UNIVERSITÉ DE SHERBROOKE
Plan de Test V2
S4 App6
Traitement de signal sur μC

1. A1

1.1 DESCRIPTION

Calculer la valeur de la variable spectralResolution (type double). Cette variable équivaut au delta de fréquences entre chaque valeur de n dans le spectre de frequence.

1.2 RÉSULTAT ATTENDU

Le résultat attendu est 19.53125 (Fe/N = 20 000/ 1024)

1.3 TEST À EXÉCUTER

Afficher la valeur de spectralResolution à l'aide du debugger

1.4 RÉSULTAT OBTENU

19.53125

2. A2 & A3

2.1 CALCUL DE FFT SANS FENÊTRAGE

2.1.1 DESCRIPTION

Calculer la FFT du signal d'entrée à 2000Hz sans fenêtrage

2.1.2 RÉSULTAT ATTENDU

2 « peaks » doivent être présent à l'indice 102 et 922. Il devrait avoir un peu de fuite spectrale.

2.1.3 TEST À EXÉCUTER

Afficher à l'aide de DMCI le module carré du spectre en dB. Comparer avec Python

2.1.4 RÉSULTAT OBTENU

La Figure 1 montre le résultat obtenu à la suite de la FFT, puis la Figure 2 montre le résultat obtenu à l'aide de python

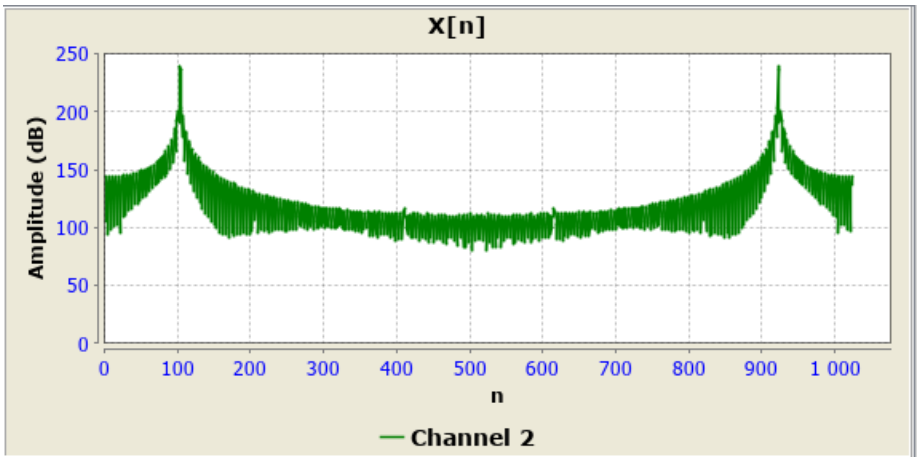


Figure 1 X[n] d'un signal d'entrée à 2000Hz sans fenêtrage sur DMCI

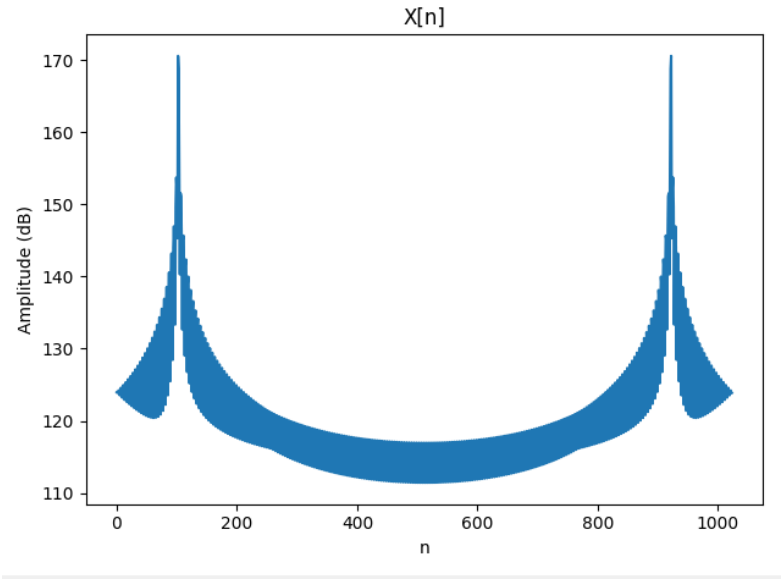


Figure 2 : X[n] d'un signal d'entrée à 2000Hz sans fenêtrage sur Python

2.2 CALCUL DE FFT AVEC FENÊTRAGE

2.3 DESCRIPTION

Calculer la FFT du signal d'entrée à 2000Hz avec fenêtrage

2.4 RÉSULTAT ATTENDU

Les résultats seront semblables au test précédent, mais il y aura moins de fuites spectrales

2.5 TEST À EXÉCUTER

Afficher à l'aide de DMCI le module carré du spectre en dB. Comparer avec le test précédent et python

2.6 RÉSULTAT OBTENU

La Figure 3 montre le résultat obtenu à la suite de la FFT, puis la Figure 4Figure 3Figure 2 montre le résultat obtenu à l'aide de python

