ELE3312
Microcontrôleurs et applications
Laboratoire 8

Auteur : Jean Pierre David

Introduction

Lorsqu'on a besoin d'échantillonner à une fréquence précise et stable, on peut connecter la sortie d'une horloge directement sur l'ADC ou le DAC. Mieux, on peut même demander au contrôleur DMA de se charger du transfert entre la mémoire et les périphériques. De cette manière, tout se passe comme si vos tableaux de données écrits en C ou en ASM se remplissent tout seul à partir de l'ADC ou, à l'inverse, comme si votre tableau était envoyé tout seul au DAC.

Objectifs

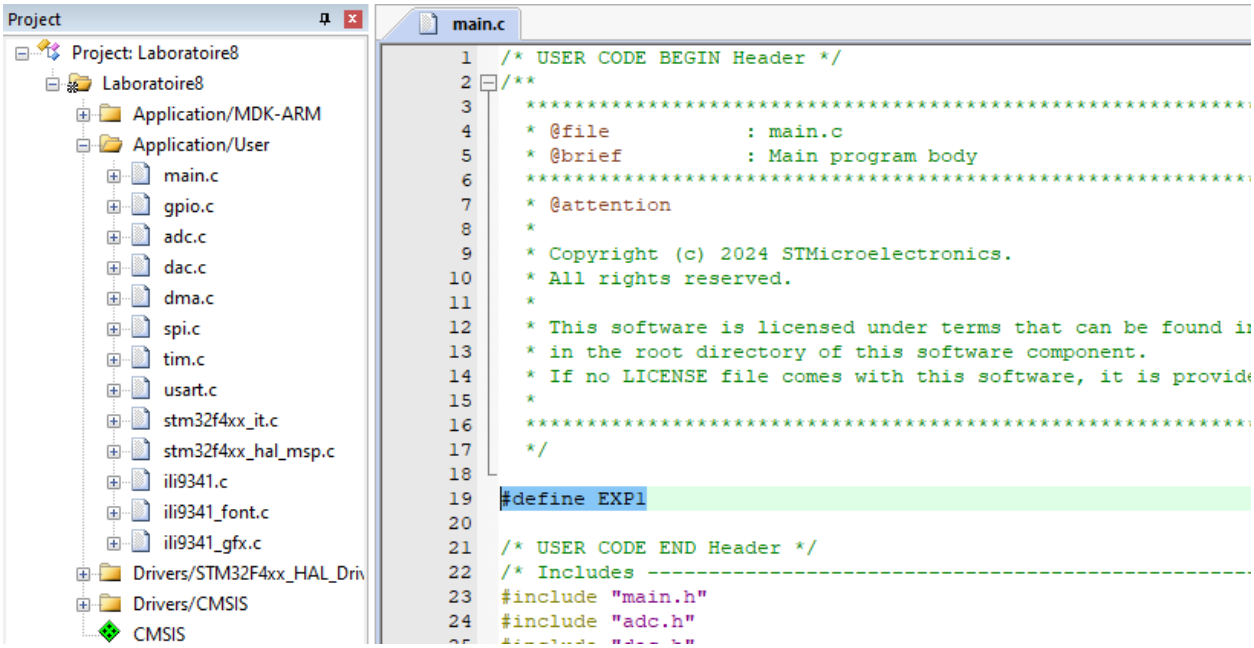
1. Utiliser les horloges comme déclencheur de l'échantillonnage
2. Découvrir et utiliser le DMA
3. Découvrir et utiliser le DAC

1ère partie : préparation à la maison

Un projet de départ est fourni dans l'archive **projet_laboratoire8.zip**.

Ce projet se fonde sur le laboratoire 7. Vous devriez donc déjà avoir la configuration pour les broches du clavier, et assurez-vous d'avoir le code qui permet de rediriger la sortie standard vers l'USART (pour que les `printf` s'affichent sur la console *Putty* – voir laboratoire 3).

