MICROCONTRÔLEUR 2 :

« MEVENGUE ENGONGOMO FRANCK ANDY »

Élève ingénieur de 2ème année

Groupe 5

Promotion 2024

« TP2 Générateur de signaux »

Année 2022/2023

I- Partie Théorique :

1 – Architecture d'un DSP :

Indiquer la différence entre un DSP à virgule fixe et un DSP à virgule flottante:

Les DSP à virgule fixe : les données sont représentées comme étant des nombres fractionnaires à virgule fixe, (exemple -1.0 à +1.0), ou comme des entiers classiques. La représentation de ces nombres fractionnaires s'appuie la méthode du « complément à deux ». L'avantage de cette représentation (qui n'est qu'une convention des informaticiens) est de permettre facilement l'addition binaire de nombres aussi bien positifs que négatifs.

Les DSP à virgule flottante : les données sont représentées en utilisant une mantisse et un exposant. La représentation de ces nombres s'effectue selon la formule suivante : n = mantisse x 2 exposant. Généralement, la mantisse est un nombre fractionnaire (-1.0 à +1.0), et l'exposant est un entier indiquant la place de la virgule en base 2 (c'est le même mécanise qu'en base 10).

Que signifie CPU?

CPU est un acronyme anglais pour Central Processing Unit.
Le CPU désigne un processeur ou microprocesseur principal d'un ordinateur, on parle aussi d'unité de traitement. Le CPU est l'élément central d'une configuration informatique dont la puissance et la vitesse de traitement influencent directement les performances.

Quelle est la plus grande fréquence de l'horloge du CPU (Clk in), page 13?

La fréquence maximale est de 100 MHz

Quel est le rôle de l'ALU, page 9?

C'est lui qui effectue les opérations arithmétiques (addition, soustraction, multiplication, division), logiques (ET, OU, …) de comparaisons (=,>,<,...) et de décalage sur les données binaires que traite le processeur.

Indiquer les différences entre les mémoires Flash, Ram et ROM, page 8 :

La mémoire FLASH est une mémoire non volatile, elle ne perd pas ses données après avoir coupé l'alimentation. Le principe de la mémoire flash repose sur les portes logiques. C'est en quelque sorte la mémoire qui remplace les HDD et tous les autres supports de stockages de type non volatile. On gagne donc en compacité, en vitesse de lecture / transfert, en consommation, en coût de production, en durée de vie et en robustesse.

Mémoire ROM

La mémoire volatile conserve les données en absence d'alimentation. C'est un type de stockage très utile pour l'utilisateur.

ROM (Read Only Memory)

Mémoire à lecture seule, lors de la fabrication de cette mémoire on lui incorpore le programme. Très souvent utilisée pour stocker le programme d'amorçage du système, on peut la retrouver sur les cartes mères pour stocker le BIOS.

Mémoire RAM

Définition mémoire volatile et RAM

Une mémoire volatile perd toute ses données lorsque l'on coupe l'alimentation, elle ne sert donc pas à stocker nos fichiers vacances mais des données qui sont en cours de traitement comme par exemple votre navigateur internet, votre jeu en cours etc…

Par opposition à la mémoire morte on appelle ce genre de mémoire, la mémoire vive ou Random Access Memory (RAM). On peut lire, écrire et effacer les données, son avantage comparé à la ROM c'est sa rapidité d'accès que ce soit sur la latence ou le débit.

Expliquer le Principe de l'architecture de type Von Neumann et celle de Harvard, page 7.

Architecture de Harvard : les instructions et les données se trouvent dans deux mémoires séparées.
Architecture de Von Neumann : Les instructions et les données se trouvent dans la même mémoire.
Avantages et inconvénients : Accès indépendant aux instructions ou aux données.

Le DSP dispose d'une centaine d'interruptions, pages 11 et 12. Quel est l'intérêt d'utiliser les interruptions ?

