Jennifer Salle & Samuel Rollet

Projet d'Architecture - Première Année

Rapport Final

Accordeur de Guitare

Ecole Supérieure d'Ingénieurs de Luminy
Université de la Méditerranée - Aix Marseille II
Mars 2006

Licence

ESIL TUNER

by Samuel Roller srollet@esil.univ-mrs.fr
and Jennifer Salle jsalle@esil.univ-mrs.fr

Program developed for an MSP430F1222 Texas Instrument ship as a tuner for guitars.

Table des Matières

1. INTRODUCTION 6
1.1 Objectif 6
1.2 Contexte 6
1.3 Planning Initial 7
2. CONCEPTION 8
2.1 Description matériel 8
2.1.1 Circuit Détaillé 8
2.1.2 Filtre MAX7400 10
2.1.3 Amplificateur MAX4462 de Texas instrument 10
2.1.4 Microcontrôleur MSP430F122 de Texas Instrument 11
2.1.5 Afficheur LCD SP5 GFX1 avec Contrôleur Sunplus 12
3. CONCEPTION 13
3.1 Organigrammes 13
3.1.1 Utilisation de l'Accordeur 13
3.1.2 Séquences de Fonctionnement 14
3.2 Fonctionnement de l'Afficheur 15
3.2.1 Données Affichées 15
3.2.2 Envois des Données à l'Afficheur 15
3.3 Analyse de Fréquence 17
3.3.1 Calcul de la Fréquence 17
3.3.2 Programmation du Filtre 18
3.4 Gestion de l'Energie 18
3.5 Développement 20
4. RÉALISATION 21
4.1 Structure des Fichiers et Données Globales 21
4.2 Algorithmes et Données 22
4.2.1 Programmation de l'Afficheur 22
4.2.2 Programmation du Filtre 23
4.2.3 Analyse de Fréquence 23
4.2.4 Utilisation du Timer 24
4.2.5 Détection en Mode Automatique 24
4.2 Description des fichiers 25
4.2.1 Fichier main.c 25
4.2.2 Fichier lcd.c 29
4.2.3 Fichier lcd.h 34
4.2.4 Fichier leds.c 35
4.2.5 Fichier leds.h 36
4.2.6 Fichier misc.c 37
4.2.7 Fichier misc.h 38
5. RÉSULTATS 39
5.1 Problèmes Rencontrés 39

5.1.1 Problème de Batteries 39
5.1.2 Vitesse d'Exécution 39
5.1.3 Limitation de la Mémoire 39
5.1.5 Hystérésis 40
5.2 Signaux 41
5.2.1 Signal Source 41
5.2.2 Signal Amplifié 41
5.2.3 Signal Filtré 42
5.2.4 Bruit sur le Signal 42
5.2.5 Horloge de Programmation du Filtre 43
5.2.6 Résultat Observé 43
6. CONCLUSION 44
Références 45
Table de fréquence des notes 46
Code Sources 47

Table des Figures

Figure 1 - Enchaînement des Différentes Tâches du Projet....7
Figure 2 - Schématisation du Circuit....8
Figure 3 - Détail du Circuit....9
Figure 4 - Filtre MAX7400....10
Figure 5 - Exemple d'amplification....10
Figure 6 - Désignation des broches du MSP430F122....11
Figure 7 - Bloc Fonctionnel du MSP430F122....11
Figure 8 - Interfaçage entre le contrôleur SPLC 501C et l'afficheur SP5 GFX1....12
Figure 9 - Cas d'Utilisations....13
Figure 10 - Diagramme d'Etat....14
Figure 11 - Séquences de commandes....16
Figure 12 - Calcul de la Période....17
Figure 13 - Réponse du Filtre....18
Figure 14 - Consommation d'Énergie du Microcontrôleur....19
Figure 15 -Signal Source....41
Figure 16 - Signal Amplifié....41
Figure 17 - Signal Filtré....42
Figure 18 - Bruit sur le Signal....42
Figure 19 - Horloge de Programmation du Filtre....43
Figure 20 - Résultat Observé....43

1. INTRODUCTION

1.1 Objectif

Les guitares sont des instruments à corde nécessitant d'être accordés régulièrement. L'objectif de ce projet était de réaliser la programmation d'un accordeur de guitare sur le circuit fourni par notre client. Ce circuit comporte entre autre un microphone, un microcontrôleur et un afficheur graphique. L'objectif était d'effectuer la programmation du microcontrôleur afin de mesurer précisément la fréquence d'une note jouée et d'afficher cette information pour permettre d'accorder une guitare. Ce projet est désigné sous le nom de ESIL Tuner.

1.2 Contexte

Dans le cadre du projet d'architecture de première année du département informatique de l'Ecole Supérieure d'Ingénieur de Luminy, nous avons réalisé cet accordeur de guitare qui nous a permis de mettre en œuvre les connaissances que nous avons acquises lors des cours et des TP. Pour cela nous avons travailler en binôme, celui-ci se constitué de Jennifer Salle et Samuel Rollet. L'enseignant chargé d'encadrer ce projet était M. Bernard Dinkespiler.

1.3 Planning Initial

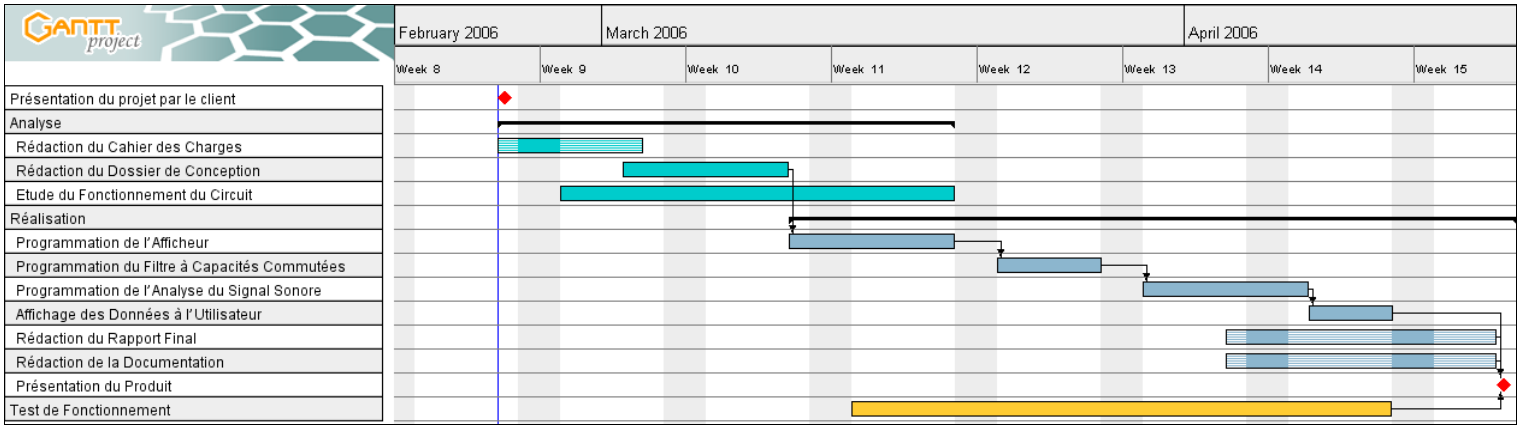


Figure 1 - Enchaînement des Différentes Tâches du Projet

La réalisation de ce projet s'est divisée en deux principales phases. Une première phase d'analyse du problème, d'apprentissage et d'étude du fonctionnement des composants sur lesquelles nous avons travailler. Une seconde phase de réalisation des objectifs fixés dans le cahier des charges. Cette seconde phase s'est divisée entre les différents éléments à réaliser pour obtenir le produit finit :

- Programmation de l'afficheur.
- Programmation du filtre à capacités commutées.
- Programmation de l'analyse du signal sonore.
- Affichage des données à l'utilisateur.

L'objectif étant d'acquérir des connaissances sur le fonctionnement de l'ensemble du produit durant le développement, l'ensemble des tâches a été réalisé en binôme pour favoriser une meilleur compréhension du produit fini. Les éventuelles parties développées individuellement ont fait l'objet d'une explication de la personne en charge à l'autre membre de l'équipe.

2. CONCEPTION

2.1 Description matériel

2.1.1 Circuit Détaillé

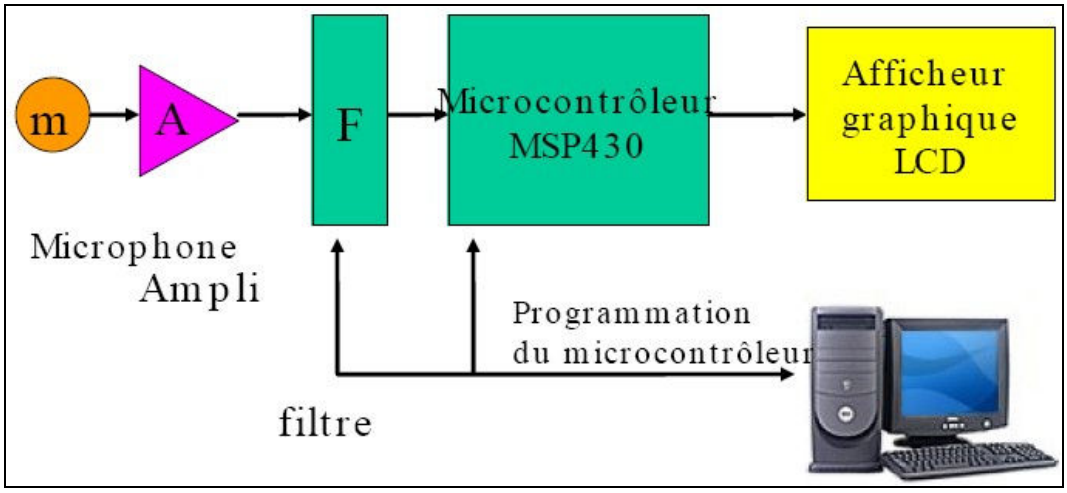


Figure 2 - Schématisation du Circuit

Le schéma ci-dessus représente le montage simplifié du circuit fournit par notre client. Il intègre :

- Un microphone.
- Un amplificateur.
- Un filtre MAX7400.
- Un microcontrôleur MSP430F122 de Texas Instrument.
- Un afficheur LCD SP5 GFX1 avec contrôleur Sunplus.

Le principe d'un accordeur est de capter la note jouée grâce au microphone, d'analyser le signal reçut et d'afficher les résultats à l'utilisateur.
Le son joué à la guitare est captée par le microphone sous forme d'un signal analogique. Il est ensuite dirigé sur l'amplificateur afin d'augmenter l'amplitude du signal par cent. Le signal amplifié est ensuite dirigé vers le filtre et sur une entrée du microcontrôleur. Le signal filtré est ensuite émis au microcontrôleur qui effectue l'analyse de la note jouée. Les résultats sont ensuite envoyés à l'afficheur LCD afin que l'utilisateur puisse accorder son instrument.