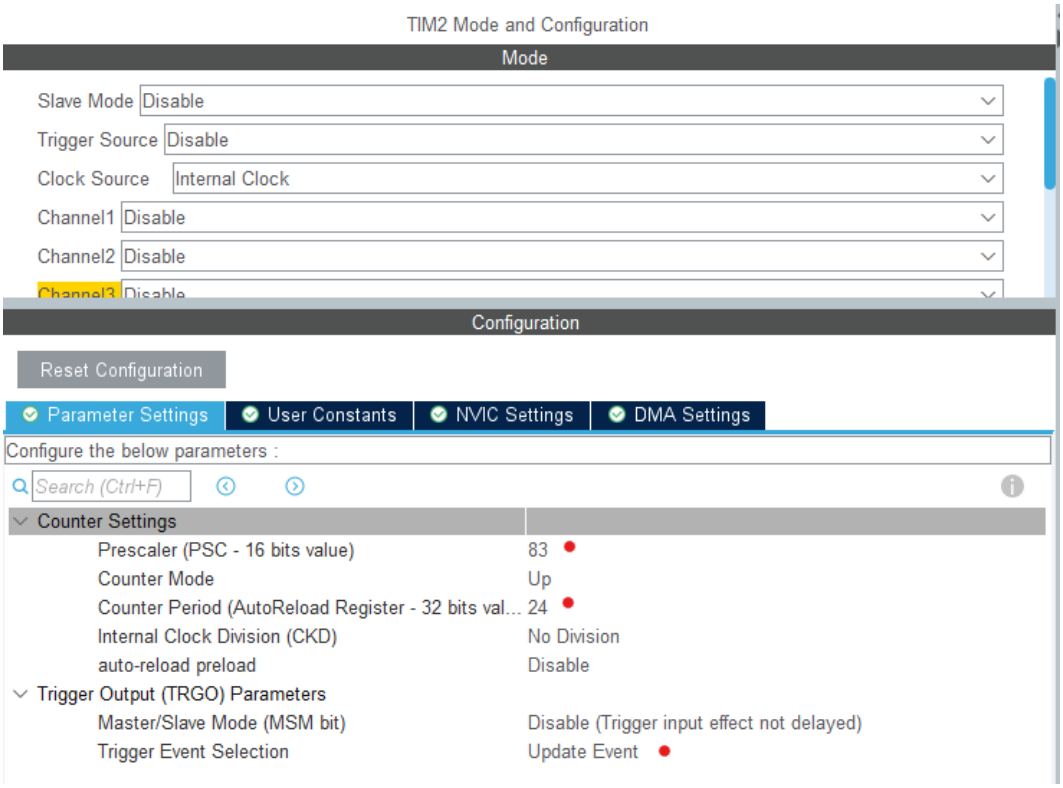
Le code nécessaire aux expériences 1 et 2 est inclut. Il suffit de définir l'expérience choisie à l'aide d'une macro (`EXP1` ou `EXP2`) en utilisant la directive préprocesseur `#define` (voir capture). Ainsi, le code correspondant sera compilé.



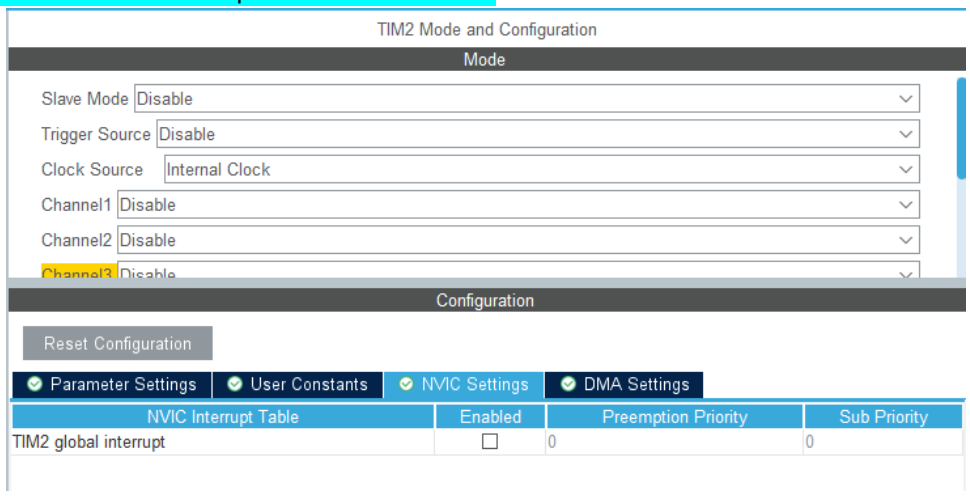
ATTENTION : les instructions de configuration surlignées en bleu dans la suite de l'énoncé doivent tout de même être effectuées.

