REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

PROJET DE FIN D'ETUDES

EN VUE DE L'OBTENTION DU DIPLOME D'INGENIEUR D'ETAT EN ELECTRONIQUE

Implémentation d'un système de communication FM par l'utilisation du DSP TMS320VC5402

Proposé et Dirigé par:

Mr Z.TERRA

Etudié par :

LOUNIS Ahmed Abd elkarim

Année universitaire 2004 - 2005

E.N.P 10 Avenue Hassen Badi EL HARRACH - ALGER

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

PROJET DE FIN D'ETUDES

EN VUE DE L'OBTENTION DU DIPLOME D'INGENIEUR D'ETAT EN ELECTRONIQUE

Implémentation d'un système de communication FM par l'utilisation du DSP TMS320VC5402

Proposé et Dirigé par:

Mr Z.TERRA

Etudié par :

LOUNIS Ahmed Abd elkarim

Année universitaire 2004 - 2005

E.N.P 10 Avenue Hassen Badi EL HARRACH - ALGER

ملخص:

يعرض هدا العمل ادماج نظام اتصال مضمن التواتر (FM) على لوحة DSP TMSP320VC5402 لـ: Texas instruments . بعد تقديم عرض للخصائص البرمجية و المادية للوحة. تم تطوير تقنية الادماج للمحاكات على الكود كومبوزر (*Code Composer*) وتشغيله خلال الزمن الحقيقي على جهاز التطوير DSKVC5402.

كلمات مفتاحية: كمبيوتر المعالجة الرقمية للاشارة (DSP) جهاز التطوير (DSK)، تضمين التواتر (FM)، تضمين التواتر ذو الحزمة المحدودة(NFM) ، حلقة التحكم الطوري (PLL).

Résumé :

Ce travaille présente l'implémentation d'un modulateur démodulateur FM sur le DSP TMS320VC5402 de *Texas Instruments*. Après une présentation des différentes caractéristiques logicielles et matérielles du DSP cible. On a décrit la technique d'implémentation sur le DSP pour la simulation sur le *Code Composer* et l'exécution en temps réel sur le Kit de développement DSK VC5402.

Mots clés :processeur de traitement numérique de signal(DSP), le Kit de développement (DSK), Modulation en fréquence (FM), Modulation en fréquence à bande étroite (NFM), Boucle de Phase Asservie (PLL).

Abstract:

this work present the implementation of a modulator and a demodulator FM on the DSP TMS320VC5402 of Texas Instruments. after a presentation of the various characteristics software and hardware of the target *DSP*. We described the technique of implementation on the *DSP* for simulation on the code Composer Studio and the execution in real time on the starter Kit DSKVC5402.

Keywords:Digital signal processing (DSP), Evaluation Module (DSK), frequency modulation (FM), Narrow band FM Modulation, (NFM), phase locked loop ((PLL).

Remerciements

Ce travail n'aurait pu se faire seul ! Ce sont les compétences, la disponibilité, le dynamisme et la bonne humeur de chacun, qui m'ont permis de poursuivre mes études et d'achever ce projet dans les meilleures conditions. C'est pourquoi je tiens chaleureusement à remercier ici :

Mon promoteur M. Z TERRA, enseignant et chercheur à l'Ecole Nationale Polytechnique d'Alger (ENP), pour ces conseils précieux, remarques et critiques pertinentes et patience infinie.

Tous ceux qui ont de prés ou de loin contribué à l'aboutissement et l'amélioration de ce travail.

En fin, j'adresse mes plus sincères remerciements à tous les enseignant qui ont contribué à ma formation.

Dédicace

Je dédie ce modeste travaille,

A la mémoire de mon très chère père.

A ma très chère mère qui durant toutes ces années d'études a su me soutenir par ses sacrifices et encouragements.

A mes sœurs Leila et Narimane.

A mon beau frère Mokhtar.

A ma grand-mère, et a toute la famille.

A mes amis et mes collègues.

Ahmed

Table des matières

Introduction Générale

Les systèmes de communication constituent un domaine très important d'application des processeurs de traitement numérique du signal (DSP : *Digital Signal Processor*). Ces DSP assurent dans ce domaine d'une manière efficace la conception, l'expérimentation et l'implémentation d'une variété d'algorithmes de traitement de signal pour des applications en temps réel. La modulation/démodulation FM constitue une opération très importante dans un système de communication analogique.

L'objectif de ce travaille est l'implémentation d'un modulateur FM à bande étroite (NFM), et d'un démodulateur FM à l'aide d'un circuit PLL sur le DSP C5402 de Texas Instruments.

Le présent PFE est organisé comme suit: Le premier chapitre comporte des généralités sur les DSP, les performances, les domaines d'application et les principales caractéristiques des composants de la famille TMS320C54x. Le deuxième chapitre décrit les outils de génération de code et de développement fournis par Texas Instruments où nous intéressons au *Code Composer Studio* et le Kit de développement DSK Vc5402. Le troisième chapitre est consacré à la présentation de la technique d'implémentation du modulateur /démodulateur FM pour la simulation sur le *Code Composer* et l'exécution en temps réel sur le Kit de développement DSK VC5402. Les principes de base, la constitution, le fonctionnement et l'évaluation des résultats sont représentés.

Généralités sur les DSP

1.1 Introduction

Dans ce chapitre nous exposons l'historique des DSP, ses principales caractéristiques et leurs applications dans les systèmes de communication, en mettant l'accent sur les DSP de la famille TMS320 C54x de *Texas Instruments*.

1.2 Historique

Lorsque les premiers microprocesseurs ont été conçus au début des années 1970, Il était hors de question de produire des circuits susceptibles de réaliser directement des opérations complexes comme la multiplication ou la division.

La solution utilisée alors, était de décomposer chaque instruction en une suite d'opérations simples, permise par le circuit interne du processeur. C'est ce que l'on appelle le microcodage. Avec ce système, chaque code machine correspond en fait à un micro programme, encodée dans le silicium. L'horloge du processeur détermine en général le nombre de cycles machines réalisées par seconds (par exemple il y a des instructions comme la multiplication, peuvent ainsi demander 36 cycles machines pour certains processeurs).

Une telle approche permet d'améliorer le jeu d'instructions, donc les processeurs disposant d'un jeu d'instructions évolué au prix du nombre de cycles machines par instruction ont été regroupé sous le nom de CISC (*Complex Instruction Set Computer*), alias calculateurs à jeu d'instructions complexes.

Le problème principal qui se pose avec ces processeurs CISC est que le nombre de cycles nécessaire pour exécuter une instruction est variable dans une large proportion, et rend le calcul de la charge logicielle du processeur très difficile, donc le but recherché était de définir un jeu d'instructions tel que, chacune d'entre elles s'exécute en un seul cycle d'horloge, ce qui conduit Aux processeurs RISC (*Reduced Instruction Set Computer)* ou calculateurs à jeu d'instructions réduit au début des années 1980.

Un DSP (*digital signal processor*) est un processeur dédié au traitement des signaux numériques. C'est un microprocesseur RISC dans lequel l'architecture et le jeu d'instructions sont optimisés afin d'obtenir un temps d'exécution le plus petit possible pour l'adapter aux algorithmes de traitement de signal [1].

1.3 Domaines d'applications des DSP

Les domaines d'applications du traitement numérique du signal sont nombreux et variés (traitements du son, de l'image, synthèse et reconnaissance vocale, analyse, compression de données, télécommunications, automatisme, etc). Chacun de ces domaines nécessite un système de traitement numérique, dont le coeur est un ou parfois plusieurs DSP ayant une puissance de traitement adaptée, pour un coût économique approprié.

Les domaines d'applications des DSP les plus courantes [2] sont :

- **Télécommunications :** modems, multiplexeurs, récepteurs de numérotation DTMF, télécopieurs, codeurs de parole GSM, modems radio, etc;
- **Interfaces vocales :** codeurs vocaux pour répondeurs, reconnaissance automatique de parole, synthèse vocale, etc;
- **Militaire :** guidage des missiles, navigation, communications cryptées, traitement radar;
- **Multimédias et du grand public :** compression des signaux audio (CD), compression des images, cartes multimédias pour PC, synthèse musicale, jeux, etc;
- **Médical :** compression d'image médicale (IRM, échographie...), traitements des signaux biophysiques (ECG, EEG...), implants cochléaires, équipements de monitoring, etc;
- **Electronique automobile :** Equipements de contrôle moteur, aide à la navigation, commande vocale, détection de cliquetis pour l'optimisation de l'avance à l'allumage, etc ;

- **Automatisation et contrôle des processus :** la surveillance et commande de la machines, contrôle des moteurs, les robots, les servomécanismes, etc;
- **Instrumentation :** Analyseurs de spectre, générateurs de fonction, et l'interprétation de signaux sismiques, etc ;

1.4 Classe des signaux à traiter

Selon le domaine d'application et le type de signal à traiter, la fréquence d'échantillonnage nécessaire est plus ou moins grande. Le temps disponible entre deux échantillons correspond à un nombre n plus ou moins important de temps de cycles T_{cycle} du DSP : $n = T_s / T_{cycle}$. La fréquence d'échantillonnage f_s est choisie d'une façons à respecter le théorème de Shannon, [3] : $f_s \geq 2f_{max}$

De plus, Pour n donné f_s doit être inférieure à $1/(nT_{cycle})$, le Tableau 1.1 donne quelque exemple des signaux physique.

Signal	Fréquence d'échantillonnage	Période d'échantillonnage	Nombre de temps de cycle T_{cycle} entre deux échantillons avec T_{cycle} =50ns
Signaux vibratoires	100 kHz	10 µs	200
Parole	8 kHz	125 µs	2 500
Audio	48 kHz	20.8 µs	417
vidéo	10 MHz	100 ns	2

Tableau 1.1 : les classes des signaux

A l'examen du tableau1.1, on comprend que les DSP sont couramment utilisés pour traiter en temps réel les signaux de la bande audio (0 à 20 kHz de largeur de bande), mais que le traitement des signaux vidéo en temps réel reste la plupart du temps du domaine des circuits intégrés dédiés ou d'architecture multi DSP [2].

1.5 Spécificités des 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 oeuvre en lui associant des mémoires (RAM, ROM) et des périphériques Figure 1.1. 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, des mémoires, des timers, des ports séries synchrones rapides, des contrôleurs DMA, et des divers ports d'E/S.

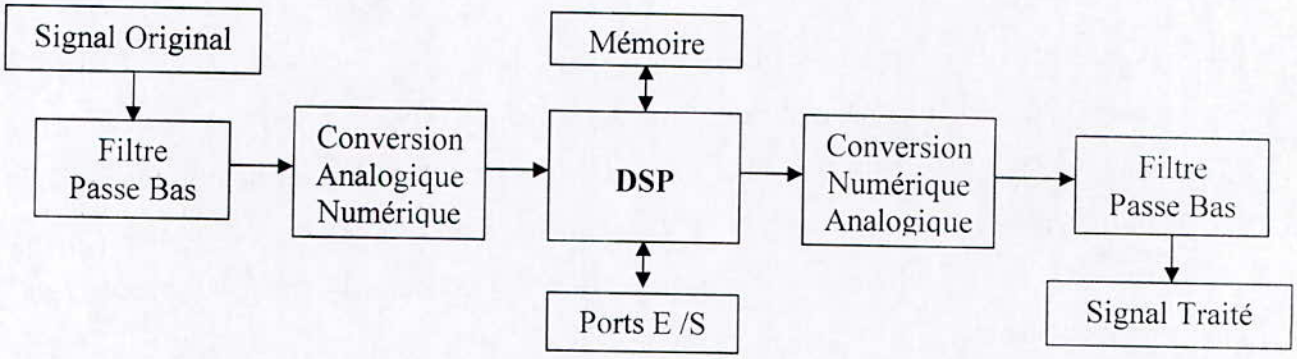


Figure 1.1 : chaîne complète typique d'un système de traitement numérique du signal

1.5.1 Principale distinction entre un microprocesseur et un DSP

Un DSP est différent à un microprocesseur par les deux spécifications suivantes :

1.5.1.1 L'opération MAC

Apres avoir été numérisé, le signal se présente sous la forme d'une suite de valeurs numériques discrètes. Cette suite de valeurs (ou 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 $A = (B \times C) + D$, que l'on appelle l'opération *MAC (Multiply and ACcumulate)*. Un microprocesseur classique nécessite plusieurs cycles d'horloge pour effectuer un tel calcul, par exemple, le 68000 de *Motorola* a besoin de :

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

Soit 80 cycles pour calculer seulement A. Si ce temps est admissible dans des applications informatiques courantes, il n'est pas acceptable pour faire des traitements rapides. Les DSP sont donc conçus pour optimiser ce temps de calcul. A cet effet, ils disposent de la fonction optimisée *MAC* permettant de calculer A beaucoup plus rapidement.

La plupart des DSP ont un jeu d'instructions spécialisés permettant de lire en mémoire une donnée, effectuer une multiplication puis une addition, et enfin écrire le résultat en mémoire, le

tout en un seul cycle d'horloge. Ce type d'opération *MAC* est effectué en un seul cycle, mais il n'est pas satisfaisant si le cycle d'horloge est trop lent. Le principal objectif d'évolution des DSP, a toujours été d'améliorer le temps de calcul d'une opération *MAC*. La figure 2 présente le progrès réalisé dans ce domaine depuis plus de trois décennies [1].

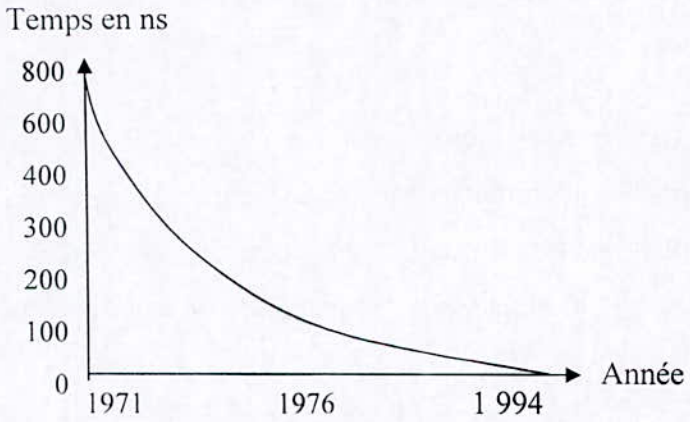


Figure 1.2 : Evolution du temps d'exécution d'une opération *MAC*, selon Texas Instruments

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 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 débordement 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.

1.5.1.2 L'accès à la mémoire

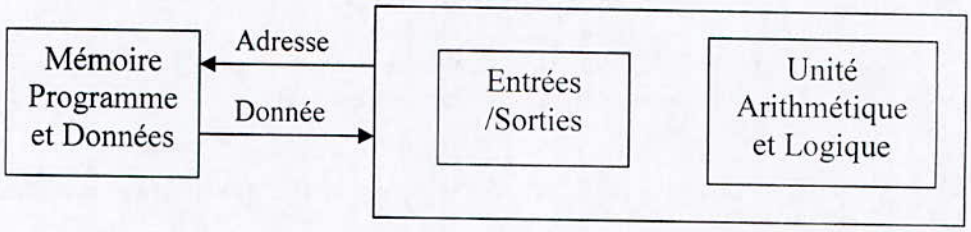
Une autre caractéristique intéressante des DSP est leurs capacités à réaliser plusieurs accès mémoire en un seul cycle. Ceci permet à un DSP de chercher en mémoire une instruction et ces données simultanément, pour réaliser une opération *MAC*. Le gain en temps est évident. Toutefois, sur certains DSP de base, ce type d'opérations simultanées est généralement limité à des instructions spéciales [2]. Ces instructions utilisent un mode d'adressage restreint, c'est à dire ne portant que sur la mémoire vive intégrée au DSP.

Les modes d'adressages des données sont un point particulier des DSP. Un DSP peut possède plusieurs unités logiques de génération d'adresse, travaillant en parallèle avec la logique du coeur du DSP. Une unité logique de génération d'adresse est paramétrée une seule fois

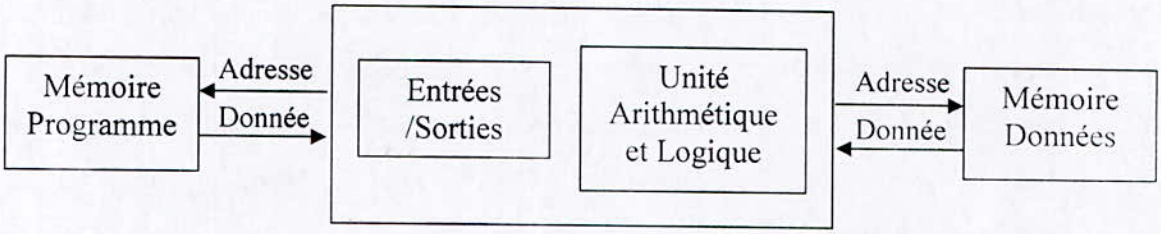
via des registres appropriés, elle génère alors toute seule, en parallèle avec l'exécution d'une opération arithmétique, les adresses nécessaires à l'accès des données. Ceci permet non seulement de réaliser les accès mémoires simultanés en un seul cycle, mais également d'incrémenter automatiquement les adresses générées. Ce mode d'adressage particulier, généralement appelé adressage indirect par registre avec post (ou pré) incrémentation, est très utilisé pour effectuer des calculs répétitifs sur une série de données rangées séquentiellement en mémoire.

1.5.2 Architecture des processeurs

L'architecture d'un microprocesseur, et donc d'un DSP, est un élément important qui conditionne directement les performances d'un processeur. Il existe deux types de structures fondamentales Figure 1.3, dites de « *Von Neuman* », et de « *Harvard* » [4].



a) Architecture de type *Von Neuman*



b) Architecture de type *Harvard*

Figure 1. 3 : représentation schématique des structures
a) architecture de *Von Neuman*,
b) architecture *de Harvard.*

1.5.2.1 Structure de *Von Neuman*

Un microprocesseur basé sur une structure de *Von Neuman* stocke les programmes et les données dans la même zone mémoire. Une instruction contient le code opératoire et l'adresse de l'opérande. Ce type des microprocesseurs incorpore principalement deux unités logiques de base

- L'Unité Arithmétique et Logique (ou *ALU*), chargée de réaliser les opérations centrales (de type multiplications, additions, soustractions, rotation, etc.),
- L'unité de charge des Entrées/ Sorties, qui commande le flux des données entre le coeur du microprocesseur et les mémoires ou les ports d'E/S.

1.5.2.2 Structure de *Harvard*

Cette structure se distingue de l'architecture de *Von Neuman* par le fait que les mémoires programmes et données sont séparées. L'accès à chacune des deux mémoires se fait via deux chemins distincts. Cette organisation permet de transférer une instruction et des données simultanément, ce qui améliore les performances, tel que l'architecture de *Harvard* est considérée comme deux fois plus rapide, mais ce gain de vitesse ce fait au prix d'une sérieuse complication de l'électronique puisque toute l'architecture des bus est doublée.

Certains processeurs séparent la mémoire programme de la mémoire de données avec un seul bus de données /adresses, il ne bénéficient pas du principale avantage (gain de vitesse) de l'architecture *Harvard*. On dit que ces processeurs utilisent une architecture *Harvard* modifié.

1.5.2.3 Utilisation de ces structures dans les DSP

L'architecture généralement utilisée par les microprocesseurs est la structure *Von Neuman* (exemples : la famille *Motorola* 68XXX, la famille *Intel* 80X86). L'architecture *Harvard* est plutôt utilisée dans des microprocesseurs spécialisés pour des applications temps réels, comme les DSP [1], [4].

Pour réduire le coût de la structure *Harvard*, certains DSP utilisent la structure de *Harvard* modifiée [4]. A l'extérieur, alors le DSP ne propose qu'un bus de données et un bus d'adresse, comme la structure *Von Neuman*. 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 un multiplexage temporel.

1.6 les formats des données utilisés dans les DSP

Deux approches sont utilisées pour la représentation binaire des nombres réels en précision finie [2], [4] :

- la représentation en virgule fixe ou format fixe ;
- la représentation en virgule flottante ou format flottante.

Les DSP sont conçus pour l'une ou l'autre de ces représentations. Toutefois, un DSP travail en virgule fixe pourra aussi effectuer des calculs en virgule flottante, mais de manière peu efficace et réciproquement.

1.6.1 Représentation binaire des nombres fractionnaires en format virgule fixe

On appelle représentation en virgule fixe des nombres fractionnaires, ou plus généralement des nombres réels avec une précision finie, une représentation comprenant une partie entière suivie d'une partie fractionnaire correspondante à des bits après la virgule.

On utilise l'expression « format Q_k » pour indiquer une représentation comportant k bit après la virgule.

Un nombre réel étant rarement en précision finie, sa représentation sur un nombre fini de bits introduit une erreur, en virgule fixe sur N bits avec un format Q_k, cette erreur est inférieure à 2^{-k}.

1.6.1.1 Les valeurs extrêmes en virgule fixe sur N bits avec un format Q_k

Les valeurs extrêmes représentables en format fixe Q_k sont :

$max = 2^{N-1-k} - 2^{-k}$

$min = -2^{-N--k}$

Représentation binaire virgule fixe, format Q_5	Valeur décimale
011 10000	3.5
001 10100	1.625
110 10001	-1.46875
100 00000	-4
011 11111	3.96875

Tableau 1.2 : représentation binaire et valeur décimale.

Valeur réelle	Représentation binaire virgule fixe, format Q_5	Equivalence décimale	Erreur commise
1/3	000 01011	0.34375	0.01041666
$\sqrt{3}$	001 01101	1.40625	0.007963562
π	011 00101	3.15625	0.014657346

Le tableau1.3 quelques représentations pour $N = 8$ bits en format fixe Q_5

Pour des réels en dehors de cette plage, on dit qu'il y a :

- *overflow* si le nombre est trop grand en valeur absolue. Il y a débordement ;
- *underflow* si le nombre est trop petit est alors représenté par zéro.

1.6.2 Représentation binaire des nombres fractionnaires en format virgule flottante

Dans la représentation binaire en virgule flottante en précision finie sur N bits, les nombres sont représentés par une mantisse M et un exposant E ; M et E représentent la valeur x : $x = M\,2^{E}$.

pour une représentation de N bits, la mantisse M est exprimée sur m bits, et l'exposant E sur e bits.avec $N = m+e$

1.6.2.1 Plage des nombres représentables en format virgule flottante

Sur N bits, avec m bits pour la mantisse normalisée entre 0.5 et 1, et e bits pour l'exposant. On peut représenter des nombres dont la valeur absolue et compris dans l'intervalle :

$$\left[\frac{1}{2}2^{-(2^{e-1})},\left(1-2^{1-m}\right)2^{2^{e-1}-1}\right]$$

$$Overflow \text{ si } |x| \text{ f } \left(1-2^{1-m}\right)2^{\left(2^{e-1}-1\right)}.$$

$$Underflow \text{ si } |x| \text{ f } \frac{1}{2}2^{-(2^{e-1}-1)}.$$

Dans ce cas la précision relative et à peu prés constante, et l'erreur de représentation est inférieure à $\frac{1}{2}2^{-2^{e-1}}2^{2-m}$ dans tout les cas.

Représentation binaire Mantisse M, exposant E	Valeur décimale $M\,2^E$
01110 010	1.75
01100 100	0.046875
10010 011	-7
01000 100	0.03125
01111 011	7.5

Tableau 1.4 : quelque représentation sur 8 bits avec m = 5 et e = 3 en virgule flottante.

Valeur réelle	M	E	Equivalence décimale	Erreur commise
1/3	01011 (0.6875)	111 (-1)	$0.6875\times2^{-1}=1.34375$	0.0134166..4%
$\sqrt{3}$	01011 (0.6875)	001 (1)	$0.6875\times2^{1}=1.375$	-0.0392135...2.77%
π	01101 (0.8125)	010 (2)	$0.8125\times2^{2}=3.25$	0.10840734...3.4%
$\sqrt{50}$	01110 (0.875)	011 (3)	$0.875\times2^{3}=7$	0.07106781...1%

Tableau 1.5 : quelques exemples de nombre réels sur 8 bits, m=5 et e=3, avec l'équivalence décimale et l'erreur commise par cette représentation. La mantisse M est normalisée : $\frac{1}{2} \le |M| \le 1$.

1.6.3 Comparaison entre les deux représentations

Pour un nombre de bits N donnée, la représentation des nombres en virgule flottante réalise un compromis entre la dynamique E et la précision M

En virgule fixe, les opérateurs de traitement sont simples mais il faut surveiller le cadrage des données pour éviter les débordements, tout en conservant un maximum de précision.

En virgule flottante, les opérateurs sont les plus complexes mais en dispose d'une grande dynamique pour une précision minimale donnée et le cadrage des données est moins critique.

1.7 Les DSP de la famille TMS320

La famille TMS320 se compose de processeurs à virgule fixe, à virgule flottante et de multiprocesseurs [1], [2], [4]. L'architecture des DSP TMS320 est conçue spécifiquement pour le traitement des signaux en temps réel. Les caractéristiques suivantes font à cette famille le choix idéal pour une large gamme de traitement des applications :

- ensemble d'instructions très flexible ;
- flexibilité opérationnelle inhérente ;
- exécution à grande vitesse
- architecture parallèle innovatrice
- rentabilité

Les DSP TMS320 offrent Des approches plus adaptables aux problèmes de traitement de signal classiques tel que le filtrage que des dispositifs de microprocesseur standard / micro-ordinateur. Ils sont très utiles pour les applications complexes qui exigent souvent l'exécution simultanée de plusieurs opérations.

Les familles les plus récentes des DSP de *Texas instrument* Tableau 1.6 sont :

- TMS320C54x, DSP format fixe ;
- TMS320C20x, DSP format fixe ;
- TMS320C24x, DSP format fixe ;
- TMS32OC62x, DSP format fixe a architecture VLIW (Very long instruction word);
- TMS32OC67x, DSP format flottant a architecture VLIW ;

Ces familles sont regroupées en 3 classes appelées plates-formes.

- TMS320C6000, formée des familles C62x et C67x ;
- TMS320C5000, formée de la famille C54x et des C54xx ;

- TMS320C2000, formée des familles C20x et C24x ;

Les Principales caractéristiques des 3 plateformes de TMS320 sont données dans le tableau 1.6.

Application		Type de DSP	caractéristiques
TMS320 DSP hautes performances			
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, en consommation : voix sur IP, alphapages.	16 bits virgule fixe.	0,54 mW/MIPS 30-200 MIPS
TMS320C2000 DSP optimisés pour les applications de contrôle			
C20x	Applications de grands 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 1.6 : Principales caractéristiques des 3 familles de TMS320

1.8 Les DSP TMS320C54x

Le DSP TMS320C54x est un processeur de traitement numérique du signal à virgule fixe, satisfait les besoins spécifiques aux applications en temps réels [2], [5].

L'unité centrale (CPU) de C54x, avec son architecture de *Harvard* modifiée, permet une réduction minimale en terme de consommation et un degré élevé de parallélisme. De plus, la souplesse des modes d'adressage et le jeu d'instruction élevé améliorent les performances du système global.

Les DSP C54x ont un degré élevé de flexibilité et de vitesse opérationnelles. Ils Combinent une architecture avancée de *Harvard* modifiée (avec un bus de mémoire programme, trois bus de mémoire de données, et quatre bus d'adresses), une CPU, une mémoire, des périphériques, et un jeu d'instructions fortement spécialisé.

1.8.1 Les avantages des DSP C54x :

- Architecture de *Harvard* établie autour d'un bus de programme, trois bus de données, et quatre bus d'adresses pour accroître les performances et la polyvalence ;
- Conception avancée d'unité centrale (CPU), un degré élevé de parallélisme ;
- Un jeu d'instructions fortement spécialisé pour des algorithmes plus rapides et pour optimisée l'opération du langage de haut niveau ;
- Conception d'architecture modulaire pour le développement rapide des dispositifs ;
- Technologie avancée pour une faible consommation, et utilisation des nouvelles techniques de conception statiques.

1.8.2 Les composants de la famille TMS320C54x

La famille C54x est aujourd'hui l'une des plus utilisées, en particulier dans le domaine des télécommunications avec les mobiles. Elle possède des bonnes performances en consommation, en vitesse, et en prix. Sont architecture se caractérise par [5] :

➢ **CPU :**

- Quatre bus internes et deux générateurs d'adresse ;
- Une ALU 40 bits pouvant travailler en double précision ou en mode dual ;
- Deux accumulateurs 40 bits ;
- Une unité de multiplication et d'addition, avec un multiplieur 17 × 17 bits ;
- Unité de comparaison, sélection, sauvegarde (CSSU), appelée aussi accélérateur de *Viterbi* ;

- Un calculateur d'exposant en un seul cycle pour faciliter les calculs en format flottant nécessaire dans certaines applications ;
- Un registre à décalage 40 bits ;
- Huit registres auxiliaires et une pile logicielle ;
- Un jeu d'instructions contenant des instructions mono-cycle spécialisées dans le traitement de signal, comme le filtrage adaptatif ou les filtres symétriques.

➢ **Mémoire :**

- 192 Kmots de 16 bits d'espace mémoire (64 Kmots pour la mémoire programme, 64 Kmots pour la mémoire données, et 64 Kmots pour les E/S), avec prolongement de la mémoire programme dans le C548, le C549, le C5402, le C5410, et le C5420.

➢ **jeu d'instructions :**

- opérations de répétition d'une Simple instruction et de répétition de bloc;
- Instructions de déplacement de bloc mémoire pour une meilleure gestion de programme et les données ;
- Instructions avec un long opérande de 32 bits ;
- Les instructions avec 2 ou les 3-opérandes exécutés simultanément ;
- Instructions arithmétiques avec enregistrement et chargement parallèle ;
- Stockage des instructions conditionnelles ;
- Rapidité du retour de l'interruption ;

➢ **périphériques**

- Ports séries synchrones ;
- Ports séries synchrones multiplexés en temps TDM pour connecter des DSP en chaîne;
- Ports séries synchrones avec buffer (BSP : *Buffred Serial Port*) dont la taille peut aller jusqu'à 2 Kmots ;
- Ports séries synchrones multicanaux McBSP (jusqu'à 128 canaux), permettant la connexion directe *full-dupllex* avec des interfaces téléphoniques T1/E1 ou H100 ;
- Un contrôleur de DMA (*Direct Memory Access*) 6 canaux ;
- Une interface parallèle avec processeur hôte (HPI pour *Host Port Interface*) ;

- Un timer.

Ces composants peuvent être alimentés selon leur référence UC, VC, LC, ou C en 1.8V, 2.5 V, 3.3 V, OU 5 V.

Le tableau1.7 liste les composants de la famille C54x [2], avec leurs caractéristiques respectives. Dans le tableau1.7 :

- Le symbole SER : signifie port série synchrone,
- Le signe ! signifie port série bufférisé,
- Le signe * indique port multiplexé en temps TDM,
- Le signe + veut dire port série multicanaux McBSP,
- Le signe & correspond à un contrôleur de DMA,
- DAT/ORO signifie espace mémoire données/programme adressable,
- COM indique les interfaces de communication et HPI.

