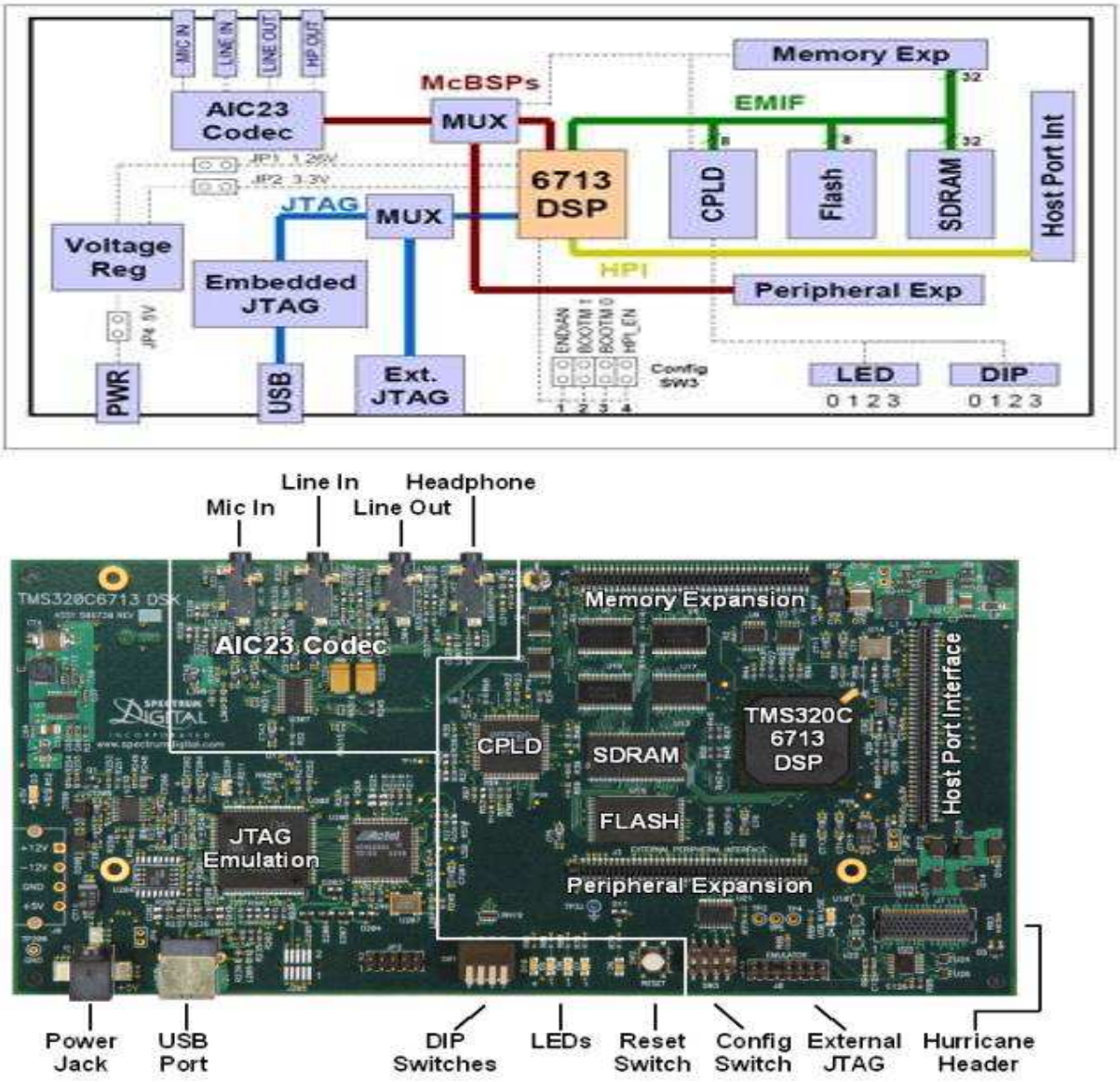


Figure 2.4 - Blocks Diagramme du C6713 DSK.

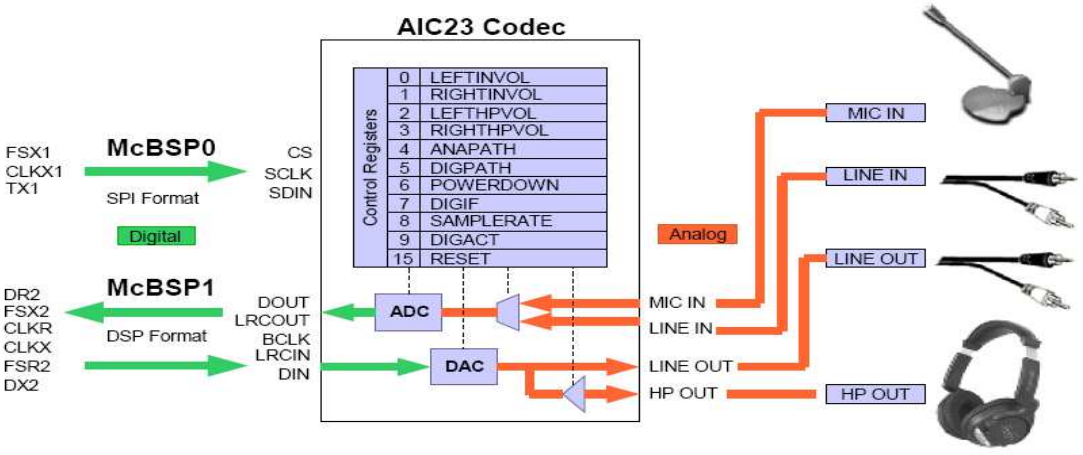


Figure 2.5- CODEC TLVAIC23.

Organisation de la mémoire du C6713 DSK

L'organisation de la mémoire de la carte C6713 DSK est décrite au niveau de la figure 2.6.

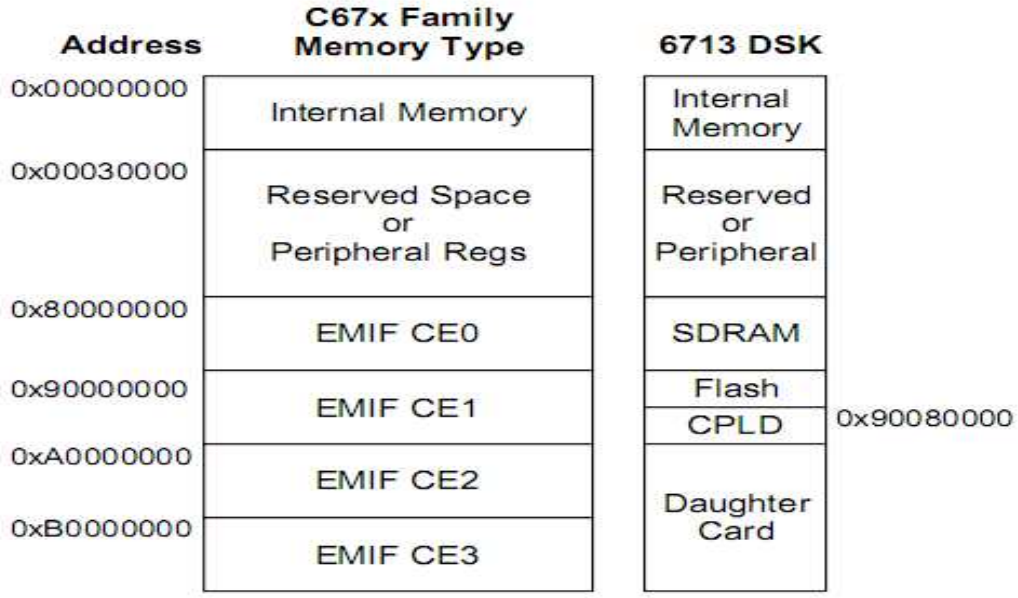


Figure 2.6- Cartographie de la mémoire du C6713 DSK.