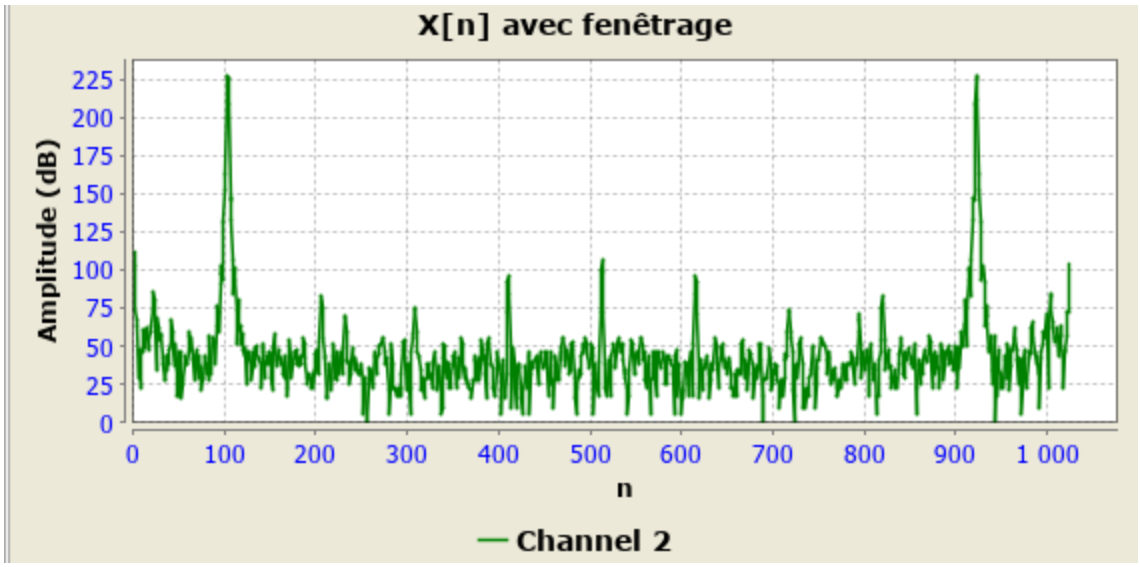


Figure 3 : X[n] d'un signal d'entrée à 2000Hz avec fenêtrage sur DMCI

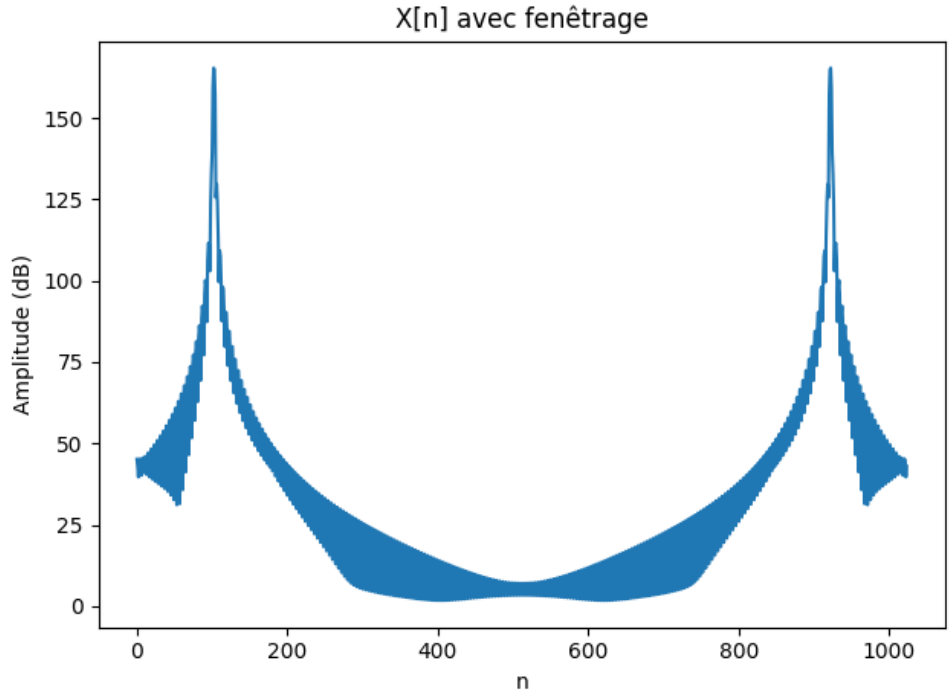


Figure 4 : X[n] d'un signal d'entrée à 2000Hz avec fenêtrage sur Python

3. A4

3.1 DESCRIPTION

Le point A4 permet de calculer la variable maxAmplFreq (type int)

3.2 RÉSULTAT ATTENDU

Le résultat devrait être équivalent ou presque aux fréquences d'entrée

3.3 TEST À EXÉCUTER

Mettre à l'entrée un sinus d'une fréquence connu, puis vérifier sur les afficheurs 7 segments si la valeur affichée et presque équivalente à la fréquence d'entrée

3.4 RÉSULTAT OBTENU

La fréquence affichée est presque équivalente.

4. CONCEPTION DES FILTRES FIR

4.1 DESCRIPTION

Affichage superposer des filtres FIR conçu avec Python (amplitude selon la fréquence)