Expérience 1

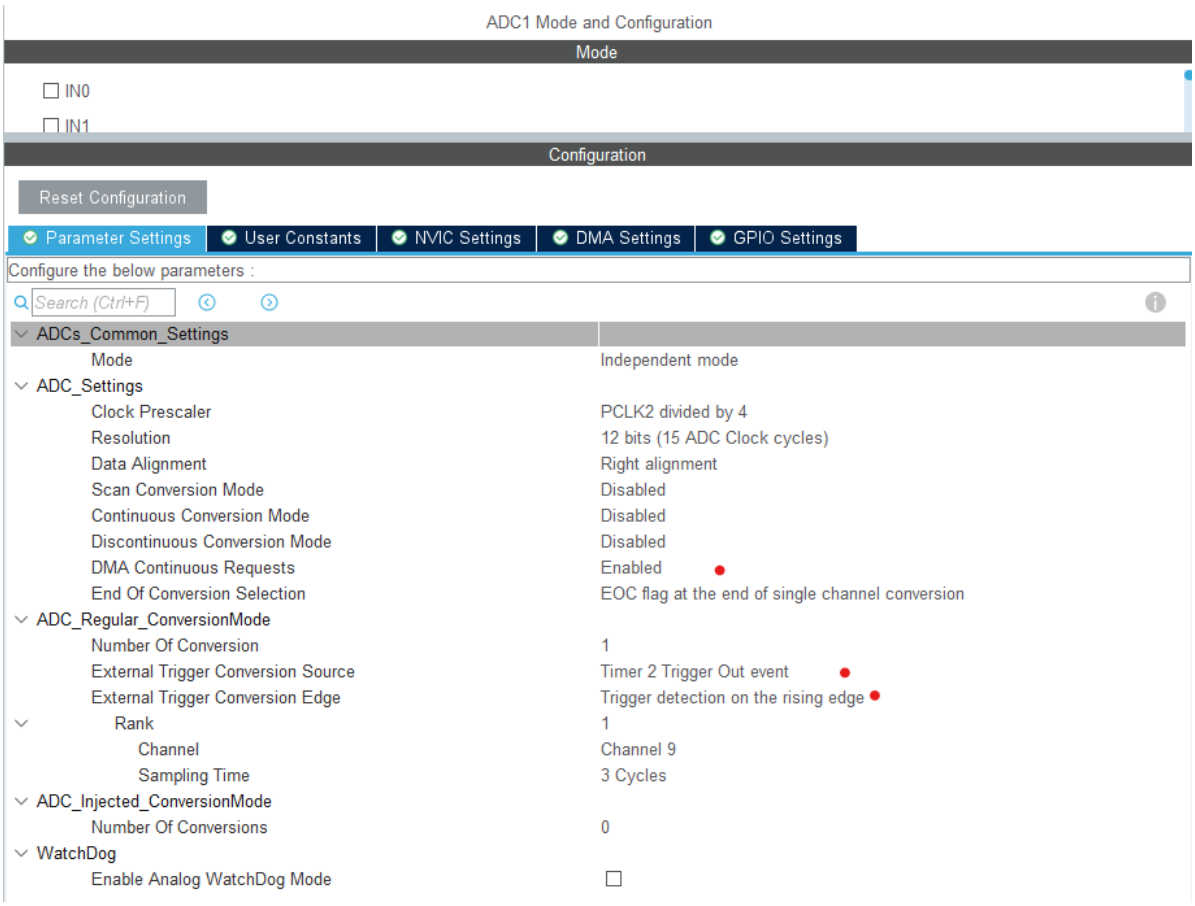
1. Ouvrez le projet STM32Cube et configurez le *Timer 2* avec les paramètres suivants :



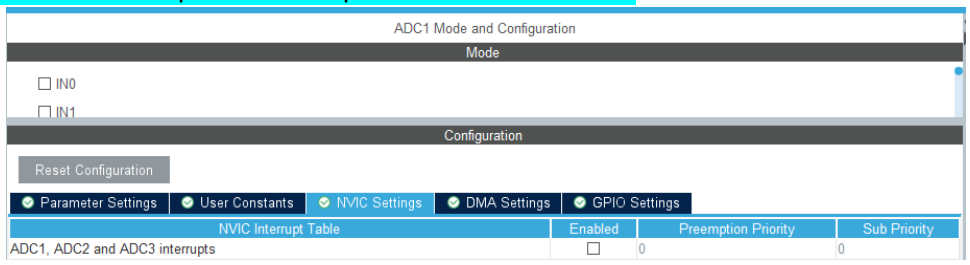
2. De cette manière, le *Timer 2* envoie un événement 40 000x par seconde (84Mz/(83+1)/(24+1)).
3. Désactivez l'interruption liée au *Timer 2* :



