LE DIMNA Mathieu
mathieu.le-dimna@yncrea.fr

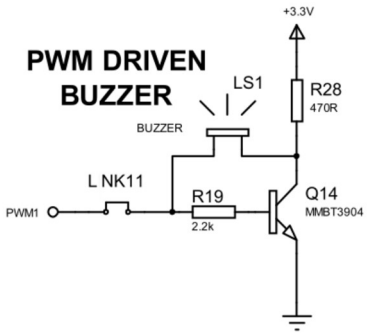
Buzzer

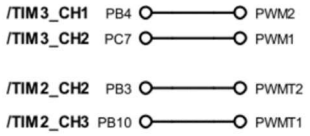
Objectif

Le but de cet exercice est de manipuler les timers pour piloter le buzzer de la carte ISEN32. A la fin de cet exercice vous serez en mesure de jouer une partition stockée dans un tableau !

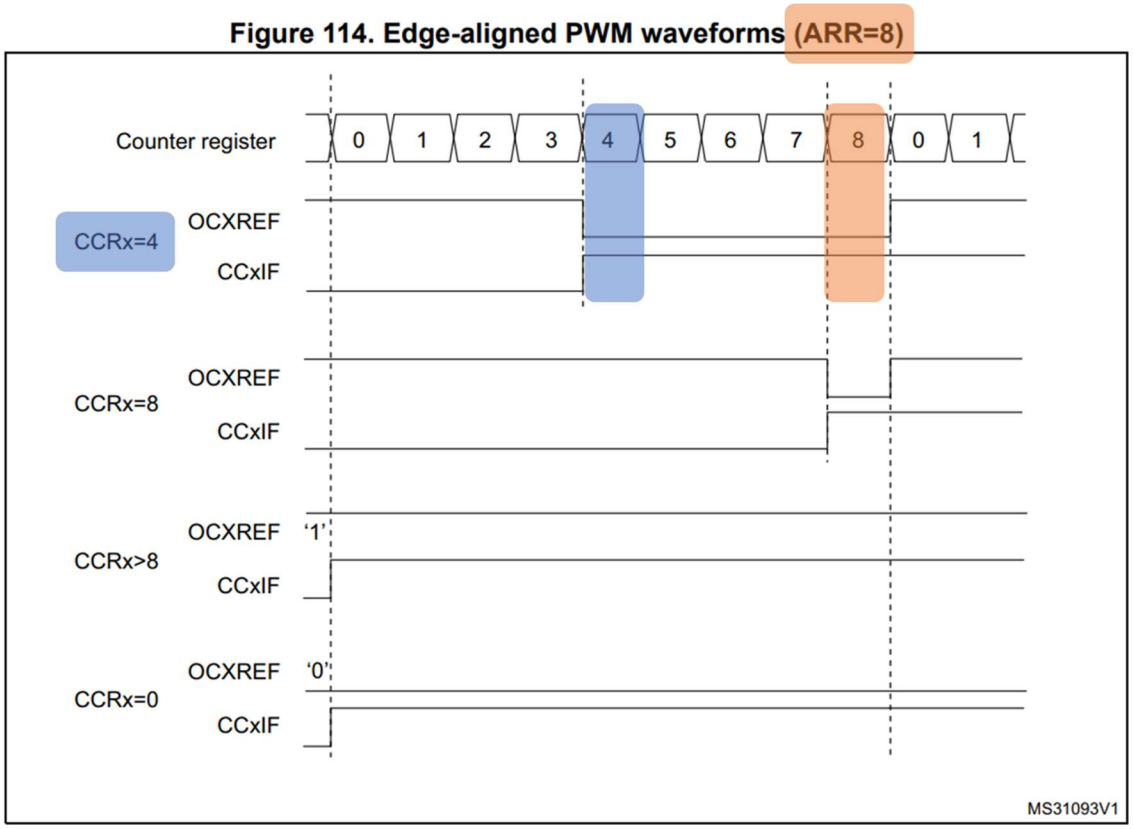
NOTE	FRÉQUENCE[HZ]
C5	1046,50
C#5	1108,73
D5	1174,66
D#5	1244,51
E5	1318,51
F5	1396,91
F#5	1479,98
G5	1567,98
G#5	1661,22
A5	1760,00
A#5	1864,66
B5	1975,53