4.2 RÉSULTAT ATTENDU

Passe bas fc à 500hz

Passe-Bade#1 → 1000 +/- 500Hz

Passe-Bade#2 → 2000 +/- 500Hz

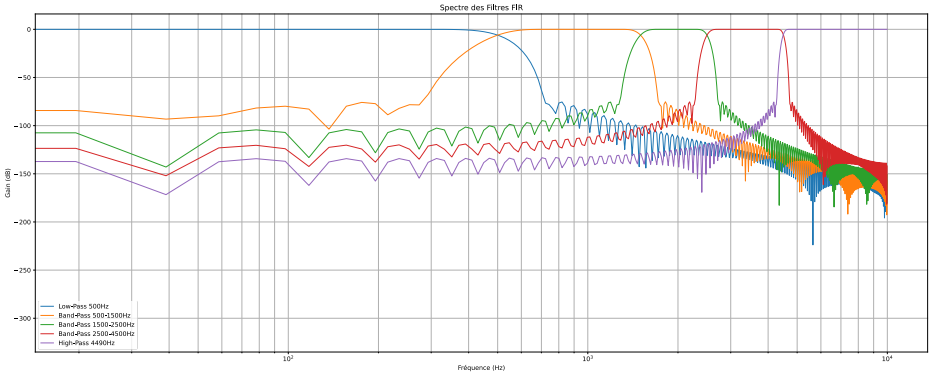
Passe-Bade#3 → 3500 +/- 1000Hz

Passe-Haut → fc = 4490 Hz

4.3 TEST À EXÉCUTER

Affichage superposer des filtre FIR conçu avec Python

4.4 RÉSULTAT OBTENU



5. CRÉATION DU FICHIER FIRCOEFFS.H

5.1 DESCRIPTION

Les filtres maintenant conçus, ils doivent être transmis vers le microcontrôleur. Pour se faire, la création du fichier .h est nécessaire.

5.2 RÉSULTAT ATTENDU

En format Q2.13 et de longueur 4N (1024). Le filtre H7 déjà conçu, le nouveau passe bas doit y correspondre.

5.3 TEST À EXÉCUTER

1. Rouler le script python et générer ainsi un nouveau fichier .h
2. Ouvrir le .h et comparer les valeurs de H7 avec les valeurs données du H7 précalculer.
3. Vérifier que tous les filtres sont créés en format Q2.13 et de longueurs 1024

5.4 RÉSULTAT OBTENUE

Le filtre H7 obtenue correspond exactement au H7 donnée, toutes les données sont en entier Q2.13 et les filtres de longueurs 1024.

```
int32c H_LP[1024] = {
{8192,0},
{5810,-5775},
{50,-8192},
{-5739,-5845},
{-8190,-101},
{-5880,5702},
{-151,8190},
{5667,5915},
{8190,201},
{5951,-5631},
{251,-8189},
{-5594,-5985},
{-8185,-301},
{-6018,5556},
```

6. B0

6.1 SIGNAL D'ENTRÉ RALLONGÉ À 4N

Afin d'utiliser la méthode « overlap and save » avec des blocs de 3N, le signal d'entrée doit être calculer pour 4N pour en appliquer la FFT.

6.2 RÉSULTAT ATTENDU

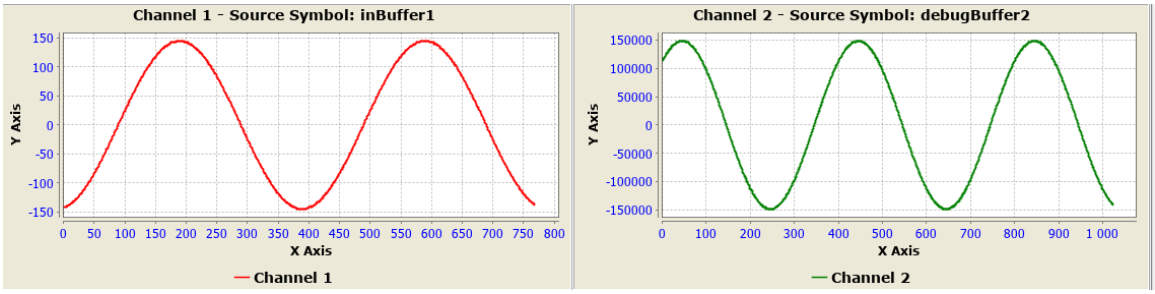
Un nouveau buffer de longueur 4N contenant le signal d'entrée.

Pour N = 256, 4N = 1024. Donc une entrée de longueur 4N

6.3 TEST À EXÉCUTER

Appliquer un signal à l'entrée (Sinus 500Hz). Avec le Debug et le DMCI, observer le buffer contenant 4N de signal d'entrée.

6.4 RÉSULTAT OBTENU



7. B1

7.1 CALCUL DE LA FFT DU SIGNAL DE 4N DE LONGUEUR

La transformer de fourrier doit être exécuter afin de passer du domaine temporel vers le spectre fréquentiel du signal obtenue en entrée. X[m] = FFT(x[n]). Ce spectre de fréquence sera utilisé afin de calculer le spectre de fréquence du signal de sortie.

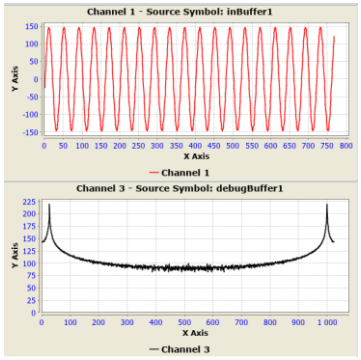
7.2 RÉSULTAT ATTENDU

Pour un signal à l'entrée contenant un Sinus, un seul « peak » seras observer sur son spectre fréquentielle (2 avec les fréquences négatives) correspondant à la fréquence du Sinus en entrée.

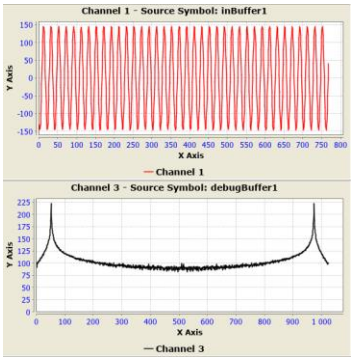
7.3 TEST À EXÉCUTER

Appliquer un signal à l'entrée (Sinus 500Hz). Avec le Debug et le DMCI, observer le buffer contenant la sortie de la transformé de fourrier rapide (X[m]).

