|  |  |  |  |
| --- | --- | --- | --- |
| Charte et Logo de l’entreprise | | | |
| **PFE : « Traitement de signal audio embarqué temps réel sur carte STM32 »**  **FIPA 2021** | | | |
|  | ENSTA Bretagne  2 rue F. Verny  29806 Brest Cedex 9, France  JÉZÉGOU Pierre-Yves,  [pierre-yves.jezegou@ensta-bretagne.org](mailto:pierre-yves.jezegou@ensta-bretagne.org) |  | Contact :  Reynet Olivier,  [olivier.reynet@ensta-bretagne.fr](mailto:olivier.reynet@ensta-bretagne.fr)  Probst Irvin,  [irvin.probst@ensta-bretagne.fr](mailto:irvin.probst@ensta-bretagne.fr) |

# Remerciements

# Résumé

# Abstract

Sommaire

[Remerciements 2](#_Toc88482028)

[Résumé 3](#_Toc88482029)

[Abstract 3](#_Toc88482030)

[Sommaire 4](#_Toc88482031)

[Introduction *(3e livrable, 2/3 de page)* 5](#_Toc88482032)

[1. Objectifs du projet 6](#_Toc88482033)

[2. Plateforme Embarquée. 7](#_Toc88482034)

[2.1. Carte de développement 7](#_Toc88482035)

[2.2. Chaine de capture 7](#_Toc88482036)

[2.2.1. Le microphone 7](#_Toc88482037)

[2.3. Sortie audio 9](#_Toc88482038)

[3. Conversion PDM → PCM 12](#_Toc88482039)

[3.1. Les échantillons 12](#_Toc88482040)

[3.1.1. PCM 12](#_Toc88482041)

[3.1.2. PDM 13](#_Toc88482042)

[3.2. Chaine de filtrage 13](#_Toc88482043)

[3.3. Intégration au microcontrôleur 16](#_Toc88482044)

[4. Les démonstrateurs 19](#_Toc88482045)

[4.1. Présentation des démonstrateurs : 19](#_Toc88482046)

[4.1.1. « Parrot » 19](#_Toc88482047)

[4.1.2. « Digital recorder » 20](#_Toc88482048)

[4.1.3. « Direct output » 21](#_Toc88482049)

[Bibliographies 23](#_Toc88482050)

[Glossaires des termes techniques 23](#_Toc88482051)

[Table des figures 24](#_Toc88482052)

[Annexes 25](#_Toc88482053)

[Annexe 1 : Logigrammes des États démonstrateur "Parrot" 25](#_Toc88482054)

[Annexe 2 : Logigrammes des États démonstrateur "Digital Recorder" 25](#_Toc88482055)

[Annexe 3 : Logigrammes des États démonstrateur "Direct Output" 25](#_Toc88482056)

# Introduction *(3e livrable, 2/3 de page)*

* Introduction générale du projet
* Présentation de la structuration du rapport

# Objectifs du projet

Ce projet de fin d’études a pour objectif de préparer de futurs projets étudiants de l’ENSTA Bretagne centrés sur l’acquisition audio sur plateforme STM32 avec des microphones PDM. On s’intéresse donc à la cible STM32 qui est aujourd’hui un standard de l’industrie embarquée. Ainsi qu’aux microphones numériques PDM qui représentent une innovation intéressante par leurs faibles bruits et donc excellent rapport signal sur bruit.

Le projet se concentre donc sur la production d’une configuration clé en main de la carte STM32 afin :

* d’acquérir un signal numérique audio en utilisant le DMA du processeur et en effectuant le filtrage des échantillons PDM en provenance du microphone
* de générer un son préalablement acquis via le microphone PDM sur le DAC du processeur

Pendant le projet, on s’intéressera plus particulièrement aux réglages du microphone afin de maitriser :

* l’échantillonnage des signaux PDM et PCM ainsi que les fréquences associées,
* le filtrage PDM → PCM afin de s’affranchir de la librairie « boite noire » fournie par ST Microelectronics pour le filtrage,
* la gestion des formats audio sur la carte cible (production de fichiers WAV),
* le fonctionnement global de la chaine d’acquisition,

Les productions techniques du projet se matérialisent par la production de démonstrateurs mettant en œuvre la chaine d’acquisition. Parmi ces démonstrateurs, au moins un aura le comportement suivant :

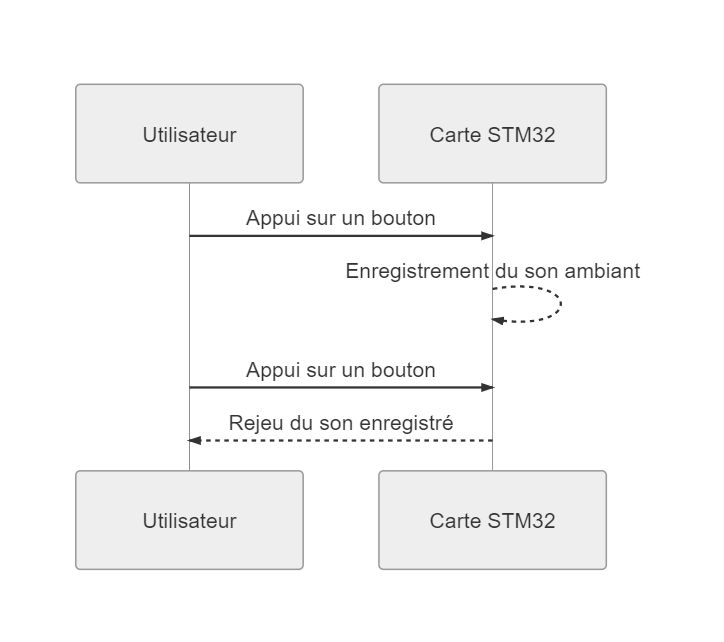
.

Figure 1 : Comportement du démonstrateur obligatoire

# Plateforme Embarquée.

## Carte de développement

Le projet se base sur la carte de découverte commercialisée par ST Microelectronics : "STM32F429I-DISC1". Cette carte intègre un microcontrôleur 32 bits STM32F429xx. Ce microcontrôleur fait partie de la série de microcontrôleurs hautes performances : STM32F4. Cette série se base sur un cœur ARM Cortex M4. Cette carte permet de se familiariser avec l’architecture ainsi que de concevoir des solutions intégrant un microcontrôleur de la famille STM32 qui est un standard de l’industrie embarqué. Cette carte intègre en plus du microcontrôleur, une interface de programmation ST-LINKV2, un écran LCD 2.4", plusieurs LED de débogage, 2 boutons poussoirs, un port micro-USB type B compatible OTG, une puce de SDRAM de 64 Mbit et un gyroscope MEMS.

Une image contenant texte, équipement électronique

Description générée automatiquement

Figure 2 : Carte Cible (STM32F429I-DISC1)

## Chaine de capture

### Le microphone

Le projet a pour objectif principal de lever les incertitudes sur l’utilisation de microphone PDM pour la capture audio sur une plateforme utilisant un microcontrôleur STM32. En effet, les microphones PDM sont une innovation intéressante pour la capture audio, car le transducteur MEMS permet de reproduire fidèlement un son en fournissant un encombrement minimum, de plus la numérisation du signal au plus près du transducteur permet un forte isolation par rapport aux bruits électromagnétiques ambiants. Un microphone PDM est généralement structuré comme suit :

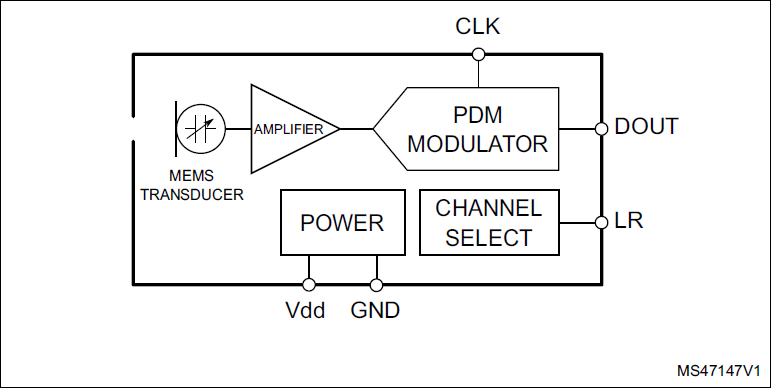


Figure 3 : Schéma bloc typique d’un microphone PDM

Sur la Figure 3, on a plusieurs blocs qui remplissent un rôle :

* MEMS Transducer :

Le transducteur MEMS est un condensateur dont la capacité varie en fonction des variations de pression locale de l’air, qui est caractéristique au déplacement d’une onde sonore.

* Amplifier :

L’amplificateur amplifie le signal en provenance du transducteur afin de fournir un signal adapté au modulateur PDM.

* PDM Modulator :

Le modulateur PDM convertie à chaque front de l’horloge (CLK) le signal analogique en provenance de l’amplificateur en un échantillon PDM (DOUT)

* Channel Select

Le sélecteur de canal détermine si modulateur PDM convertie le signal sur un front montant ou descendant de l’horloge et donc définir si le microphone travail sur le canal gauche ou le canal droit, quand on utilise deux microphone PDM sur un unique flux, afin de reproduire un son en stéré

* Power :

Le bloc Power correspond à l’alimentation du microphone PDM

Pour le projet, nous travaillons dans une configuration mono pour du microphone PDM ce qui nous donne les connexions électriques suivantes :

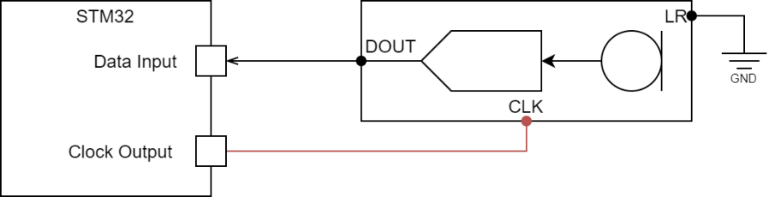


Figure 4 : Connexion du Microphone PDM en configuration mono

Notons que sur la Figure 4 le signal "LR" qui correspond au sélecteur gauche/droite est connecté directement à GND, indiquant que notre microphone travaille sur le canal gauche. Cependant, car nous travaillons en configuration mono, le canal utilisé par le microphone n’a pas d’importance.

Dans notre application, le microcontrôleur intègre un périphérique SAI (Serial Audio Interface). Ce périphérique génère les signaux nécessaires pour le contrôle des microphones PDM, ainsi que l’acquisition des échantillons PDM. On couple ce périphérique au DMA du microcontrôleur qui permet d’acquérir les échantillons en tache de fond. Ce qui permet de filtrer le signal PDM pour le convertir en PCM uniquement quand le DMA nous indique que suffisamment d’échantillons ont été chargés en mémoire. Laissant des temps d’inactivité au microcontrôleur pour exécuter d’autre tâches. La chaine de complète capture est représentée en Figure 5.