Le Code Composer Studio

L'évaluation des performances des algorithmes sur le DSP est effectuée en utilisant Code Composer Studio (CCS), son fonctionnement est illustré par la figure suivante,

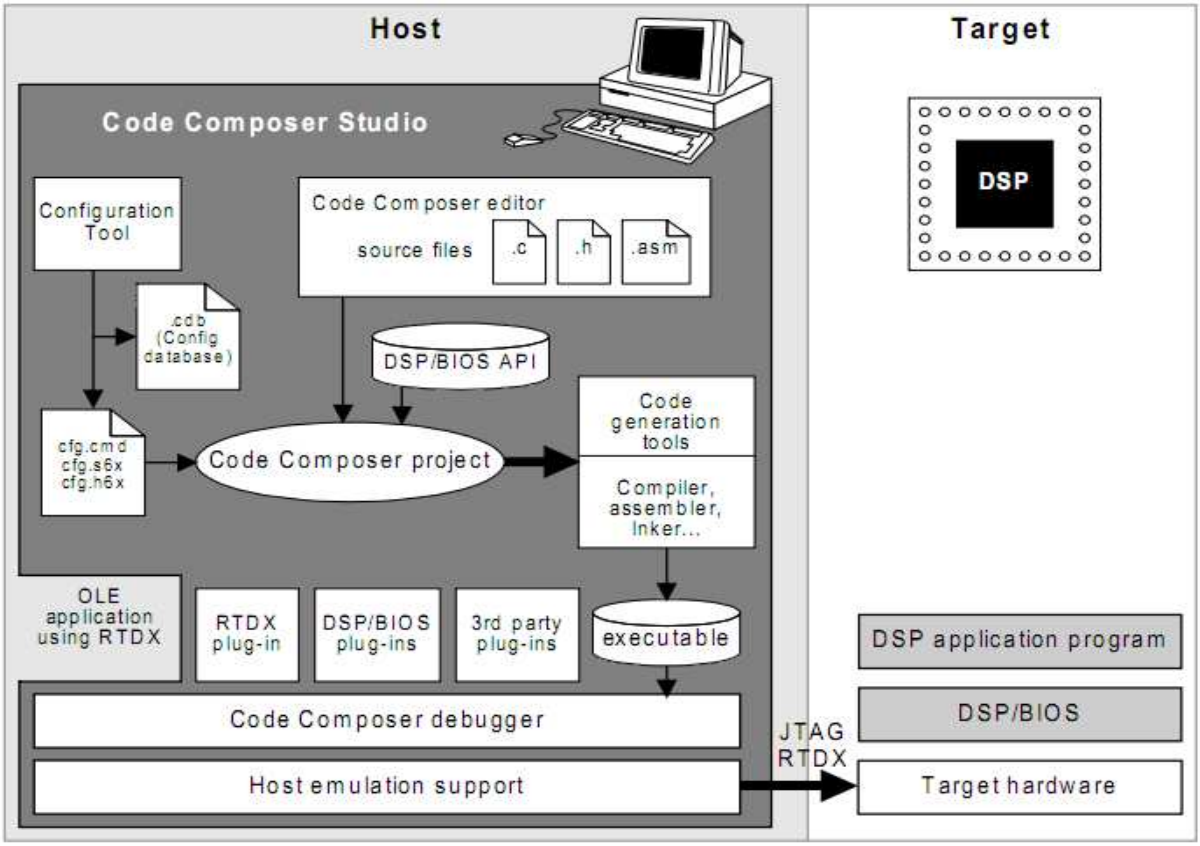


Figure 2.7- Structure du système du développement du TMS320C6713.

Le logiciel Code Composer Studio est une plate-forme de développement qui inclut les éléments suivants,

- Environnement de développement intégré (IDE) : il permet l'édition ('built'), et la correction ('debug') des programmes destinés au DSP.
- Outils de génération du code pour le TMS320C6000 : ces outils sont le compilateur, l'assembleur et l'éditeur de lien. Le compilateur C/C++ permet de compiler le programme source (xxx.c) pour le convertir en assembleur (xxx.asm), l'assembleur reçoit le fichier xxx.asm et le convertit en langage machine ou fichier objet (xxx.obj), enfin l'éditeur de liens (linker) qui combine les fichiers objet et les fichiers librairies et le fichier xxx.cmd pour produire un fichier exécutable avec une extension .out, c'est ce fichier qui sera chargé sur le processeur C6713 pour être exécuter.
- DSP/BIOS: c'est un outil d'analyse en temps réel, pour s'en servir, on doit créer un fichier de configuration 'xxx.cdb', où seront définis les objets utilisés par l'outil DSP/BIOS. Ce fichier permet aussi de faciliter l'organisation de la mémoire et la gestion du vecteur des interruptions, en offrant la possibilité de les faire sur un environnement visuel via la section de gestion de la mémoire MEM (Memory Section Manager), et via HWI (Hardware Interrupt Service Routine Manager) pour les interruptions.
- JTAG (Joint Team Action Group) et RTDX (Real Time Data Exchange) (voir Figure suivante): le RTDX permet un échange de données en temps réel entre l'hôte (PC par

exemple) et la destination (la carte DSK dans notre cas), il permet aussi l'analyse et la visualisation des données au cours de l'exécution du programme, alors que le lien JTAG est utilisé pour atteindre l'émulateur (qui se trouve à l'intérieure du DSP), c'est ce dernier qui permet au CCS de contrôler en temps réel l'exécution du programme [3].

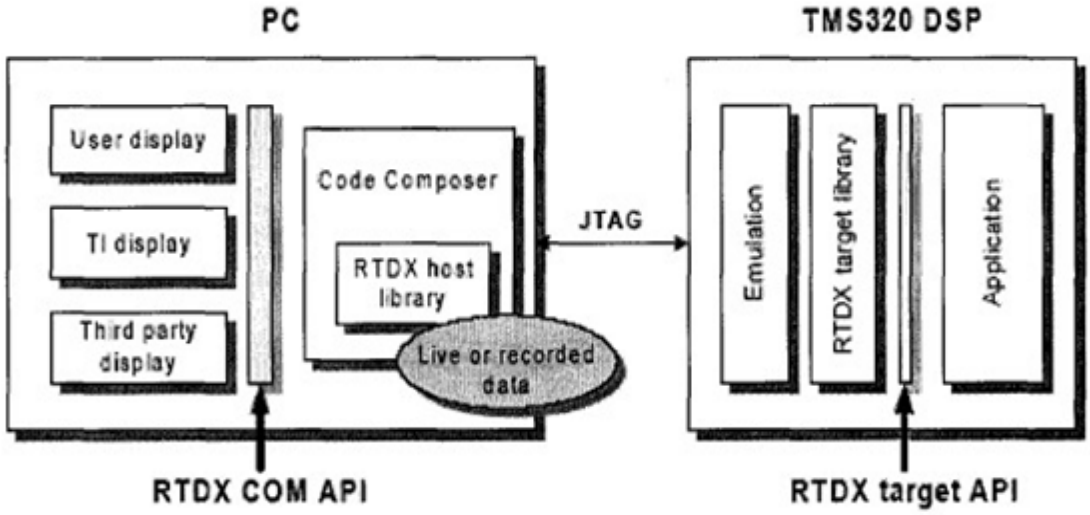


Figure 2.8- L'outil JTAG

- Simulateur intégré : le logiciel Code Composer Studio offre la possibilité de tester des programmes pour DSP sans utiliser la carte DSK à l'aide d'un simulateur intégré.

Remarque : Grâce à une collaboration entre Texas Instruments Inc. et MathWorks Corp, on pourrait de plus configurer la carte à l'aide le Matlab/Simulink. Cette collaboration a réduit immensément la complexité de programmation de la carte.
Dans le cadre de ce projet, nous avons préféré d'utiliser le langage C grâce à sa portabilité par les différents fabricants des cartes DSP (ce n'est pas le cas pour la programmation avec Matlab/Simulink).

Lancement du CCS

Avant de lancer le CCS, il est impératif de suivre dans l'ordre les étapes suivantes,

1. S'assurer que le Code Composer Studio (CCS) est fermé.
2. Mettre sous tension la carte en connectant le jack d'alimentation. Si elle était déjà sous tension, débrancher puis rebrancher le jack d'alimentation.
3. Vérifier que la carte s'initialise correctement : l'ordinateur doit signaler la connexion USB et les 4 diodes groupées présentes sur la carte doivent clignoter après une quinzaine de secondes et rester allumées.
4. Utilisez l'utilitaire de diagnostic pour tester la fonctionnalité de la carte, pour le faire, on procède comme suit :
   - Appuyer sur l'icône « 6713 DSK Diagnostics Utility v3.1 » qui se trouve sur le bureau.
   - Appuyer sur START pour lancer le diagnostic.

- Vérifier qu'à la fin du diagnostic que tous les indicateurs soient en vert comme le montre la figure suivante,

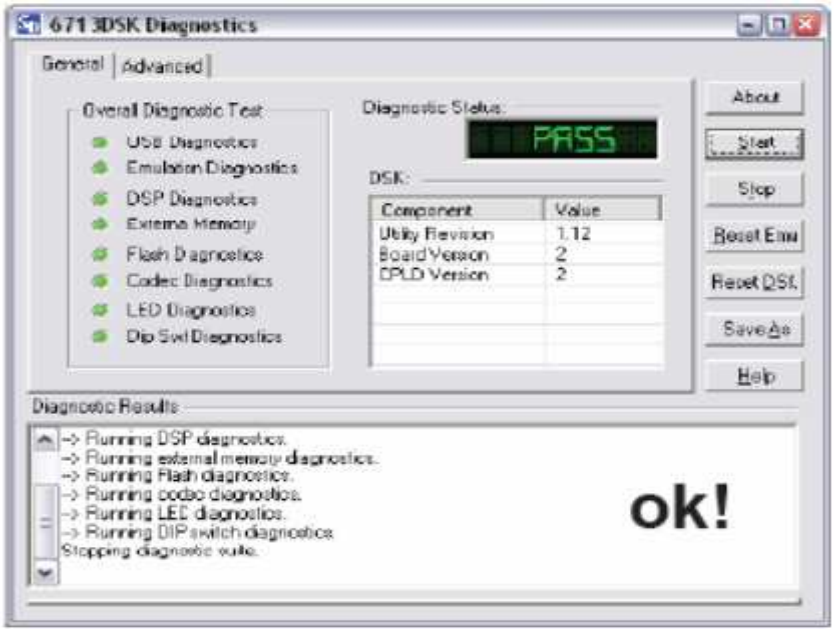


Figure 2.9– la fenêtre de la fin du diagnostic de la carte

5. Lancer le Code Composer Studio en cliquant sur « 6713 DSK CC Studio 3.1 » qui se trouve par défaut sur le bureau.
6. Établir la connexion avec la carte en cliquant sur « Debug » dans le menu principal et en choisissant « Connect ». On peut vérifier le statut de la connexion qui s'affiche dans le coin gauche de la fenêtre.