Nom	RAM	ROM	DAT/ PRO	SER	Alim	COM	Timers	MHz	T_{cycle}	MIPS	Boîtier
C541# -40	5 Ko	28 Ko	64Ko/ 64Ko	2	5 V	-	1	PLL	25	40	100TQFP
C542# -40	10 Ko	2 Ko	64Ko/ 64Ko	2 ! *	5 V	HPI	1	PLL	25	40	128, 144TQFP
LC541B -66	5 Ko	28 Ko	64Ko/ 64Ko	2	3.3 V	-	1	SW/PLL	15	66	100TQFP
LC542# -40	10 Ko	2 Ko	64Ko/ 64Ko	2 ! *	3.3 V	HPI	1	PLL	25	40	128, 144TQFP
LC542# -50	10 Ko	2 Ko	64Ko/ 64Ko	2 ! *	3.3 V	HPI	1	PLL	20	50	128, 144TQFP
LC543# -40	10 Ko	2 Ko	64Ko/ 64Ko	2 ! *	3.3 V	-	1	PLL	25	40	100TQFP
LC543# -50	10 Ko	2 Ko	64Ko/ 64Ko	2 ! *	3.3 V	-	1	PLL	20	50	100TQFP
LC545A -50	6 Ko	48 Ko	64Ko/ 64Ko	2 !	3.3 V	HPI	1	SW/PLL	20	50	128, 144TQFP
LC545A -66	6 Ko	48 Ko	64Ko/ 64Ko	2 !	3.3 V	HPI	1	SW/PLL	15	66	128, 144TQFP
LC546A -50	6 Ko	48 Ko	64Ko/ 64Ko	2 !	3.3 V	-	1	SW/PLL	20	50	100TQFP
LC546A -66	6 Ko	48 Ko	64Ko/ 64Ko	2 !	3.3 V	-	1	SW/PLL	15	66	100TQFP
LC548 -66	32 Ko	2 Ko	64Ko/ 8Mo	3 ! *	3.3 V	HPI	1	SW/PLL	15	66	144TQFP
LC548 -80	32 Ko	2 Ko	64Ko/ 8Mo	3 ! *	3.3 V	HPI	1	SW/PLL	12.5	80	144TQFP
LC549 -66	32 Ko	16 Ko	64Ko/ 8Mo	3 ! *	3.3 V	HPI	1	SW/PLL	15	66	144BGA, TQFP

LC549 -80	32 Ko	16 Ko	64Ko/ 8Mo	3 ! *	3.3 V	HPI	1	SW/PLL	12.5	80	144BGA, TQFP
VC549- 100	32 Ko	16 Ko	64Ko/ 8Mo	3 ! *	2.5 V	HPI	1	SW/PLL	10	100	144BGA, TQFP
VC5402 -100	16 Ko	4 Ko	64Ko/ 8Mo	2 + &	1.8/ 3.3 V	HPI	2	SW/PLL	10	100	144BGA, TQFP
VC5410 -100	64 Ko	16 Ko	64Ko/ 8Mo	3 + &	2.5 V	HPI	1	SW/PLL	10	100	144BGA, TQFP
VC5420 -100	200	-	64Ko/ 256 Ko	6 + &	1.8 V	HPI	1	SW/PLL	10	200	176BGA, 144TQFP
UC 5402	16	4	64Ko/ 8Mo	2 + &	1.2/ 1.8 V	HPI	1	SW/PLL	10	100	
UVC 5402	16	4	64Ko/ 8Mo	2 + &	1.2/ 1.8 V	HPI	1	SW/PLL	10	100	

Tableau 1.7 : les composants de la famille C54x

D'autres composants, comme le 5409et le 5416 (tés faible consommation à160 MHz), sont annoncés.

1.9 Conclusion

Nous avons montré l'intérêt de l'utilisation des DSP dans les différents domaines de traitement de signal, en exposant ses avantages qui font une grande utilité dans plusieurs applications .

D'autre part, autant les DSP sont d'une efficacité redoutable lorsqu'on les utilise en bon escient autant leurs performances s'effondrent dés lors qu'on les sort de ce cadre. L'un des points critiques concerne le volume susceptible d'être traité. La capacité d'adressage des DSP est par nature limitée de quelques dizaines à quelques centaines de Kmots. Toute tentative pour leur faire gérer le plus grand espace mémoire se traduit immanquablement par une diminution sensible du temps d'exécution, qui vient alors similaires ou inférieure à celle des processeurs CISC.

Une présentation des outils de génération de code et les outils de développement d'applications sur les DSP de Texas Instruments est nécessaire. Ceci constitue l'objet du deuxième chapitre.

Outils de développement logiciel et matériel

2.1 Introduction

L'utilisation des DSP est relativement sophistiquée, il est important de disposer d'outils de développement performants et conviviaux qui permettant de minimiser le temps de mise sur le marché d'une nouvelle application. *Texas instruments* a développé une série d'outils logiciels est matériels ainsi qu'un système d'aide et d'information à l'utilisateur, exploitant en particulier un site Web très riche. Cet ensemble et enrichi par de nombreux produits proposés par des tierce-parties. Certains de ces outils offrent notamment un environnement de développement intégré matériel et logiciel : IDE (*integrated Development Environment).*

Dans le cadre de ce projet, nous intéressons au *Code Composer Studio CCS2* comme un outil de développement logiciel, et le Kit de développement DSKC5402 comme un outil de développement matériel.

2.2 Les outils de développement logiciel

Les outils de développement logiciel permettent de développer du code, de le simuler et de le debugger. Ces outils fonctionnent généralement sur PC Windows ou NT, et parfois sur stations HP ou SPARC. Certaines tierce-parties offres de plus des systèmes d'exploitation temps-réel compatibles avec la famille C54x, tels que *Virtuoso351* de *Eonic Systems* ou *CMX-RTX* de *CMX Company*.

Les principaux outils de développement logiciel de la famille C54x sont [2]:

Outil de développement logiciel	Assembleur, éditeur de lien –*Assembly language tools*
	Compilateur C–*Code generation tools–Optimising C* compiler
Outils de simulation et de déboggage	Simulateur et interface de déboggage DOS ou Windows
	Environnement de développement intégré : Code Composer

Tableau 2.1 : Les outils de développement logiciel

2.2.1 Programmation en C et en assembleur

On a le choix entre écrire une application en C [6], [7], ou en Assembleur [8]. L'écriture directe en Assembleur permet d'obtenir de meilleure performance en temps d'exécution et en occupation mémoire, mais la programmation en langage C est plus simple et plus rapide qu'écrire le même algorithme en assembleur. De plus, l'écriture en C garantie la portabilité du programme sur différents processeurs.

Le programme assembleur généré par le compilateur C à partir du fichier source en langage C est beaucoup moins efficace, à la fois en termes de vitesse d'exécution et en terme d'occupation mémoire que le même programme écrit directement en assembleur. Toutefois, l'optimiseur C [7], permet de diminuer cette différence d'efficacité.

D'autres caractéristiques du compilateur C permettant d'augmenter l'efficacité du programme généré [6], telles que la possibilité d'insérer des lignes assembleur directement dans le code C, d'appeler des fonctions assembleur à partir du code C, et l'utilisation des opérateurs *intrinsics*.

Le compilateur C pour les DSP C54x accepte des programmes sources en assembleur *CANSI.* la figure2.1 décrit la chaîne élémentaire de développement d'un programme en C.

Le compilateur C (cc500) génère un fichier source en assembleur d'extension (.asm) à partir du fichier source (.c). L'assembleur convertit ce fichier en un fichier objet en langage machine au format COFF (*Common Object File Format :* destiné à favorisé la programmation modulaire). L'éditeur de liens combine plusieurs fichiers relogeables en format COFF, pour générer un fichier COFF exécutable (.out) par un DSP C54x.

La programmation en C et intéressante pour réduire le temps de développement d'une application, et on l'utilise pour les parties du programme dont le temps d'exécution n'est pas

critique. On utilise la programmation en assembleur pour les bloc de programme effectuent des calculs arithmétiques intensifs dont le temps d'exécution est critique.

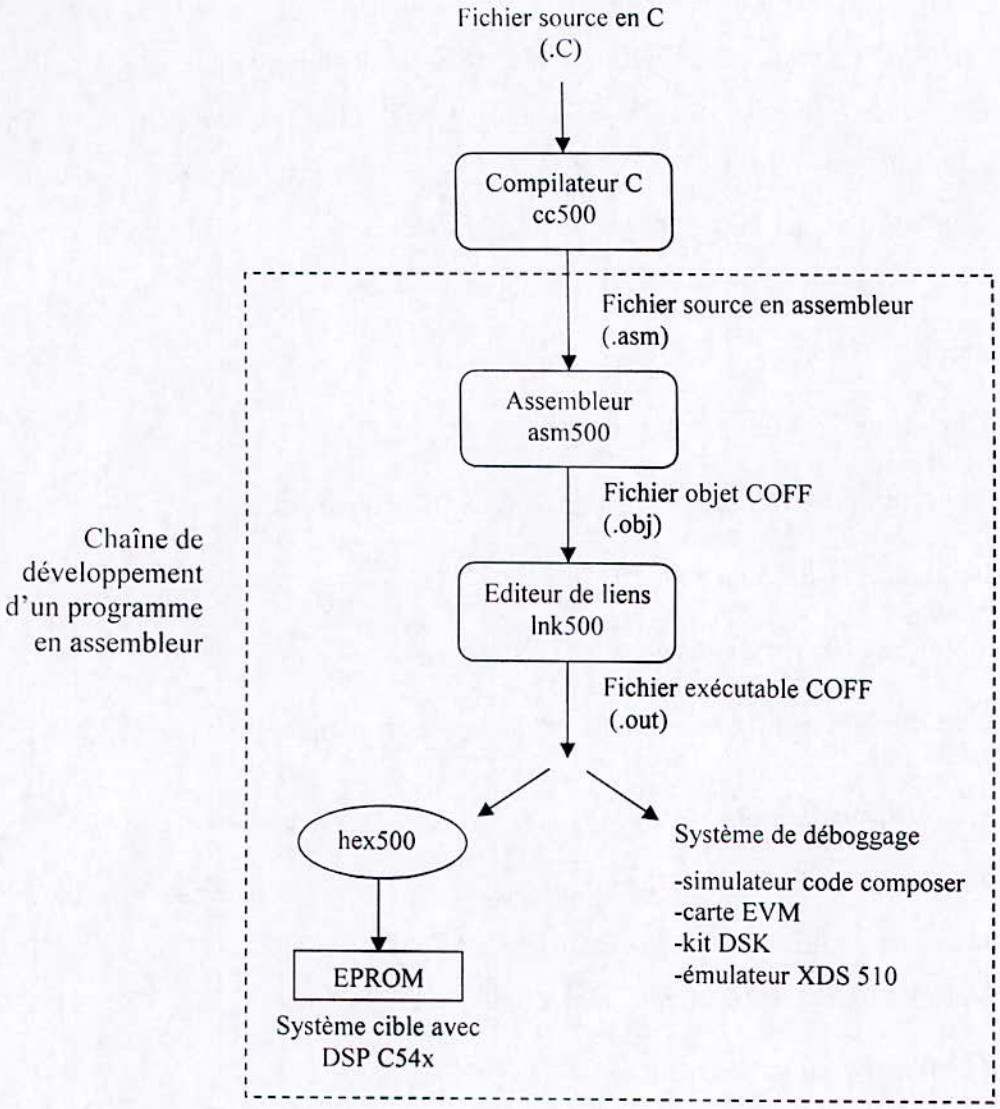


Figure 2.1 chaîne élémentaire de développement d'un programme en C [2].

2.2.2 Outils de simulation et de déboggage

Le simulateur permet de simuler l'exécution d'un programme écrit en C ou en assembleur pour un DSP de la famille TMS320C54x. le simulateur tourne sur un calculateur hôte. Il utilise une interface de déboggage standard permet la vérification d'un programme sans les aspects temps réel.

2.3 Environnement de développement intégré Code Composer Studio

2.3.1 Code Composer Studio setup

Avant d'utilisé le *Code Composer Studio,* il faut configure le système par le *Code Composer Studio Setup* qui définit la cible ou le simulateur de cible que l'on utilise pour le teste sur le code *Composer Studio* [9] et [10]. Cette définition s'appelle la configuration du système, et elle se compose d'un driver qui manipule la communication avec la cible, plus autres informations qui décrivent les caractéristiques de la cible utilisée. La configuration du système est nécessaire avant d'établir une application.

L'utilisation du *Setup* est pour:

- Choisir une configuration standard fournie par *TI* ;
- Importer une configuration fournie par les tierce-parties ;
- Adapter une configuration ;
- Créer une nouvelle configuration ;
- Exporter une configuration ainsi que d'autres peuvent l'employer.

2.3.2 Code Composer Studio

Code Composer Studio est un environnement de développement intégré (IDE), commun aux déférents DSP de *Texas Instruments*, qui peut s'utilisé seul ou en lien avec une cible matérielle fonctionne en temps réel tel que le DSKC5402 [2], [9], [10] et [11].

Les principales caractéristiques du *Code Composer* sont:

- Le simulateur *Code Composer* permet de construire un projet, d'éditer les fichier, de les compiler ou de les assembler, de faire l'édition de liens, de tester et de debogger une application mono ou multiprocesseur en simulation ou en temps réel sur une cible matérielle. Les cibles matérielles supportées sont les cartes d'évaluation EVM, les Kits de développements DSK, ainsi que les émulateurs de Texas Instruments et ses tierce-parties.
- Le simulateur *Code Composer* permet le teste de l'application sans contrainte de temps réel.

- Code composer est un debogger symbolique permettent de référencer les variables et les constantes par leurs noms, plutôt que par leur adresse physiques.
- *Code Composer* offre par ailleurs des outils graphiques permettent de tracer et d'analyser des signaux contenus en mémoire. Ces tracés peuvent se faire en temps ou en fréquence. D'autres types de tracés sont disponibles, comme les diagramme de l'œil ou les constellations.
- *Code Composer* dispose d'un langage de commande appeler GEL (*GoDSP Extension language*), qui permet d'automatisé de manière personnalisée le teste d'une application, en ajoutant par exemple des fonctions aux items des menus de base.

Code composer dispose de toutes les possibilités du simulateur et de l'interface de déboggage DOS ou Windows.

2.3.2.1 L'interface utilisateur graphique :

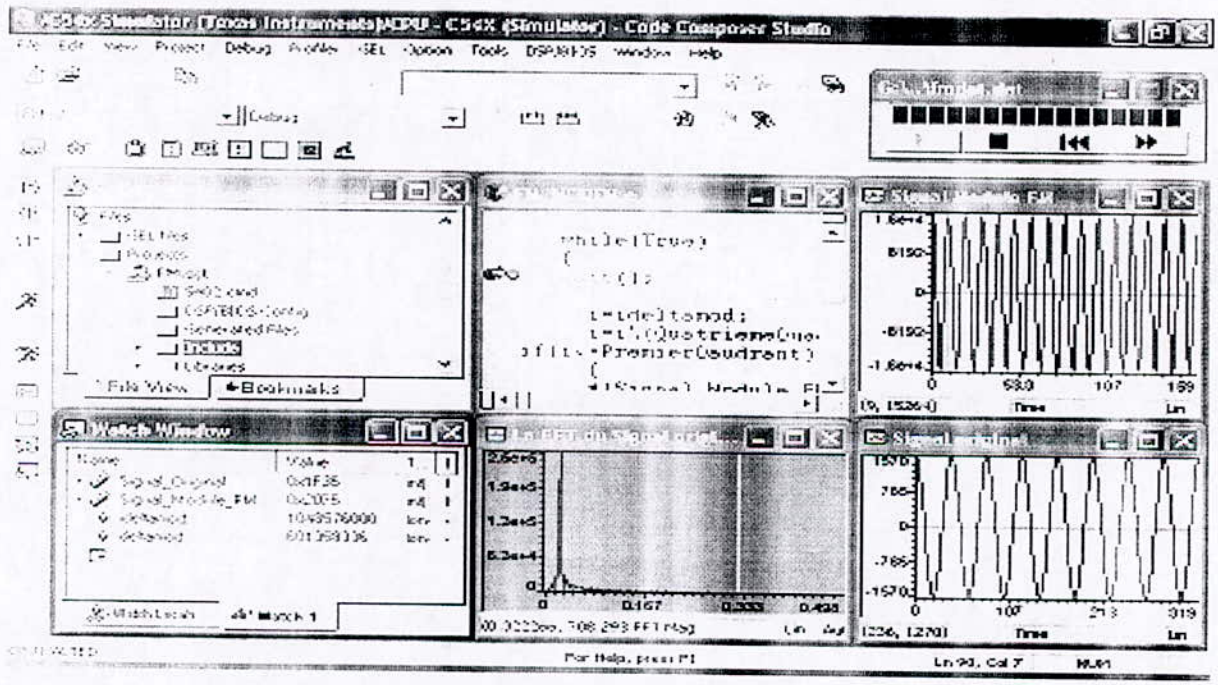


Figure 2.2: l'interface graphique du Code Composer

2.3.2.2 Barre de menu et icône

En haut de la fenêtre principale se trouve une barre de menus déroulants : *File*, Edit, *View*, etc. A gauche de la fenêtre principale, on trouve des icônes qui sont des raccourcis vers des commandes qui sont aussi accessibles par la barre de menus. Les déférentes commandes de code composer peuvent être appelées soit en les sélectionnent dans des menus déroulants

accessible de la bar de menus en haut de la fenêtre principale, soit en cliquant sur l'icône associer s'il existe.

2.3.2.3 Les fenêtres

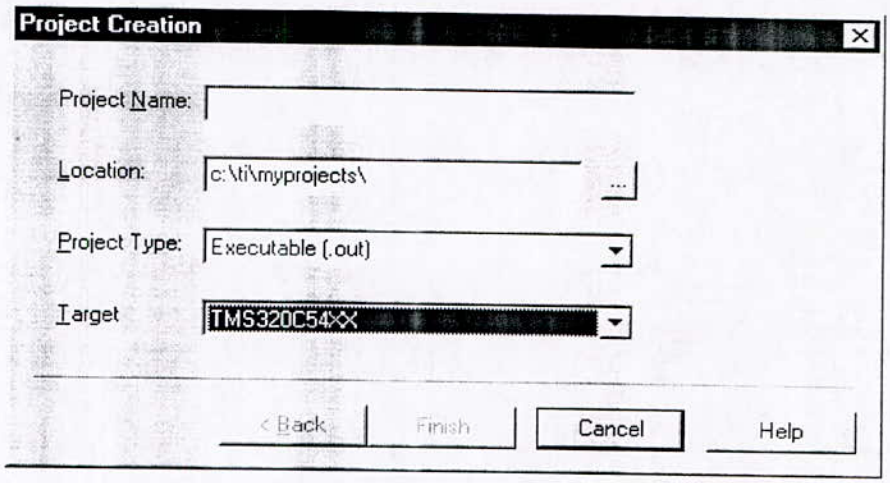
La fenêtre principale peut contenir de nombreuse fenêtres spécialisées comme :

- Une fenêtre *Project View* montrant la structure du projet en cours.
- La fenêtre *Wacth* servent à visualiser des variables.
- La fenêtre *Dis-Assembly* contenant le code assembleur de programme.

Il existe d'autre fenêtre comme les fenêtres mémoire qui visualisent les mémoire, la fenêtre appeler *Call Stack* (pile d'appels) indiquant les déférentes appels de fonctions qui sont exécutés, la fenêtre *C54x Registers* affichant le contenue des registres du DSP, contenus que l'on peut modifier en cliquant dessus dans la fenêtre et les fenêtres graphiques.

2.3.2.4 Création d'un fichier projet

Pour crée un nouveau projet en appelant dans la barre de menus, la command *Projet–New*. En réponse une fenêtre de dialogue s'ouvre.



Dans le champ *Project Name* on indique le nom du nouveau projet. Le champ *Location* pour indiquer le répertoire où l'on veut stocker se projet. Dans champ *Project Type* en choisie entre *Exécutable* (.out) ou *Library* (lib). *Exécutable* indique que le projet génère un fichier exécutable (.out). *Library* pour crée une librairie d'objet. Le fichier du projet a l'extension *pjt*.

Pour ajouter des fichier au projet on clique sur la commande *Project-Add files to project* dans la barre de menus. le projet contient des fichiers source en c (.c) ou en

assembleur (.asm) ou les deux, un fichier de commande de l'éditeur de lien (.cmd), des fichiers header (.h), et des fichiers de la bibliothèque.

2.3.2.5 Compilation, Assemblage et édition de lien

On compile le projet à l'aide de la commande *Project-Built*. Il est possible de changer les options de compilation, d'assemblage ou d'édition de lien du projet par la commande *Project-Options*.

2.3.2.6 Deboggage intégré

1- Chargement d'un programme

Pour tester le programme on va d'abord le charger par la commande *File-Load Program*. En réponse à cette commande une fenêtre *Dis-Assembly* s'ouvre.

Pour tester le programme simultanément à l'aide du code C et de l'assembleur. On clique sur la commande *View-Mixed Source/Asm*.

2- Points d'arrêt, break points

Les points d'arrêt servent à arrêter l'exécution d'un programme à la lecture d'instructions particulières ou lorsqu'une condition et remplie.

Pour placer un point d'arrêt sur une instruction, en positionnant le curseur devant l'instruction désiré, et en clique sur l'icône *toggle breakpoint* représenter par une main ouverte  .

Lors de l'exécution du programme, le programme s'arrêtera juste avant d'exécuter l'instruction sur laquelle on a mis le point d'arrêt.

Pour désactiver le point d'arrêt il suffit de faire les mêmes opérations. L'icône  pour désactiver tous les points d'arrêt active dans le programme.

3- Exécution, commande run

On peut lancer l'exécution d'un programme, en cliquant sur la commande *run* représenter par l'icône  . Le programme s'exécute jusqu'au premier point d'arrêt rencontrer.

4- Exécution libre, commande run free

La commande *run free* permet de lancer l'exécution d'un programme en demandant qu'il ignore tous les types des points d'arrêts.

On peut arrêter l'exécution par la commande *debug-Halt* ou par l'icône  .

5- Exécution pas à pas

On peut exécuter le programme en pas à pas en cliquant sur l'un des 3 icônes d'exécution pas à pas correspondant au 3 commandes *Debug-run to cursor* ou sur l'icône  , *Debug- stepOver* ou par l'icône  , et *Debug-stepOut* ou par l'icône .

L'exécution pas à pas de type *run to cursor* exécute les instructions l'une après l'autre.

L exécution pas à pas de type *StepOver* exécute les instruction l'une après l'autre. En considérant un appel d'une fonction comme une seule instruction et on ne détail pas les instructions dans la fonction appelée.

L'exécution pas à pas de type *StepOut* exécute les instructions l'une après l'autre, et si elle est actionnée à l'intérieur d'un sous programme, on sort du sous programme en un pas sans détailler les instructions.

6- Animation, commande animate

La commande *Debug-Animate*, que l'on peut lancer en cliquant sur l'icône  , permet de démarrer une exécution, où lorsque le programme rencontre un point d'arrêt, le simulateur mis à jour tous les fenêtres non connecter à un point sonde, puis reprend l'exécution. On règle la vitesse à l'aide de la commande *Options-Animate Speed*. On ajuste ainsi la durée minimale en secondes entre Points 2 d'arrêt.

7- Points sonde, probe points et Association des fichier à des entrée/sortie

Cette étape est très nécessaire dans la simulation sur le *Code Composer*. Les points sonde ressemblent à des points d'arrêt. On leur associe soit une fenêtre soit un fichier. Lorsque un programme atteint un point sonde, l'exécution n'est pas interrompue, mais si une fenêtre est associée au point sonde, elle est mise à jour et si un fichier est associé, il est accédé pour lire ou écrire un échantillon dans le fichier. On peut ainsi réaliser un transfert entre une mémoire du DSP et le fichier.

On peut ajouter un point sonde en cliquant sur l'icône *Probe Point* représenter par  ou en utilisant la commande *Debug-Probe point*. Pour connecter une fenêtre à un point sonde il faut d'abord que la fenêtre existe. Après avoir crée le point sonde, on sélectionne dans la liste des points sonde celui qu'on veut connecter, et on précise à quelle fenêtre ou quel fichier on veut le connecter.

Pour désactiver le point sonde il suffit de faire les mêmes opérations. L'icône  pour désactiver tous les points sonde active dans le programme.

8- Ré-initialisation du processeur

Au cours de la mise au point de l'application, il peut avoir besoin de ré-initialiser le processeur.

La commande *Debug-Reset* arrête l'exécution et initialise tous les registres dans l'état qui est le leur à la mise sous tension du DSP. La commande *Debug-Restart* recharge le compteur programme avec le point d'entrée du programme.

Il peut parfois être nécessaire de restaurer le noyau de communication entre le Kit DSK et le *PC*. La commande *Load-Karnel* charge se noyau.

2.4 L'outils de développement matériel : Le DSKVC5402

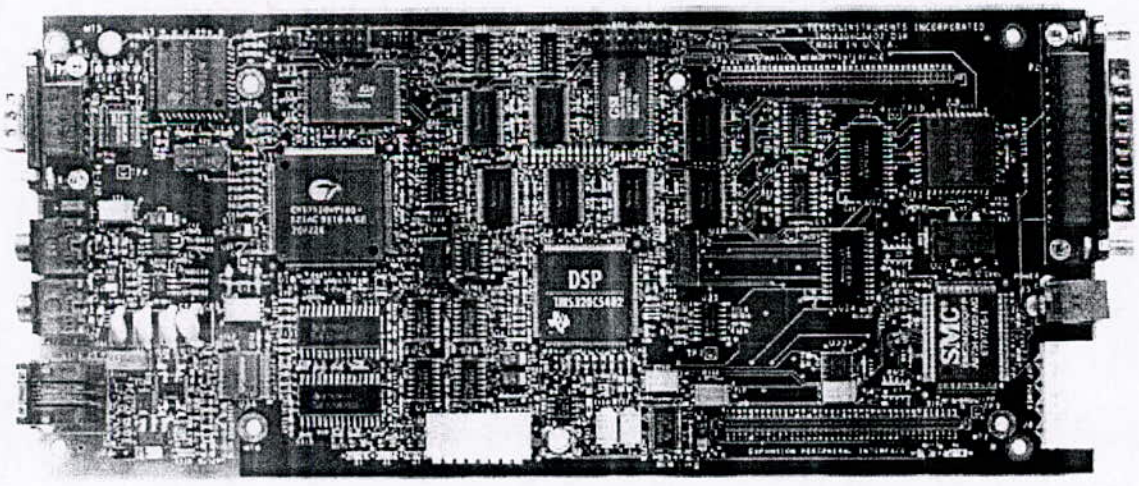


Figure 2.3 : La carte DSK C5402

Le Kit DSKVC5402 est un système à très faible coût, contenant une carte DSP TMS320VC5402 qui peut travaillée jusqua100 MHz. Il a un contrôleur de port parallèle (HPI) pour la lier par PC à l'environnement de développement intégré (CCS). Il a deux branches audio pour l'entrée audio et la sortie haut-parleur. Un connecteur d'interface

modulaire Rj-11 est inclus dans le DSK pour l'accès de DAA. De plus, le port série d'I/O RS-232 et une interface d'émulation JTAG. Cette carte est actionnée par une alimentation universelle de 5V.

Le DSKVC5402 vient également avec le CCS, le compilateur C, l'assembleur, l'éditeur de liens, et le code composer debugger.

Le Kit DSKC5402 est un système idéal pour un apprentissage des processeurs de traitement de signal, permettant de développer des applications temps réel sur des signaux physiques [12], [13], [14] et [15].

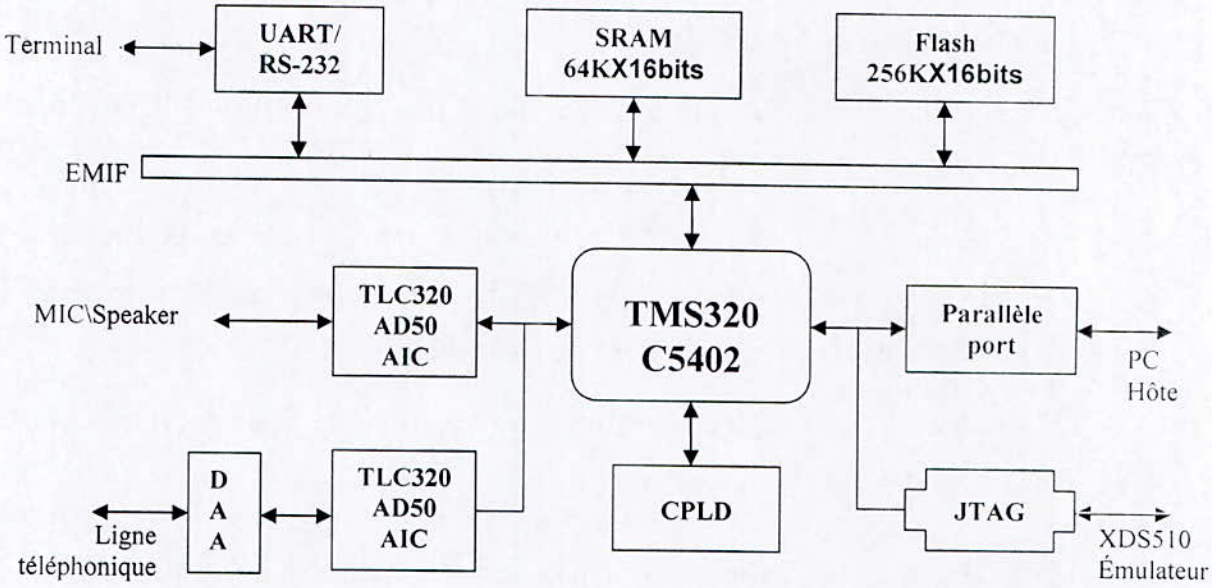


Figure 2.4 : un schéma fonctionnel du DSKVC5402

2.4.1 Description des blocs internes du DSKVC5402 :

A l'addition au DSP TMS320VC5402, Le support de développements matériels de la carte DSK basés sur les composants suivants :

- **L'interface d'émulation** : Le DSK contient une interface d'émulation JTAG (*joint test action group*) directement compatible avec le *Code Composer Debugger*, le DSK peut être utilisé par le système *XDS510* via un câble d'émulation JTAG, relié au port externe JTAG de DSK.

- **Le port parallèle** : Les communications de PC hôte au DSK sont assurées par l'intermédiaire de l'interface de port parallèle compatible *Ieee-1284*. Un connecteur *DB-25* standard est utilisé.

Le port parallèle du DSK est capable de :

- Interfacer au port HPI du DSP
- Fournir des commandes de reset
- Contrôler le TBC (pour la commande de JTAG)
- Accéder à certains registres internes du CPLD's
- Programmer le CPLD

Le PC hôte peut accéder soit à HPI ou à TBC.

- **Les mémoires externes** : Le DSP est lié à la SRAM externe, la mémoire Flash et le connecteur d'interface d'expansion mémoire par les16 bits de l'interface de mémoire externe (EMIF). Le DSK contient une SRAM de 64 Kmots de 16 bits extensibles à 256 Kmots qui fonctionne avec un état d'attente à 100MHz. On a également une mémoire Flash de 256 K mots de 16 bits. l'EMIF de DSP est également relié à un connecteur d'interface d'expansion pour permettre l'utilisation d'autres cartes périphériques.

- **L'interface téléphonique** : Une interface de réseau analogique DAA (*data acces arrangement*) et un contrôleur d'interface analogique (AIC) *TLC320AD50* permettent l'accès du DSP à l'interface téléphonique via le port série bufférisé McBSP0. Optionnellement, le McBSP0 peut être relié à un connecteur d'expansion sous la commande du logiciel.