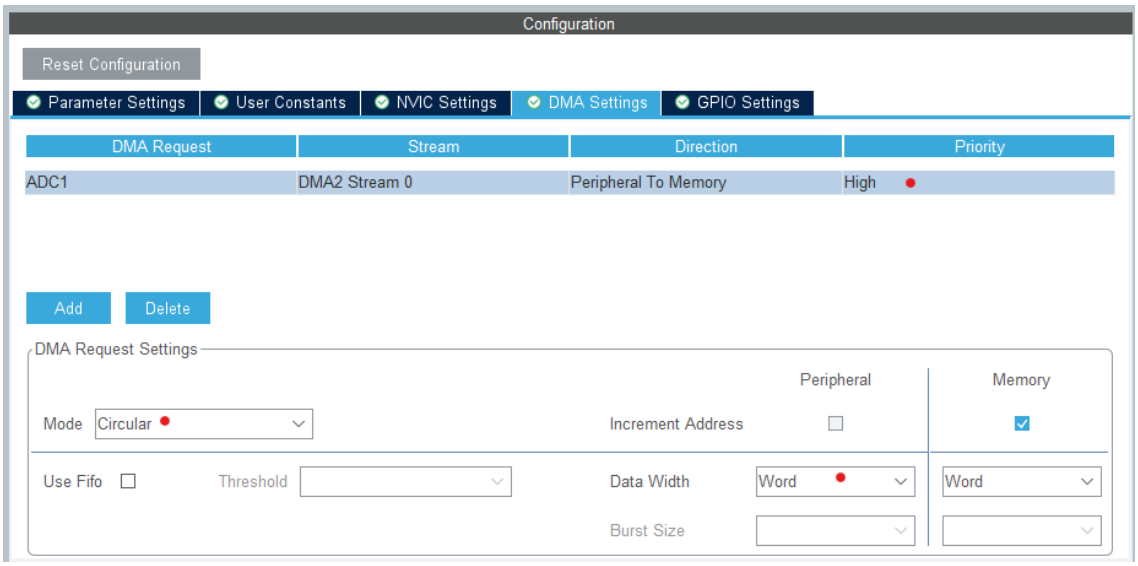
4. Configurez maintenant l'ADC1 comme suit. Vous remarquerez qu'il est maintenant déclenché par le *Timer 2* et que le DMA le met en mode continu. NB: pour activer le "DMA Continuous Requests", il faut ajouter une configuration pour le transfert DMA de l'ADC1 (cf point 6).



5. Assurez-vous que les interruptions sont désactivées :



6. Ajoutez une configuration pour le transfert DMA de l'ADC1 :



7. Notez qu'on utilise un tampon circulaire et que la taille des données est le mot.
8. Générez le code et ouvrez Keil.
9. Ajoutez la fonction qui sera appelée en fin de transfert DMA :

```
 /* USER CODE BEGIN 4 */
/**
 * @brief Retargets the C library printf function to the USART.
 * @param None
 * @retval None
 */
 PUTCHAR_PROTOTYPE
{
 /* Place your implementation of fputc here */
 /* e.g. write a character to the USART2 and Loop until the end of transmission */
 HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);
return ch;
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
  if (hadc->Instance == ADC1) {
    flag_done = 1;
  }
}
 /* USER CODE END 4 */
```

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
      if (hadc->Instance == ADC1) {
            flag_done = 1;
      }
}
```

10. Définissez les variables suivantes :

```
/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
ili9341_t *_screen;

#define TABLE_LENGTH 10000
uint32_t tab_value[TABLE_LENGTH];
volatile int flag_done = 0;

/* USER CODE END PV */
```

```
#define TABLE_LENGTH 10000
uint32_t tab_value[TABLE_LENGTH];
volatile int flag_done = 0;
```

11. Démarrez le *Timer 2* et l'ADC1 pour remplir automatiquement le tableau avec TABLE_LENGTH mots :