Programmation en C

La réalisation d'une application avec CCS se fait par la création d'un projet (un fichier avec extension .pjt) suivi de la configuration des différentes options nécessaires à son exécution. Dans un projet, on retrouve plusieurs fichiers regroupés selon leurs types dans des répertoires différents, ces répertoires sont,

- Include: contient les fichiers de l'en-tête *.h.
- Libraries : contient les fichiers librairies *.lib.
- Source : contient les fichiers sources du projet, ces fichiers peuvent être des programmes en langage C, C++, ou/et des programmes en assembleur * .asm.

Dans ce travail, nous nous sommes contentés de l'utilisation du langage C.

2.4.2.1 Création du fichier exécutable

Pour créer le fichier exécutable qui sera chargé sur le DSP, il faut passer par plusieurs étapes :

1. Dans le menu principal, on clique sur « Project », et on choisit « New ». Dans cette fenêtre, on donne un nom pour notre projet, sa location sur le disque dure de

l'ordinateur, son type (exécutable .out ou librairie .lib) et la carte DSP (Target) utilisée (C6713 pour notre cas).

2. Il faut écrire le programme principal en C ou en Assembleur en cliquant dans le menu principal sur « File » puis « new ».
3. Il faut joindre tous les fichiers indispensables (ne pas oublier le programme principal) pour le bon fonctionnement du programme, pour le faire, on doit cliquer dans le menu principal sur « Project », puis « add files to project », et finalement sur « open ».
4. Joindre aussi le fichier de commande (.cmd ou .lcf). Ce type de fichier divise la mémoire de la carte en des sections.
5. Avant de générer le code, il faut configurer les options de compilation et de lien (Project → build options) tel qu'elle est illustrée par la figure suivante,

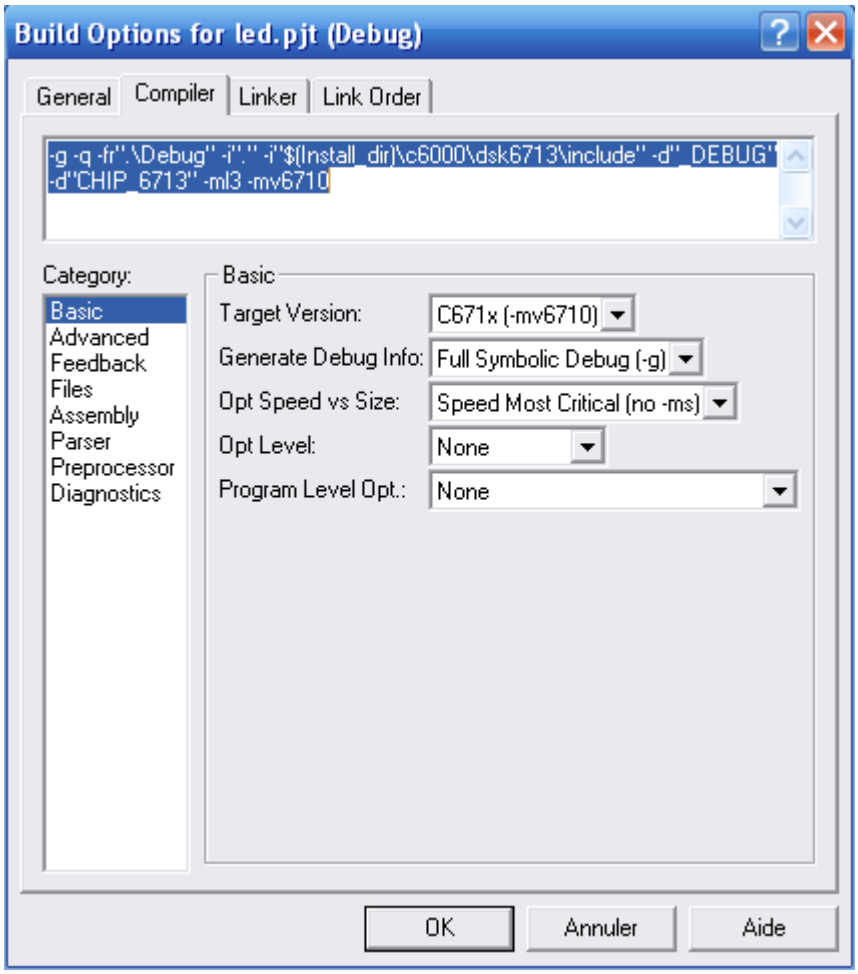


Figure 2.10- Fenêtre de configuration des options de compilation et de lien.

6. Après l'étape de configuration, on construit le projet (Project →Rebuild All) qui génère le fichier exécutable avec l'extension .out. Ce dernier est ensuite chargé sur le DSP (File →Load) et finalement, il ne reste qu'à exécuter le programme (Debug→ Run).

2.4.2.2 Fichiers supports

Les fichiers supports suivants sont utilisés pour les exemples traités ainsi que notre projet principal,

1. C6713dskinit.c: contient les fonctions initialisant le DSK, le codec, les ports séries et les entrées/sorties. Il n'est pas inclus avec CCS.
2. C6713dskinit.h: fichier en-tête avec des fonctions prototypes. Des fonctionnalités telles que celles utilisées pour sélectionner l'entrée micro au lieu d'entrée-ligne (par défaut), le gain d'entrée, et ainsi de suite, elles sont toutes obtenues à partir de ce fichier.
3. C6713dsk.cmd: fichier de commande. Ce fichier peut être modifié lors de l'utilisation de la mémoire externe à la place de la mémoire interne.
4. Vectors_intr.asm: une version modifiée d'un « vector file » inclus avec le CCS pour gérer les interruptions. Douze interruptions, INT4 à INT15, sont disponibles, INT11 est sélectionnée pour ce « vector file». Elles sont utilisées pour les programmes d'interruption pilotés.
5. Vectors_poll.asm: c'est un « vector file» pour les programmes utilisant le polling.
6. rts6700.lib, dsk6713bsl.lib, csl6713.lib: run-time, la carte, et bibliothèques supports pour la puce, respectivement. Ces fichiers sont inclus avec le CCS. Ces trois fichiers d'extension .lib sont localisés dans : C6000\cgtools\lib, C6000\dsk6713\lib, et c6000\bios\lib, respectivement.

Remarque : Pour ne pas perdre le temps dans la recherche de ces fichiers à chaque fois que nous créons un projet, nous les avons rassemblé tous dans un même dossier.

Chapitre 3 - implémentation d'un filtre FIR

Avant d'entamer l'implémentation du filtre FIR sur le DSP, nous avons proposé deux programmes pour tester les outils DSK et ceci dans le but de bien comprendre les étapes citées dans le chapitre précèdent consacré à la création d'un fichier exécutable. En effet, le premier exemple est un programme servant comme un générateur des sinusoïdes que nous pouvons les visualiser sur un oscilloscope ou bien sur le logiciel CCS lui-même. Le deuxième exemple est un programme d'un filtre passe-tout qui nous a beaucoup servis pour l'utilisation des différentes entrées/sorties ainsi que la vérification de leur bon fonctionnement.

exemples test de la carte tms320c6713 dsk

Exemple 1 : sine8_buf

Cet exemple génère des sinusoïdes de 1 KHz de fréquence avec huit points. Le plus important dans ce programme est la capacité de CCS de tracer les courbes dans le domaine temporel et fréquentiel.

Le programme principal sine8_buf.c (programme en langage C) de cet exemple (voir annexe A), crée une mémoire tampon pour y stocker les données de sortie, ces dernières nous servent pour tracer les graphes dans le domaine temporel et fréquentiel.

Pour y arriver, il faut passer par les étapes suivantes,

1. Créer un projet (sine8_buf.pjt) et préciser sa location et la carte utilisée (c671x pour notre cas).
2. Joindre tous les fichiers indispensables à savoir,
   - Le programme source sine8_buf.c et le fichier source C6713dskinit.c.
   - Les bibliothèques : rts6700.lib, dsk6713bsl.lib, csl6713.lib.
   - Le fichier de commande : C6713dsk.cmd.
   - Le fichier vectors_intr.asm qui gère les interruptions.
3. Sélectionner (Project → Scan All File Dependencies), cela rajoute tous les fichiers en-tête dépendant du projet.
4. Configurer les options de compilation et de lien (Project → build options),
   - Appuyer sur « Compiler » puis « Basic » et choisir,
     target version=C671x(-mv6710).
   - Appuyer dans la même fenêtre sur « advanced » et choisir,
     Memory Models=Far(-mem_model :data=far).
   - De même appuyer sur « Preprocesseur » et régler,
     - ***Pre-Define Symbol=CHIP_6713*** (attention CHIP en majuscule)
     - ***Include Search Path= C:\CCStudio_v3.1\C6000\dsk6713\include***

- Maintenant, appuyer sur le menu « linker » dans la même fenêtre, puis indiquer l'emplacement des librairies. Pour notre cas, nous les avons rassemblé dans un dossier 'comme cité ci-dessus' sur le bureau nommé 'support'. En effet, spécifier le chemin, ***library Search Path=***

  C:\DocumentsandSettings\Administrateur\Bureau\support\lib
- Finalement, appuyer sur OK.

