ISEN ALL IS DIGITAL! BREST | yncréa

TP électronique

Microcontrôleur STM32L152

Timers

Jean-Jacques Meneu

Chapitre 1. Timer avec Interruption et Horloge Interne

Nous allons utiliser le Timer3 comme une base de temps. A la fin de la base de temps, une interruption est déclenchée.
Créer un nouveau projet avec CubeMX pour le board NUCLEO-152RE

Tous les périphériques ont une horloge fournie par un bus interne au microntrôleur afin de fonctionner convenablement.
Question : Grâce à la datasheet, déterminer le bus qui est connecté au timer 3

Dans le projet de de STM32CubeIDE, avec la configuration graphique, aller dans le fenêtre Clock Configuration et déterminer la fréquence du bus qui fournit Timer3.

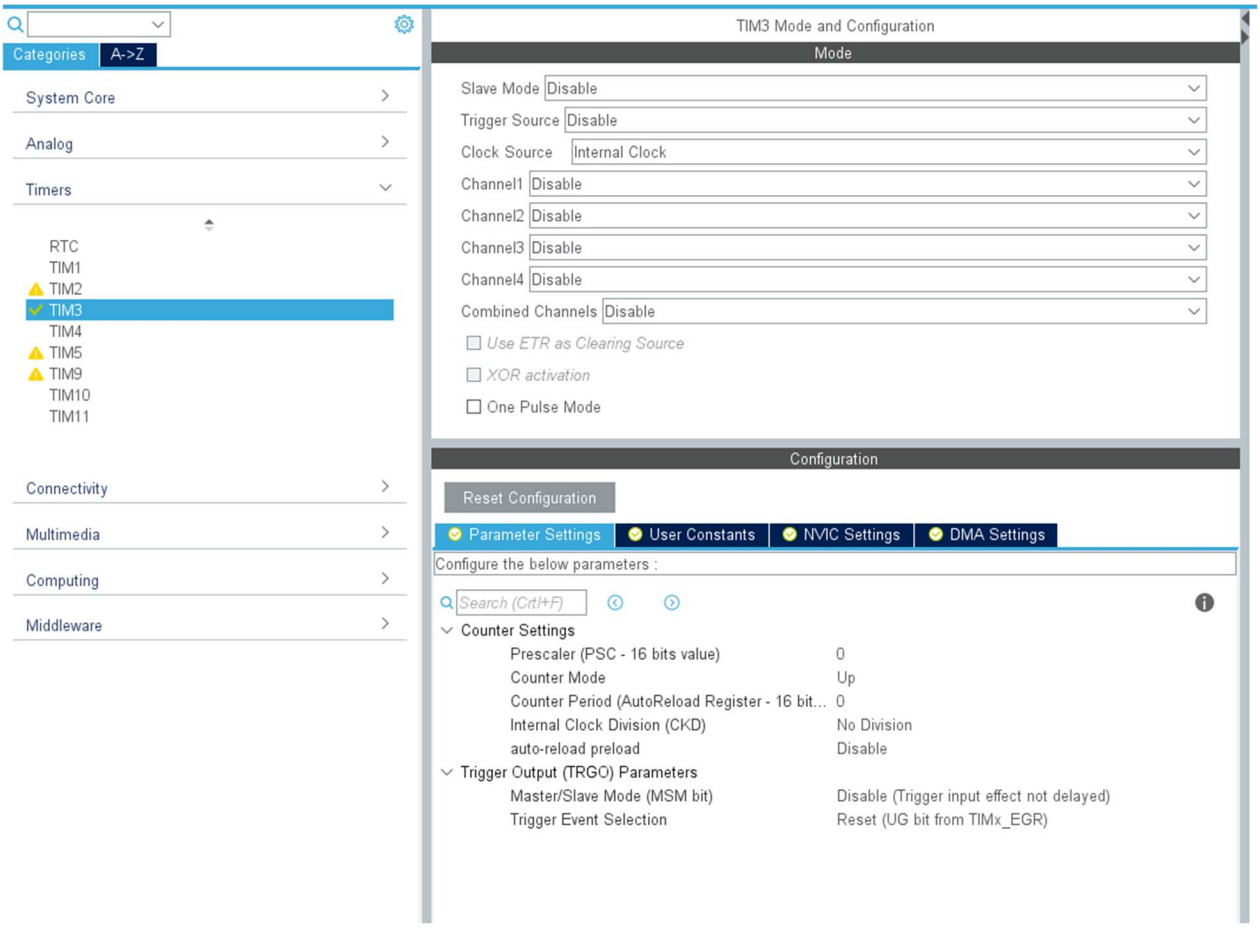
Nous conservons cette valeur par défaut pour nos calculs.