On utilise les interruptions, principalement dans deux buts :

permettre des communications non bloquantes avec des périphériques externes ;
Commuter entre les tâches dans un ordonnanceur. Un autre usage, celui-là non prévu initialement, est l'introduction de malversations : lors de la restauration du contexte, si le contenu de la zone de sauvegarde a été altéré depuis l'appel (c'est le cas si l'interruption ou le déroutement provoque en mode maître une altération du contenu de la zone de sauvegarde ou de la pile), le contexte restauré sera totalement différent du contexte d'appel, et pourra passer la main à des suites d'instructions hostiles. Les systèmes comme le matériel des microprocesseurs s'efforcent de plus en plus de rendre ces tâches difficiles pour les pirates, mais la faille - bien que fortement réduite aujourd'hui - continue dans une certaine mesure à exister.

Quelle est la fonction d'un chien de garde, page 14 ?

Un chien de garde est un mécanisme (physique ou informatique) qui a pour rôle de repérer et de réagir aux anomalies qui peuvent survenir comme des pannes, une surconsommation ou même des attaques. Il peut dans ce cas là réinitialiser ou redémarrer le système.

Quel est l'intérêt d'utiliser des timers, page 15 ? Quel est le nom de l'interruption associée au timer 0 ?

Les timers permettent de programmer des actions sur des durées précises. On les utilise pour mesurer le temps, déclencher des événements ou temporiser des actions. À la page 15, l'interruption associée au timer 0 est TINT0.

Quelles sont les fonctions d'un échantillonneur bloqueur et d'un convertisseur analogique numérique,page 16 ? Indiquer le nombre d'entrées du module convertisseur analogique numérique, le nombre d'échantillonneurs bloqueurs et le nombre de convertisseurs analogiques numériques.

Ces 2 composants servent à convertir les signaux analogiques en signaux numériques.
L'échantillonneur bloqueur va prendre une valeur analogique et la garder constante pendant une période définie puis, le convertisseur analogique-numérique est utilisé pour convertir cette valeur analogique continu en un signal numérique.
Dans un DSP TMS320F2808, il y a 2 convertisseurs analogiques numériques, chacune ayant 16 entrées analogiques (A0 à A15) et chaque entrée ayant un échantillonneur bloqueur.

Quelles sont les fonctions du module Modulation de Largeur d'Impulsion, page 17 ?

Ce module permet de générer des signaux de modulation de largeur d'impulsion PWM (Pulse Width Modulation). C'est un moyen qui permet de contrôler la puissance fournie à un système en variant la largeur des impulsions d'un signal carré. On peut ainsi par exemple contrôler la puissance d'un moteur, l'intensité d'une LED, réguler des tensions, etc.

2 – Générateur de signaux :

- Expliquer la différence entre une grandeur continue et une grandeur discrète.

Une grandeur continue est une valeur qui peut prendre une infinité de valeurs dans un intervalle continue et connue. Sa variation peut être décrite par une fonction continue qui sera généralement graphiquement représentée par une courbe.
Une grandeur discrète quant à elle ne peut prendre qu'un nombre fini de valeurs, souvent obtenu par échantillonnage. Ces grandeurs sont souvent utilisées dans les systèmes numériques et les algorithmes de traitement du signal.

