R3.07 : INFORMATIQUE EMBARQUEE
TP6 : Interruptions, structure de données

1 Objectifs

Ce TP initie d'abord à la mise en œuvre des **interruptions** sur le timer 0 en faisant changer d'état toutes les secondes une LED dans la routine d'interruption.

Ensuite, il fait créer une **structure de données** heures/minutes/secondes afin d'afficher l'heure sur l'**écran du terminal série**. Celle-ci s'actualisera toutes les secondes dans la routine d'interruption du timer 0.

Enfin, il permet de régler l'heure grâce au **clavier du terminal série**. Cette tâche est hébergée dans la boucle infinie de la fonction **`main()`**.

2 Projet TP6A : interruptions sur timer 0

2.1 Cahier des charges

Le programme à écrire doit assurer une interruption (priorité haute) du timer 0 toutes les secondes exactement. Dans la routine d'interruption, on change l'état de la LED connectée à RD0, ce qui permettra de vérifier que le programme fonctionne bien.

Pour l'instant **`main()`** comporte, outre les initialisations, une boucle infinie vide : **`while(1) {}`**

Toutes les initialisations du timer 0 (y compris celles liées à ses interruptions) seront faites dans une fonction **`void init_TIMER0(void)`**.

Toutes les initialisations des entrées/sorties TOR seront faites dans une fonction **`void init_ES(void)`**.

2.2 Validation du travail

- Écrivez votre programme et testez-le : 30 allumages/extinction de la LED durent 60 secondes si tout va bien.

3 Projet TP6B : communication avec le terminal série

3.1 Cahier des charges

Il s'agit de continuer le programme précédent en réalisant les initialisations qui permettent de communiquer avec le terminal série GtkTerm en 19200,8,N,1. Quand cela est fait, le message d'accueil **`"HORLOGE\r\n"`** est affiché sur le terminal série.

Dans la routine d'interruption, on incrémentera une variable de type **`unsigned int`** et on l'affichera.

La boucle **`while(1)`** de **`main()`** est toujours vide.

3.2 Préparation

La communication avec un terminal série est décrite à la partie **3.2 Préparation** du TP3.

La déclaration et l'initialisation d'un **`unsigned int`** se font comme ceci :

```
unsigned int n = 0;
```

L'affichage de cet **`unsigned int`** sur 5 caractères avec les 0 non significatifs affichés se fait comme ceci :

```
printf("%05u", n);
```

Le concept de **variable globale** peut vous être utile. Il est expliqué à la partie 4.2.

3.3 Validation du travail

- Écrivez votre programme et testez-le : Le message d'accueil est affiché, la LED change d'état toutes les secondes et **`n`** s'incrémente au même rythme.

4 Projet TP6C : création d'une structure pour la gestion du temps

4.1 Cahier des charges

Une structure de données de nom **`myTimeStruct`** sera créée. Elle permettra de gérer le temps au format heure, minute, seconde.

Une variable globale **`temps`** de type **`myTimeStruct`** sera créée et initialisée avec des valeurs non nulles, par exemple pour recréer l'heure 23:58:30.

Dans la routine d'interruption, on gérera et affichera **`temps`** comme il faut, par exemple **`23:58:30`**. On s'assurera que lors de l'incrémentation d'une seconde, certaines valeurs ne peuvent être dépassées. Par exemple, si les secondes sont à 59, elles passent à 0 et les minutes s'incrémentent, sans toutefois dépasser 60, etc.

La boucle **`while(1)`** de **`main()`** est toujours vide.

4.2 Préparation

La déclaration de la structure **`myTimeStruct`** se fait comme ceci, juste après les **`#include`** et **`#define`** de votre programme :

```
typedef struct {
    unsigned char h;
    unsigned char m;
    unsigned char s;
} myTimeStruct;
```

Ensuite, pour créer et initialiser la **variable structurée `temps`** (juste après le bloc de code précédent) :

```
myTimeStruct temps = {23, 58, 30};
```

Par la suite, à titre d'exemple, si vous voulez accéder au **champ** des secondes de **`temps`**, vous le faites par **`temps.s`**. Si vous voulez incrémenter le champ des minutes, vous pouvez le faire par :

```
temps.m = temps.m + 1; // Alternative : temps.m++;
```

Remarque : En déclarant la variable **`temps`** en dehors de toute fonction, on la rend **globale**, c'est-à-dire qu'elle est accessible et modifiable depuis toute fonction. Cela correspond à notre besoin puisque la fonction de routine d'interruption (qui ne peut admettre de paramètres ni retourner de valeur vu son caractère asynchrone) doit la gérer et **`main()`** également quand il s'agira de régler l'heure avec le clavier du terminal série. Il ne faut pas abuser des variables globales car il n'est pas sécurisé que toutes les fonctions puissent y accéder et les modifier (mais parfois on ne peut faire autrement). D'autres techniques d'accès/modification existent pour pallier à cette non-sécurité mais elles dépassent le cadre de cette introduction.

Remarque : Les compilateurs C disposent tous d'une bibliothèque **`time.h`**, beaucoup plus riche que ce que l'on est en train de faire. N'hésitez pas à vous y intéresser dans le cadre de projets plus conséquents. En outre, des composants électroniques dits *Real Time Clock* (avec batterie et quartz), permettent de donner l'heure et la date à un microcontrôleur, via le bus SPI par exemple. Ainsi le microcontrôleur peut horodater tout ce qui doit l'être.

Remarque : Les types et les variables structurés sont à l'origine des **langages orientés objets** comme Java ou C++.

4.3 Validation du travail

- Écrivez votre programme et testez-le. La gestion des heures, minutes, secondes fonctionne-t-elle bien ?

5 Projet TP6D : réglage de l'heure

5.1 Cahier des charges

Il s'agit maintenant de pouvoir régler l'heure. Cela sera fait dans le **`while(1)`** de **`main()`**. On scrutera le bit **`RCIF`** et on exploitera le registre **`RCREG`**. Par exemple, les appuis sur les touches suivantes ont les effets suivants :

`'s'` remet à 0 les secondes.

`'m'` décrémente les minutes.

`'M'` incrémente les minutes.

`'h'` décrémente les heures.

`'H'` incrémente les heures.

5.2 Validation du travail

- Écrivez votre programme et testez-le. Pour ce dernier point, proposez à l'enseignant une procédure de test.

6 S'il reste du temps

- Implémentez les cahiers des charges B et C du TD8 (gestion du relâchement du bouton RB1).