- **L'interface audio** : deux interfaces Microphone\Speaker (via les deux prises 3.5 mm) sont fournies, par l'intermédiaire de la seconde interface analogique AD50, qui est connectée au deuxième port série bufférisé McBSP1 du DSP. Le McBSP1 (avec les signaux des deux timers et une interruption externe) est aussi relié à un connecteur d'expansion.

- **Les interfaces d'expansion** : Le DSK fournit deux connecteurs d'expansion pour :
  - Prolonger la capabilité de DSK;
  - Utiliser des E\S spécifique.

  Un connecteur d'expansion permet l'accès au EMIF et d'être utilisé pour l'extension mémoire. l'autre connecteur permet l'accès au périphérique du DSK.

- **Le CPLD** (*complex programmable logic device*): Le DSKVC5402 utilise un Cy37128 pour fournir des interfaces d'état et de contrôle pour le logiciel de DSP. Le CPLD assure les fonctions suivantes :

- Contrôle de reset
- Registres de Contrôle/Etat de memoire-mappe de DSP
- L'entretien de DSP avec les périphériques
- Contrôle l'utilisation des LED et l'état des commandes de contact (DIP)
- Contrôle d'interruption de PC hôte et de DSP
- Contrôle d'émetteurs récepteurs de données

Le dispositif est à base EEPROM que l'on peut programmer par l'intermédiaire d'une interface JTAG située à la partie haute à gauche du DSP. Les fichiers sources de CPLD sont écrits en langage VHDL (langage de conception matériel).

- **L'UART** : Le DSK contient un debug UART interfacé à l'espace d'expansion d'E\S commencé à l'adresse 0x4000. L'UART est une industrie standard TL16C550. L'UART emploie les trois lignes d'adresse A0 à A2 du DSP pour accéder aux huit registres internes. la sélection de l'UART se fait par les lignes d'adresse supérieures du DSP qui sont décodées par le CPLD.

L'UART est connecté par un convertisseur de niveau (l'interface série asynchrone) Rs-232 à un connecteur DB-9 pour tenir compte du raccordement direct au PC. L'UART peut interrompre le DSP quand un événement se produit, la ligne d'interruption de l'UART (INT) est reliée à la ligne INT1 du DSP à travers le CPLD. L'UART déclenche une interruption quand un des événements suivants se produit :

- Recevoir un état d'erreur ou une commande de coupure ;
- les données en réception sont prêtes;
- le registre de transmission de données est vide;
- Changement de l'état des broches du modem d'entrée.

Lors de la réception d'une 'interruption, le DSP doit lire le registre d'identification d'interruption (IIR) pour déterminer lesquels des quatre événements se sont produits. Le DSP peut alors entretenir l'événement en conséquence.

- **Les switchs** : Le DSK contient une commande de contacte de huit positions DIP (Dual In-line Package), pour des options externes d'utilisateurs.
- Un bouton poussoir pour le reset du DSK manuellement.
- **Les indicateurs** : Quatre LEDs, une verte utilisé comme indicateur d'alimentation (LED PWR), et trois jaunes dépendantes de l'utilisateur.

- **L'alimentation** : Une entrée d'alimentation de +5VDC, obtenue à partir d'une broche d'alimentation externe. Le DSK utilise les tensions 1.8V pour le noyau du DSP et 3.3V pour les E\S obtenues par les régulateurs de tension linéaire qui sont internes et emploient l'alimentation externe comme une source.

Les deux blocs les plus intéressants sont le DSP TMS320VC5402 et l'interface analogique (AIC) *TLC320AD50*.

2.4.2 Le DSP TMS320C5402

Le DSP TMS320C5402 présente une architecture importante par rapport aux précédents TMS320C2x et C5x, seuls les périphériques et l'interfaçage de ce processeur restent classiques [2], [5], [12], [16] et [17].

L'architecture de base est de type *Harvard* modifiée, comprenant un bus pour la mémoire programme, trois bus pour la mémoire donnée et quatre bus d'adresse. Les mots de programme et de données sont sur 16 bits et l'adressage est également sur 16 bits. Cet adressage permet d'accéder à 64 Kmots de mémoire programme, 64 Kmots de mémoire données et 64 Kmots d'entrée/sortie, soit un total de 192 Kmots.

L'accès simultané au programme et aux données permet de paralléliser les opérations : ainsi 3 lectures et une écriture mémoire peuvent être effectuées en en un temps de cycle. Cela permet le stockage du résultat parallèlement à une opération de calcul. Les données peuvent être transférées de la mémoire donnée à la mémoire programme en un temps de cycle.

L'étude de l'architecture du processeur permet de constater le fort parallélisme de l'Unité Centrale (CPU). Elle comprend un multiplieur/accumulateur non « *pipeliné* » composé d'un multiplieur 17 par 17 bits ainsi que d'un additionneur 40 bits « l'unité MAC », Les multiplications peuvent se faire sur 16 bits en format signé ou sur 17 bits en format non signé, et le résultat peut être arrondi sur 16 bits.

Une unité arithmétique et logique (ALU) sur 40 bits peut effectuer différentes opérations et de manipulation de bits, dont le résultat et stocké au choix dans un des deux accumulateurs 40 bits. La présence des deux accumulateurs à l'avantage d'éviter le stockage en mémoire de résultat intermédiaires et les calculs enchaînés, certaines opérations utilise les 2 accumulateurs simultanément.

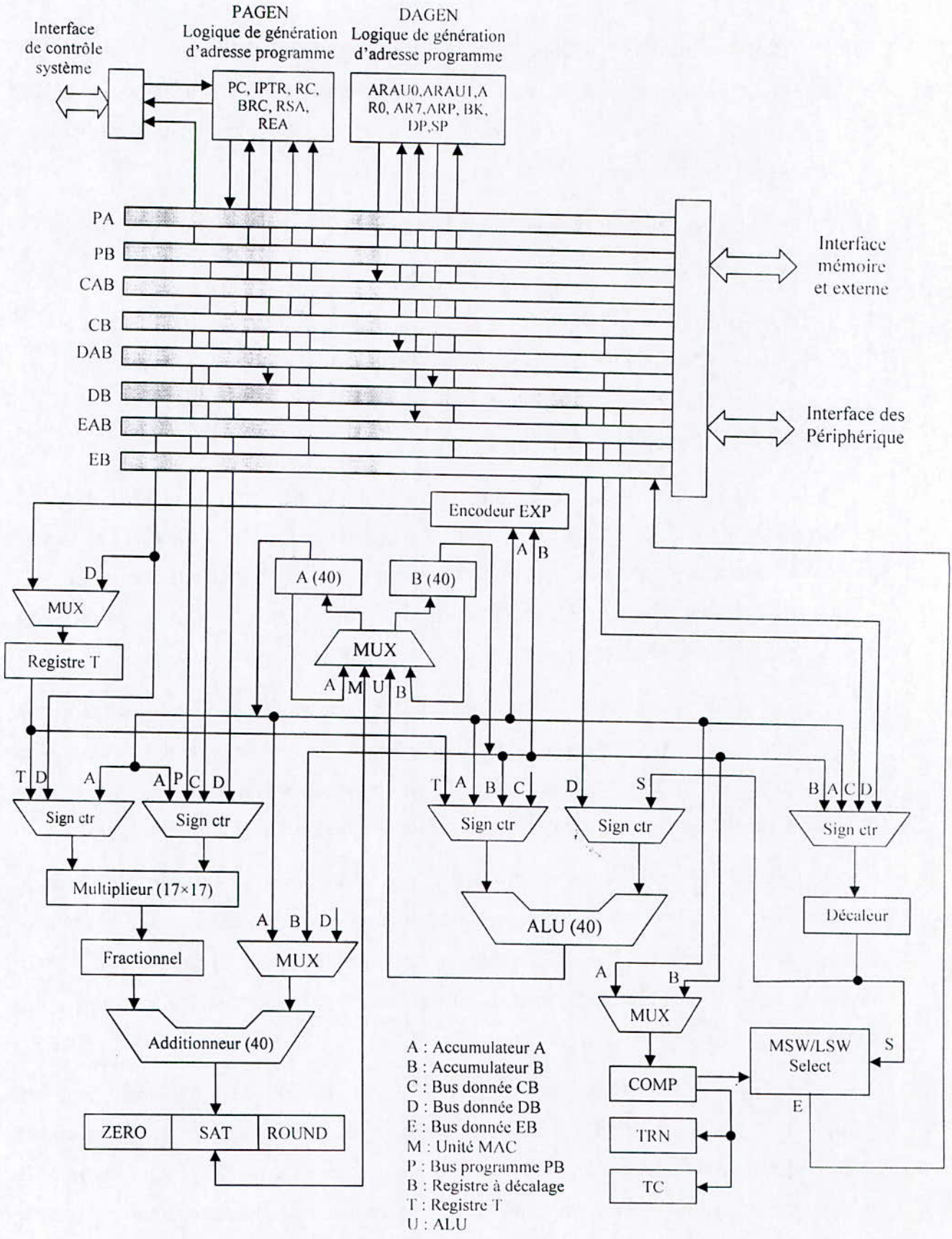


Figure 2.5 : L'architecture du processeur [17].

L'ALU effectue ses opérations en un cycle et peut travailler soit sur des mots de 32 bits, soit en deux mots de 16 bits. La normalisation de la mantisse et la recherche de l'exposant sont effectuées par le registre encodeur d'exposant également en un cycle.

Deux générateurs d'adresse, Huit registres auxiliaires et deux ARAU dédiées permettent des adressages doubles avec calculs d'adresse en parallèle.

Une unité de comparaison, sélection et stockage (CSSU) permet d'implémenter l'algorithme de Viterbi utilisé par certains modems.

- Les périphériques

Les processeurs C54x disposent de plusieurs périphériques intégrés auxquels sont associés des registres de contrôle mappés en mémoire.

Les périphériques disponibles dépendent de la version du circuit et comprennent :

- Deux timer 16 bits générant une fréquence ajustable, pour la synchronisation de certains périphériques;
- Deux ports séries bufférisés (McBSP) ;
- Un port série à multiplexage temporel (TDM) ;
- Un port d'interface hôte (HPI) ;
- Un générateur de temps d'attente ;
- Une broche drapeau XF (External Flag) qui peut être mise à 1 ou à 0 par logiciel ;
- Une broche appelée $\overline{BIO}$ qui peut être testée par le logiciel ;
- Un port JTAG pour l'émulation ;
- Un contrôleur de DMA à 6 canaux

Ces périphériques sont contrôlés par l'intermédiaire des registres situés dans le plan mémoire. L'écriture dans un registre de contrôle de périphérique demande un temps de cycle supplémentaire par rapport à une écriture mémoire.

Les données traitées par les périphériques sont stockées dans des registres situés en mémoire.

Lorsque les périphériques ne sont pas utilisés, les horloges associées sont interrompues de façon à minimiser la consommation.

2.4.3 L'AIC

Le TLC320AD50 est un AIC *(voice-bande analogue interface controller)*, permet la conversion numérique-analogique (D/A) et analogique-numérique (A/D) à haute résolution en utilisant la technologie sigma-delta. Ce dispositif se compose d'une paire de conversion synchrone de 16 bits (un pour chaque direction) et inclus un filtre passe bande anti-recouvrement en entrée (après le ADC) et un filtre de reconstitution en sortie (avant le DAC). D'autres fonctions incluses la synchronisation (taux d'échantillonnage) et le contrôle (amplificateur à gain programmable, PLL, protocole de transmission, etc.). L'architecture de sigma-delta produit la conversion de haute résolution A/D et de D/A à un faible coût de système [18].

Les principales caractéristiques de TLC320AD50 sont :

- technologie sigma-delta de 16 bits CDA et DAC
- Interface de porte série
- taux de conversion (A/D et D/A) programmable
- contrôle de gain d'entrée et de sortie programmable
- fréquence d'échantillonnage maximale 22.05 KHz
- format de données du complément à deux

La fréquence de coupure du filtre n'est pas commandée par le programmeur, et la bande passante mesurée linéairement avec la fréquence d'échantillonnage.

Le filtre anti-recouvrement ramène le débit numérique au taux d'échantillonnage. La sortie du filtre anti-recouvrement est un mot de 16 bits en format complément à deux synchronisé à la fréquence d'échantillonnage choisie. La largeur de bande du filtre est de 0.439 × (fréquence d'échantillonnage).

Le filtre de reconstitution fournit des données numériques à un taux de 256 fois le taux d'échantillonnage. La sortie des données à grande vitesse du filtre de reconstitution est alors employée dans le DAC sigma-delta afin d'obtenir une bonne résolution. La largeur de bande de ce filtre est la même que celle du filtre d'anti-recouvrement.

Dans cette étude nous intéressons à l'interface audio qui est connectée au port McBSP1 du DSP via le LTC320AD50 comme l'indique la figure 2.6.

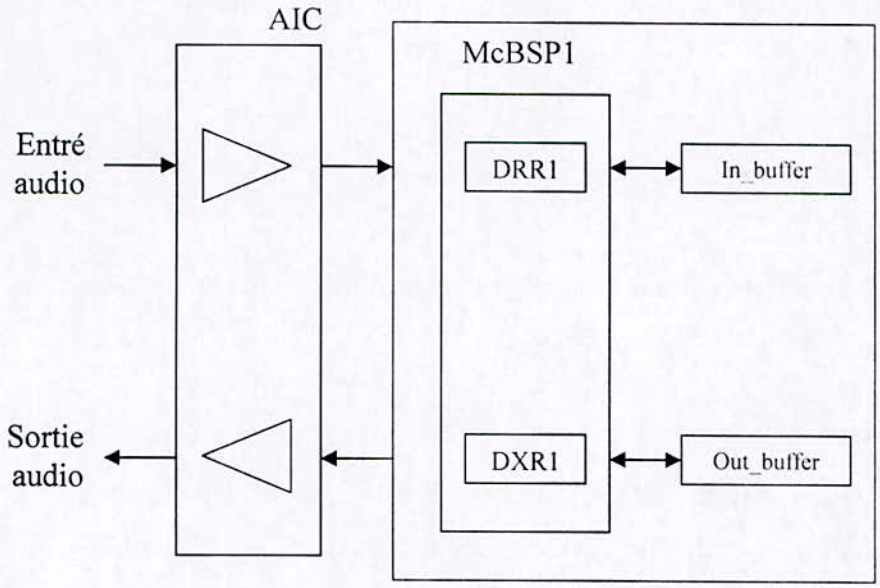


Figure 2.6 : Principe de réception et émission des données

2.5 La configuration de la carte de DSK

Pour que le CCS puisse fonctionner sur la carte de DSK, l'utilisateur doit configurer le CCS. Ceci est fait en cliquant sur l'icône **CCS2 (c5000)** du *setup*. Effacer la configuration actuelle avant d'importer les configurations pour le DSK C5402 par l'intermédiaire de l'émulateur port parallèle. Une fois la configuration est complète, après avoir cliquer sur *Save and quit*, le CCS2 peut être relancé et la marche normale s'applique pour créer le projet [9].

2.6 Gestion de projet

La bibliothèque de la carte DSK C5402 permet l'initialisation de la carte et les périphériques de contrôle qui sont exigées pour l'application [12] et [19]. Cette bibliothèque est disponible dans deux fichiers séparés (dsk5402.lib et drv5402.lib) en ajoutant des fichiers d'en-tête pour les sous-programmes de la bibliothèque. Les deux bibliothèques doivent être jointes avec le projet. Ces fichiers peuvent être ajoutés au projet en sélectionnant *projet-add file to project*. Aller au dossier c:\ti\c5400\dsk5402\lib (si le CCS est installé dans le disque C), choisir *object and library* dans le champ de type fichier, et puis sélectionner drv5402.lib et dsk5402.lib. Les fichiers d'en-tête qui sont associés à la bibliothèque du DSK C5402 doivent

être inclus dans le programme. Pendant que le port série bufférisé multicanale (McBSP) et le codec sont utilisés, codec.h et mcbsp54.h sont inclus dans le programme :

```
#include <board.h>

#include <codec.h>

#include <mcbsp54.h>
```

En outre, la bibliothèque du langage C, rts.lib, doit également être incluse dans le projet.

2.7 Conclusion

Dans ce chapitre nous avons vu les principes d'utilisations des outils de génération de code (C et assembleur) et l'outil de développement logiciel (*Code Composer Studio*) et de développement matériel (DSK VC54025). Cela nous donne une large possibilité de simuler et développer des applications pour faciliter la tache de conception.

Implantation d'un modulateur / démodulateur FM sur le DSP TMS320VC5402

3.1 Introduction

Ce chapitre constitue une présentation de la technique d'implémentation d'un modulateur démodulateur FM pour la simulation sur le *Code Composer* et l'exécution en temps réel sur le Kit de développement DSK VC5402. Les principes de base, la constitution, le fonctionnement, et l'évaluation des résultats sont représentés.

Des notions de base de la modulation et de la démodulation, jugées nécessaires pour la compréhension de la suite de ce chapitre sont représentées également.

3.2 Modèle d'un système de communication

Le but d'un système de communication est de transmettre des signaux porteurs d'information à partir d'une source à un destinataire.

Dans un système de communication en trouve, [20] :

- Un, ou des émetteurs pour rôle de modifier le signal porteur d'information à une forme plus adopté aux caractéristiques du canal
- Le canal de transmission : c'est le support de l'information
- Un, ou des récepteur pour but de reconstituer le signal original

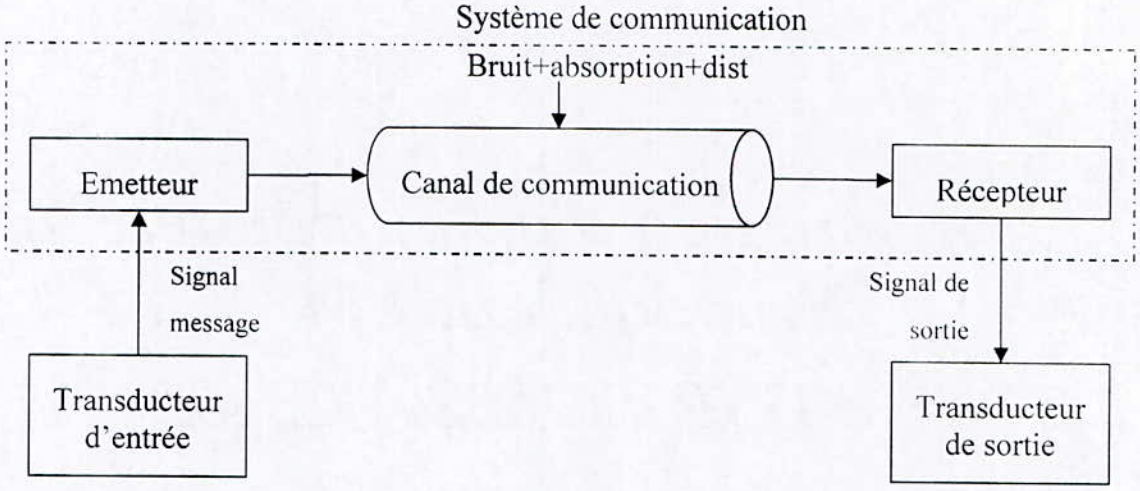


Figure 3.1 : Algorithme générale d'un système de communication

Dans les deux bloc d'émission et de réception, la mise en forme et la reconstitution du signal est le résultat de plusieurs opérations telles que le filtrage, l'amplification, la modulation et la démodulation. Ces deux dernières opérations (modulation, démodulation) c'est le but de ce chapitre tel que nous intéressons à la modulation FM, et la démodulation à l'aide d'un PLL et leurs approximations dans le temps discret pour l'implémentation sur le DSP.

3.3 La modulation FM

La modulation a pour fonction de transposer l'information sur une porteuse, afin d'occuper un espace fréquentiel que l'on a choisi. On module une porteuse dont la fréquence est beaucoup plus élevée que celle du signal message à transmettre. Cette nouvelle fréquence est plus favorable à la transmission. La modulation consiste à faire varier les caractéristiques de la porteuse (amplitude, fréquence, phase) en fonction du signal message à transmettre.

L'intérêt de la modulation réside donc dans la possibilité d'avoir un signal aisément discernable et qui se propage correctement

La modulation en fréquence (FM) ou angulaire, inventé et commercialisé après la modulation d'amplitude (AM). Leur principale avantage est représenté par leur Indépendance du niveau du signal démodulé par rapport au signal reçu, ce qui implique une meilleure immunité au bruit additive que la modulation AM. La modulation FM est très utilisé dans les application Radio, télévision et on l'utilise aussi dans les communications satellite et micro-onde, [20].

La modulation de fréquence est un procédé qui consiste à faire varier la fréquence d'un signal porteur sinusoïdal en fonction d'un signal message à transmettre.

3.3.1 Le signal FM

Le signal FM est générer par l'utilisation d'un signal message en bande de base pour fait varier la fréquence instantané de la porteuse sinusoïdale, [21].

La fréquence instantanée de la sinusoïde $\cos\theta(t)$ est définie par :

$$\omega(t) = \frac{d}{dt}\theta(t) \tag{3.1}$$

Alors la fréquence instantanée du signal FM de fréquence de porteuse ω_c est liée au signal message $m(t)$ par la relation suivante :

$$\omega(t) = \omega_c + 2\pi\ k_f m(t) \tag{3.2}$$

Tel que k_f est appeler la constante de sensibilité ou de déviation. On définie β comme l'indice de modulation FM par :

$$\beta = \frac{\text{la déviation de fréquence maximale}}{\text{la bande passante du signal modulant}}$$
$$= \frac{k_f}{f_m} \max|m(t)|$$

f_m : est la bande passante du signal modulant

L'angle $\theta(t)$ du signal FM est donnée par :

$$\theta(t) = \int_0^t \omega(\tau)\, d\tau = \omega_c t + \theta_m(t) \tag{3.3}$$

$$\theta_m(t) = 2\pi\ k_f \int_0^t m(\tau)\, d\tau \tag{3.4}$$

Le signal FM généré par le signal message $m(t)$ est donc :

$$S(t) = A_c\ \cos\left[\omega_c t + \theta_m(t)\right] \tag{3.5}$$

3.3.2 La bande passante d'un signal FM

En général, il n'existe pas une formule simple et exacte pour la bande passante d'un signal FM. Elle dépend de la forme du signal modulant, et de l'indice de modulation β, [21]. Si on considère comme significatives les raies transportant au moins 98% de la puissance moyenne du signal, la largeur de bande du signal modulé serait donnée de façon empirique par la règle de Carson :

$$\beta_T = 2(\Delta f + f_m) \tag{3.6}$$

où $\Delta f = k_f \, max|m(nT)|$ (3.7)

Δf est la déviation maximale de fréquence.

A la différence entre la modulation AM, la bande passante du signal FM ne dépend pas seulement de la bande passante du signal modulant mais on a un élargissement du spectre de Δf justifier par la non linéarité de l'opération de la modulation FM, [3].

3.3.3 La modulation FM par un signal sinusoïdale

Le spectre du signal modulé peut être exactement déterminé dans l'hypothèse d'un signal modulant purement sinusoïdale.

Soit le signal message $m(nT)=A_m \cos(w_m t)$, Le signal FM généré par ce message est :

$$S(t)=Ac \cos(w_c t + \beta \sin(w_m t)) \tag{3.8}$$

Que l'on peut l'écrire aussi sous la forme :

$$S(t) = A_c \sum_{n=-\infty}^{+\infty} J_n(\beta)\cos[(w_c + nw_m)t] \tag{3.9}$$

Où $J_n(\beta)$ est la fonction de Bessel du première espèce d'ordre-n. On peut la calculé par :

$$J_n(x) = \sum_{m=0}^{\infty}(-1)^m \frac{(\frac{1}{2}x)^{n+2m}}{m!(n+m)!} \tag{3.10}$$

Une autre relation très intéressante de $J_n(\beta)$ est donnée par le théorème de Parseval :

$$\sum_{n=-\infty}^{+\infty} J_n^2(\beta) = 1 \tag{3.11}$$

Le spectre de $S(t)$ est :

$$S(f) = \frac{A_c}{2} \sum_{n=-\infty}^{+\infty} J_n(\beta) \left[\delta(f - f_c - nf_m) + \delta(f + f_c + nf_m)\right] \tag{3.11}$$

Le spectre *S(f)* est théoriquement de largeur de bande infinie en raison de l'incrément *n*. L'analyse montre toutefois que les rais d'ordre *n* supérieure devient rapidement négligeable lorsque $|f|$ croit, de telle sorte que *S(f)* n'est sensiblement différent de zéro qu'au voisinage de $|f|=|f_c|$ [3].

3.3.4 La modulation FM en bande étroite (NFM) :

Si $\theta_m(t) \ll 1 \quad \forall t$, la modulation FM est dite à bande étroite. Et le signal modulé NFM peut approximé à [21]:

$$\begin{aligned} A_c &= A_c \cos\left[\omega_c t + \theta_m(t)\right] \\ &= A_c \cos\omega_c t \ \cos\theta_m(t) - A_c \ \sin\omega_c t \ \sin\theta_m(t) \\ &\approx A_c \cos\omega_c t - A_c \theta_m(t) \ \sin\omega_c t \end{aligned} \tag{3.12}$$

Ou en notation complexe :

$$S(t) \approx A_c \, \mathrm{Re}\left\{e^{j\omega_c t}\left[1 + j\theta_m(t)\right]\right\}$$

Le spectre du signal NFM est donné par, [20] :

$$S(f) = \frac{A_c}{2}\left[\delta(f+f_c) + \delta(f-f_c) + \phi_m(f-f_c) - \Phi_m(f+f_c)\right] \tag{3.13}$$

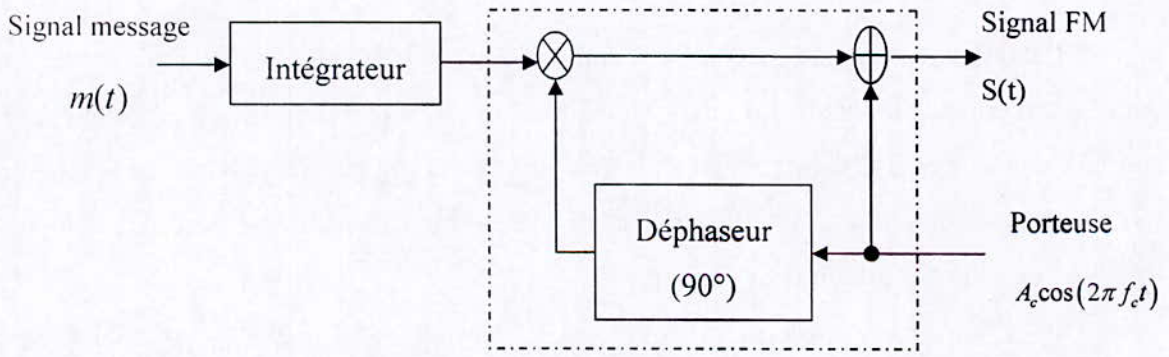


Figure 3.2 : Schéma bloc de la modulation FM à bande étroite

3.3.5 L'approximation de la modulation FM en temps discret

L'approximation du signal FM dans le temps discret est obtenue lorsque en remplace l'intégrale par la somme, [21].

L'approximation de l'angle du signal est :

$$\theta(nT) = \sum_{0}^{n-1} \omega(kT)\, T = \omega_c nT + \theta_m(nT) \tag{3.14}$$

Où :

$$\theta_m(nT) = 2\pi\, k_f T \sum_{0}^{n-1} m(kT) \tag{3.15}$$

On peut représenté l'angle $\theta(nT)$ par la relation de récurrence suivante :

$$\theta(nT) = \theta\big((n-1)T\big) + \omega_c T + 2\pi\, k_f m\big((n-1)T\big) \tag{3.16}$$

Le signal FM généré est :

$$S(nT) = A_c \cos(\theta(nT)) \tag{3.17}$$

La méthode suivie pour l'implémentation de la modulation FM sur un DSP, dépend la puissance de calcule, et de type de ce dernier.

3.4 La Démodulation FM

À la réception, il faut reconstitué le signal message afin d'être utilisé. Donc on fait une démodulation FM.

3.4.1 Démodulation FM à boucle de phase asservie (PLL)

La boucle de phase asservie est aujourd'hui l'un des composants les plus répandus dans le traitement du signal. En raison de sa fiabilité, de sa facilité de mise en oeuvre, et de son faible coût, cette structure a investi beaucoup d'appareils tels que les modems, téléphones mobiles, postes de radio AM-FM, radars aussi bien routiers que militaires, sonars, téléviseurs, processeurs microinformatiques, etc...

La démodulation FM à l'aide d'un PLL (*phase-locked loop*) peut être utilisé avec plus de performance que le discriminateur de fréquence. Son principe est : un oscillateur contrôlé en tension (VCO : *Voltage Controlled Oscillator*) produit un signal périodique que la boucle tend à rendre synchrone avec le signal incident. Si celui-ci est modulé en fréquence, la tension de commande de VCO varie proportionnellement à la déviation instantanée de fréquence, [3].

Il existe plusieurs types de circuit à boucle de phase asservie (PLL). Cependant, tous les fonctionnements sont basés sur les mêmes principes.

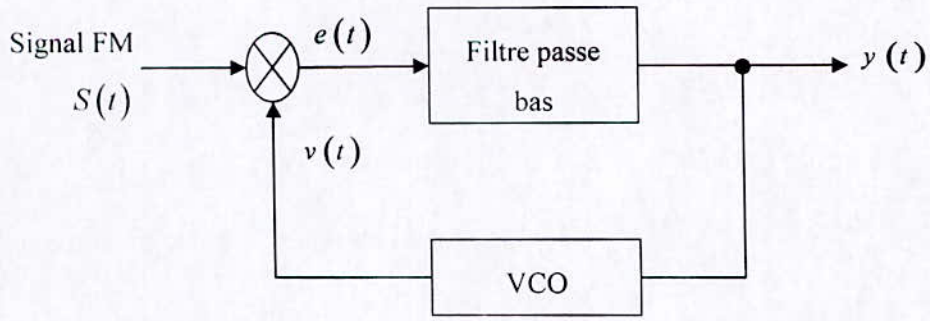


Figure 3.3 : Boucle de phase asservie

La démodulation en PLL contient 3 blocs, [20] :

- **Un comparateur de phase** : le circuit compare la phase des deux signaux *s(t)* et *v(t)*, et fournie une tension d'erreur *e(t)*dont la valeur moyen est proportionnel a la déférence de déphasage de ces deux signaux.

- **Un filtre passe bas** : Le rôle de ce filtre est d'extraire la valeur moyenne de la tension *e(t)*.

- **Le VCO** : est un oscillateur commandé en tension, pour produire une tension *v(t)* en quadrature de phase de *s(t)* approximativement.

Soit le *s(t)* signal FM à l'entré du PLL :

$$S(t) = A_c \cos\left[2\pi f_c t + \theta_m(t)\right] \quad \text{tel que} \quad \theta_m(t) = 2\pi\, k_f \int_0^t m(\tau)\, d\tau \tag{3.18}$$

A la sortie du VCO on a :

$$v(t) = A_v \sin\left[2\pi f_c t + \theta_v(t)\right] \tag{3.19}$$

tel que : $\theta_v(t) = 2\pi\, k_v \int_0^t y(\tau)\, d\tau$ (3.20)

Donc à l'entré du filtre passe bas les fréquence élevées sont éliminé, et l'erreur *e(t)* donné par :

$$e(t) = A_c A_v k_m \sin\left[\theta_e(t)\right] \tag{3.21}$$

$$\theta_e(t) = \theta_v(t) - \theta_m(t).$$

$$= 2\pi\, k_v \int y(\tau)\, d\tau - \theta_m(t). \tag{3.22}$$

Où, k_m est le gain du multiplieur, et *h(t)* est la réponse pulsionnel du filtre passe bas.