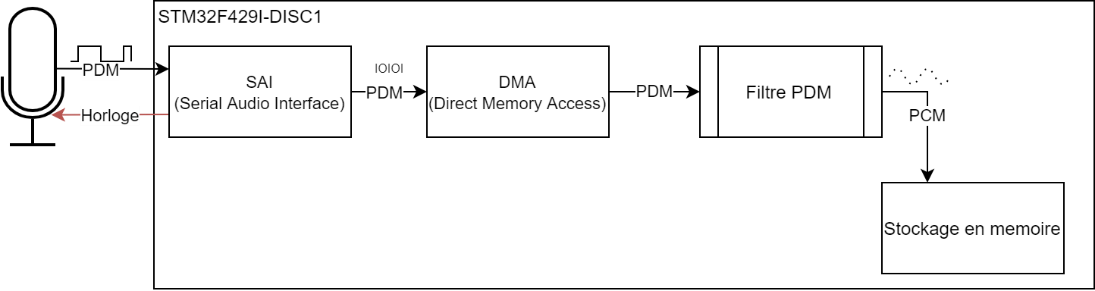


Figure 5 : Chaine de capture

## Sortie audio

Un deuxième objectif de projet et de reproduire les sons captés sur un haut-parleur/casque. On a étudié précédemment l’aspect capture des données du son. Pour l’aspect reproduction on utilise le DAC intégré au microcontrôleur afin de recréer le signal analogique correspondant aux sons captés. Le DAC intégré à notre microcontrôleur a deux canaux de 12 bits, dans la configuration utilisée par notre carte il peut générer une tension comprise en 0.2V et → avec dans notre cas ce qui nous donne une amplitude maximale :

Et donc un quantum :

Ces deux valeurs du DAC nous indiquent que pour adapter le signal PCM filtrer il faut produire des échantillons PCM sur 12 bits. Comme la sortie du DAC est unipolaire (0-3V) il faut ajouter un offset de 2047 () afin de positionner le « zéro » de notre signal au milieu de l’excursion du DAC. Ainsi le haut-parleur agissant comme un filtre passe-bande dont la bande passante et comprise entre 20 Hz et 20 kHz gomme cet offset nécessaire à la reproduction des valeur négative du signal audio.

Pour générer un son on utilise un haut-parleur dont l’impédance est très faible (), c’est pourquoi il faut s’assurer que le courant fournit par le DAC ne peut pas excéder le courant maximum qui peut être fournis par le microcontrôleur :

et

Une solution simple serait de connecter le haut-parleur entre la sortie du DAC et la masse comme sur le montage suivant :

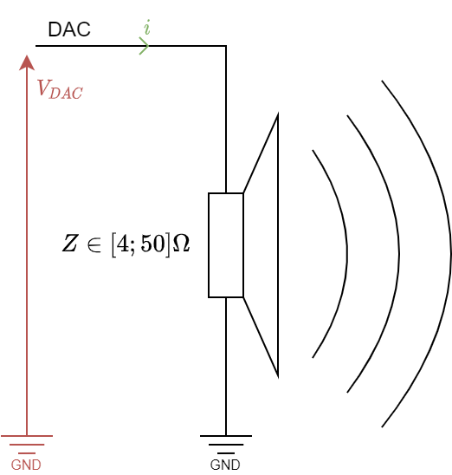


Figure 6 : Montage direct d’un haut-parleur sur le DAC

Dans ce montage on a pour la tension maximale de sotie du DAC et une impédance du haut-parleur  :

Dans ce cas le courent à fournir et très supérieur au maximum spécifier. Il faut donc trouver solution où . J’ai donc fait le choix de placé en série du haut-parleur une résistance . Créant ainsi le montage suivant :

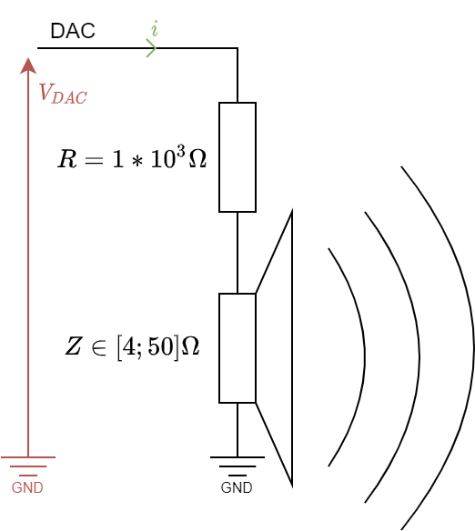


Figure 7 : Branchement d’un haut-parleur en série avec une résistance de 1 kΩ sur le DAC

Dans ce montage on obtient pour   une impédance du haut-parleur :

Ce courant maximum donne une marge suffisante par rapport aux capacités du microcontrôleur pour s’assurer que le risque de destruction des ports du DAC est minimum.

Pour simplifier les connexions/déconnexions d’un haut-parleur/casque, j’ai donc fabriqué un câble que l’on connecte sur les connecteurs Dupont de la carte et fournit une prise jack 3,5 mm stéréo préchargée d’une résistance de sur chacun des deux canaux. Ce câble suit le schéma suivant :

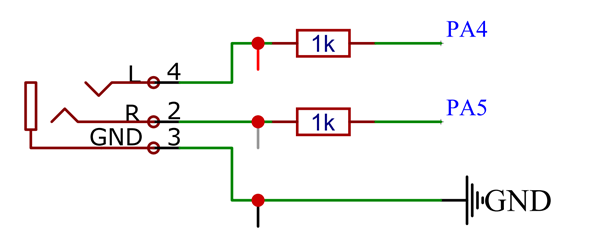
.

Figure 8 : Schéma Électrique du câble DAC → Jack 3,5 mm

On remarque sur le schéma que j’ai fait sortir des fils pour connecter des sondes côté haut-parleur afin de mesurer les signaux reproduits par le DAC.

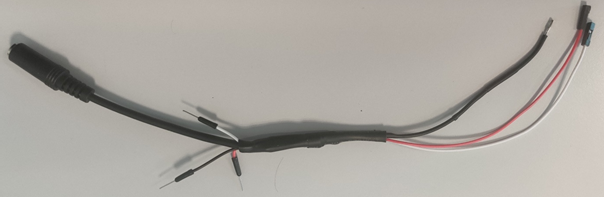


Figure 9 : Câble DAC → Jack 3.5mm

Le câble ainsi fabriqué utilise le code couleur des connecteurs RCA (Blanc → Gauche ; Rouge → Droit) pour les fils transportant le signal et le fil Noir correspondant à la masse

Du point de vue du logiciel, le DAC est configuré pour convertir l’échantillon suivant de chaque canal quand il reçoit l’évènement en provenance Timer 2 du microcontrôleur. On peut configurer la fréquence de cet évènement via deux valeurs:

* "Prescaler" (PSC)
* "Counter Period" (ARR)

On calcule ces deux valeurs grâce à l’expression suivante :

Où correspond à la fréquence d’entrée du Timer.

Par exemple si l’on veut que le DAC ait une fréquence d’échantillonnage de 48 kHz () avec une fréquence . On fixe et donc on obtient :

Pour fournir les échantillons à reproduire au DAC on utilise un buffer circulaire par canal du DAC. Les échantillons de ces buffers sont ensuite fournis au DAC à la fréquence par le DMA du microcontrôleur. Le DMA Informe le programme principal de sa position dans les buffers par deux interruptions, une première qu’il lèvent quand il a atteint la moitié du buffer et une seconde quand il atteint la fin de celui-ci. Ces Interruptions sont ensuite traitées par le programme principal afin d’y charger de nouveaux échantillons, faisant ainsi en sorte qu’au moins une moitié du buffer soit prête à être reproduite par le DAC.

# Conversion PDM → PCM

## Les échantillons

### PCM

Le PCM (Pulse Code Modulation) est une représentation numérique d’un signal analogique résultant de sa numérisation par échantillonnage. En effet, pour construire un signal PCM à partir d’un signal analogique, un convertisseur analogique numérique (ADC) mesure la valeur du signal analogique à une fréquence appelée fréquence d’échantillonnage. Le résultat de cette mesure est alors stocké numériquement. Un signal PCM est donc défini par deux valeurs :

* sa fréquence d’échantillonnage ,
* le nombre de bits utilisés pour stocker la mesure .

Cette deuxième valeur influe sur la précision de la valeur mesurée. En effet, plus est important, plus le signal échantillonné aura de valeurs possibles et donc plus la mesure sera précise.

Prenons par exemple le signal suivant que l’on observe pendant 1 seconde :

Avec  Hz, et

Notre ADC peut mesurer une valeur appartement à l’intervalle qu’il stocke dans un entier encodé sur 4 bits (0000 correspond à la valeur 0 et 1111 à la valeur 1). L’ADC effectue ses mesures à une fréquence . Cela nous donne donc les signaux suivants :

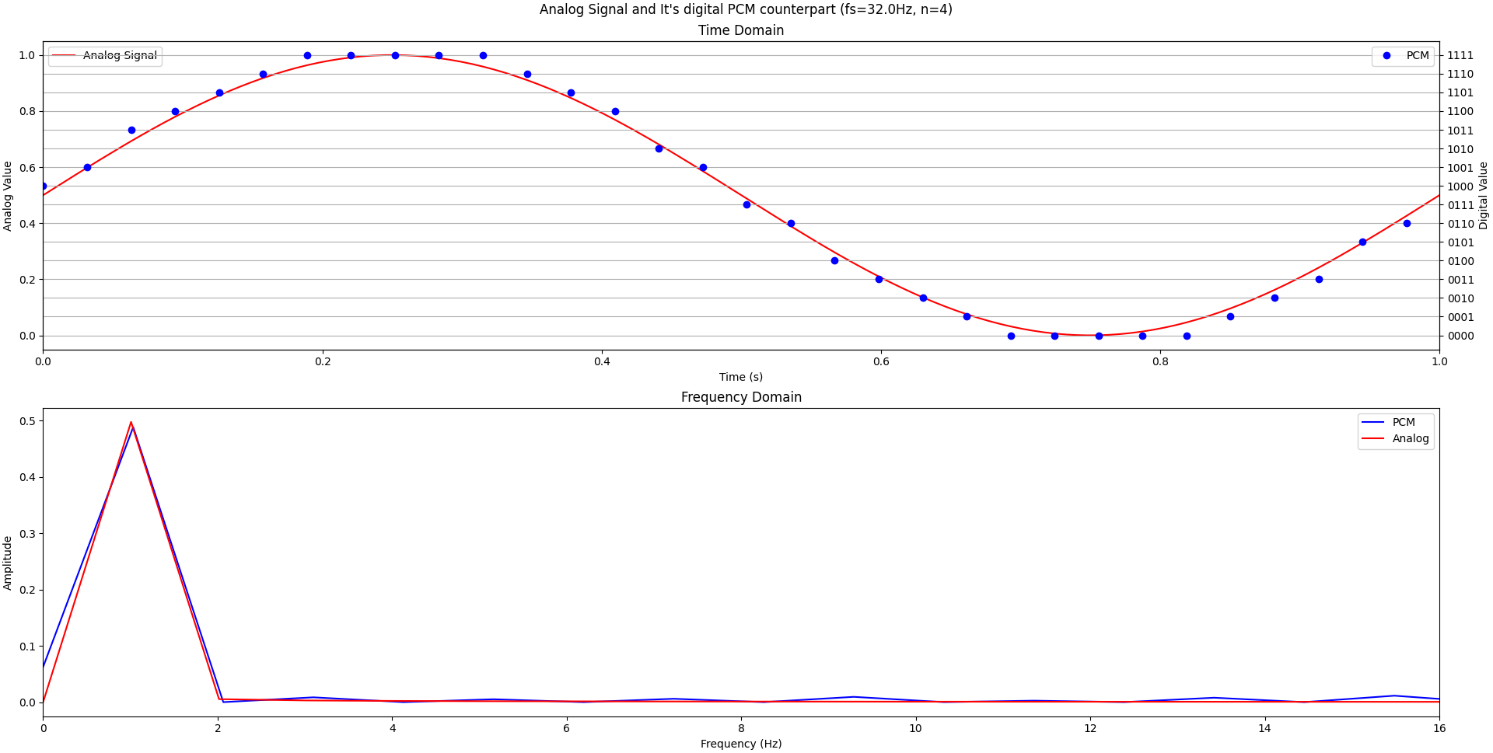


Figure 10 : Signal Analogique et sa conversion numérique en PCM (, )

Les sons PCM sont généralement encodés sur 16 bits (65 536 valeurs possibles) et échantillonnés à pour les CD audio et pour la bande sonore d’un fichier vidéo.