Le tableau ci-contre fait le lien entre les notes en gamme de Do de la 5e octave et leurs fréquences associées. Ce tableau servira de référence pour votre code, il sera possible de convertir ces fréquences en valeurs permettant de configurer un timer du STM32 et jouer les notes.





Pour émettre une note il faut générer un signal sur la pin PC7 du STM32 embarqué sur la carte Nucleo. Cette pin est reliée au transistor du schéma ci-dessus par le net PWM1. Vous devrez utiliser le mode PWM du timer TIM3 sur le channel CH2. Observez le comportement du signal interne OCxREF[6] en fonction des valeurs de registre ARR et CCRx.

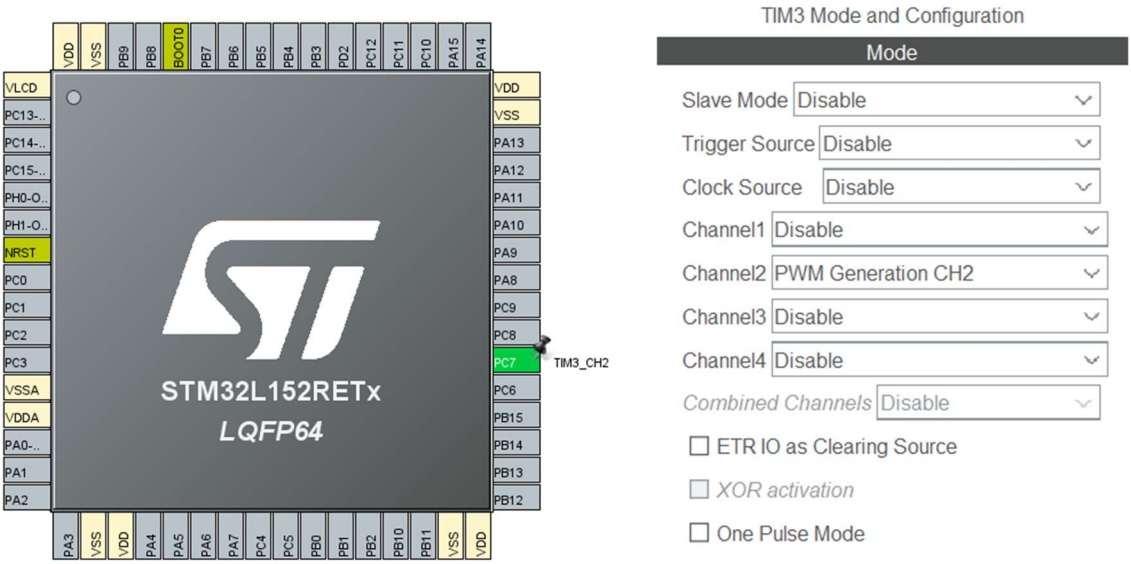


[6] Le X est remplacé par le numéro de canal (Ex : 2 pour CH2)

