RAPPORT PROJET DSP

Ibrahima DIENG – Tarek AMRANI

MT5A

Tuteur : M. Aziza

Année scolaire : 2019-2020

PLAN

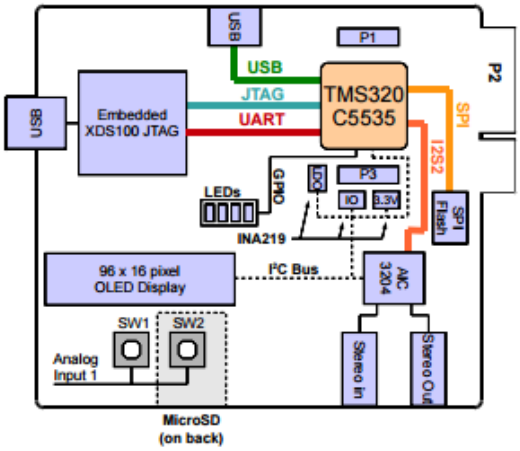
INTRODUCTION

Ce projet a pour but de nous permettre de prendre en main la carte DSP utilisée dans le cadre de cette série de TP avec le logiciel CCS.

TP1 : Prise en main de la carte eZDspC5535

Ce TP est composé de deux exercices qui nous permettent de prendre en main l'environnement de travail en première partie et en deuxième partie d'expliquer des programmes.

- **La carte eZdspC5535**



Titre : Carte DSPC5535

*1ᵉ Partie : Prise en main et mise en œuvre d'exemples d'applications pratiques

Exercice 1 : création d'un fichier de configuration matériel

Cet exercice consiste à suivre un canva déjà défini pour parvenir à connecter la carte eZdspC5535 et l'environnement de travail. A partir de là, on peut procéder aux tests des différents exemples de codes.

- Lancer **Code Composer Studio v7** (menu démarrer)
- Choisir un répertoire de travail à partir de « **Browse → Select Workspace Directory** » (pointer sur le disque **C:** et choisir de préférence **seance1** comme nom de « **workspace** »).
  Si vous possédez plusieurs répertoires de travail, le changement s'effectue en sélectionnant **File → Switch Workspace...**
- Sélectionner **File→New→ Target Configuration File**
- Enter un nom au fichier de configuration matériel (**C5535_eZdsp.ccxml**)
- Dans la fenêtre de sélection du fichier de configuration, choisir **Texas Instruments XDS100v2 USB Emulator** dans le menu **Connection**. Puis taper "5535" dans le champ **Device** et sélectionner **eZdsp5535** à partir de la liste filtrée
- Sauvegarder le fichier de configuration (bouton **save**)
- Tester le fichier de configuration (bouton **Test Connection**)
- Pour revenir au fichier de configuration, sélectionner **View→ Target Configurations** de manière à visualiser les éléments du fichier de configuration
- A ce stade, la connexion entre l'environnement de développement et la carte est validée.

Exercice 2 : Importation de projets CCS7

Explication des différents codes

Dans cette partie, nous avons importé des projets sur Ametice et les avons testés. La procédure est la suivante :

- A ce stade votre workspace doit contenir deux projet : **led** et **ezdsp5535bsl**
- Ouvrir le fichier **main.c** du projet led
- Sélectionner **Project→Build Project**. A la fin de la compilation un message s'affiche dans la console (fichier « led.out » généré correctement)
- Sélectionner **Run→Debug** : la vue « CCSdebug » se lance et le pointeur de programme se positionne au point d'entrée de votre projet. Pour passer de la vue « CCSdebug » à la vue « CCSEdit », utiliser les deux butons situés dans le coin en haut à droite de votre fenêtre
- Sélectionner **Run→Resume** pour lancer le programme. Les LEDs de la carte eZdsp se mettent à clignoter
- Pour revenir au début du programme utiliser la commande **Run→Restart**

Ainsi, nous avons testé les différents exemples et avons obtenus les résultats suivants :