### PDM

Le PDM (Pulse Density Modulation) et une représentation numérique d’un signal analogique. Cette représentation se caractérise par un flux de bits à haute fréquence, flux dans lequel l’amplitude du signal analogique et déterminer à partir de la concentration de bit a 1 ou à 0.

Reprenons notre signal analogique que l’on observe pendant 1 seconde :

Avec  Hz, et

Le signal PDM sera alors le suivant :

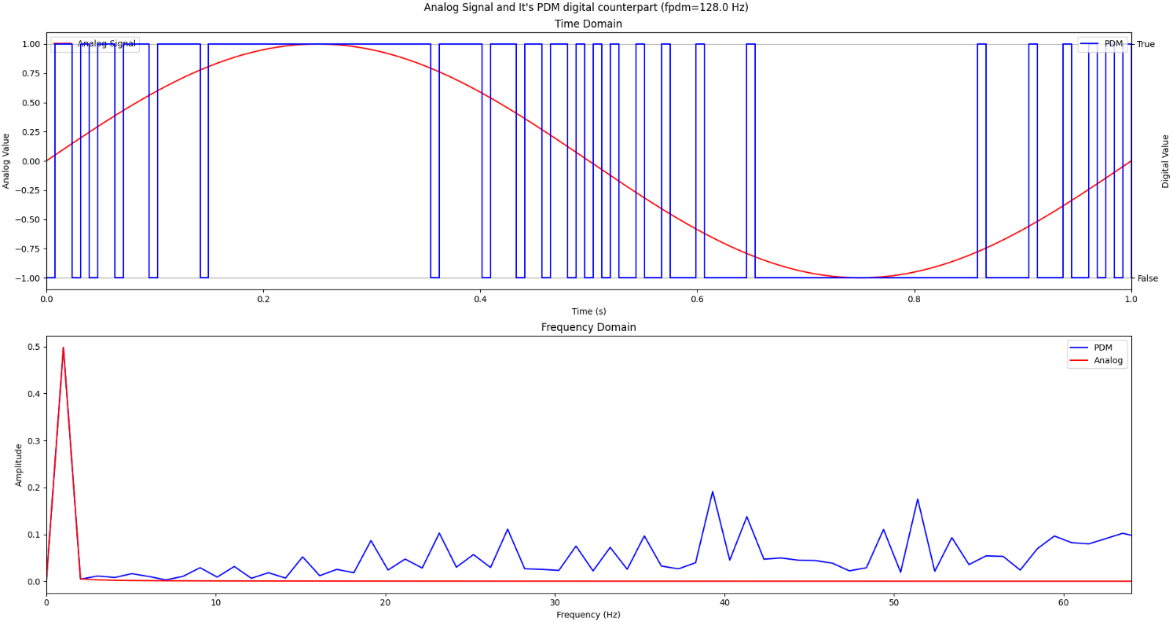


Figure  : Signal Analogique et sa conversion numérique en PDM ()

On distingue qu’une grande valeur positive est caractérisée par une forte concentration locale d’échantillons PDM à 1 et une grande valeur négative est caractérisé par une forte concentration locale d’échantillons PDM à 0. Un signal PDM est caractérisé par la fréquence du flux qui lui est associé. Notons que par rapport à la fréquence d’échantillonnage d’un signal PCM celle d’un signal PDM doit être bien plus élevée pour qualité similaire. Généralement la fréquence du flux PDM est entre 48 et 128 fois supérieur à la fréquence d’échantillonnage PCM.

## Chaine de filtrage

Le passage d’un signal PDM échantillonné à haute fréquence () vers un signal PCM échantillonné à plus basse fréquence () se fait par le filtrage du signal PDM par un filtre passe bas ayant une fréquence de coupure maximale . Une fois le signal PDM filtré, on le sous-échantillonne afin de faire correspondre sa fréquence d’échantillonnage à celle recherchée pour le signal audio (). Ce sous-échantillonnage est réalisé avec un facteur , généralement .

On applique ensuite au signal un offset et un gain afin de l’adapter aux niveaux de sorties recherchés pour le signal PCM.

La chaine de filtrage correspondante est représentée en Figure 12.

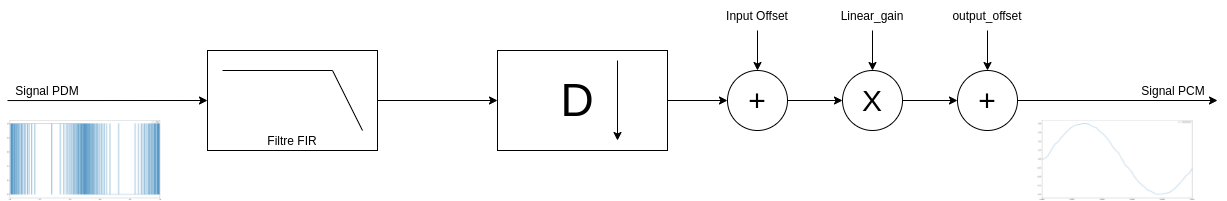


Figure 12 : Chaine de filtrage PDM → PCM

Pour mieux comprendre le fonctionnement du filtre, nous allons étudier un signal pendant son passage au travers de cette chaine de filtrage. Nous utiliserons un signal en dent de scie suivant l’expression suivante :

Le signal que nous utiliserons dans cet exemple a les paramètres suivants :

|  |  |
| --- | --- |
| Paramètre | Valeur |
| Fréquence PDM |  |
| Fréquence d’échantillonnage PCM |  |
| Fréquence fondamentale |  |
| Ordre de la série de Fourier |  |
| Durée du signal |  |

Ce qui donne le signal suivant (centré une période) :

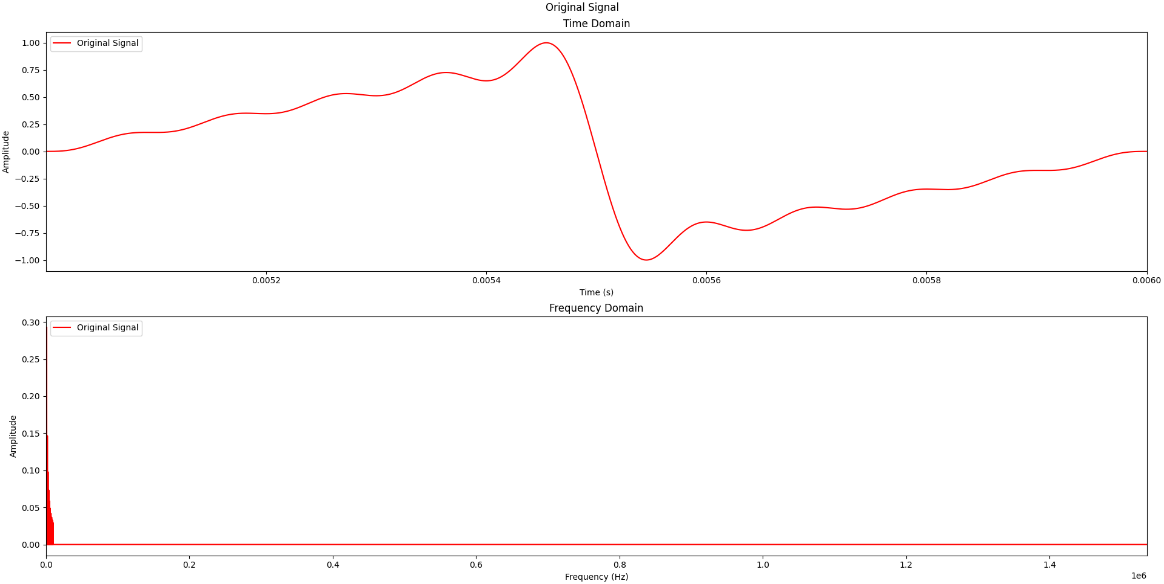


Figure 13 : Signal Original

Avant d’envoyer le signal dans la chaine de filtrage, on le converti en PDM ce qui nous donne le signal suivant :

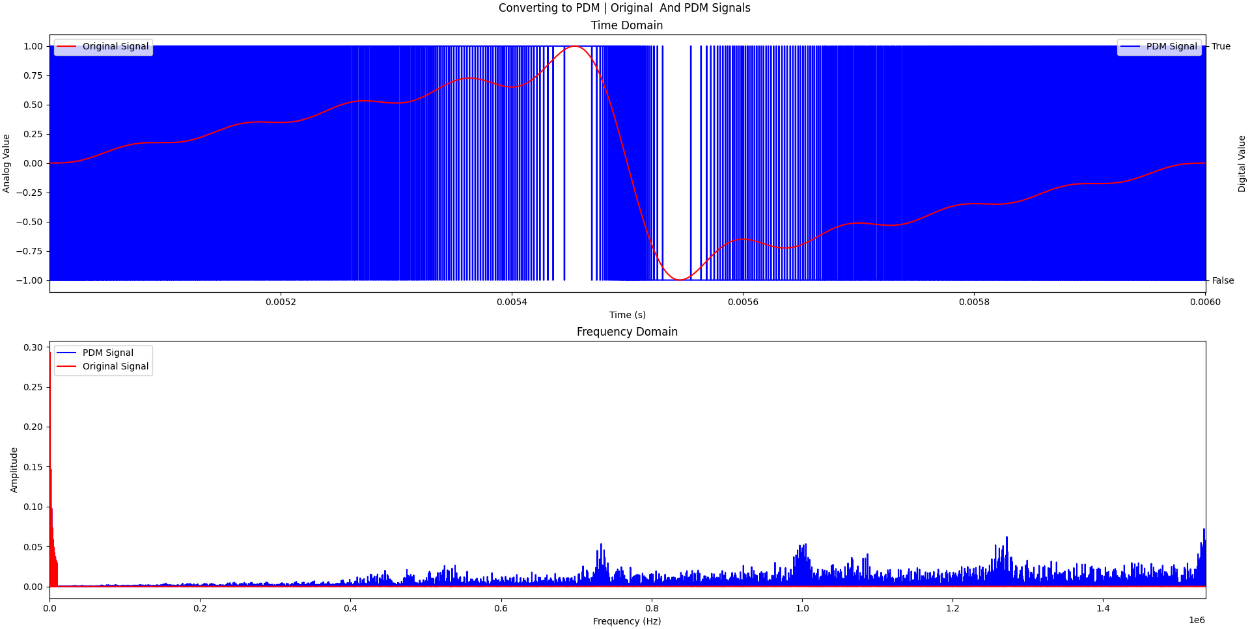


Figure 14 : Conversion PCM (rouge) → PDM (bleu)

La première étape de la chaine est d’appliquer un filtre passe-bas au signal, dans notre cas, c’est un filtre numérique FIR (Finite Impulse Response) d’ordre qui nous permet d’éliminer les hautes fréquences caractéristiques d’un signale PDM pour ne laisser que les fréquences utiles. Ici l’ordre du filtre FIR est . À la suite du filtrage on obtient le signal suivant:

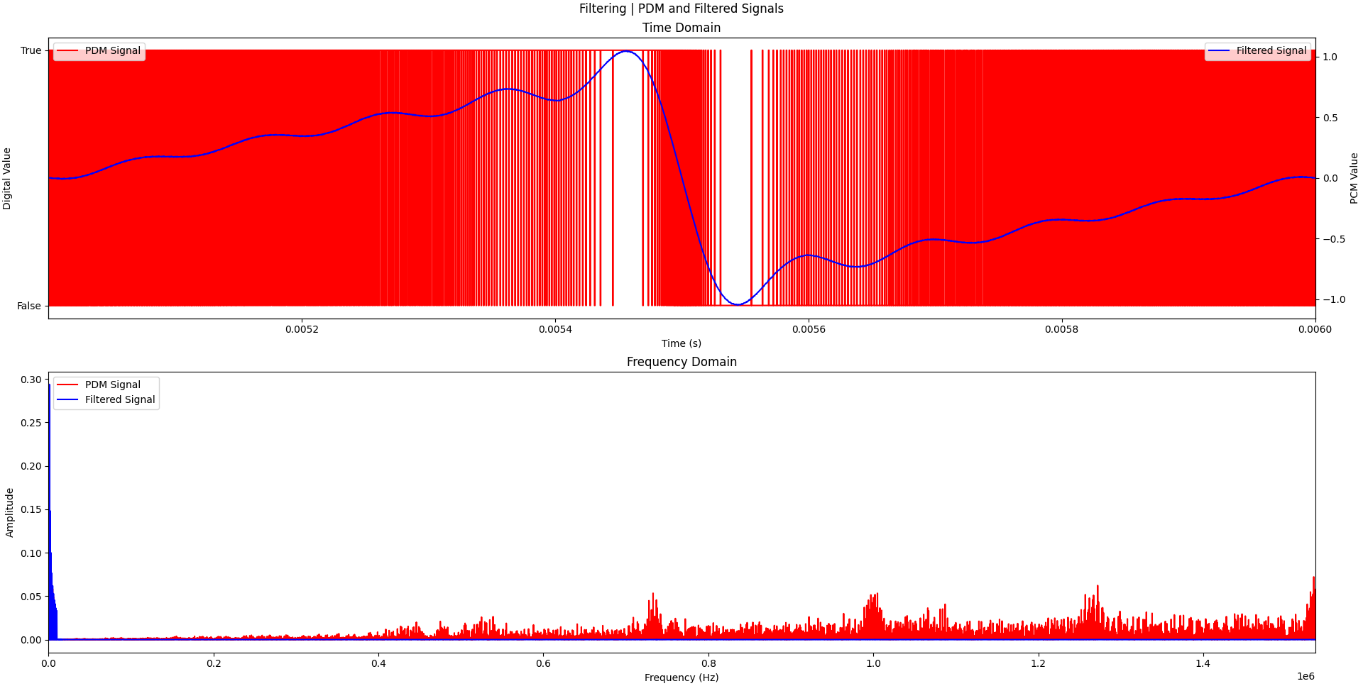


Figure  : Conversion PDM (Rouge) → PCM (Bleu)

Une fois le signal filtré, il a la même fréquence d’échantillonnage que le signal PDM. Or, le signal PDM est échantillonné à haute fréquence pour éloigner le plus possible les bruits générés par sa nature de la bande utile, maintenant que le signal a été filtré la bande nécessaire est très inférieur à celle du signal PDM. On va donc sous-échantillonner le signal filtré à la fréquence recherchée pour le signal PCM. Pour se faire un va sous-échantillonner le signal filtré par un facteur D (on sélectionne un échantillon sur D) :

Sous-échantillonnage qui nous donne le signal suivant :

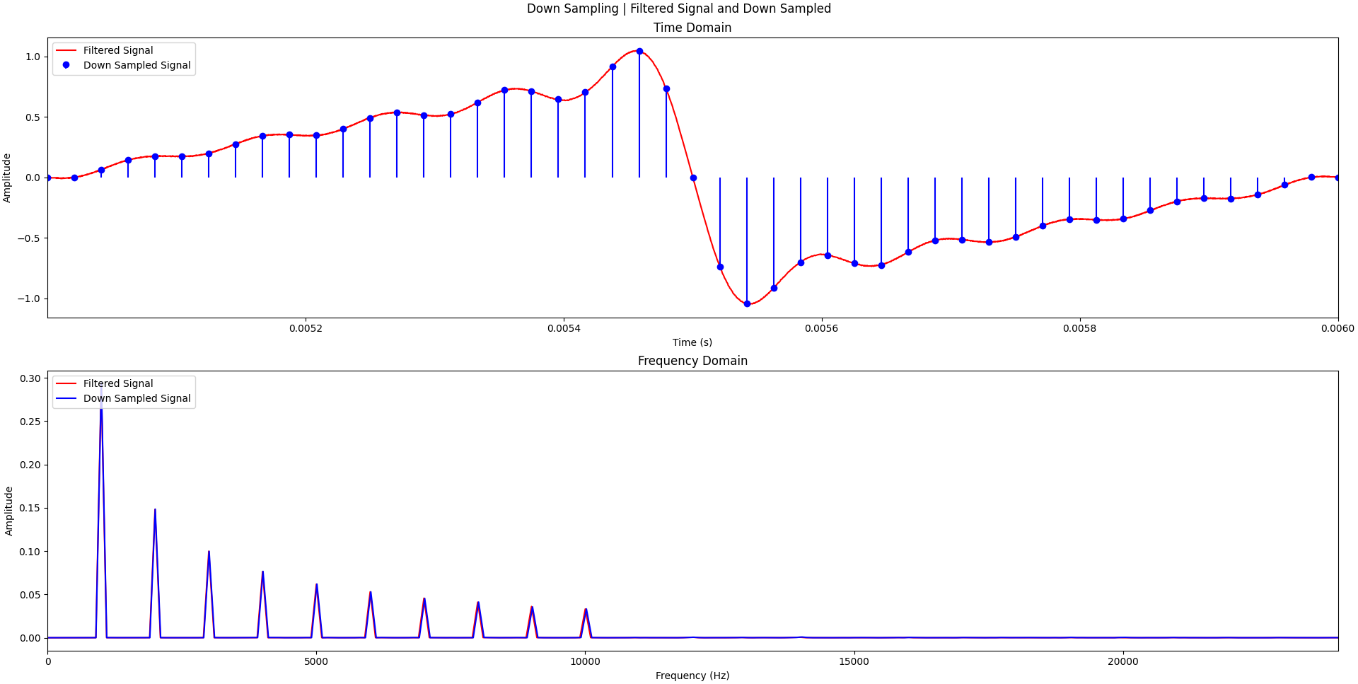


Figure  : Sous-échantillonnage du signal filtré

Une fois le signal sous-échantillonné on applique un gain et un offset sur le signal pour l’adapter à la sortie PCM Nous obtenons alors le signal suivant[[1]](#footnote-1) en sorti de chaine :

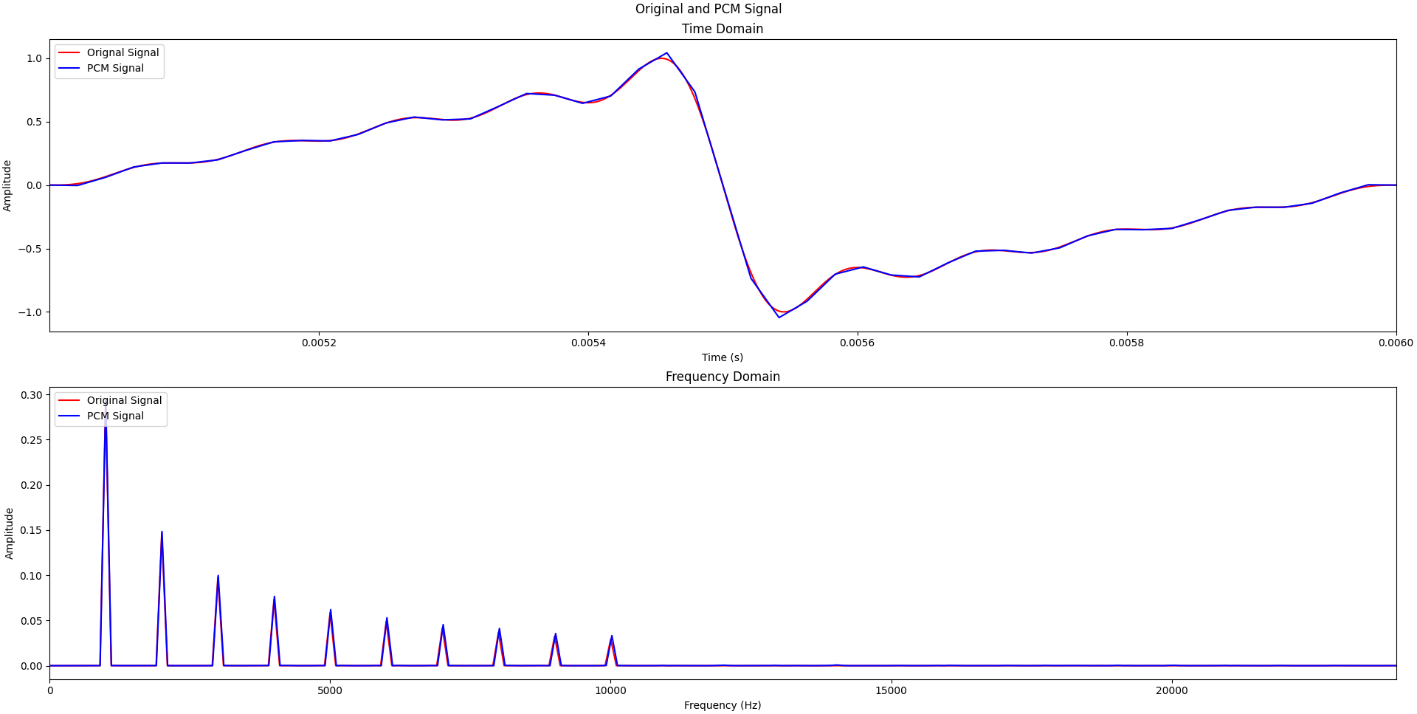


Figure  : Signal original (Bleu) et signal en sortie de chaine de filtrage (Rouge)

## Intégration au microcontrôleur

### Fonctionnement

Les échantillons PDM sont chargés en RAM par le DMA dès qu’un bloc de 16 échantillons PDM (16 bits) est mis à disposition par le périphérique SAI. Afin de notifier le programme de sa position dans le buffer, le DMA lève 2 interruptions, une première quand il passe la moitié du buffer la seconde à la fin de celui-ci. Ces interruptions sont traitées par le programme dès le lever via l’appel de fonction de "Callbacks". Ces fonctions agissent alors sur deux variables, une première qui notifie le programme principal de la présence de nouvelles données dans le buffer et une seconde qui indique dans quelle moitié du buffer lire les données. Une fois ces données récupérées depuis le buffer du DMA on les fait passer dans la chaine de filtrage présenté précédemment. Ce qui nous donne le processus suivant que l’on intègre au programme principal :

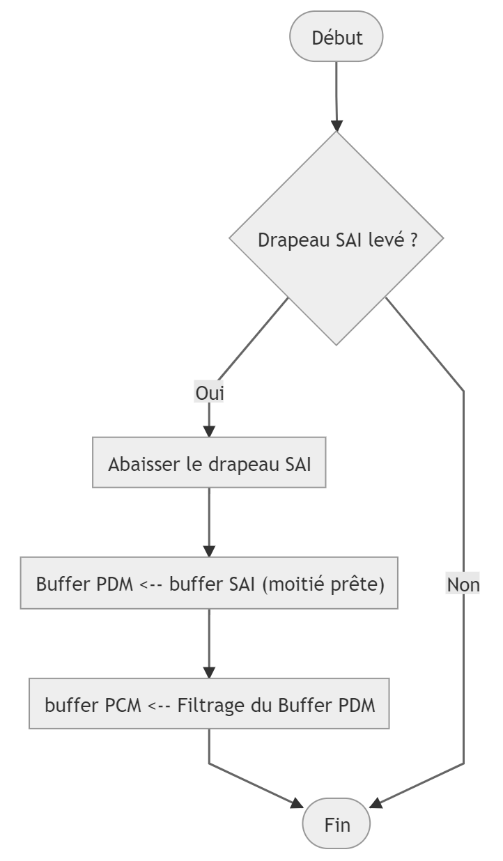


Figure 18 : Logigramme, Gestion des interruptions SAI

Dans ce processus, on vérifie si le drapeau SAI est levé. Dans le cas où le drapeau est levé on charge dans le buffer PDM la moitié dont le DMA vient de terminer l’écriture, puis on filtre le buffer PDM et on stocke le résultat dans le buffer PCM à la disposition du reste du programme.

