ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cours : Analyse des Signaux ELE2700 Professeur M. Corinthios

Département de Génie Électrique

Séance 4-TP3

Partie 1

Introduction à la technologie STM32F407 pour le traitement numérique du signal (DSP)

Saad Chidami - 2014

Table des matières

ntroduction	4
Dbjectif du laboratoire	7
Génération de signal à l'aide de Matlab	8
Génération du signal à l'aide du DSP	9
Acquisition de signal à l'aide du DSP	.17
Filtrage à l'aide du DSP	.22
Conclusion	.28

Liste des figures

Figure 1 : Bloc Diagramme représentant le flux général d'une application de DSP	4
Figure 2 : Composantes d'un processeur Cortex-M4f	5
Figure 3 : Photo du microcontrôleur sur la carte de développement STM32F4DISCOVERY	6
Figure 4 : Script Matlab pour la génération de signal	8
Figure 5 : Signal généré avec une fréquence de 60 Hz	8
Figure 6 : Signal additionné de bruit (1 kHz)	8
Figure 7 : Localisation du Bloc Target Setup	9
Figure 8 : Détail du bloc Target Setup	10
Figure 9 : Bloc Simulink pour le Arbitrary Waveform Generator	11
Figure 10 : Détail du bloc Arbitrary Waveform Generator	. 12
Figure 11 : Configuration du générateur de formes	. 14
Figure 12 : Schéma Simulink pour la génération de signal	. 15
Figure 13 : Bouton pour la compilation des blocs et la programmation du DSP	. 15
Figure 14 : Signal généré par le DSP mesuré à l'oscilloscope	. 16
Figure 15 : Bloc Simulink pour le Regular ADC	. 17
Figure 16 : Détail du bloc Regular ADC	. 18
Figure 17 : Schéma préliminaire Simulink	. 19
Figure 18 : Configuration du niveau de priorité générateur de signaux	. 20
Figure 19 : Configuration du niveau de priorité générateur de l'ADC	. 20
Figure 20 : Signaux générés et échantillonnés par le DSP, mesurés à l'oscilloscope	.21
Figure 21 : Bloc Simulink pour le Digital Filter Design	. 22
Figure 22 : Paramètres pour le filtre coupe-bande	. 23
Figure 23 : Bloc Simulink pour Switch	24
Figure 24 : Blocs Simulink pour Debounce et Digital Input	24
Figure 25 : Configuration du bloc Digital Input	. 25
Figure 26 : Configuration du bloc Debounce	. 26
Figure 27 : Schéma Simulink représentant le système à implémenter	27
Figure 28 : Signal mesuré et signal filtré	. 27

Introduction

Lors du développement d'applications qui requièrent l'analyse de signaux, l'utilisation de matériel spécialisé est souvent nécessaire. Ce matériel consiste en un microprocesseur spécialisé appelé : « DSP » ou « *Digital Signal Processor* » (processeur de signaux numériques).

Ce microprocesseur peut être programmé à l'aide d'instructions spécialisées qui permettent le traitement de signaux en temps réel ou du moins le plus rapidement possible. Ces processeurs se retrouvent dans différentes applications comme : les baladeurs audio, les appareils biomédicaux, les systèmes téléphoniques et les applications vidéo.

Un système autonome ou embarqué qui nécessite du traitement de signal, se résume en général au schéma suivant :



Figure 1 : Bloc Diagramme représentant le flux général d'une application de DSP¹

Le signal analogique est échantillonné à l'aide d'un périphérique appelé ADC (« *analog to digital converter* ») ou CAN (convertisseur analogique-numérique). Cette étape permet la conversion d'une unité analogique (des Volts en général) vers une unité numérique (binaire) utilisable par le DSP.

Lorsque le traitement voulu (programmé) est effectué, le système transfère les nouvelles données vers le monde extérieur à l'aide d'un périphérique DAC *(« Digital to Analog converter »)* ou CNA (convertisseur numérique-analogique).

Cette représentation peut être différente et dépendra uniquement de l'application, mais la présence du DSP est essentielle au fonctionnement du système.

¹ <u>http://upload.wikimedia.org/wikipedia/commons/thumb/b/bc/DSP_block_diagram.svg/410px-DSP_block_diagram.svg.png</u>

Le DSP qui sera utilisé dans les laboratoires est un processeur Cortex-M4f développé par la compagnie ARM².



Figure 2 : Composantes d'un processeur Cortex-M4f

Ce processeur a une architecture Harvard de 32bits avec une extension DSP et un coprocesseur en point flottant.

La compagnie ARM est dite « *fabless* », c'est-à-dire qu'elle est spécialisée en conception et qu'elle ne fabrique pas elle-même les composantes, mais qu'elle vend des licences à d'autres compagnies qui possèdent des unités de production (Apple, Texas Instrument, etc...). Le matériel utilisé est développé par la compagnie STmicroelectronics, le processeur est intégré dans le microcontrôleur STM32F407.