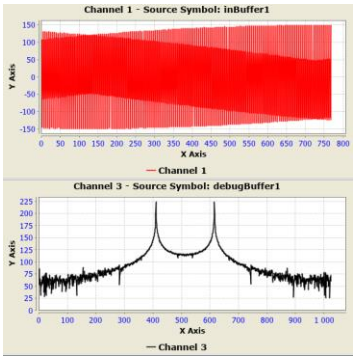
7.4 RÉSULTAT OBTENU



(500 Hz)



(1000Hz)



(8000Hz)

B2 – FILTRAGE FRÉQUENTIELLE

7.5 MULTIPLICATION Y = XH

Le calcul semble simple à première vue mais le calcul contient des nombre complexe et plusieurs méthodes sont disponible afin de calculer la multiplication.

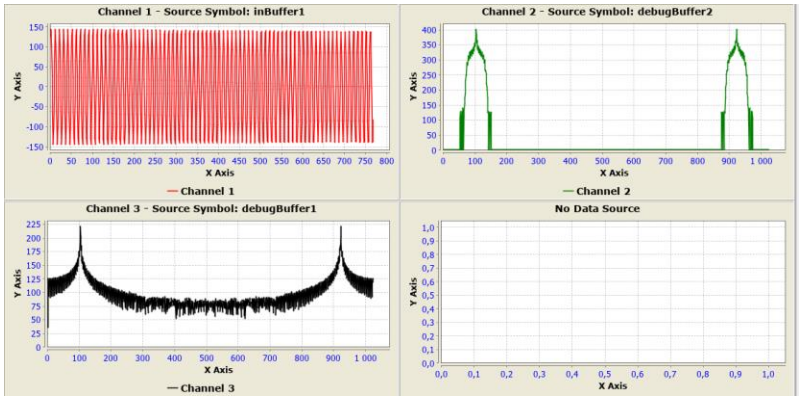
7.6 RÉSULTAT ATTENDU

Une multiplication de nombre complexe approprié.

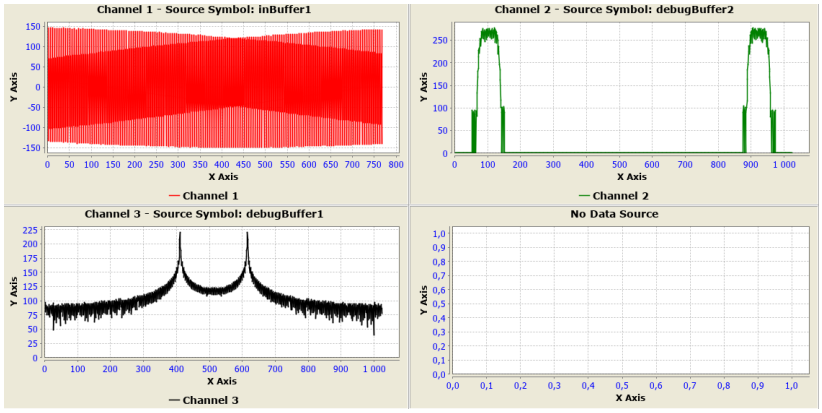
7.7 TEST À EXÉCUTER

Appliquer différente fréquence d''entrée pour un même filtre. Observer l'amplitude finale.

7.8 RÉSULTAT OBTENU



2000Hz = x[n] H5 = 2000 +/-500Hz DebugBuffer2 = Y = XH



8000Hz = x[n] H5 = 2000 +/-500Hz DebugBuffer2 = Y = XH

8. B0 À B4

8.1 DESCRIPTION

La combinaison des 5 points permet de filtrer le signal d'entrée. Le signal sera comparé à l'entrée et à la sortie.

8.2 TEST À EXÉCUTER

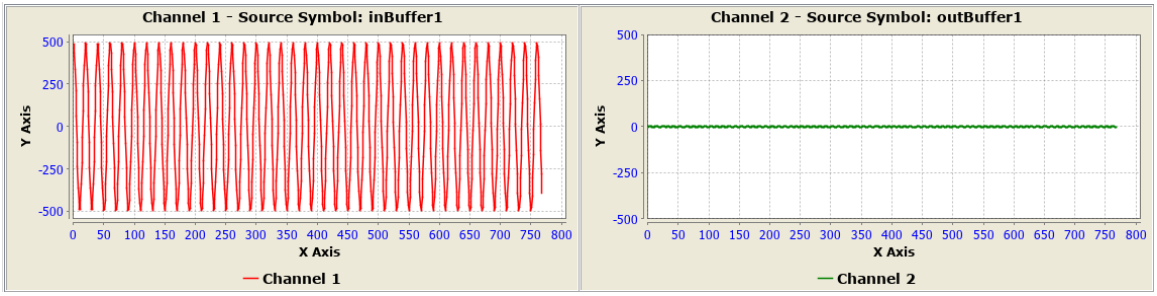
Tester une fréquence inférieure et supérieure à la bande passante du filtre H6 (1000Hz) et H5 (2000Hz). De plus, une fréquence équivalente à la fréquence de coupure sera testé.

8.3 RÉSULTAT ATTENDU

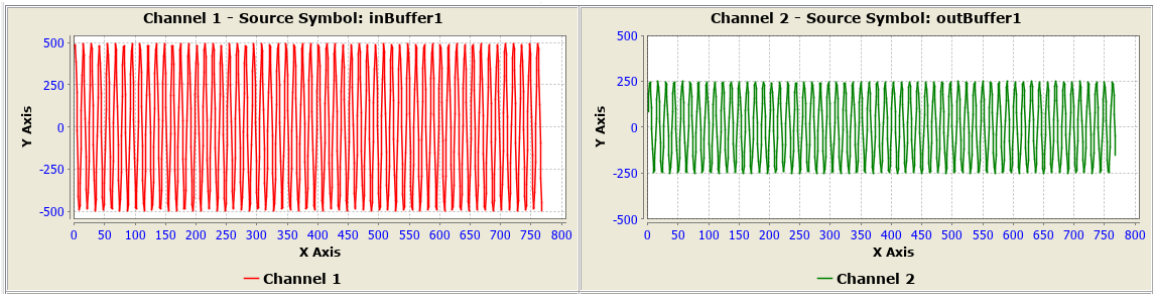
Lorsque la fréquence testée est inférieure ou supérieure à la bande passante du filtre, la valeur devrait varier autour de 0. Lorsque la fréquence testée est égale à la fréquence de coupure, le signal devrait être atténué de moitié.