Nous désirons obtenir une interruption toutes les secondes. TIM3 est un 'General Purpose Timer' et qui contient les traditionnels prescalaire, Autoreload et counter registers.

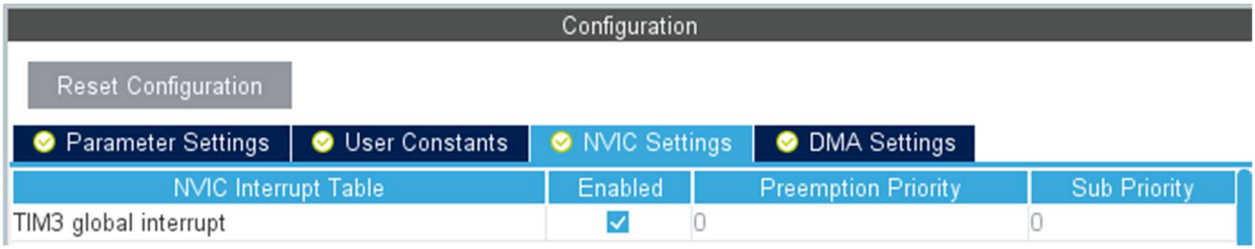
Question : Donner les noms de ces 3 registres. Déterminer leur adresse et les valeurs qui peuvent être programmées.

STM32CubeIDE permet de rentrer les configurations grâce à une interface homme-machine plutôt que de le programmer de manière fastidueuse en C. STM32CubeIDE génère le code de notre configuration mais il est très important de comprendre le code généré. Nous l'étudierons plus tard dans ce TP.

Aller dans la configuration du Timer3 et entrer les valeurs calculées précédemment



Puisque nous voulons générer une interruption toutes les secondes, dans le NVIC setting, il faut cocher la case Enabled du TIM3 global interrupt.



Vérifier que la pin qui allume ou éteint la diode est configurée correctement avant de générer le code.

Lorsque l'interruption est générée, la fonction TIM3_IRQHandler est appelée

```
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */

  /* USER CODE END TIM3_IRQn 0 */
  HAL_TIM_IRQHandler(&htim3);
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}
```

Question : Rappeler la provenance du nom de fonction d'interruption? As t'on le droit de le changer ? Où est-elle définie ?

Question : Expliquer ce que représente la variable htm3. Faire le lien avec la question sur les adresses. Faire le lien avec les adresses des registres PSC et ARR pour vérifier comment l'écriture dans ces registres s'effectue.