Un microcontrôleur est un : « **système embarqué** (ou **système enfoui**) est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Le terme désigne aussi bien le matériel informatique que le logiciel utilisé. Ses ressources sont généralement limitées. Cette limitation est généralement d'ordre spatial (encombrement réduit) et énergétique (consommation restreinte). »³

² <u>http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php</u>

³ <u>http://fr.wikipedia.org/wiki/Syst%C3%A8me_embarqu%C3%A9</u>



Figure 3 : Photo du microcontrôleur sur la carte de développement STM32F4DISCOVERY

Objectif du laboratoire

L'objectif du laboratoire est de s'initier à l'utilisation du matériel pour une application dans le domaine du traitement de signal.

- La première étape consiste à utiliser la carte comme un générateur de signal :
 - construction du signal à l'aide de Matlab.
 - Configuration de la carte de développement à l'aide de Simulink
 - Utilisation et configuration du périphérique DAC pour générer le signal construit
- La deuxième étape consiste à développer une application DSP :
 - Configurer un des périphérique ADC du microcontrôleur
 - Connecter la sortie du signal du générateur (construit lors de la première étape)
 - Initiation aux filtres numériques et implémentation
 - Implémenter de la logique à travers Simulink
 - Restituer le résultat du filtre à travers un DAC

Génération de signal à l'aide de Matlab

Le script Matlab, figure 4, permet de générer un signal sinusoïdal ayant une fréquence de 60 Hz comme nous pouvons l'observer sur la figure 5.



Figure 4 : Script Matlab pour la génération de signal

Sur ce même signal est superposé un signal ayant une fréquence de 1kHz (figure 6). Ce signal sera considéré comme du bruit pour le restant du laboratoire.



Figure 5 : Signal généré avec une fréquence de 60 Hz



Figure 6 : Signal additionné de bruit (1 kHz)

Le signal généré est échantillonné à une fréquence de 10Khz et il est sauvegardé dans un fichier texte sous le nom : "my_data.txt ".

Génération du signal à l'aide du DSP

Afin de pouvoir générer le signal à l'aide du DSP et de l'observer à l'oscilloscope, il va falloir programmer le DSP. Cette étape va s'effectuer à l'aide de Simulink.

Vous devez donc ouvrir une fenêtre Simulink, et y glisser le bloc « Target Setup » :



Figure 7 : Localisation du Bloc Target Setup

Ce bloc permet la configuration du processeur, il faut donc s'assurer que tous les paramètres soient semblables à ceux affichés sur la figure 8 :

- Compilateur GNU ARM
- Nom du microcontrôleur (STM32F407VG) de la carte STM32F4DISCOVERY
- Source de l'horloge 8MHz, qui permet d'avoir une cadence du processeur à 168MHz
- Activation de la compilation automatique et le téléchargement vers le microcontrôleur, car Simulink utilise le compilateur GNU ARM pour « transformer » les blocs en code C pour les transférer vers le DSP. Donc l'application va être executée sur le DSP et non à travers Simulink.

Block Parameters: Target Setup			
stm32f4_target_setup (mask) (link)			
Use this block to setup STM32F4 Target in a Simulink model.			
The sample time of this block is the system base sampletime. It is automatically computed based on sampletime of every block in the system model and is used to configure Systick Counter of the target.			
Parameters			
Compiler GNU ARM			
MCU STM32F407VG (LQFP100); STM32F4DISCOVERY			
Clock Configuration STM32F4DISCOVERY Default (HSEOSC-8MHz/HCLK-168MHz)			
Show memory configuration			
✓ Enable Auto Compile and Download			
Full Chip Erase before Download			
Programmer/Debugger ST-Link			
Show/Edit Control Strings (Recommended for advanced users only)			
Compiler Control String			
-cortex-m4 -mthumb -mthumb-interwork -mfloat-abi=hard -mfpu=fpv4-sp-d16 -ffast-math -Wall -Wextra -Ofast			
Assembler Control String			
Wall -Wextra -Ofast -Wa,-amhls=\$(<:.s=.lst) -DSTACK_SIZE=\$(STACK_SIZE) -DHEAP_SIZE=\$(HEAP_SIZE)			
Linker Control String			
cref,no-warn-mismatch -mcpu=cortex-m4 -mthumb -mthumb-interwork -mfloat-abi=hard -mfpu=fpv4-sp-d16			
Run Application After Download			
Enable Execution Time Profiler None			
Manually set base sample time			
v			
OK Cancel Help Apply			

Figure 8 : Détail du bloc Target Setup

- Activer 'Manually set base sample time'.
- Choisir pour 'Sample time' 0.00001.

Une fois que le bloc est configuré et est déposé dans la fenêtre Simulink, ce dernier va compiler tous les blocs suivants et les transférer vers la carte de développement. Donc il faut s'assurer que celle-ci est branchée et fonctionnelle.