Le bloc de filtrage utilise une bibliothèque C que j’ai retravaillée pour l’adapter à nos besoins. La bibliothèque utilise la fenêtre FIR d’ordre fournie par la bibliothèque Python "Scipy". En effet un script Python compagnon de la bibliothèque permet de générer une LUT correspondant à la fenêtre FIR sous forme du tableau constant C à deux dimension  ; . L’axe est deux fois plus grand que l’ordre du filtre car la bibliothèque applique le filtre sur les groupes d’échantillons PDM de 16 bits en deux fois 8 bits. On retrouvera donc sur l’axe le positionnement de l’échantillon de 8 bits dans le buffer, et sur l’axe le résultat de la convolution de l’échantillon avec la fenêtre. Ainsi sur la cible il suffit de sommer le résultat pour chaque échantillon du buffer de filtrage. L’amplitude de la LUT est amplifiée de façon occuper au maximum le conteneur de sortie dont on configure la taille à la génération de la LUT.

Pour gagner en performance et en simplicité, plutôt que travailler sur des échantillons PDM distinct, la librairie utilise des blocs de 16 échantillons PDM qui correspondent au format des données transmises par le périphérique SAI. Cette utilisation de blocs de 16 bits applique donc un premier sous-échantillonnage au signal PDM d’un facteur 16.

Pour filtrer notre signal PDM, cette librairie nous fournit deux fonctions :

* "pdm\_fir\_flt\_put" :

Cette fonction ajoute un mot PDM au buffer du filtre sur lequel on applique la fenêtre FIR. La taille de ce buffer est fixée par l’ordre du filtre FIR. Il a un fonctionnement circulaire qui permet de reproduire l’aspect glissant de la convolution d’un signal et d’une fenêtre.

* "pdm\_fir\_ftl\_get" :

Cette fonction calcule la convolution entre le buffer du filtre est la fenêtre FIR, et donc qui produit un échantillon PCM en appliquant l’algorithme suivant :

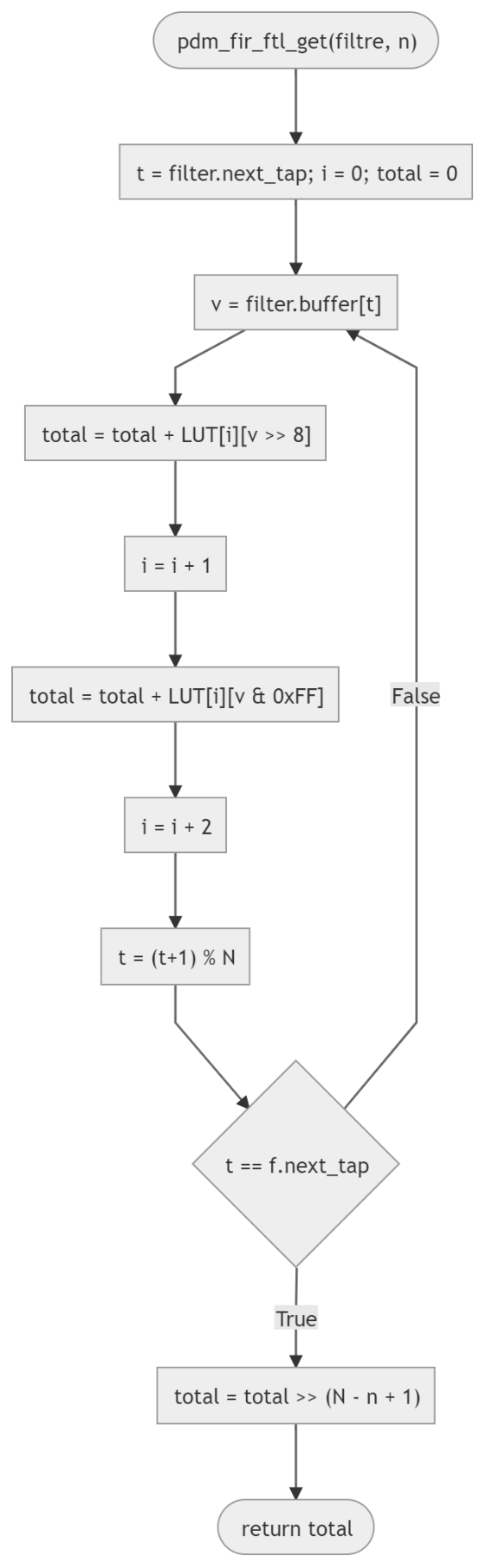


Figure : Logigramme de la fonction "pdm\_fir\_ftl\_get"

La bibliothèque nous permet donc d’implémenter le bloc de filtrage et une partie du bloc de sous-échantillonnage de la chaine de filtrage PDM (cf. Figure 12). Pour compléter la chaine j’ai donc ajouté à la bibliothèque une structure de donnée contenant la configuration de la chaine de filtrage (facteur de sous-échantillonnage, facteur d’amplification, offset…) ainsi que la fonction :

* "pdm\_fir\_flt\_chunk" :

Cette fonction produit le signal PCM correspondant au buffer PDM qu’on lui fournit en lui appliquant les différentes étapes de la chaine de filtrage. C’est cette fonction que l’on appelle en Figure 18 dans le bloc "buffer PCM <-- Filtrage du Buffer PDM" pour filtrer le signal PDM.

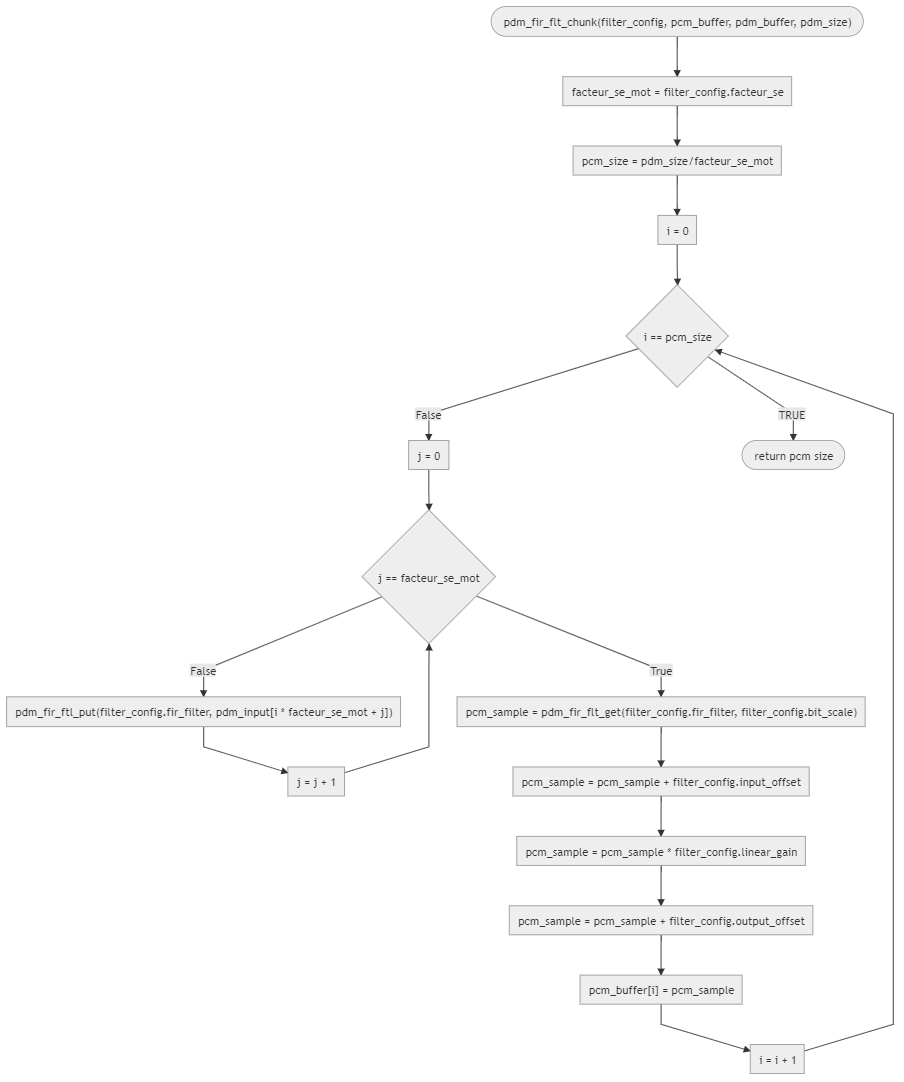


Figure : Logigramme de la fonction "pdm\_fir\_ftl\_chunck"

En conclusion les données du microphone PDM sont recueillies par le périphérique SAI qui est coupler au DMA du microcontrôleur pour charger les échantillons PDM en RAM. Une fois ces échantillons charger le programme principal les fais passer dans la chaine de filtrage pour produire les échantillons PCM correspondant. Cette chaine de filtrage provient d’une bibliothèque C qui nous fournis le filtre FIR, le reste de la chaine étant fournit par des sources que j’ai développé pour compléter la bibliothèque

### Performances de la chaine de filtrage

Nous avons une bibliothèque C qui gère le filtrage du signal PDM mais avant de pourvoir développer les démonstrateurs il faut s’assurer que les contrainte temps réel liée a l’acquisition du signal sont respecter par celle-ci. Pour ce faire nous utiliserons le démonstrateurs "Direct Output", qui rejoue en temps réel les sons captés. Le démonstrateur a été instrumenté de façon à fournir les signaux suivant sur le GPIO :

* Un signal à 3V entre la levée du drapeau SAI et le début du filtrage et a 0V le reste du temps
* Un signal à 3V pendant le filtrage et a 0V le reste du temps

Ces deux signaux nous donnant les chronogrammes suivants :

