# Examen “électronique programmable 2”

J.-M Friedt, 7 avril 2023

L'archive `https://github.com/jmfriedt/l3ep` contient dans son sous répertoire `FFT` une bibliothèque pour calculer la transformée de Fourier d'une séquence de mesures inspirée de la note d'application Maxim 3722 maintenant devenue introuvable sur le web [1]. L'archive originale `.zip` est fournie pour référence mais ne sera pas utile, tous les fichiers nécessaires ayant été extraits et modifiés pour les besoins de cet exercice. Exploiter une bibliothèque ne nécessite pas d'en comprendre le fonctionnement mais uniquement les fonctionnalités fournies et les types de données en entrée et en sortie.

- le fichier d'entête `fft.h` définit le nombre de points sur lesquels effectuer les calculs
- les types manipulés sont des entiers signés sur 16 bits dont seuls les 8 bits de poids faible représentent les données, donc comprises entre -128 et 127
- le prototype de la fonction `fft` est fourni dans `fft.h` : cette fonction prend en entrée les mesures (supposées réelles) et renvoie **dans le même tableau dont le contenu aura été écrasé** le module de la transformée de Fourier discrète de ces mesures.

Nous allons dans un premier temps nous familiariser avec l'utilisation *sur PC* :

1. proposer un `Makefile` qui compile un exécutable nommé `fft` à destination du processeur Intel qui équipe les PCs. On s'assurera que `maxqfft.c` et l'exécutable sont re-compilés sous les conditions que le code source (d'extension .c) **ou son fichier d'entête** (.h) soient modifiés. Quel compilateur utiliser dans ce cas ? On se rappellera que la bibliothèque mathématique se nomme `libm.a` : comment se lier à cette bibliothèque ?
2. **exécuter** le programme résultant et observer sa sortie. Nous avons simulé l'acquisition des données dans la fonction `getSamplesFromADC()` : le résultat est-il en accord avec vos attentes ? Justifier. On pourra par exemple tracer la séquence de sortie avec son outil favori (Octave, Python, gnuplot ...)
3. Modifier le `Makefile` pour compiler à destination de l'Atmega32U4 après avoir ajouté une méthode `clean` qui efface les objets et l'exécutable du répertoire de travail. Quel compilateur est désomais utilisé ? Pourquoi avoir ajouté la méthode clean lors du changement d'architecture d'Intel à Atmel ?
4. Cette fois l'affichage pris en charge par `usb_write()` ne peut plus s'effectuer par `printf()` puisque l'Atmega32U4 ne connaît pas le concept de console, mais doit passer par USB. Modifier donc `main.c` pour initialiser le port USB de l'Atmega32U4 et proposer une fonction `usb_write()` qui affiche le résultat du calcul sur le port USB sondé par `minicom`. Démontrer son bon fonctionnement en comparant la sortie de l'Atmega32U4 avec le résultat obtenu auparavant sur PC.
5. Nous désirons analyser la transformée de Fourier d'un créneau : modifier la fonction `getSamplesFromADC()` pour remplir le tableau de données non pas d'une sinusoïde mais d'un créneau de période de 20 échantillons et de rapport cyclique de 50% avec des valeurs comprises entre -127 et 127. Comment est effectuée cette opération ?
6. Lors du remplissage du tableau de données, tester le bit de poids fort de l'octet de poids faible du mot de 16 bits, et imposer à 0xff l'octet de poids fort du mot de 16 bits si ce bit est à 1, et laisser l'octet de poids fort en l'état sinon. Comment tester le bit de poids fort de l'octet de poids faible et comment imposer à 1 les bits de poids fort de l'octet de poids fort ? Quelle représentation des nombres pour simplifier l'arithmétique justifie cette extension du bit de poids fort ?
7. Au lieu de tracer la transformée de Fourier discrète, tracer dans un premier temps la séquence temporelle pour vérifier son adéquation avec la définition proposée ci-dessus.
8. Tracer et analyser la transformée de Fourier discrète du créneau : répond-elle à vos attentes ? Justifier
9. Modifier le créneau pour qu'il soit de période de 10 échantillons et toujours de rapport cyclique de 50% : comment a été modifiée la transformée de Fourier discrète de cette séquence ?
10. Quelle est l'espace occupé par l'exécutable en mémoire non-volatile du microcontroleur ? Comment cette information a-t-elle été obtenue ? Comment cette taille se compare-t-elle avec l'espace disponible ? Justifier, notamment comment avez vous obtenu la seconde information ?
11. Quelle est l'occupation en mémoire volatile ? Comment cette information a-t-elle été obtenue ? Comment cette taille se compare-t-elle avec l'espace disponible ? Justifier, notamment comment avez vous obtenu la seconde information ?

