STMicroelectronics

Info2 : STM32
TP4 Les timers

Ce qui est écrit dans le programme BUT1

Apprentissage visé : être capable de concevoir un programme organisé en fonctions et implantable dans une cible de type microcontrôleur
Contenu : Timers.

Le but de ce TP4 Info2

Pour l'apprentissage visé : Configurer le timer 7 puis le timer 10 pour faire clignoter 2 leds à des fréquences différentes par interruption.

Matériel utilisé

Kit de développement : STM32F429 discovery : 32-Bit Cortex®-M4F - 180Mhz – 2MB Flash – 3x12 bits 16 channels 2.4MSPS ADC – 2x12 bits DAC – 12 x 16 bits timers and 2x32bits timers - SPI - I²C - 500 µA/MHz
Ecran : QVGA : 320x240 pixels - 262K color TFT - 24-bit RGB Parallel Pixel Output 8 bits-per-pixel (RGB888) – 4 wires resistive touch screen

M. Garcier

Table des matières

A. Les timers 1
1. Programme sans interruption 1
2. Programme avec interruption 1
3. Calcul du temps entre deux interruptions 2
4. Questions à faire sur feuille de brouillon et à faire valider à l'enseignant 3
B. Créer un nouveau projet 3
C. Compilation et programmation du kit de développement STM32 4
D. Faire clignoter la led verte et la rouge avec 2 timers 4
1. Faire clignoter la led verte toutes les 0,5s avec le timer 7 4
2. Faire clignoter la led rouge toutes les 1,234s avec le timer 10 4

A. Les timers

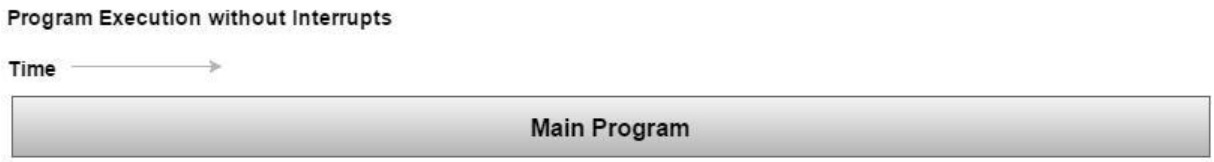
Les timers sont des périphériques qui sont très souvent utilisés dans les microcontrôleurs car ils permettent :

- De générer une temporisation programmable (time base generator).
- Générer des signaux pwm (commande de moteur DC et servomoteurs).
- Compter des événements sur une GPIO.

Dans ce TP nous allons les utiliser pour compter un intervalle de temps. Lorsque ce temps est écoulé une interruption est générée et une fonction appelée routine d'interruption sera est appelée. C'est dans cette routine d'interruption que sont écrites les lignes de codes à exécuter lorsque le temps est écoulé.

1. Programme sans interruption

Pour bien comprendre l'intérêt de la programmation avec interruption que nous allons voir par la suite, regardons dans un premier temps les limites de la programmation classique (c'est-à-dire sans interruption).
Il est possible de représenter l'exécution d'un programme avec la figure suivante. Les lignes de code s'exécutent les unes à la suite des autres et le temps s'écoule peu à peu.

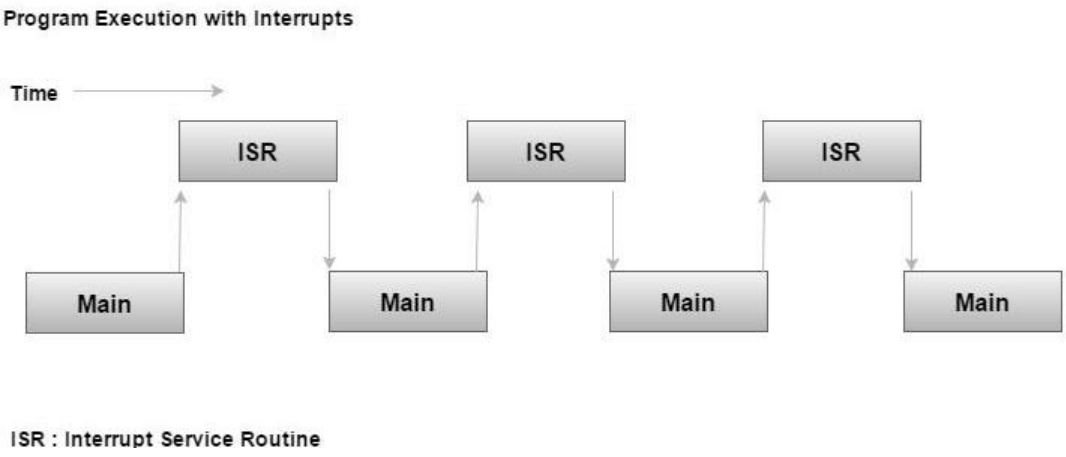


L'inconvénient est qu'il n'est pas possible de faire autre chose que le programme main(), or, dans un système à microcontrôleur il y a des moments où l'on souhaiterait interrompre le programme principal pour faire une autre tâche pendant un court instant.