8.4 RÉSULTAT OBTENU

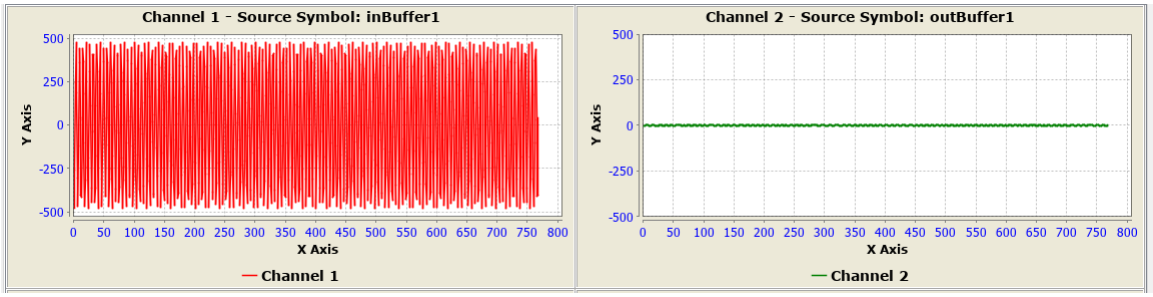
La figure suivante montre le résultat obtenu avec le filtre H5 et une fréquence d'entrée de 1000Hz (inférieur donc null).



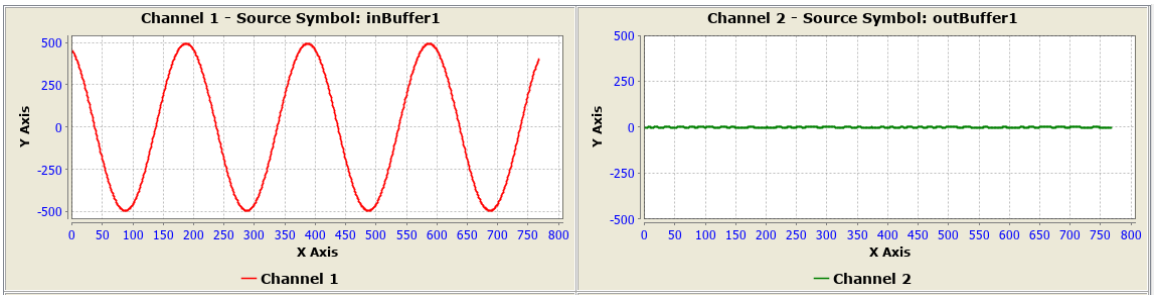
La figure suivante montre le résultat obtenu avec le filtre H5 et une fréquence d'entrée de 2000Hz (f = fc donc divisé en 2).



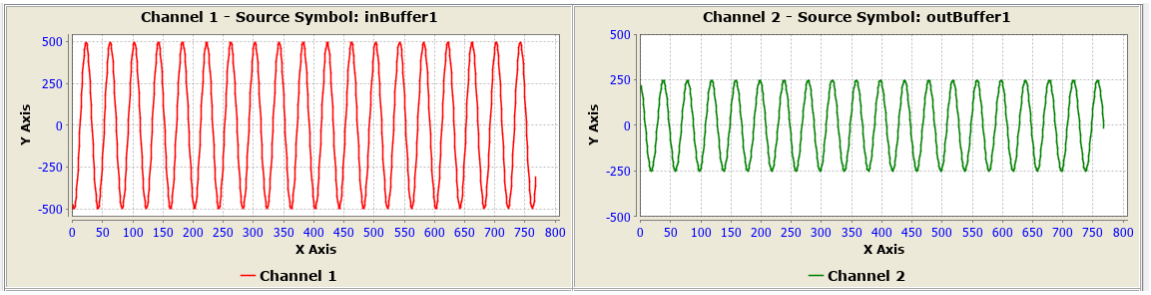
La figure suivante montre le résultat obtenu avec le filtre H5 et une fréquence d'entrée de 3500Hz (supérieur donc null).



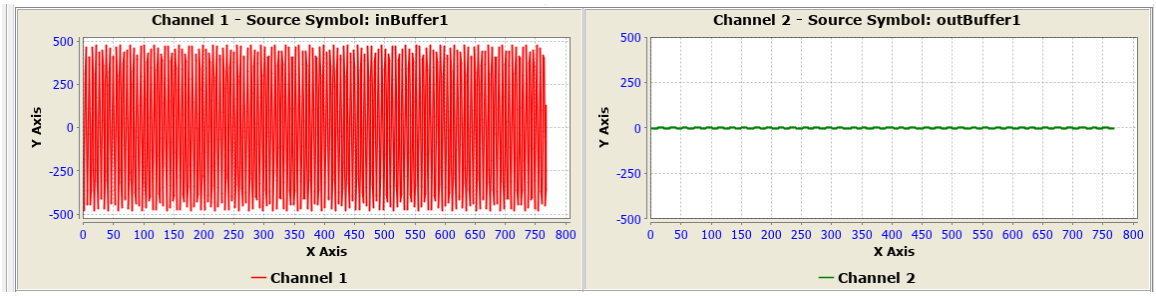
La figure suivante montre le résultat obtenu avec le filtre H6 et une fréquence d'entrée de 100Hz (inférieur donc nulle).



