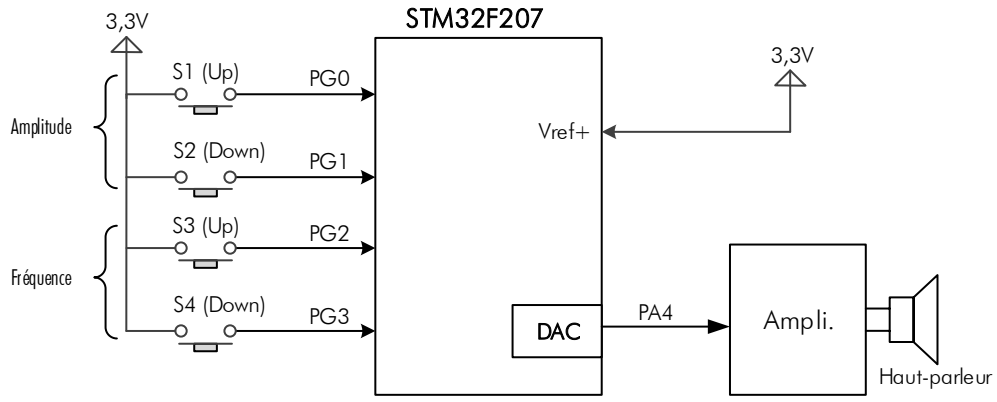
TP INTERRUPTION

Nous allons utiliser une carte base de microcontr6leur STM32 possédant un amplificateur audio (connecté à un haut-parleur) et 4 boutons poussoirs. Nous avons pour but de générer un son sous forme d'une sinusoïde dont nous réglerons le volume par 2 boutons et la fréquence par les 2 autres boutons.



1. Initialisation des Port

La procédure Init_Boutons( ) permet la configurations des lignes PG0 à PG3.

2. Fonctionnalité des boutons

On vent que l'appui sur chacun des boutons conduise à une modification de 2 variables globales définies sur des entiers : g_amplitude et g_frequence. La modification de ces variables permettra par la suite de contrôler le son émis.

g_amplitude représente l'amplitude de la sinusoïde exprimée en dixième de volt. Cette amplitude initialisée à 0, 1V (donc g_amplitude =1) va évoluer entre 0 et 3,3V (donc g_amplitude entre 0 et 33) par pas de 0,1 V (donc des pas de 1 sur g_amplitude).

g_fréquence représente la fréquence de la sinusoïde exprimée en Hertz. Cette fréquence initialisée à 100Hz (donc g_frequence =100) va évoluer entre 0 et 4kHz (donc g_frequence entre 0 et 4000) par pas de 100 Hz (donc des pas de 100 sur g_frequence).

Pour être plus précis, on vent que l'appui :

- Sur Bouton S1 conduise à l'augmentation de l'amplitude (incrAmpl)
- Sur Bouton S2 conduise à la diminution de l'amplitude (decrAmpl)
- Sur Bouton S3 conduise à l'augmentation de la fréquence (incrFreq)
- Sur Bouton s4 conduise à la diminution de la fréquence (decrFreq)

Attention à ne pas oublier la saturation.

a) Ecrivez en C les 2 fonctions void incrAmpl (void) et void decrFreq (void).

3. Déclenchement d'interruptions sur appui des boutons

Nous allons utiliser la particularité de générer des interruptions sur des signaux externes, Nous allons utiliser les interruptions EXTI0 a EXTI3 qui correspondront respectivement aux lignes PG0..3. II est en effet