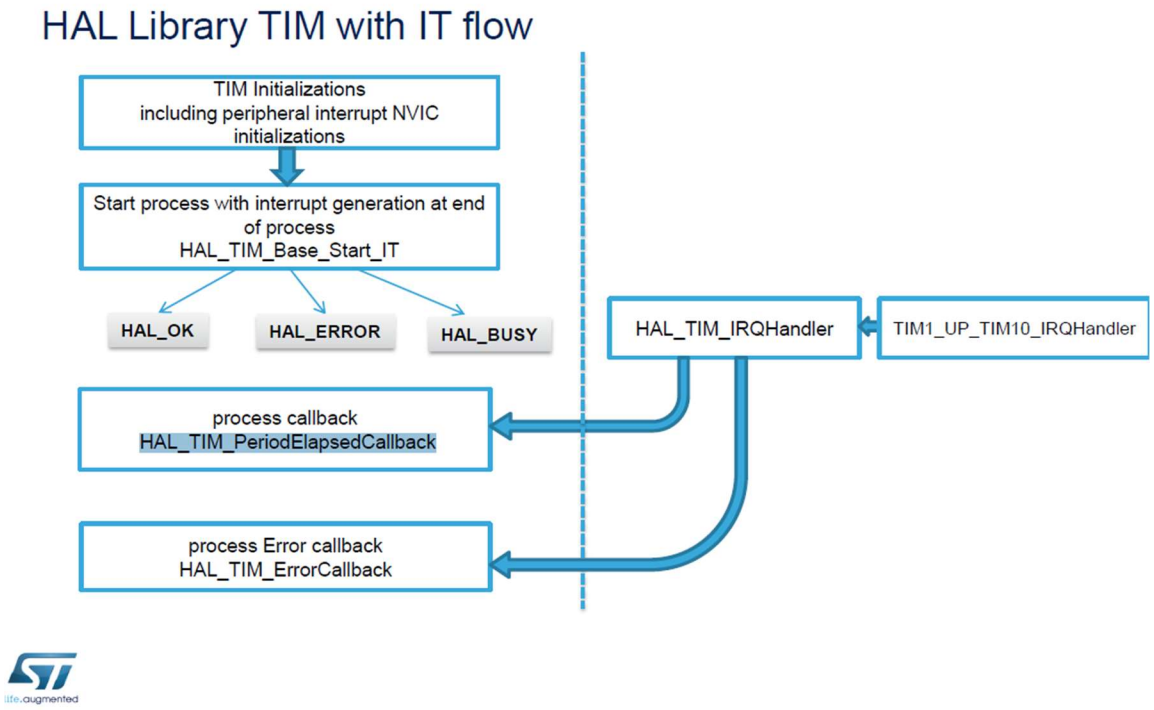
Une fonction __weak signifie que la fonction peut être réécrite et que la nouvelle fonction sera choisie par rapport à la fonction déclarée weak pendant la compilation.

```
/**
  * @brief  Period elapsed callback in non-blocking mode
  * @param  htim TIM handle
  * @retval None
  */
__weak void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);

  /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_TIM_PeriodElapsedCallback could be implemented in the user file
   */
}
```

HAL signifie Hardware Abstract Layer et aide à programmer le microcontrôleur sans connaître en détails l'architecture de ce même microcontrôleur. Ci-dessous, la logique des fonctions HAL pour l'interruption du timer.

3.2.1 Use TIM with interrupt 266



La plupart des fournisseurs fournissent des librairies équivalentes et vous permettra de programmer très rapidement une nouvelle plateforme.

Toutes les fonctions (API : Application Programming Interface) sont détaillées dans le User Manual UM1816 : Description of STM32L1 HAL and Low-Layers drivers.

P428, la fonction HAL_TIM_PeriodElapsedCallback est expliquée

HAL_TIM_PeriodElapsedCallback

Function name	**void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)**
Function description	Period elapsed callback in non blocking mode.
Parameters	• **htim:** : TIM handle
Return values	• **None:**

De même, p525, la fonction HAL_TIM_IRQHandler est présentée

HAL_TIM_IRQHandler

Function name	**void HAL_TIM_IRQHandler (TIM_HandleTypeDef * htim)**
Function description	This function handles TIM interrupts requests.
Parameters	• **htim:** TIM handle
Return values	• **None:**