Et on a :

$$y(t) = \int_{-\infty}^{+\infty} e(\tau)\, h(t-\tau)\, d\tau \tag{3.23}$$

Donc, on écrit :

$$\frac{d\theta_e(t)}{dt} = k_0 \int_{-\infty}^{+\infty} \sin\left[\theta_e(\tau)\right] h(t-\tau)\, d\tau - \frac{d\theta_m(t)}{dt} . \tag{3.24}$$

et k_0 définie par : $k_0 = 2\pi\ k_m k_v A_c A_v$.

θ_e est très petite ($\sin\left[\theta_e(t)\right]$; $\theta_e(t)$) , alors on obtient le modèle linéaire du PLL :

$$\frac{d\theta_m(t)}{dt} = k_0 \int_{-\infty}^{+\infty} \theta_e(\tau)\, h(t-\tau)\, d\tau - \frac{d\theta_e(t)}{dt} \tag{3.25}$$

Si en passe à la transformer de Fourier :

$$\Phi_e(f) = \frac{1}{L(f)-1}\, \Phi_m(f) \tag{3.26}$$

Où : $L(f) = k_0 \dfrac{H(f)}{j2\pi f}$. (3.27)

Et on a : $Y(f) = \dfrac{k_0}{2\pi\ k_v} H(f)\Phi_e(f)$

$$= \frac{jf}{k_v} L(f)\Phi_e(f) . \tag{3.28}$$

Alors :

$$Y(f) = \frac{\left(jf/k_v\right)L(f)}{L(f)-1}\, \Phi_m(f) \tag{3.29}$$

Lorsque on pose $|L(f)|$? 1 , on peut approximer Y(f) à :

$$Y(f) ; \frac{jf}{k_v}\, \Phi_m(f) . \tag{3.30}$$

Qui correspond, dans le domaine temporaire à :

$$Y(t) ; \frac{k_f}{k_v}\, m(t) . \tag{3.30}$$

3.4.2 L'approximation de la démodulation FM à PLL en temps discret

Pour l'implémentation sur le DSP, on utilise l'approximation en temps discret de la boucle PLL précédente, où le comparateur de phase est donné par un multiplieur pour adaptée les calculs aux caractéristiques du DSP C54x, et le filtre passe bas est réalisé par un filtre IIR pour minimisé le nombres de temps de cycles consommé par le filtre.

Après le filtrage et l'échantillonnage du signal modulé en FM *S(t)* reçue, on passe à l'entré du PLL où on a le signal *S(nT)* tel que, [21] :

$$S(nT) = A_c \cos\left[\omega_c nT + \theta_m(nT)\right] \quad . \tag{3.31}$$

S(nT) est multiplié par la sortie du bloc VCO, d'après le schéma de la Figure3.4, la phase du VCO est donné par :

$$\phi\left((n+1)T\right) = \phi(nT) + \omega_c T + 2\pi\ k_v T\ y(nT) \quad . \text{ Pour } n \geq 0. \tag{3.32}$$

Tel que :

$$\phi(nT) = \omega_c nT + \theta_v(nT) \quad .$$

Et :

$$\theta_v(nT) = \theta(0) + 2\pi\ k_v T \sum_{k=0}^{n-1} y(nT) \quad .$$

La sortie du VCO :

$$\begin{aligned} v(nT) &= A_v \sin(\phi(nT)) \\ &= A_v \sin\left[\omega_c nT + \theta_v(nT)\right] \end{aligned} \tag{3.33}$$

La sortie du multiplieur :

$$e(nT) = \frac{A_c A_v}{2} \sin\left[\theta_v(nT) - \theta_m(nT)\right] \tag{3.34}$$

tel que les composantes des fréquences hautes sont éliminées par le filtre passe bas [20].

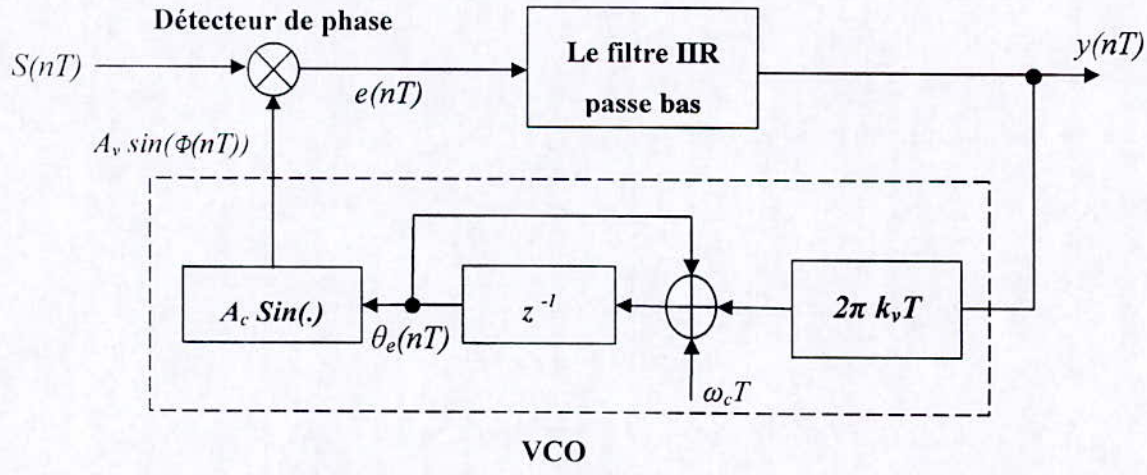


Figure 3.4 : PLL en temps discret

L'erreur de phase entre le signal d'entrer FM et le signal de sortie de VCO est l'angle de signal de sortie du multiplieur :

$$\theta_e(nT) = \theta_v(nT) - \theta_m(nT) \tag{3.35}$$

La sortie du détecteur de phase est appliqué à l'entré de filtre IIR de fonction de transfert $H(z)$, pour éliminer les fréquence hautes et minimisé l'erreur de phase $\theta_e(nT)$.

Le PLL est un système non linéaire à cause des caractéristiques de détecteur de phase, on peut linéarisé le système PLL comme suit :

$$L(z) = \frac{Y(z)}{\phi_m(z)} = \frac{H(z)}{H(z)\dfrac{2\pi\ k_v T\ z^{-1}}{1-z^{-1}} - 1} \tag{3.36}$$

À des fréquences bas, où $z \simeq 1$, $L(z)$ est approximé à:

$$L(z)\ ;\ \frac{z-1}{2\pi\ k_v T} \tag{3.37}$$

donc:

$$Y(z)\ ;\ \phi_m(z)\frac{z-1}{2\pi\ k_v T} \tag{3.38}$$

dans le domaine temporaire :

$$y(nT)\ ;\ \frac{\theta\big((n+1)T\big) - \theta_m(nT)}{2\pi\ k_v T} \tag{3.39}$$

et enfin :

$$y(nT)\ ;\ \frac{k_\omega}{k_v} m(nT). \tag{3.40}$$

Le modèle linéaire du PLL utilisé :

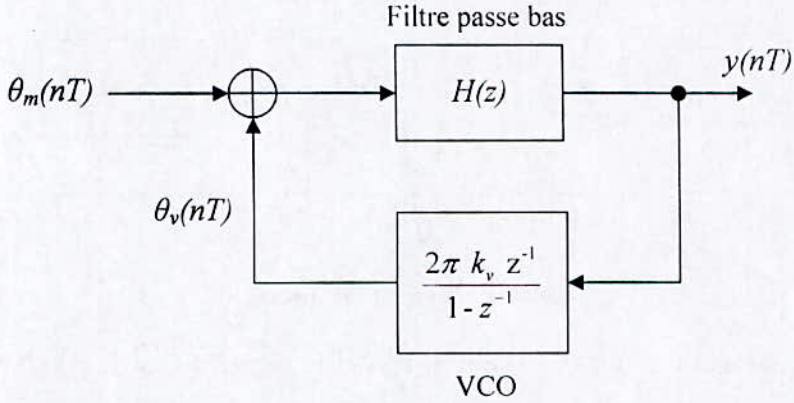


Figure 3.5 : Le modèle linéaire du PLL

3.4.3 Le filtre IIR passe bas

Le filtre *H(z)* est un estimateur de signal modulant (message). les paramètres du filtre sont choisie d'une façon que la bande passante est suffisante pour le signal message en bande de base, et aussi doit être plus étroite que possible à cause de bruit.

Le choix du filtre IIR est justifié par leurs bonnes caractéristiques sur la bande passante et l'atténuation donnée, avec un ordre inférieur que le filtre FIR, pour obtenir une bonne estimation du signal modulant avec un minimum de temps de calcul sur l'échantillon à filtré.

3.5 Implantation du modulateur et démodulateur FM sur le C54x

Dans ce projet pour l'implantation du modulateur FM sur la carte DSK, on utilise l'interface audio tel que l'interface microphone est utilisée pour l'acquisition des données, soit le signal message dans le cas de la modulation ou le signal FM dans le cas de la démodulation. A la sortie du convertisseur A/D, Le premier échantillon de l'interface analogique AD50 est transféré à la mémoire des données du DSP par l'intermédiaire du port série McBSP1 pour le traité. Après le traitement de cet échantillon par l'intermédiaire du port série McBSP1, et après la conversion D/A on le transmet par l'interface speaker à l'extérieure de la carte, et en recommence le traitement pour le nouvel échantillon. Cependant on utilise dans le programme, les fonctions d'initialisation du C5402, McBSP et des deux convertisseurs DAC et ADC ainsi que le programme d'interruption fournies par Texas Instruments.

Dans le cas de la simulation, on prend en considération de toutes les remarques précédentes. Les signaux d'entrées et de sorties sont sauvegardés en mémoire.

Les signaux d'entrée et de sortie à la carte sont représentés avec la précision de 16 bits. Les valeurs intermédiaires peuvent être maintenues à la précision de 32 bits afin de réduire l'accumulation des erreurs dues aux représentations en précision finie.

La génération de forme d'onde sinusoïdale est assurée par la méthode de lecture en table exposée dans le paragraphe suivant.

3.5.1 Calcul des cosinus et des sinus par lecture en table [22]

Pour générer les valeurs de cosinus et de sinus, on utilise la méthode de lecture en table (*lookup-table*). C'est la méthode la plus flexible et conceptuellement simple pour

produire des formes d'onde sinusoïdales. La technique implique simplement la lecture d'une série de données stockées en mémoire, représentant les échantillons discrets de forme d'onde à produire. Les valeurs de données peuvent être obtenues par un prè-échantillonnage de la sinusoïde dans une période. Le signal sinusoïdal est généré par accès aux emplacements mémoires où les échantillons sont stockés, en utilisant un buffer circulaire. Cette technique est également employée pour produire la musique de l'ordinateur.

La table de sinusoïde contient les valeurs des échantillons espacés en équidistance dans une période. La table de sinusoïde de N points peut être calculée en évaluant la fonction

$$x(n) = \cos(\frac{2\pi i}{NT}), \qquad i = 0,1,\ldots,N-1.$$

Comme les DSP C54x sont en format virgule fixe, donc Ces valeurs d'échantillons doivent être représentées en entier. L'erreur commise sur le calcul des échantillons de la fonction sinusoïdale est déterminée par l'arithmétique en précision finie ; format fixe employé pour représenter ces données et la longueur de la table N.

Une sinusoïde de fréquence f désirée est produite en lisant les valeurs stockées dans la table, avec une constante de saut Δ (l'incrément d'adresse). L'index de lecture s'enveloppe autour de l'extrémité de la table toutes fois ce index excède N - 1. La fréquence de la sinusoïde produite dépend de la fréquence d'échantillonnage T, la longueur de la table N et de l'incrément d'adresse Δ.

$$f = \frac{\Delta}{NT} \text{ Hz}. \tag{3.41}$$

Alors, pour une table de sinusoïde de longueur N, une sinusoïde de fréquence f à une fréquence d'échantillonnage f_s, peut être produite en employant l'incrément d'adresse Δ :

$$\Delta = \frac{Nf}{f_s}. \tag{3.42}$$

$\Delta \le \frac{N}{2}$ Pour respecter le théorème de Shannon.

C'est-à-dire pour générer L échantillons d'une sinusoïde $x(l)$; $l=0,1,\ldots,N-1$ de fréquence f, en utilisant un index de lecture circulaire k tel que :

$$k = (m + l\Delta)_{\text{mod } N} \tag{3.43}$$

Où m détermine la phase initiale de la sinusoïde. Il est important de remarquer que l'incrément Δ peut être un nombre réel et $(m + l\Delta)$ est un nombre entier. C'est-à-dire, Δ est composé d'une partie entière et d'une partie fractionnelle. Donc si ces valeurs fractionnelles de

Δ sont utilisées, des échantillons de points entre les entrées de la table doivent être estimés. La solution consiste à rendre cet l'incrément Δ de nombre réel au nombre entier le plus proche.

Deux sources d'erreur dans l'algorithme de lecture en table causent la distorsion harmonique :

1. Une erreur de quantification d'amplitude est produite par la représentation des valeurs de la table de la sinusoïde avec des nombres en précision finie.

2. Des erreurs de quantification de temps sont présentées quand des points entre les entrées de la table sont prélevées, qui augmentent avec l'incrément d'adresse Δ.

Plus la table est longue, la deuxième erreur sera moins significative. Pour réduire la condition de mémoire, et pour produire une sinusoïde en précision élevée, nous pouvons tirer profit de la symétrie de forme d'onde, qui a en effet comme conséquence, une duplication des valeurs stockées. Par exemple, les valeurs sont répétées (indépendamment du changement de signe) quatre fois chaque période. Ainsi seulement un quart de la mémoire est nécessaire pour représenter la forme d'onde. Cependant, on aura une complexité de l'algorithme pour connaître quel quadrant du cercle de la forme d'onde qui doit être produit et avec le signe correct. Le meilleur compromis sera déterminé par la mémoire et la puissance disponibles de calcul pour une application donnée sur le DSP cible.

3.5.2 Le modulateur FM

3.5.2.1 Le choix des paramètres de la modulation FM

Pour implanter un modulateur FM il faut calculer la phase $\theta(nT)$ de l'équation (3.14), et on déduit par lecture en table le signal FM de (3.17) .

la porteuse $A_c \cos(2\pi f_c nT)$ est générée par le DSP, où $f_s=1/T$ est la fréquence d'échantillonnage des convertisseurs A/D et D/A de L'AIC . Cette fréquence d'échantillonnage est programmable de valeur maximale *16 KHz*. Comme dans notre projet le signal FM est généré par le DSP, alors le choix de la fréquence de la porteuse f_c dépend principalement de la fréquence de coupure haute de $S(nT)$, $f_{copure}=f_c+B_T/2$, qui doit être inférieure à $f_s/2$.

Pour une bonne visualisation du signal FM de sortie on choisit $f_c=1600Hz$, et la fréquence haute du signal message $f_m \leq 500\ Hz$. Il ne reste que le choix sur la déviation maximale de fréquence Δf qui dépend de la valeur maximale du signal message qui est

supposé très faible, et de la constante de modulation k_f que l'on suppose un multiple de 2 pour rendre les calculs dans le programme plus précis.

On a :

$\Delta f = max|\ m(nT)|\ k_f$, où $max|\ m(nT)|$ est faible.

Alors $\Delta f < k_f$ et $\Delta f \leq fs/2-(fc+fm)$.

Pour notre choix $fs/2-(fc+fm) = 5900\ Hz$. Donc il nous suffit de prendre $k_f = 2^{10}\ Hz$ ce choix n'est pas seulement pour rendre les calculs plus précis mais aussi pour vérifier la condition sur le spectre de signal dans la phase de démodulation qui sera exposé ultérieurement.

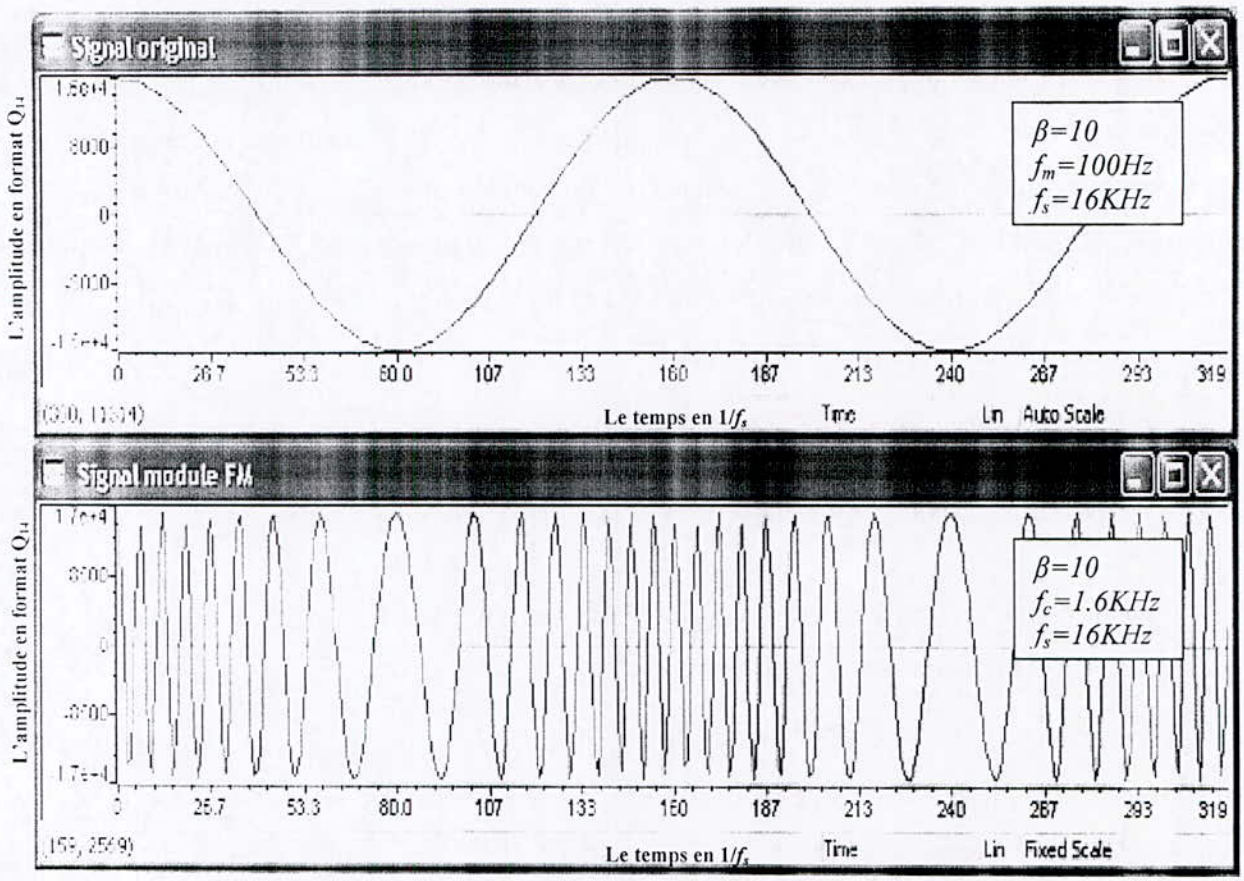


Figure 3.6 : la modulation FM

3.5.2.2 Implantation de la modulation FM sur un DSP C54x

On va exposer la technique utilisée dans le programme de la modulation FM, sachant qu'on a pris les paramètres précédents. Le signal FM est généré par la méthode de lecture en table, et en travaillant sur des entier en format *Q14* qui rend l'erreur de quantification d'amplitude $\leq 2^{-k}$.

La méthode de lecture en table est très utile pour la génération d'une sinusoïde même si la fréquence varie avec le temps comme dans le cas de la modulation FM. L'incrément d'adresse Δ dépend de N, f_s, et la fréquence du signal désiré par la relation (3.42).

Où $f=f_c+k_f m(nT)$ et $fs=16KHz$.

$m(nT)$ est le signal message en format $Q14$. Alors avec l'erreur commise sur le calcul de la fréquence f à cause de la quantification d'amplitude du signal message, on cherche à diminuer la distorsion harmonique due à la quantification du temps par un choix convenable de la taille N *de la table.*

Si $N= \alpha \times f_s$ et α est un entier, alors Δ est un entier si k_f et f_c sont des entiers, ce qui est notre cas ($f_c=1600Hz$, $k_f=2^{10}$). Mais avec cette solution nous avons besoin de $\alpha \times f_s$ mots mémoire, on prend alors $\alpha = 1$ et on ne stocke en mémoire que la partie de la sinusoïde du premier quadrant de la forme d'onde pour réduire l'espace mémoire utilisé par la table de la sinusoïde.

Avec ces conditions Δ devient égale à f, et comme $k_f=2^{10}$ et $m(nT)$ en format $Q14$ on peut écrire :

$$f(nT)=f_c+ m(nT). \tag{3.44}$$

Où $m(nT)$ est considéré comme des échantillon en format Q_4 par leur représentation en format Q_{14}, et cela justifie le choix de k_f.

La table de la sinusoïde utilisée, contient $N/4+1$ échantillons en format Q_{14} calculés par :

$$x(nT)=2^{14}cos(2\pi i / N) \qquad i=0,1,...,N/4.$$

La lecture en table est directement liée à la fréquence du signal sinusoïdal désiré. Dans la génération du signal FM $S(nT) = A_c cos(\theta(nT))$, pour rendre les calculs plus simples et le programme plus efficace, on évite le calcul d'intégrale de $m(nT)$ tel que les calculs donnés par (3.14) et (3.15) sont remplacés par le calcul de la fréquence instantanée, donnée par la relation (3.44), et le calcul d'index k pour la lecture de la table donnée par (3.43) où $m=0$.

L'expression (3.43) n'est pas utilisable directement dans la boucle infinie car la valeur de n doit être limitée, donc il faut une relation de récurrence entre $k(n)$ et $k(n-1)$ donnée par l'identité $(a+b)_{mod\ N}=(a_{mod\ N}+b_{mod\ N})_{mod\ N}$ qui nous conduit à :

$k(n)=((n-1)\Delta_{mod\ N}+\Delta_{mod\ N})_{mod\ N}$, et comme $\Delta \leq N/2$.

Alors $k(n)=(k(n-1)+\Delta)_{mod\,N}$.

3.5.2.3 Simulation de la modulation FM sur le Code Composer

On utilise deux buffers Signal_Original et Signal_Module_FM pour simuler les changements des données entre la mémoire des données et le port série McBSP1, le premier buffer est utilisé pour les échantillons d'entrée et le deuxième pour les échantillons de sortie.

3.5.3 Le démodulateur FM

Dans cette partie, on va exposer la technique d'implantation d'un démodulateur FM à l'aide d'un PLL, avec les mêmes paramètres utilisés pour la modulation.

3.5.3.1 Implantation de la démodulation FM sur un DSP C54x

Le circuit PLL utilisé est donné par la figure3.4, où le bloc VCO est un modulateur FM, dont l'entrée est le signal original estimé (signal démodulé), et la sortie est le signal $V(nT)$ généré par la méthode de lecture en table.

Pour générer le signal $V(nT)$ par la méthode de lecture en table, on fait varier sa fréquence proportionnellement au signal original estimé. Et comme le signal $V(nT)$ doit être déphasé de $\pi/2$ par rapport au signal modulé FM, alors on initialise l'index de lecture de la table k par la valeur m dans l'équation (3.43).

Le signal modulé FM est un cosinus, et la longueur de la table en lecture $N=16000$. Alors pour générer le signal $V(nT)$ déphasé de $\pi/2$ par rapport au signal FM, en commence la lecture de la table par le quatrième quadrant correspond à $m=12000$.

Le multiplieur est réalisé par une simple multiplication de $V(nT)$ par $S(nT)$ donc :

$e(nT)=V(nT)\,S(nT)$.

$$=\frac{A_c A_v}{2}\left\{\sin\left[\theta_c(nT)\right]+\sin\left[2w_c nT+\theta_m(nT)+\theta_v(nT)\right]\right\} \qquad (3.45)$$

Le deuxième sinus de $e(nT)$ est une modulation FM avec une fréquence de porteuse $2f_c$ donc on peut estimé leur bande passante β_T' par la règle empirique de Carson :

$\beta_T'=2(\Delta f'+f_m)$.

Lorsque le système se stabilise $\Delta f' \approx 2\,\Delta f$. Donc on a un élargissement de spectre qu'il faut le limité dans le choix des paramètres de la modulation pour éviter le chevauchement des spectres des deux sinus de $e(nT)$.

3.5.3.2 Détermination des coefficients du filtre passe bas :

Dans l'implantation de la démodulation FM en boucle asservie (PLL) , le choix du filtre numérique passe bas est très important pour l'adapter au DSP cible, et pour répondre au paramètre de la modulation.

Le filtre passe bas utilisé est un filtre IIR, que l'on peut déterminer par une des différentes méthodes existantes, telle que la méthode de l'impulsion invariante, et la méthode de la transformer bilinéaire [22]. Certains logiciels peuvent être utilisés pour faciliter la tâche de conception. En utilisant MATLAB pour calculer les coefficients du filtre, et pour tracer sa réponse fréquentielle. MATLAB est capable de concevoir un filtre IIR de type Butterworth, ChebyshevI, ChebyshevII, et filtre elliptique dans les quatre cas différents du filtre : passe-bas, passe-haut, passe-bande et stop-bande. La conception du filtre IIR employant MATLAB exige deux paramètres ; l'ordre du filtre L et la fréquence de coupure normalisée par rapport à la fréquence de Nyquist $(f_s/2)$ correspondante à l'atténuation désirée R_s en dB [23].

Dans notre cas, nous intéressons au filtre IIR passe bas de type ChebyshevII pour une meilleur atténuation $R_s=60\ dB$ où $Wn=1000/8000$ avec un ordre $L=6$, les coefficient de numérateur et de dénominateur sont donnés dans les deux vecteurs b et a respectivement par la commande MATLAB :

[b,a] = cheby2(L, R_s, Wn).

Donc le filtre IIR de ChebyshevII conçu par MATLAB passe-bas d'ordre 6 de fonction de transfert $H(z)$ est donnée par:

$$H(z)=\frac{0.0012-0.0048\,z^{-1}+0.0094\,z^{-2}-0.00114\,z^{-3}+0.0094\,z^{-4}-0.0048\,z^{-5}+0.0012\,z^{-6}}{1-5.1611\,z^{-1}+11.1507\,z^{-2}-12.9033\,z^{-3}+8.4316\,z^{-4}-2.9491\,z^{-5}+0.4312\,z^{-6}}$$

Pour minimiser le risque d'*Overflow* due aux quantification des coefficients, et pour limiter l'accumulation des erreurs, le filtre est implanté en cascade, c'est-à-dire en mettant la fonction de transfert $H(z)$ sous la forme d'un produit de trois fonction de transfert [2].

$H(z)=G\,H_1(z)\,H_2(z)\,H_3(z)$.

Où G est un gain : $G=0.0012$.

$$H_1(z)=\frac{1-0.5147\,z^{-1}+z^{-2}}{1-1.5846\,z^{-1}+0.6311\,z^{-2}}$$

$$H_2(z)=\frac{1-1.7067\,z^{-1}+z^{-2}}{1-1.7063\,z^{-1}+0.7494\,z^{-2}}$$

$$H_3(z) = \frac{1 - 1.8373\, z^{-1} + z^{-2}}{1 - 1.8702\, z^{-1} + 0.9117\, z^{-2}}$$

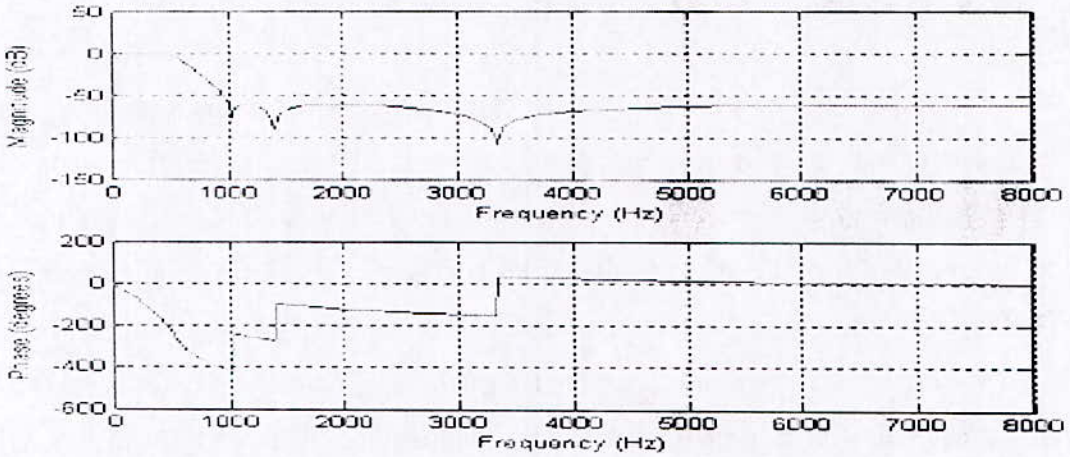


Figure 3.7 : La réponse fréquentielle en module et en phase de *H(z)*

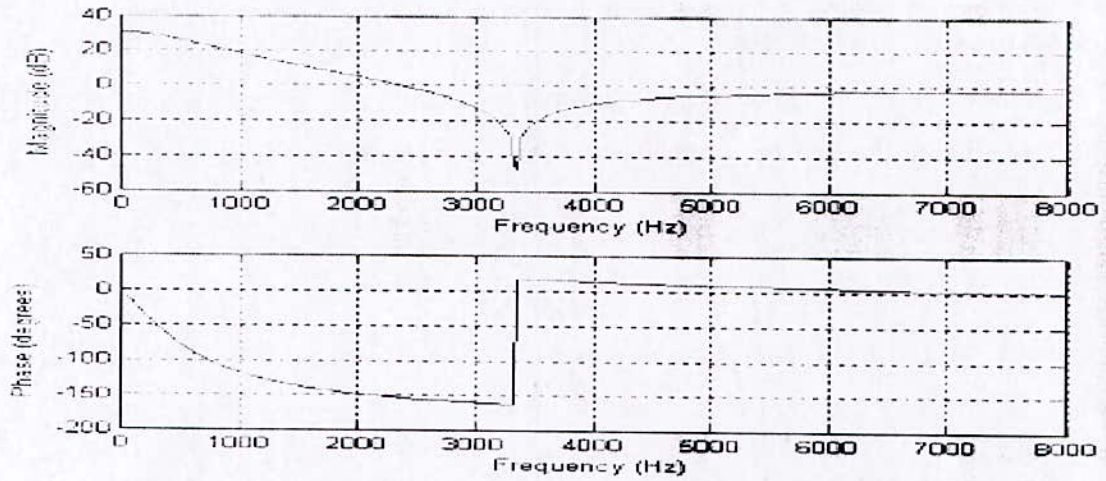


Figure 3.8 : La réponse fréquentielle en module et en phase de $H_1(z)$

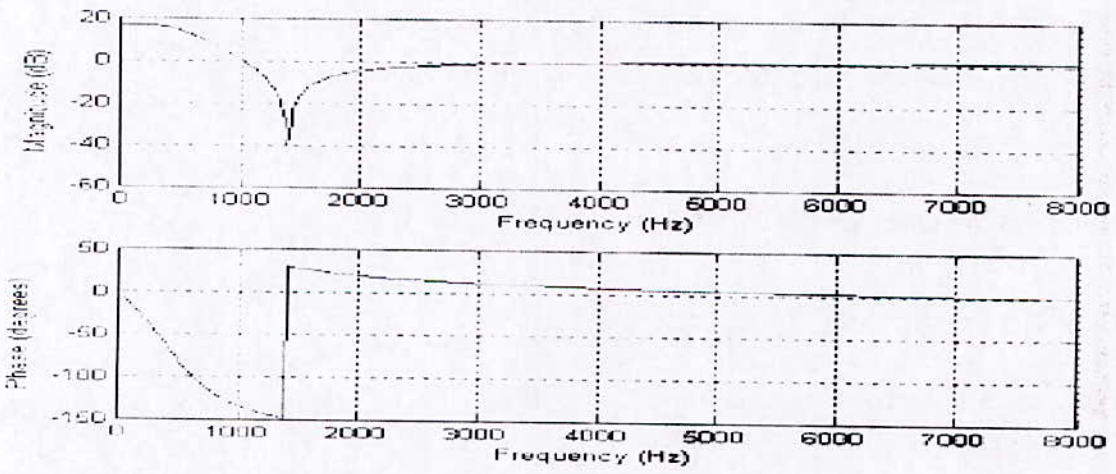


Figure 3.9 : La réponse fréquentielle en module et en phase de $H_2(z)$

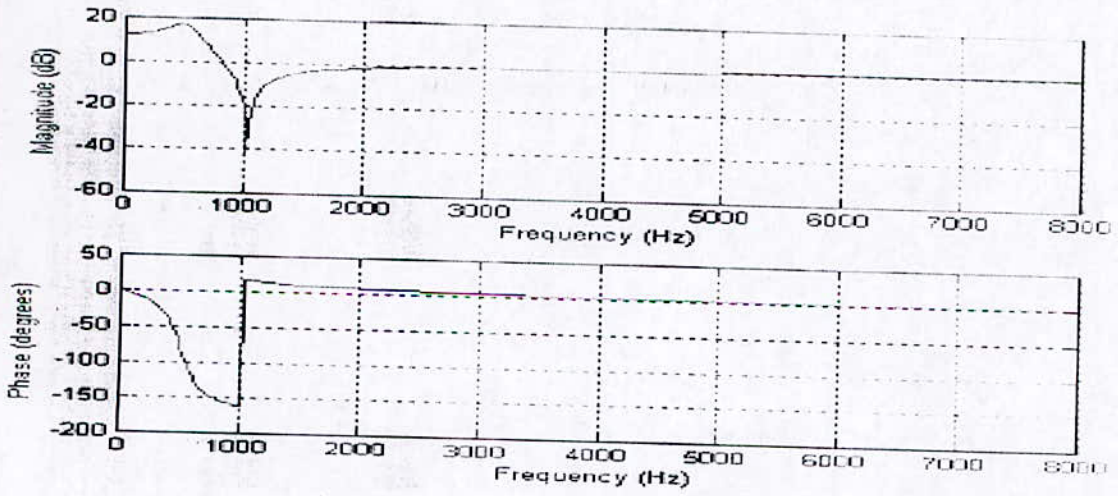


Figure 3.10 : La réponse fréquentielle en module et en phase de $H_3(z)$

Des facteurs d'échelle sont insérés entre les cellules pour éviter la saturation des signaux, $G=G_0G_1G_2\,G_3$. La structure à implanter est de la forme :

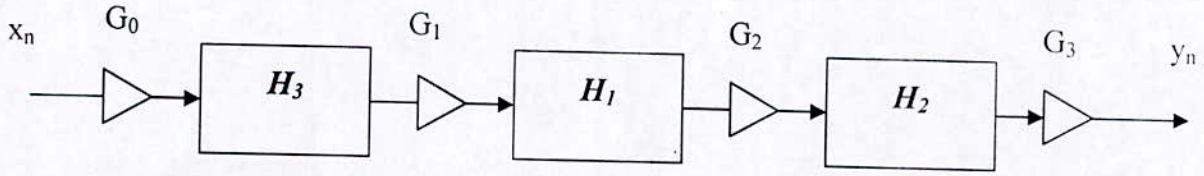


Figure 3.11: Structure en cascade et facteurs d'échelle

Les valeurs des facteurs d'échelle sont :

G_0=0.2314

G_0=0.0972

G_0=0.1803

G_0=0.2960

Les facteurs d'échelle sont choisis d'une façon à baisser le gain maximal de chaque fonction de transfert afin d'éviter la saturation des signaux causés par la quantification de calcul.

3.5.3.3 La simulation de la démodulation FM sur le Code Composer

Pour la simulation de la démodulation sur le code composer, on utilise le buffer Signal_Module_FM utilisé dans la simulation de la modulation, pour représenter le signal d'entrée et le buffer Signal_Estime qui représente le signal message estimé. Les signaux de

sorties du VCO et du multiplieur sont représentés par les deux buffers Sortie_VCO et Sortie_Multiplieur respectivement.

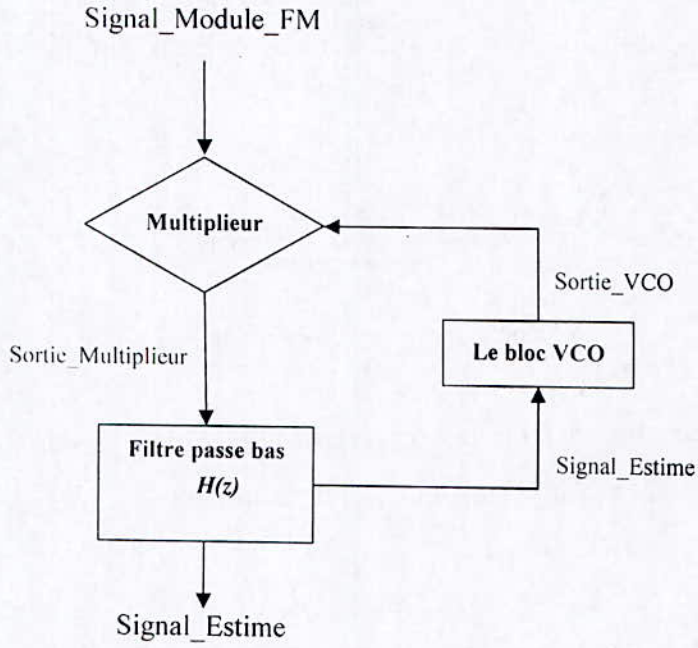


Figure 3.12 : L'organigramme du programme de simulation du modulateur FM

3.6 Résultats de la simulation

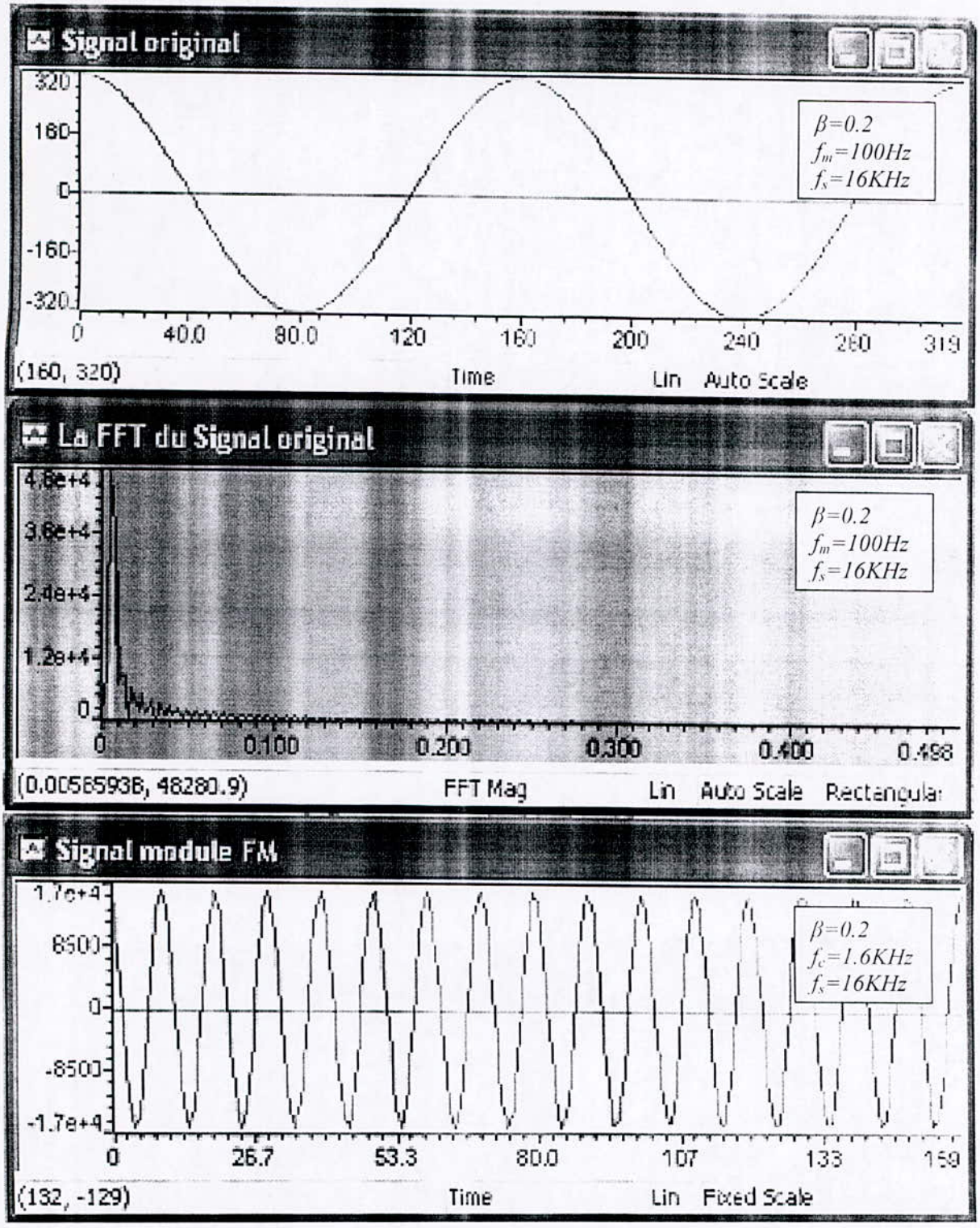
On a présenté dans ce paragraphe les résultats de la simulation pour la modulation et de la démodulation FM.

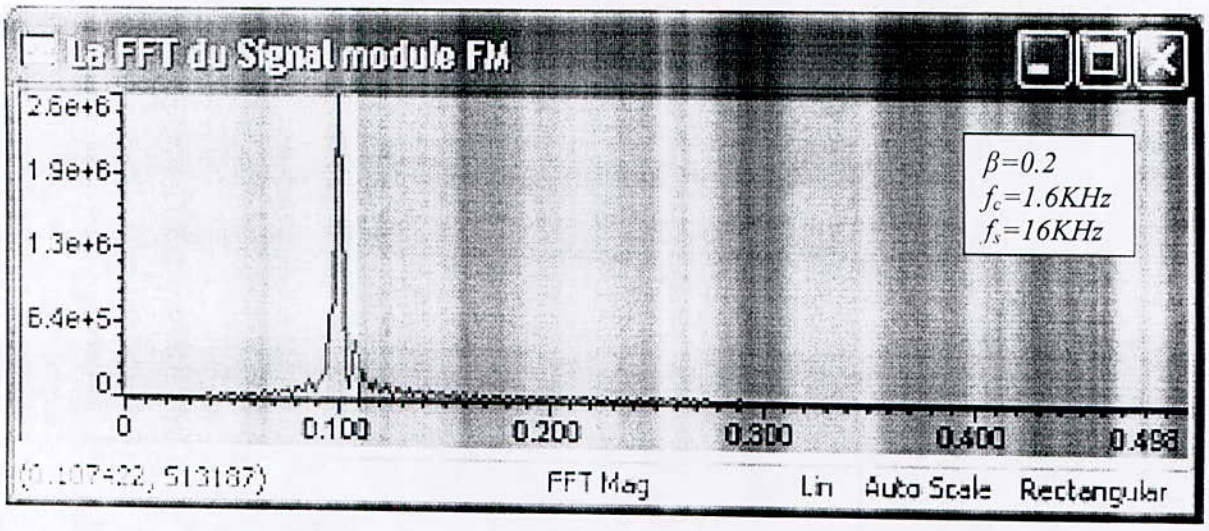
3.6.1 La modulation FM

On trace les deux zones mémoires correspondantes à ces deux buffers en temps et en fréquence par l'outil graphique fournit par le *Code Composer* pour différentes valeurs de l'indice de modulation β. Les données d'entrées sont fournies par l'association d'un fichier des données au buffer Signal_Original.

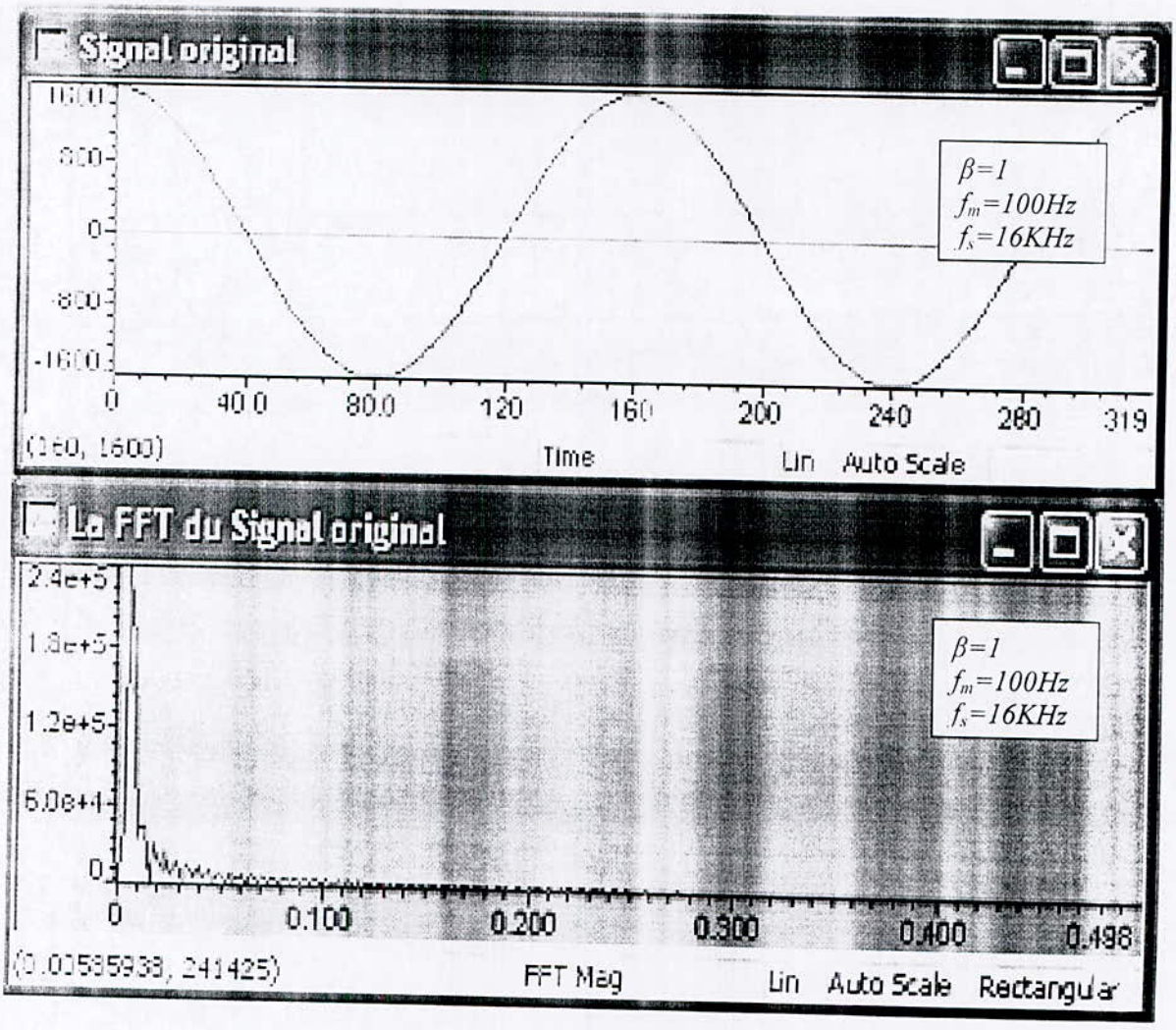
Pour les deux tracés, l'échelle des ordonnées est en format Q_{14}, chaque unité est de $1/2^{14}$. L'échelle des abscisses est en $1/f_s$ pour chaque unité. On représente le signal message $m(nT)$ en deux période, et le signal modulé en FM $S(nT)$ correspondant à une période de $m(nT)$ pour différents β.

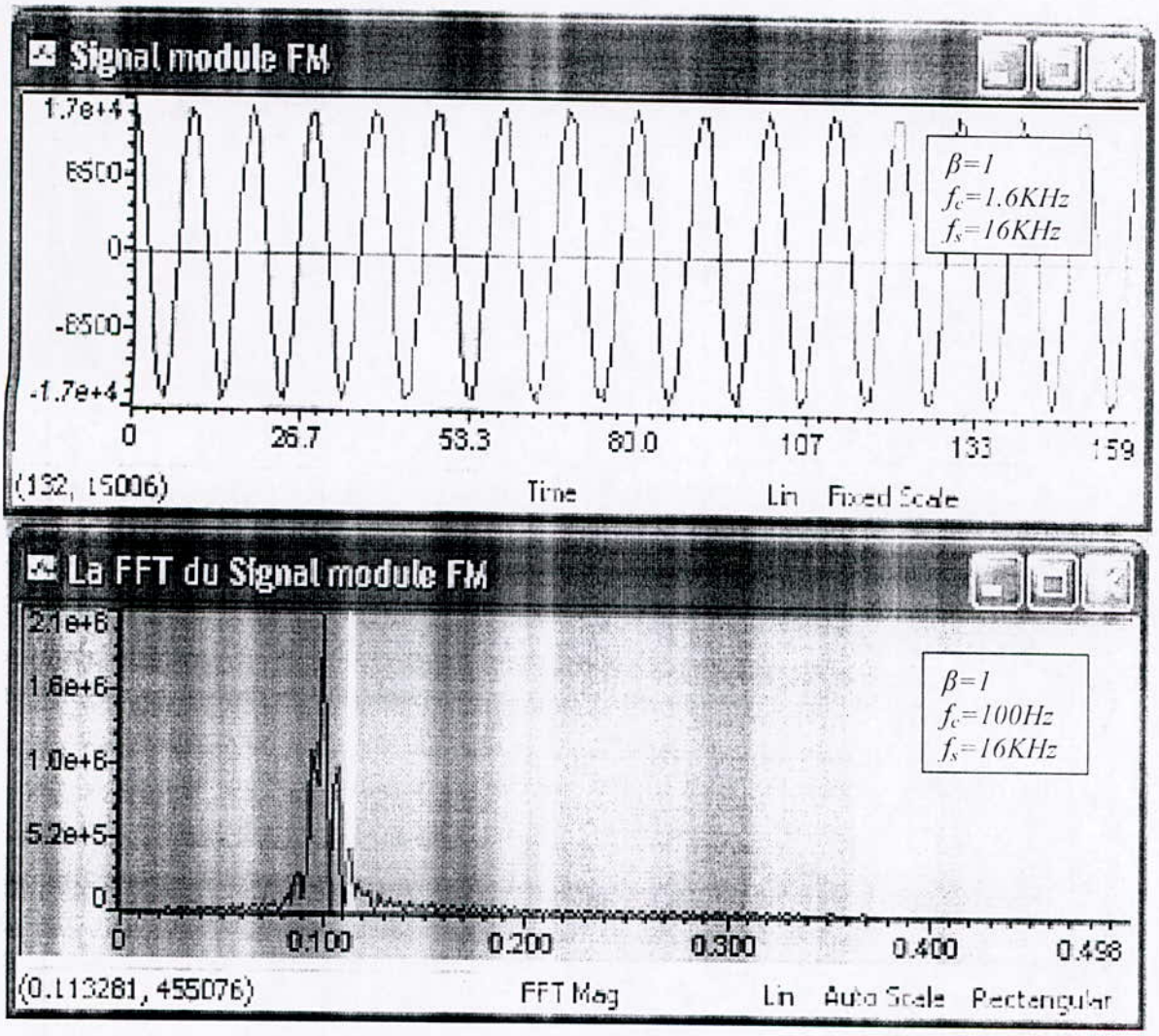
Pour β=0.2 :


Signal original
320
160
0
-160
-320
0
40.0
80.0
120
160
200
240
280
319
β=0.2
f_m=100Hz
f_s=16KHz
(160, 320)
Time
Lin Auto Scale
La FFT du Signal original
4.8e+4
3.6e+4
2.4e+4
1.2e+4
0
0
0.100
0.200
0.300
0.400
0.498
β=0.2
f_m=100Hz
f_s=16KHz
(0.00565936, 48280.9)
FFT Mag
Lin Auto Scale Rectangula
Signal module FM
1.7e+4
8500
0
-8500
-1.7e+4
0
26.7
53.3
80.0
107
133
159
β=0.2
f_c=1.6KHz
f_s=16KHz
(132, -129)
Time
Lin Fixed Scale

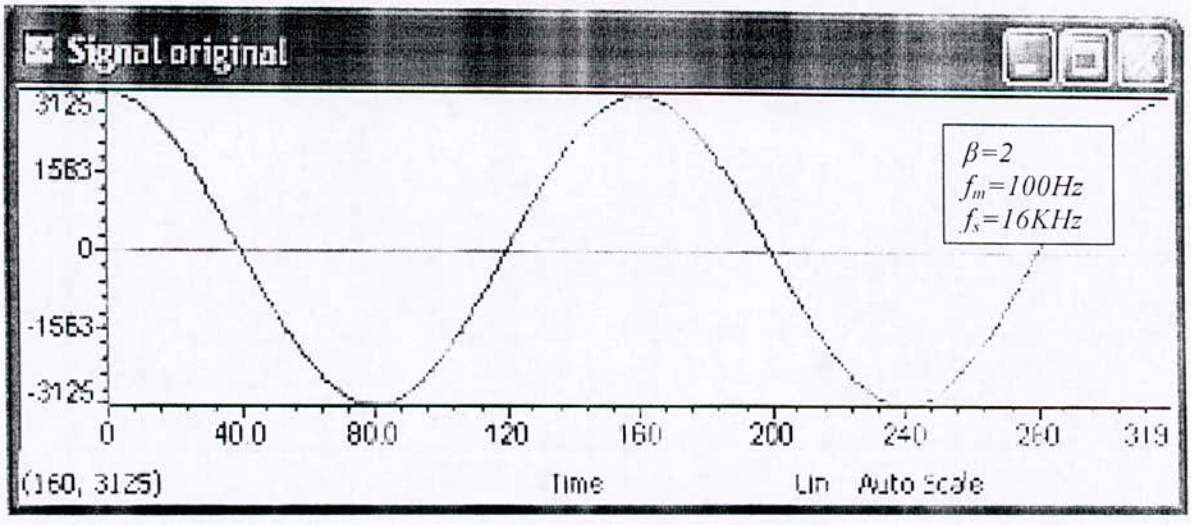

La FFT du Signal module FM
2.6e+6
1.9e+6
1.3e+6
6.4e+5
0
0
0.100
0.200
0.300
0.400
0.498
β=0.2
fc=1.6KHz
fs=16KHz
FFT Mag
Lin Auto Scale Rectangular

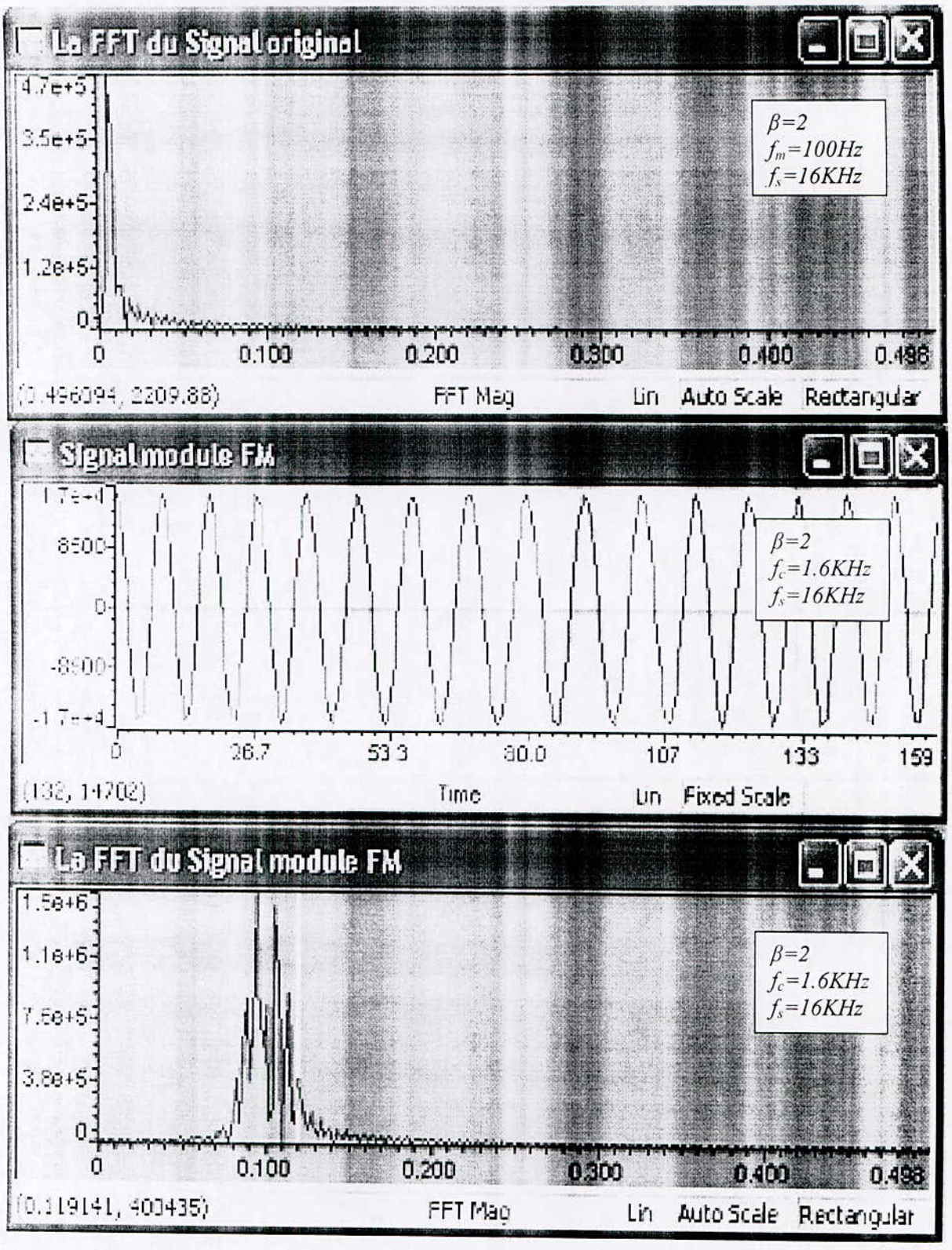
Pour $\beta=1$:


Signal original
0
40.0
80.0
120
160
200
240
280
319
β=1
fm=100Hz
fs=16KHz
Time
Lin Auto Scale
La FFT du Signal original
2.4e+5
1.8e+5
1.2e+5
5.0e+4
0
0
0.100
0.200
0.300
0.400
0.498
β=1
fm=100Hz
fs=16KHz
FFT Mag
Lin Auto Scale Rectangular

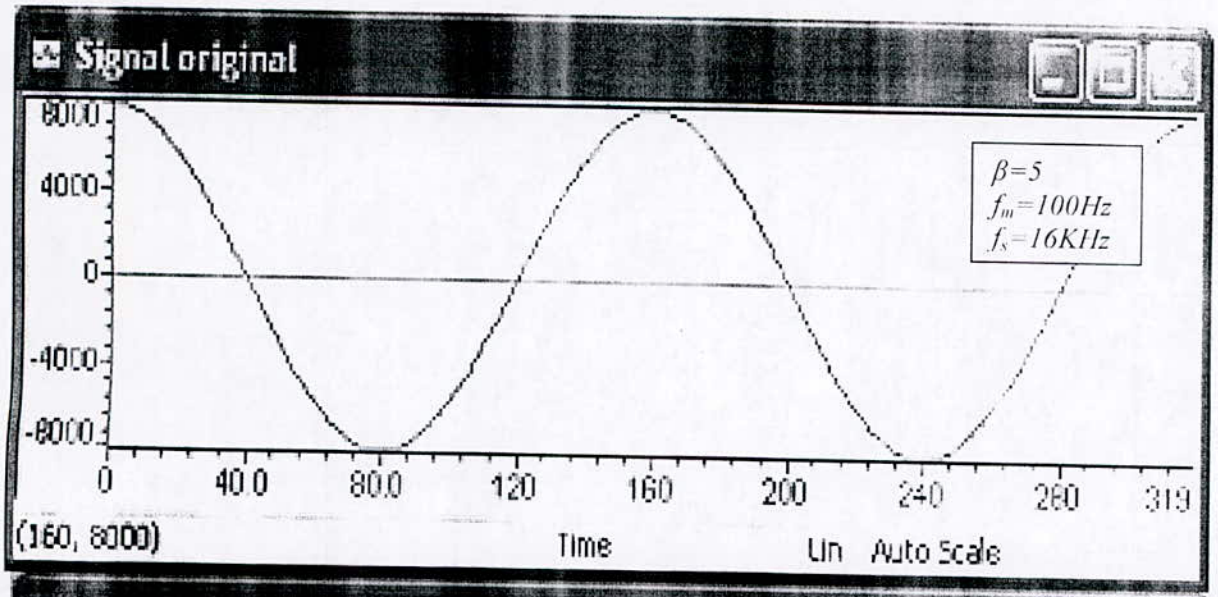

Signal module FM
1.7e+4
8500
0
-8500
-1.7e+4
0
26.7
53.3
80.0
107
133
159
(132, 15006)
Time
Lin Fixed Scale
β=1
fc=1.6KHz
fs=16KHz
La FFT du Signal module FM
2.1e+6
1.6e+6
1.0e+6
5.2e+5
0
0
0.100
0.200
0.300
0.400
0.498
(0.113281, 455076)
FFT Mag
Lin Auto Scale Rectangular
β=1
fc=100Hz
fs=16KHz

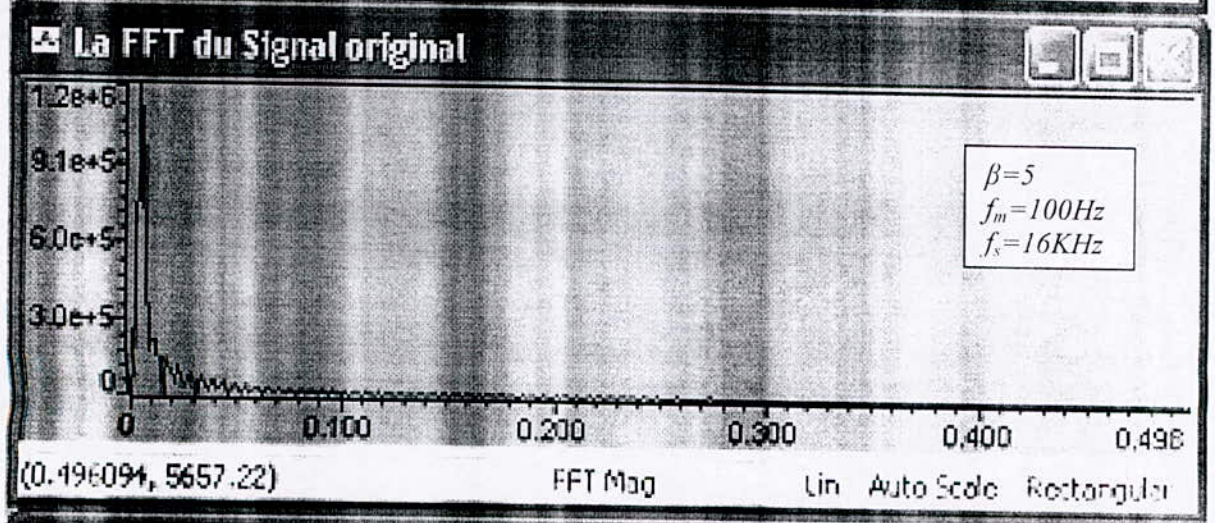
Pour $\beta=2$:

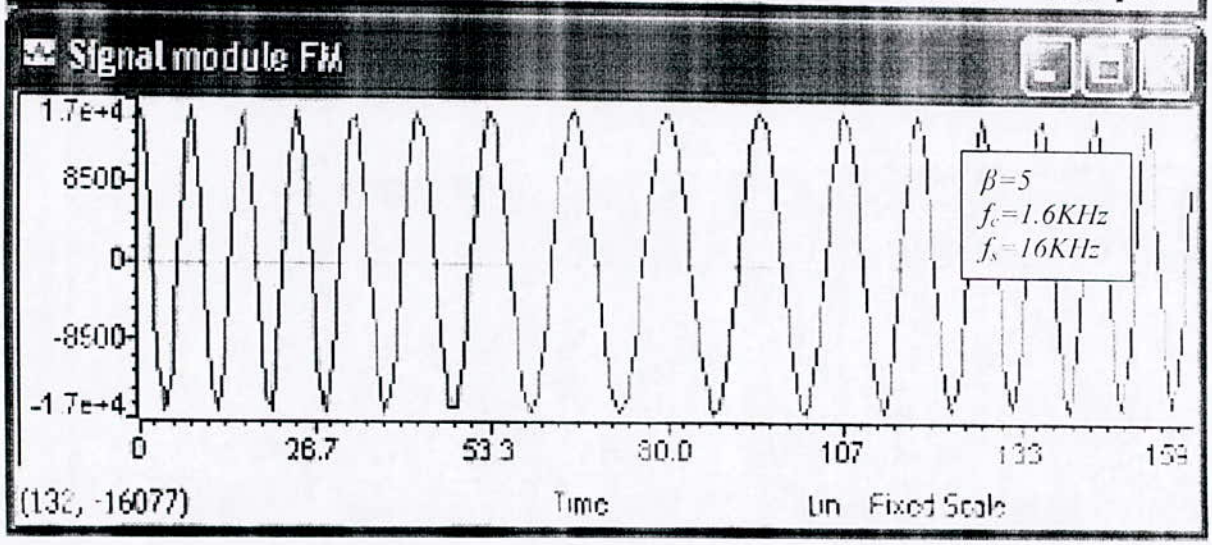

Signal original
3125
1563
0
-1563
-3125
0
40.0
80.0
120
160
200
240
280
319
(160, 3125)
Time
Lin Auto Scale
β=2
fm=100Hz
fs=16KHz

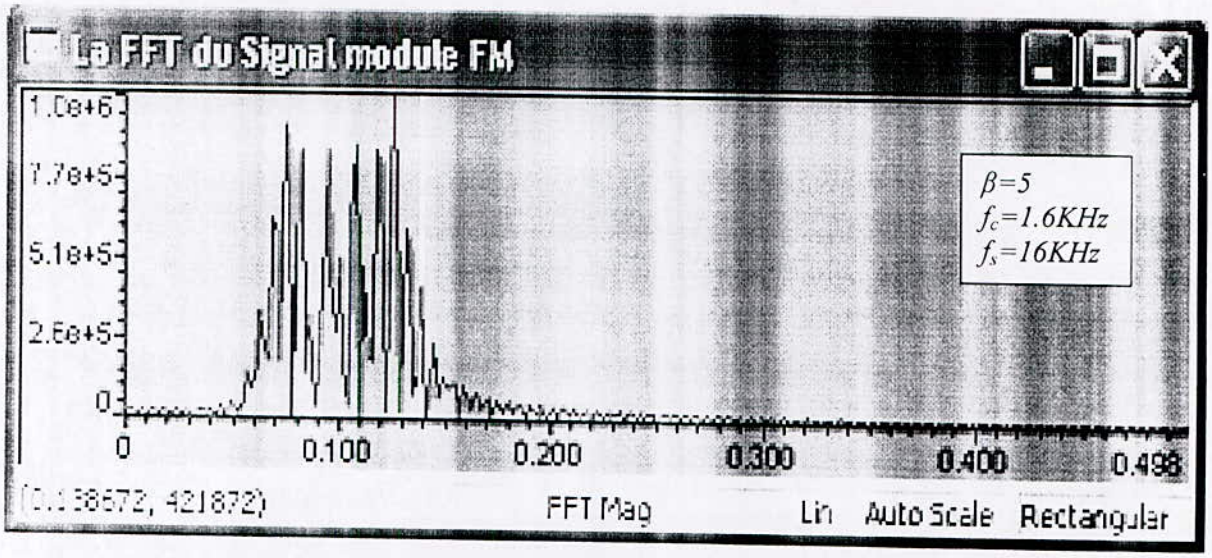

La FFT du Signal original
4.7e+5
3.5e+5
2.4e+5
1.2e+5
0
0
0.100
0.200
0.300
0.400
0.496
β=2
f_m=100Hz
f_s=16KHz
(0.496094, 2209.88)
FFT Mag
Lin
Auto Scale
Rectangular
Signal module FM
8500
0
-8500
0
26.7
53.3
80.0
107
133
159
β=2
f_c=1.6KHz
f_s=16KHz
Time
Lin
Fixed Scale
La FFT du Signal module FM
1.5e+6
1.1e+6
7.6e+5
3.8e+5
0
0
0.100
0.200
0.300
0.400
0.496
β=2
f_c=1.6KHz
f_s=16KHz
(0.119141, 400435)
FFT Mag
Lin
Auto Scale
Rectangular

Pour $B=5$:

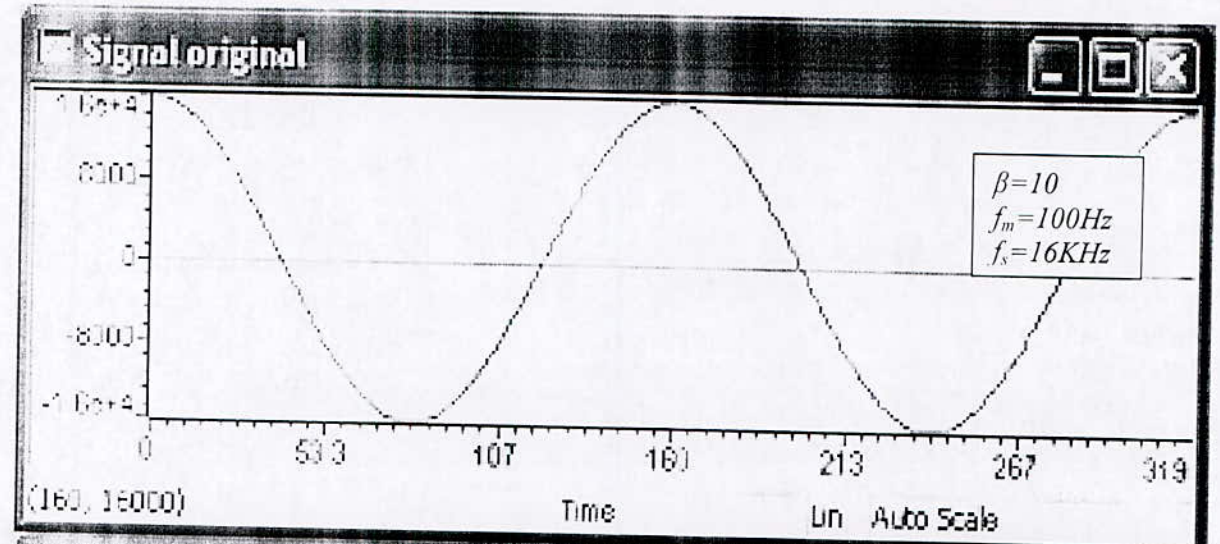

Signal original
8000
4000
0
-4000
-8000
0
40.0
80.0
120
160
200
240
280
319
β=5
fm=100Hz
fs=16KHz
(160, 8000)
Time
Lin Auto Scale

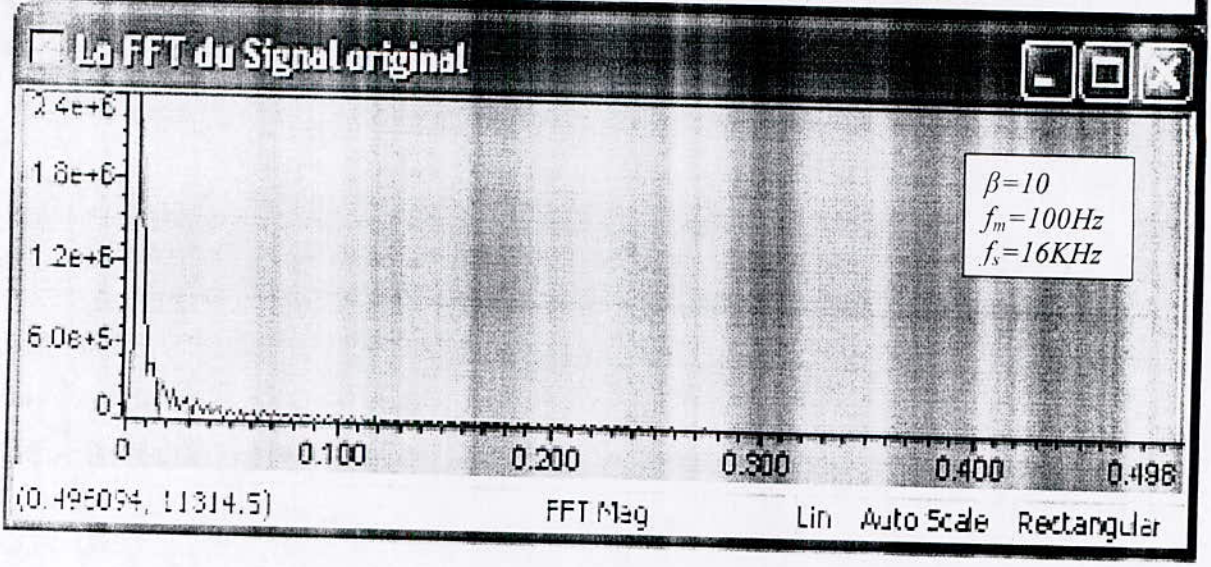

La FFT du Signal original
1.2e+6
9.1e+5
6.0e+5
3.0e+5
0
0
0.100
0.200
0.300
0.400
0.498
β=5
fm=100Hz
fs=16KHz
(0.496094, 5657.22)
FFT Mag
Lin Auto Scale Rectangular

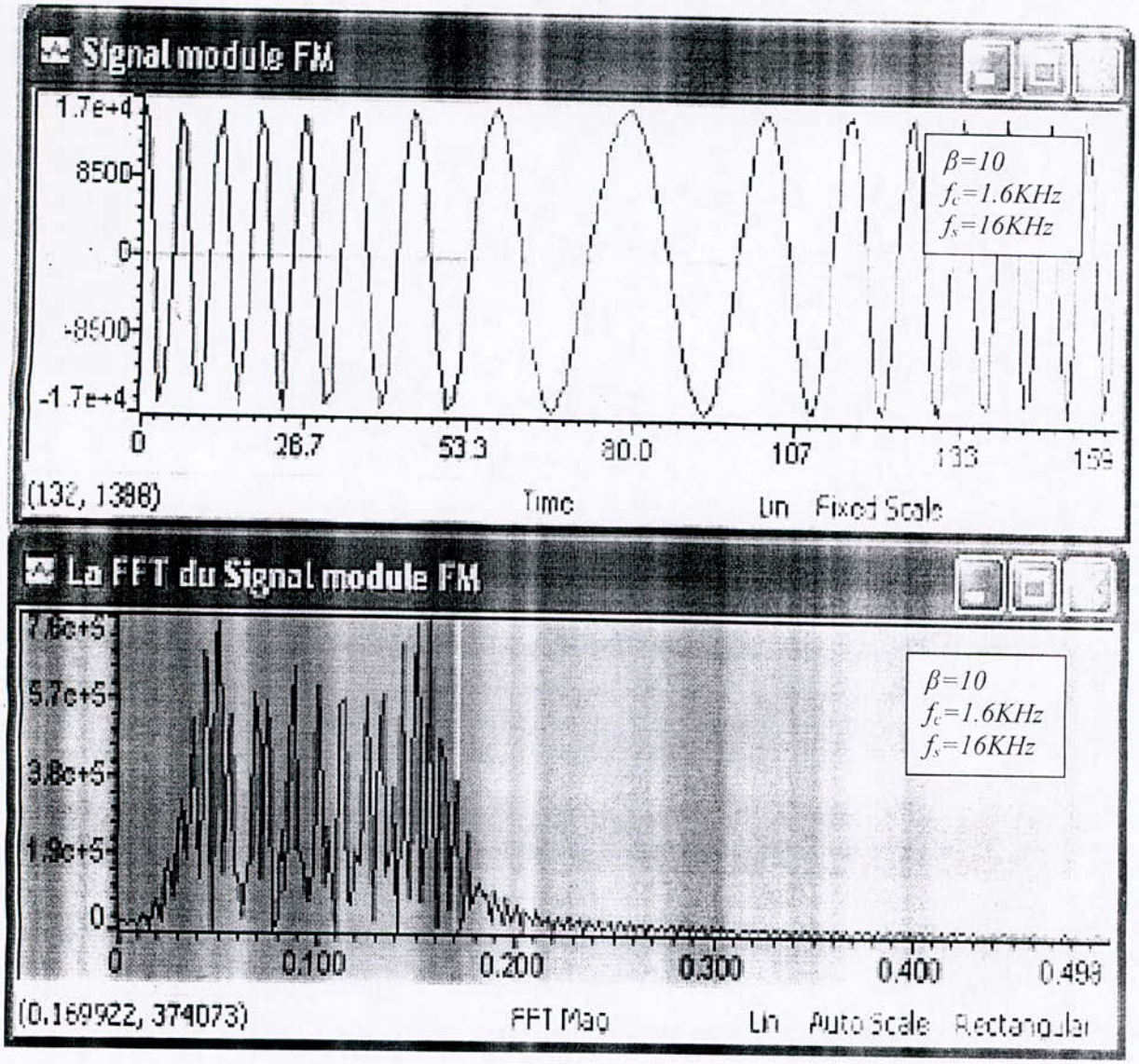

Signal module FM
1.7e+4
8500
0
-8500
-1.7e+4
0
26.7
53.3
80.0
107
133
159
β=5
fc=1.6KHz
fs=16KHz
(132, -16077)
Time
Lin Fixed Scale


La FFT du Signal module FM
1.0e+6
7.7e+5
5.1e+5
2.6e+5
0
0
0.100
0.200
0.300
0.400
0.498
β=5
fc=1.6KHz
fs=16KHz
FFT Mag
Lin
Auto Scale
Rectangular

Pour $\beta=10$:


Signal original
0
533
107
160
213
267
319
β=10
fm=100Hz
fs=16KHz
Time
Lin
Auto Scale


La FFT du Signal original
1.8e+6
1.2e+6
6.0e+5
0
0
0.100
0.200
0.300
0.400
0.498
β=10
fm=100Hz
fs=16KHz
FFT Mag
Lin
Auto Scale
Rectangular



Les résultats obtenus sont regroupés dans le tableau 3.1.

L'indice de modulation β	La bande passante du signal modulant (f_m)	La fréquence de coupure bas du signal FM	La fréquence de coupure haut du signal FM	La bande passante du signal modulé
0.2	*100Hz*	*1518.7*	*1718.752*	*200.052Hz*
1	*100Hz*	*1812.5008*	*1406.256*	*406.2448Hz*
2	*100Hz*	*1312.5008*	*1906.256*	*593.7552Hz*
5	*100Hz*	*1000*	*2218.752*	*1218.752*
10	*100Hz*	*500*	*2718.725*	*2218.725*

Tableau 3.1 : les résultat de simulation de la modulation FM sur le Code Composer

On constate dans le tableau3.1 que le spectre du signal modulé croit proportionnellement avec β.

A faible niveau ($\beta \ll 1$), la largeur de bande utile est environ égale à deux fois la fréquence du signal modulant [20], seules les raies à $|f_c|$ et $|f_c \pm f_m|$ ont des amplitude non négligeables. Ceci est conforme au résultat donnée en (3.11), avec $J_0(\beta) \approx 1$ et $J_1(\beta) \approx \beta/2$.

Dans le cas générale, les coefficient de $J_n(\beta)$ tend rapidement vers zéro, lorsque n tend vers l'infini pour satisfaire à la relation de Parseval [3]. La largeur de la bande utile reste donc bornée et peut être estimée par la règle empirique de Carson :

$$\beta_T \approx 2\,(\Delta f + fm)$$
$$= 2\,(\beta + 1)\,fm$$

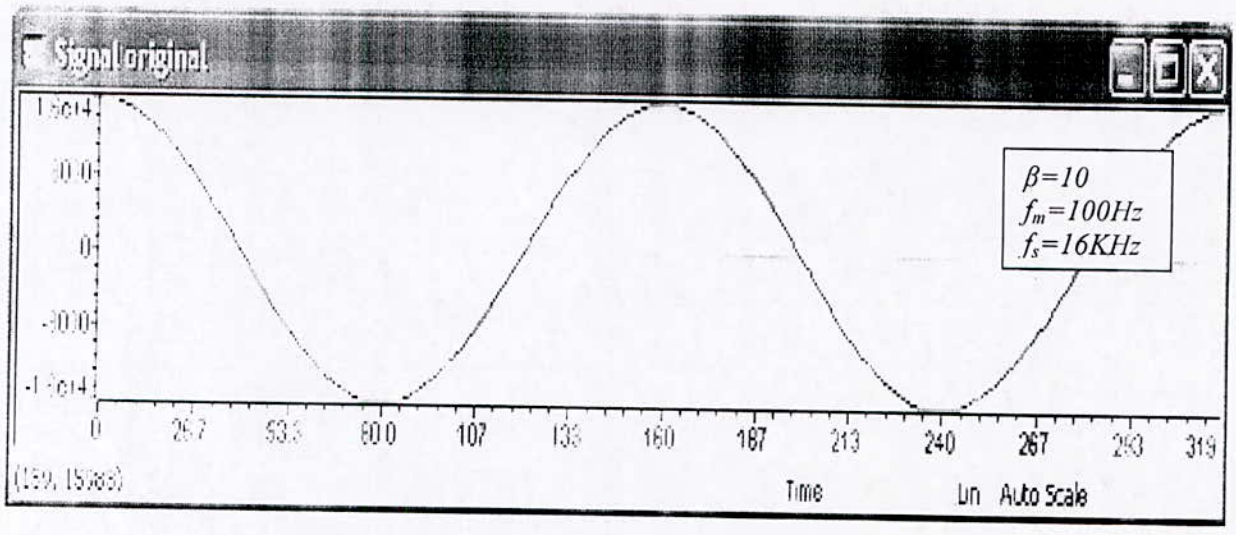
Pour une modulation sinusoïdale en fréquence, un changement de fréquence du signal modulant entraîne par conséquent non seulement un effet de dilatation (ou contraction) spectral, mais également une modification des coefficients $J_n(\beta)$. La largeur de bande utile reste dans ce cas, sensiblement constante conformément à la règle de Carson, [3].

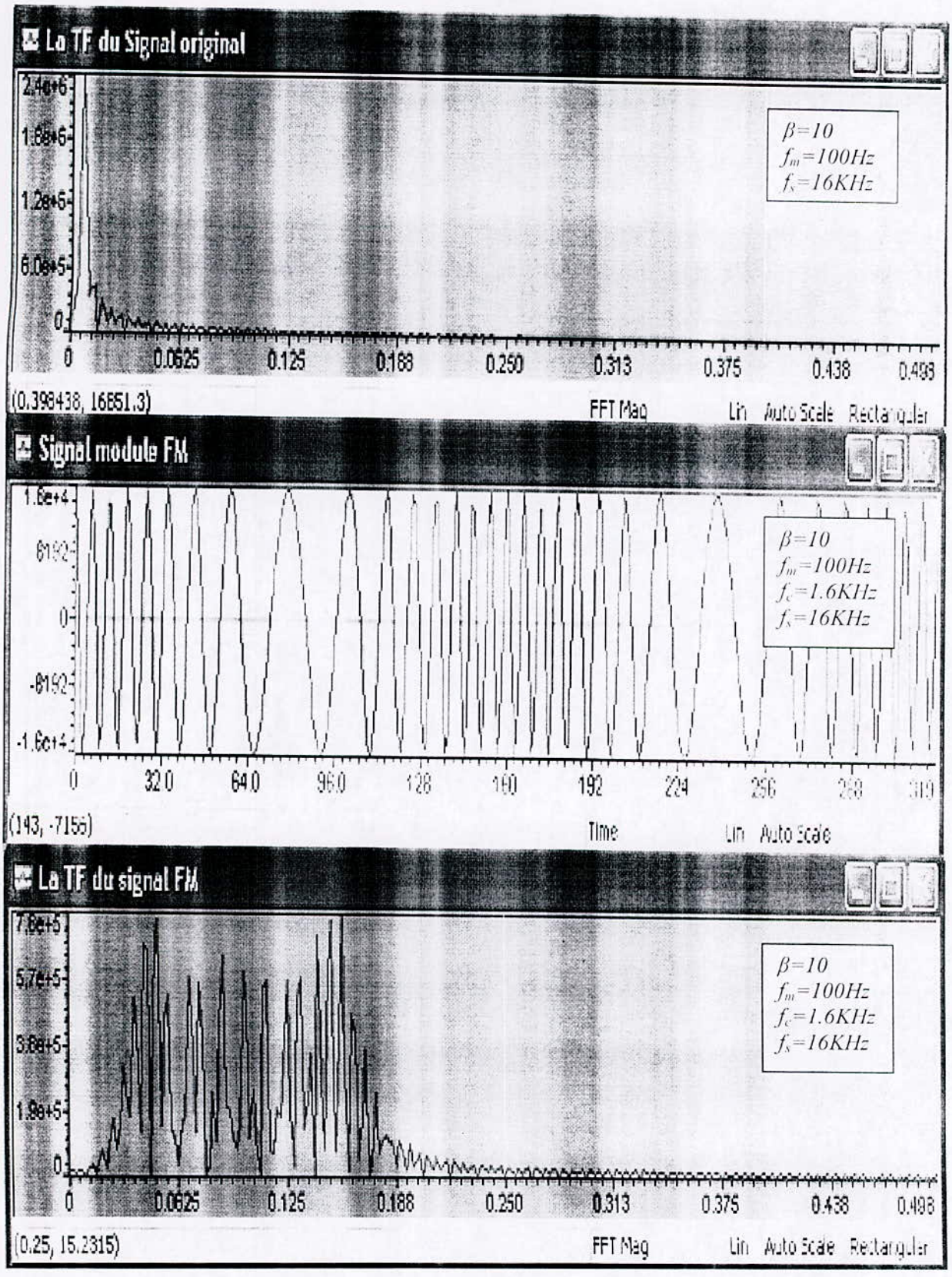
3.6.2 La démodulation

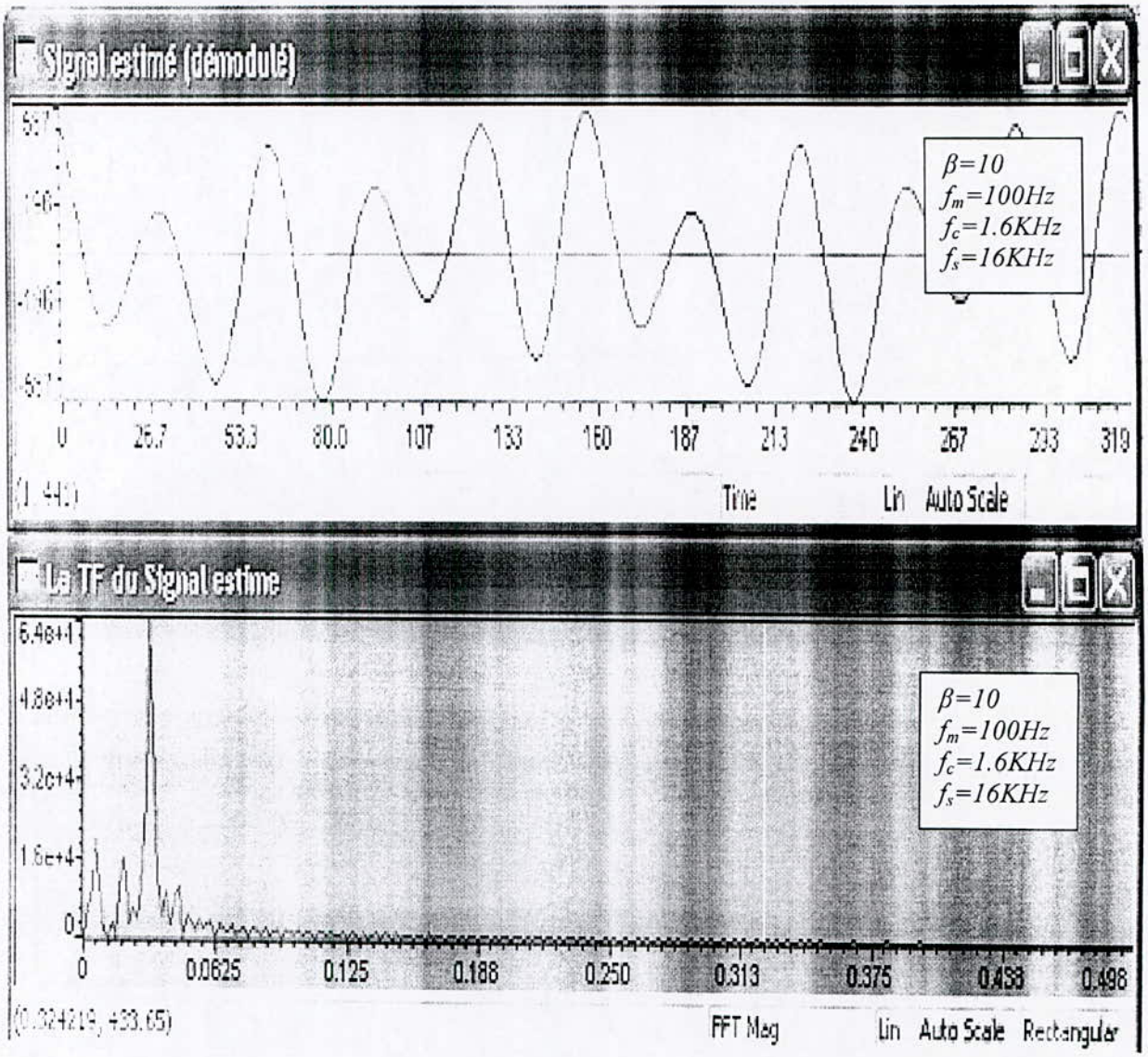
On trace les zones mémoires correspondantes aux buffers : Signal_Module_FM et Signal_Estime, en temps et en fréquence pour différents signaux d'entrés fournis par l'association d'un fichier des données au buffer Signal_Original.

De même dans de la modulation, L'échelle des ordonnées est en format Q_{14}, chaque unité est de $1/2^{14}$. L'échelle des abscisses est en $1/f_s$ pour chaque unité.

