TP Processeurs de Traitement du Signal

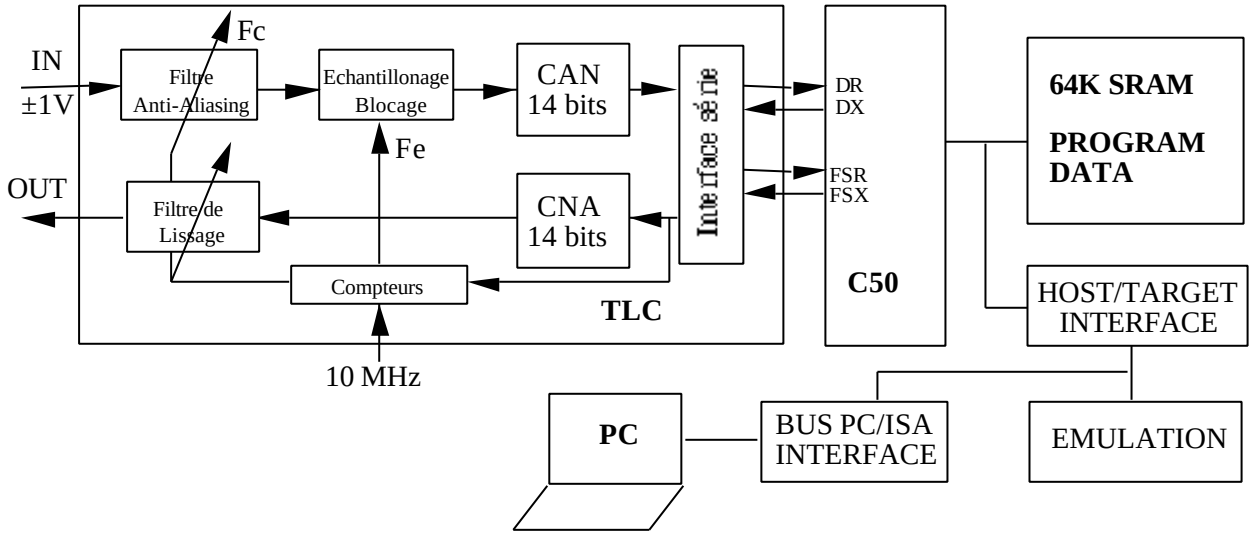
Nous allons étudier dans ce TP l'implantation d'algorithmes de traitement du signal dans le Processeur de Traitement du Signal TMS320C50 de Texas Instruments.

Pour le fonctionnement du TP et l'accès à la chaîne de développement, cliquez sur l'icône *Traitement du Signal/TMS/TMSC25 C50/* TMSC50 dans le dossier *EII*. Les fichiers sources nécessaires se trouvent dans *Informations ENSSAT /TP ENSSAT/TMS/TP_EII2*. Copiez tous ces fichiers (sauf le répertoire enseignants) dans votre répertoire de travail.

I. Présentation de l'environnement de développement.

Chaque poste est équipé d'un PC sur lequel on pourra éditer, compiler, assembler et exécuter du code pour TMS320C50. Le PC est relié à une carte de développement de Texas Instruments EVM C50 ou Starter Kit C50 suivant les postes, sur laquelle se trouve un DSP C50, une interface avec le monde analogique (Analog Interface Circuit) TLC.

Le TLC contient les éléments indispensables à une chaîne de TNS (Filtrage anti-repliement, CAN, Interface avec le DSP, CNA, Filtre de lissage). Il contrôle la fréquence d'échantillonnage (Fe) et la fréquence de coupure (Fc) des filtres d'entrée et de sortie.



Deux registres TA et TB permettent de fixer les valeurs maximales de deux compteurs. Ceux-ci sont ensuite utilisés pour diviser la fréquence de l'horloge externe ($F_{MasterClock}$=10.368 MHz) afin de régler Fe et Fc selon les besoins.