```
147
148   ili9341_fill_screen(_screen, ILI9341_BLACK);
149   ili9341_text_attr_t text_attr = {&ili9341_font_11x18,ILI9341_WHITE, ILI9341_BLACK,0,0};
150   char text[] = {"! Hello !"};
151   ili9341_draw_string(_screen, text_attr,text);
152
153   HAL_TIM_Base_Start_IT(&htim2);
154   HAL_ADC_Start_DMA(&hadc1, tab_value, TABLE_LENGTH);
155
156   /* USER CODE END 2 */
157
158   /* Infinite loop */
159   /* USER CODE BEGIN WHILE */
```

```
HAL_TIM_Base_Start(&htim2);
HAL_ADC_Start_DMA(&hadc1, tab_value, TABLE_LENGTH);
```

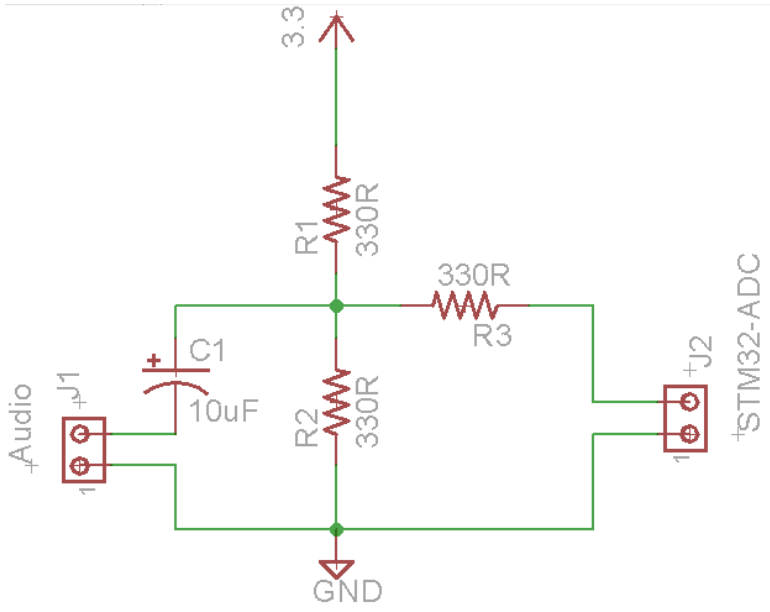
12. À partir de maintenant, le tableau va se remplir tout seul avec les échantillons. Et une fois qu'il sera rempli, le drapeau flag_done deviendra vrai. On peut donc écrire une routine qui traite tous les éléments du tableau. Il ne faut pas perdre de vue que l'échantillonnage continue en parallèle et vient ré-écrire sur les données du tableau
13. La routine suivante calcule diverses valeurs (à vous de les comprendre) :

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
static int count = 0;
while (flag_done == 0) {};       // Wait for DMA
flag_done = 0;                   // Reset flag_done
HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_5);
int max_value;
int min_value;
float rms = 0;
float mean = 0;
min_value = max_value = tab_value[0];
for (int i=1; i<TABLE_LENGTH; i++) {
if (tab_value[i]>max_value) max_value = tab_value[i];
if (tab_value[i]<min_value) min_value = tab_value[i];
mean+=tab_value[i];
}
mean/=TABLE_LENGTH;
for (int i=0; i<TABLE_LENGTH; i++) {
float ac_value = tab_value[i]-mean;
rms+=ac_value * ac_value;
}
rms/=TABLE_LENGTH;
int db_value = 10*log(rms);
```

```
ili9341_fill_screen(_screen, ILI9341_BLACK);
char buf[80];
sprintf(buf,"Min:%i\r\n",min_value);
ili9341_text_attr_t text_attr = {&ili9341_font_11x18,ILI9341_WHITE,
ILI9341_BLACK,40,0};
ili9341_draw_string(_screen, text_attr, buf);
sprintf(buf,"Max:%i\r\n",max_value);
ili9341_text_attr_t text_attr2 = {&ili9341_font_11x18,ILI9341_WHITE,
ILI9341_BLACK,80,0};
ili9341_draw_string(_screen, text_attr2, buf);
sprintf(buf,"dB:%i\r\n",db_value);
ili9341_text_attr_t text_attr3 = {&ili9341_font_11x18,ILI9341_WHITE,
ILI9341_BLACK,120,0};
ili9341_draw_string(_screen, text_attr3, buf);
printf("Done : %i\r\n",count);
count++;
  }
  /* USER CODE END 3 */
```

14. Compilez et exécutez le programme en mode debug. Vous devriez voir la del LD2 changer d'état à chaque fois que tableau est rempli. De plus, les mesures du signal échantillonné s'affichent sur le LCD. Comme présentement il n'y a rien à l'entrée, on mesure juste du bruit.
15. Implantez le circuit suivant sur votre breadboard pour faire l'acquisition d'un signal audio :

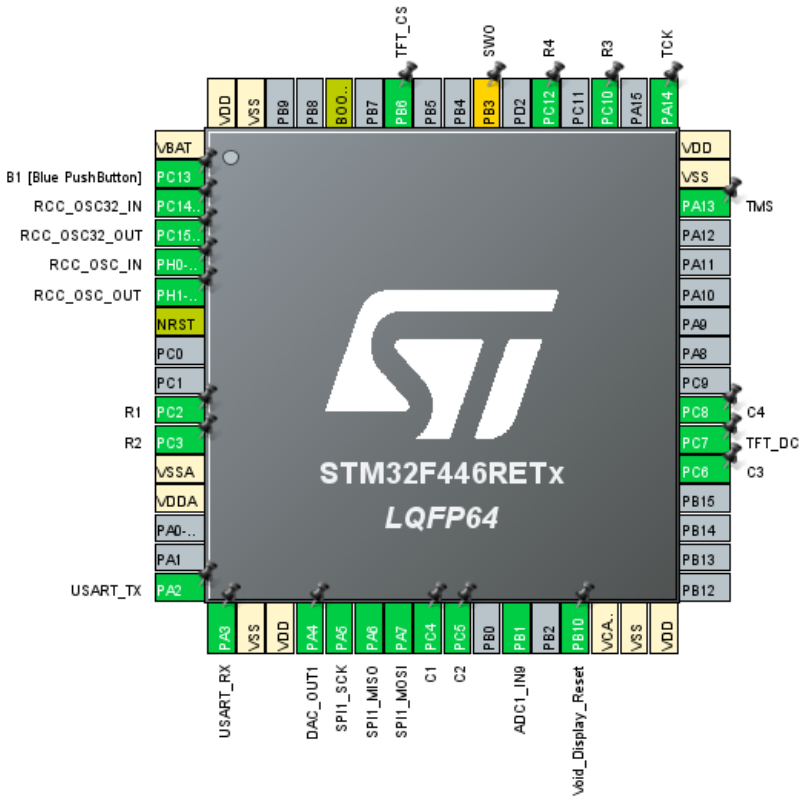


16. Connectez maintenant une source audio (ex. tél cellulaire) à votre entrée audio et regardez l'influence du volume sur les valeurs affichées à l'écran.

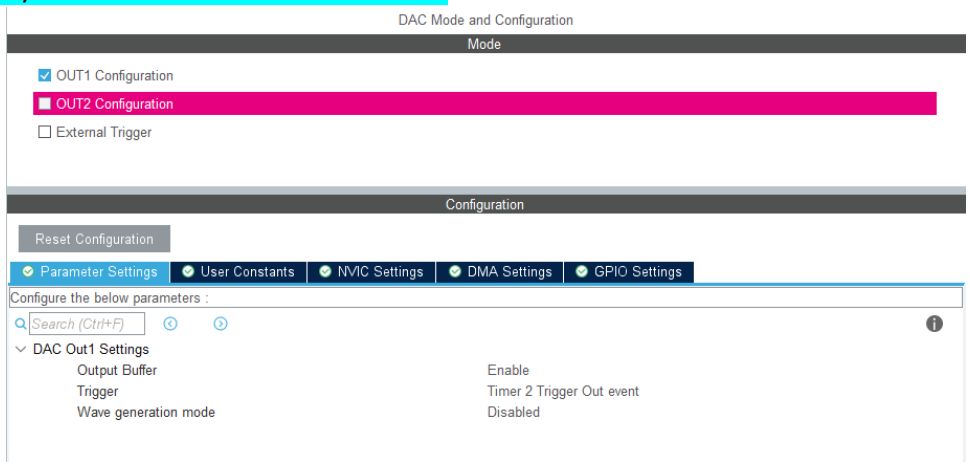
Expérience 2

Nous allons maintenant utiliser le même principe pour réaliser une conversion digital-analogique.

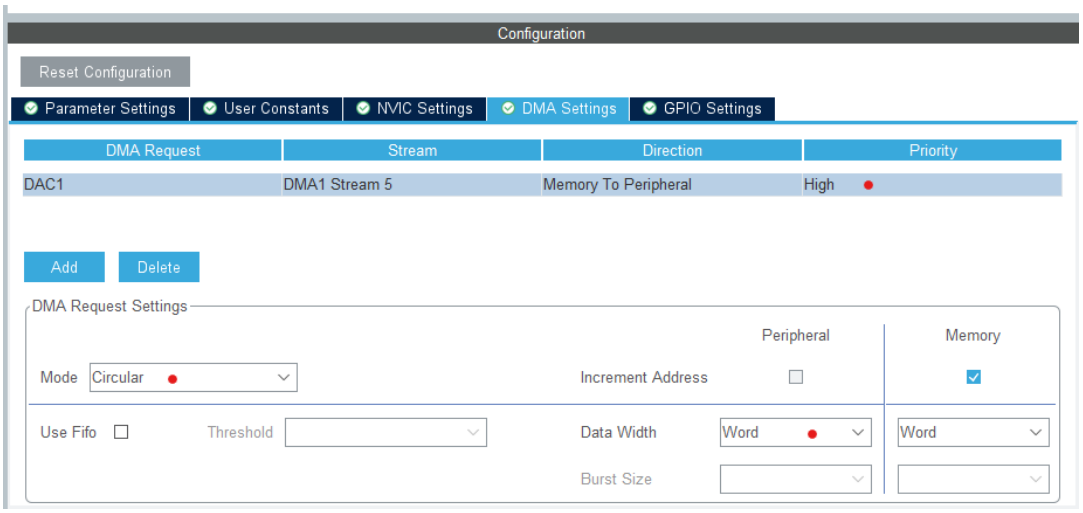
1. Ouvrez STM32Cube et activez la sortie 1 du DAC sur la broche PA4 :



2. Synchronisez le DAC avec le Timer 2 :



3. Configurez le DMA :



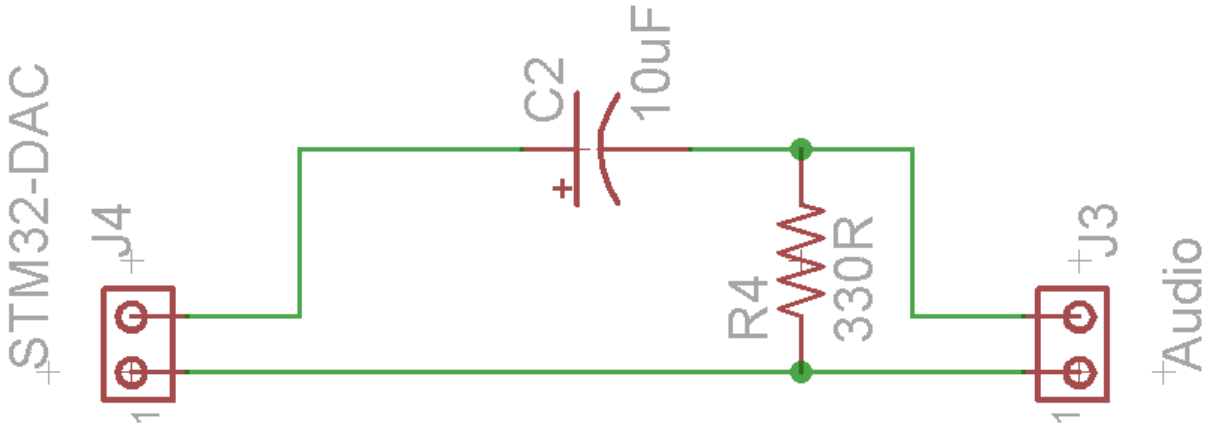
4. Générez le code et ouvrez Keil.
5. Initialisez le tableau de données à envoyer et démarrez le Timer2 et le DMA :

```
149
150   ili9341_fill_screen(_screen, ILI9341_BLACK);
151   ili9341_text_attr_t text_attr = {&ili9341_font_11x18,ILI9341_WHITE, ILI9341_BLACK,0,0};
152   char text[] = {"! Hello !"};
153   ili9341_draw_string(_screen, text_attr,text);
154
155   for (int i=0; i<TABLE_LENGTH; i++){
156     float value = (i %100)*(4097.00/100);
157     tab_value[i] = (int) value;
158   }
159
160   HAL_TIM_Base_Start_IT(&htim2);
161   //HAL_ADC_Start_DMA(&hadc1, tab_value, TABLE_LENGTH);
162   HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_1, tab_value, TABLE_LENGTH, DAC_ALIGN_12B_R);
163
164   /* USER CODE END 2 */
165
```

```
for (int i=0;i<TABLE_LENGTH;i++) {
      float value = (i%100)*(4097.0/100);
      tab_value[i]=(int) value;
}

HAL_TIM_Base_Start(&htim2);
HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_1, tab_value,
TABLE_LENGTH, DAC_ALIGN_12B_R);
```