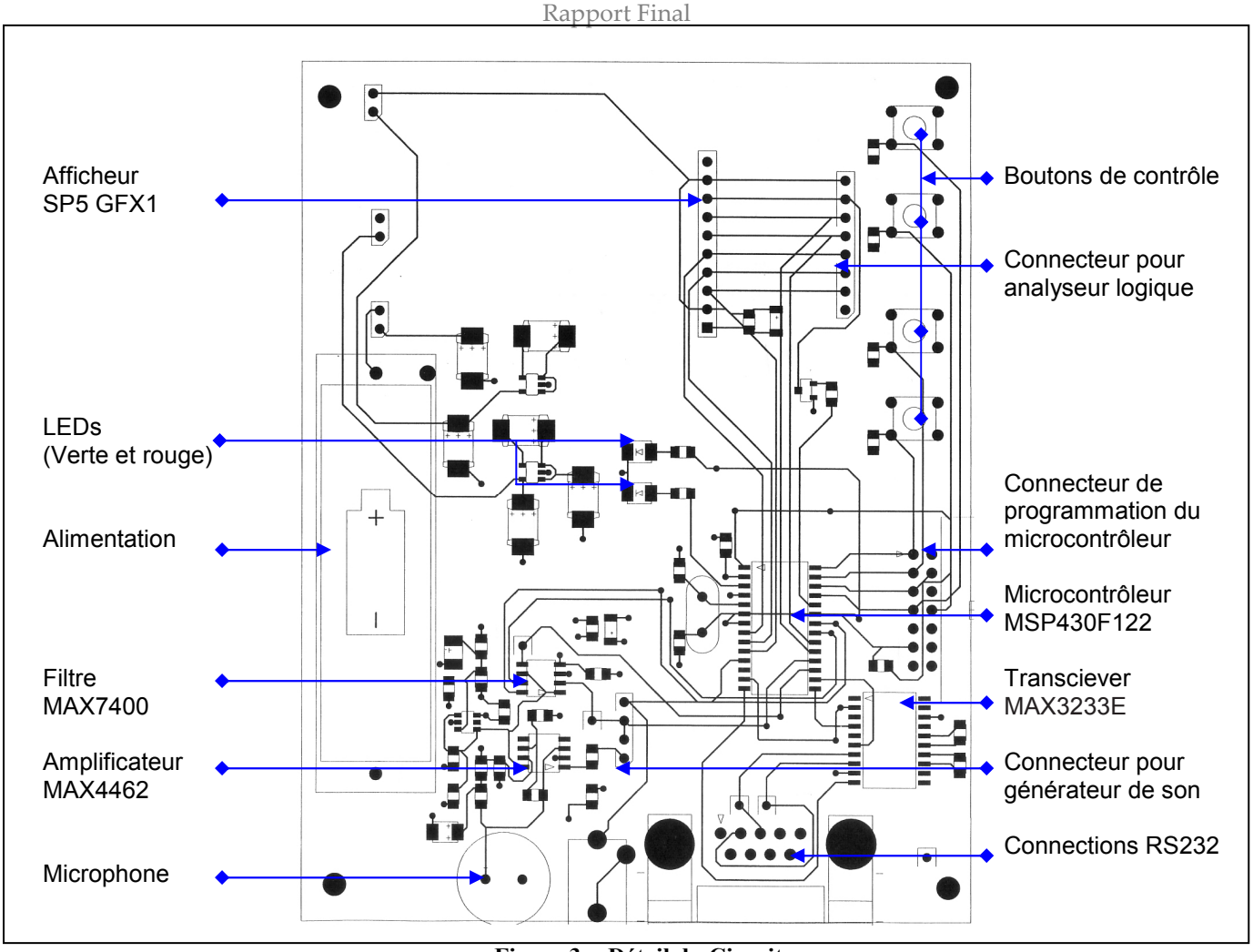


Figure 3 - Détail du Circuit

Le circuit offre d'autres composants, une partie destinée au développement, une autre au fonctionnement de l'accordeur.

Interface de développement :

- Un connecteur permettant la programmation du microcontrôleur par l'intermédiaire d'une sonde permettant le branchement sur un port parallèle d'un PC.
- Un connecteur RS232 permettant le branchement sur un PC pour consulter l'échantillonnage pendant le développement.
- Un connecteur permettant le branchement d'un analyseur logique pour tester le fonctionnement de l'afficheur.
- Un connecteur permettant le branchement d'un générateur de sons afin de tester le fonctionnement correct de l'accordeur sans utiliser le micro.

Interface d'utilisation :

- Quatre boutons poussoirs utilisés pour configurer l'accordeur (choix du mode, de la note…)
- Deux LED indiquant la justesse de la note jouée.

2.1.2 Filtre MAX7400

Le filtre MAX7400 est un filtre du 8ème ordre, passe bas.
Il supporte des fréquences d'entrée allant de 1Hz à 10kHz. Cela englobe la quasi-totalité des notes possibles pour un instrument (cf. Liste des fréquences des notes en annexe).
Il fonctionne sur une alimentation unique de 5V et consomme 2mA.
Il offre un mode de veille qui réduit sa consommation d'énergie à 0.2µA.
Le choix de la fréquence à filtrer se fait par le réglage de l'horloge en entrée.

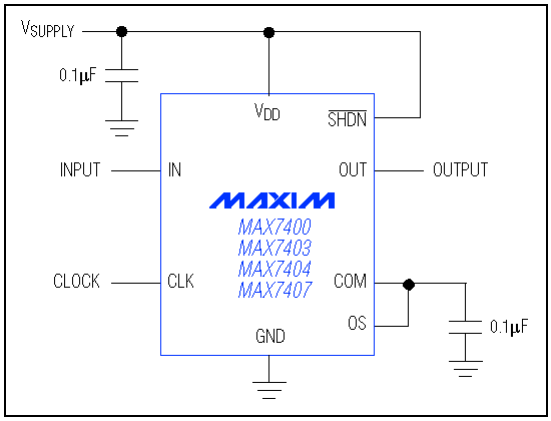


Figure 4 - Filtre MAX7400

2.1.3 Amplificateur MAX4462 de Texas instrument

L'amplificateur MAX4462 est un amplificateur de haute précision avec une faible consommation d'énergie. Il comporte une entrée contrôlant son activation et son extinction. Il offre un gain de 100, c'est-à-dire que 10mV en entrée deviennent 1V en sortie. L'erreur du gain des de plus ou moins 0.1%.

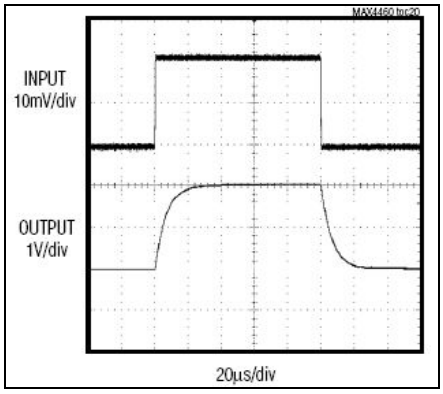


Figure 5 - Exemple d'amplification

2.1.4 Microcontrôleur MSP430F122 de Texas Instrument

Le microcontrôleur basse consommation MSP430F122 regroupe différents éléments offrant une série de périphériques pour différentes applications.
L'architecture, combinée avec cinq modes basse consommation est optimisée pour maximiser la durée de vie des batteries d'appareils portables.
Il intègre un processeur RISC 16bits avec un générateur de valeurs constantes, pour maximiser l'efficacité du code.
Le DCO, oscillateur contrôlé numériquement, permet le réveille des modes basse consommation en moins de 6µs.
Il intègre un timer de 16bits, et 22 broches d'entrée sortie.
Il a aussi des capacités de communication intégrées utilisant des modes synchrones (SPI) et asynchrones (UART).

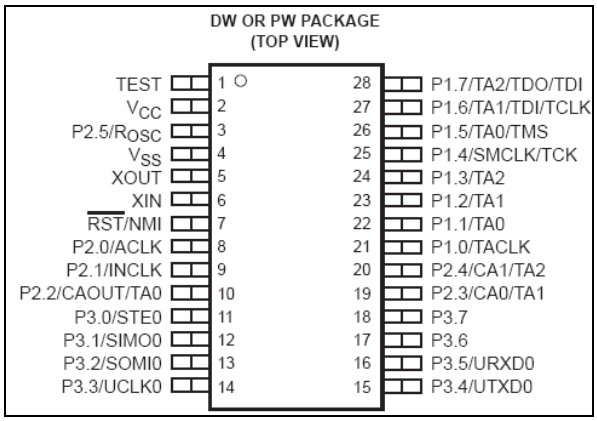


Figure 6 - Désignation des broches du MSP430F122

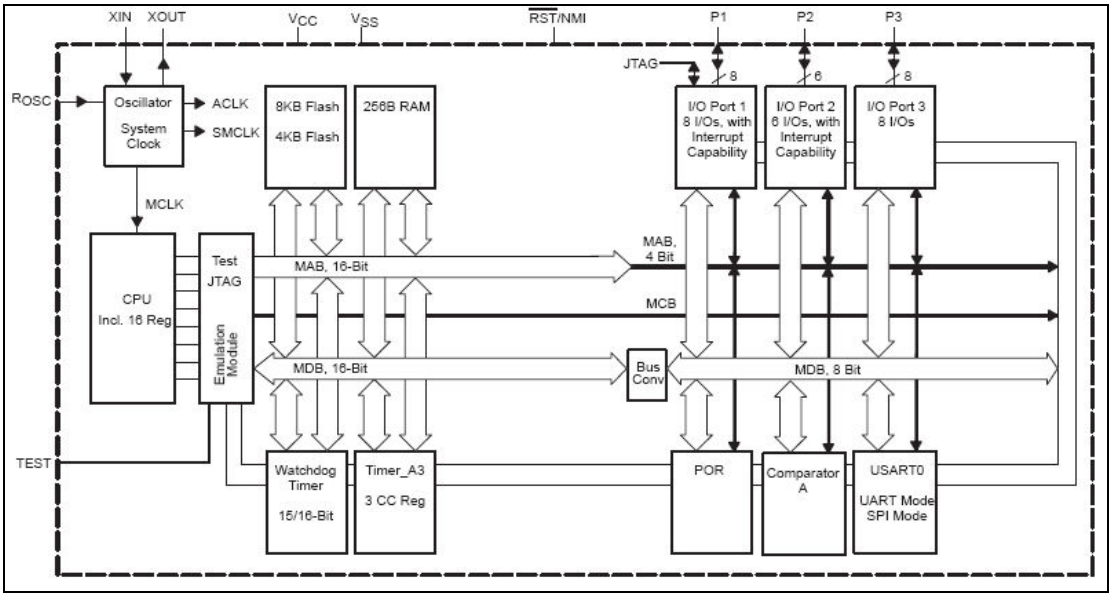


Figure 7 - Bloc Fonctionnel du MSP430F122

2.1.5 Afficheur LCD SP5 GFX1 avec Contrôleur Sunplus

Le SP5 GFX1 intègre le dernier contrôleur LCD LSI pour fournir une combinaison unique de 128x64 pixels compacte et facile à utiliser. Un rétro éclairage assure une excellente visibilité dans un environnement peu éclairé.
Il offre un connecteur en ligne de 10 broches. L'interface série assure que le microcontrôleur puisse contrôler l'affichage avec un minimum de lignes d'entrée sortie.