Ces valeurs peuvent être choisies en modifiant les fichiers d'initialisation de la carte, développés en assembleur : *InitEVM.asm (quand vous développez en C)* ou *Filtre.asm (quand vous développez en assembleur).*

Les outils disponibles après la commande *TMS* sur le PC sont:

- dspac, dspcg: un compilateur C (parser, générateur de code assembleur) pour la famille TMS en Virgule Fixe
- dspa: un assembleur pour la famille TMS Virgule Fixe permettant d'obtenir les fichiers objets.
- dsplnk: un éditeur de lien produit l'exécutable (.out) à partir des fichiers objets (programme source, programme de configuration) et du *mapping* mémoire choisi.
- EVM5x pour les cartes EVM ou DSK5D pour les cartes Starter Kit: un debbuger/émulateur interactif permettant de télécharger, d'exécuter et d'observer du code sur la carte DSP

II. Implantation d'un filtre passe bande RII sélectif

On s'intéresse à un filtre sélectif Passe Bande centré sur F0 = 2400 Hz, de bande passante [1600 3200] et de bande atténuée [0 1400] U [3400 8000] dont le gabarit est donné figure 1.1. La sélectivité du filtre vaut 0.8, l'ondulation de la courbe dans la bande passante est de 3 dB, l'affaiblissement doit être de 30 dB. La fréquence d'échantillonnage est égale à 16 KHz.

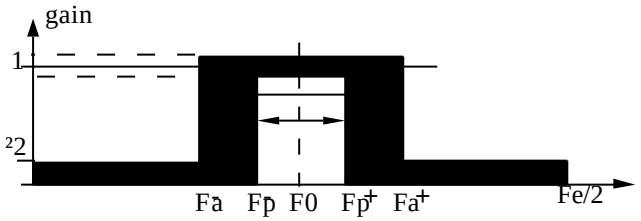


Figure 1.1

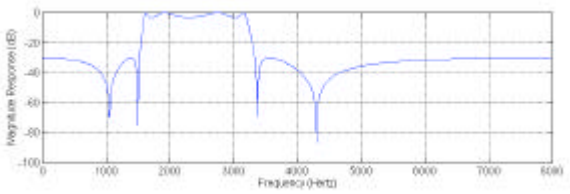


Figure 1.2

Ce filtre est d'ordre 8, une représentation cascadée de 4 filtres d'ordre 2 et non cascadée est donnée ci après.

$$H(z) = \frac{0.3344 - 0.5567.z^{-1} + 0.3344.z^{-2}}{1 - 1.382.z^{-1} + 0.877.z^{-2}} * \frac{0.5 - 0.2428.z^{-1} + 0.5.z^{-2}}{1 - 0.853.z^{-1} + 0.8417.z^{-2}} * \frac{0.5 + 0.1186.z^{-1} + 0.5.z^{-2}}{1 - 0.6234.z^{-1} + 0.969.z^{-2}} * \frac{0.5 - 0.9159.z^{-1} + 0.5.z^{-2}}{1 - 1.5964.z^{-1} + 0.98.z^{-2}}$$

$$H(z) = \frac{0.0418 - 0.1567.z^{-1} + 0.3265.z^{-2} - 0.4849.z^{-3} + 0.5546.z^{-4} - 0.4849.z^{-5} + 0.3265.z^{-6} - 0.1567.z^{-7} + 0.0418.z^{-8}}{1 - 4.4548.z^{-1} + 10.8034.z^{-2} - 17.0825.z^{-3} + 19.2862.z^{-4} - 15.6429.z^{-5} + 9.0510.z^{-6} - 3.4087.z^{-7} + 0.7013.z^{-8}}$$

II.1 Forme cascadée du filtre RII en langage C

Pour chaque filtre à tester vous disposez de :

C_IIR/INITEVM.ASM : Fichier contenant la procédure principale (initialisation, configuration du composant TLC, Lecture/Écriture des échantillons).