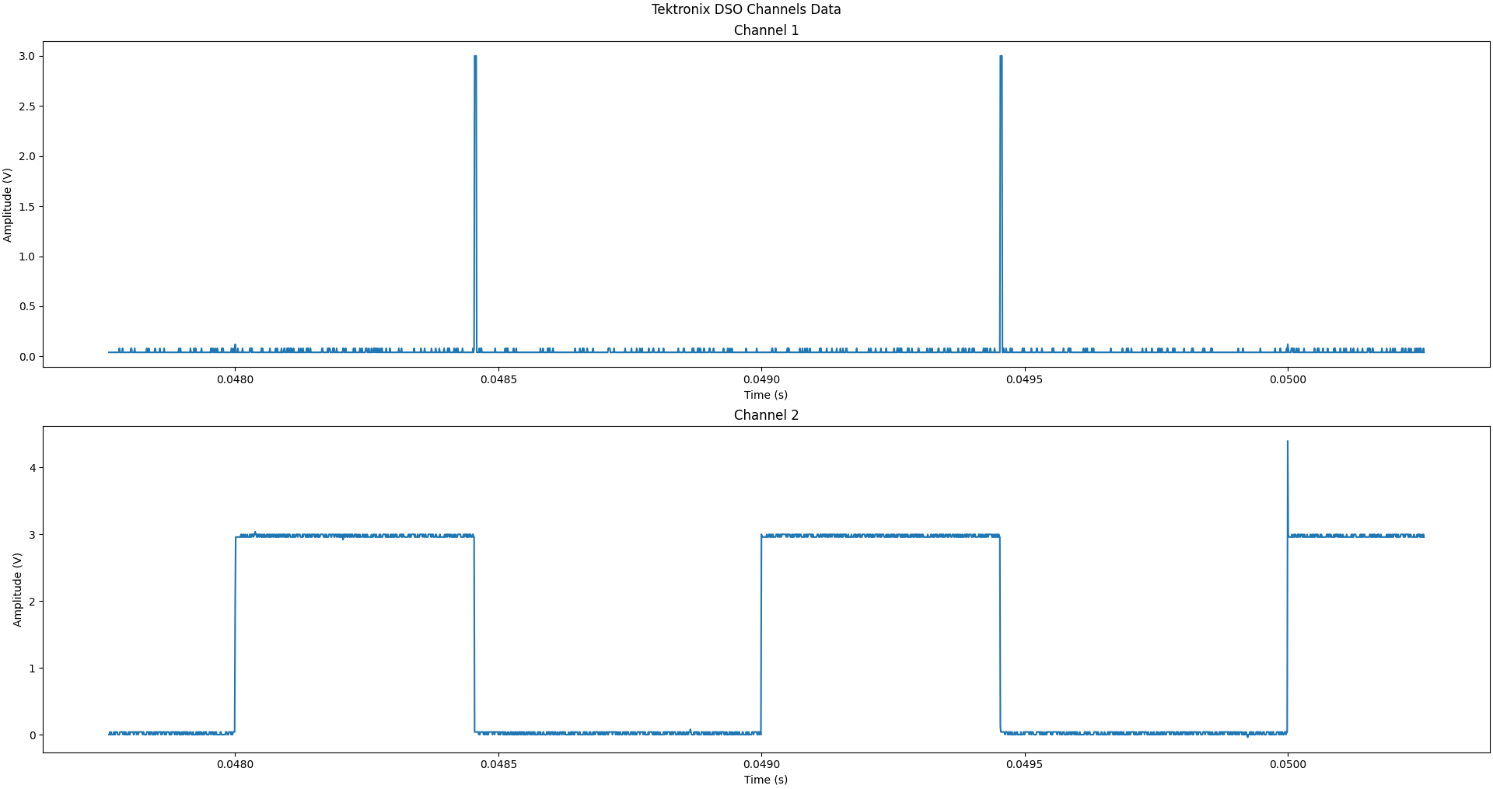


Figure 21 : Signaux « Drapeau SAI » (Channel 1) et « filtrage » (Channel2) pour le démonstrateur "Direct Output"

Dans ces mesures nous nous intéresserons plus particulièrement au temps de filtrage par rapport au temps entre deux drapeau SAI. En effet quand l’on enregistre les sons ambiant l’ensemble des traitement effectué (filtrage, copie sur le DAC, enregistrement des échantillons…) sur le signal doit être compris dans le cycle entre drapeau SAI pour maintenir l’intégrité du signal capté. Notre objectif pour ces mesures est de déterminer le profil de l’occupation du cycle SAI par le filtrage en fonction des fréquences d’échantillonnage et de l’horloge système.

Dans le démonstrateur "Direct Output" qui nous sert de base, la chaine de filtrage a été généré avec les paramètres suivants :

|  |  |
| --- | --- |
| **Paramètre** | **Valeur** |
| Fréquence du signal PDM |  |
| Ordre de la fenêtre FIR |  |
| Fréquence de coupure |  |
| Facteur de sous échantillonnage |  |
| Offset avant amplification |  |
| Facteur d’amplification linéaire |  |
| Offset après amplification |  |
| Taille des échantillons PCM (bits) |  |

Nous ferons donc trois séries de mesure à trois fréquences de l’horloge système différente. Pour chacune de ses séries on a fixé le buffer de filtrage à 1ms. C’est-à-dire à dire que le cycle SAI dure 1ms.

Nous ferons donc varier la fréquence d’échantillonnage PCM jusqu’à atteindre le point où temps nécessaire pour filtrer les données est supérieure à 1ms et donc que le filtrage du signal (seul) ne nous permet plus de respecter les contraintes temps-réels fixés par le cycle SAI.

L’abaque et la figure suivante présentes le résultat des mesures réalisées à trois fréquences d’horloge système (72, 108 et 144 MHz) pour les fréquences d’échantillonnage PCM entre 8 et 192 kHz avec un pas de 8 kHz :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |
|  |  | Occupation |  | Occupation |  | Occupation |
| 8 kHz | 80 µs | 8 % | 52 µs | 5,20 % | 44 µs | 4,4 % |
| 16 kHz | 150 µs | 15 % | 108 µs | 10,8 % | 88 µs | 8,8 % |
| 24 kHz | 230 µs | 23 % | 164 µs | 16,4 % | 128 µs | 12,8 % |
| 32 kHz | 310 µs | 32 % | 212 µs | 21,2 % | 166 µs | 16,6 % |
| 40 kHz | 390 µs | 39 % | 264 µs | 26,4 % | 204 µs | 20,4 % |
| 48 kHz | 450 µs | 45 % | 316 µs | 31,6 % | 248 µs | 24,8 % |
| 56 kHz | 530 µs | 53 % | 372 µs | 37,2 % | 284 µs | 28,4 % |
| 64 kHz | 600 µs | 60 % | 420 µs | 42,0 % | 328 µs | 32,8 % |
| 72 kHz | 670 µs | 67 % | 476 µs | 47,6 % | 368 µs | 36,8 % |
| 80 kHz | 750 µs | 75% | 524 µs | 52,4 % | 408 µs | 40,8 % |
| 88 kHz | 820 µs | 82% | 576 µs | 57,6 % | 448 µs | 44,8 % |
| 96 kHz | 900 µs | 90 % | 628 µs | 62,8 % | 488 µs | 48,8 % |
| 104 kHz | 980 µs | 98 % | 680 µs | 68,0 % | 532 µs | 53,2 % |
| 112 kHz | 1060 µs | 106 % | 732 µs | 73,2 % | 572 µs | 57,2 % |
| 120 kHz | Non mesuré | Non mesuré | 784 µs | 78,4 % | 612 µs | 61,2 % |
| 128 kHz | Non mesuré | Non mesuré | 840 µs | 84,0 % | 652 µs | 65,2 % |
| 136 kHz | Non mesuré | Non mesuré | 892 µs | 89,2 % | 696 µs | 69,6 % |
| 144 kHz | Non mesuré | Non mesuré | 940 µs | 94,0 % | 736 µs | 73,6 % |
| 152 kHz | Non mesuré | Non mesuré | 995 µs | 99,5 % | 776 µs | 77,6 % |
| 160 kHz | Non mesuré | Non mesuré | 1060 µs | 106,0 % | 816 µs | 81,6 % |
| 168 kHz | Non mesuré | Non mesuré | Non mesuré | Non mesuré | 860 µs | 86,0 % |
| 176 kHz | Non mesuré | Non mesuré | Non mesuré | Non mesuré | 896 µs | 89,6 % |
| 184 kHz | Non mesuré | Non mesuré | Non mesuré | Non mesuré | 940 µs | 94,0 % |
| 192 kHz | Non mesuré | Non mesuré | Non mesuré | Non mesuré | 985 µs | 98,5 % |

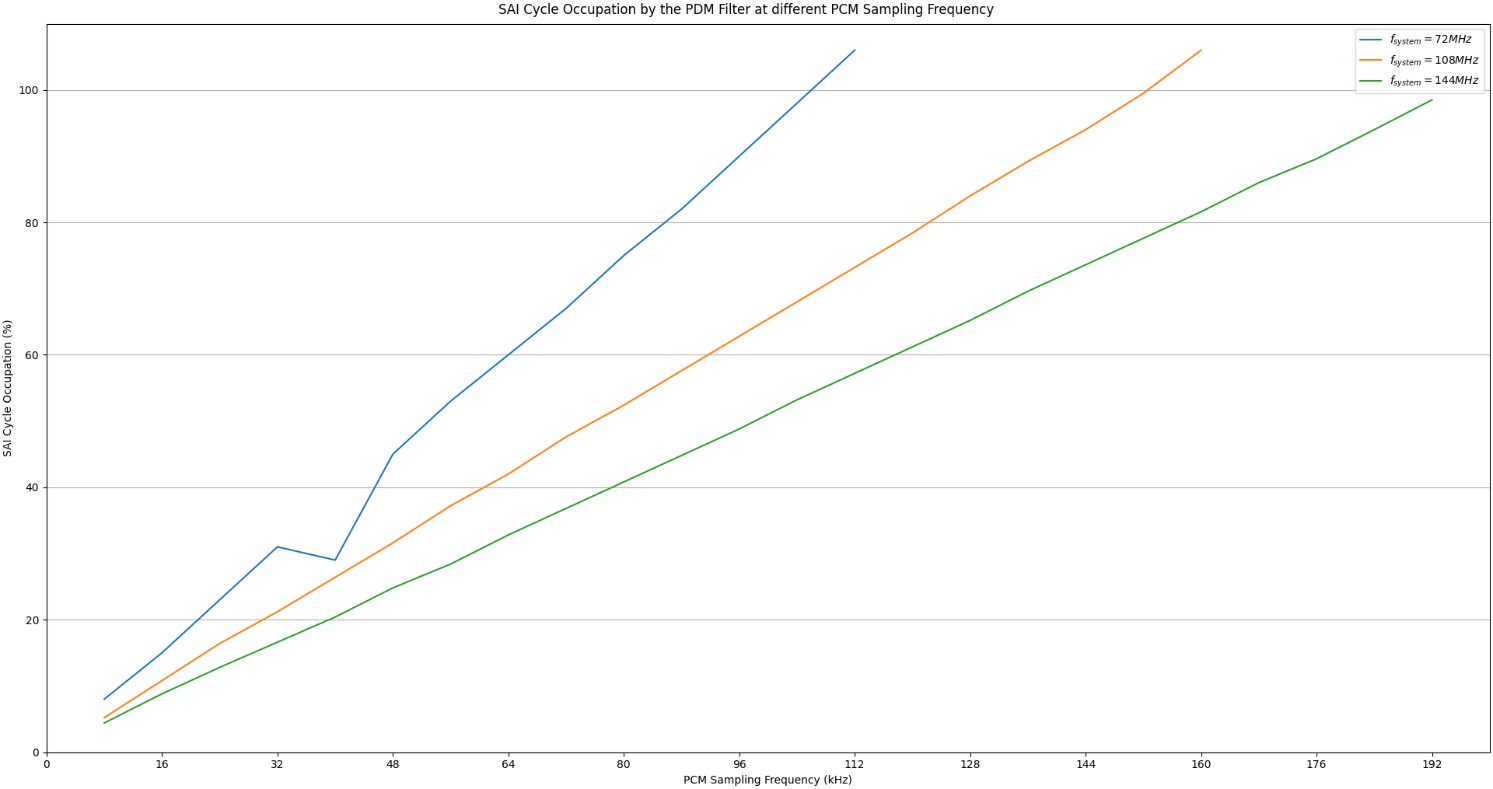


Figure : Occupation du cycle SAI par le filtre PDM selon la fréquence d'échantillonnage PCM

On remarque sur les deux courbes que l’occupation du cycle SAI par le filtre est une fonction linaire de la fréquence d’échantillonnage PCM. On remarque aussi que le coefficient directeur de cette fonction semble être proportionnelle avec la fréquence de l’horloge système, ce qui semble cohérent avec le fait que la l’horloge système influx directement sur le nombre d’instruction par seconde qui sont exécutés par le microcontrôleur.

Dans le cas du démonstrateur "Direct Output" où l’on recherche une reproduction en temps réel des sons ambiant avec un bonne qualité audio, les paramètres :

Permettent un taux d’occupation du cycle SAI de 45% laissant ainsi une grande marge pour la reproduction du signal sur le DAC en offrant l’ensemble de spectre audible ().

## Présentation des démonstrateurs :

Afin de maitriser les différents aspects de la chaine de capture et de la carte cible, j’ai développé trois démonstrateurs, permettant de mettre en lumière différentes applications que l’on peut avoir pour ce type de microphones.