✓ **Pour la Led :**

```
*****************************************
*       #include "stdio.h"
*       #include "ezdsp5535.h"
*       extern Int16 led_test( );
*       int  TestFail    = (int)-1;
*
*       void StopTest()
*       {
*         //SW_BREAKPOINT;
*         return;
*       }
*
*        main( )
*       void main( void )
*       {
```

```
*           /* Initialize BSL */
*            EZDSP5535_init( );
*
*          /* Display test ID */
*    printf( "\nTesting *LEDs...\n");

*  /* Call test function */
*    TestFail = led_test( );
*
*    /* Check for test fail */
*   if ( TestFail != 0 )
*    {
*      /* Print error message */
*       printf( "     FAIL... error *code %d...
quitting\n", TestFail );
*    }
*    else
*    {
*       /* Print pass message */
*            printf( "    PASS\n" );
*             printf( "\n***ALL Tests *
          Passed***\n" );
*           }
*           StopTest();
* }

****************************************
```

Explication code LED

- Le code pour la Led est composé d'une fonction StopTest() qui permet de contrôler le break-point et de la fonction principale main qui permet de contrôler l'affichage des Leds.
- Les Leds sont connectées avec le TMS320C5535 qui reçoit les instructions via la source USB

✓ Pour le Switch

```
#include "stdio.h"
#include "ezdsp5535.h"

extern Int16 switch_test( );

int  TestFail    = (int)-1;

void StopTest()
{
    //SW_BREAKPOINT;
    return;
}
 *  main( )
 */
void main( void )
{
    /* Initialize BSL */
    EZDSP5535_init( );

    /* Display test ID */
    printf( "\nTesting Switches...\n");

    /* Call test function */
    TestFail = switch_test( );

    /* Check for test fail */
    if ( TestFail != 0 )
    {
        /* Print error message */
        printf( "     FAIL... error code %d... quitting\n", TestFail );
    }
    else
    {
        /* Print pass message */
        printf( "    PASS\n" );
        printf( "\n***ALL Tests Passed***\n" );
    }

    StopTest();
}

```

Tableau d'analyse du projet CCS4

	LEDS	***SWITCH***	***LCD-OSD***	***Improvedaic3204***
Périphériques TMS320C5535	LEDs	GPAIN	96*16 pixels OLED	Aic3204 Stereo input output
Protocole de communication utilisé	GPIO	SPI	I2C	I2S2
Librairies / Fonctions utilisées	Led stdio eZdsp5535	stdio eZdsp5535	stdio eZdsp5535	Aic3204, PLL, Stdio, LedFlash, Stereo

Titre : Tableau d'analyse

TP2 : Aic3204 Audio Codes

- **1e Partie : Importation du projet audio de base sous CCS4**

Exercice 1 : Importation du projet ImprovedAic3204

L'objectif de cette partie est la mise en œuvre d'une application audio utilisant le codec aic3204. Nous importons le projet de base sous CCS auquel on ajoutera la fonctionnalité de filtrage souhaitée. Ce projet utilise le codec aic3204 dont on programmera l'entrée « stereo in » et sortie « stereo out ».

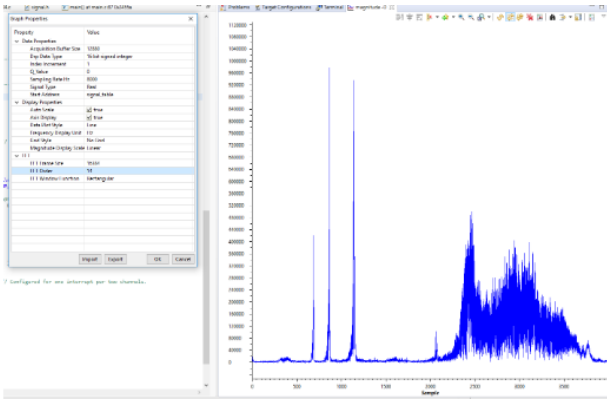
On vérifie le bon fonctionnement de ce programme en écoutant le signal de sortie qui est un mélange entre deux sons. En effet, on envoie un signal à l'entrée stereo in et grâce à la fonction read du codec on récupère le signal et on peut l'écouter. Il faut s'assurer aussi des paramètres d'entrée de ce programme qui sont :

- Le gain d'amplification du signal
- La fréquence d'échantillonnage définie à 48000 Hz
- La fréquence de la PLL à 100 Hz