Caractéristiques :

- 128x64 pixels.
- LCD à contrastes élevés.
- Visibilité optimale sans angle de vision.
- Rétro éclairage jaune et vert.
- Basé sur un contrôleur Sunplus SPLC 501C.
- Interface de données série.

Le contrôleur Sunplus SPLC 501C est un contrôleur d'afficheur à cristaux liquides destiné à être connecté directement avec un microprocesseur. Il convertis l'entrée 8-bit du microprocesseur en un signal de contrôle de l'afficheur à cristaux liquides. Il contient une RAM de 65x132bits ce qui permet une correspondance 1 à 1 avec les pixels de l'afficheur.

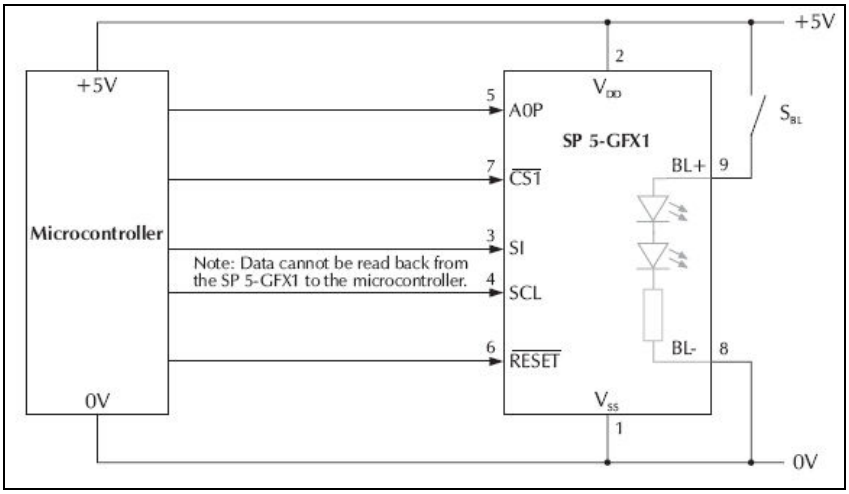


Figure 8 - Interfaçage entre le contrôleur SPLC 501C et l'afficheur SP5 GFX1.

3. CONCEPTION

3.1 Organigrammes

3.1.1 Utilisation de l'Accordeur

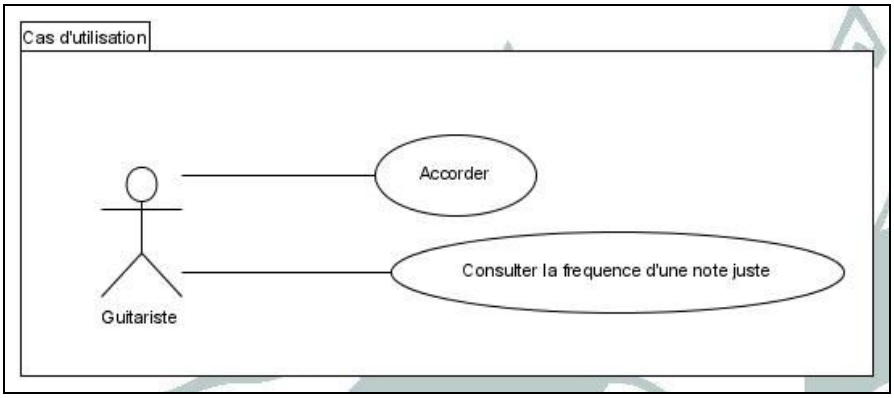


Figure 9 - Cas d'Utilisations.

L'accordeur offrira une principale fonction à l'utilisateur : accorder les cordes de sa guitare. Il pourra consulter sur l'afficheur la fréquence de la note jouée, et un diagramme lui affichera la distance relative entre la note jouée et la note juste.
De plus il lui permettra de consulter la fréquence d'une note juste sur l'afficheur.

3.1.2 Séquences de Fonctionnement

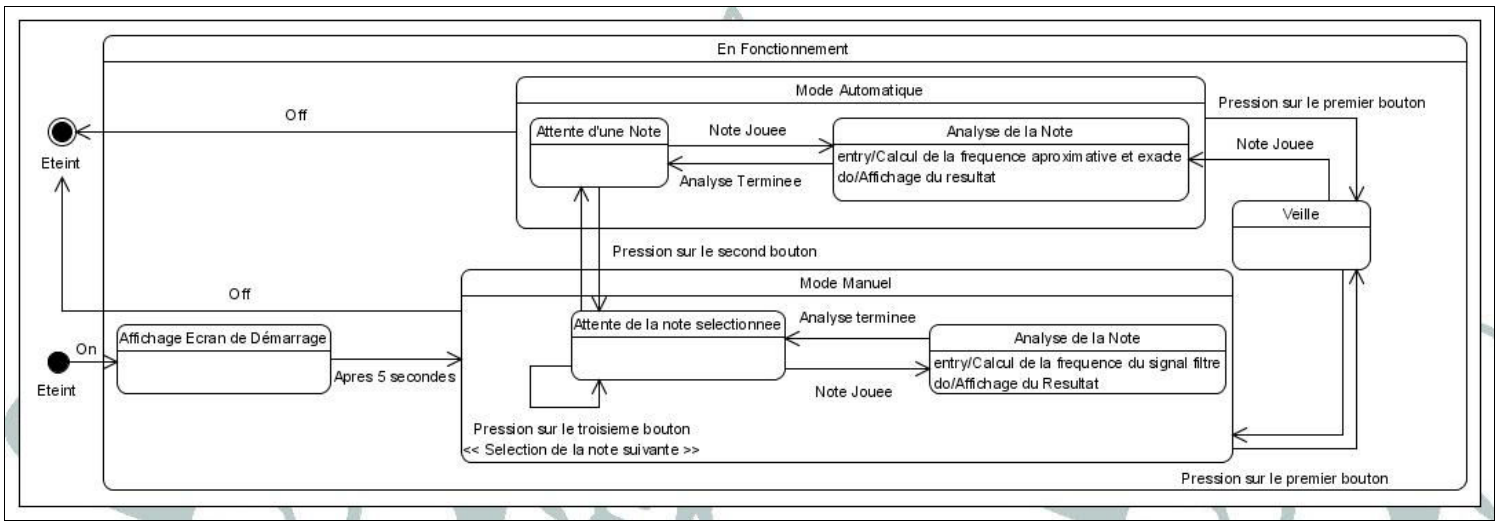


Figure 10 - Diagramme d'Etat

L'allumage de l'afficheur s'effectue par une pression sur le bouton On (bouton numéro 4). Pendant cinq secondes, l'accordeur affiche le nom du programme ; à ce moment la, il n'est pas capable de détecter la note jouée. Après ces cinq secondes, il démarre en mode manuel et se met en attente d'une note. La note attendue par défaut et le Mi (82.3Hz).
Si l'utilisateur presse le troisième boulon, il peut changer la note à accorder. Le choix des notes est cycliques : Mi (82.4Hz), La (110), Ré (146.8Hz), Sol (196Hz), Si (246.9Hz), Mi (329.6Hz). Chaque pression sur le troisième bouton fait changer la note attendue. Si une note est jouée, l'analyse de la note débute : on calcule sa fréquence suivant la méthode expliquée plus loin, ensuite, le résultat est affiché à l'écran : la fréquence de la note jouée est affichée et le graphique affiche la distance entre cette fréquence et la note juste.
Si l'utilisateur presse le second bouton il provoque un changement de mode. Chaque pression fait passer du mode manuel au mode automatique et vice-versa.
En mode automatique, l'accordeur attend une note, quelle qu'elle soit. Lorsqu'une note est jouée : on commence par calculer une fréquence approximative afin de déterminer la note juste la plus proche. Ensuite on calcule la fréquence exacte, et on affiche les résultats suivant le même modèle qu'en mode manuel.
L'utilisateur a aussi la possibilité de mettre son accordeur en veille en effectuant une pression sur le premier bouton.

3.2 Fonctionnement de l'Afficheur

3.2.1 Données Affichées

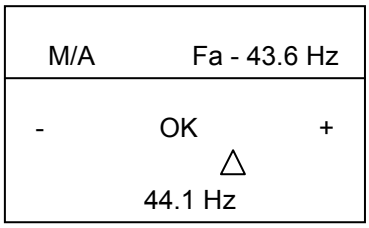
L'afficheur est un afficheur LCD SP5 GFX1 avec un contrôleur Sunplus SLPC501C. Il affiche le nom du produit au démarrage, puis les informations nécessaires à l'accordage de la guitare, comme décrit ci-après.

Au démarrage :

- Affichage du nom du produit pendant cinq secondes.

En fonctionnement :

- Affichage du mode : Automatique ou Manuel.
- Affichage de la note jouée (choisie par l'utilisateur en mode manuel, détecté en mode automatique).
- Affichage de la fréquence attendu et de la fréquence de la note jouée.
- Affichage d'un graphique indiquant la différence de fréquence entre la note juste et la note jouée :
    - Curseur au centre : note juste.
    - Curseur vers la gauche : note trop basse.
    - Curseur vers la droite : note trop haute.



Aperçu de l'affichage.

3.2.2 Envois des Données à l'Afficheur

Le contrôleur Sunplus est connecté en série au contrôleur MSP430 du circuit. Les signaux de commande pour l'afficheur sont envoyés par le programme du microcontrôleur MSP430. Les principaux signaux nécessaires sont :

- *Display On/Off* : pour allumer ou éteindre l'afficheur
- *Display Start Line Set* : pour sélectionner l'adresse où on veut écrire en mémoire au démarrage.
- *Page Address Set* et *Column Address Set* : pour sélectionner la position d'un pixel.
- *Display Data Write* : pour écrire 8bits de donnée dans la mémoire.
- *Display All Point On/Off* : pour forcer l'affichage de tous les pixels.