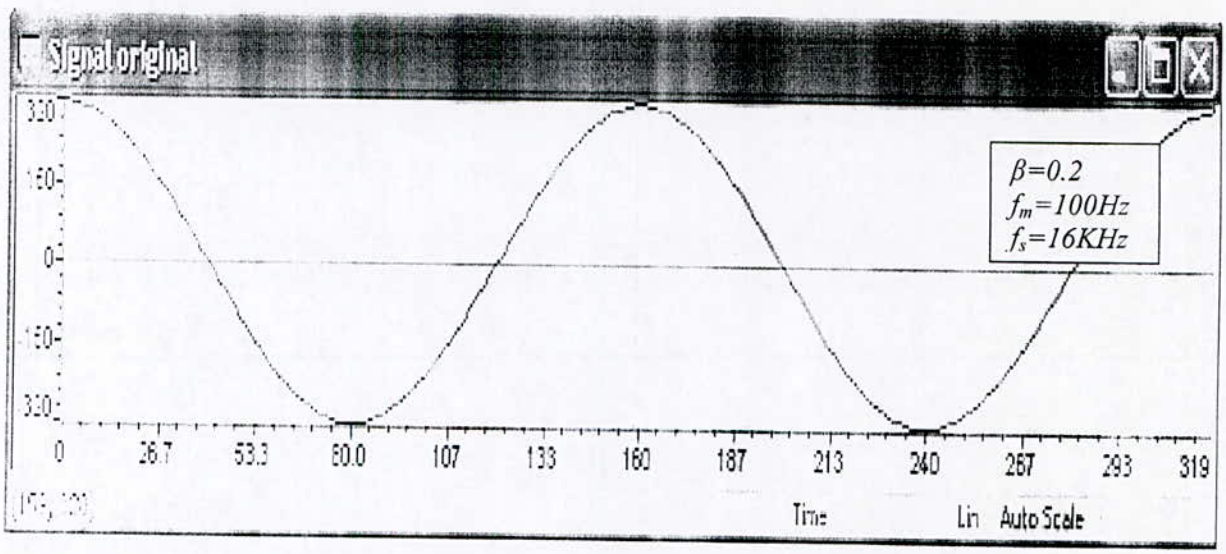
Pour un signal sinusoïdale d'indice de modulation $\beta=10$:

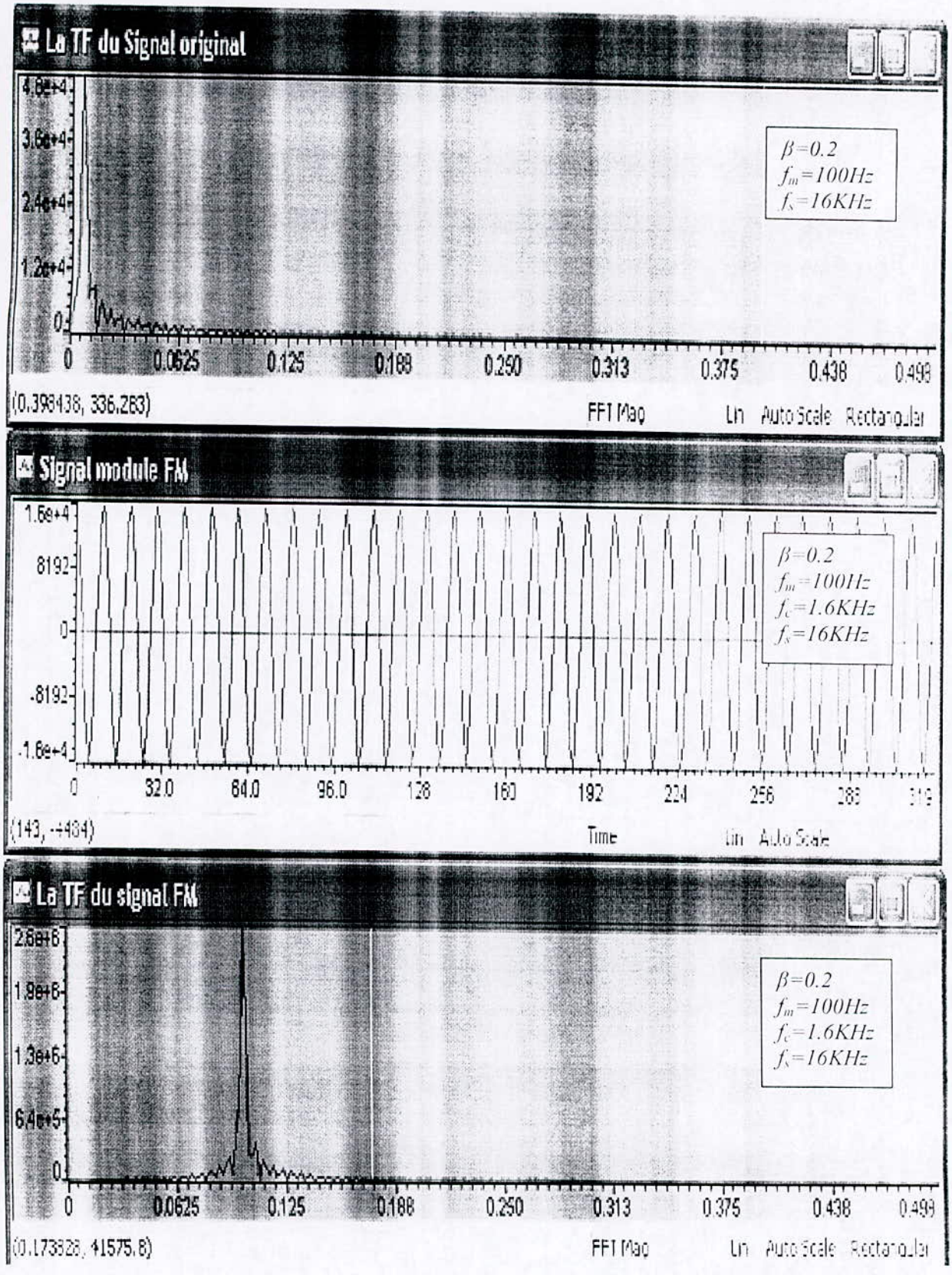


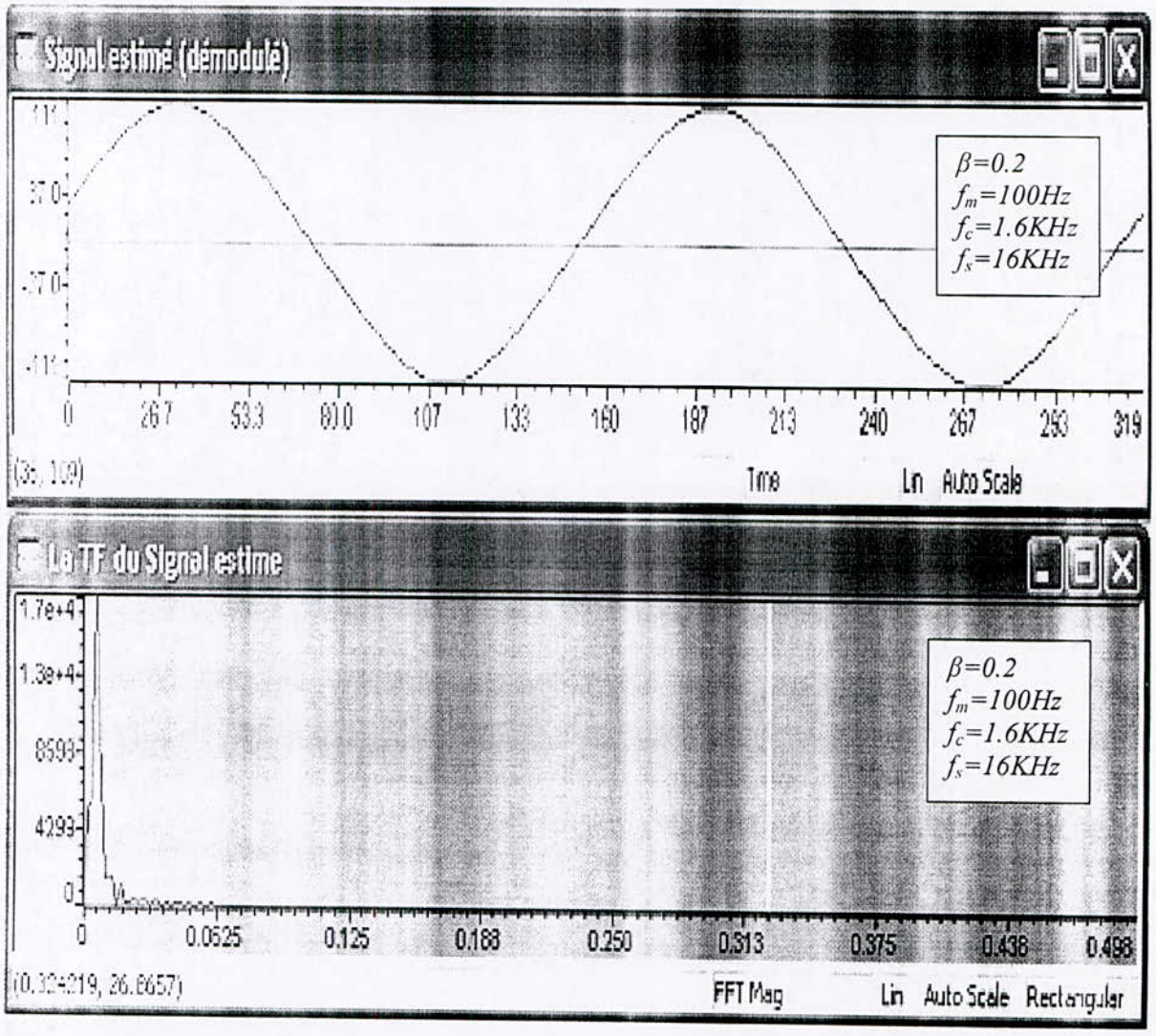

La TF du Signal original
2.4e+6
1.8e+6
1.2e+6
6.0e+5
0
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
β=10
fm=100Hz
fs=16KHz
(0.398438, 16651.3)
FFT Mag
Lin Auto Scale Rectangular
Signal module FM
1.6e+4
6192
0
-6192
-1.6e+4
0
32.0
64.0
96.0
128
160
192
224
256
288
319
β=10
fm=100Hz
fc=1.6KHz
fs=16KHz
(143, -7156)
Time
Lin Auto Scale
La TF du signal FM
7.6e+5
5.7e+5
3.8e+5
1.9e+5
0
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
β=10
fm=100Hz
fc=1.6KHz
fs=16KHz
(0.25, 15.2315)
FFT Mag
Lin Auto Scale Rectangular



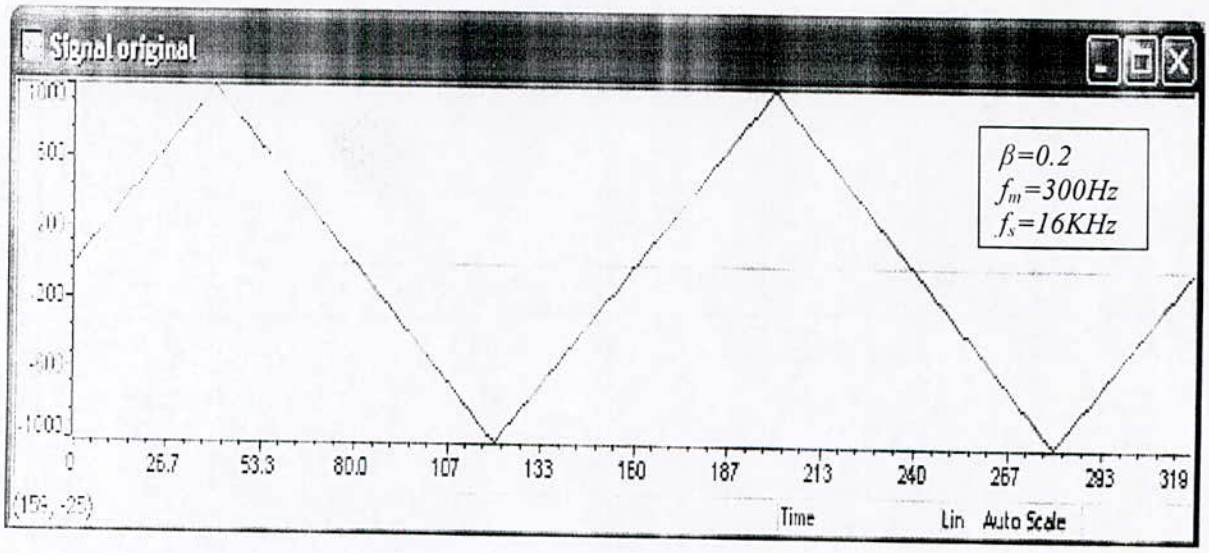
Pour un signal sinusoïdale d'indice de modulation $\beta=0.2$:

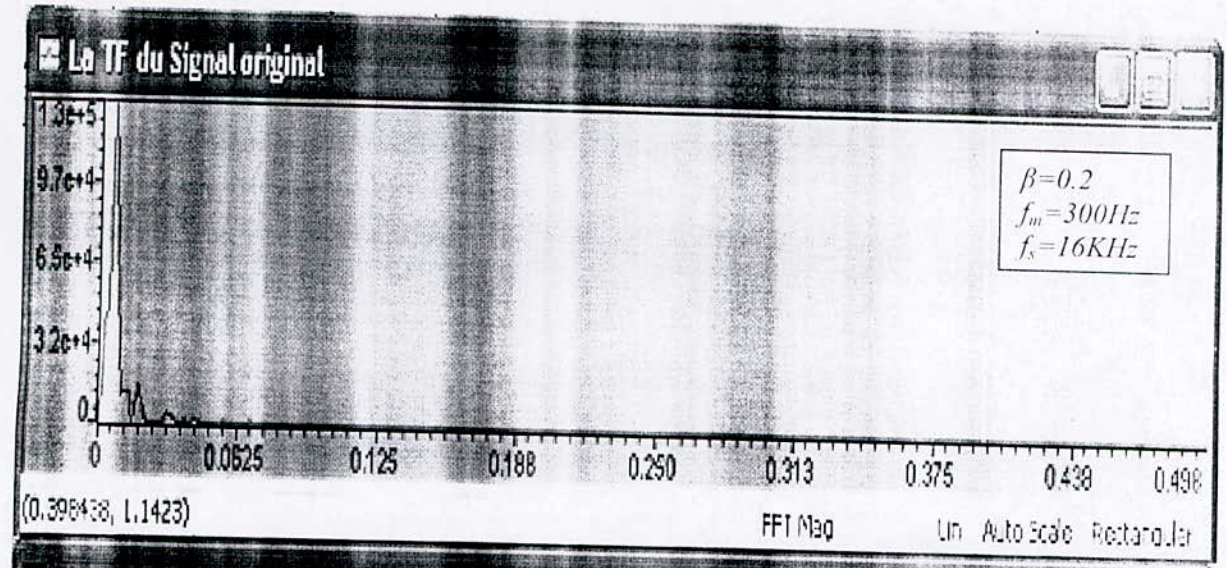


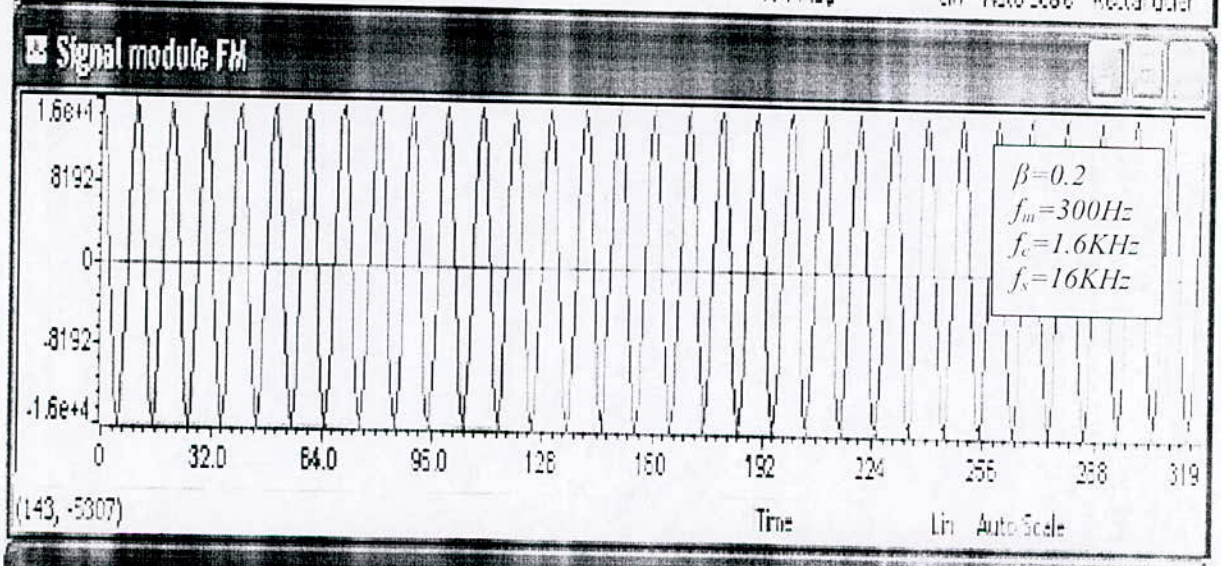

La TF du Signal original
4.8e+4
3.6e+4
2.4e+4
1.2e+4
0
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
β=0.2
fm=100Hz
fs=16KHz
(0.398438, 336.283)
FFT Mag
Lin Auto Scale Rectangular
Signal module FM
1.6e+4
8192
0
-8192
-1.6e+4
0
32.0
64.0
96.0
128
160
192
224
256
288
319
β=0.2
fm=100Hz
fc=1.6KHz
fs=16KHz
(143, -4484)
Time
Lin Auto Scale
La TF du signal FM
2.6e+6
1.9e+6
1.3e+6
6.4e+5
0
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
β=0.2
fm=100Hz
fc=1.6KHz
fs=16KHz
(0.173828, 41575.8)
FFT Mag
Lin Auto Scale Rectangular

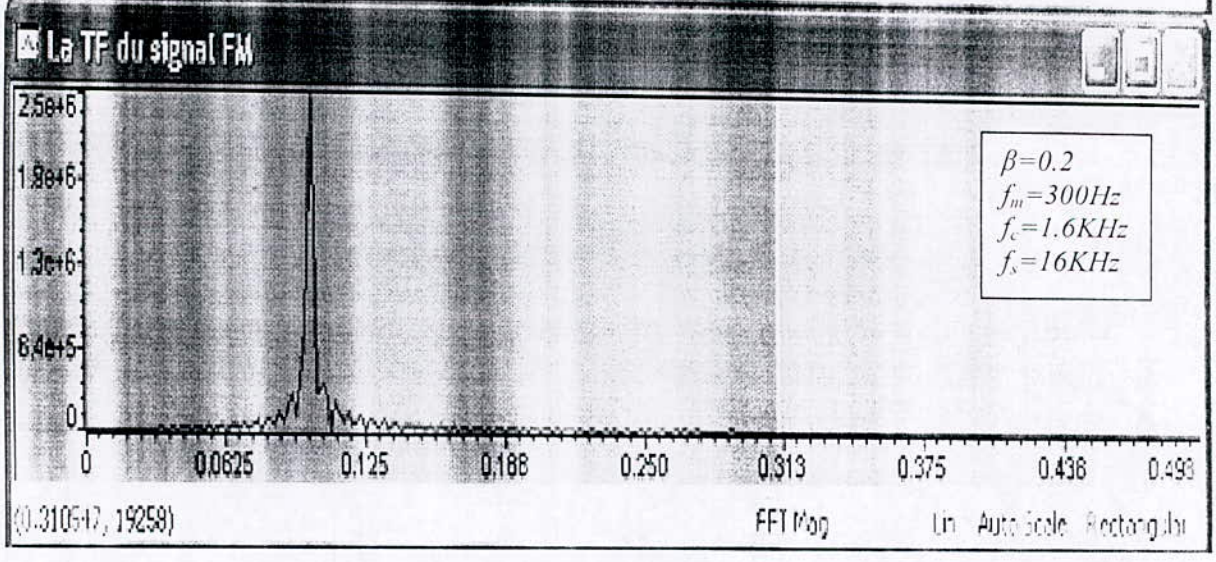


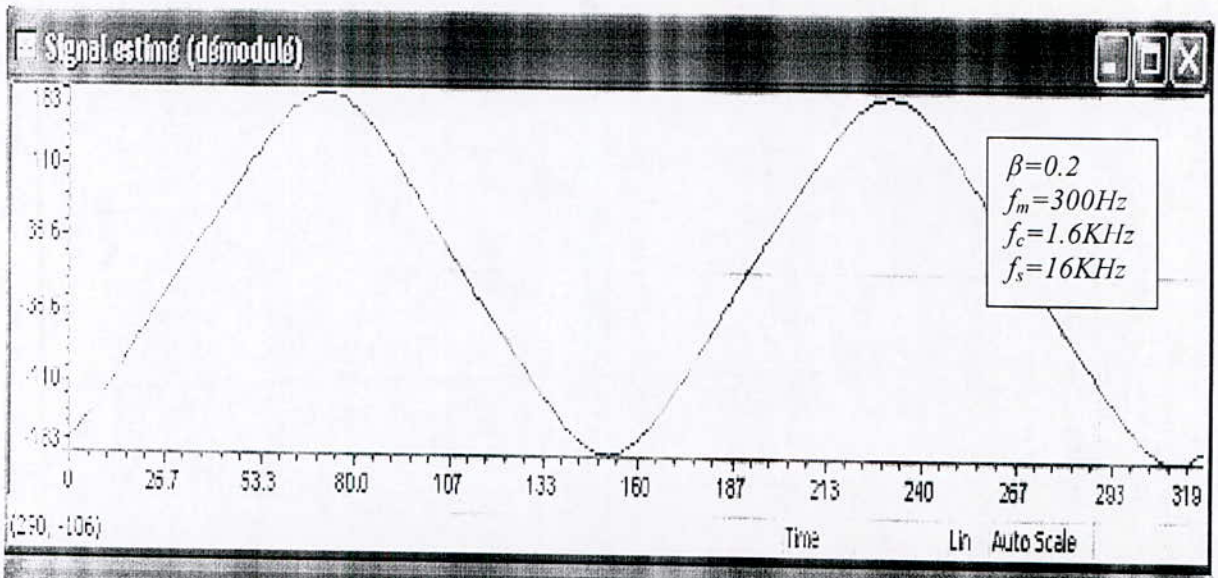
Pour un signal triangulaire $\beta=0.2$:

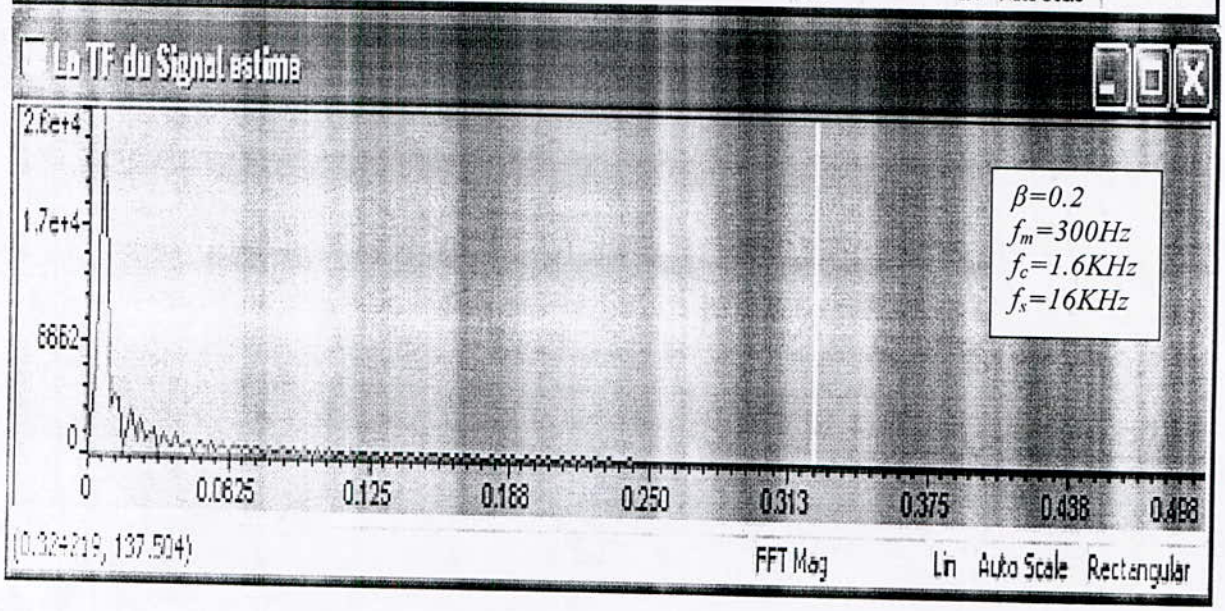



La TF du Signal original
1.3e+5
9.7e+4
6.5e+4
3.2e+4
0
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
β=0.2
fm=300Hz
fs=16KHz
(0.396438, 1.1423)
FFT Mag
Lin Auto Scale Rectangular

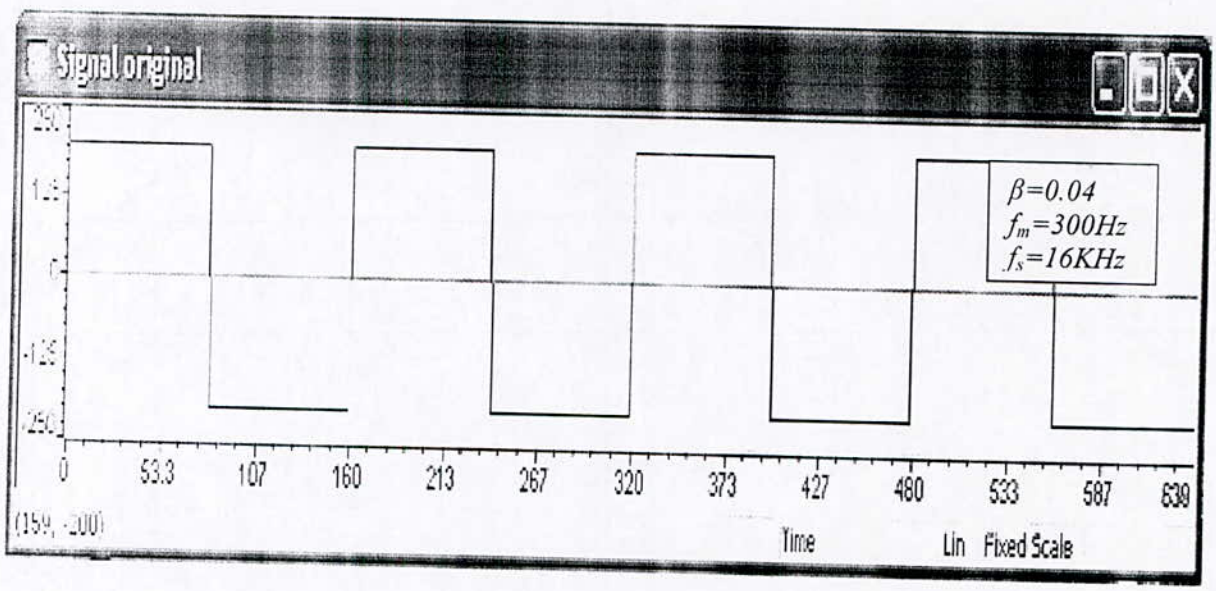

Signal module FM
1.6e+1
8192
0
-8192
-1.6e+4
0
32.0
64.0
96.0
128
160
192
224
256
288
319
β=0.2
fm=300Hz
fc=1.6KHz
fs=16KHz
(143, -5307)
Time
Lin Auto Scale

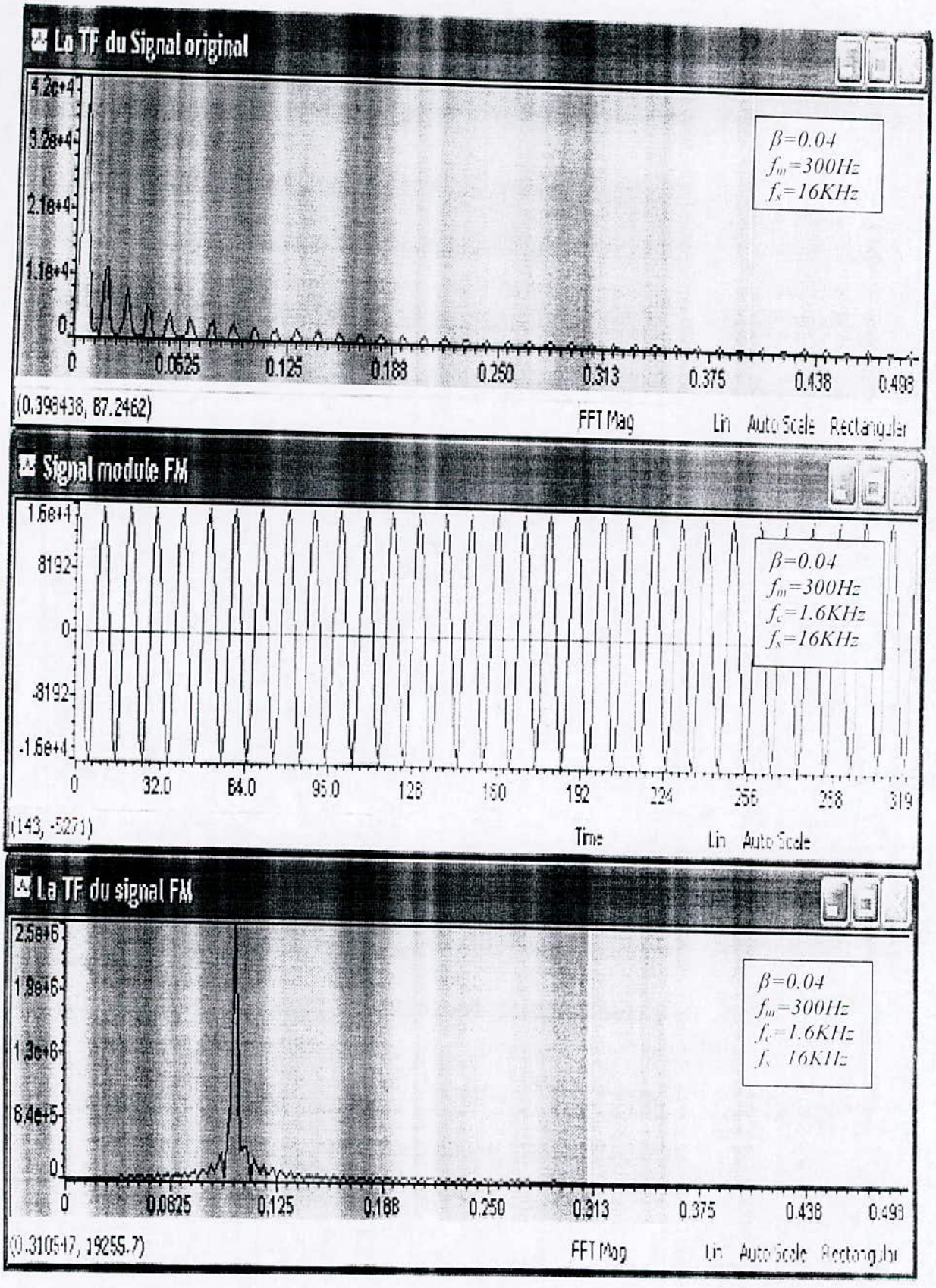

La TF du signal FM
2.5e+6
1.9e+6
1.3e+6
6.4e+5
0
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.436
0.498
β=0.2
fm=300Hz
fc=1.6KHz
fs=16KHz
(0.310547, 19258)
FFT Mag
Lin Auto Scale Rectangular

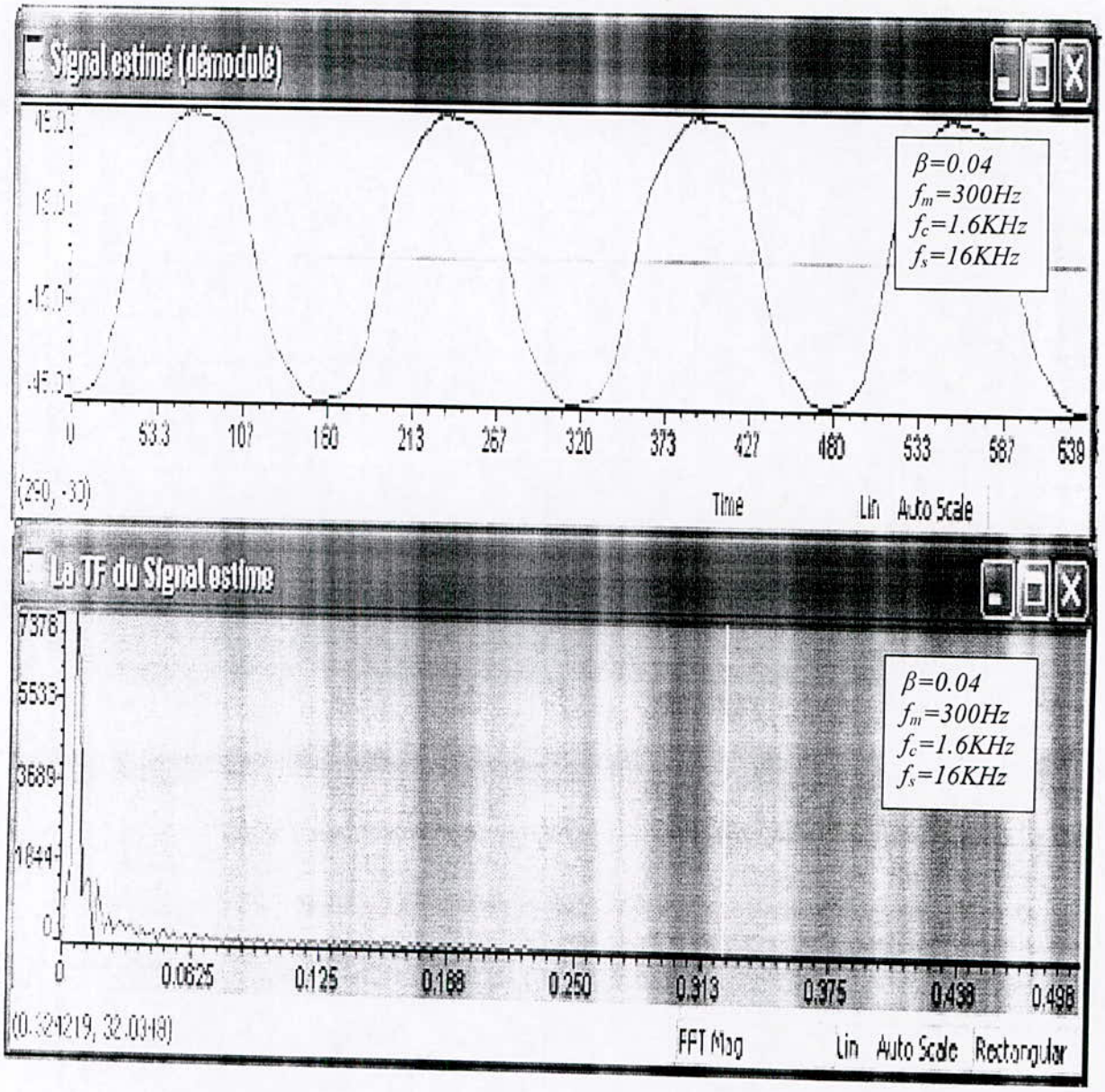




Pour un signal carré $\beta=0.04$:




La TF du Signal original
β=0.04
fm=300Hz
fs=16KHz
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
(0.398438, 87.2462)
FFT Mag
Lin Auto Scale Rectangular
Signal module FM
β=0.04
fm=300Hz
fc=1.6KHz
fs=16KHz
8192
-8192
-1.6e+4
0
32.0
64.0
96.0
128
160
192
224
256
288
319
(143, -5271)
Time
Lin Auto Scale
La TF du signal FM
β=0.04
fm=300Hz
fc=1.6KHz
fs 16KHz
0
0.0625
0.125
0.188
0.250
0.313
0.375
0.438
0.498
(0.310647, 19255.7)
FFT Mag
Lin Auto Scale Rectangular



Pour des valeurs de β élevé, par exemple $\beta=10$, le signal original estimé est équivalent à un signal modulé en AM à cause du spectre du signal modulé FM qui est très large, et par conséquence une partie des rais non désiré est passé dans le filtre passe bas, donc ont été réalisé une conversion FM /AM.