- **2e partie : Implantation d'un filtre numérique**

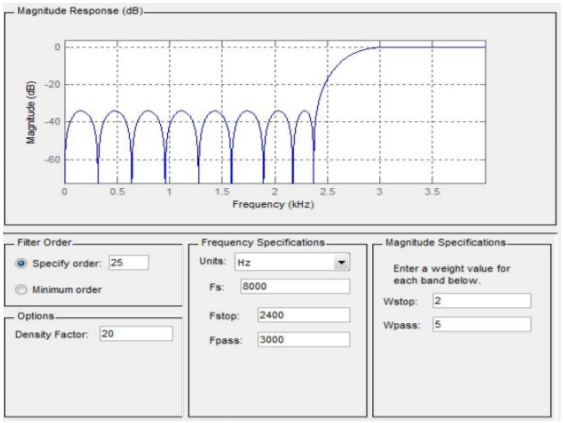
Exercice 1 : Définition du gabarit du filtre (logiciel Matlab) et extraction de la réponse impulsionnelle

L'objectif est de réaliser un filtre qui permet d'extraire un signal à partir de deux signaux mélangés. Définition du gabarit du filtre (logiciel Matlab) et extraction de la réponse impulsionnelle, après implantation du filtre sur cible DSP et réalisation de la fonction de filtrage en langage C. Un signal numérique est une suite de nombres, obtenue par exemple par échantillonnage d'un signal analogique. Une fréquence d'échantillonnage est généralement associée au signal numérique. Le filtrage numérique est un traitement effectué sur cette suite de nombres, pour transformer le signal. Ce calcul peut être fait sur un ordinateur à partir de signaux stockés en mémoire. Dans les circuits d'électronique numérique, il est fait en temps réel par des microprocesseurs spécialisés (DSP Digital Signal Processor), que l'on rencontre par exemple dans les téléphones portables pour le traitement du son. Tout d'abord nous avons utilisé le logiciel Matlab et son application « signal analysis » pour visualiser les signaux ainsi que leur spectre.



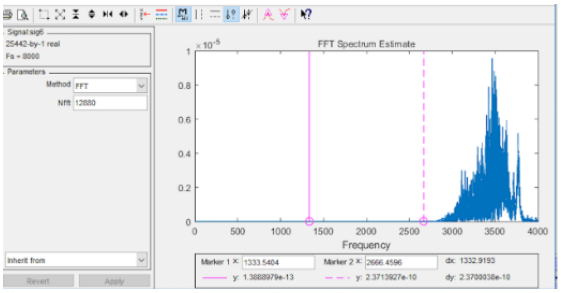
Titre : signal train et oiseau

Après, il faut définir un filtre afin de récupérer que le signal de l'oiseau (de haute fréquence). À partir de la fenêtre 'Filter Designer & Analysis Tool ', on a définir un filtre FIR, c'est un filtre passe-haut d'ordre 25. En dessous, nous avons une photo de ses différentes caractéristiques. Après l'optimisation, on a défini le Fstop est 2400Hz et Fpass 3000Hz.



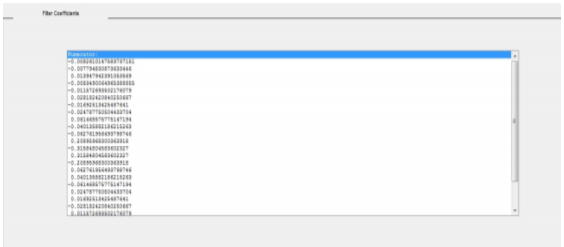
Titre : Donnée du filtre

Les spectre du signal après l'application du filtre. On remarque que le signal de bruit est presque atténué d'où le bon fonctionnement du filtrage. On peut vérifier l'élimination du signal bruit en écoutant le signal de sortie filtré.



Titre : Signal obtenu après filtre

Pour récupérer les coefficients du filtre, on utilise la commande « Générate C Header ».On utilise ces coefficients dans l'exercice suivant pour filtrer le signal sous CCS4.

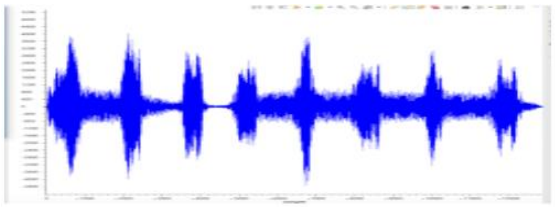


Titre : coefficients du filtre

Exercice 2 : Implantation du filtre sur cible DSP et réalisation de la fonction de filtrage en langage C

Pour implanter le filtre sur DSP, nous avons simplifié le projet ImprovedAic3204 de

manière à ne garder que la fonctionnalité audio et allégé le code en supprimant la partie liée aux LED. Nous avons modifié le programme de manière à envoyer le signal contenu dans le tableau de données signal_table (signal locomotive + oiseau) du fichier « signal.h » sur la sortie ligne en utilisant le périphérique de liaison série I2S et le périphérique audio aic3204. Nous avons visualisé le signal mélangé en temporelle. Le résultat obtenue sous CCS4 est presque le même que Matlab.