La figure suivante montre le résultat obtenu avec le filtre H6 et une fréquence d'entrée de 1000Hz (f = fc donc divisé en 2).



La figure suivante montre le résultat obtenu avec le filtre H6 et une fréquence d'entrée de 3500Hz (supérieur donc nulle).



C1 – FILTRE IIR

8.5 CALCUL DES COEFFICIENTS

Déterminer les coefficients du filtre IIR. Pour un filtre IIR coupe-bande elliptique d'ordre4

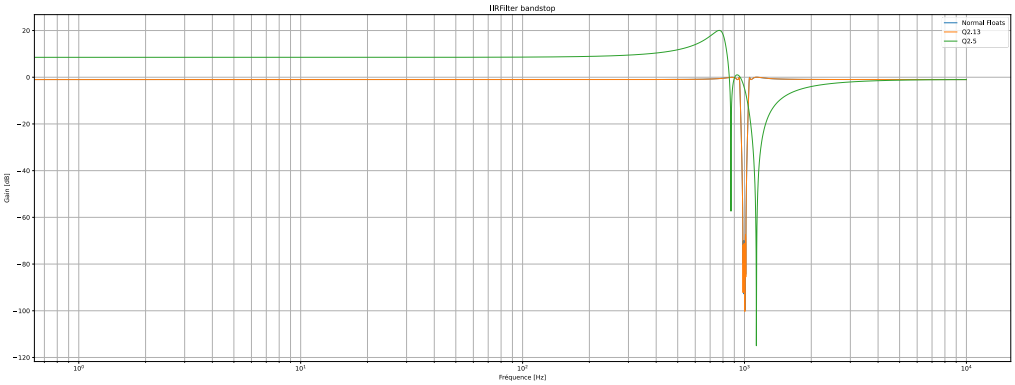
8.5.1 RÉSULTAT ATTENDU

Obtenir un Gain atténué à -70dB entre 950 et 1050Hz et un Gain de 0 pour les autres fréquences.

8.5.2 TEST À EXÉCUTER

Afficher le spectre fréquentiel du filtre développer sur python. Tester ce filtre pour différents formats de données QXY (Q2.5, Q2.13)

8.5.3 RÉSULTAT OBTENU



8.6 CRÉATION DU FICHIER IIRCOEFFS.H

8.7 DESCRIPTION

Les filtres maintenant conçus, ils doivent être transmis vers le microcontrôleur. Pour se faire, la création du fichier .h est nécessaire.

8.8 RÉSULTAT ATTENDU

En format Q2.13, un double tableau [6][4] de dimension.

8.9 TEST À EXÉCUTER

Exécuter le script python et observer le fichier IIRcoeffs créer.

8.10 RÉSULTAT OBTENUE

Toutes les données sont en entier Q2.13.

```
int32_t IIRCoeffs[N_SOS_SECTIONS][6] = {
{ 7005, -13319, 7005, 8192, -15164, 7877},
{ 8192, -15593, 8191, 8192, -15427, 7917},
{ 8192, -15560, 8192, 8192, -15466, 8155},
{ 8192, -15606, 8192, 8192, -15628, 8159},
};
```

Toutes les données sont en entier Q2.5.

```
int32_t IIRCoeffs[N_SOS_SECTIONS][6] = {
{ 27, -52, 27, 32, -59, 30},
{ 32, -60, 31, 32, -60, 30},
{ 32, -60, 32, 32, -60, 31},
{ 32, -60, 32, 32, -61, 31},
};
```

8.11 IMPLÉMENTATION DANS MPLAB

Maintenant le filtre conçu, il faut que l'implémentation dans MPlab soit fonctionnelle.

Affichages des valeurs calculer en sortie par rapports aux entrées pour différentes fréquences d'entrées. (500Hz, 900Hz, 1000 Hz, 1100Hz 5000Hz)

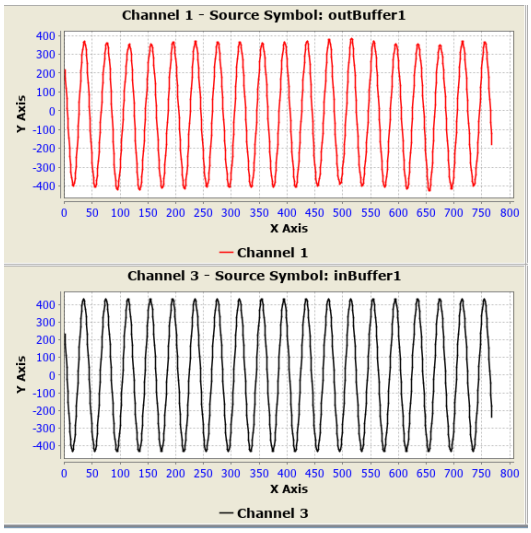
Résultat Attendu

Obtenir un Gain atténué à -70dB entre 950 et 1050Hz et un Gain de 0 pour les autres fréquences soit le même signal à l'entrée qu'à la sortie.