Dans le cas des faibles indices de modulation, $\beta=0.2$ par exemple, et pour les deux signaux : triangulaire, et carré ; l'estimation est bonne avec une certaine distorsion causée par l'erreur de quantification d'amplitude (précision finie) et l'approximation utilisé par le modèle linéaire du PLL. Un déphasage entre le signal message et le signal estimé introduit par le filtre numérique IIR passe bas.

3.7 Evaluation de la simulation

La modulation FM représente une bonne discrimination contre le bruit additif et le phénomène d'interférence que la modulation AM, car le niveau du signal modulé ne dépend pas de niveau du signal modulant, au prix d'un élargissement de la largeur de bande occupée par le signal modulé.

La fréquence de la porteuse générée par le DSP est faible, mais parfois la génération des fréquences pures étant utiles dans de nombreuses applications. Les porteuses de fréquence élevée sont générées, soit à l'extérieur de manière analogique, ou soit par l'utilisation d'un multiplicateur de fréquence à la sortie du DSP et en conversant le signal FM à bande étroite de porteuse faible, en un signal FM à large bande de porteuse élevée.

Pour la démodulation FM, le circuit PLL est préférable à utiliser dans des conditions difficiles de transmission, mais dans l'implantation sur le DSP on est limité par le type du DSP cible et la contrainte en temps.

Un détecteur d'enveloppe peut être utilisé en amont du convertisseur FM/AM réalisé par le circuit PLL dans le cas d'une modulation à haut niveau ($\beta ?\ 1$) pour reconstituer le signal message.

3.8 Réalisation pratique

Pour le teste en temps réel sur le DSK VC5402, on utilise deux buffers aic_in et aic_out pour l'acquisition des échantillons proviens de l'interface audio et transmettre les échantillons traités a l'interface speaker respectivement.

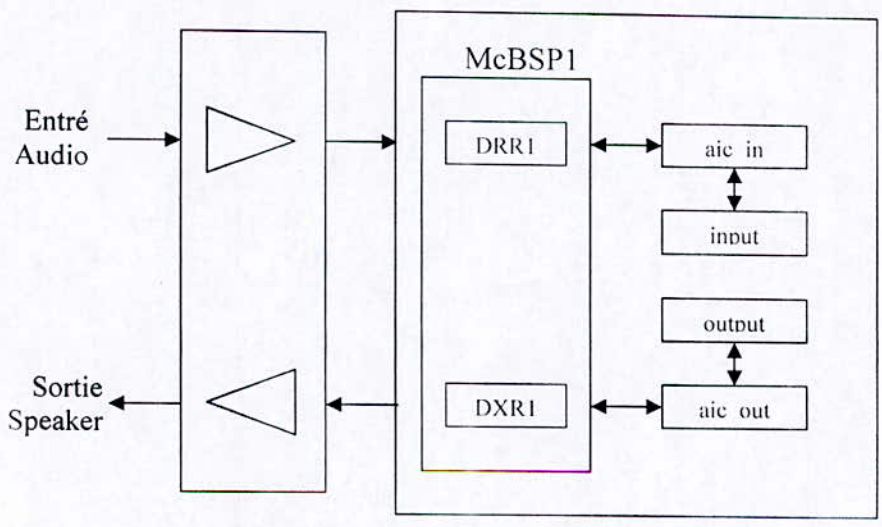


Figure 2.6 : réception et émission des donnée

Pour la restitution des données d'entrées et de sorties on utilise les deux buffers input et output comme l'indique la Figure 3.13.

Comme on a été pris par le temps on s'est arrivé à tester seulement le programme de modulation en temps réel sur la carte de DSK VC5402 donné dans l'annexe. Le programme de démodulation exige une adaptation du signal FM aux caractéristiques du démodulateur utilisé.

Conclusion générale

Dans le travaille réalisé, nous avons montré les principes de base, le fonctionnement, la constitution et la technique utilisée pour l'implémentation du modulateur démodulateur FM sur le DSP C5402. Nous avons utilisé la méthode de lecture en table pour rendre les programmes moins chargés et plus efficaces dans les deux phases de modulation et de démodulation. Une meilleure performance est réalisée par le choix du circuit PLL utilisé au terme de temps d'exécution et de la simplicité à implanter. Il est conçu pour répondre parfaitement aux caractéristiques du DSP C5402.

Nous avons constaté à partir des résultats obtenus de la simulation, qu'il est nécessaire de faire un compromis entre les caractéristiques de la modulation FM d'une part et Les caractéristiques du DSP C5402 et les périphériques de DSK d'autre part. Ce compromis est réalisé par le choix du modèle des blocs et la méthode utilisée dans l'implémentation, tels que :

- On ne peut pas généré des fréquences des porteuses élevées avec les convertisseurs A/D et D/A utilisés dans le DSK C5402. Ceci ne permet pas l'utilisation des paramètres réels de la modulation/démodulation FM que dans des conditions plus appropriées sur le signal message et l'utilisation des circuits analogiques.
- Il faut adopter les calculs au type de DSP C5402, qui est en format fixe, par élimination des calculs en format flottant dans le choix de la méthode utilisée.
- Les valeurs des calculs intermédiaires sont limités par la capacité de l'accumulateurs A (ou B), donc on doit le tenir compte dans l'implémentation du filtre IIR par l'utilisation des facteurs d'échelle entre les blocs d'ordre deux constituants le filtre.

L'implémentation d'un modulateur démodulateur FM sur le DSP C5402 devient intéressante, lorsque nous intéressons à la génération d'une fréquence de porteuse pure et nous utilisons des circuits analogiques complémentaires.

Ce travaille peut être complété par l'utilisation des circuit analogiques tels que des multiplieurs de fréquence à la sortie du modulateur et un bloc FI par exemple à l'entrée du démodulateur. Il peut être également exploité dans les travaux pratiques de système de communication.

ANNEXE A

Le programme de simulation du modulateur /démodulateur FM

```
/*************************************************************************/
/*                FM_MODEM.C                                             */
/*               ==============                                          */
/*                                                                       */
/*  Modulateur/démodulateur FM                                           */
/*  les paramètres:                                                      */
/*   k =2 pi1024  La constante de modulation                             */
/*   fs= 16KHz   La fréquence d'échantillonnage                          */
/*   fc= 1.6KHz    La fréquence de la porteuse                           */
/*************************************************************************/

#include <type.h>
#include "parametres.h"

/*************************************************************************/
/*  Function Prototypes:                                                 */
/*                                                                       */
/*   LP_IIR_Filter function:                                             */
/*      Le filtre passe bas                                              */
/*                                                                       */
/*   Wait function:                                                      */
/*      attend l'echantillon suivant                                     */
/*      lire l'echantillon d'entré et emetre l'echantillon traité        */
/*                                                                       */
/*************************************************************************/

void Wait()                 ;
static void LP_iir_Filtre(u16,bool )   ;

/*************************************************************************/
/* Variables Globales                                                    */
/*************************************************************************/

s16   Signal_Original[SIZE_BUFF]        ;   /* Le signal modulant          */
s16   Signal_Module_FM[SIZE_BUFF]       ;   /* le signal modulé FM         */
s16   Signal_Estime[SIZE_BUFF]          ;   /* Le signal demodulé          */
s16   Tmp0                              ;
s16   Sortie_VCO[SIZE_BUFF]             ;   /* Le signal de sortie de VCO  */
s16   Sortie_Multiplieur[SIZE_BUFF]     ;   /* Le signal de sortie du multiplieur */
s16   w_1[Ns*2]                         ;   /* Le buffer du filtre IIR     */
s16   coef[Ns*5]=
 { /*i est l'index du bloc de filtre d'ordre 2*/
  /*les coef du filtre sont organisés comme suit:
```

```
 A[i][1],A[i][2],B[i][2],B[i][0],B[i][1]*/
  -25962,10340 ,1593  ,1593  ,-819  ,
  -27956,12278 ,2952  ,2952  ,-5038 ,
   -30641,14937 ,4850  ,4850  ,-8910   };

/*****************************************************************************/
   MAIN
/*****************************************************************************/

void main()
{
 s32 ideltamod,deltamod,ideltademod,deltademod;
 u32 i;
u16 j;
 bool ctrfiltre;

 j=0;
 ideltamod=ideltademod=0;
 ctrfiltre=True;

for(i=0;i<Ns*2;i++)                  /* Initialisation du buffer  */
  { w_1[i]=0;                        /* du filtre IIR             */
  }

   while(True)
   {
   Wait();

    // Le Modulateur
    i=ideltamod;
    i=i%(QuatriemeQuadrant);

 if(i<=(PremierQaudrant))                    // La lecture en table sur
    {                                        // le premier quadrant
    *(Signal_Module_FM+j)=*(cos_tab+i);      // pour le signal modulé
    }
 else if(i<(DeuxiemeQuadrant))
    {
    i=PremierQaudrant-(i%(PremierQaudrant));
    *(Signal_Module_FM+j)=-*(cos_tab+i);
    }
 else if(i<=(TroisiemeQuadrant))
    {
    i=i%(DeuxiemeQuadrant);
    *(Signal_Module_FM+j)=-*(cos_tab+i);
    }
 else
    {
    i=PremierQaudrant-(i%(TroisiemeQuadrant));
    *(Signal_Module_FM+j)=*(cos_tab+i);
    }

    deltamod=(fc+Signal_Original[j]/16)%(QuatriemeQuadrant);    // Calcul de l'index de lecture
    ideltamod=ideltamod+deltamod;                               // en table pour le signal modulé
    ideltamod=ideltamod%(QuatriemeQuadrant);
```

```
// Le bloc VCO
i=12000+ideltademod;
i=i%(QuatriemeQuadrant);

if(i<=(PremierQaudrant))                    // La lecture en table sur
  {                                         // le premier quadrant
  *(Sortie_VCO+j)=*(cos_tab+i);             // pour le signal VCO
  }
else if(i<(DeuxiemeQuadrant))
  {
  i=PremierQaudrant-(i%(PremierQaudrant));
  *(Sortie_VCO+j)=-*(cos_tab+i);
  }
else if(i<=(TroisiemeQuadrant))
  {
  i=i%(DeuxiemeQuadrant);
  *(Sortie_VCO+j)=-*(cos_tab+i);
  }
else
  {
  i=PremierQaudrant-(i%(TroisiemeQuadrant));
  *(Sortie_VCO+j)=*(cos_tab+i);
  }

  // Le miltiplieur
  tmp1=(s32)(Signal_Module_FM[j]*Sortie_VCO[j]);
  Sortie_Multiplieur[j]=(s16)(tmp1>>15);

  // Le filtre IIR
 LP_iir_Filtre(j, ctrfiltre);

  Signal_Estime[j]=Tmp0;

  deltademod=(fc+Tmp0/16)%(QuatriemeQuadrant);   // Calcul de l'index de lecture
  ideltademod=ideltademod+deltademod;            // en table pour le signal modulé
  ideltademod=ideltademod%(QuatriemeQuadrant);

  j=(j+1)%SIZE_BUFF;
  ctrfiltre=False;

  }
}

void LP_iir_Filtre(u16 k,bool ctr )
{
 u16 i,j,l,m,h;
 s32 w_0; s16 temp;
 m=Ns*5;
 h=Ns*2;
 j=0;
 if(ctr==True)
 {l=0;}

 w_0=(s32)Sortie_Multiplieur[k]*5500;
```

```
for(i=0;i<Ns;i++)
{
   w_0=_smas(w_0,*(w_1+l),*(coef+j));j++;l=(l+Ns)%h;
   w_0=_smas(w_0,*(w_1+l),*(coef+j));j++;
   temp=*(w_1+l);
   *(w_1+l)=w_0>>15;
   w_0=_lsmpy(temp,*(coef+j));j++;
   w_0=_smac(w_0,*(w_1+l),*(coef+j));j++;l=(l+Ns)%h;
   w_0=_smac(w_0,*(w_1+l),*(coef+j));j=(j+1)%m;l=(l+1)%h;
}

Tmp0=(s16) (w_0>>15);

 return;
}

void Wait()
{
        /* changement des données */
  return;
}
```

ANNEXE B

Le programme de la modulation pour le teste sur le DSK

```
/*************************************************************************/
/*                DSK_FM_MOD.C                                           */
/*               ==============                                          */
/*                                                                       */
/*  Modulateur/démodulateur FM                                           */
/*  les paramètres:                                                      */
/*   k =2 pi1024  La constante de modulation                             */
/*   fs= 16KHz   La fréquence d'échantillonnage                          */
/*   fc= 1.6KHz    La fréquence de la porteuse                           */
/*************************************************************************/

//les fonctions des périphériques fournies par Texas Instruments
#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>

// les paramètres du modulateur/démodulateur FM
#include "parametres.h"

/*************************************************************************/
/*  Function Prototypes:                                                 */
/*                                                                       */
/*  Init function:                                                       */
/*      Ouvrir le codec                                                  */
/*      determiner les parametre du codec                                */
/*                                                                       */
/*  Wait function:                                                       */
/*      Attend l'échantillon suivant                                     */
/*      Lire l'échantillon d'entrée et émettre l'échantillon traité      */
/*                                                                       */
/*************************************************************************/
void Init() ;
void Wait() ;

/*************************************************************************/
      Variables Globales
/*************************************************************************/

HANDLE  hHandset          ;
u16    length=SIZE_COS_TABLE ;
s16    aic_in,*input       ; //aic_in léchantillon d'entré
```

```
s16    aic_out,*output       ; //aic_out l'echantillon de sortie

/*************************************************************************/
        MAIN
/*************************************************************************/
void main()
{
 s32  ideltamod,deltamod;
u32 i;

 Init();

 ideltamod=0;
 input=&aic_in;
 output=&aic_out;

while(True)
  {
   Wait() ;

   i=ideltamod;
   i=i%(QuatriemeQuadrant);

if(i<=(PremierQaudrant))                    // La lecture en table sur
   {                                        // le premier quadrant
   *output =*(cos_tab+i);                   // pour le signal modulé
   }
 else if(i<(DeuxiemeQuadrant))
   {
   i=PremierQaudrant-(i%(PremierQaudrant));
   *output = -*(cos_tab+i);
   }
 else if(i<=(TroisiemeQuadrant))
   {
   i=i%(DeuxiemeQuadrant);
   *output = -*(cos_tab+i);
   }
 else
   {
   i=PremierQaudrant-(i%(TroisiemeQuadrant));
   *output = *(cos_tab+i);
   }

   deltamod=(fc+*input16)%(QuatriemeQuadrant);  // Calcul de l'index de lecture
   ideltamod=ideltamod+deltamod;                // en table pour le signal modulé
   ideltamod=ideltamod%(QuatriemeQuadrant);   }
}
void Init()
{

  if(brd_init(100))  // initialisation du processeur à 100 MHz
   return;

   /* Ouvrir le Codec */
  hHandset = codec_open(HANDSET_CODEC);          /* Acquire handle to codec */
```

```
   /* Determine les parametre du codec */
   codec_dac_mode(hHandset, CODEC_DAC_16BIT);        /* DAC en mode 16-bit   */
   codec_adc_mode(hHandset, CODEC_ADC_16BIT);        /* ADC en mode 16-bit   */
   codec_ain_gain(hHandset, CODEC_AIN_6dB);          /* gain 6dB de l'entrée analog à l'ADC */
   codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB); // gain -6dB de DAC à la sortie analog
   codec_sample_rate(hHandset,SR_16000);          /* frequence d'echantillonnage 16KHz  */

return;
}

void Wait()
{

    /* Attend l'echantillon suivant*/
  while (!MCBSP_RRDY(HANDSET_CODEC)) {};

    /* Lire l'echantillon d'entrée*/
    aic_in= *(volatile s16*)DRR1_ADDR(HANDSET_CODEC);

    /* emettre l'echantillon traité */
    *(volatile s16*)DXR1_ADDR(HANDSET_CODEC) = aic_out;

 return;
}
```

ANNEXE C

Le programme de démodulation FM pour le teste sur le DSK

```
/*************************************************************************/
/*              DSK_DEMOD.C                                              */
/*            ==============                                             */
*/                                                                        */
/*  Modulateur/démodulateur FM                                           */
/*  les paramètres:                                                      */
/*   k =2 pi1024  La constante de modulation                             */
/*   fs= 16KHz   La fréquence d'échantillonnage                          */
/*   fc= 1.6KHz    La fréquence de la porteuse                           */
/*************************************************************************/
//les fonctions des périphériques fournies par Texas Instruments
#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>

// les parametres du modulateur/demodulateur FM
#include "parametres.h"

/*************************************************************************/
/*  Function Prototypes:                                                 */
/*                                                                       */
/*  Init function:                                                       */
/*     Ouvrir le codec                                                   */
/*     determiner les parametre du codec                                 */
/*                                                                       */
/*  Wait function:                                                       */
/*     Attend l'echantillon suivant                                      */
/*                                                                       */
/*     Lire l'echantillon d'entrée et émettre l'echantillon traité       */
/*  LP_IIR_Filter function:                                              */
/*     Le filtre passe bas                                               */
/*                                                                       */
/*************************************************************************/
void Init() ;
void Wait() ;
static void LP_iir_Filtre(u16,bool )   ;

/*************************************************************************/
      Variables Globales
/*************************************************************************/
```

```
HANDLE hHandset          ;
s16    Sortie_VCO        ;        /* Le signal de sortie de VCO        */
s16    Sortie_Multiplieur ;       /* Le signal de sortie du multiplieur */
u16    length=SIZE_COS_TABLE ;
s16    aic_in,*input     ;                    //aic_in léchantillon d'entré
s16    Tmp0                    ;
s16    w_1[Ns*2]               ;        /* Le buffer du filtre IIR   */
s16    coef[Ns*5]=
{ /*i est l'index du bloc de filtre d'ordre 2*/
 /*les coef du filtre sont organisés comme suit:
   A[i][1],A[i][2],B[i][2],B[i][0],B[i][1]*/
    -25962,10340 ,1593  ,1593  ,-819  ,
    -27956,12278 ,2952  ,2952  ,-5038 ,
     -30641,14937 ,4850  ,4850  ,-8910   };

/*************************************************************************
   MAIN
*************************************************************************/

void main()
{
 s32 ideltamod,deltamod;
 u32 i;
bool ctrfiltre;

ideltamod=0;
 input=&aic_in;
 output=&aic_out;
 ctrfiltre=True;

for(i=0;i<Ns*2;i++)              /* Initialisation du buffer  */
  { w_1[i]=0;                    /* du filtre IIR             */
  }

  while(True)
  {
  Wait();
   // Le bloc VCO
   i=12000+ideltademod;
   i=i%(QuatriemeQuadrant);

if(i<=(PremierQaudrant))                  // La lecture en table sur
   {                                      // le premier quadrant
   *(Sortie_VCO+j)=*(cos_tab+i);          // pour le signal VCO
   }
else if(i<(DeuxiemeQuadrant))
   {
   i=PremierQaudrant-(i%(PremierQaudrant));
   *(Sortie_VCO+j)=-*(cos_tab+i);
   }
else if(i<=(TroisiemeQuadrant))
   {
   i=i%(DeuxiemeQuadrant);
   *(Sortie_VCO+j)=-*(cos_tab+i);
   }
```

```
  else
     {
     i=PremierQaudrant-(i%(TroisiemeQuadrant));
     *(Sortie_VCO+j)=*(cos_tab+i);
     }

     // Le miltiplieur
     tmp1=(s32)(*in_put*Sortie_VCO[j]);
     Sortie_Multiplieur=(s16)(tmp1>>15);

     // Le filtre IIR
    LP_iir_Filtre(j, ctrfiltre);

     *output=Tmp0;

     deltademod=(fc+Tmp0/16)%(QuatriemeQuadrant);        // Calcul de l'index de lecture
     ideltademod=ideltademod+deltademod;                 // en table pour le signal modulé
     ideltademod=ideltademod%(QuatriemeQuadrant);

     j=(j+1)%SIZE_BUFF;
     ctrfiltre=False;

    }
}

void LP_iir_Filtre(u16 k,bool ctr )
{
 u16 i,j,l,m,h;
 s32 w_0; s16 temp;
 m=Ns*5;
 h=Ns*2;
 j=0;
 if(ctr==True)
 {l=0;}

 w_0=(s32)Sortie_Multiplieur*5500;

 for(i=0;i<Ns;i++)
 {
    w_0=_smas(w_0,*(w_1+l),*(coef+j));j++;l=(l+Ns)%h;
    w_0=_smas(w_0,*(w_1+l),*(coef+j));j++;
    temp=*(w_1+l);
    *(w_1+l)=w_0>>15;
    w_0=_lsmpy(temp,*(coef+j));j++;
    w_0=_smac(w_0,*(w_1+l),*(coef+j));j++;l=(l+Ns)%h;
    w_0=_smac(w_0,*(w_1+l),*(coef+j));j=(j+1)%m;l=(l+1)%h;
 }

 Tmp0=(s16) (w_0>>15);

  return;
}
void Init()
{
```

```
  if(brd_init(100))  // initialisation du processeur à 100 MHz
   return;

   /* Ouvrir le Codec */
   hHandset = codec_open(HANDSET_CODEC);              /* Acquire handle to codec */

   /* Determine les parametre du codec */
   codec_dac_mode(hHandset, CODEC_DAC_16BIT);        /* DAC en mode 16-bit   */
   codec_adc_mode(hHandset, CODEC_ADC_16BIT);        /* ADC en mode 16-bit   */
   codec_ain_gain(hHandset, CODEC_AIN_6dB);          /* gain 6dB de l'entrée analog à l'ADC */
   codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB); // gain -6dB de DAC à la sortie analog
   codec_sample_rate(hHandset,SR_16000);      /* frequence d'echantillonnage 16KHz  */

return;
}

void Wait()
{

    /* Attend l'echantillon suivant*/
 while (!MCBSP_RRDY(HANDSET_CODEC)) {};

    /* Lire l'echantillon d'entrée*/
    aic_in= *(volatile s16*)DRR1_ADDR(HANDSET_CODEC);

    /* emettre l'echantillon traité */
    *(volatile s16*)DXR1_ADDR(HANDSET_CODEC) = aic_out;

 return;
}
```

ANNEXE D

*Le fichiers de commande et les deux fichier cosine.c et parametres.***h**

Le fichier cosine : contient la table de cosinus

```
/*********************************************************************
   cosine.c
*********************************************************************/
#include "type.h"
#include "parametres.h"
#pragma DATA_SECTION(cos_tab,"costab")
/*************************************
1/4 wave sine table.
*************************************/

s16 cos_tab[SIZE_COS_TABLE+1] =
{
 /*la table de cosinus contient L=4001 échantillons calculer par
 2^14cos(2πi/L) i= 0,1,...,4000.*/
....
}
```

Le fichier paramètres : contient les paramètres du modulateur/démodulateur FM

```
/*********************************************************************
  parametrs.h
*********************************************************************/
#include "type.h"
#define  SIZE_COS_TABLE 4000
#define  Ns  3
#define  fe 16000
#define  fc 1600
#define  SIZE_BUFF 320
#define  PremierQaudrant      SIZE_COS_TABLE
#define  DeuxiemeQuadrant     2*SIZE_COS_TABLE
#define  TroisiemeQuadrant    3*SIZE_COS_TABLE
#define  QuatriemeQuadrant    4*SIZE_COS_TABLE
extern  s16 cos_tab[SIZE_COS_TABLE+1];
```

le fichier de commande:

```
/*********************************************************************
 C5402 DSK DSP Memory Map
/*********************************************************************
```

```
MEMORY
{
  PAGE 0:  VECS:    origin = 0080h, length = 0080h  /* Internal Program RAM */
           PRAM:    origin = 7600h, length = 8000h  /* Internal Program RAM */

  PAGE 1: SCRATCH:  origin = 0060h, length = 0020h  /* Scratch Pad Data RAM */
          DMARAM:   origin = 0C00h, length = 0300h  /* DMA buffer */
          DATA:     origin = 1100h, length = 0080h  /* Internal Data RAM    */
          STACK:    origin = 1180h, length = 0560h  /* Stack Memory Space   */
      INRAM:    origin = 1900h, length = 0100h  /* Internal Data RAM    */
      HPRAM0:   origin = 1A00h, length = 0002h  /* HPI memory accessible by Host and DSP */
      HPRAM1:   origin = 1A02h, length = 0280h  /* HPI memory accessible by Host and DSP */
      HPRAM2:   origin = 1C82h, length = 0280h  /* HPI memory accessible by Host and DSP */
      EXRAM:    origin = 1F10h, length = 4800h  /* External Data RAM    */
}

/************************************************************************/
 DSP Memory Allocation
/************************************************************************/

SECTIONS
{
  .cinit     > PRAM  PAGE 0
  .text      > PRAM  PAGE 0
  .vectors > VECS   PAGE 0

  init_var > PRAM  PAGE 0
  detect    > PRAM  PAGE 0
  vrcprg    > PRAM  PAGE 0
  matprg   > PRAM  PAGE 0

  .stack     > STACK PAGE 1
  .trap      > SCRATCH PAGE 1

  .const     > EXRAM PAGE 1
  .data      > EXRAM PAGE 1
  .bss       > EXRAM PAGE 1
  .cio       > EXRAM PAGE 1

  .switch  > EXRAM PAGE 1

  costab   > EXRAM PAGE 1        /* table for cosine */

}
```

Références

[1]. B. Bouchez, Applications Audio Numériques des DSP, Publitronic, Paris.

[2]. G. Baudoin et F. Virolieu, les DSP famille TMS320 C54x : Développement d'applications, Dunod, Paris, 2000.

[3]. F. Coulon, Théorie et Traitement des Signaux, Dunod, Presses polytechniques romandes, 1984.

[4]. O. Sentieys, Processeur de Traitement Numérique de Signal (DSP), IRISA/ENSSAT Lannion, Université de Rannes I.

[5]. Texas Instruments , TMS320 C54x DSP Reference Set, Volume1: CPU and Peripherals, (SPRU131), 2001.

[6]. Texas Instruments , TMS320 C54x Optimizing C\C++ Compiler User's Guide, (SPRU103), 2001.

[7]. Texas Instruments , Optimized DSP library for C Programers on the TMS320 C54x, (SPRA480), 2000.

[8]. Texas Instruments , Assembly Language Tools User's Guide, (SPRU102), 2001.

[9]. Texas Instruments , Code Composer Studio IDE Quick Start, (SPRU405).

[10]. Texas Instruments, Code Composer Studio Online Documentation, (SPRU415), 2001.

[11]. Texas Instruments , Code Composer Studio Command Window, (SPRA601), 1999.

[12]. Texas Instruments , TMS320C5402 DSK HELP, (SPRH75A),1998-1999.

[13]. Texas Instruments , TMS320C54x DSKplus User's Guide, (SPRU191), 1998.

[14]. Texas Instruments , DSP Starter Kit (DSK), (SPRU368),2001.

[15]. Texas Instruments ,Using the TMS320C5402 DSK sine program and evmdsk54x.dll, (SPRA866), 2002.

[16]. Texas Instruments , TMS320VC5402 fixed point, Digital Signal Processing, (SPRU079E), 2000.

[17]. Texas Instruments , TMS320C54x DSP Reference Set Volume5: Enhanced Peripherals, (SPRU302),1999.

[18]. Texas Instruments , Data Converter Plug-in Help,(SPRH16A), 2001.

[19]. Texas Instruments , TMS320 DSP Algorithm Standard API Reference, (SPRU320), 2000.

[20]. Simon Haykin, Communication Systems, Weley et Sons, New York, 1983.

[21]. Steven A. Tretter, Communication System Design Usign DSP Algorithm With Laboratory Expriments for the TMS320C30, Plenum Press, New York and Landon, 1995.

[22]. Sen M. Kuo et Bob H. Lee, Reel Time Signal Processing: Implumentations, Applications and Expriments With the TMS320C55x, John Weiley,2001.

[23]. Vinay K.ingle, John G.Proakis, Digital Signal Processing Using MATLAB, Brooks/Cole ,Northeastern University.

De nombreuses documentations accessibles sur les sites Internet :

- http://www.ti.com/dsps.
- http://www.bdti.com.
- http://www.dspworld.com/dsps
- http://www.techonline.com/dsps
- http://www.ti.com/sc/docs/psheets/abstractapps/slaa214.htm
- http://www.ti.com/sc/docs/tools/dsp/c5000/softup.htm

ملخص:

يعرض هدا العمل ادماج نظام اتصال مضمن التواتر (FM) على لوحة DSP TMSP320VC5402 لـ: Texas instruments . بعد تقديم عرض للخصائص البرمجية و المادية للوحة. تم تطوير تقنية الادماج للمحاكات على الكود كومبوزر (*Code Composer*) وتشغيله خلال الزمن الحقيقي على جهاز التطوير DSKVC5402.

كلمات مفتاحية: كمبيوتر المعالجة الرقمية للاشارة (DSP) جهاز التطوير (DSK)، تضمين التواتر (FM)، تضمين التواتر ذو الحزمة المحدودة(NFM) ، حلقة التحكم الطوري (PLL).

Résumé :

Ce travaille présente l'implémentation d'un modulateur démodulateur FM sur le DSP TMS320VC5402 de *Texas Instruments*. Après une présentation des différentes caractéristiques logicielles et matérielles du DSP cible. On a décrit la technique d'implémentation sur le DSP pour la simulation sur le *Code Composer* et l'exécution en temps réel sur le Kit de développement DSK VC5402.

Mots clés :processeur de traitement numérique de signal(DSP), le Kit de développement (DSK), Modulation en fréquence (FM), Modulation en fréquence à bande étroite (NFM), Boucle de Phase Asservie (PLL).

Abstract:

this work present the implementation of a modulator and a demodulator FM on the DSP TMS320VC5402 of Texas Instruments. after a presentation of the various characteristics software and hardware of the target *DSP*. We described the technique of implementation on the *DSP* for simulation on the code Composer Studio and the execution in real time on the starter Kit DSKVC5402.

Keywords:Digital signal processing (DSP), Evaluation Module (DSK), frequency modulation (FM), Narrow band FM Modulation, (NFM), phase locked loop ((PLL).