La limite est donc la suivante : seul le programme main() peut être réalisé. Il n'est pas possible de faire plusieurs tâches.

2. Programme avec interruption

Cela se représente de la façon suivante :



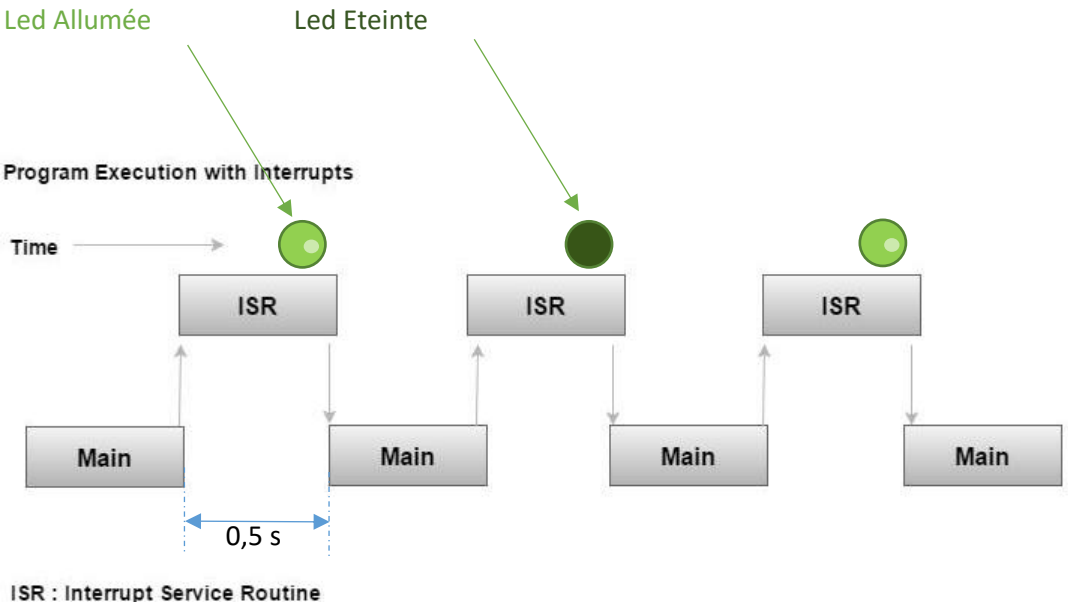
ISR : Interrupt Service Routine

Le code est réalisé en plusieurs parties. Le programme principal main() est toujours présent mais le programmeur ajoute des routines d'interruptions (des mini-tâches) appelé en Anglais ISR (Interrupt Service Routine) qui pourront être faites en plus du main(). Pour cela, des interruptions sont autorisées.

Le programme principal main( ) s'exécute puis une interruption se produit. Cette interruption entraîne l'arrêt temporaire de l'exécution du main( ), pour faire l'ISR.

Dans le cadre de ce TP, nous allons configurer un timer pour qu'il génère automatiquement une interruption dès qu'un certain temps (temporisation) se sera écoulé. Cela aura pour effet d'arrêter l'exécution du main( ) pour partir faire l'exécution de la routine d'interruption. Une fois la routine faite le programme reprend ce qu'il faisait dans le main( ).

Par exemple on peut imaginer une interruption toutes les 0,5 secondes qui fasse changer l'état d'une led :



3. Calcul du temps entre deux interruptions

La note d'application du fabricant AN4013.pdf page 11/45 donne :

$$F_{timer} = \frac{F_{clk\ timer}}{(PSC+1)(ARR+1)}$$

Avec :

F_{timer} : Fréquence entre deux interruptions du timer.
PSC : La valeur du Prescaler
ARR : La valeur du Autoreload Register

Rem : un prescaler est un diviseur de fréquence.