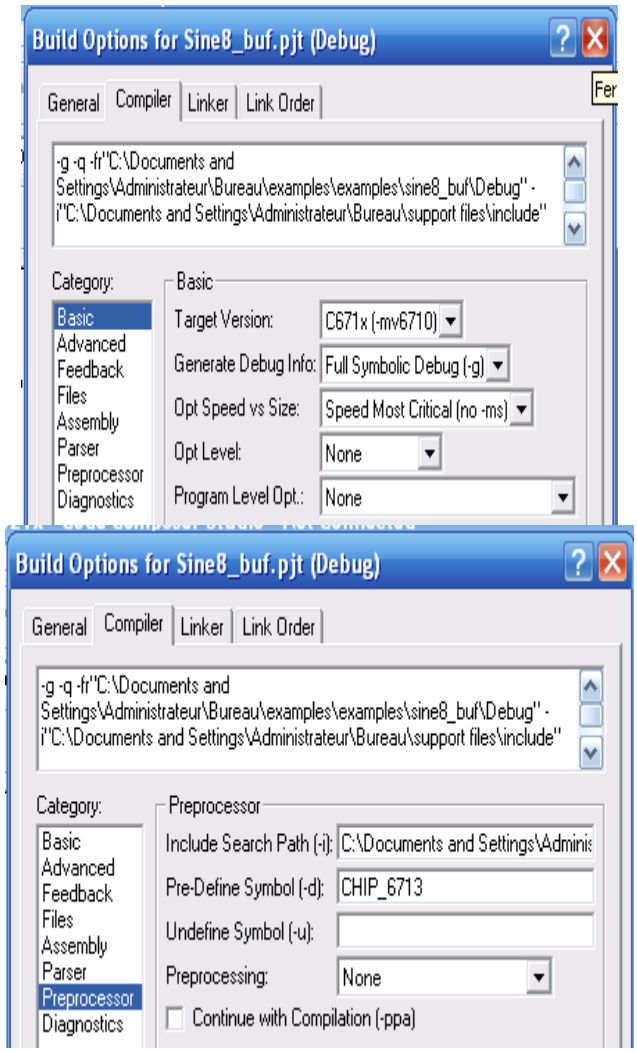


Figure 3.1-configuration des options de la compilation

5. Construire le projet (Project →Rebuild All) qui génère le fichier exécutable 'sine8_buf.out'.
6. Charger ce fichier exécutable sur le DSP (File →Load).

Afin de visualiser le signal de sortie, nous devons connecter la sortie 'line out' de la carte à un oscilloscope, le résultat obtenu est le suivant,

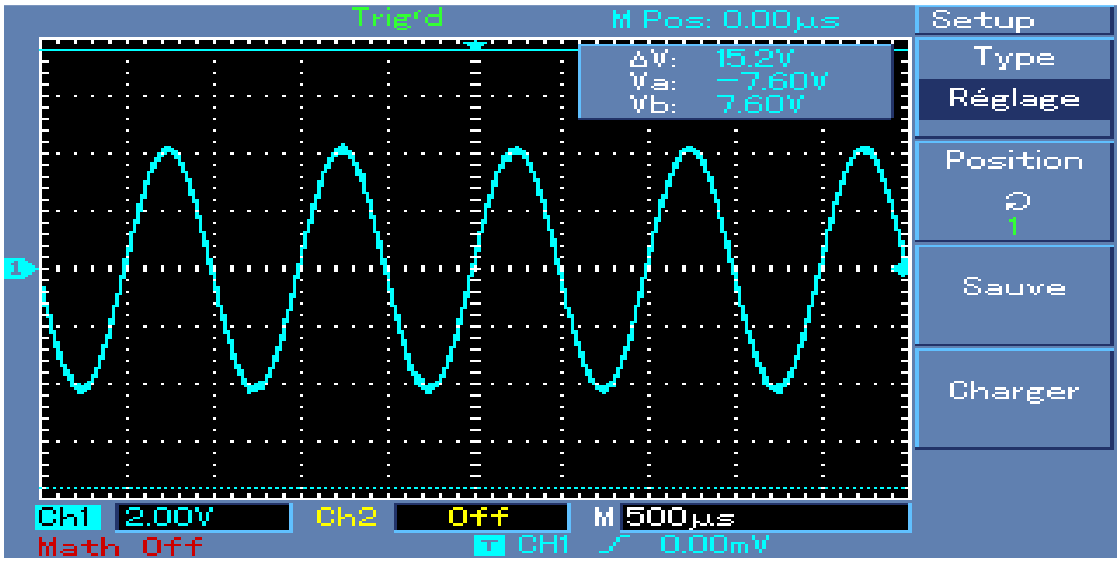


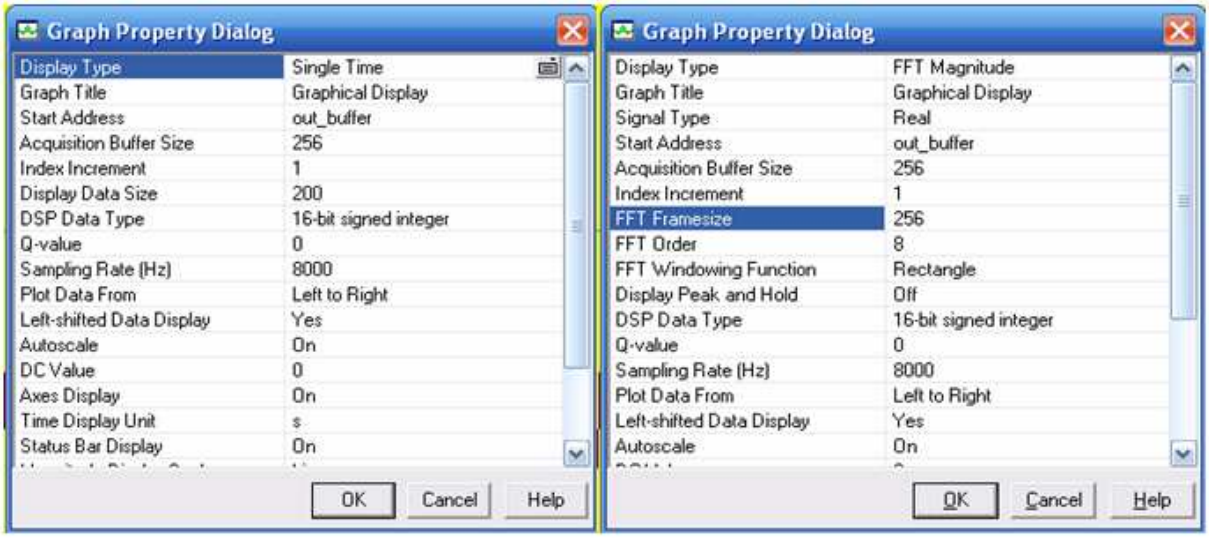
Figure3.2- le signal sinusoïdale généré par le DSK

- La tension crête à crête → V_{cc}=0.8 x gain =0.8 x 10= 8 V. On note qu'on fixé le dans le programme à 10 et la tension de sortie par défaut est de 0.8 V.
- La fréquence : f=1 kHz.

Afin de visualiser le même résultat sur le CCS dans le domaine temporel et/ou dans le domaine fréquentiel on procède comme suit,

1. Sélectionner (View → Graph puis 'Time' pour le domaine temporel ou 'Frequence' pour le domaine fréquentiel).

2. Configurer les paramètres comme le montre la figure suivante :



(1) (2)

Figure 3.3 - Les paramètres pour le traçage des graphes dans le domaine temporel (1) et fréquentiel (2).