Ce sont les principales commandes nécessaires pour l'affichage de données sur l'écran. Le schéma ci-après montre comment elles sont enchaînées pour écrire une donnée en mémoire.

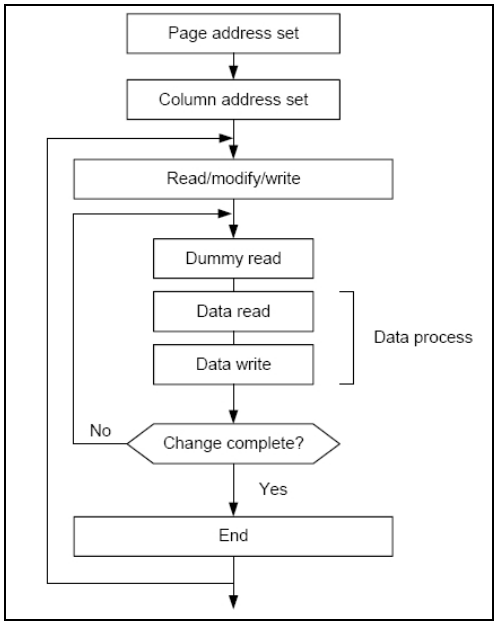


Figure 11 - Séquences de commandes.

Pour faciliter l'affichage des données par l'accordeur nous développeront une librairie de fonctions. Chaque fonction sera chargée de l'affichage d'un élément : ligne, mode, fréquence, note, schéma.

3.3 Analyse de Fréquence

3.3.1 Calcul de la Fréquence

La fréquence du signal se calcule à l'aide de la formule suivante : $f = 1/T$
Il faut préalablement calculer la période.

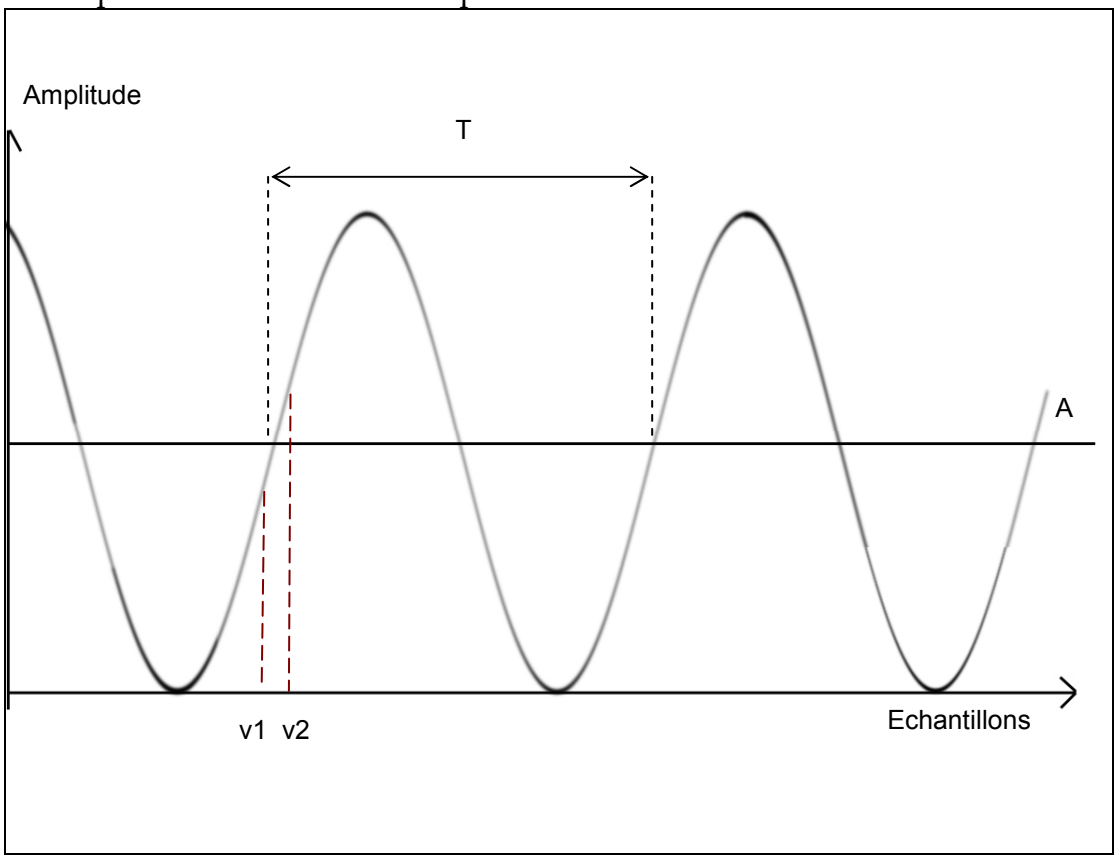


Figure 12 - Calcul de la Période

Pour la recherche de la période, nous avons décidé d'analyser celle-ci par rapport au milieu de la courbe plutôt que de prendre la période au deux crêtes consécutifs. En effet, le maximum de la courbe peut être dure à repérer car il peut être différent pour chaque crête. Ainsi, la période sera détectée lorsque entre deux échantillonnages consécutifs, la courbe passe d'une valeur inférieure à A à une valeur supérieure.
Toutefois, étant donné le bruit et pour éviter des erreurs de détection, on effectue une hystérésis. On teste le passage en deux valeurs, v1 et v2, l'une plus petite que A et l'autre plus grande, afin de déterminer le passage effectif par A. Grâce à cette méthode, on détecte le début et la fin d'une période, ainsi que sa durée.

Afin d'obtenir une meilleure précision, nous avons choisi de calculer la durée de plusieurs périodes. Par la suite, on calcule la moyenne pour obtenir le résultat.

3.3.2 Programmation du Filtre

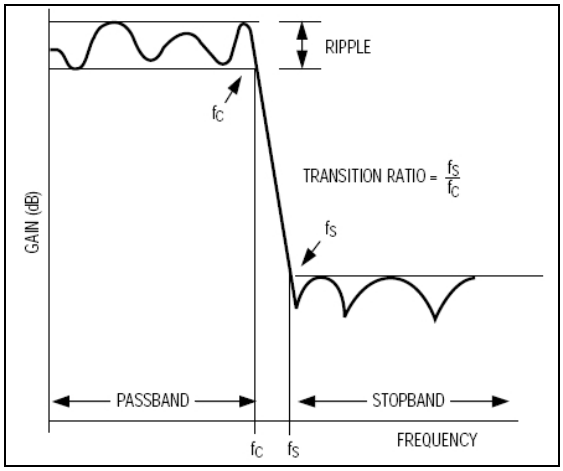


Figure 13 - Réponse du Filtre

Afin d'effectuer une analyse de fréquence correcte, il est nécessaire de filtrer le signal reçu par le microphone. On utilise pour cela le filtre MAX7404 présenté plus haut. Afin de définir la fréquence de rejet (f_c), on fait varier la fréquence de l'horloge connectée sur l'entrée *CLK*. L'horloge externe doit avoir un rapport co-cyclique entre 40% et 60%.

$$f_c = f_{CLK} / 100$$

La fréquence de cette horloge sera programmée par le microcontrôleur en fonction de la note qui doit être accordée.

3.4 Gestion de l'Energie

L'accordeur peut avoir plusieurs états : en marche, en veille et à l'arrêt. Cela afin d'économiser l'énergie et d'augmenter la durée de vie de la batterie.

Pour implémenter ces fonctions on utilisera les modes de basse consommation ou de veille des différents composants du circuit.

Pour le mode à l'arrêt, la quasi-totalité des composants sera éteint.

- L'alimentation de l'afficheur et de son contrôleur est suspendue.
- L'alimentation du filtre et des amplificateurs est aussi suspendue.
- Le microcontrôleur est en mode basse consommation 4 (LPM4).

Le CPU, les horloges, le DCO et les oscillations du quartz sont arrêtés afin de réduire au maximum la consommation. En mode basse consommation, le microcontrôleur peut être réveillé par une interruption. Une pression sur le quatrième bouton quand l'accordeur est éteint générera une interruption qui

active le microcontrôleur qui se charge d'alimenter l'ensemble des composants et démarre l'accordeur.

Pour le mode veille :

- L'afficheur est en mode Sleep : l'horloge, l'alimentation du LCD et tous les cristaux liquides sont arrêtés.
- Le filtre et les amplificateurs restent alimentés pour détecter si une note de musique est jouée afin de réactiver l'accordeur.
- Le microcontrôleur est en mode basse consommation 3 (LPM3).

L'horloge du filtre (ACLK) est le seul composant du microcontrôleur qui reste activé. En effet, elle est nécessaire au bon fonctionnement du filtre. Si une note est jouée ou une pression sur le premier bouton est effectuée par l'utilisateur, une interruption vient réveiller le microcontrôleur.

En mode de marche tous les composants sont complètement alimentés. L'afficheur affiche les informations courantes, le filtre est prêt à recevoir un signale, et le microcontrôleur est en mode actif (AM).

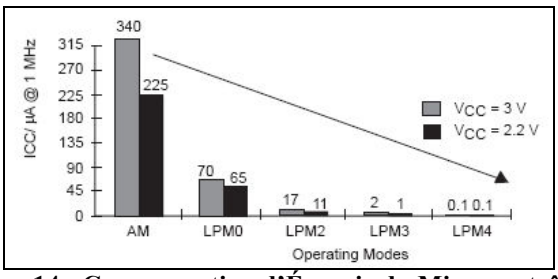


Figure 14 - Consommation d'Énergie du Microcontrôleur

On observe ici les différences de consommation d'énergie du microcontrôleur. En mode LPM3 et LPM4 la consommation est quasiment nulle, on les utilise pour limiter la consommation de l'accordeur.

3.5 Développement

L'environnement de développement que nous avions prévu d'utiliser au départ est le Code Composer Essentials for MSP430 v.1.0, le langage utilisé est le langage C, étendu par les bibliothèques *"MSP430x12x2.h"*et *"In430.h"* spécifique au microcontrôleur. Toutefois, nous avons eu quelques problèmes à cause des limitations techniques (16 bits en variable global) et le manque de confort en période de développement. Ainsi, nous avons utilisé l'environnement de développement *IAR Embedded Workbench IDE* v.4.3A. Les commentaires du code source ont été fait suivant la norme Doxygen pour faciliter la génération de la documentation.

Le circuit fournit par le client intègre un connecteur permettant la programmation du microcontrôleur par l'intermédiaire d'une sonde permettant le branchement sur un port parallèle d'un PC.

La taille maximale du programme implanté dans le microcontrôleur ne dépasse pas 4Ko. Le code du programme a été pensé pour limiter l'utilisation de la mémoire et compilé ave les options de compilations maximales pour optimiser la taille du code objet.

4. RÉALISATION

4.1 Structure des Fichiers et Données Globales

Notre code comporte les données globales suivantes :

- char **modeCourant**

Mémorisation des modes d'opération.

- **NoteID noteCourante**

Mémorisation de la note à accorder.

- int **timeLapse**

Temps écoulé pendant 100 périodes pour déterminer la fréquence jouée.

- int **filterHalfPeriod**

Nombre d'oscillations du cristal pour une demie période de l'horloge programmant le filtre.

Il y a sept fichiers source :

- main.c
- lcd.c
- lcd.h
- leds.c
- leds.h
- misc.c
- misc.h

Les fichiers sources *.c débutent par une entête (nom du module, auteurs et description) respectant les normes d'écriture de commentaire du *Doxygen*, indiquent les inclusions et si nécessaire les déclarations des fonctions et des variables. Les fonctions sont ensuite codées.
Les fichiers *.h, quant à eux, contiennent les prototypes des fonctions du module correspondant.

4.2 Algorithmes et Données

4.2.1 Programmation de l'Afficheur

4.2.1.1 Initialisation

L'utilisation de l'afficheur nécessite d'effectuer une séquence d'opérations précises afin de l'initialiser. Cette séquence est décrite par la figure 20, p.37 de la documentation du contrôleur SPLC501C. Les commandes envoyées sont documentées à la page p35-36.
A la fin de cette séquence nous avons ajoutés certaines opérations pour que l'état de l'afficheur corresponde à nos attentes.

La séquence d'opérations est la suivante :

- Mise de Reset à 0
- RESETB1 à 0 : Remise à configuration d'origine.
- Démarrage de l'alimentation.
- Les opérations suivantes doivent s'effectuer en moins de 5ms.
    - RESETB1 à 1 : Fin de la remise à zéro.
    - Envoi de la commande RESET CIRCUIT.
    - Envoi de la commande LCD BIAS SETTING (polarisation).
    - Envoi de la commande ADC SELECTION.
    - Envoi de la commande COMMON OUTPUT STATE SELECTION
    - Envoi de la commande BUILT IN RESISTANCE RATIO.
    - Envoi de la commande ELECTRONIC VOLUM CONTROL.
    - Envoi de la commande POWER CONTROL SETUP
- Envoi de la commande Display ON.
- Envoi de la commande DISPLAY START LINE SET.
- Allumage du rétro éclairage.

4.2.1.4 Affichage

Les opérations nécessaires à l'affichage de données sont décrites par la figure 22 p.38 de la documentation.

Les commandes sont les suivantes :

- Envois de la commande DISPLAY START LINE SET (déjà effectué lors de l'initialisation).
- Envois de la commande PAGE ADRESS SET.
- Envois de la commande COLUMS ADDRESS SET.

- Envois de 8bits de données suivant la technique présentée ci-après.

4.2.1.3 Envois de commandes ou de données

Les commandes et données sont envoyées sous forme de 8bits en série sur la broche SI1.

- Données : Mise de SAOP1, [P2.0] à 1.
- Contrôle : Mise de SAOP1, [P2.0] à 0.
- Pour chaque bit :
  - Mise à jours de SI1.
  - Tic de l'horloge SCL1.

4.2.2 Programmation du Filtre

La programmation du filtre se fait par l'intermédiaire de CLOCK_FILTER.
La fréquence du signal envoyé sur cette patte détermine la fréquence de coupure du filtre. Le rapport est $f_c = f_{CLK} / 100$.

Le tableau ci-après présente les fréquences des différentes notes à accorder et la fréquence f_{CLK} associée. Pour générer f_{CLK} il faut faire varier CLOCK_FILTER. Le cristal fonctionnant à 8MHz, on peut déduire le nombre d'oscillations du cristal par périodes et par demi périodes. A chaque demi période on inverse la sortie CLOCK_FILTER.

f_c (Hz)	f_{CLK} (kHz)	**Nombre d'oscillation du cristal par périodes**	**Par demie période**
82.4	8.24	970	485
110	11	727	363
146.8	14.68	546	273
196	19.6	408	204
246.9	24.69	324	162
329.6	32.96	242	121

4.2.3 Analyse de Fréquence

L'analyse de la fréquence se fait en utilisant l'ADC. L'ADC échantillonne le signal analogique pour produire une valeur dans le registre ADC10MEM. Les valeurs échantillonnées sont des valeurs absolues comprises entre 0x0 et 0x3FF. Elles sont centrées sur 0x200.
Pour détecter une période :
On attend le passage en 0x200 : début de la période.
On attend tant que les valeurs échantillonnées sont supérieures à 0x200. (demi période supérieure).

On attend tant que les valeurs échantillonnées sont inférieures à 0x200. (demi période inférieur).

Pour limiter les erreurs, on échantillonne 100 périodes consécutivement en mémorisant le temps écoulé. On peut ensuite calculer la fréquence moyenne.
Pour éviter les problèmes d'hystérésis un léger délai d'une demie milliseconde est imposé entre chaque échantillonnage.

L'analyse de la fréquence se fait aussi bien sur le signal filtré que non filtré. Le signal non filtré est analysé une première fois en mode automatique avant l'analyse du signal filtré.

4.2.4 Utilisation du Timer

Le timer A utilise comme source l'horloge ACLK, elle même basée sur le cristal, sans diviseur.
Il est configuré en mode continu, c'est-à-dire qu'il compte de 0x0 à 0xFFFF.
On l'utilise pour générer deux intervalles de temps différents :

- Le premier toutes les 8000 oscillations (1 milliseconde) pour compter le temps écoulé pendant l'analyse des fréquences.
- Le second pour générer le signal d'horloge qui programme le filtre.

On utilise pour cela les registres TACCR1 et TACCR2 pour générer des interruptions après le nombre d'oscillations désirées.

4.2.5 Détection en Mode Automatique

En mode automatique, la fréquence est d'abord analysée de manière approximative sur FILTER IN, ensuite, le filtre est programmé suivant la note la plus proche de celle détectée sur le signal non filtré, et le signal filtré est analysé.
Pour déterminer la note la plus proche de la note jouée, on teste le temps écoulé pendant l'analyse du signal non filtré. On compare ce temps à celui de la colonne "Valeur Maximale" pour déterminer la note la plus proche.

Note	Fréquence (Hz)	Laps de temps (ms/100 périodes)	Valeur maximale (ms/100 périodes)
Mi Grave	82.4	1213	1365
La	110	909	1071
Ré	146.8	681	795
Sol	196	510	595
Si	246	405	457
Mi	329.6	303	353

4.2 Description des fichiers

4.2.1 Fichier main.c

```
#include <msp430x12x2.h>
#include <In430.h>
#include "leds.h"
#include "lcd.h"
#include "misc.h"
```

Fonctions

- void **init** (void)
  Fonction d'initialisation des composants.
- void **frequencyAnalyse** (void)
  Fonction d'analyse des notes jouées.
- void **main** (void)
  Point d'entrée du programme.
- __interrupt void **interruptHandlerP1** (void)
  Gestionnaire d'interruptions sur les pattes 1.x : boutons de réglages.
- __interrupt void **interruptHandlerTimerA** (void)
  Gestionnaire d'interruptions du timer A, fonctionnement en mode continu.

Description détaillée

Auteur:

Samuel Rollet
Jennifer Salle

Depuis:

Mars 2006

Version:

1.0

Point d'entrée du programme.

Projet : Accordeur de Guitare - Projet d'Architecture - Première Année Ecole Supérieure d'Ingénieurs de Luminy, Université de la Méditerranée - Aix Marseille II

Initialisation, boucle d'analyse de fréquences, traitement des interruptions par le microcontrôleur MSP430F1222.

```
                          MSP430F1222
                        -----------------
                        |                 |
                    <--|TEST         P1.7|<--TDI_TDO1      (BOUTON 1)
                        |                 |
                    <--|VCC          P1.6|<--TDI_VPP1      (BOUTON 2)
                        |                 |
(LED)    RED_LED1<--|P2.5         P1.5|<--TTMS1         (BOUTON 3)
                        |                 |
           GND1/G<--|V55          P1.4|<--TCK1          (BOUTON 4)
                        |                 |
        |-------|-<--|XOUT         P1.3|-->BCL_LIGHT_ON1 (LCD)
        |Crystal|    |                 |
        |-------|->--|XIN          P1.2|-->GREEN_LED1    (LED)
                        |                 |
                 <--o|RTS*         P1.1|-->CLOCK_FILTER  (*Filtre)
                        |                 |
(LCD)        AOP1<--|P2.0         P1.0|-->SHDWN_FILTER  (*Filtre)
                        |                 |
(LCD)        SCL1<--|P2.1         P2.4|-->RESET_B1      (LCD)
                        |                 |
(LCD)         SI1<--|P2.2         P2.3|-->CS1           (LCD)
                        |                 |
      EXTERNAL_IN-->|P3.0      P3.7/A7|<--FILTER_IN     (*Filtre)
                        |                 |
                   X--|P3.1      P3.6/A6|<--FILTER_OUT    (*Filtre)
                        |                 |
           INVALIDX--|P3.2         P3.5|--X
                        |                 |
                   X--|P3.3         P3.4|--X
                        |                 |
                        -----------------
```

Spécifications techniques:

- Contrôleur: MSP430F1222 de Texas Instrument.
- Ecran : SP 5-GFX1 de Lascar avec contrôleur SPLC501C de Sunplus
- Filtre : MAX7404 de MAXIM
- Amplificateur : MAX4462 de MAXIM
- Cristal : 8MHz
- Compilé avec IAR Embedded Workbench IDE 4.3A (4.3.1.0)

Documentation des fonctions

void frequencyAnalyse (void)

Fonction d'analyse des notes jouées.

Analyse en boucle de la fréquence d'entrée sur A6 ou A7. Apres 100 périodes, la fréquence moyenne est affichée. En mode automatique, la note la plus proche de la fréquence jouée est affichée.

void init (void)

Fonction d'initialisation des composants.

Choix du cristal comme source des horloges. Paramétrage des E/S comme schématisé plus haut (Les pattes inutilisées sont paramétrées comme des sorties pour limiter la consommation d'énergie). Configuration de l'ADC10 : échantillonnage de A6 : Filter Out. Initialisation de l'écran LCD. Initialisation du TimerA : Mode continue. Initialisation des variables globales : Mode manuel, Consommation normale, note : Mi grave.

__interrupt void interruptHandlerP1 (void)

Gestionnaire d'interruptions sur les pattes 1.x : boutons de réglages.

Pin1.7 : Bouton numéro 1 : Mise en veille.

Pin1.6 : Bouton numéro 2 : Changement de mode.

Pin1.5 : Bouton numéro 3 : Choix de note à accorder en mode manuel.

Pin1.4 : Bouton numéro 4 : On/Off. Suspension des interruptions, traitement, temporisation, rétablissement des interruptions.