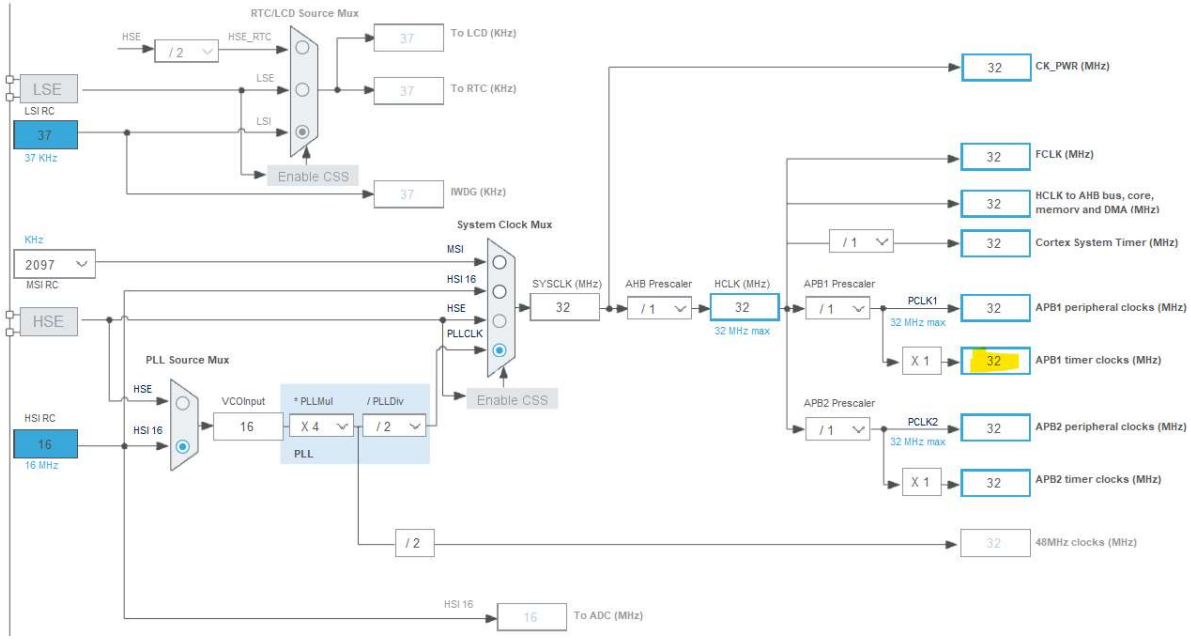
LE DIMNA Mathieu
mathieu.le-dimna@yncrea.fr

Fichier IOC

Paramétrez la pin associée au pilotage du buzzer comme étant liée au périphérique TIM3 CH2 et paramétrez le Channel 2 en mode PWM.