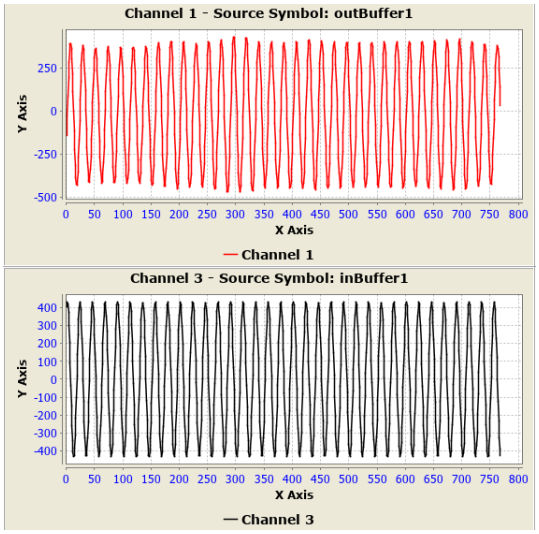
Test à exécuter

À l'aide de l'outil DCMI disponible pour MPlab, afficher les valeurs sous formes de graphiques les valeurs à l'entrée et à la sortie du filtre IIR.

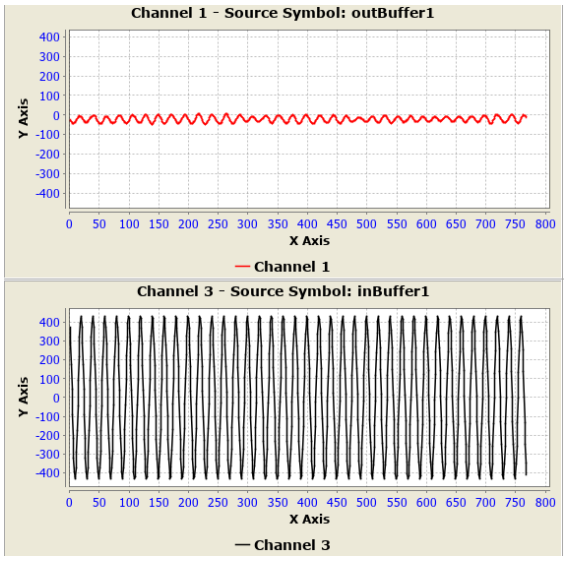
Résultats obtenu



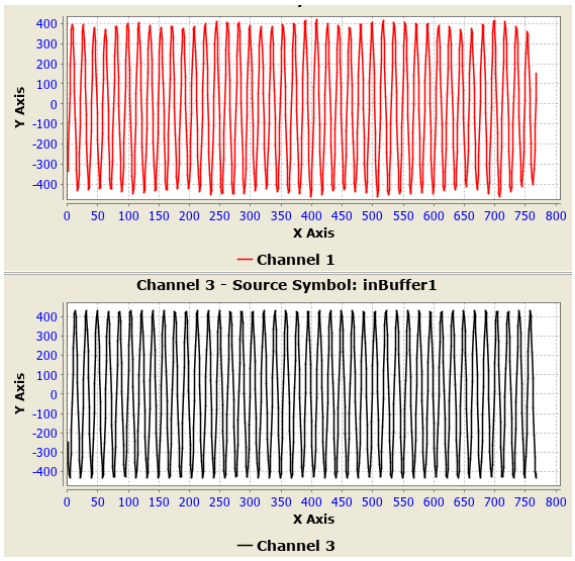
(500 Hz)



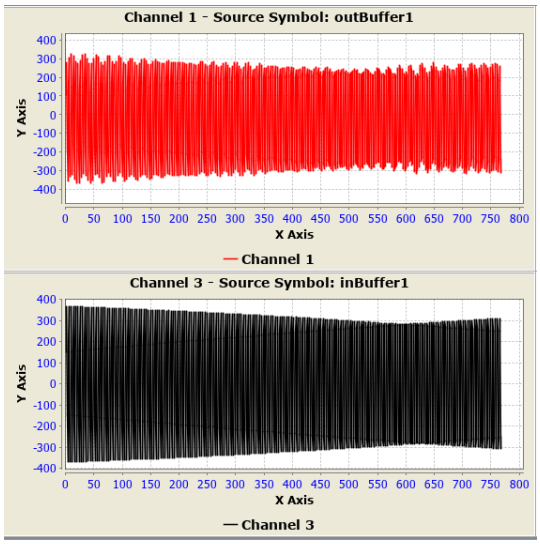
(900 Hz)



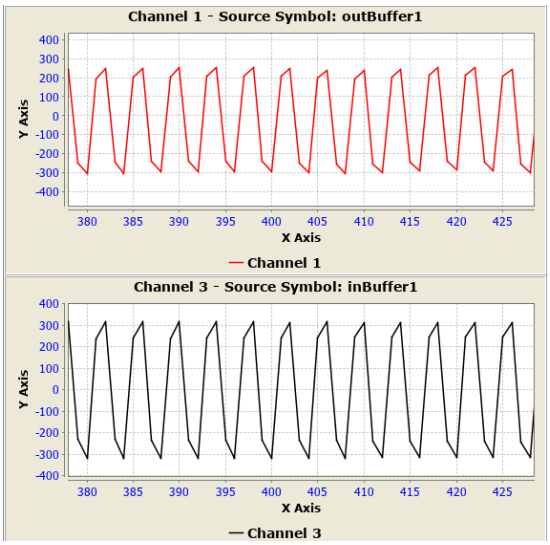
(1000 Hz)