__interrupt void interruptHandlerTimerA (void)

Gestionnaire d'interruptions du timer A, fonctionnement en mode continu.

CCR1 : interruptions toutes les 8000 oscillations du cristal, c'est-à-dire toutes les 1ms.
CCR2 : interruptions toutes les 'filterHalfPeriod', utilisé pour générer l'horloge programmant le filtre.

void main (void)

Point d'entrée du programme.

Initialisation des composants et appel à l'analyse de fréquences.

Documentation des variables

int filterHalfPeriod

Nombre d'oscillations du cristal pour une demie période de l'horloge programmant le filtre.

char modeCourant

Mémorisation des modes d'opérations.

Fonctionnement :

- Manuel : bit0.
- Automatique : bit1.

Consommation :

- Normal : bit2.
- Sommeil : bit3.
- Off : bit4.

NoteID noteCourante

Mémorisation de la note à accorder.

int timeLapse

Temps écoulé pendant 100 périodes pour déterminer la fréquence jouée.

4.2.2 Fichier lcd.c

```
#include <msp430x12x2.h>
#include <In430.h>
#include "lcd.h"
```

Fonctions

- void **sendToLcd** (unsigned char byte)
  Envoi 8bits au contrôleur SPLC501C de l'afficheur LCD sans se soucier de la valeur de SAOP1.
- void **dataDisplayDotsLcd** (void)
  *Fonction écrivant une série de 5 points sur 5*8 pixels.*
- void **dataDisplayHzLcd** (void)
  Fonction écrivant Hz pour 'Hertz' à la position courante sur l'écran.
- char **dataDisplayChar** (**CharID** charToPrint, char colonne, char page)
  Fonction affichant le caractère charToPrint à l'écran, à la position colonne sur la page page.
- void **clearPageLcd** (char page, char colonneDebut)
  Fonction effaçant une page à partir d'une colonne donnée.
- void **initLcd** (void)
  Initialise l'afficheur LCD pour qu'il soit prêt à recevoir les commandes d'affichage de données.
- void **sendControlToLcd** (unsigned char byte)
  Envoi une commande de contrôle au contrôleur SPLC501C de l'afficheur LCD.
- void **sendDataToLcd** (unsigned char byte)
  Envoi 8bits de données au contrôleur SPLC501C de l'afficheur LCD.
- void **lcdPowerOff** (void)
  Fonction désactivant l'afficheur LCD.
- void **dataDisplayAllZeroLcd** (void)
  Fonction qui efface l'ensemble des pixels de l'écran.
- void **dataDisplayStartupLcd** (void)
  Fonction d'affichage de l'écran de démarrage: "ESIL TUNER" et séquence d'animation.
- void **dataDisplayModeLcd** (char mode)
  Fonction affichant le mode de fonctionnement de l'afficheur : manuel ou automatique.
- void **dataDisplayNoteLcd** (**NoteID** note)
  Fonction affichant la note à accorder et sa fréquence.
- void **dataDisplayStaticLcd** (void)
  Fonction affichant les éléments statiques de l'afficheur : ligne, diagramme.
- void **dataDisplayCursorLcd** (char positionCurseur)
  Fonction affichant le curseur à l'écran.
- void **dataDisplayFrequenceJoueeLcd** (float frequence)
  Fonction affichant une fréquence en bas de l'afficheur.

Description détaillée

Module offrant des fonctions de gestion du LCD de l'accordeur depuis le microcontrôleur. Pré-requis : la fonction **initLcd()** doit être appelé avant l'utilisation des autres fonctions.

```
                      MSP430F1222
                  -----------------
(LCD)    AOP1<--|P2.0         P1.3|-->BCL_LIGHT_ON1 (LCD)
                |                 |
(LCD)    SCL1<--|P2.1         P2.4|-->RESET_B1      (LCD)
                |                 |
(LCD)     SI1<--|P2.2         P2.3|-->CS1           (LCD)

       LCD
        --
       |10|-->10
       | 9|-->V51/G                    (éclairage: positive power)
       | 8|-->DGND_BCKLIGHT    [P1.3]  (éclairage: negative power)
       | 7|-->CS1              [P2.3]  (chip select input: I/O)
       | 6|-->RESETB1          [P2.4]  (remise à config d'origine)
       | 5|-->SAOP1            [P2.0]  (flag control/display data)
       | 4|-->SCL1             [P2.1]  (clk de l'entree serie)
       | 3|-->SI1              [P2.2]  (entree serie)
       | 2|-->V51/G                    (matrice lcd: positive power)
       | 1|-->GND1/G                   (matrice lcd: negative power)
        --
```

Documentation des fonctions

void clearPageLcd (char *page*, char *colonneDebut*)

Fonction effaçant une page à partir d'une colonne donnée.

Paramètres:

page Page à effacer, sous la forme 0xBi : i le numéro de page.
colonneDebut Colonne à partir de laquelle la page doit être effacée.

void dataDisplayAllZeroLcd (void)

Fonction qui efface l'ensemble des pixels de l'écran.

N'active pas l'affichage de l'écran.

char dataDisplayChar (CharID *charToPrint*, char *colonne*, char *page*)

Fonction affichant le caractère charToPrint à l'écran, à la position colonne sur la page *page*.

Les caractères sont affichés sur 2 pages.

Paramètres:

charToPrint Identifiant du caractère à afficher.
colonne Colonne à partir de laquelle le caractère doit être affiché.
page Page sur laquelle la moitié supérieure du caractère est affichée. (sous la forme 0xBi : i le numéro de page.)

Renvoie:

colonne + largeur du caractère + 2 colonnes d'espace.

void dataDisplayCursorLcd (char *positionCurseur*)

Fonction affichant le curseur à l'écran.

Paramètres:

positionCurseur Position du curseur : 7 positions, position centrée : 4.

void dataDisplayDotsLcd (void)

Fonction écrivant une série de 5 points sur 5*8 pixels.

Utilisée pour l'affichage du diagramme. Fonction privée à ce module.

Voir également:

void **initLcd(void)**
void dataDisplayDiagrammeLcd(void)

void dataDisplayFrequenceJoueeLcd (float *frequence*)

Fonction affichant une fréquence en bas de l'afficheur.

Les valeurs affichées sont entre 0 et 400Hz, les valeurs supérieures sont ignorées.

Paramètres:

frequence Fréquence à afficher.

void dataDisplayHzLcd (void)

Fonction écrivant Hz pour 'Hertz' à la position courante sur l'écran.

Fonction privée à ce module. Note:

- La position où Hz doit être écrit doit être précisée avant l'appelle à cette fonction.

Voir également:

void **initLcd(void)**

void dataDisplayModeLcd (char *mode*)

Fonction affichant le mode de fonctionnement de l'afficheur : manuel ou automatique.

Affiche M ou A en haut à gauche du LCD.

Paramètres:

mode Manuel bit0 à 1, auto bit1 à 1

void dataDisplayNoteLcd (NoteID *note*)

Fonction affichant la note à accorder et sa fréquence.

Affiche un point en mode automatique.

Paramètres:

note Identifiant de la note dont les informations doivent être affichées.

void dataDisplayStartupLcd (void)

Fonction d'affichage de l'écran de démarrage: "ESIL TUNER" et séquence d'animation.

Note:

- L'écran doit avoir été initialise avant d'utiliser cette fonction avec void **initLcd(void)**.
- Utiliser **dataDisplayAllZeroLcd(void)** pour effacer les données présentes en mémoire avant d'écrire.

Voir également:

void **initLcd(void)**
void **dataDisplayAllZeroLcd(void)**

void dataDisplayStaticLcd (void)

Fonction affichant les éléments statiques de l'afficheur : ligne, diagramme.

void initLcd (void)

Initialise l'afficheur LCD pour qu'il soit prêt à recevoir les commandes d'affichage de données.

Effectue la séquence de commandes décrite par la figure 20, p.37 de la documentation du contrôleur SPLC501C. Ensuite: rend l'affichage inactif pour éviter l'affichage de données non désires, éteint l'éclairage.

void lcdPowerOff (void)

Fonction désactivant l'afficheur LCD.

Envoi de la commande 'display off', 'display all point off', reset (pin 2.4) à 0, backlight off (pin 1.3).

void sendControlToLcd (unsigned char *byte*)

Envoi une commande de contrôle au contrôleur SPLC501C de l'afficheur LCD.

Met l'indicateur SAOP1 à 0: données de contrôles en entrée du connecteur série. Transmet les 8 bits correspondant à la commande spécifiée dans byte, bit par bit.

Paramètres:

byte Un octet contenant le code associé à la commande à transmettre (sfrb sur 1 octet).

Voir également:

"Liste des commandes : p.35-36 de la doc du SPLC501C."
sendToLcd(sfrb byte)

void sendDataToLcd (unsigned char *byte*)

Envois 8bits de données au contrôleur SPLC501C de l'afficheur LCD.

Met l'indicateur SAOP1 à 1: données en entrée du connecteur série. Transmet les 8 bits correspondant aux données spécifiées dans byte, bit par bit.

Paramètres:

byte Un octet contenant le code associé à la commande à transmettre. (sfrb sur 1 octet).

Voir également:

sendToLcd(sfrb byte)

void sendToLcd (unsigned char *byte*)

Envoi 8bits au contrôleur SPLC501C de l'afficheur LCD sans se soucier de la valeur de SAOP1.

Transmet les 8 bits spécifiés dans byte, bit par bit. Note:

- Cette fonction ne devrai pas être utilisée directement, seulement par l'intermédiaire de **sendDataToLcd()** et **sendControlToLcd()**.
- Cette fonction n'est pas publique (pas utilisable en dehors de ce module).
- ***Paramètres:***

  byte Un octet contenant les informations à transmettre (sfrb sur 1 octet).
- ***Voir également:***

  sendControlToLcd(sfrb byte)
  sendDataToLcd(sfrb byte)

4.2.3 Fichier lcd.h

```
#include "misc.h"
#include "leds.h"
```

Énumérations

- enum **CharID** { **CHAR_0**, **CHAR_1**, **CHAR_2**, **CHAR_3**, **CHAR_4**, **CHAR_5**, **CHAR_6**, **CHAR_7**, **CHAR_8**, **CHAR_9**, **CHAR_M**, **CHAR_S**, **CHAR_R**, **CHAR_L**, **CHAR_i**, **CHAR_o**, **CHAR_l**, **CHAR_e**, **CHAR_a**, **CHAR_pt**, **CHAR_2pt**, **CHAR_ESIL_E**, **CHAR_ESIL_S**, **CHAR_ESIL_I**, **CHAR_ESIL_L**, **CHAR_ESIL_T**, **CHAR_ESIL_U**, **CHAR_ESIL_N**, **CHAR_ESIL_R** }
  Définition des identifiants des caractères affichables.