Dans le main.c, écrire la fonction HAL_TIM_PeriodElapsedCallback et écrire le code avec une fonction de la librairie HAL pour faire changer d'état la LED.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
  /* A compléter avec une fonction de la librarie HAL qui
  permet de basculer la LED branchée sur PA5*/
}
```

Expliquer comment cette fonction est programmée.

Initialisation du Timer

La fonction MX_TIM3_Init(); permet d'initialiser les valeurs du timer. Vérifier que les valeurs entrées lors de la configuration avec l'interface graphique du STM32CubeIDE se retrouvent dans cette fonction.

Pour fonctionner, nous avons encore besoin d'autorisation les interruptions et de démarrer le timer. De nouveau, une fonction de la librairie HAL permet de HAL_TIM_Base_Start_IT.

Question : Retrouver cette fonction dans le AN1816 et déterminer la paramètre qui doit être transmis

Mettre un point d'arrêt devant la fonction HAL_TIM_Base_Start_IT et avancer pas à pas.
Ouvrir la fenêtre de debug TIM3 et repérer les registres qui changent en allant pas à pas dans la fonction.
Relever les adresses de ces registres. Donner la fonction des bits qui sont modifiés

Par exemple, le registre CR1 bit 0 permet de démarrer le timer

Question : Mettre la ligne HAL_TIM_Base_Start_IT en commentaire et écrire directement dans les registres

Attention ! Pour qu'une interruption fonctionne, il faut également que le NVIC soit configurée. Ceci est fait dans la fonction MX_TIM3_Init lors de l'appel de la fonction HAL_TIM_Base_Init(&htim3)