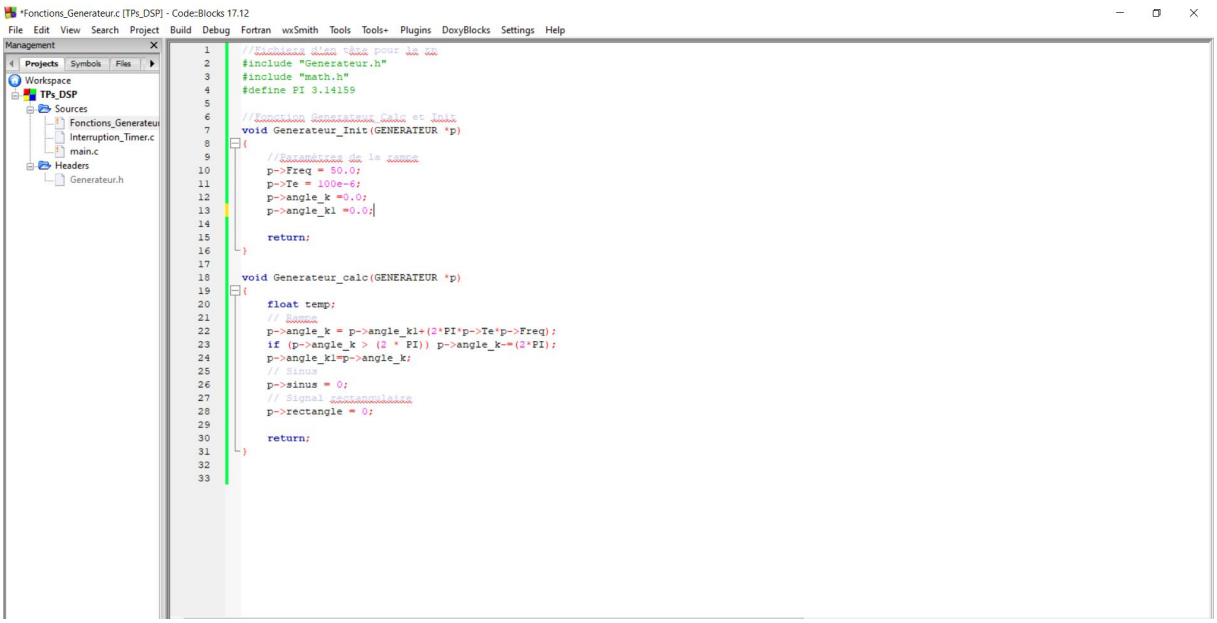
- A quoi correspond une équation de récurrence ?

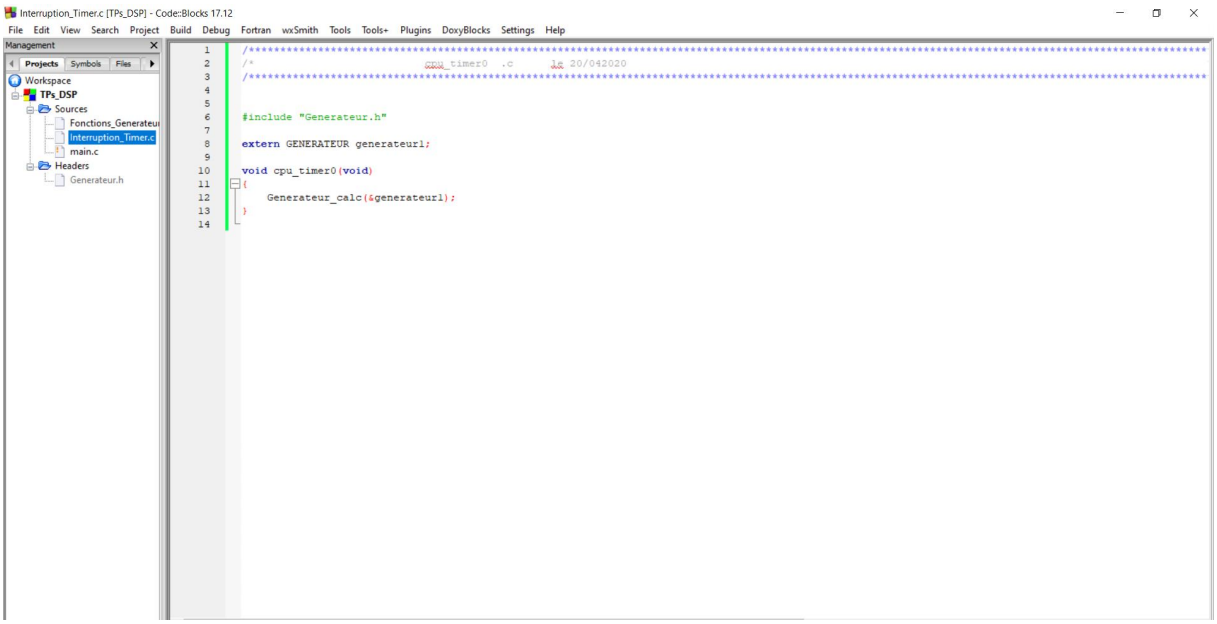
Une équation de récurrence est une équation qui définit ses termes à l'aide des termes précédents tel que X(n)=f(n-k).

