TP du 21 avril 2016 – Interruptions et timers

Ce TP se fera avec le LaunchPad et le petit module rouge pour avoir 2 boutons.

Il y aura davantage de programmes à lire que de lignes de code à écrire !

Prenez le temps de bien voir à quoi sert chaque instruction dans les exemples donnés. Vous pourrez ensuite facilement faire les modifications demandées.

1) **Interruption sur une entrée**

Prenez le programme Inter. Examinez-le. Testez-le.

Avec certains boutons, vous pourriez observer des rebonds de contact.

- Modifiez le programme pour qu'il réagisse au relâchement du bouton.
- Modifiez-le ensuite pour qu'il réagisse aux deux flancs.

2) **Communication entre la routine d'interruption et le programme principal**

On a souvent besoin que l'action se produise dans le programme principal. Créez une variable globale `char flagInter`. Remplacer l'action sur la LED dans le routine d'interruption par l'instruction : `flagInter = 1;`

Dans la boucle principale, écrivez :

```
if (flagInter) {
  Led1Toggle;
  flagInter = 0;
}
```

Si vous avez observé des rebonds de contacts, vous pouvez les supprimer en ajoutant `AttendeMs(20)` après le `Led1Toggle`.

3) **Timer en mode « polling »**

Prenez le programme Timer.c Examinez-le. Que doit-il se passer ? Testez-le.

Modifiez le programme en changeant la ligne d'initialisation du Timer :

```
TACTL = TASSEL_2 + ID_1 + MC_2;
```

- Que doit-il se passer ? Testez !

4) **Timer en mode « Interruption »**

Prenez le programme Timer-Inter-Compare.c. Examinez-le. Testez-le.

- Que se passe-t-il si on modifie au départ la valeur de TACCR0 ? (par exemple : `CCR=1000)`. Pourquoi ?

5) **PWM par Timer et interruptions**

Prenez le programme Timer-Inter-2. Examinez-le en détail. Testez-le.

Vous avez presque ce qu'il faut pour gérer un PWM (et même deux) :

- Mettez Led1On et Led1Off aux bons endroits pour faire un PWM
- Augmentez la fréquence (16 MHz, pas de division)
- Utilisez CCR1 pour le second PWM (sur Led2).