Fonctionnement :

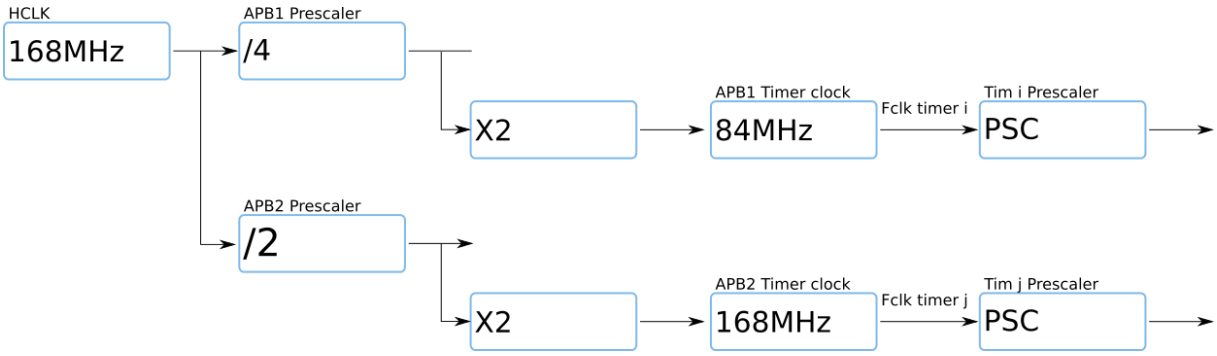
La valeur contenue dans le timer est incrémentée à la fréquence de $\frac{F_{clk\ timer}}{(PSC+1)}$

Lorsque la valeur du timer atteint la valeur contenue dans l'ARR, une interruption est automatiquement générée.

L'exécution du main( ) est interrompue pour exécuter la routine d'interruption du timer

Le timer est réinitialisé automatiquement et repart pour un nouveau cycle.

Les timers sont câblés de la façon suivante dans le STM32:



168 MHz correspond à l'horloge interne du STM32.
Des prescalers et des multiplieurs de fréquence sont calculé automatiquement par le fabricant pour générer la fréquence qui pilote chaque timer.

Dans le STM32F429 il y a 14 timers qui peuvent être pilotés à des fréquences différentes (p66/1751 datasheet)
Pour les timers i=2,3,4,5,6,7,12,13,14 → Fclk timer i = 84 MHz
Pour les timers j=1,8,9,10,11 → Fclk timer j = 168 MHz

4. Questions à faire sur feuille de brouillon et à faire valider à l'enseignant

- Quelle est la différence entre F_{timer} et $F_{clk\ timer}$?
- Combien vaut $T_{clk\ timer}$ du timer 7 ?
- Combien vaut $T_{clk\ timer}$ du timer 10 ?
- Faire un fichier Excel qui permette de calculer T_{timer} en utilisant la formule de la page précédente et trouvez des valeurs entières de PSC et d'ARR pour avoir les temps d'interruption suivants :

Ce tableau est à remplir sur votre brouillon et à faire valider par l'enseignant :

	Timer 7	**Timer 10**
T_{timer}	0,5s	1,234s
PSC		
ARR		

B. Créer un nouveau projet

La méthode de création de projet sera la même que celle faite dans le TP1 mais il vous faudra activer les timer 7 et 10 dans STM32 Cube MX.
Vous vous aiderez du lien suivant pour paramétrer correctement vos PSC et ARR des timer 7 et 10.
https://deepbluembedded.com/stm32-timer-interrupt-hal-example-timer-mode-lab/
Attention : Dans ce lien ils n'utilisent pas du tout le même STM32 que nous.

et

Vous reprendrez le sujet du TP1 pour créer un nouveau projet dans le Workspace I:\INFO2_STM32 et dont le nom de votre projet : Info2_TP4_1

C. Compilation et programmation du kit de développement STM32

Récupérez un kit de développement et connectez-le au port USB de votre PC.

Le port USB sur la carte STM32 est très fragile, manipulez-le avec précaution.
Le bouton poussoir noir ne doit jamais être utilisé

D. Faire clignoter la led verte et la rouge avec 2 timers

1. Faire clignoter la led verte toutes les 0,5s avec le timer 7

Pour information :

- Par défaut, même si vous avez configuré vos timers sous STM32 Cube MX, les interruptions sont désactivées. Il ne faut donc surtout pas oublier de les autoriser en utilisant la fonction : HAL_TIM_Base_Start_IT comme indiqué dans le site ressource : https://deepbluembedded.com/stm32-timer-interrupt-hal-example-timer-mode-lab/
- La routine d'interruption est commune à tous les timers. Elle est présente en bas de votre main.c mais elle ne teste pour l'instant que le timer 6. A vous de la compléter pour qu'elle teste en plus le timer 7
  Le nom de la routine d'interruption est : HAL_TIM_PeriodElapsedCallback

  C'est dans la routine d'interruption que vous pouvez faire changer l'état de la led (on vers off, ou off vers on). Vous pouvez utiliser la fonction HAL_GPIO_TogglePin

Sauvez, compilez et testez si cela fonctionne.

2. Faire clignoter la led rouge toutes les 1,234s avec le timer 10

Il ne reste maintenant plus qu'à ajouter 1 ou 2 lignes de code pour faire la même chose avec la led rouge mais cette fois toutes les 1,234s et avec le timer 10 que vous avez normalement déjà paramétré dans STM32 Cube MX.