```
if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
{
  Error_Handler();
}
```

De même pour le bit 0 de SR

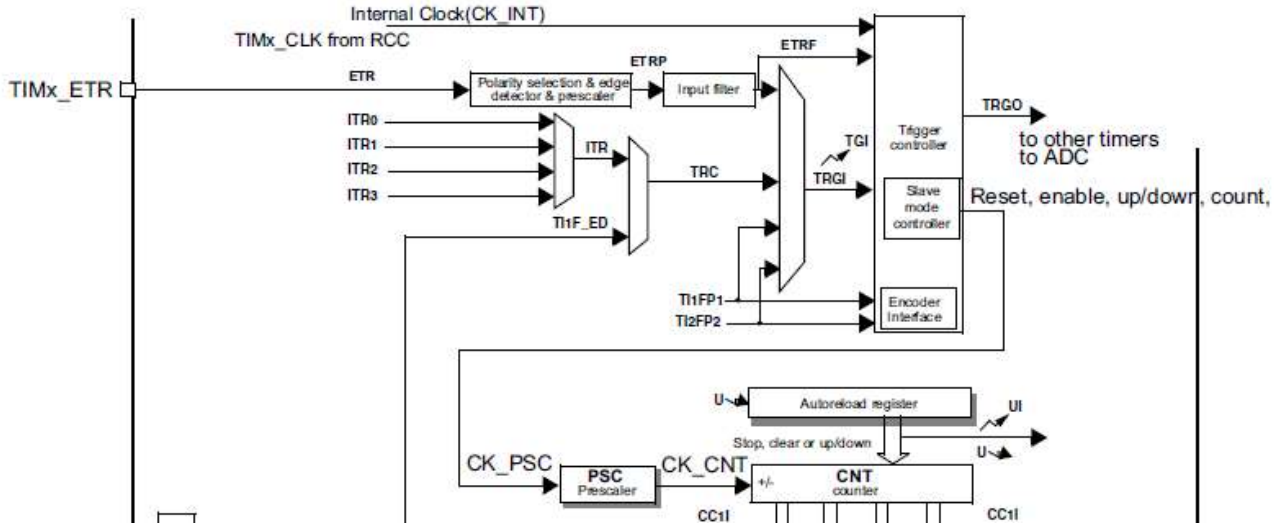
Question : Quel numéro d'interruption est lu dans le fichier xPSR.

Chapitre 2. Timer avec Interruption et Horloge Externe

Le but de cet exercice est de changer le statut de la diode après avoir appuyé 5 fois sur le bouton bleue de la carte Nucleo.

Question : Repérer le numéro de la broche du microcontrôleur qui est connectée au bouton bleu. Quelles sont les fonctions supportées par cette broche ?

La figure ci-dessous présente le schéma de l'horloge pour un timer. Dans le premier chapitre, nous avons utilisé l'horloge interne, c'est-à-dire le signal Internal_Clock(CK_INT). Pour que le bouton nous serve d'horloge, nous devons connecter le bouton bleu à une pin TIMx_ETR d'un timer.



Question : la broche connectée au bouton B1 peut-elle être utilisée comme fonction TIMx_ETR ?

Tous les timers ne supportent pas le fonction TIMx_ETR.

Question : Faire la liste des timers du STM32L152RE et préciser lesquels supportent la fonction TIMx_ETR et préciser le numéro de broche correspondant.

Utiliser le Timer2 ou le Timer3 par la suite. Grâce à CubeMX, initialiser les broches.

Question :
Configurer le timer pour basculer la LED après 5 appuis sur le bouton bleu. L'information est dans le Reference Manual dans la section des timers. Suivez les indications données dans le Reference manual.
La configuration de la pin a-t-elle changé ? En particulier, quelle est la configuration des registres MODER et AFR ?
Connecter le hardware du Nucleo et écrire le programme.

Pour le programme, trouver la fonction HAL qui permet de démarrer le timer et générer une interruption.

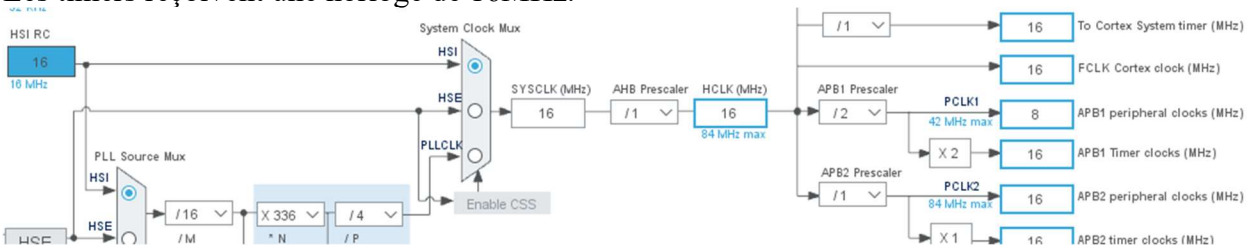
Question : en regardant la configuration du Timer entre une horloge interne et une horloge externe, déterminer le(s) registres et bits à programmer. Retrouver les explications dans le Reference Manual

Chapitre 3 Timer PWM

Question : La LED est sur la broche PA5. Vérifier les fonctions disponibles sur cette broche. A-t-on un canal de timer connecté à cette broche ?

Avec votre choix de timer, configurer le channel du timer pour clignoter à 2 Hz avec un ratio de 50%.