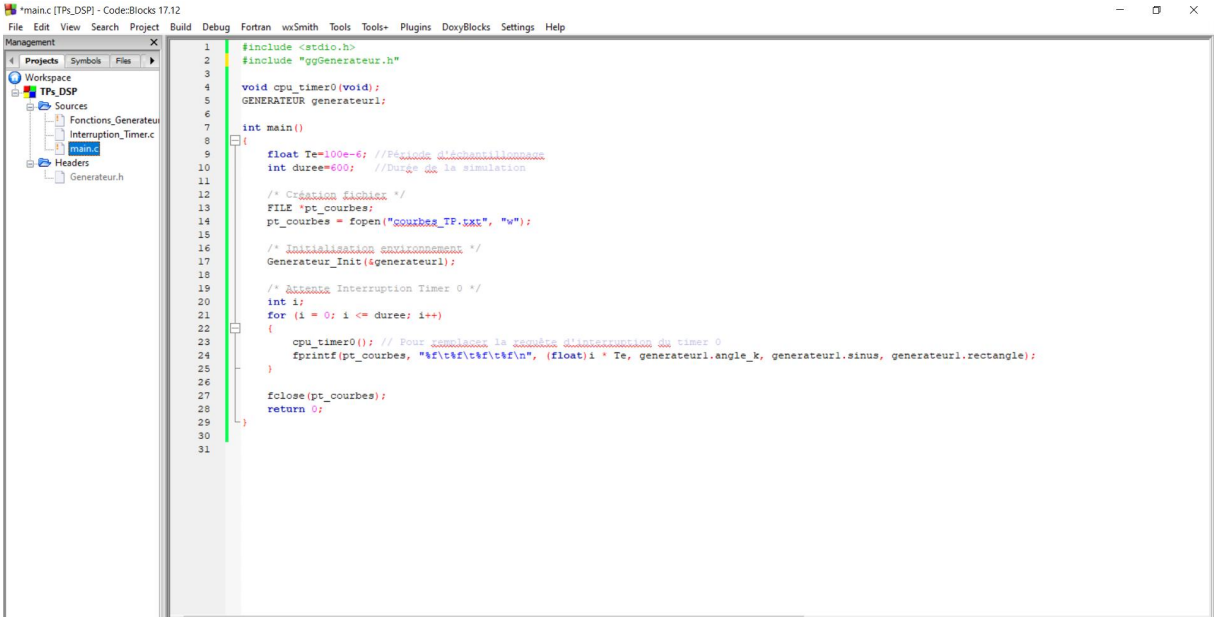
- Expliquer l'origine de l'équation de récurrence : θ(k)= θ(k-1) + 2*π*f*Te :

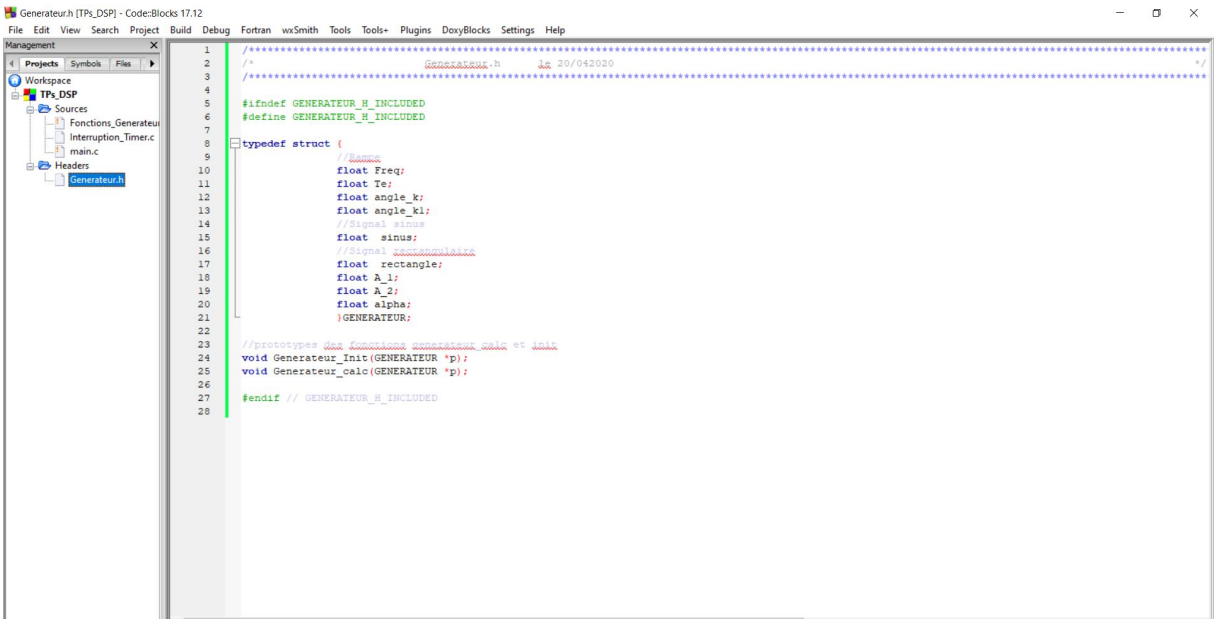
Cette équation récurrente est celle utilisée pour modéliser la variation d'un angle d'un système périodique. On voit que θ(k) dépends de θ(k-1), d'où le terme de récurrence. En effet, θ(k) représente l'angle à l'instant k, θ(k-1) représente l'angle à l'instant k-1, 2*π*f ou Oméga représente la pulsation et Te et la période d'échantillonnage. On utilise cette équation pour mettre à jour l'angle à chaque instant k dans les programmes de génération de signaux.