L'objectif de cette partie du laboratoire est de générer le signal construit à l'aide de Matlab, pour cela nous allons utiliser le bloc « Arbitrary Waveform Generator »

Ce bloc est disponible sous l'onglet « On-chip Peripherals » comme illustré sur la figure 9 :



Figure 9 : Bloc Simulink pour le Arbitrary Waveform Generator

Ce bloc n'implémente pas q'un simple DAC, il configure un channel DMA (« *Direct Memory Access* ») entre la mémoire de données vers un DAC. La DMA est un périphérique qui permet de créer un lien direct entre différents périphériques sans utiliser le processeur, ceci peut être comparable à du parallélisme, mais pour des tâches automatiques. Dans notre application, la DMA va permettre d'automatiser le transfert des différents points du signal de la mémoire du microcontrôleur vers le DAC permettant ainsi d'avoir un processeur dédié uniquement au traitement de signal.

Block Parameters: Arbitrary Waveform Generator			
stm32f4_dmadac (mask) (link)			
This block implements Digital to Analog Conveter (DAC) Module as an Arbritary Waveform Generator.			
It uses DMA 12-bits Right Align Mode to output analog voltage at a rate defined by the sample time.			
A user can select either Timer 6 (dafault) or Timer 7 as DAC DMA output trigger source.			
Data or Workspace Variable for DAC Channel X must be a row vector and of type double between 0 - Vref.			
Parameters			
What do to at the end of data array Loop Data			
✓ DAC1 (A4)			
Data or Workspace Variable for DAC1			
[3 2 1]			
☑ DAC2 (A5)			
Data or Workspace Variable for DAC2			
[1 2 3]			
Advanced settings			
Sample time (sec)			
0.001			
OK Cancel Help Apply			

Figure 10 : Détail du bloc Arbitrary Waveform Generator

La figure 10 représente le détail de configuration du bloc Simulink. Pour notre application il est nécessaire de configurer adéquatement le bloc.

Les paramètres du bloc sont les suivants :

- S'assurer que le signal de sortie soit périodique : « Loop Data »
- Laisser DAC1 coché (sortie sur PA4)
- Les données du signal doivent être contenues entre []. Donc soit ouvrir le fichier my_data.txt et copier le contenu pour le coller dans l'espace réservé du bloc ou mettre le nom de la variable contenant le signal dans le workspace de Matlab (par exemple 'added').
- Décocher DAC2 car nous n'avons qu'un signal à générer
- Cocher « advanced settings »
- S'assurer que le timer6 est sélectionné
- Que la tension de référence est 3.3V. La tension de référence d'un DAC ou d'un ADC est la plage de tensions d'entrée ou de sortie pour le périphérique. Sachant que ces

périphériques ont une résolution de 12bits alors la plus petite valeur mesurée ou de sortie sera égale à 3.3/4095.

- Désactiver le tampon de sortie DAC output Buffer Disable.
- et configurer la période d'échantillonnage à 1/10000 (voir Fs de Matlab). Une particularité du périphérique et de multiplier cette fréquence par 2 pour obtenir la fréquence d'échantillonnage voulue. (i,e, sample time = 1/10000/2.)

Il faudra donc s'assurer que les configurations soient semblables à la figure 11.

Block Parameters: Arbitrary Waveform Generator			
stm32f4_dmadac (mask) (link)			
This block implements Digital to Analog Conveter (DAC) Module as an Arbritary Waveform Generator.			
It uses DMA 12-bits Right Align Mode to output analog voltage at a rate defined by the sample time.			
A user can select either Timer 6 (dafault) or Timer 7 as DAC DMA output trigger source.			
Data or Workspace Variable for DAC Channel X must be a row vector and of type double between 0 - Vref.			
Parameters			
What do to at the end of data array Loop Data 🔹			
✓ DAC1 (A4)			
Data or Workspace Variable for DAC1			
.7024210e+00 1.6671953e+00 1.5872095e+00 1.5073129e+00]			
DAC2 (A5)			
✓ Advanced settings			
Timer 6			
Enable Simulation			
Input Vref			
3.3			
DAC Output Buffer Disable			
Sample time (sec)			
1/10000/2			
OK Cancel Help Apply			

Figure 11 : Configuration du générateur de formes



Figure 12 : Schéma Simulink pour la génération de signal

Une fois la configuration du bloc terminée (Figure 12), connebter au Scope les pins GND et PA4.

Compiler et programmer le DSP. Pour cela il faut activer le bouton « Build Model » qui se situe à droite de toutes les icônes (Figure 13).



Figure 13 : Bouton pour la compilation des blocs et la programmation du DSP

À cette étape-ci, tout est automatique :

- génération de code C
- compilation
- Programmation du DSP
- Execution du code sur DSP

Si aucune erreur ne s'est produite lors d'une de ces étapes, alors on peut observer sur l'oscilloscope un signal semblable à la figure 14. Ce signal sera mesuré à la borne PA4.



Figure 14 : Signal généré par le DSP mesuré à l'oscilloscope

Cette étape de génération de signal n'est pas nécessaire dans les applications DSP ou dans des phases de tests, un simple générateur de fonctions peut être utilisé.