La PLL n'est pas nécessaire. Configurer les horloges pour être connectées au HSI directement. Les timers reçoivent une horloge de 16MHz.



Générer le code. Comme précédemment, il faut démarrer le timer. Nous voulons générer un signal PWM.

Question :
Trouver une fonction de la librairie qui permet de démarrer le timer comme PWM.
Ecrire le programme et tester.

Aller pas à pas dans la fonction et décrire les changements de registres et les bits qui sont modifiés.

Mettre cette ligne en commentaire et modifier directement les registres.

Chapitre 4 Timer Input Capture

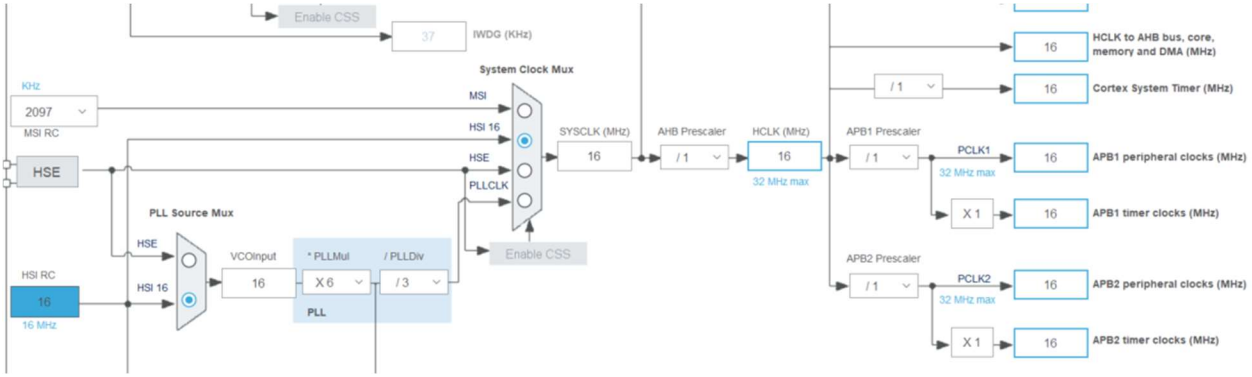
L'input capture est une fonction du timer qui permet de copier la valeur compteur dans un registre sur un événement donner. Factuellement, la valeur du registre CNT sera copiée dans le registre CCR.

Nous allons nous servir de cette fonction pour mesurer le temps entre deux impulsions sur le bouton poussoir B1.

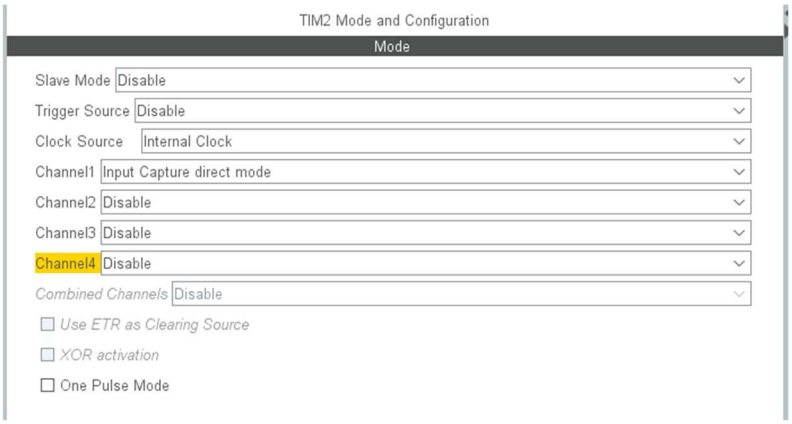
Dans l'exercice précédent sur le PWM, nous avons utilisé le canal (channel en anglais) en sortie car nous devions allumer et éteindre une LED. Pour cet exercice, nous devons donc prendre un timer qui accepte que le canal soit configuré en entrée.

Question ;

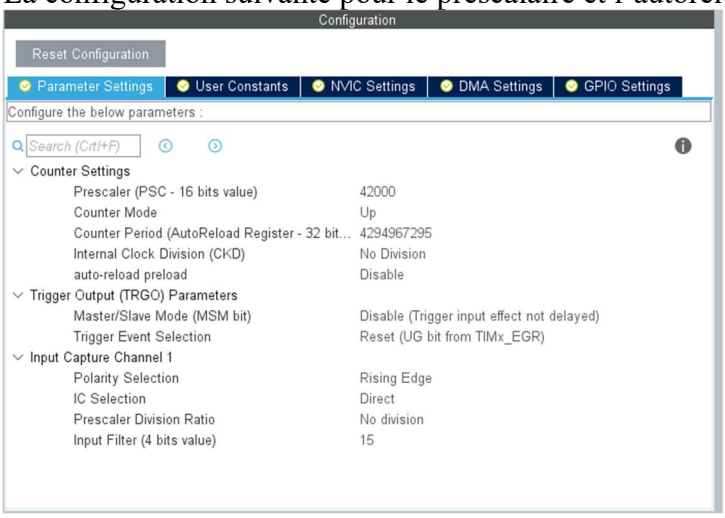
Vérifier les fonctionnalités disponibles sur la broche PC13 connectée au bouton bleu. Est-il possible de connecter le canal d'un timer en entrée?
Etudier la timer2, et déterminer la broche qui est connectée au channel 1 du timer2
Avec cette configuration des horloges, quelle est la fréquence d'entrée du Timer2



Si nous utilisons un préscalaire égal à 15 999, quelle est la résolution maximale que nous pouvons espérer ?
Quelle est la taille du registre CNT du Timer2. En déduire la durée maximum en ms avant un overflow du compteur. Cette configuration permet-elle de calculer le temps entre 2 appuis sur le bouton ? Justifier la réponse.
Créer un nouveau projet avec CubeMX. Pour le Timer2, utiliser l'horloge interne et le channel1 comme Input Capture Direct Mode



La configuration suivante pour le prescalaire et l'autoreload register .



Prévoir une interruption à chaque appui sur le bouton bleu.

Dans le programme principal, démarrer le timer grâce à la fonction HAL

