Rapport d’avancement : BE intégration, partie ASM

1. Emission d’un signal carré

Le but ici est de mettre en place l'émission d’un signal carré grâce au timer intégré à la carte, de fréquence fixe, pour pouvoir plus tard émettre un signal audio.

Tout d’abord nous avons récupéré la bibliothèque GASSP72, pour le traitement des interruptions Timer.

Import de la librairie dans le code c et déclaration du handler :

```
1  #include <gassp72.h>
2  #include <etat.h>
3
4  extern void timer_callback(void);
```

On doit générer des interruptions dans le code principal.c grâce à la librairies GASSP72 :

```c
#include <gassp72.h>
#include <etat.h>

extern void timer_callback(void);

int Periode_en_Tck = 72*10*3;



int main(void)
{
                // activation de la PLL qui multiplie la fréquence du quartz par 9
                CLOCK_Configure();
                // config port PB1 pour être utilisé en sortie
                GPIO_Configure(GPIOB, 1, OUTPUT, OUTPUT_PPULL);
                // initialisation du timer 4
                // Periode_en_Tck doit fournir la durée entre interruptions,
                // exprimée en périodes Tck de l'horloge principale du STM32 (72 MHz)
                Timer_1234_Init_ff( TIM4, Periode_en_Tck);//Periode_en_Tck );
                // enregistrement de la fonction de traitement de l'interruption timer
                // ici le 2 est la priorité, timer_callback est l'adresse de cette fonction, a créér en asm,
                // cette fonction doit être conforme à l'AAPCS
                Active_IT_Debordement_Timer( TIM4, 2, timer_callback);//timer_callback );
                // lancement du timer
                Run_Timer( TIM4 );
        while   (1)
        {

        }
}
```

On doit ensuite créer le handler pour les interruption en langage d’assemblage pour générer le signal carré :

- On exporte le timer_callback.
- Comme on veut la sortie sur PB1, on doit mettre à 0 ou à 1 cette sortie, grâce à la librairie GASSP72 (On a configuré cette sortie dans le principal.c).
- On initialise une variable tot, qui va valoir successivement 00 et FF, (tot est sur 2 octets), grâce à l’instruction MVN.
- On compare cette variable à zéro pour avoir le flag à 0 ou 1.
- On utilise le mnémo BEQ, pour sauter à on si le flag vaut 1 (PB1 à 1), ou non (PB1 à 0).

```
1
2
3           area  madata, data
4   tot     dcd     0
5   GPIOB_BSRR      equ     0x40010C10      ; Bit Set/Reset register
6
7           area  moncode, code
8           export  timer_callback
9           include etat.inc
10
11  timer_callback proc
12
13          ldr r10,=tot
14          LDRSB R9,[r10]
15          MVN r9,r9
16          STRB r9,[r10]
17
18
19
20          CMP r9,#0
21          BEQ on
22
23          ; mise a 1 de PB1
24          ldr     r3, =GPIOB_BSRR
25          mov     r1, #0x00000002
26          str     r1, [r3]
27          B out
28  on
29                  ; mise a zero de PB1
30          ldr     r3, =GPIOB_BSRR
31          mov     r1, #0x00020000
32          str     r1, [r3]
33
34  ; N.B. le registre BSRR est write-only, on ne peut pas le relire
35  out     BX lr
36          endp
37          end
38
```

On génère bien ainsi un signal carré.

2. DFT virgule fixe.

Dans cette deuxième partie, nous devons effectuer le calcul de la transformée de Fourier discrète, pour effectuer le traitement du signal décrit dans la partie Signal de ce bureau d'études.

Comme il est impossible de calculer directement des sinus et des cosinus en langage d'assemblage, nous nous basons sur des tables de résultats précalculées.

En discret nous avons besoin d'autant de données de tables que nous calculerons de points, nous avons choisi de retenir 64 points et donc de générer une table de sinus et cosinus à 64 résultats de chaques.