Par contre, il peut générer des signaux beaucoup plus complexes que ceux générés par un simple générateur de fonctions.

Sauver le modele pour pouvoir l'acceder a la deuxieme partie.

Acquisition de signal à l'aide du DSP

L'étape suivante est de configurer le périphérique ADC qui va permettre l'acquisition du signal précédemment. Dans le même fichier Simulink, glisser le bloc Regular ADC qui se situe dans la section On-chip Peripherals (figure 15).



Figure 15 : Bloc Simulink pour le Regular ADC

La configuration s'effectue de la manière suivante :

- Choisir le module ADC1
- Le type float pour les données (de 0.0 à 4095.0) Single.
- Un ADC prescaler de 2
- L'entrée de l'ADC doit être la broche PA6
- Et finalement la période d'échantillonnage égale à 1/10000, celle-ci peut être différente tant que l'on respecte le critère de Nyquist.

Un aperçu de la configuration est disponible sur la figure 16.

Source Block Parameters: Re	gular ADC			×
stm32f4_regular_adc (mask) (link)				^
This block implements Regular Analog to Digital Converter (ADC) Module.				
Regardless of the specified data type, the output values are always RAW ADC data between 0 to 4095.				
To convert to voltage, multiply the output values with Vref/4095.				
Parameters				
ADC Module 1			•	
Output Data Type Single				-
ADC Prescaler: 2 (HCLK: 168MHz, fADC: 84MHz, ADC :5.6MSps) 2				•
Read AN0 (Pin: A0)				
Read AN1 (Pin: A1)				
Read AN2 (Pin: A2)				
Read AN3 (Pin: A3)				
🗌 Read AN4 (Pin: A4)				
Read AN5 (Pin: A5)				
Read AN6 (Pin: A6)				
Read AN7 (Pin: A7)				
Read AN8 (Pin: B0)				
Read AN9 (Pin: B1)				
Read AN10 (Pin: C0)				
Read AN11 (Pin: C1)				
Read AN12 (Pin: C2)				
Read AN13 (Pin: C3)				
Read AN14 (Pin: C4)				
🗌 Read AN15 (Pin: C5)				
Read Temperature Sensor (Internal Pin)				
Read VREFINT (Internal Pin)				
Read VBAT (Internal Pin)				
Sample time (sec)				
1/10000				v
	ОК	Cancel	Help	Apply
	UN	Curreer	noip	1999

Figure 16 : Détail du bloc Regular ADC

Désactiver 'Enable Custom Port labels'

Pour pouvoir échantillonner le signal il faut effectuer un « *loopback* », c'est-à-dire réinjecter physiquement le signal de sortie du générateur de signal vers l'entrée de l'ADC. Ceci peut être fait par l'installation d'un cavalier (« jumper ») entre PA4 et PA6.



Figure 17 : Schéma préliminaire Simulink

Transferer le bloc 'Gain' a la figure 17 de : Simulink/Commonly used Blocks. Afin de tester le système, il faut rajouter un bloc « Gain » pour pouvoir convertir les données échantillonnées de l'ADC en tension (Gain = 3.3/4095) et ajouter un autre bloc « Regular DAC » qui sera configuré comme illustré sur la figure 17.

Pour pouvoir s'assurer que les blocs s'effectueront de façon séquentielle et dans l'ordre que l'on désire, il est parfois nécessaire de configurer les blocs Simulink pour qu'il respecte un certain ordre. Cet ordre est souvent défini par le sens des connexions, mais lorsque, comme dans ce cas-ci, certains blocs ne sont pas connectés, alors il est nécessaire de définir l'ordre de priorité.

Pour modifier l'ordre de priorité des blocs, il suffit de cliquer sur le bouton droit de la souris et de sélectionner « Block Properties » et de mettre le bloc le plus prioritaire à 1 et le suivant à 2. Nous pouvons voir les modifications effectuées sur les figures 18 et 19.

Block Properties: (link) Arbitrary Waveform Generator ×		
General Block Annotation Callbacks		
- Usage		
Open Block: Click on the link to open the block. Description: Text saved with the block in the model file.		
Priority: Specifies the block's order of execution relative to other blocks in the same model. Tag: Text that appears in the block label that Simulink generates.		
Open Block: Arbitrary Waveform Generator		
Description:		
Priority:		
1		
Tag:		
OK Cancel Help Apply		

Figure 18 : Configuration du niveau de priorité générateur de signaux

2	Block Properties: (link) Regular ADC	×
General	Block Annotation Callbacks	
Usage		
Open Bloc	:k: Click on the link to open the block.	
Priority: S	pecifies the block's order of execution relative to other blocks in the same model.	
Tag: Text	that appears in the block label that Simulink generates.	_
Open Bloc	k: <u>Regular ADC</u>	
Descriptio	n:	
		-
Priority:		- 11
12 I I I I I I I I I I I I I I I I I I I		
iag:		- 11
	OK Cancel Help Appl	у

Figure 19 : Configuration du niveau de priorité générateur de l'ADC

Après compilation et programmation du DSP les signaux générés par le DSP sont illustrés sur la figure 20. En jaune le signal du générateur de signal et en bleu le signal mesuré par l'ADC sur la broche PA5. (De plus, voir le signal sur PA4 ==PA6).