### « Parrot »

Le démonstrateur Parrot est un programme qui enregistre un son puis le rejoue le son sur un haut-parleur. Pour remplir ses fonctions, le programme s’appuie sur la machine de 4 états suivante :

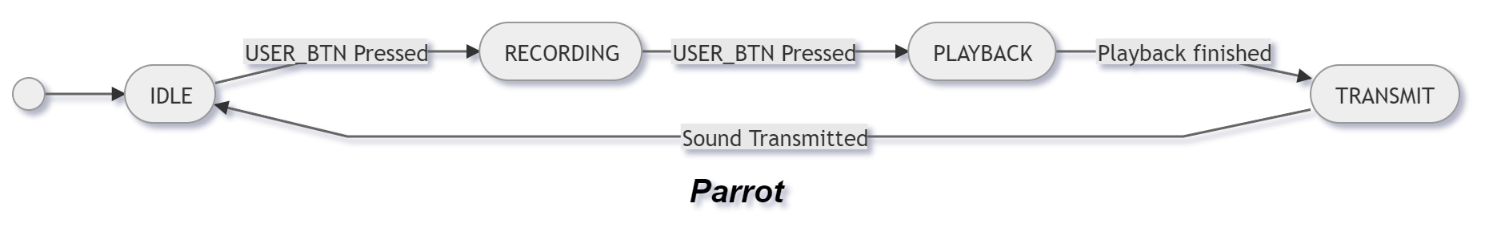


Figure : machine d’état du démonstrateur "Parrot"

Le programme passe donc dans les états suivants :

* "IDLE" :

Le programme est en attente d’un appui sur le bouton "USER" (bouton bleu) pour passer dans l’état "RECORDING"

* "RECORDING" :

Le programme enregistre les sons ambiants et conserve jusqu’à 3 secondes en mémoire RAM. Il passe dans l’état "PLAYBACK" lors d’un appui sur le bouton "USER"

* "PLAYBACK" :

Le programme rejoue le son enregistré en mémoire sur les 2 canaux du DAC. Il passe dans l’état "TRANSMIT" une fois que la totalité du son enregistré a été rejouée.

* "TRANSMIT" :

Le programme transmet le son enregistré sous forme de données WAV sur la ligne série du l’UART1, une fois l’ensemble des données transmises, le programme passe à l’état "IDLE"

Le logigramme de chacun des états et disponible en Annexe 1 : Logigrammes des États démonstrateur "Parrot"

L’état "TRANSMIT" peut être désactivé en commentant la définition de la définition C "#define TRANSMIT" (Core/Src/main.c l.80). Quand cet état est désactivé, le programme passe directement de l’état "PLAYBACK" à l’état "IDLE" sans transmettre les données sur la ligne série.

Pour ce démonstrateur j’utilise le schéma bloc des périphériques suivants :

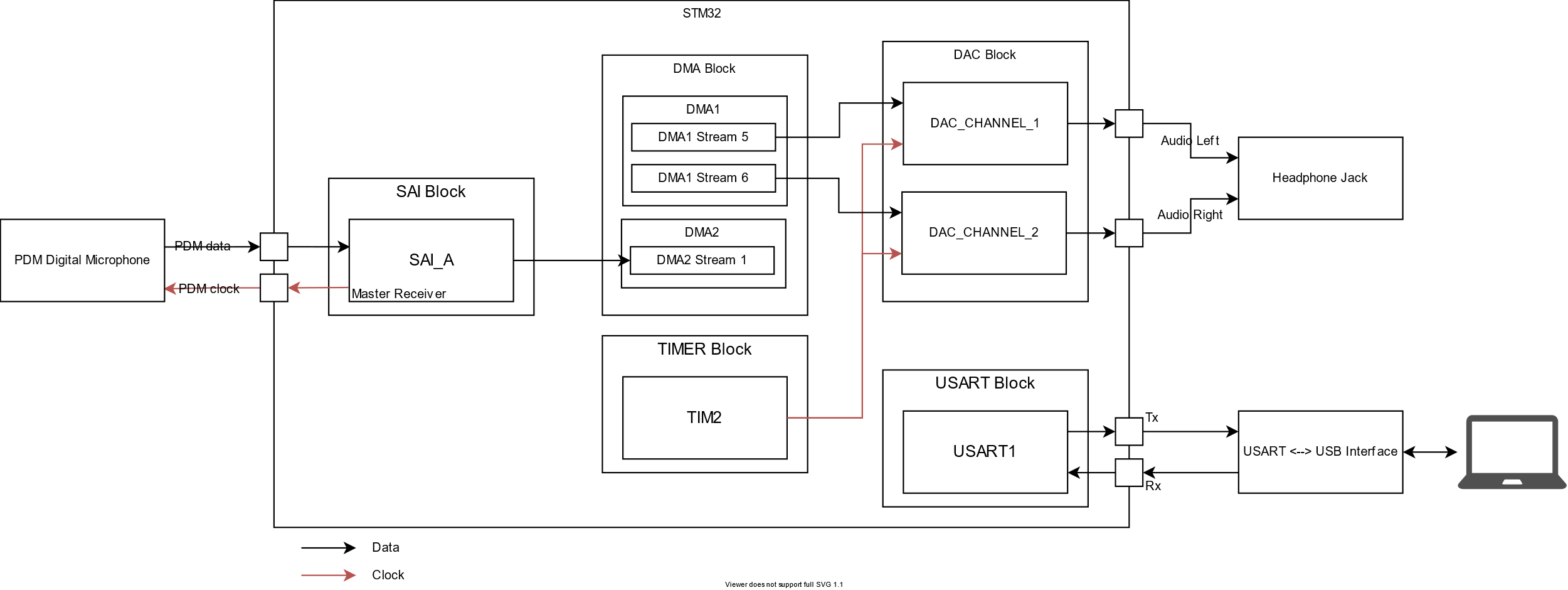


Figure : Schéma Bloc des périphériques utilisés par le démonstrateur "Parrot"

Ce démonstrateur produit un signal PCM échantillonné sur 12 bits à 32 kHz. Pour ce démonstrateur, j’ai choisi un facteur de sous-échantillonnage de 64 donc . Pour le filtre, j’ai généré la LUT pour une fenêtre FIR d’ordre avec une fréquence de coupure

Pour ce démonstrateur, j’ai fait le choix d’échantillonner le signal PCM à 32 kHz afin de pouvoir stocker 3 secondes de son () en saturant au maximum la mémoire RAM () du microcontrôleur, si l’on utilise une fréquence d’échantillonnage PCM de 48 kHz le temps maximum d’enregistrement descend à 2 secondes () ce qui peut paraitre court quand on utilise le démonstrateur.

Lors de la transmission des échantillons PCM, le programme amplifie le signal pour utiliser l’ensemble des 16 bits par échantillons mis à disposition par le fichier WAV

### « Digital recorder »

Le démonstrateur "Digital recorder" est un programme qui reproduit les fonctions d’un enregistreur audio, en sauvegardant le son capté par le microphone sous forme de fichiers WAV sur la clé USB connectée à l’interface USB\_HS. Pour remplir ses fonctions, le programme s’appuie sur la machine de 3 états suivante :

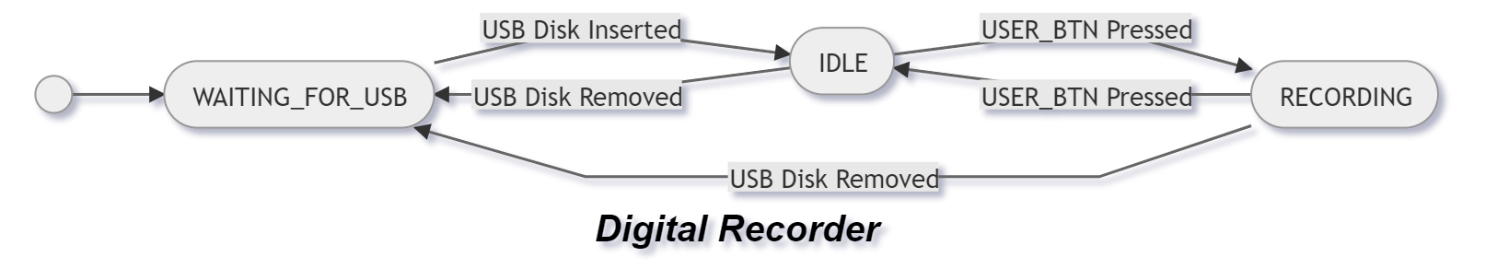


Figure 25: Machine d’état du démonstrateur "Digital Recorder"

Le programme passe donc dans les états suivants :

* "WAITING\_FOR\_USB" :

Le programme est en attente de l’insertion d’une clé USB qui le fera passer dans l’état "IDLE". Si la clé USB est retirée, le programme reviendra directement dans cet état, peu importe son état courant.

* "IDLE" :

Le programme est en attente d’un appui sur le bouton "USER" (bouton bleu) pour passer dans l’état "RECORDING"

* "RECORDING" :

Le programme enregistre les sons ambiants dans un fichier WAV sur la clé USB, lors de l’appui sur le bouton "USER" le programme finalise l’enregistrement et passe dans l’état "IDLE".

Le logigramme de chacun des états et disponible en Annexe 2 : Logigrammes des États démonstrateur "Digital Recorder"

Pour ce démonstrateur j’utilise le schéma bloc des périphériques suivants :

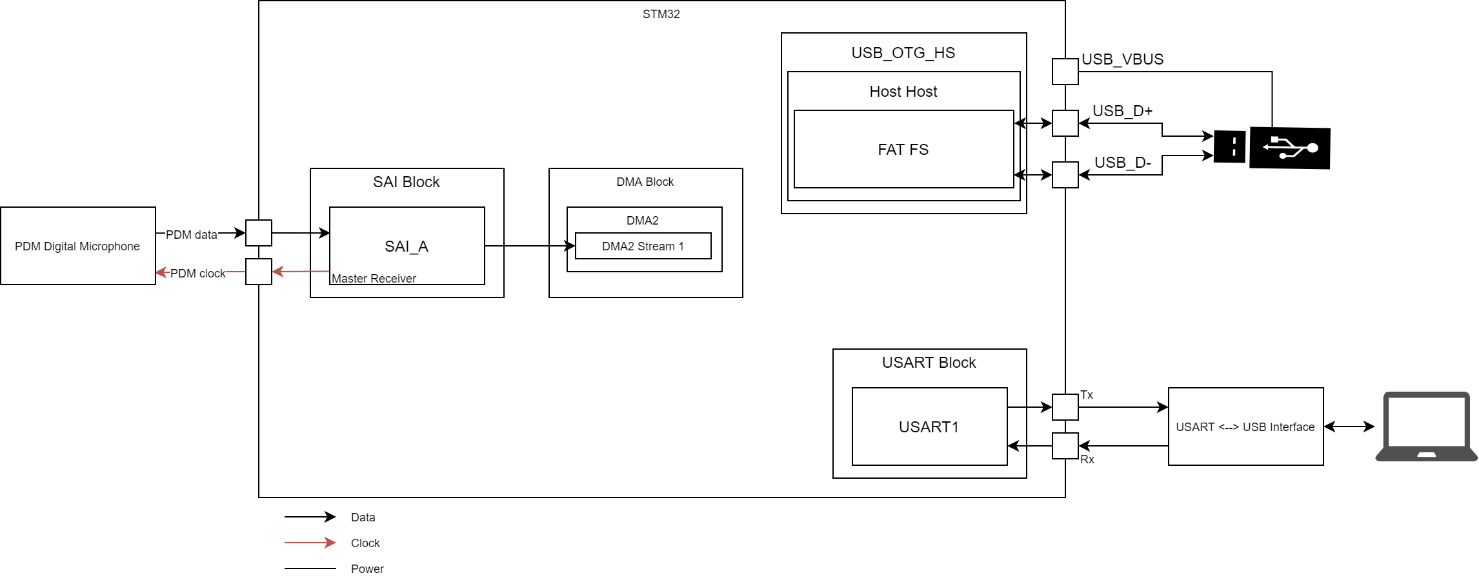


Figure : Schéma Bloc des périphériques utilisés par le démonstrateur "Digital Recorder"

Ce démonstrateur produit un signal PCM échantillonné sur 16 bits à 48 kHz. Pour ce démonstrateur, j’ai choisi un facteur de sous-échantillonnage de 64 donc . Pour le filtre, j’ai généré la LUT pour une fenêtre FIR d’ordre 16 avec une fréquence de coupure

### « Direct output »

Le démonstrateur "Direct Output" est un programme qui enregistre un son puis le rejoue le son sur un haut-parleur. Pour remplir ses fonctions, le programme s’appuie sur la machine de 2 états suivante :

Une image contenant texte, périphérique, sombre, mètre

Description générée automatiquement

* "IDLE" :

Le programme est en attente d’un appui sur le bouton "USER" (bouton bleu) pour passer dans l’état "RECORDING"

* "RECORDING" :

Le programme enregistre les sons ambiants et les rejoue en temps réel sur le DAC. Il retourne dans l’état "IDLE" lors d’un appui sur le bouton "USER".