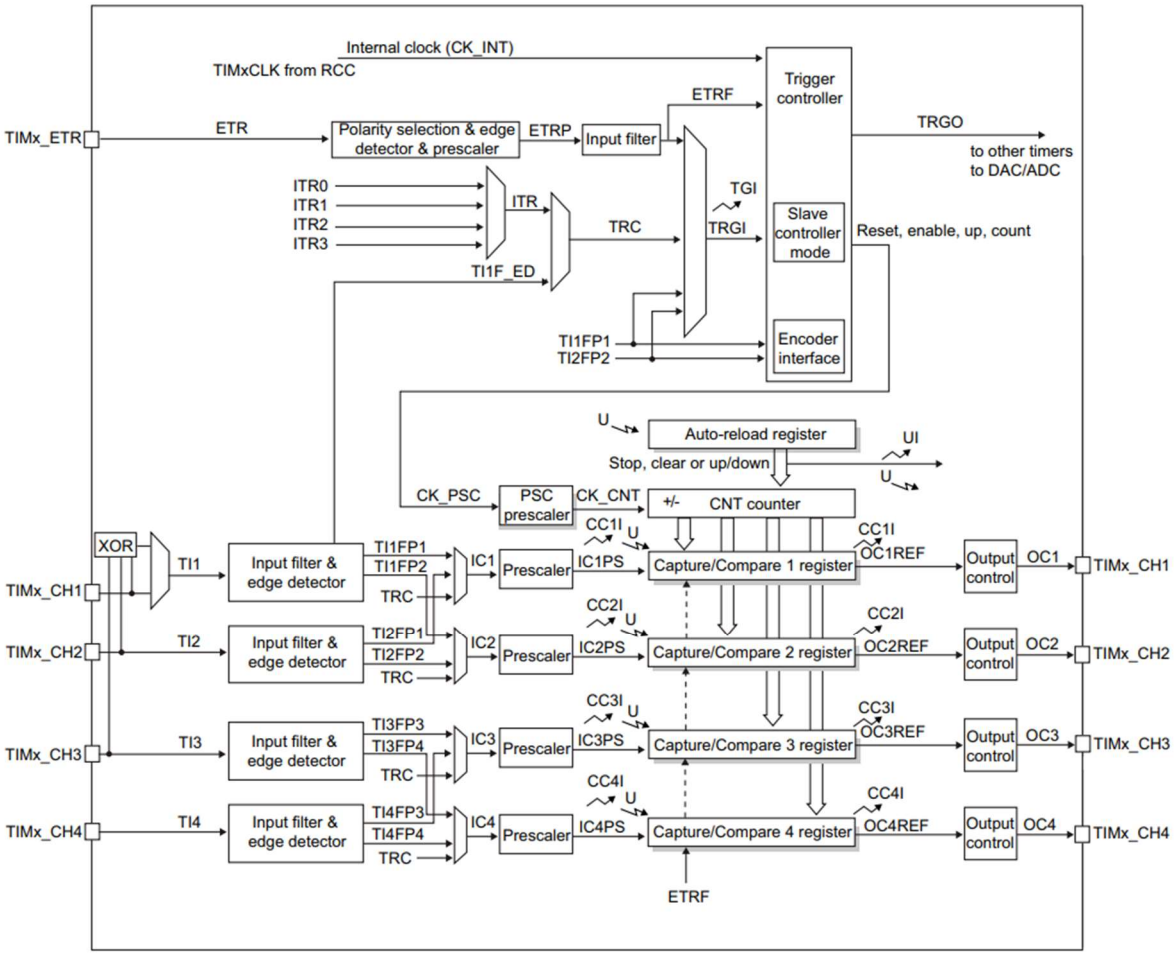


Une fois le GPIO configuré vous pouvez dès à présent paramétrer les valeurs appliquées au périphérique TIM3 pour la génération du signal PWM. Vérifiez que la fréquence appliquée aux timers APB1 est bien de 32MHz



LE DIMNA Mathieu
mathieu.le-dimna@yncrea.fr

Calcul des différentes valeurs



Fréquence de comptage

La fréquence de comptage est fondamentale pour calculer les différentes valeurs utiles à notre signal PWM. Elle définit le temps que met le timer à incrémenter le registre CNT. *Notez que notre fréquence f_{CK_INT} a été soigneusement définie à 32MHz. La période de comptage (en secondes) est l'inverse de la fréquence de comptage.*

$$f_{CK\_CNT} = \frac{f_{CK\_INT}}{PSC + 1}$$
$$f_{CK\_CNT} : Fréquence\ de\ comptage$$
$$PSC : Prescaler, permet\ de\ diviser\ l'horloge\ d'entrée\ du\ timer$$
$$f_{CK\_INT} : Fréquence\ d'entrée\ du\ timer$$

Fréquence de débordement

Le débordement est la base du timer, la valeur appliquée au registre ARR du timer est comparée à chaque incrément du registre CNT, si celle-ci correspond le registre CNT est remis à zéro. De cette manière le timer exécute son comptage périodiquement.

$$t_{ARR} = t_{CK\_CNT} * (ARR + 1)$$
$$t_{ARR} : Période\ de\ rafraichissement\ du\ comptage$$
$$ARR : Registre\ d'auto - rafraichissement$$

Temps de comparaison

Pour générer le signal PWM il faut contrôler indépendamment le temps à l'état haut et la période du signal. La période est réglée par le registre ARR vu précédemment, pour le temps à l'état haut il faut se référer au registre CCRx[7].

$$t_{OCx} = t_{CK_{CNT}} * CCRx$$
$$t_{OCx} : Période \text{ à } l'état\ haut\ du\ signal\ PWM$$
$$CCRx : Valeur\ du\ registre\ de\ Capture\ Compare$$

Avec toutes ces informations, vous serez en mesure d'éditer les registres du timer 2 Channel 3 du STM32 pour générer une note de musique.