Les résultats obtenus sont présentés ci-dessous,

1. Dans le domaine temporel :

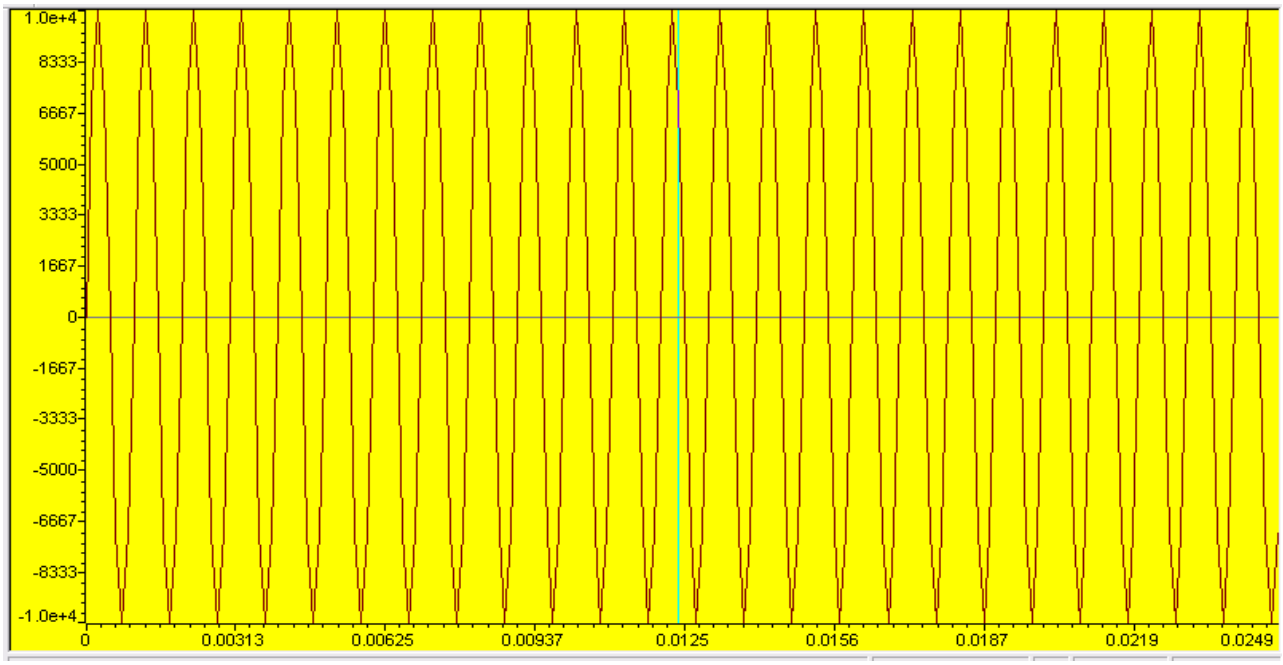


Figure3.4- graphe d'une onde sinusoïdale de 1 kHz dans le domaine fréquentiel avec le CCS

2. Dans le domaine fréquentiel :

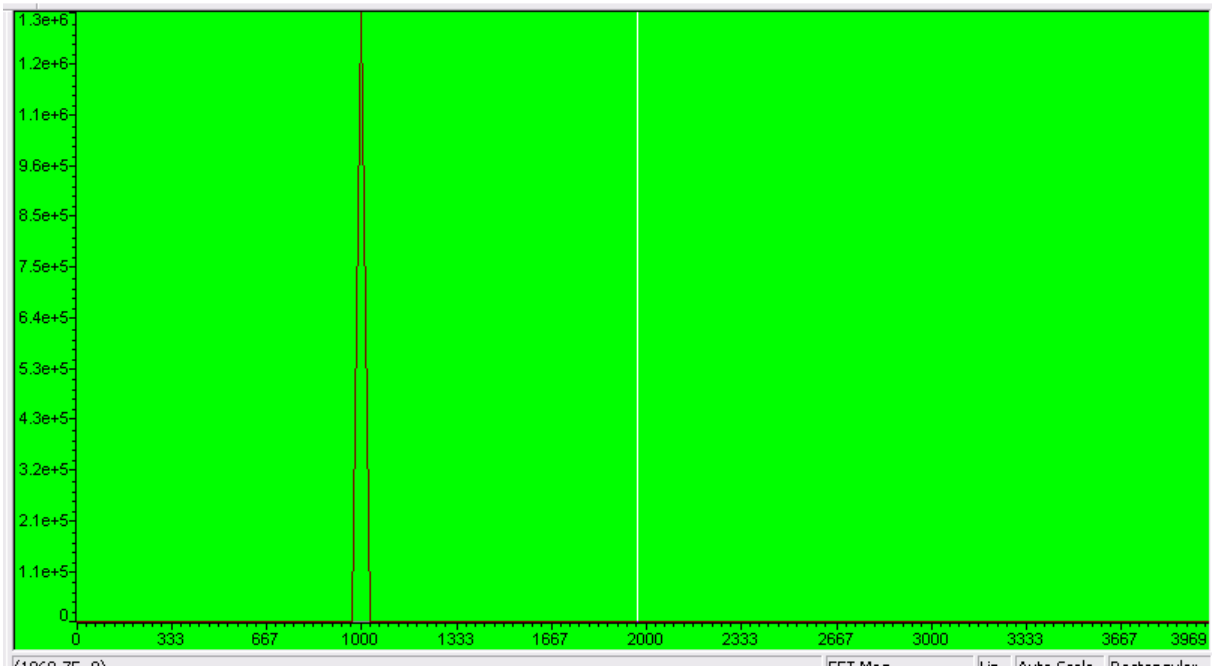


Figure3.5- graphe d'une onde sinusoïdale de 1 kHz dans le domaine fréquentiel avec le CCS

Exemple 2 : filtre passe-tout

Ce deuxième exemple nous a permis de vérifier le bon fonctionnement des différentes entrées/sorties ainsi que leur utilisation.

Voir le programme principal en langage C dans l'annexe B.

Pour créer le fichier exécutable, il faut suivre les mêmes étapes citées dans l'exemple précèdent.

Afin de vérifier le bon fonctionnement de ce filtre, il y a deux méthodes,

1. La première consiste à attaquer la carte par un signal vocal à travers l'entrée 'MIC IN' et récupérer la sortie à l'aide d'un écouteur branché dans la sortie 'HEADPHONE', dans ce cas il faut changer la valeur initiale du registre 4 0x0011 par 0x0015 dans le fichier en-tête c6713dskinit.h.
2. La deuxième méthode est la plus pratique pour la visualisation des signaux de l'entrée et la sortie. En effet, à l'aide d'un GBF, nous attaquons l'entrée 'LINE IN' par un signal sinusoïdal dont la tension crête à crête ne doit pas dépasser 6 V, la sortie sera récupérée à l'aide de l'oscilloscope en le connectant à la sortie 'LINE OUT' de la carte.

Le résultat obtenu est présenté dans la figure 3.6.

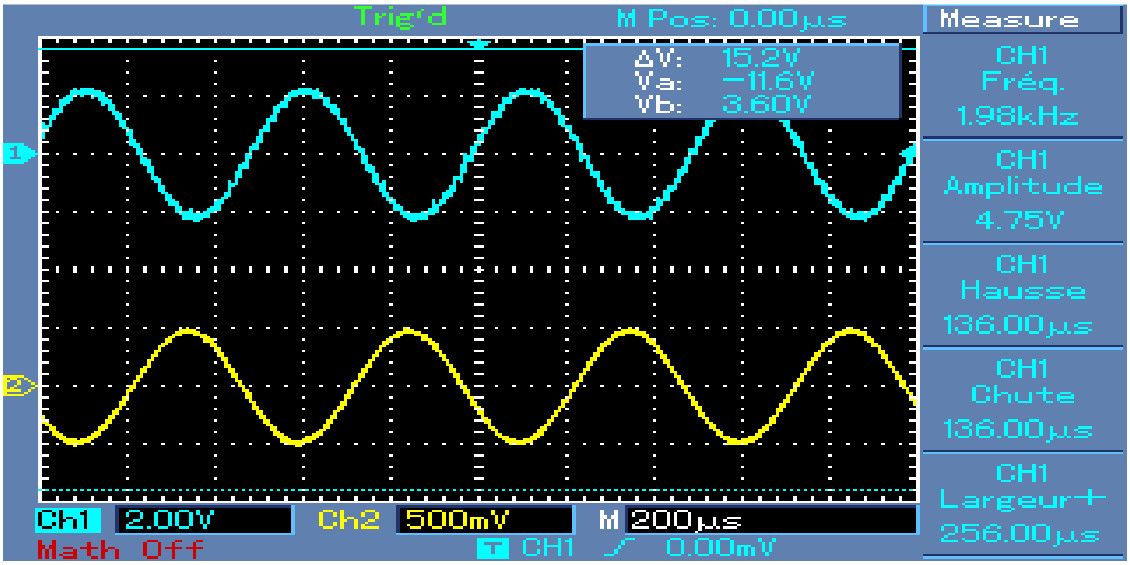


Figure3.6- les signaux d'entrée et sortie pour le filtre passe-tout

Nous constatons que le signal de sortie (en bleu) a la même fréquence que celui de l'entrée (en jaune).

Remarques :

- Le signal d'entrée et de sortie n'ont pas les mêmes amplitudes, pour remédier à ce problème, il faut changer la valeur initiale du registre 0 dans le fichier c6713dskinit.h 0x0017 par 0x001c.

- Le signal de sortie est déphasé par rapport au celui de l'entrée, cela est dû au temps de réponse des différentes composantes de la carte.

Filtres numériques

Les filtres numériques présentent par rapport aux filtres analogiques les avantages-inconvénients suivants :

Avantages :

- Les filtres numériques sont insensibles aux conditions extérieures (chaleur, humidité, etc.)
- Certains filtres numériques sont impossibles à réaliser de manière analogique (exemple: les filtres RIF).
- Les filtres numériques ne sont pas sensibles aux non idéalités d'un $10^{ème}$ amplificateur opérationnel. Ainsi, un filtre RII du 10 ordre est tout à fait envisageable (attention quand même au bruit de calcul !).
- La problématique du bruit change d'aspect: dans le filtrage numérique on parle de 'bruit de quantification' et de 'bruit de calcul'. Le premier est lié au nombre de bits employés pour la quantification (8 bits, 16 bits, etc.). Le second est négligeable si l'unité de calcul est de type 'floating point'. De toute manière, le bruit numérique est localisé: on sait d'où il vient et il reste stable.

Inconvénients :

- Les filtres numériques nécessitent un filtrage analogique anti-repliement à l'échantillonnage et à la restitution.
- Les performances d'un filtre sont directement proportionnelles à la puissance de l'unité de calcul (processeur ou DSP).
- Beaucoup de problèmes peuvent apparaître si l'unité de calcul est de type 'fixed point'. Les paramètres d'un filtre nécessitent parfois une double précision pour être opérationnels, ce qui ralentit les performances.

On trouve essentiellement deux grandes familles de filtres numériques :

A. Filtres RIF : Ces filtres non récursifs n'ont pas de contre-réaction.

Avantages :

- Toujours stables.
- Phase linéaire si symétrie des coefficients $\Rightarrow$ pas de distorsion de phase.
- Possibilité de réaliser toutes sortes de filtres en dessinant simplement des gabarits de réponse en amplitude $H(f)$ et en calculant la transformée de Fourier inverse $h(t)$.

Inconvénients :

- Beaucoup de calculs par rapport à un RII équivalent au niveau des performances.
- Le retard du filtre (de groupe ou de phase) peut être important si le nombre de 'taps' est élevé.

B. Filtres RII : Ces filtres récursifs ont une contre-réaction.

Avantage :

- Beaucoup moins de calculs par rapport à un RIF équivalent au niveau des performances.

Inconvénients :

- Il faut vérifier la stabilité.
- Phase non linéaire → distorsion de phase.

Dans le cadre de notre projet, nous avons choisi l'implémentation d'un filtre FIR vu les avantages cités ci-dessus ainsi que la facilité de manipulation.

Filtre à réponse impulsionnelle finie

Equation aux différences :

La sortie est une combinaison linéaire d'un ensemble fini d'éléments d'entrées :

$$y(n)= \sum_{k=0}^{N-1} h(k)x(n-k) \qquad |h(n)| < \infty,$$

où, h(n) est la réponse impulsionnelle du filtre et N est sa longueur.

Le gabarit réel d'un filtre FIR est le suivant,

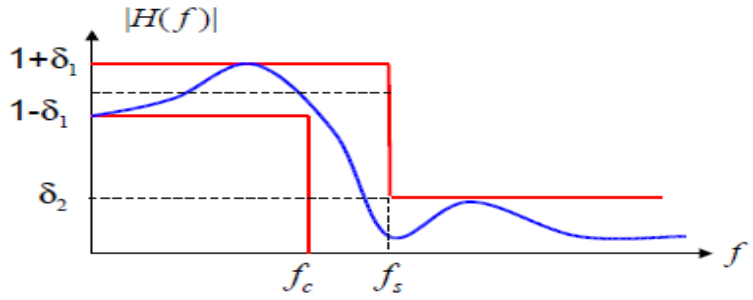


Figure 3.7 : Gabarit réel de la réponse fréquentielle d'un filtre FIR.

Le filtre est caractérisé par,

- La bande passante BP.
- La bande atténuée (ou coupée).

- La largeur ΔF= fs - fc de la zone de transition.
- L'amplitude des oscillations en bande passante δ_1.
- L'amplitude des ondulations en bande atténuée δ_2.

Pour calculer les coefficients de la réponse impulsionnelle, on a recours à deux méthodes : la première est celle du calcul manuel, qui demande beaucoup de temps, et que nous ne traitons pas dans ce rtravail, tandis que la deuxième méthode est celle du calcul à l'aide de l'outil 'fdatool' de Matlab.

Pour y arriver, il nous faut qu'à taper dans la fenêtre de commande Matlab 'fdatool', cette icommande permet d'ouvrir la fenêtre suivante,

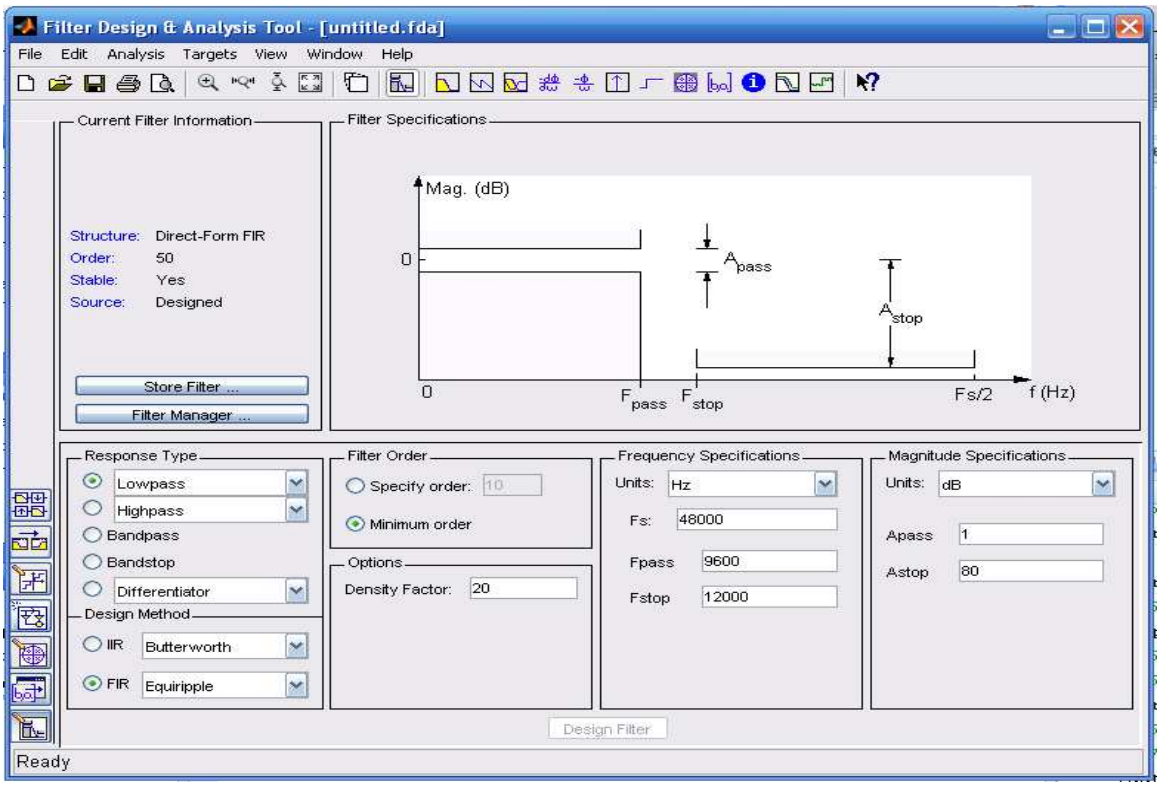


Figure 3.8 - la fenêtre 'fdatool' de MATLAB.

A ce niveau, nous précisons le type, les caractéristiques (citées ci-dessus), la méthode de conception du filtre,…

Sélectionner ensuite « Design Filter », puis (Target →Generate C Header…) pour générer les coefficients de la réponse impulsionnelle dans un fichier en-tête.

Implémentation d'un filtre FIR passe-bas

Il s'agit de l'implémentation d'un filtre FIR passe-bas de fréquence de coupure Fc=1500 Hz.

Algorithme

Pour implémenter ce filtre FIR sur le DSP qui fait objet de notre étude, il faut réserver une partie de la mémoire pour stocker les N coefficients de la réponse impulsionnelle, une autre partie sera consacrée pour les échantillons de l'entrée. Cette dernière devra mettre à jour à chaque réception d'une nouvelle donnée x(n). En effet, chaque nouvel échantillon doit écraser l'avant dernière en gardant la même taille du tableau comme le montre le schéma suivant :

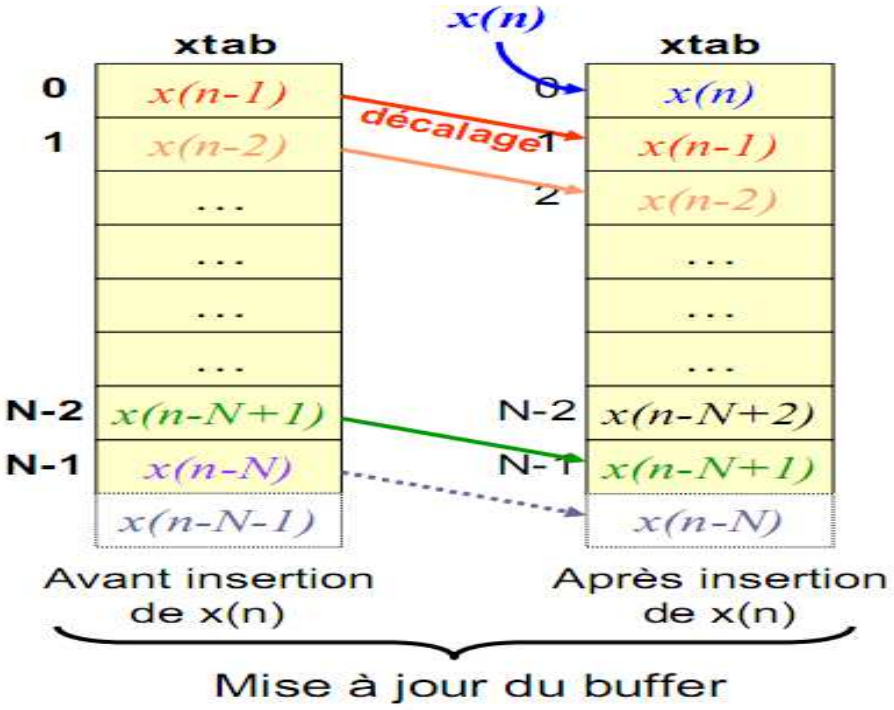


Figure 3.9- mémoire de donnée pour un filtre FIR.

Simulation

Pour tester ce filtre et s'assurer de son bon fonctionnement, nous avons passé par les étapes suivantes :

a) La création d'un projet nommé « filtre_fir.pjt » en CCS : pour y arriver, il faut joindre au programme principal en langage C « filtre_fir.c » (voir annexe C), les fichiers suivants (dont l'utilité est déjà mentionnée dans le chapitre précédent), c6713dskinit.c, Vectors_intr.asm, c6713dsk.cmd, rts6700.lib, dsk6713bsl.lib et csl6713.lib.
   Construire le projet (Project →Rebuild All) qui génère le fichier exécutable 'filtre_fir.out' et finalement charger ce fichier exécutable sur le DSP (File →Load).

b) La création d'un script MATLAB pour la génération d'une somme de deux signaux sinusoïdaux, la première sinusoïde a une fréquence de 1 kHz et la deuxième de 2 kHz (le signal qui sera filtré) (voir script dans l'annexe).

c) Relier l'entrée 'LINE IN' du DSK avec le PC à l'aide d'un câble JACK.

d) Relier la sortie de la carte à un oscilloscope pour visualiser le signal de sortie.

e) Lancer le programme.

Le résultat obtenu sur l'oscilloscope est présenté au niveau de la figure 3.10.

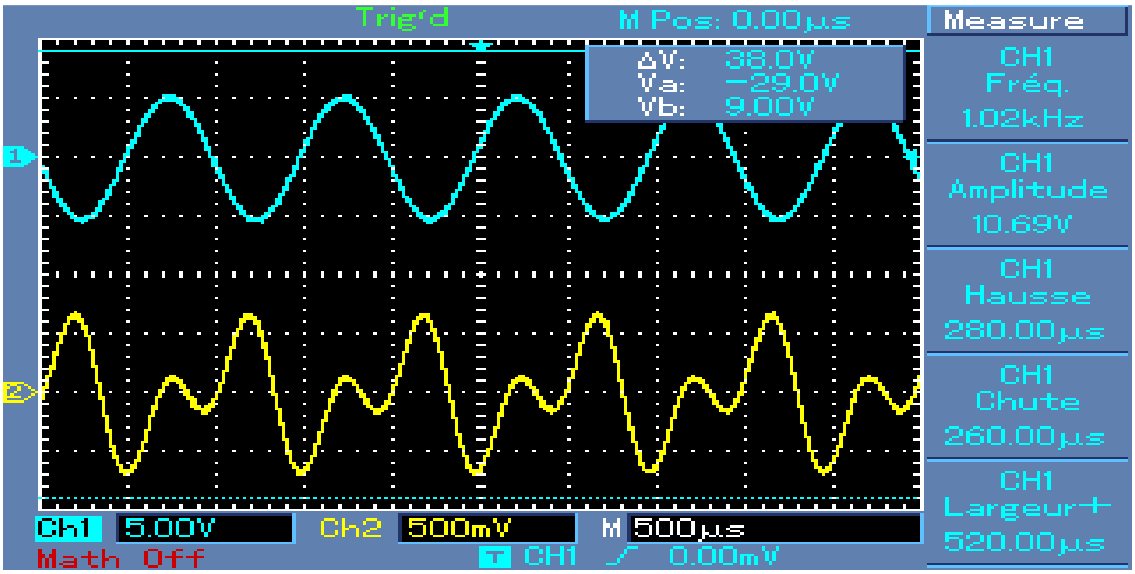


Figure 3.10 - Résultat du filtrage.

La sortie de la carte (signal en bleu) est un signal de fréquence de 1.02 kHz, d'où la validité du résultat escompté. En effet, le filtre a laissé passer le premier signal (de fréquence de 1KHz) et il a rejeté le deuxième (de 2 KHz).

Afin de s'assurer de la validité de la réponse du filtre implémenté sur le DSP, nous avons fait une comparaison entre sa réponse et celle obtenue théoriquement avec l'outil 'fdatool' de MATLAB. En effet, nous avons attaqué la carte par un signal sinusoïdal à partir d'un GBF dont la fréquence varie autour de la fréquence de coupure, les résultats obtenus sont présentés dans le tableau et la figure suivants,

f (Hz)	400	700	1020	1120	1430	1480	1500	1520	1540	1560
V_e (volt)	2	2	2	2	2	2	2	2	2	2
V_s (Volt)	2.0	2.0	2.0	1.97	1.97	1.85	1.67	1.52	1.40	1.11
V_s / V_e	1.0	1.0	1.0	0.985	0.985	0.925	0.835	0.760	0.700	0.555

f (Hz)	1580	1590	1610	1620	1660	1680	1700	1720	1740	1760
V_e (volt)	2	2	2	2	2	2	2	2	2	2
V_s (Volt)	0.88	0.77	0.60	0.43	0.11	0.07	0.02	0.005	0.001	0.003
V_s / V_e	0.440	0.385	0.300	0.215	0.055	0.035	0.010	0.002	0.000	0.000

f (Hz)	1780	1800	1860	1900	2000	2100	2200	2300	2400	2500

V_e (volt)	2	2	2	2	2	2	2	2	2	2
V_s (Volt)	0	0	0	0	0	0	0	0	0	0
V_s / V_e	0	0	0	0	0	0	0	0	0	0

Tableau 3.1 – la variation du rapport Vs/ Ve en fonction de la fréquence du GBF

Avec, **f**=fréquence, V_e= la tension de l'entrée et V_s= la tension de la sortie

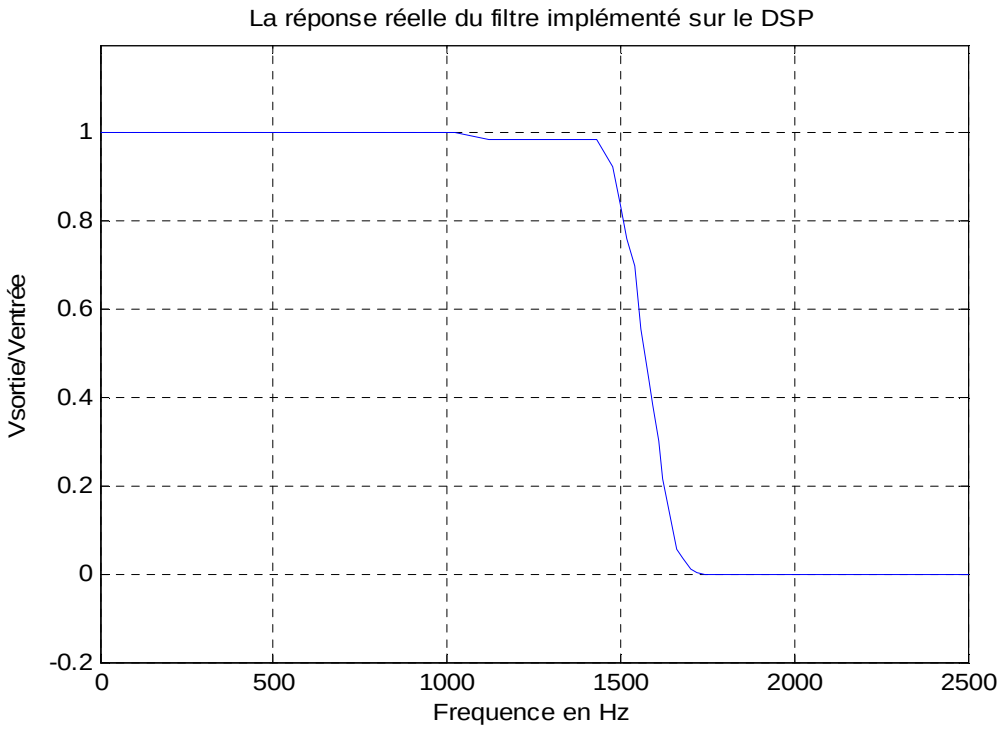


Figure 3.12 – la réponse du filtre FIR de fréquence de coupure de 1500Hz

La réponse du filtre qui nous a généré les coefficients de la réponse impulsionnelle est la suivante,