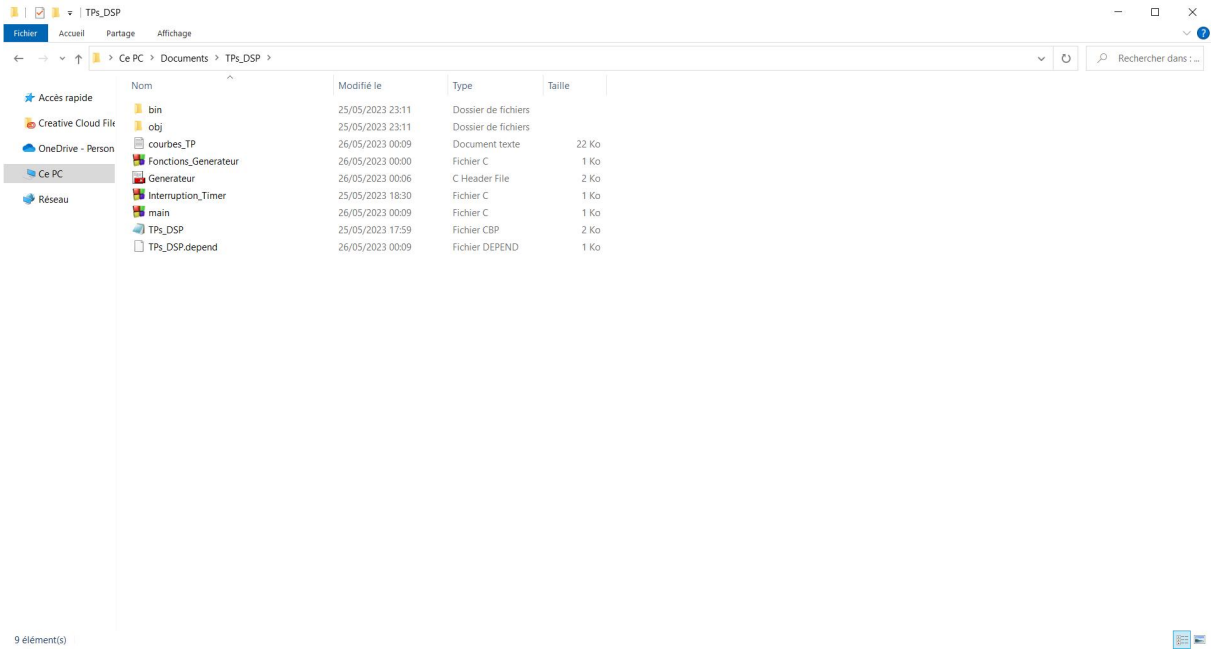
- **Nouveau Projet TPs_DSP :**






*main.c [TPs_DSP] - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
TPs_DSP
Sources
Fonctions_Generateur
Interruption_Timer.c
main.c
Headers
Generateur.h
#include <stdio.h>
#include "ggGenerateur.h"
void cpu_timer0(void);
GENERATEUR generateur1;
int main()
{
float Te=100e-6; //Période d'échantillonnage
int duree=600; //Durée de la simulation
/* Création fichier */
FILE *pt_courbes;
pt_courbes = fopen("courbes_TP.txt", "w");
/* Initialisation environnement */
Generateur_Init(&generateur1);
/* Attente Interruption Timer 0 */
int i;
for (i = 0; i <= duree; i++)
{
cpu_timer0(); // Pour simuler la survenue d'interruption du timer 0
fprintf(pt_courbes, "%f\t%f\t%f\t%f\n", (float)i * Te, generateur1.angle_k, generateur1.sinus, generateur1.rectangle);
}
fclose(pt_courbes);
return 0;
}


Generateur.h [TPs_DSP] - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
TPs_DSP
Sources
Fonctions_Generateur
Interruption_Timer.c
main.c
Headers
Generateur.h
/* Generateur.h le 20/042020 */
#ifndef GENERATEUR_H_INCLUDED
#define GENERATEUR_H_INCLUDED
typedef struct {
//Entrée
float Freq;
float Te;
float angle_k;
float angle_k1;
//Signal sinus
float sinus;
//Signal rectangulaire
float rectangle;
float A_1;
float A_2;
float alpha;
}GENERATEUR;
//prototypes des fonctions generateur_calc et init
void Generateur_Init(GENERATEUR *p);
void Generateur_calc(GENERATEUR *p);
#endif // GENERATEUR_H_INCLUDED



- Explication des lignes de code :

Ces lignes de code sont responsables du calcul de la rampe dans la fonction Generateur_calc.

- **p->angle_k = p->angle_k1 + (2 * PI * p->Te * p->Freq) ;** Cette ligne calcule la nouvelle valeur de p->angle_k en ajoutant à la valeur précédente (p->angle_k1) le déplacement correspondant à une rampe. La formule utilisée est (2 * PI * p->Te * p->Freq), où PI est une constante définie comme 3.14159, p->Te représente un intervalle de temps et p->Freq est la fréquence de la rampe.
- **if (p->angle_k > (2 * PI)) p->angle_k -= (2 * PI) ;** Cette ligne vérifie si la valeur de p->angle_k dépasse la plage d'une rampe complète (2 * PI). Si c'est le cas, on soustrait une rampe complète (2 * PI) pour maintenir la valeur de p->angle_k dans la plage appropriée.
- **p->angle_k1 = p->angle_k ;** Cette ligne met à jour la valeur précédente de p->angle_k (p->angle_k1) avec la nouvelle valeur calculée (p->angle_k). Cela prépare la mise à jour lors de l'appel suivant de la fonction Generateur_calc.