C_IIR/MAIN.C : Procédure de filtrage et d'initialisation des coefficients du filtre

C_IIR/compile : Batch de compilation

- A partir du code fourni dans le fichier main.c, déterminez le graphe flot du filtre. ***NB: quand nous utilisons le C, le P register n'est pas mis en mode shift automatique pour l'élimination du bit de signe redondant.***

- Nous considérons que l'entrée du filtre est comprise dans l'intervalle]-1,1[. Déterminez la dynamique des signaux $y_i(n)$ et $d_i(n)$

- Déterminez le codage des données, des coefficients et les recadrages nécessaires afin que le calcul s'effectue sans débordement et avec le maximum de précision

- Complétez le programme C avec la valeur des coefficients et les différents décalages dans le fichier main.c

- Vérifiez le gabarit du filtre passe-bande implanté et l'absence de débordement en mettant une amplitude élevée pour le signal d'entrée

- Vérifiez l'influence de la fréquence d'échantillonnage sur la fréquence de coupure en changeant TA et TB dans InitEVM.asm pour avoir Fe à 10kHz .

II.2 Forme non cascadée du filtre RII en langage C

Les fichiers se trouvent dans le répertoire C_*IIR_NC*.

- Reprenez la démarche présentée dans la partie II.1 pour déterminer
    - le graphe flot du filtre.
    - la dynamique des signaux $y(n)$ et $d(n)$
    - le codage des données, des coefficients et des différents recadrages.

- Complétez le programme C avec la valeur des coefficients et les différents décalages dans le fichier main.c

- Exécutez le filtre et analysez sa stabilité. Concluez par rapport aux résultats vus en cours.

II.3 Forme cascadée du filtre RII en langage *assembleur.*

Pour chaque filtre à tester vous disposez de :

A_IIR/FILTRE.ASM	Fichier contenant la procédure principale (initialisation, configuration du composant TLC, Lecture/Écriture des échantillons).
A_IIR/FIL.ASM	Procédure de filtrage et d'initialisation des coefficients du filtre
A_IIR/compile	Batch de compilation

- Reprenez la démarche présentée dans la partie II.1 pour déterminer

- le graphe flot du filtre. ***NB: ici le registre P est mis en mode shift automatique pour l'élimination automatique du bit de signe redondant.***
- la dynamique des signaux
- le codage des données, des coefficients et des différents recadrages

- Complétez le programme assembleur avec la valeur des coefficients et les différents décalages dans le fichier *fil.asm*

- Vérifiez le gabarit du filtre passe-bande implanté et l'absence de débordement. Comparez avec l'assembleur généré précédemment par le C. Observez notamment comment sont cascadés les calculs des filtres d'ordre 2.

II.4 Etude comparée des performances du C et de l'assembleur.

Sur l'exemple du filtre cascadé, nous allons comparer les performances du programme écrit en C et du programme écrit en assembleur.

Mesure de performances : le C50 possède une broche de sortie XF qui peut être mise à 1 ou à 0 par une instruction (respectivement SETC XF et CLRC XF). Insérez une mise à 1 au début du code et une mise à zéro à la fin du code et visualisez sur l'oscilloscope par observation de la broche XF la durée d'exécution du code pour le C et l'assembleur.

Analyse des codes : le code C est donc moins rapide que l'assembleur dans des proportions assez importantes. L'observation de certaines parties du code permet d'expliquer ce phénomène :

- Observez la gestion des boucles en C et en assembleur.
- Observez la gestion de la double précision en C et en assembleur.
- Observez la gestion des accès aux tables de coefficients et d'échantillons en C et en assembleur.