Description détaillée

Entête du module offrant des fonctions de gestion du LCD de l'accordeur depuis le microcontrôleur.

Documentation du type de l'énumération

enum CharID

Définition des identifiants des caractères affichables.

Valeurs énumérées:

CHAR_0
CHAR_1
CHAR_2
CHAR_3
CHAR_4
CHAR_5
CHAR_6
CHAR_7
CHAR_8
CHAR_9
CHAR_M
CHAR_S
CHAR_R
CHAR_L
CHAR_i
CHAR_o
CHAR_l
CHAR_e
CHAR_a
CHAR_pt
CHAR_2pt
CHAR_ESIL_E
CHAR_ESIL_S
CHAR_ESIL_I
CHAR_ESIL_L
CHAR_ESIL_T
CHAR_ESIL_U
CHAR_ESIL_N
CHAR_ESIL_R

4.2.4 Fichier leds.c

```
#include <msp430x12x2.h>
#include "leds.h"
```

Fonctions

- void **ledOn** (**LedID** led)
  Allume la LED spécifiée.
- void **ledOff** (**LedID** led)
  Eteint la LED spécifiée.
- void **switchLed** (**LedID** led)
  Inverse la LED spécifiée.

Description détaillée

Module offrant des fonctions de gestion des LED de l'accordeur depuis le microcontrôleur.

```
                MSP430F1222
            -----------------
        /|\|              XIN|-
         | |                 |
         --|RST          XOUT|-
           |                 |
           |             P1.2|-->GREEN_LED
           |                 |
           |             P2.5|-->RED_LED
```

Documentation des fonctions

void ledOff (LedID *led*)

Eteint la LED spécifiée.

Paramètres:

led Identifiant de la LED à éteindre.

void ledOn (LedID *led*)

Allume la LED spécifiée.

Paramètres:

led Identifiant de la LED à allumer.

void switchLed (LedID *led*)

Inverse la LED spécifiée.

Si elle est allumée elle s'éteint, sinon elle s'allume.

Paramètres:

led Identifiant de la LED à inverser.

4.2.5 Fichier leds.h

```
#include "misc.h"
```

Énumérations

- enum **LedID** { **RED_LED**, **GREEN_LED** }
  Définition des identifiants des LED.

Description détaillée

Entête du module offrant des fonctions de gestion des LED de l'accordeur depuis le microcontrôleur.

Documentation du type de l'énumération

enum LedID

Définition des identifiants des LED.

Valeurs énumérées:

RED_LED
GREEN_LED

4.2.6 Fichier misc.c

```
#include <msp430x12x2.h>
#include "misc.h"
```

Fonctions

- void **wait** (long length)
  Fonction offrant un délai a l'aide d'une boucle.

Description détaillée

Module offrant des fonctions utilitaires annexes.

Documentation des fonctions

void wait (long *length*)

Fonction offrant un délai a l'aide d'une boucle.

Paramètres:

length Nombre d'occurrences de la boucle de délais.

4.2.7 Fichier misc.h

```
#include "leds.h"
```

Énumérations

- enum **NoteID** { **MI1**, **LA**, **RE**, **SOL**, **SI**, **MI2**, **NONE** }
  Définition des identifiants des notes.

Description détaillée

Entête du module offrant des fonctions de gestion des LED de l'accordeur depuis le microcontrôleur.

Documentation du type de l'énumération

enum NoteID

Définition des identifiants des notes.

Valeurs énumérées:

MI1
LA
RE
SOL
SI
MI2
NONE

5. RÉSULTATS

5.1 Problèmes Rencontrés

Le développement de ESIL TUNER nous a confronté à différents problèmes, aussi bien logiciels que matériels.

5.1.1 Problème de Batteries

Au début du développement nous avons été confronté à un comportement inattendu de notre programme et de la plaquette. L'écran LCD clignotait de manière anormale, le chargement du programme et le débuguage fonctionnaient de manière aléatoire. Après avoir étudié les erreurs possibles dans les dernières modifications de notre programme nous avons constaté que la batterie ne fournissait plus assez d'énergie.

5.1.2 Vitesse d'Exécution

Le début du développement c'est effectué en utilisant l'horloge interne. Après avoir paramétré l'utilisation du cristal, les temps d'exécution se sont accélérés. La différence la plus notable se situait au niveau des boucles d'attentes qui produisaient des délais moins importants.

5.1.3 Limitation de la Mémoire

La limitation de la mémoire du microcontrôleur MSP430F1222 à 4Ko a été un handicape pendant le développement de ce projet. En effet après avoir fini le développement des fonctions de gestion de l'afficheur, plus de 80% de la mémoire était utilisée.
Un temps important a été consacré à ce moment là à l'optimisation du code. La taille des variables a été réduite partout ou cela était possible ainsi que les appels de fonctions. Un appel de fonction coûtant 8 octets, les fonctions peu appelées et avec un minimum de code ont étés supprimés pour intégrer leur code dans le corps des fonctions.
L'habitude d'effectuer une programmation modulaire où les modules correspondent à des éléments bien identifiés et où chaque action est isolée dans une fonction n'est pas adaptée à un développement pour un microcontrôleur. Une telle structure de programme est coûteuse en mémoire. Ainsi, du code qui devrait se trouver dans différents modules de par ses fonctions mais qui s'exécute de manière consécutive a été regroupé pour éviter les appels multiples de fonctions.
Le paramétrage des options de compilation nous a aussi permis de gagner de l'espace. Le code livré ne peut pas être chargé sur le contrôleur sans paramétrer le compilateur avec une optimisation maximale de la taille.

5.1.4 Variables Statiques

Les fonctions chargées de l'affichage des caractères sur l'écran LCD comportent des tableaux de données de taille assez importantes à leur début.
Si ces tableaux ne sont pas déclarés comme *static const* le code assembleur qui était généré par l'ancienne version du compilateur produisait une erreur d'exécution du programme. Celle-ci était évitée en les déclarant comme *static const*. En effet, les variables qui ne sont pas *static const* et qui ont une valeur initiale sont copiées à chaque entrée dans la fonction. Des blocs de données trop importants engendrant des erreurs, il était nécessaire de les déclarer comme constantes.

5.1.5 Hystérésis

Le signal échantillonné par l'ADC n'est pas un signal parfait et comporte du bruit. Cela peut fausser le test de passage en une valeur donnée ; en effet, si l'échantillonnage est rapide, le bruit autour de la valeur testé peut être détecté comme un franchissement du seuil. Nous avions anticipé ce problème lors de la conception et envisagions de tester le franchissement consécutif de deux valeurs proches par le signal pour éviter ce problème.
Cependant la première version du code effectuant l'analyse de fréquence ne testait le passage qu'en une seule valeur. Après les premiers tests de l'analyse du signal non filtré avec un générateur de fréquence, et ceux-ci s'étant révélés très concluant, nous avions conclus que deux franchissements consécutifs n'étaient pas nécessaires.
Cependant, l'application du même algorithme au signal filtré donnait des résultats aléatoires.
Après des tests sur le bon fonctionnement du filtre, de la plaquette et de notre code, nous avons identifié l'erreur. Le signal amplifié et non filtré, avec lequel notre analyse de fréquence fonctionnait, était saturé. Cela limitait considérablement le bruit au niveau de la valeur moyenne dont nous testons le franchissement.
En revanche, pour un signal sinusoïdal, comme le signal filtré, le bruit provoquait les erreurs prévues.
Pour résoudre le problème, nous avons introduit un délai entre chaque échantillonnage afin d'assurer une évolution assez importante du signal pour que le bruit ne vienne pas perturber l'échantillonnage.

5.2 Signaux

Pour effectuer les tests d'ESIL TUNER, nous avons utilisé un générateur de fréquence et un oscilloscope. Les schémas suivants ont été pris pour la note *Si*.