En résumé, ces lignes de code permettent de calculer et de mettre à jour la valeur de la rampe dans la structure GENERATEUR à chaque appel de la fonction Generateur_calc.

_ **Quel est le nombre d'interruptions réalisées par le programme ?**

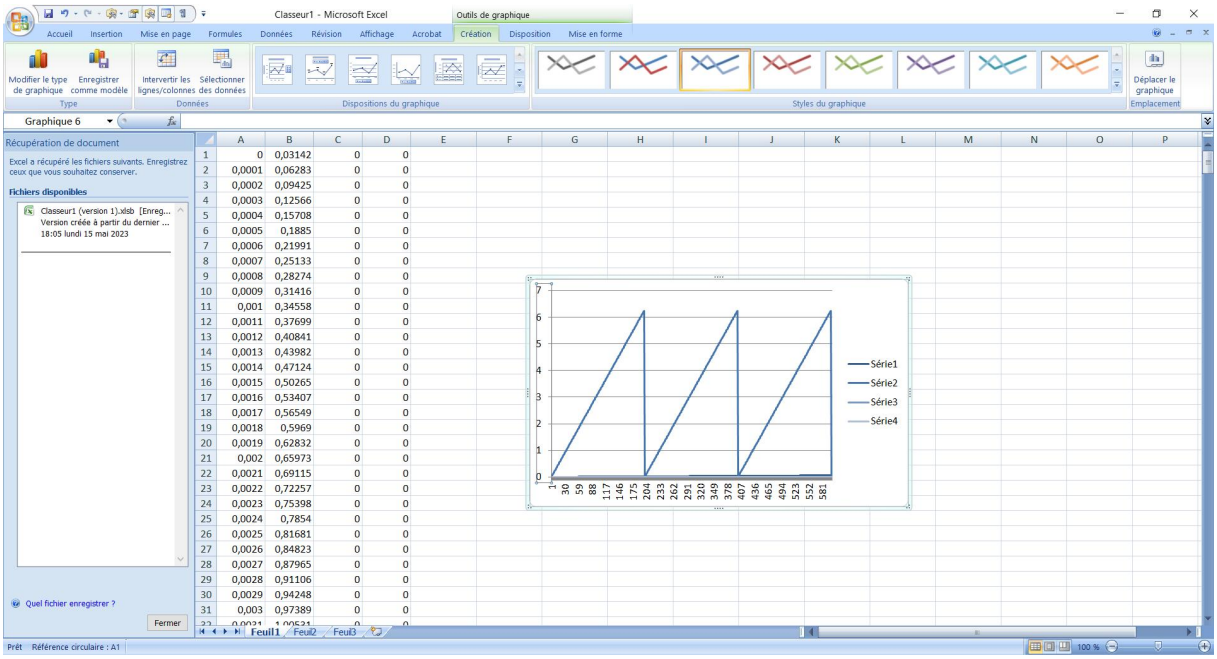
Dans le code donné, le nombre d'interruptions réalisées par le programme dépend de la valeur de la variable durée. La boucle for utilise durée comme condition de sortie pour contrôler le nombre d'itérations.

Dans ce cas précis, la boucle for itère de i = 0 à i <= durée, avec un pas de 1. Donc, le nombre total d'itérations, et par conséquent le nombre d'interruptions, sera égal à durée + 1.

_ **Quelle est la durée de la simulation :**

Dans le code, la valeur de durée est fixée à 600, ce qui signifie que la boucle sera exécutée 601 fois. Par conséquent, le programme réalisera 601 interruptions avant de se terminer.

_ Faire les captures d'écrans suivantes avec f=50 Hz et f=500 Hz et mettre vos noms :



- Modifier le fichier Generateur.h et fichier fonctions_Generateur.c :