Titre : signal temporel sous Matlab

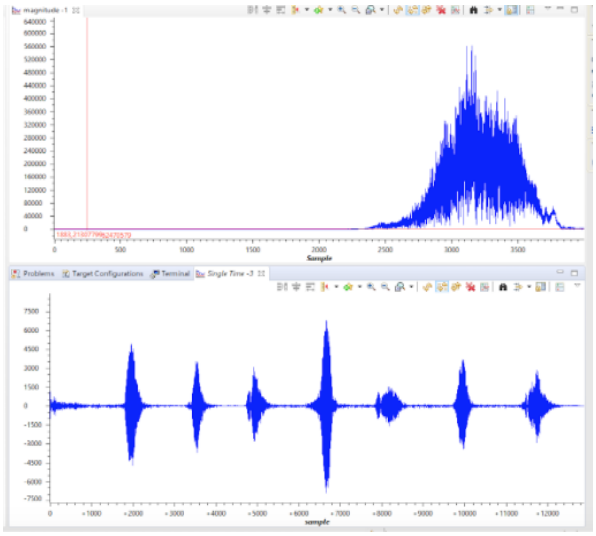
Pour réaliser la fonction de filtrage en langage C, on doit utiliser les coefficients qu'on a reçu par Matlab. On a créé un tableau tabcoeff[] qui contient les coefficients du filtre, et un tableau signal_table[] qui contient le signal mélangé numérique. Où la taille de tabcoeff[ ] est 23(l'ordre du filtre), et la taille de signal_table[ ] est 12880.Un produit de convolution entre le signal mélangé signal_table[] et les coefficients du filtre tabcoeff[] devra être implémenté en langage C.

Nous avons filtré le signal mélangé en faisant la convolution entre le signal x(n) et les coefficients du filtre h(k). Un produit de convolution entre le signal mélangé x(n) et les coefficients du filtre h(k). N étant l'ordre du filtre. Les coefficients du filtre h(k) sont obtenus dans l'exercice 2. Ce qui nous permet d'avpoir le code suivant en C :

```
/* New. Default to XF LED off */
asm(" bclr XF");

for ( i = 0  ; i < SAMPLES_PER_SECOND * 600L  ;i++  )
{


  printf( "\n***Program has is looping**\n" );
/*for(k=0; k<25; k++)
{       if ( n<k) k = 25;
        else
        y[n]=y[n]+signal_table[n-k]*B[k] ;
        }*/

// aic3204_codec_read(&left_input,&right_input); // Configured for one interrupt per two channels

//  mono_input = stereo_to_mono(left_input, right_input);

    /* New. LED flasher */



     left_output =signal_table[n] ;    // Stereo to mono
     right_output = signal_table[n] ;
     n++;
     if(n==12881) n=0;

 aic3204_codec_write(left_output, right_output);
   }

/* Disable I2S and put codec into reset */
 aic3204_disable();

 printf( "\n***Program has Terminated***\n" );
 SW_BREAKPOINT;
```

Titre : fonction de convolution en C

Après avoir écouté le signal filtré, on trouve qu'il n'y a pas de bruit locomotif, il est bien filtré. Et ensuite nous avons visualisé le signal filtré en temporelle et fréquentielle, selon la figure ci-dessous, on peut voir il reste que le signal de haut fréquence. Il est pareil du signal qu'on a reçu par Matlab. Ce résultat est bon.



Titre : signal filtré en temporel et fréquentiel

- **3e Partie : Réalisation de la fonction de filtrage en langage Assembleur**

L'objectif de cette partie est de développer un code permettant de filtrer un signal audio en temps réel. Pour cela nous ajoutons la fonction aic3204_codec_read, permettant entre autres de récupérer le signal audio.