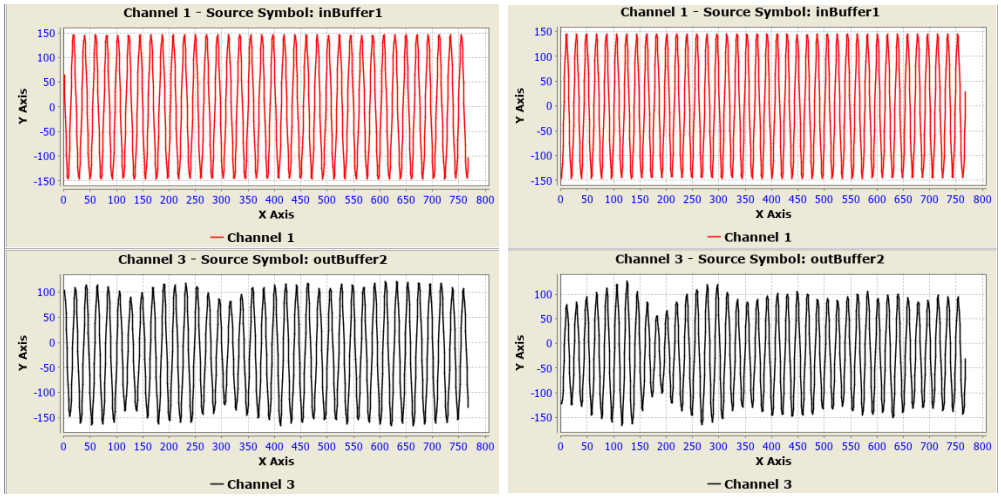
(1100Hz)



(5000 Hz)



(5000 Hz zoomed in)



(950 Hz) (1050Hz)

9. FILTRE IIR EN RESSOURCES LIMITÉS (Q2.5)

En supposant que notre microcontrôleur est limité en ressource et que la résolutions de nos données sont mis à l'échelle de Q2.5. Quel seras le résultat en pratique du filtre IIR coupe bande.

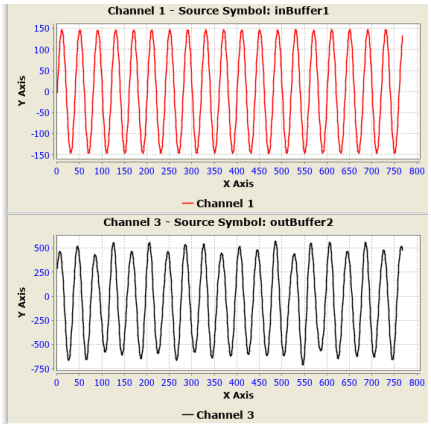
9.1.1 RÉSULTAT ATTENDU

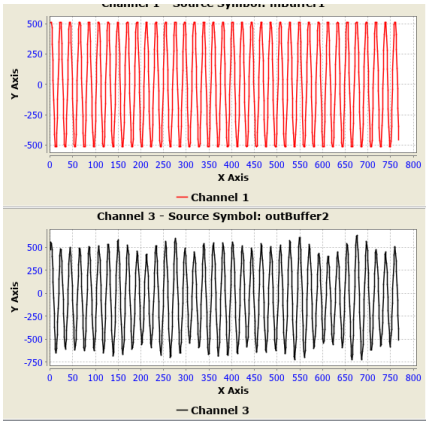
Obtenir un Gain atténué à -70dB entre 950 et 1050Hz et un Gain de 0 pour les autres fréquences soit le même signal à l'entrée qu'à la sortie.

9.1.2 TEST À EXÉCUTER

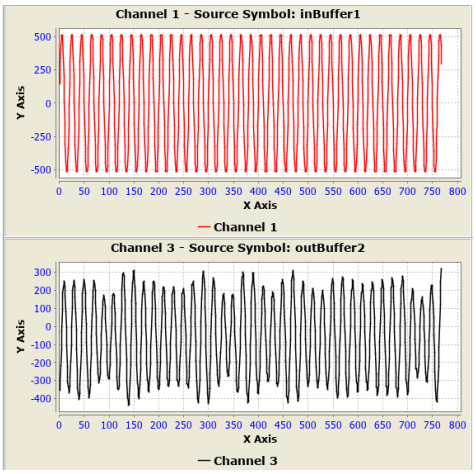
À l'aide de l'outil DCMI disponible pour MPlab, afficher les valeurs sous formes de graphiques les valeurs à l'entrée et à la sortie du filtre IIR.

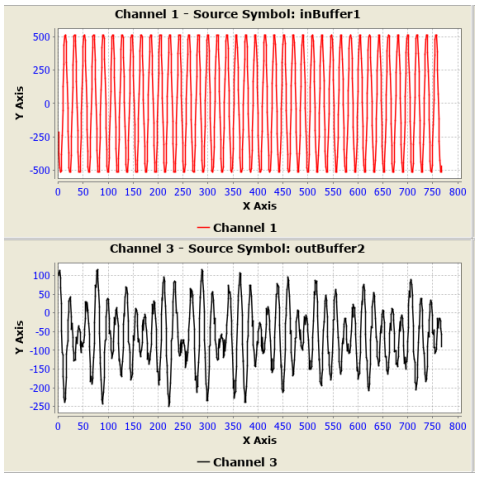
9.1.3 RÉSULTATS OBTENU



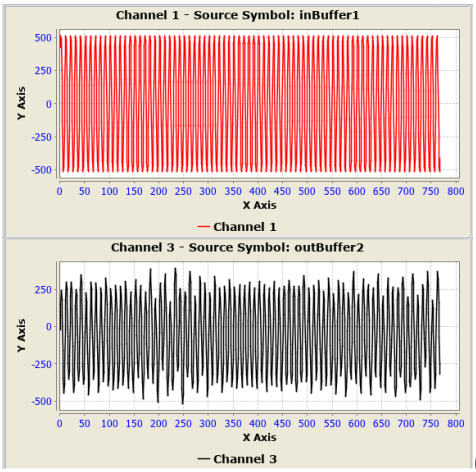


(500Hz) (950Hz)





(1000Hz) (1050Hz)



(2000Hz)