Figure 20 : Signaux générés et échantillonnés par le DSP, mesurés à l'oscilloscope

Filtrage à l'aide du DSP

L'étape suivante est de rajouter un filtre qui permettra d'éliminer le bruit de 1kHz. Dans le même fichier Simulink, glisser le bloc « *Digital Filter Design* » qui se situe dans la section « *DSP System Toolbox* » (figure 21).



Figure 21 : Bloc Simulink pour le Digital Filter Design

En ouvrant le bloc, nous avons accès à tout les paramètres nécessaires pour configurer différents types de filtres numériques.

Pour éliminer le bruit, un filtre passe-bas peut être utilisé, mais comme la fréquence du bruit est connue, et que la performance des filtres numériques est supérieure à celle des filtres analogiques, l'implémentation d'un filtre coupe-bande peut être très rapidement mise en place :

- Choisir un filtre de type band stop
- Cocher l'onglet pour un filtre IIR de type Butterworth
- Ordre 10
- Unité en Hertz
- Fréquence d'échantillonnage de l'ADC égale à 10kHz
- Fréquence de coupure 1 égale à 700 Hz
- Fréquence de coupure 2 égale à 1400 Hz
- Cliquer sur le bouton Design Filter (en bas de la fenêtre)

La configuration du filtre est illustrée sur la figure 22, et l'allure du filtre s'affiche également sur l'écran lorsque bouton Design Filter est activé.



Figure 22 : Paramètres pour le filtre coupe-bande

Pour s'initier à la logique avec Simulink, le filtre sera activé ou désactivé à l'aide du bouton bleu disponible sur la carte de développement. Dans le même fichier Simulink :

- glisser le bloc « Switch» qui se situe dans la section « Signal Routing » (figure 23).
- glisser le bloc « *Debounce*» qui se situe dans la section « *On-chip peripherals/IO* » (figure 24).
- glisser le bloc « *Digital Input*» qui se situe dans la section « *On-chip peripherals/IO* » (figure 24).
- Configurer l'entrée PA0 (Pin 0 voir Fig. 25) connectée au bouton poussoir bleu (« User »)

- Période d'échantillonnage à 0.01s, il n'est pas nécessaire de réduire cette période, car cela surchargera le processeur (figure 25).
- Configurer le bloc Debounce en Toggle latch et le prescale à 4 (figure 26). Le Toggle latch permet à chaque appui du bouton poussoir de changer d'état 0 ou 1 et de maintenir l'état.

4 🎦 Simulink	
Commonly Used Blocks	
Continuous	
Discontinuities	
Discrete	
Logic and Bit Operations	
Lookup Tables	
Math Operations	
Model Verification	
Model-Wide Utilities	
Ports & Subsystems	
Signal Attributes	x+x→ Switch
Signal Routing	1

Figure 23 : Bloc Simulink pour Switch

Switch Choisir > = 0 Threshold 0.

4	STM32F4 Target Boot Loader Device Configuration On-chip Peripherals ADC CAN CRC DAC DCMI FLASH		
	FSMC I2C	(ional) Debounce	Streets.
	Ю		

Figure 24 : Blocs Simulink pour Debounce et Digital Input

Source Block Parameters: Digital Input		
stm32f4_digital_input (mask) (link)		
This block implements Digital Input Module.		
Parameters		
Port A		
Speed (MHz) 100		
Type (Pull-Up/Pull-Down) None 🔹		
✓ Use Pin 0		
Use Pin 1		
Use Pin 2		
Use Pin 3		
Use Pin 4		
Use Pin 5		
Use Pin 6		
Use Pin 7		
Use Pin 8		
Use Pin 9		
Use Pin 10		
Use Pin 11		
Use Pin 12		
Use Pin 13		
Use Pin 14		
Use Pin 15		
Sample time (sec)		
0.01		
Enable custom port labels		
OK Cancel Help Apply		

Figure 25 : Configuration du bloc Digital Input

Function Block Parameters: Debounce		
(mask) (link)		
Mode, 1. Normal: - Output state is 0 (FALSE) when input is steady at 0 for a Debounce period (or longer). - Output state is 1 (TRUE) when input is steady at non-zero for a Debounce period (or longer).		
 Single Pulse: Output state will generate a single pulse (one time-step) when input signal change from 0-to-NonZero and steady for a Debounce period. 		
 Toggle Latch: Output state will toggle when input signal change from 0-to-NonZero and steady for a Debounce period 		
Debounce period, - Debounce period (sec) = Debounce count * Sample time (sec)		
Note: the block is Masked Subsystem, to see its function: from the popup menu on mouse right-click, select "Look Under Mask".		
Parameters		
Mode Toggle Latch 🔹		
Prescale (Debounce count) 4		
OK Cancel Help Apply		

Figure 26 : Configuration du bloc Debounce

Lorsque les blocs sont configurés, il faut relier les blocs comme illustré sur la figure 27. L'algorithme est le suivant :

- Si le bouton est appuyé une fois, alors le bloc Debounce maintient la sortie à 1 si son état passé est 0, et la sortie vers le DAC est la sortie du filtre
- Si le bouton est appuyé, le filtre est contourné si l'état passé du bloc Debounce est 1 et son nouvel état est 0.