## Références

[1] Maxim, *Developing FFT Applications with Low-Power Microcontrollers*, note d'application 3722 (23 Mars 2006)

# Réponses

1. Le `Makefile` est de la forme

```
CC=gcc

all: fft

main.o: main.c
	$(CC) -c main.c

maxqfft.o: maxqfft.c maxqfft.h
	$(CC) -c maxqfft.c

fft: main.o maxqfft.o
	$(CC) -o fft maxqfft.o main.o -lm
```

   qui utilise `gcc` comme compilateur, compile les objets `main.o` contenant la fonction principale `main()` et `maxqfft.c` contenant l'implémentation de la fonction `fft()`, et la génération de l'exécutable lie les deux objets pour résoudre les dépendances. Nous ajoutons `maxqfft.h` comme dépendance de la compilation de `maxqfft.o` afin de forcer la recompilation si le fichier d'entête est modifié. L'édition de lien avec la bibliothèque mathématique `libm.a` s'obtient par `-lm` lors de la génération de l'exécutable

2. l'exécution du programme se traduit par l'affichage du module de la transformée de Fourier que nous redirigeons dans un fichier par `./exec > fichier`. Le contenu de ce fichier se trace par exemple dans `gnuplot` par `plot "fichier" using line` et une unique raie spectrale est observée. Le signal est une sinusoïdéfinie par `x_n_re[i]=(short)(127*sin((float)(i)/4.));` donc une fonction rélle dont le module de la transforée de Fourier est paire. La position du pic à l'abscisse d'indice 10 cohérent avec la définition du signal puisque la FFT sur 256 points (constante `N` dans `maxqfft.h`) donne 128 points dans la moitié réelle du spectre et l'argument de la fonction trigonométrique s'écrit $\sin(2\pi \cdot f \cdot t)$ donc $2\pi \times f = 1/4 \Leftrightarrow f = 1/(4 \times 2\pi)$ donc l'abscisse du pic est $256/2/\pi/4 = 10$ en accord avec notre observation

3. Nous modifions dans le `Makefile` le compilateur utilisé `CC=avr-gcc -mmcu=atmega32u4` et ajoutons une règle

```
clean:
	rm *.o fft
```

   qui efface les objets et l'exécutable. Sans cette règle, nous risquerions de lier des objets pour Atmega32U4 avec d'anciens objets pour processeur intel : lier des fichiers pour des cibles différentes ne peut aboutir et se traduit par l'échec de l'édition de lien pour générer l'exécutable. Il faut donc toujours nettoyer tous les objets et exécutables lors du changement de cible.

4. On ajoute

```
void usb_write(char *c,short s)
{sprintf(c,"%hd\n",s);}
```

   pour afficher la variable sur 16 bits `s` dans le tableau de caractères (octets) `c`. Le programme est lié avec la bibliothèque LUFA pour communiquer sur le bus USB.

5. Le créneau de période 20 échantillons et de rapport cyclique 50% est défini par

```
void getSamplesFromADC(short *x_n_re)
{int i;
 for (i=0;i<N;i++)
  {if ((i%20)<10) x_n_re[i]=127; else x_n_re[i]=-127;
   if (x_n_re[i]&0x0080) x_n_re[i]|=0xFF00;
  }
}
```

6. la représentation en complément à deux doit propager le bit de poids fort représentatif du signe lors de l'extension de la taille de la donnée de 8 à 16 bits. Le test du bit de poids fort s'obtient par masquage, et la propagation du bit de signe avec un "ou" logique sur l'octet de poids fort,

7. la séquence temporelle pour valider la forme du créneau se trace en commentant la ligne `fft(x_n_re);` pour afficher les signaux d'entree

8. Cette fois le composante fondamentale du signal se trouve en abscisse $256/20 = 12,8$ qui apparaît en abscisse 13, suivie des harmoniques impaires qui caractérisent la transformée de Fourier d'un créneau.
9. la fréquence augmente donc les pics dans la transformée de Fourier se décalent vers les hautes fréquences : le mode fondamental est à $256/10 = 25,6$ qui apparaît en 26 suivi des harmoniques impaires.
10. Les deux fonctions pour obtenir l'occupation en mémoire non-volatile sont `avr-size` (champ `text`) ou `avr-objcopy -Obinary`
11. Seule `avr-size` permet d'évaluer l'occupation en mémoire volatile