5.2.1 Signal Source

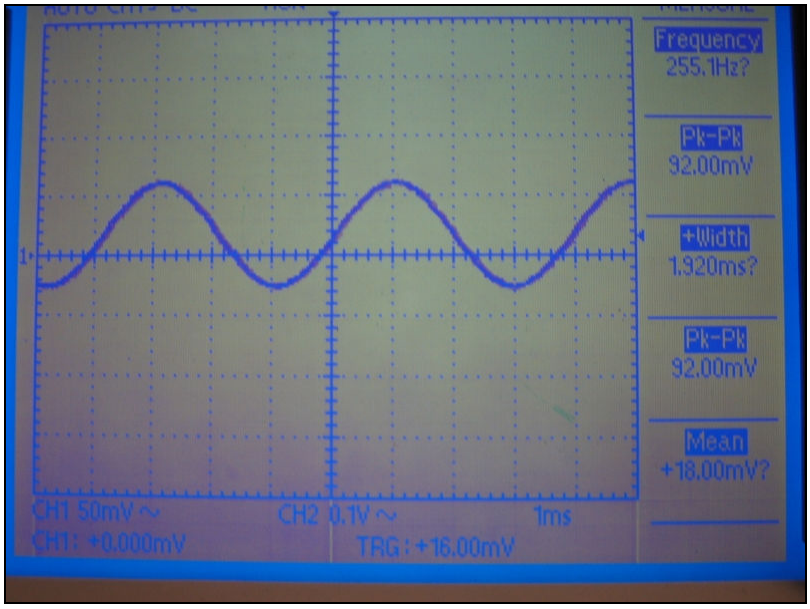


Figure 15 -Signal Source

5.2.2 Signal Amplifié

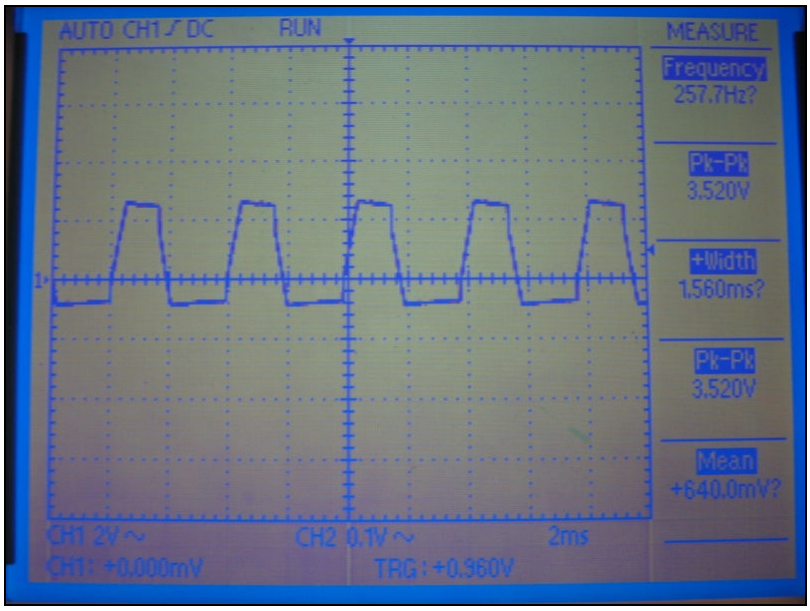


Figure 16 - Signal Amplifié

5.2.3 Signal Filtré

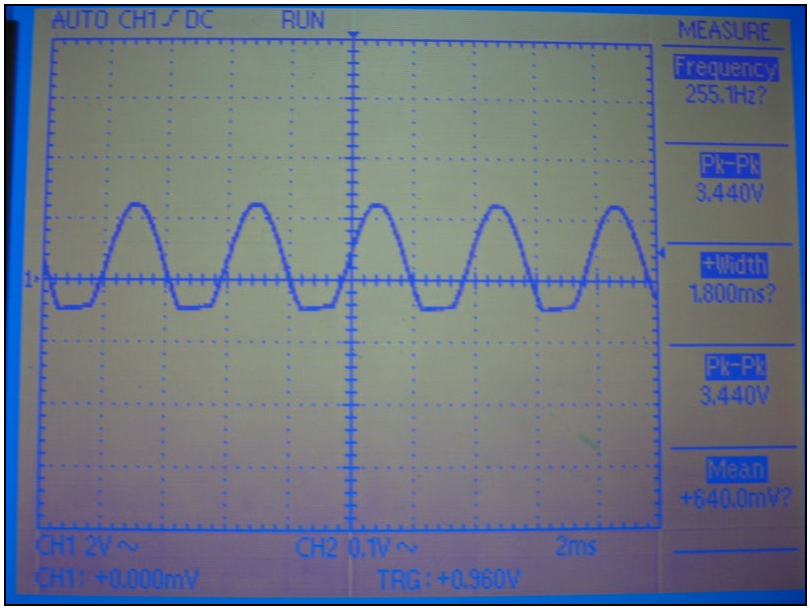


Figure 17 - Signal Filtré

5.2.4 Bruit sur le Signal

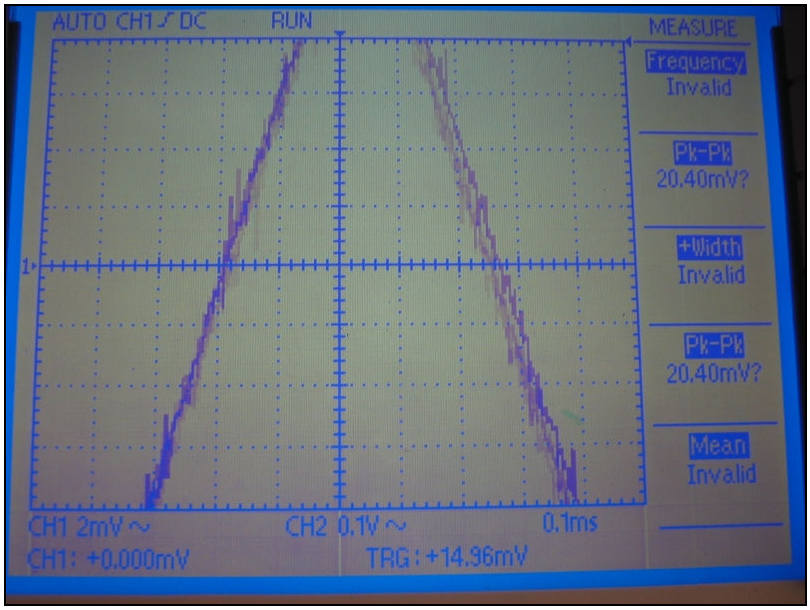


Figure 18 - Bruit sur le Signal

5.2.5 Horloge de Programmation du Filtre

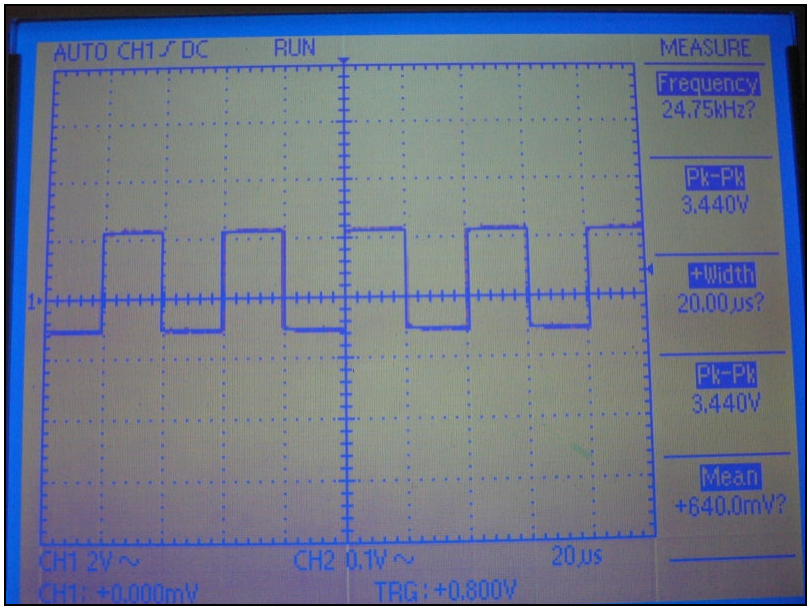


Figure 19 - Horloge de Programmation du Filtre

5.2.6 Résultat Observé

Figure 20 - Résultat Observé

6. CONCLUSION

Le produit fournit au client respecte les exigences de fonctionnement définie dans le cahier des charges, notamment les fonctions codées se sont avérées concluantes. Cependant, à cause des limitations de mémoire, toutes les fonctions ne sont pas regroupées dans le produit fini. Par exemple, nous n'avons pas pu affiché le curseur comme il était prévu avec le client. Toutefois, cela n'empêche pas un bon fonctionnement du produit.
De plus, le planning initial a bien été respecté et nous avons pu livrer le produit à la date prévue.

Equipe de développement:

Samuel Rollet *Jennifer Salle*

ANNEXE 1
Références

Afficheur

SPLC501C Data Sheet v1.4, Sunplus Technology Co., Ltd., Hsin Chu: Taiwan, 2002

SP5-GFX1 Data Sheet Issue 1, Lascar Electronics Inc., P.A.: Erie, 2003

Panel Instrument Application Notes Issue 2, Lascar Electronics Inc., P.A.: Erie

SP 5-GFX1, Lascar Electronics Inc., P.A.: Erie, 2006, http://www.lascarelectronics.com/PRODUCTS.CFM?STOCKNO=SP%205%2DGFX1&CFID=20530829&CFTOKEN=98183249 (Avril 2006)

Amplificateur

MAX4460/MAX4461/MAX4462 Data Sheet Rev 3, Maxim Integrated Products, C.A.: Sunnyvale, 2005

Contrôleur

MSP430x12x Mixed Signal Microcontroller (Rev. C), Texas Instruments, Incorporated, T.X.: Dallas, 2006

MSP430x1xx Family User's Guide (Rev. E), Texas Instruments, Incorporated, T.X.: Dallas, 2005

Code Composer Essentials Code Examples C, Texas Instruments, Incorporated, T.X.: Dallas, 2006

Product Folder : MSP430F122 - 16-Bit Ultra-Low-Power Microcontroller, 4kB Flash, 256B RAM, USART, Comparator, Texas Instruments, Incorporated, T.X.: Dallas, 2006, http://focus.ti.com/docs/prod/folders/print/msp430f122.html (Avril 2006)

Filtre

MAX7400/MAX7403/MAX7404/MAX7407 Full Data Sheet, Maxim Integrated Products, C.A.: Sunnyvale, 1999

MAX7400, MAX7403, MAX7404, MAX7407 8th-Order, Lowpass, Elliptic, Switched-Capacitor Filters, Maxim Integrated Products, C.A.: Sunnyvale, 2006, http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1899 (Avril 2006)

ANNEXE 2
Table de fréquence des notes

octave	-1	0	1	2	3	4	5	6	7	8	9
NOTES											
Do	16.3 Hz	32.7 Hz	65 Hz	131 Hz	262 Hz	523 Hz	1 046.5 Hz	2 093 Hz	4 186 Hz	8 372 Hz	16 744 Hz
Do diese ou Re bemol	17.3 Hz	34.6 Hz	69 Hz	139 Hz	277 Hz	554 Hz	1 109 Hz	2 217 Hz	4 435 Hz	8 870 Hz	17 740 Hz
Re	18.3 Hz	36.7 Hz	74 Hz	147 Hz	294 Hz	587 Hz	1 175 Hz	2 349 Hz	4 698 Hz	9 396 Hz	18 792 Hz
Re diese ou Mi bemol	19.4 Hz	38.9 Hz	78 Hz	156 Hz	311 Hz	622 Hz	1 244.5 Hz	2 489 Hz	4 978 Hz	9 956 Hz	19 912 Hz
Mi	20.5 Hz	41.2 Hz	83 Hz	165 Hz	330 Hz	659 Hz	1 318.5 Hz	2 637 Hz	5 274 Hz	10 548 Hz	21 098 Hz
Fa	21.8 Hz	43.6 Hz	87 Hz	175 Hz	349 Hz	698.5 Hz	1 397 Hz	2 794 Hz	5 588 Hz	11 176 Hz	
Fa diese ou Sol bemol	23.1 Hz	46.2 Hz	92.5 Hz	185 Hz	370 Hz	740 Hz	1 480 Hz	2 960 Hz	5 920 Hz	11 840 Hz	
Sol	24.5 Hz	49.0 Hz	98 Hz	196 Hz	392 Hz	784 Hz	1 568 Hz	3 136 Hz	6 272 Hz	12 544 Hz	
Sol diese ou La bemol	26.0 Hz	51.9 Hz	104 Hz	208 Hz	415 Hz	831 Hz	1 661 Hz	3 322 Hz	6 645 Hz	13 290 Hz	
La	27.5 Hz	55.0 Hz	110 Hz	220 Hz	440 Hz	880 Hz	1 760 Hz	3 520 Hz	7 040 Hz	14 080 Hz	
La diese ou Si bemol	29.1 Hz	58.0 Hz	117 Hz	233 Hz	466 Hz	932 Hz	1 865 Hz	3 729 Hz	7 458 Hz	14 918 Hz	
Si	30.8 Hz	62.0 Hz	123 Hz	247 Hz	494 Hz	988 Hz	1 975 Hz	3 951 Hz	7 902 Hz	15 804 Hz	

Note : Les notes identifiées sont celles jouées par une guitare.

ANNEXE 3
Code Sources

Les codes sources sont disponibles sous leur format original dans le dossier *ESIL_TUNER/sources/*.