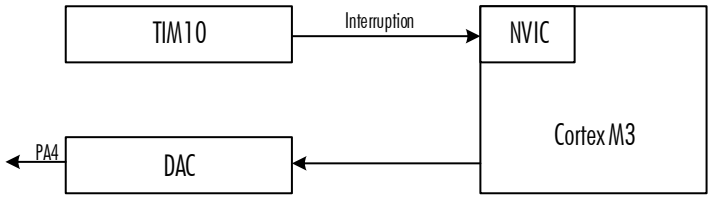
possible de programmer le STM32 de manière à générer une interruption EXTI0 lors d'une modification du niveau logique sur l'entre EXTI0 de 0 vers 1 (ce qui correspond à l'appui sur le bouton 1) (déclenchement sur un front). II en sera de même pour EXTI1 lors de l'appui sur le bouton 2, EXTI2 lors de l'appui sur le bouton 3, EXTI3 lors de l'appui sur le bouton 4.

b) pourquoi et à quel moment de la réalisation, est-il intéressant d'interdire les interruptions ?

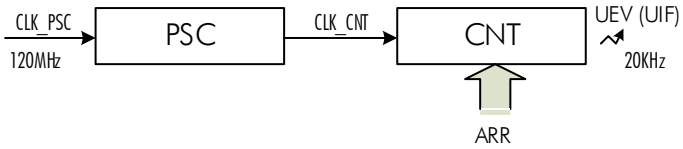
c) Ecrivez le code C de la routine d'interruption « void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin). »

3. Gestion du son

Le principe permettant ici de générer un son est illustré ci-dessous. Un Timer (N°10) envoie régulièrement une interruption au contrôleur d'interruption (NVIC). L'interruption ainsi déclenche, calcule la valeur d'un échantillon à convertir puis lance la conversion sur un DAC dont la sortie analogique est connecte sur la sortie PA4.



On veut que la fréquence à laquelle doivent intervenir les conversions soit 20kHz. La fréquence du bus APB2 est de 60Mhz.



d) Ecrire le pseudocode de la configuration du TIM10 en mode interruption.

4. Conversion numérique analogique

L'utilisation du DAC est relativement simple. Après l'avoir activé, il faut écrire la valeur à convertir dans le registre DAC_HDRXXXX.

On a décidé de générer un signal sinusoïdal pour produire du son. Pour cela, on a rangé 20 échantillons régulièrement espacés (en temps) d'une période d'un sinus sous la forme d'une variable globale :

```
const   int  g_sin[21] ={ 0, 39, 75, 103, 121, 127, 121, 103, 75, 39, 0, -39, -
75, -103, -121, -127, -121, -103, -75, -39, 0};
```

En admettant que nous avez réglé la génération des interruptions de TIMER10. Si la routine de traitement des interruptions de TIMER10 est la suivante :

```
int   i   ;
TIM10 -> SR= 0 ;
g_time++ ;
i = g_time%20 ;
DAC ->DHR12R1 = g_offsetADC + g_amplADC* g_sin[i] ;
```

Où g_time, g_offsetADC et g_amplADC sont des variables globales qui ensuite dépendront des variables globales g_frequence et g_amplitude.

e) A quelle valeur doit être initialisée g_offsetADC pour que la valeur moyenne du signal de sortie soit centre (c'est-dire VREF/2).

f) Si g_amplADC=1, quelle est la valeur en volt de l'amplitude de la sinusoïde observé en PA4 ? (amplitude d'une sinusoïde =(Vmax - Vmin)/2).

g) Quel doit être l'amplitude maximale de g_amplADC pour ne pas dépasser la gamme de tension possible de PA4 ?

h) Quelle est la fréquence du signal observé en PA4 ?

5. Réglage de priorité :

Le calcul des échantillons et l'envoi vers le DAC (déclenchés par les interruptions TIMER10) doivent être prioritaires sur les interruptions issues des boutons. Les interruptions issues des boutons partagent le même niveau de priorité.

i) Quels sont les registres permettant d'autoriser ces 5 interruptions ? et quelles valeurs leur donner pour les autoriser ?

j) Quels sont les registres qui vont permettre de régler les priorités de ces 5 interruptions ?

k) Ecrire le code C de la procédure d'initialisation des interruptions Init_Interrupts().

15.5.2 TIM10/11/13/14 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled