Laboratoire d'actionneurs intégrés (LAI)
Commande embarquée de moteurs

Laboratoire 1 : Prise en main du Microcontrôleur, PWM, ADC

Objectifs : - maîtrise de la fonctionnalité du timer
- utilisation d'un port gpio
- compréhension et génération d'un PWM
- lecture d'une valeur analogique

Etapes :
1) Démarrage :
- Démarrer STM32CubeIDE et créer un nouveau projet STM32 et choisir la carte « NUCLEO-F401RE » (de type nucleo 64) dans la rubrique « board selector ». ATTENTION, ne PAS cliquer sur le nom du microcontrôleur en bleu sinon le menu d'initialisation des périphériques n'apparaît pas par la suite.
- Choisir un nom de projet. Les autres options à sélectionner sont langage *C*, *executable* et type *STM32Cube* (puis répondre « Yes» à la question sur l'initialisation des périphériques)
- Le logiciel de développement vous présente une interface graphique (cubeMX) qui vous permet de configurer les périphériques en générant automatiquement le code.
- Générer le code (Alt-K), compiler le code (Ctrl-B) et essayer le débugger (F11).

2) Allumer une LED :
- la LED se trouve sur le port PA5, elle est déjà configurée dans le modèle de la carte nucleo donc il suffit de l'allumer et de l'éteindre dans le code principal.
- La fonction qui « toggle » la led est HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin)
  Les constantes LD2_GPIO_Port et LD2_Pin sont définies automatiquement par cubeMX dans le fichier main.h (User Gpio Label)
- Appeler cette dernière fonction dans la boucle infinie de main() et insérer un « breakpoint » sur l'instruction while et effectuer un débogage pas-à-pas pour voir la led clignoter.

3) Générer une interruption à 20kHz (fréquence PWM usuelle) sur le timer 1:
le compteur doit faire une montée et descente « symétrique » 20'000 fois par seconde et générer les conversions AD à la période :

Dans l'interface graphique, à la rubrique Timers :
- Configurer le périphériques « TIM1 » pour qu'il utilise l'horloge interne du microcontrôleur (dans « clock source »). Configurer les 2 premiers « channels » en « PWM Generation CHX CHXN », ils nous serons utiles pour faire tourner un moteur et le 3ème en « output compare no output » ce qui nous servira à démarrer la conversion AD.
- Déplacer les signaux CH1N et CH2N sur les pattes PB13 et PB14
- Le « counter mode » du TIM1 doit être « center aligned mode 1 ». La période est à calculer en sachant que la fréquence d'horloge du TIM1 est de 84MHz, qu'on cherche à avoir 20kHz et qu'on compte et décompte. On peut déclarer une constante PERIPWM dans l'onglet « user constants » et l'utiliser directement pour spécifier la période dans « counter period » afin d'avoir de la flexibilité par la suite.
- L'entrée Pulse de chaque PWM/Compare correspond au niveau de compare qui conduit à une commutation du signal de sortie. Vérifier qu'elle est bien à 0 pour le channel 3 afin de provoquer une conversion AD à l'underflow. Il faut aussi configurer le mode du channel 3 à « toggle on match »

A la rubrique Analog:
- L'ADC1 doit être configurer pour utiliser les entrées IN0,IN1 et IN7 **à** IN13 (à cocher).
- Ajouter une requête DMA de mode **circulaire** dans l'onglet DMA de l'ADC1 (important)
- Dans l'ADC1, effectuer 1 conversion AD en mode « scan », avec DMA en continu, EOC à la fin de toutes les conversions.

- La conversion doit être déclenchée (trigged) de manière externe «on both rising and falling edge » et sa source est le compare 3 du timer 1. La conversion s'effectue sur la patte 9 (Rank 1 => channel 9).

Générer le code.

- Déclarer un tableau de résultats de manière globale :
  uint16_t resultats[1] ; //doit absolument être un uint16_t
- Démarrer le Channel 3 du timer 1 au moyen de la fonction HAL_TIM_OC_Start, il s'agit de lui passer comme arguments un pointeur sur la variable globale qui contient sa configuration (htim1 dans ce cas), ainsi qu'un entier dont la valeur est TIM_CHANNEL_3.
- Démarrer l'ADC1 au moyen de la fonction HAL_ADC_Start_DMA et lui passer la variable de configuration de l'adc1 (à vous de la trouver…), le tableau de résultats créé et la taille du dit tableau (1 seule conversion AD).
- ajouter la routine de « call back »
  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* AdcHandle) à la fin du fichier main.c
  Si elle existe, cette routine est appelée automatiquement par le code généré par CubeMX.
- Modifier cette routine pour qu'elle allume/éteigne la LED tous les 20000 fois qu'elle est appelée avec une variable (de manière à avoir une led qui s'allume et s'éteigne une fois par seconde). Ne pas oublier d'enlever l'allumage/extinction de la led dans la boucle main

4) Générer un signal PWM sur la patte PA8 :

Générer un PWM :

- Démarrer la gestion du PWM sur le channel 1 du timer 1 à l'aide de HAL_TIM_PWM_Start (même genre d'arguments)

Dans l'interruption :

- Changer le taux de PWM (registre TIM1->CCR1) à partir de la valeur mesurée sur le potentiomètre (qui se trouve dans le tableau de résultats déclaré précédemment). D'autre part, il faut faire une petite conversion entre la valeur mesurée (0->4096) et la valeur du timer (0-période)

Observer le PWM à l'oscilloscope.

6) On peut ensuite générer un PWM complémentaire sur le « Channel 1N » Au moyen de l'instruction HAL_TIMEx_PWMN_Start(&htim1,TIM_CHANNEL_1);
Et ajouter ensuite un deadtime de 0.5us dans cubeMX (configuration timer 1, automatic output state et choisir un deadtime en cycles d'horloge) et mesurer les deux signaux 1 et 1N à l'oscilloscope.

Version 1.7