Université Littoral Côte d'Opale

TD 3 : Chronomètre

Systèmes Embarqués
ING2, EILCO
P.-A. Hébert
2019–2020

Objectifs

- Mesurer le temps avec précision, à l'aide d'un périphérique interne timer.
- Utiliser le mécanisme d'interruption, dans le but d'exécuter une tâche de fond, « parallèlement » à l'action de la boucle principale.
- Gérer un dialogue opérateur (clavier et écran d'un PC), en communiquant via le port USB grâce à la classe `Serial`.

1 Cahier des charges

Vous devez réaliser un chronomètre, avec les contraintes suivantes :

- comptage *précis* des millièmes de seconde, des secondes et des minutes ;
- commandes de : *marche*, *arrêt*, *affichage sur l'écran du PC* et de *remise à zéro* ;
- affichage permanent d'un menu de commandes ;
- clignotement de la LED intégrée pour marquer le passage des secondes ;
- reconnaissance de commandes, saisies au clavier du PC ;
- communication avec le PC par la liaison USART, à 9600 bauds.

2 Quelques orientations de développement

- La précision requise incite à l'utilisation d'un timer.
- Les instructions de comptage du temps devant être effectuées sans délai, il faudra utiliser les interruptions proposées par le timer. Autrement dit, c'est le timer qui décidera quand déclencher ces instructions, en interrompant le processeur.
- Ces interruptions périodiques auront lieu **toutes les millisecondes**, et serviront à mettre à jour les variables globales de temps ;
- Les fonctions temporelles de l'*Arduino* (`delay`, `micros`, `millis`, etc.) fonctionnent grosso modo de cette manière, en s'appuyant sur le *timer 0*. Mais l'objectif ici est d'apprendre à manipuler à la fois les timers et les interruptions, quel que soit le microcontrôleur :

  Vous n'utiliserez donc pas les fonctions de gestion de temps de l'Arduino !
- Le *timer 0* étant déjà utilisé par ces fonctions de temps, vous utiliserez l'autre timer 8 bits de l'*Uno* : le ***timer 2***.
- Vous pourrez proposer le menu dans la fonction `Loop`, et attendre une saisie : cela n'empêchera pas la mesure du temps de vos fonctions temporelles, grâce aux interruptions.
- Dialogue avec l'opérateur sur le PC assuré par la liaison série USART de l'*Arduino*, grâce à l'objet `Serial` et ses méthodes :
  - affichages : `Serial::print, Serial::println`
  - saisies : `Serial::parseInt, Serial::readString`.

Vous lirez attentivement les explications du cours sur les timers.

Question 1

Écrire une fonction `void init_T2()` destinée à configurer le *timer 2* pour qu'il génère une interruption toutes les millisecondes, précisément.

Comme c'est précisé dans le cours, il faut pour cela paramétrer l'ensemble des registres du *timer 2*, avec les objectifs suivants :

1. le placer en mode *CTC* : cela laisse plus de choix de périodes d'interruption ;
2. définir sa vitesse de comptage (choix du prédiviseur) et affiner la durée de son cycle (choix de la valeur de comparaison A) ; pour cela, utiliser l'équation liant la période de cycle de comptage et ces 2 paramètres (cf. cours) ;
3. autoriser son interruption ;
4. initialiser le compteur de manière à ce que le 1er cycle de comptage dure aussi longtemps que les suivants (soit 1 milliseconde).

Question 2

Écrire une fonction `void desactive_T0()` destinée à arrêter le *timer 0* : on n'a plus besoin de la mesure du temps faite par l'*Arduino*, et on ne veut pas que ses interruptions viennent perturber celles du *timer 2*.

Question 3

Définir une structure `Horloge` destinée à mémoriser nos 3 variables de temps : *millisecondes, secondes* et *minutes*.

Définir une instance globale de cette structure, que vous nommerez `chrono`. Vous la déclarerez `volatile` de manière à forcer le compilateur à la lire systématiquement en mémoire.

Question 4

Écrire la fonction d'interruption associée aux fins de cycle du timer : `ISR(TIMER2_COMPA_vect)`. C'est cette fonction qui mesure le temps : « appelée » chaque milliseconde *par le timer 2*, elle doit mettre à jour la structure `chrono`.

Question 5

Écrire une fonction `void affiche_chrono()`.

Question 6

Placer dans la fonction `void Loop()` une unique instruction : l'affichage du chrono.
Et remplir la fonction `void Setup()` de manière à lancer l'évolution du chronomètre.

3 Questions subsidiaires

Question 7

Mettre en place le clignotement de la LED : elle devra s'allumer au moment précis où la seconde s'incrémente, et s'éteindre un quart de seconde plus tard.

Question 8

Dans la fonction `void Loop()`, mettre en place le mécanisme de proposition de menu, d'attente d'un choix de la part de l'opérateur au clavier PC (instruction *bloquante*), et d'exécution de cette commande.

Question 9

Reprendre la question précédente, mais en évitant toute instruction bloquante : même si le mécanisme d'interruption permet de contourner ce blocage, il est toujours préférable de ne pas « monopoliser » le processeur inutilement.
Il devient ainsi possible d'ajouter des traitements à la fonction `Loop`.