Le logigramme de chacun des états et disponible en Annexe 3 : Logigrammes des États démonstrateur "Direct Output"

Ce démonstrateur quand il est dans l’état "RECORDING" permet si l’on branche un casque relativement bien isolé des sons ambiants d’offrir une démonstration relativement impressionnante des capacités de notre chaine de capture.

Pour ce démonstrateur j’utilise le schéma bloc des périphériques suivants :

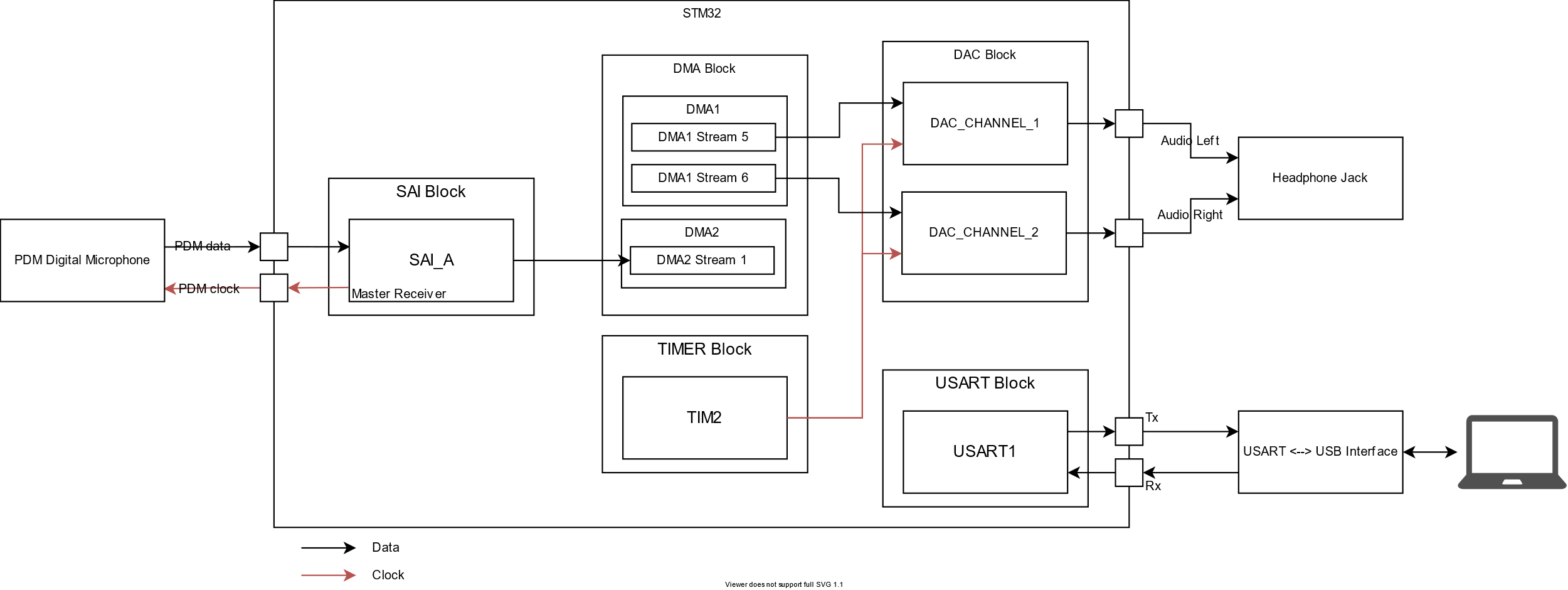


Figure 27 : Schéma Bloc des périphériques utilisés par le démonstrateur "Direct Output"

Ce démonstrateur produit un signal PCM échantillonné sur 12 bits à 48 kHz. Pour ce démonstrateur, j’ai choisi un facteur de sous-échantillonnage de 64 donc . Pour le filtre, j’ai généré la LUT pour une fenêtre FIR d’ordre 16 avec une fréquence de coupure

# Bibliographies

olegv142. (2017). *PDM bitstream FIR filter*. Retrieved from Github.com: https://github.com/olegv142/pdm\_fir

ST Microelectronics. (2018, January). STM32F427xx STM32F429xx. *Datasheet | STM32F427xx STM32F429xx*.

ST Microelectronics. (2019, July). Interfacing PDM digital microphones using STM32 MCUs and MPUs. *AN5027 | Interfacing PDM digital microphones using STM32 MCUs and MPUs*.

ST Microelectronics. (2020, August). Discovery kit with STM32F429ZI MCU. *UM1670 | Discovery kit with STM32F429ZI MCU*.

# Glossaires des termes techniques

|  |  |
| --- | --- |
| **Terme ou acronyme** | **Définition** |
| ADC | "Analog to Digital Converter". Convertisseur analogique vers Numérique. |
| Buffer | Espace mémoire tampon |
| Buffer Circulaire | Un buffer circulaire est un buffer de taille fixe auquel on a rejoint le début et la fin de façon qu’il puisse recevoir des valeurs de façon infinies, les nouvelles valeurs remplaçant les anciennes au fur et à mesure que l’on en ajoute |
| DAC | "Digital to Analog Converter". Convertisseur numérique vers Analogique. |
| DMA | "Direct Memory Access". Composant du microcontrôleur permettant l’échange de données entre la RAM et les périphériques sans impacter l’exécution du programme principal |
| GPIO | "General Purpose Input Output" Entrées/Sortie du microcontrôleur qui ne sont pas affectées a des périphérique spécifique |
| LookUp Table (LUT) | « Table de Correspondance » Structure de données qui contient des données précalculer pour afin de réduit le temps nécessaire au programme pour effectuer une opération complexe en la remplaçant par une consultation de la table. Cependant, une LUT peut prendre beaucoup d’espace en mémoire. |
| MEMS | "MicroElectroMechanical Systems", un MEMS est un système dont la taille est de l’ordre de quelques micromètres, cette taille très réduite permet d’associer les propriétés électriques d’un semi-conducteur à celle d’un capteur mécanique en ayant un encombrement minimum |
| PCM | "Pulse Code Modulation". Modulation d’un signal numérique où chaque échantillon stocke le niveau du signal à un instant T |
| PDM | "Pulse Density Modulation". Modulation d’un signal numérique où le niveau du signal et définie par la densité d’échantillons à "1" |
| Quantum | Le Quantum d’un ADC/DAC correspond à la plus petite variation de tension qu’il peut reproduire/distinguer, il se calcule grâce à la formule suivante : où correspond à la plage de tension de travail du convertisseur en Volt, et correspond au nombre de bits du convertisseur |
| SAI | "Serial Audio Interface". Interface numérique de transfert de signaux audio en série |
| USB OTG | "Universal Serial Bus" Extension du protocole USB qui permet l’échange de données entre deux périphériques USB sans avoir à passer par un ordinateur Hôte (ex. Téléphone → Clé USB) |
|  |  |
|  |  |

# Table des figures

[Figure 1 : Comportement du démonstrateur obligatoire 6](#_Toc88482057)

[Figure 2 : Carte Cible (STM32F429I-DISC1) 7](#_Toc88482058)

[Figure 3 : Schéma bloc typique d’un microphone PDM 7](#_Toc88482059)

[Figure 4 : Connexion du Microphone PDM en configuration mono 8](#_Toc88482060)

[Figure 5 : Chaine de capture 8](#_Toc88482061)

[Figure 6 : Montage direct d’un haut-parleur sur le DAC 9](#_Toc88482062)

[Figure 7 : Branchement d’un haut-parleur en série avec une résistance de 1 kΩ sur le DAC 10](#_Toc88482063)

[Figure 8 : Schéma Électrique du câble DAC → Jack 3,5 mm 10](#_Toc88482064)

[Figure 9 : Câble DAC → Jack 3.5mm 10](#_Toc88482065)

[Figure 10 : Signal Analogique et sa conversion numérique en PCM (, ) 12](#_Toc88482066)

[Figure 11 : Signal Analogique et sa conversion numérique en PDM () 13](#_Toc88482067)

[Figure 12 : Chaine de filtrage PDM → PCM 14](#_Toc88482068)

[Figure 13 : Signal Original 14](#_Toc88482069)

[Figure 14 : Conversion PCM (rouge) → PDM (bleu) 15](#_Toc88482070)

[Figure 15 : Conversion PDM (Rouge) → PCM (Bleu) 15](#_Toc88482071)

[Figure 16 : Sous-échantillonnage du signal filtré 16](#_Toc88482072)

[Figure 17 : Signal original (Bleu) et signal en sortie de chaine de filtrage (Rouge) 16](#_Toc88482073)

[Figure 18 : Logigramme, Gestion des interruptions SAI 17](#_Toc88482074)

[Figure 19 : Signaux de copie sur le DAC et de filtrage pour le démonstrateur "Direct Output" 18](#_Toc88482075)

[Figure 20 : machine d’état du démonstrateur "Parrot" 19](#_Toc88482076)

[Figure 21 : Schéma Bloc des périphériques utilisés par le démonstrateur "Parrot" 20](#_Toc88482077)

[Figure 22: Machine d’état du démonstrateur "Digital Recorder" 20](#_Toc88482078)

[Figure 23 : Schéma Bloc des périphériques utilisés par le démonstrateur "Digital Recorder" 21](#_Toc88482079)

[Figure 24 : Schéma Bloc des périphériques utilisés par le démonstrateur "Direct Output" 22](#_Toc88482080)

# Annexes

## Annexe 1 : Logigrammes des États démonstrateur "Parrot"

Annexe disponible dans le fichier « [Annexe\_1\_logigrammes\_parrot.html](https://piwithy.fr/files/Public/ensta/pfe_stm32/report/Annexe_1_logigrammes_parrot.html)»

## Annexe 2 : Logigrammes des États démonstrateur "Digital Recorder"

Annexe disponible dans le fichier « [Annexe\_2\_logigrammes\_digital\_recorder.html](https://piwithy.fr/files/Public/ensta/pfe_stm32/report/Annexe_2_logigramme_digital_recorder.html)»

## Annexe 3 : Logigrammes des États démonstrateur "Direct Output"

Annexe disponible dans le fichier « [Annexe\_3\_logigrammes\_direct\_output.html](https://piwithy.fr/files/Public/ensta/pfe_stm32/report/Annexe_3_logigramme_direct_output.html)»

1. Il a été appliqué sur ce signal un gain linéaire de 1 et un offset de 0 (signal identique au signal sous-échantillonné) [↑](#footnote-ref-1)