```
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
```

Question : Vérifier les bits et leurs fonctions après l'exécution de cette fonction (on ne s'occupera pas du registre DMAR)

Connecter sur le board les broches PC13 et PA0

La récupération de l'interruption de l'input Capture se faite grâce à la fonction HAL

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef * htim)
```

Question :
Ecrire cette fonction pour calculer le temps entre 2 appuis.

Tester le programme avec un chronomètre. Indépendamment du code C, en mode debug, pendant que le programme est en pause à un point d'arrêt, le timer continue à tourner. Ainsi, en positionnant convenablement le point d'arrêt, appuyer sur le bouton bleu et relever les valeurs du timer. Faire continuer le programme. Appuyer de nouveau sur le bouton bleu. Par soustraction

des 2 valeurs contenues dans le registre CCR, vous pouvez calculer le temps entre les deux pressions.

Chapitre 5 Timer Watchdog

Le watchdog timer est très différent des autres compteurs vus précédemment. Le watchdog timer sert à vérifier que l'application est toujours sous contrôle. Par analogie avec un conducteur de train. Ce dernier doit appuyer régulièrement sur une pédale afin de montrer qu'il continue à conduire le train. Si il n'appuie plus sur la pédale, le train s'arrêtera.
Un watchdog timer doit être mis à jour régulièrement. Si ce n'est pas le cas, après un temps fixé par la configuration du watchdog timer, ce dernier va faire une reset de l'application.

Ce type de timer est simple et nécessite peu de registres ;

Question :

- Lister les registres de l'independant Watchdog et leurs fonctions
- Quelle est la fréquence d'horloge fournie à l'IWDG
- Calculer le temps du watchdog avec les valeurs par défaut
- Configurer le watchdog pour obtenir un temps de 4 secondes

Créer un nouveau projet en configurant IWDG avec un temps de 4s.
Le watchdog a besoin d'être lancé. Ceci ne fonctionne pas en changeant un bit (bit0 : CEN : Counter Enable) comme dans un timer classique.

Question :

Dans le programme, mettre le point d'arrêt devant HAL_Init. Lancer le programme en mode debug et vérifier ce qui se passe.

```
79    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
80    HAL_Init();
```

Le watchdog ne se gère pas comme un timer classique.
Question :

Lire Page 551 du reference manual la gestion des registres su watchdog. Vérifier que cette procédure est utilisée dans la fonction MX_IWDG_Init();

Dans la boucle infinie du programme principal, écrire le programme suivant :

- attente de 2 secondes
- Rafraichir le watchdog
- Basculer la diode

Vérifier que le programme ne s'arrête pas et ne repasse jamais au point d'arrêt HAL_Init().

Modifier le délai du programme précédent pour le passer à 6 secondes.
Relancer le programme. Expliquer le nouveau comportement du programme.

Donner des suggestions pour obtenir un clignotement de la LED toutes les 6 secondes tout en conservant un watchdog de 4 secondes.