On met en forme le signal audio à envoyer à l'aide de Matlab pour qu'on puisse le plus possible le prolonger dans le but de pouvoir l'écouter. Pour ce nous importons sous Matlab

le fichier signalMatlabnorm.txt qui représente une suite de nombres signés. Les commandes utilisées sous Matlab pour prolonger ce signal sont :

signalMatlabnorm_mat= table2array(signalMatlabnorm)

signalMatlabnorm_mat_long= table2array(signalMatlabnorm_mat)

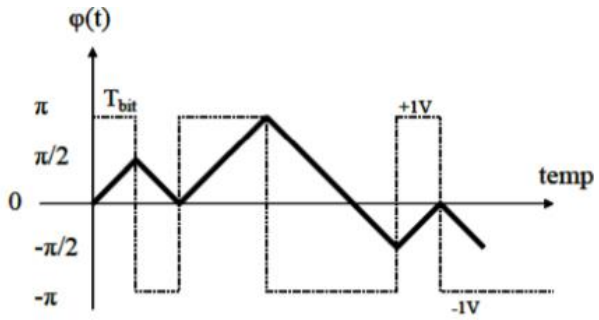
et pour l'écouter:

sound(signalMatlabnorm_mat_long,8000)

TP3 : Projet DSP La Modulation MSK

- **1e Partie : Principe de modulation MSK**

Pendant la durée d'un bit, la phase évolue linéairement avec une pente positive ou négative suivant la valeur du bit, et prend à la fin de la transmission la valeur particulière de $\pm\pi / 2$.



Titre : évolution phase porteuse en MSK modulation

- **2e Partie : Implantation sur DSP**
  - **Exercice 1**

Même procédure que le projet ImprovedAic.

Exercice 2

Tout d'abord, nous avons créé un tableau décrivant le signal numérique d'entrée qui se compose du train de bit suivant :

{1,-1,-1,1,1,1,1,-1,1,-1}.

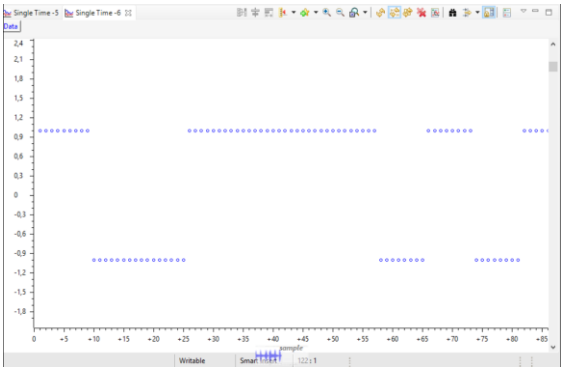
La fréquence d'échantillonnage est de 8kHz et Tbit = 8*T0. La ligne ajoutée pour créer le train de bits est la suivante :

signed train_bit[10]={1,-1,-1,1,1,1,1,-1,1,-1};

Le signal tain bits est généré et constituée du signal échantillonné comme suit :

```
void main( void )
{
    /* Initialize BSL */
    USBSTK5505_init( );

    /* Initialize PLL */
    pll_frequency_setup(100);

    /* Initialise hardware interface and I2C for code */
    aic3204_hardware_init();

    /* Initialise the AIC3204 codec */
    aic3204_init();

    printf("\n\nRunning Improved Audio Template Project\n");
    printf( "<-> Audio Loopback from Stereo IN --> to HP/Lineout\n\n" );

    /* Setup sampling frequency and 30dB gain for microphone */
    set_sampling_frequency_and_gain(SAMPLES_PER_SECOND, GAIN_IN_dB);

    /* New. Default to XF LED off */
    asm(" bclr XF");

    for ( i = 0  ; i < SAMPLES_PER_SECOND * 600L  ;i++  )
    {
       sign= train_bit[j];

       if(k<8) k++;
       else{
           k=0;
           if(j<9) j++;
           else j=0;
           }

        aic3204_codec_write(sign, sign);
```

Après génération, on obtient le signal suivant :



Titre : Signal obtenu d'après la description en C ci-dessus

Pour le calcul de la phase, on a pas réussi à le faire durant le dernier TP de même que la detérmination des composants TXI et TXQ.

Conclusion

Cette série de TP nous a permis de prendre en main l'outil CCS pour comprendre le fonctionnement des DSP. A travers les différents TP, on a eu à étudier les différents codes et aussi à implémenter des codes en C et Assembleur pour effectuer des taches précises comme la modulation MSK lors du dernier TP.