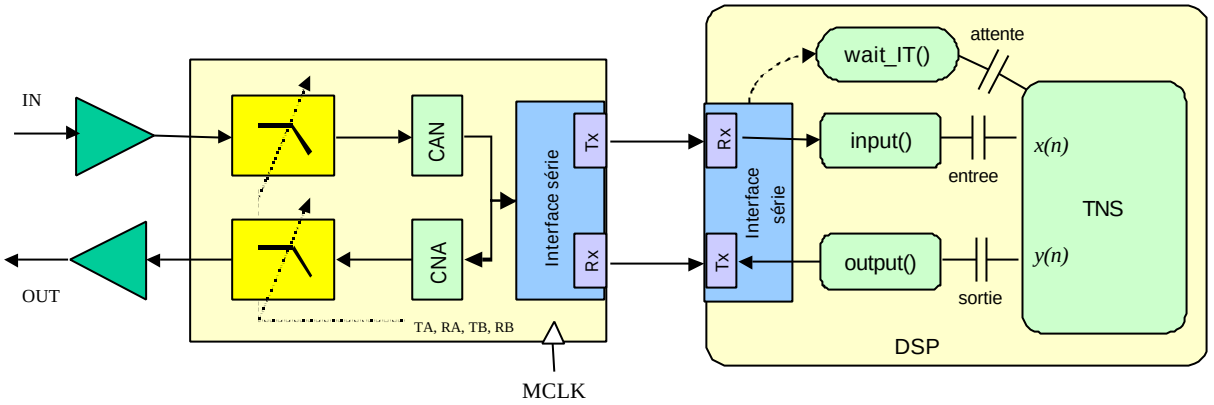
III. Observation des phénomènes classiques dus à l'échantillonnage.

Nous allons observer les effets de l'échantillonnage (sous-échantillonnage, repliement) vus de façon théorique en cours, en utilisant un filtre passe-tout, et en jouant sur les fréquences Fe et Fc. Les fichiers se trouvent dans le répertoire ALLPASS. Le fichier filtre.asm écrit en assembleur initialise le convertisseur TLC et réalise un écho du signal d'entrée sur la sortie, en passant néanmoins par les filtres du TLC. Compilez linkez le fichier *Filtre.asm* en utilisant le batch *Compile.bat*, lancez le debugger, puis chargez et exécutez le code *Filtre.out*. Utilisez le générateur de fonction pour entrer une sinusoïde sur l'entrée analogique de la carte et visualisez la sortie sur l'oscilloscope.

- Prenez les valeurs *CAS 1* pour TA et TB (Fe=16KHz, Fc_{in} et Fc_{out}=8KHz). Vérifiez que pour tout signal de fréquence inférieure à Fe/2, on a bien un filtre passe tout.