6. Les données du tableau `tab_value` sont maintenant envoyées les unes après les autres au DAC à chaque impulsion du *Timer 2*, soit 40 000 fois par secondes. Vous pouvez visualiser la sortie PA5 sur un oscilloscope. Vous pouvez aussi l'entendre dans un casque écouteur (**attention au volume !**). Pour cela utilisez le schéma suivant :



7. Pour envoyer un nouveau tableau à chaque fois que le précédant est envoyé, vous pouvez activer le DMA en mode normal plutôt que le mode circulaire (dans STM32Cube). Vous devez alors ajouter la fonction de callback suivante pour relancer un nouveau transfert DMA :

```
void HAL_DACEx_ConvCpltCallbackCh1(DAC_HandleTypeDef* hdac) {
   HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_1, tab_value, TABLE_LENGTH,
DAC_ALIGN_12B_R);
}
```

Il vous appartient alors de changer le tableau `tab_value` que vous envoyez à chaque appel. Typiquement, on utilise un système à deux tableaux. Sur le temps qu'on envoie un tableau avec le DMA, on remplit le second. Ensuite, on inverse le rôle de chacun des tableaux. Un exemple de projet à deux tableaux est disponible sur le site du cours. Ce sera votre point de départ pour la partie en laboratoire.

Assurez-vous de bien comprendre le code en général.
Vous serez évalués en début de laboratoire.

Genre de question possible : quelle est la fréquence du son produit ?

2ème partie : Laboratoire à réaliser en salle à Polytechnique

1) Réalisez une sirène (un signal carré périodique) dont la fréquence va lentement passer de XX Hz à YY Hz et revenir ensuite jusque XX Hz avec une progression géométrique. À toutes les ZZ millisecondes, votre fréquence doit être multipliée par une constante. Le temps total d'un aller-retour sera de 12 secondes. XX, YY et ZZ vous seront donnés par votre chargé de laboratoire.

   Par exemple, si XX=400Hz, YY=1600Hz et ZZ=1000 millisecondes (une seconde), on multipliera la fréquence par $(YY/XX=4)^{1/6}$ *(=1.2599…) à chaque seconde. De cette manière, après 6 secondes on aura multiplié la fréquence initiale par 4. Ensuite, on divisera la fréquence par* $4^{1/6}$ *à chaque seconde pour revenir à la fréquence initiale après 12 secondes. Et la boucle recommence.*

Remarque importante : votre fréquence d'échantillonnage est 40kHz et ne peut en aucun cas être modifiée au cours du temps.

2) Ajoutez un effet de vibrato en modulant l'amplitude du son à une fréquence de 2Hz. L'amplitude de votre son devrait se comporter ainsi :

   A = 1+k*sin(4pi*t), avec k : un paramètre entre 0 et 1, qui définit l'indice de la modulation

   Vous trouverez plus d'information sur la modulation d'amplitude au lien suivant : https://fr.wikipedia.org/wiki/Modulation_d%27amplitude

Suggestion : utilisez un tableau de 1000 éléments pour encoder les valeurs du sinus sur une période. À chaque 20000 échantillons (0.5 seconde), vous aurez balayé la table au complet.

!!! Il vous est grandement conseillé de préparer cette partie AVANT le laboratoire !!!