Figure 27 : Schéma Simulink représentant le système à implémenter

Le résultat du filtre est illustré sur la figure 28.



Figure 28 : Signal mesuré et signal filtré

Conclusion

L'objectif de ce laboratoire est de mettre en pratique la théorie de l'analyse des signaux et d'introduire les notions de :

- DSP
- Programmation de DSP sous Simulink
- Génération de signal
- Acquisition de données
- Filtrage
- Logique pour activation ou désactivation du flux de données

Partie 2

Générateur de sinusoïdes

Objectif du laboratoire

L'objectif du laboratoire est de s'initier à l'utilisation du matériel pour générer un signal sinusoïdal.

- La première étape consiste à utiliser la carte comme un générateur de sinus à l'aide du bloc Sine Wave
- La deuxième étape à utiliser la carte comme un générateur de sinus à l'aide du bloc Discrete Filter

L'utilisation de Simulink en mode simulation, tout le long du laboratoire, permet de rapidement vérifier les résultats avant de programmer la carte

Génération de sinusoïde à l'aide du bloc Simulink : Sine Wave

Afin de pouvoir générer le signal à l'aide du DSP et de l'observer à l'oscilloscope, il va falloir programmer le DSP. Cette étape va s'effectuer à l'aide de Simulink.

Vous devez donc ouvrir une fenêtre Simulink, et y glisser le bloc « Target Setup » :



Figure 29 : Localisation du Bloc Target Setup

Ce bloc permet la configuration du processeur, il faut donc s'assurer que tous les paramètres soient semblables à ceux affichés sur la figure 8 :

- Compilateur GNU ARM
- Nom du microcontrôleur (STM32F407VG) de la carte STM32F4DISCOVERY
- Source de l'horloge 8MHz, qui permet d'avoir une cadence du processeur à 168MHz
- Activation de la compilation automatique et le téléchargement vers le microcontrôleur, car Simulink utilise le compilateur GNU ARM pour « transformer » les blocs en code C pour les transférer vers le DSP. Donc l'application va être executée sur le DSP et non à travers Simulink.
- S'assurer que Manually set base sample time est désactivé (pour le moment)

Block Parameters: Target Setup							
stm32f4_target_setup (mask) (link)							
Use this block to setup STM32F4 Target in a Simulink model.							
The sample time of this block is the system base sampletime. It is automatically computed based on sampletime of every block in the system model and is used to configure Systick Counter of the target.							
Parameters							
Compiler GNU ARM							
MCU STM32F407VG (LQFP100); STM32F4DISCOVERY							
Clock Configuration STM32F4DISCOVERY Default (HSEOSC-8MHz/HCLK-168MHz)							
Show memory configuration							
✓ Enable Auto Compile and Download							
Full Chip Erase before Download							
Programmer/Debugger ST-Link 👻							
Show/Edit Control Strings (Recommended for advanced users only)							
Compiler Control String							
=cortex-m4 -mthumb -mthumb-interwork -mfloat-abi=hard -mfpu=fpv4-sp-d16 -ffast-math -Wall -Wextra -Ofast							
Assembler Control String							
Wall -Wextra -Ofast -Wa,-amhls=\$(<:.s=.lst) -DSTACK_SIZE=\$(STACK_SIZE) -DHEAP_SIZE=\$(HEAP_SIZE)							
Linker Control String							
cref,no-warn-mismatch -mcpu=cortex-m4 -mthumb -mthumb-interwork -mfloat-abi=hard -mfpu=fpv4-sp-d16							
Run Application After Download							
Enable Execution Time Profiler None							
Manually set base sample time							
OK Cancel Help Apply							

Figure 30 : Détail du bloc Target Setup

Le bloc Simulink *Sine Wave* est un bloc qui permet de générer des signaux de forme sinusoïdales. La description détaillée du bloc est disponible sur le site web de Mathworks :

http://www.mathworks.com/help/simulink/slref/sinewave.html

Une autre façon d'obtenir de l'information est d'y accéder à partir de l'aide de Matlab⁴



Figure 31 : Aperçu de l'aide Matlab

Le bloc Sine Wave est disponible sous l'onglet Simulink/Sources :

Simulink Commonly Used Blocks Continuous	J. M	Band-Limited White Noise	^ ₩}	Chirp Signal	\bigcirc	Clock	1	Constant
Discontinuities Discrete Logic and Bit Operations	J.¶>	Counter Free- Running	"71/I>	Counter Limited	12:34	Digital Clock	Silben oSign Positive •	Enumerated Constant
Lookup Tables Math Operations Model Verification	untitled.mat >	From File	smin	From Workspace	F	Ground		In1
Model-Wide Utilities Ports & Subsystems Signal Attributes	∭}	Pulse Generator		Ramp	\mathbb{M}	Random Number	<i>M</i> ►	Repeating Sequence
Signal Routing Sinks Sources	\bigwedge	Repeating Se- quence Interp	հՆ	Repeating Sequence Stair		Signal Builder	00	Signal Generator
Additional Math & Discrete Aerospace Blockset	Ð	Sine Wave		Step	M	Uniform Ran- dom Number		

Figure 32 : Localisation du Bloc Sine Wave

⁴ Assurez-vous de toujours prendre connaissance du fonctionnement interne des blocs ou des fonctions que vous utilisez!

Le bloc Regular DAC est disponible sous l'onglet :

4	Waijung Blockset Custom Code Hardware Modules Misc Profiler STM32F4 Target Boot Loader Device Configuration On-chip Peripherals	
	CAN	
	CRC	Regular DAC
	DAC	

Figure 33 : Localisation du Bloc Regular DAC

Et doit être configuré comme suit :

Sink Block Parameters: Regular DAC ×
stm32f4_regulardac (mask) (link)
This block implements Digital to Analog Conveter (DAC) Module.
Parameters
Input Type Volts (double)
DAC1 (A4)
✓ DAC2 (A5)
✓ Advanced settings
Input Vref
3.3
DAC Output Buffer Disable 👻
Sample time (sec)
-1
Enable custom port labels
OK Cancel Help Apply



Après création d'une fenêtre Simulink et avoir déposé les blocs ci-dessus, vous devez obtenir une fenêtre semblable à celle-ci :



Figure 35 : Schéma Simulink pour la génération de signal

Supposons que l'on veuille générer un signal sinusoïdal dont la fréquence est de 100 Hz. Pour cela il faut configurer le bloc *Sine Wave* comme tel :

- Sine Type : Time Based
- Time : Use simulation time
- Amplitude : 1V
- Bias : 1.5V (le sinus va être centré sur cette valeur)
- Frequency : 2*pi*100 (100 Hz)
- Phase : 0
- Sample time : 1/(100 * 10)

Source Block Parameters: Sine Wave							
Sine Wave							
Output a sine wave:							
O(t) = Amp*Sin(Freq*t+Phase) + Bias							
Sine type determines the computational technique used. The parameters in the two types are related through:							
Samples per period = 2*pi / (Frequency * Sample time)							
Number of offset samples = Phase * Samples per period / (2*pi)							
Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.							
Parameters							
Sine type: Time based 🔹							
Time (t): Use simulation time 🔹							
Amplitude:							
1							
Bias:							
1.5							
Frequency (rad/sec):							
2*pi*100							
Phase (rad):							
0							
Sample time:							
1/(100*10)							
☑ Interpret vector parameters as 1-D							
OK Cancel Help Apply							

Figure 36 : Configuration du bloc Sine Wave

Pour visualiser le signal avant de programmer la carte DSP, nous allons utiliser Simulink en mode simulateur, pour cela un bloc *Scope* a été rajouté au schéma :

- Simulation stop time : 1/100, donc une période de signal
- Appuyez sur le bouton RUN (cercle vert avec triangle noir : PLAY)



Figure 37 : Configuration de la simulation

3		Scope		 ·	- 🗆 🗙
🚍 @ 🔍 🕫 & 🖾 🅦 🖏 🔛 🗎 🐂					۲ د
2.5					
2					·····
1.5					
0.5					
0.00 0.001 0.002					

Ouvrir le bloc Scope et on obtient le signal ci-dessous :

Figure 38 : Sinus de 100 Hz en mode simulation

Pour programmer la carte :



Figure 39 : Bouton pour la compilation des blocs et la programmation du DSP



Figure 40 : Signal généré par le DSP mesuré à l'oscilloscope

Donc le signal mesuré à l'oscilloscope concorde avec celui obtenu par simulation, sauf que la résolution du signal est faible.

Si l'on revient au signal généré par simulation on remarque que le signal change par paliers et nous en avons 10 par période de signal, ceci est caractéristique des signaux discrets.

Pour améliorer la résolution nous allons modifier le *sample time* pour obtenir 100 échantillons par période :

- Sample time : 1/(100 * 100)

On relance la simulation et on reprogramme la carte, comme précédemment, et l'on obtient les figures suivantes avec une résolution accrue.



Figure 41 : Sinus de 100 Hz en mode simulation avec 100 échantillons



Figure 42 : Signal généré par le DSP mesuré à l'oscilloscope

Comme nous pouvons le voir, il est assez facile de générer un signal sinusoïdal à l'aide de Simulink, par contre il est nécessaire de connaitre toute la théorie qui permet de générer ce signal. Par contre cette simplicité a un coût : la génération du code de programmation du DSP n'est pas optimisée (temps de calcul trop long) et nous aurons donc une bande passante réduite à 2kHz.

Par contre si nous utilisons le module *Arbitrary Waveform Generator*, module utilisant des techniques optimisées pour DSP la bande passante est de 2MHz.

Génération de sinusoïde à l'aide du bloc Simulink : Discrete filter

Cette partie du laboratoire s'inspire de la théorie vue en classe, nous pouvons construire un générateur de sinusoïdes comme un système à temps discrète dont la réponse impulsionnelle est la sinusoïde désirée. Rappelons-nous alors que la fonction de transfert H(z) du filtre n'est que la transformée en Z de la réponse impulsionnelle. En particulier, la transformée de la séquence :

$$x[n] = Asin(\beta n)$$

Est

$$X(z) = \frac{z^{-1}sin\beta}{1 - 2cos\beta z^{-1} + z^{-2}}$$

Et celle de

$$x[n] = Acos(\beta n)$$

Est

$$X(z) = \frac{1 - z^{-1} \cos\beta}{1 - 2\cos\beta z^{-1} + z^{-2}}$$

Donc, en appliquant une impulsion à l'entrée du filtre, ce dernier produira la sinusoide désirée.

En se basant sur les modules Simulink précèdement utilisés, nous allons utiliser deux nouveaux blocs : *Discrete Impulse* et *Discrete Filter*.



Figure 43 : Localisation du Bloc Discrete Impulse



Figure 44 : Localisation du Bloc Discrete Filter

Et reproduire le schéma Simulink suivant :



Figure 45 : Schéma Simulink pour la génération de signal

La configuration des blocs dépendra de la fréquence de la sinusoide à générer, comme dans l'exercice précédent nous allons nous fixer une fréquence de 100Hz.

Donc pour générer un sinus, le bloc Discrete Filter devra avoir l'allure suivante :

$$X(z) = \frac{z^{-1}sin\beta}{1 - 2cos\beta z^{-1} + z^{-2}}$$

La configuration aura l'allure suivante :

1	Function Block Parameters: Discrete Filter	x					
Discrete Filter							
Independently filter each channel of the input over time using a discrete IIR filter. Specify the numerator and denominator coefficients in ascending order of powers of 1/z.							
A DSP System T	Toolbox license is required to use a filter structure other than Direct form II.						
Main Data T	Types						
Filter structure:	Direct form II transposed 🔹						
Data	Source Value						
Numerator:	Dialog [0 sin(omega0)]						
Denominator:	Dialog [1 -2*cos(omega0) 1]						
Initial states:	Dialog 👻 0						
External reset:	None 🔻						
Input processin	ng: Columns as channels (frame based)						
✓ Optimize by skipping divide by leading denominator coefficient (a0)							
Sample time (-1	for inherited): -1						
	OK Cancel Help Apply						

Figure 46 : Configuration du bloc Discrete Filter

Dans le bloc Simulink, la variable «omega0» est à calculer selon la démarche suivante :

Supposons que le β soit de 100Hz et que notre fréquence d'échantillonnage soit de 1000 Hz soit de 10 échantillons par période du sinus.

Alors; omega0 = 2 * pi * 100 * 1/1000 = pi/5

Alors le bloc Discrete Impulse sera configuré comme sur la figure suivante :

Source Block Parameters: Discrete Impulse						
Discrete Impulse (mask) (link)						
Output a discrete unit impulse. The impulse will be offset by the number of samples in the Delay parameter.						
Main Data Types						
Delay (samples):						
0						
Sample time:						
1/1000						
Samples per frame:						
1						
OK Cancel Help Apply						

Figure 47 : Configuration du bloc Discrete Impulse

Une simulation rapide, avec les mêmes configurations que l'exercice précédent, permet de valider les résultats :



Figure 48 : Résultat de la Simulation pour le bloc Discrete Filter à 10 échantillons

Pour augmenter la résolution de 100 échantillons par période de sinus, notre fréquence d'échantillonnage doit être de 10000 Hz.



Alors; omega0 = 2 * pi * 100 * 1/10000 = pi/50

Figure 49 : Résultat de la Simulation pour le bloc Discrete Filter à 100 échantillons

Après programmation de la carte DSP, nous pouvons remarquer la similarité des résultats.

Conclusion

Le laboratoire permet de générer à l'aide de Simulink des sinusoïdes. Deux approches ont été testées :

La première basée sur le bloc Simulink Sine Wave, qui donne accès à tous les paramètres d'une fonction sinusoidale.

La seconde approche est basée sur la théorie, à partir de la réponse impulsionnelle d'un filtre, nous pouvons générer un signal.

Nous avons également pu voir la notion de résolution, avec le nombre d'échantillons par période.

Les méthodes utilisées dans ce laboratoire, sont simples, mais elles demandent des calculs intensifs, car les blocs Simulink ne sont pas optimisés pour des systèmes embarqués. Donc l'utilisation de méthodes plus optimisées seront utilisées si l'on veut sugmenter la fréquence des sinus.