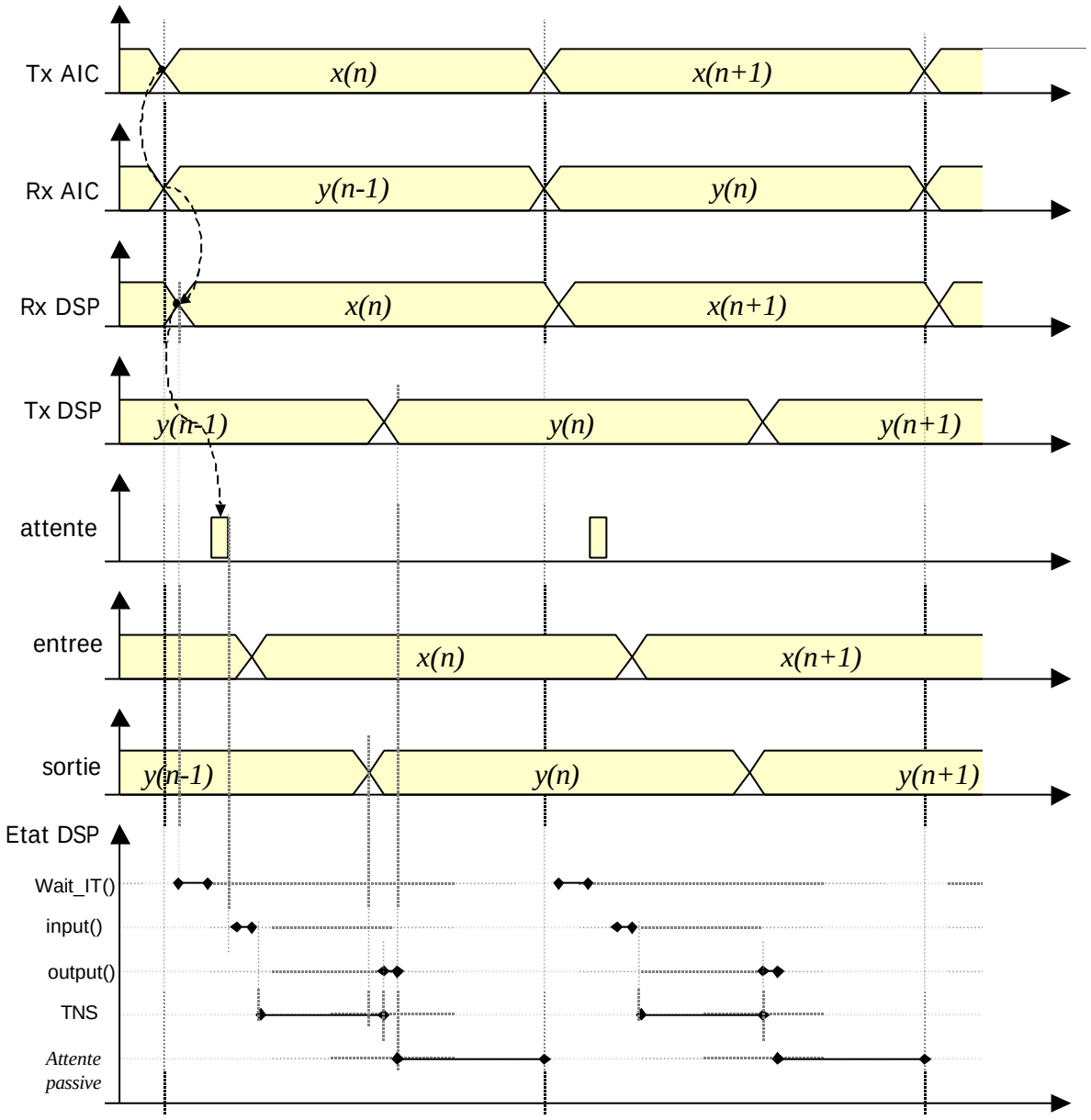
- Prenez les valeurs *CAS 2* pour TA et TB (Fe=16KHz, Fc_{in}=16KHz, Fc_{out}= 8KHz). Mettez un signal à 10KHz à l'entrée. Quelle est la nature du signal observé en sortie (forme, fréquence …)? Pourquoi ?
- Prenez les valeurs *CAS 3* pour TA et TB (Fe=16Khz, Fc=16Khz). Mettez un signal sinusoïdal à 10Khz à l'entrée. Quelle est la nature du signal observé en sortie (forme, fréquence …)? Pourquoi ?

IV. Annexe : traitements temps réel



Description fonctionnelle de la chaîne de traitement

```
extern void input();
extern void output();
extern void init_DSP();
extern void debut();
extern void fin();

/* Variables globales       */

int entree,sortie;
int attente;

/* Programme principal     */

main()
{


init_DSP();

        for (;;)
        {           input();
                    sortie = entree ;
                   output();

                    while(attente==0){};
                    attente=0;
        }
}
```


Tx AIC
x(n)
x(n+1)
Rx AIC
y(n-1)
y(n)
Rx DSP
x(n)
x(n+1)
Tx DSP
y(n-1)
y(n)
y(n+1)
attente
entree
x(n)
x(n+1)
sortie
y(n-1)
y(n)
y(n+1)
Etat DSP
Wait_IT()
input()
output()
TNS
Attente passive