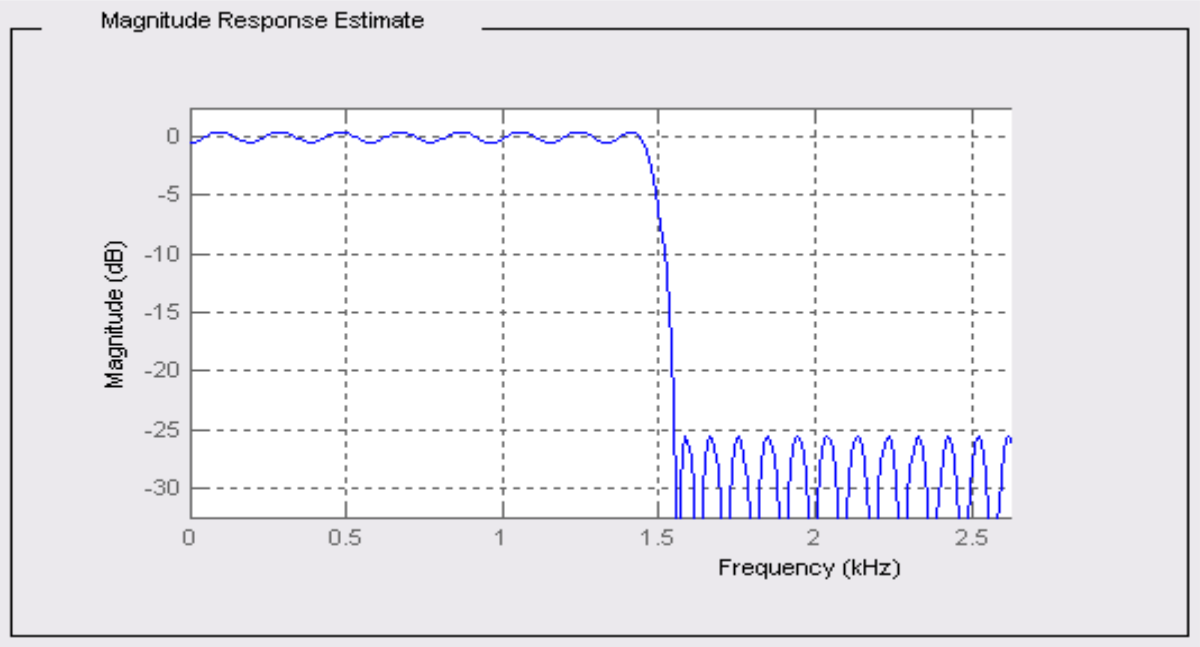


Figure 3.12 – la réponse du filtré désigné par 'fdatool' dans MATLAB

Nous constatons qu'il y a une ressemblance entre la réponse théorique et pratique du filtre implémenté sur le DSP, au niveau de la bande passante et de la bande de transition.

Conclusion

Durant la période de ce stage, nous avons pu constater que les DSP ne représentent qu'une partie du traitement numérique du signal, mais il reste essentiel par son rôle important dans la chaîne de traitement.

Le travail réalisé au cours de cette période nous a permis d'élargir nos compétences dans le domaine du traitement numérique du signal, ainsi que la découverte du fonctionnement et de l'architecture des DSP constituant un élément clé dans le TNS.

Comme perspectives, nous recommandons de faire la programmation sur DSP en utilisant le langage assembleur, vu ses innombrables avantages par rapport à la programmation en langage C. nous recommandons également l'implémentation des algorithmes plus complexes sur le DSP, par exemple,

les algorithmes de codage canal, de compression et des modulateurs numériques : ASK, PSK, FSK, OFDM,…

Pour conclure, maîtriser ou avoir des bases sur l'utilisation du DSP et le traitement numérique du signal devient indispensable vu le nombre de domaines concernés de nos jours.

Annexe A : Programme en langage C ‘sine8_buf.c’ [3]

```
//sine8_buf  générateur des sinusoïdes. Tampons sortie tracées avec le CCS
#include "dsk6713_aic23.h" // fichier support pour le codec et le DSK
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;  // fréquence d’échantillonnage
int loop = 0;                                          // index de la table
short gain = 10;                                       //Le gain
short sine_table[8]={0,707,1000,707,0,-707,-1000,-707};//valeurs des sinusoïdes
short out_buffer[256];                  // Tampon sortie
const short BUFFERLENGTH = 256;         // taille de mémoire tampon de sortie
int i = 0;                                             // pour le nombre de tampons
interrupt void c_int11()                //interrupt service routine
{
 output_left_sample(sine_table[loop]*gain);  //sortie sinusoïdale
 out_buffer[i] = sine_table[loop]*gain;      // mettre en mémoire tampon de sortie
 i++;                                        //incrémenter le compteur des tampons
 if(i==BUFFERLENGTH) i=0;                          // compteur
 if (loop < 7) ++loop;
 else loop = 0;                          // réinitialiser l’index de la table
 return;
}
void main()
{
 comm_intr();                                      //init DSK, codec, McBSP
 while(1);                               //boucle infinie
}
```

Annexe B : Programme principal en langage C du filtre passe-tout

```
//passe_tout.c. programme en c d’un filtre passe tout

#include "dsk6713_aic23.h"              // fichier support pour le codec et le DSK
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; // fréquence d’échantillonnage
interrupt void c_int11()        //interrupt service routine
{
  short sample_data;
  sample_data = input_sample(); //échantillons d’entrée
  output_sample(sample_data);   // échantillons de sorties
  return;
}
```

```
void main()
{
  comm_intr();                 //init DSK, codec, McBSP
  while(1);                    //boucle infinie
}
```

Annexe C : Programme principal en langage C du filtre FIR passe-bas de fréquence de coupure 1500Hz

```
#include "dsk6713_aic23.h"                    // fichier support pour le codec et le DSK
#define Nbr_coeff 81

Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; // fréquence d'échantillonnage
int sorie = 0;                                  //initialiser la sortie du filtre
short h[Nbr_coeff]=
{
-6, 13, 42, 52, 14, -40, -45, 18, 74, 35, -70, -100, 12, 137, 89, -105, -192, -14,
228, 188, -142, -338,-76,361, 361, -179, -579, -208, 569, 685, -209, -1040, 519, 992,
1496,
 -229, -2492, -1827, 3237, 9832, 12872, 9832, 3237, -1827, -2492, -229, 1496, 992, -
519,
 -1040, -209, 685, 569, -208, -579, -179, 361, 361, -76, -338, -142,188,228, 14, -192, -
105, 89, 137, 12, -100, -70, 35, 74, 18, -45, -40, 14, 52, 42, 13, -6};

short X_retard[Nbr_coeff];                  //échantillons retardés

interrupt void c_int11()     //ISR
{
    short i;

    X_retard [0]=input_sample();                  //le nouvel échantillon d'entrée
    sortie = 0;                  // initialiser la sortie du filtre
    for (i = 0; i< N; i++)
      sortie =sortie+ (h[i] * X_retard [i]);  //sortie(n) += h(i)* x(n-i)
    for (i = Nbr_coeff-1; i > 0; i--)
      X_retard [i] = X_retard [i-1];      // déplacer les échantillons retardés
    output_sample(sortie >> 15);   //échantillon de sortie du filtre en format short
    return;
}

void main()
  {
    comm_intr();              //init DSK, codec, McBSP
    while(1);                //infinite loop
```

```
}
```

Annexe D: Script MATLAB pour la génération de la somme de deux signaux sinusoïdaux de fréquences 1kHz et 2kHz

```
F1=1000;     % fréquence du premier signal
F2=2000;     % fréquence du deuxième signal
fe=8000;      % fréquence de l'échantillonnage
N= 2^16;     % Nombre d'échantillons
temps=(0:N-1)/fe;    %axe temporel en seconde
 x1=sin(2*pi*F1*temps);   %premier signal
 x2=sin(2*pi*F2*temps);   %deuxième signal
 for u=1:100,
          sound(x1+x2) ;    %écouter le mélange des deux fréquences
 end
```

Bibliographie

[1] M. Correvon(2010). Introduction aux DSP orientés & applications industrielle - ***Notes du cours*** [Présentation PDF]. Repéré dans le site www.iai.heig-vd.ch.

[2]dsk6713_TechRef de TEXAS INSTRUMANT (2006), Datasheet du C6713 DSK.

[3] Rulph Chassaing(2005). Digital Signal Processing and Applications with the C6713 and C6416. USA, New Jersey: John Wiley & Sons, Inc.

[4] Sylvain MONTAGNY(2008). Microprocesseurs microcontrôleurs - *Notes du cours* [Présentation PDF]. Repéré dans le site *www.sfa.univ-savoie.fr*.

[5]TMS320 datasheet pdf datenblatt - Texas Instruments

WEBOGRAPHIE

[6]www.aix-mrs.iufm.fr/formations/filieres/ge/data/DSP

[7]fr.wikipedia.org/wiki/Processeur_de_signal_num%C3%A9rique