Code utilisateur

Pour qu'un son audible sorte du buzzer il est **nécessaire** que le temps à l'état haut du signal PWM soit fixée à 200µs peu importe la fréquence de la note. Vous modifierez donc seulement la valeur du registre ARR pour changer la fréquence jouée au buzzer.

Préparation

Créez une structure délcarée en *typedef* nommée *TypeDef_Note* qui aura comme membres

- **name** *: const char *,* nom de la note (Ex : C#5)
- **frequency** *: double*, fréquence de la note en hertz
- **ARR** : *uint16_t*, valeur à appliquer au registre ARR du timer

Appelez le professeur si vous avez des doutes sur ces types.

Créez 3 fonctions

- **buzzer_play_note** : Joue une note au buzzer
  - Paramètres
    - **_htim** : *TIM_HandleTypeDef *,* pointeur vers la structure de pilotage du Timer (Définit par la librairie HAL)
    - **_note** : *TypeDef_Note *,* pointeur vers la note à jouer
  - Type de retour : *void*
- **buzzer_mute** : Coupe le son du buzzer
  - Paramètres
    - **_htim** : *TIM_HandleTypeDef **, pointeur vers la structure de pilotage du Timer (Définit par la librairie HAL)
  - Type de retour : *void*
- **buzzer_play_note_by_name** : Joue une note à partir d'une chaine de caractères, si elle correspond à une note du tableau passé en paramètre la note sera jouée. Elle appellera *buzzer_play_note* et *buzzer_mute*.
  - Paramètres
    - **_htim** : *TIM_HandleTypeDef *,* pointeur vers la structure de pilotage du Timer (Définit par la librairie HAL)
    - **_notes** : *TypeDef_Note **, tableau de notes
    - **_notes_sz** : *size*_t, taille du tableau de notes
    - **_name** : *const char **, nom de la note à jouer

[7] X étant le channel choisi

- Type de retour : *void*

Dans votre code, déclarez le tableau suivant

```
TypeDef_Note notes[] = {
      {"C5",      1046.50, 0},
      {"C#5",     1108.73, 0},
      {"D5",      1174.66, 0},
      {"Eb5",     1244.51, 0},
      {"E5",      1318.51, 0},
      {"F5",      1396.91, 0},
      {"F#5",     1479.98, 0},
      {"G5",      1567.98, 0},
      {"G#5",     1661.22, 0},
      {"A5",      1760.00, 0},
      {"A#5",     1864.66, 0},
      {"B5",      1975.53, 0}
};
```

Et la définition suivante

```
#define MUTE "-"
```

Notez que les valeurs d'ARR ne sont pas initialisées dans le tableau. Créez une procédure d'initialisation qui paramètre les valeurs d'ARR en fonction de la fréquence.

Voici la partition à jouer

```
const char* buzzer_partition[] =
{"G5",MUTE,"G5",MUTE,"G5",MUTE,"A5",MUTE,"B5","B5","B5","B5","A5","A5","A5"
,"A5","G5",MUTE,"B5",MUTE,"A5",MUTE,"A5", MUTE,"G5","G5","G5","G5",MUTE};
```

Dans votre ***while(1)*** vous parcourrez le tableau en jouant une note toutes les 100ms. (Utiliser HAL_Delay ou autre mécanisme de temporisation).

Pour modifier le registre ARR vous pouvez y accéder directement depuis la structure TIM_HandleTypeDef

```
htim2.Instance->ARR = /* Nouvelle valeur */;
```