LABORATOIRE 3 | *Systèmes d'information*
Interruptions

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

But du laboratoire (4 périodes)

1. Le but de ce laboratoire est de mettre en pratique vos connaissances théoriques sur les interruptions. Vous allez dans un premier temps réaliser une interruption simple puis, vous allez expérimenter les effets des interruptions sur le déroulement normal d'un programme.
2. Vous devez réaliser ce laboratoire par groupe de deux personnes.
3. La durée estimée pour réaliser ce laboratoire est de **quatre périodes**.
4. Vous trouverez toutes les fichiers relatifs aux labos sur le site web du cours (http://sin.begincoding.net)

Partie 1 – Interruptions de base

Nous avons vu comment fonctionnent les interruptions durant les cours. Nous allons maintenant mettre en pratique ces connaissances sur un exemple simple. Dans celui-ci, on désire réaliser un programme utilisant le moins d'énergie possible et qui, lors de l'appui sur le bouton central, change l'état de toutes les leds. Pour ce faire, on utilisera les interruptions dont le schéma de fonctionnement est rappelé ci-dessous. Pour rappel, ce schéma montre comment quelques sources externes et les périphériques peuvent interrompre le processeur.

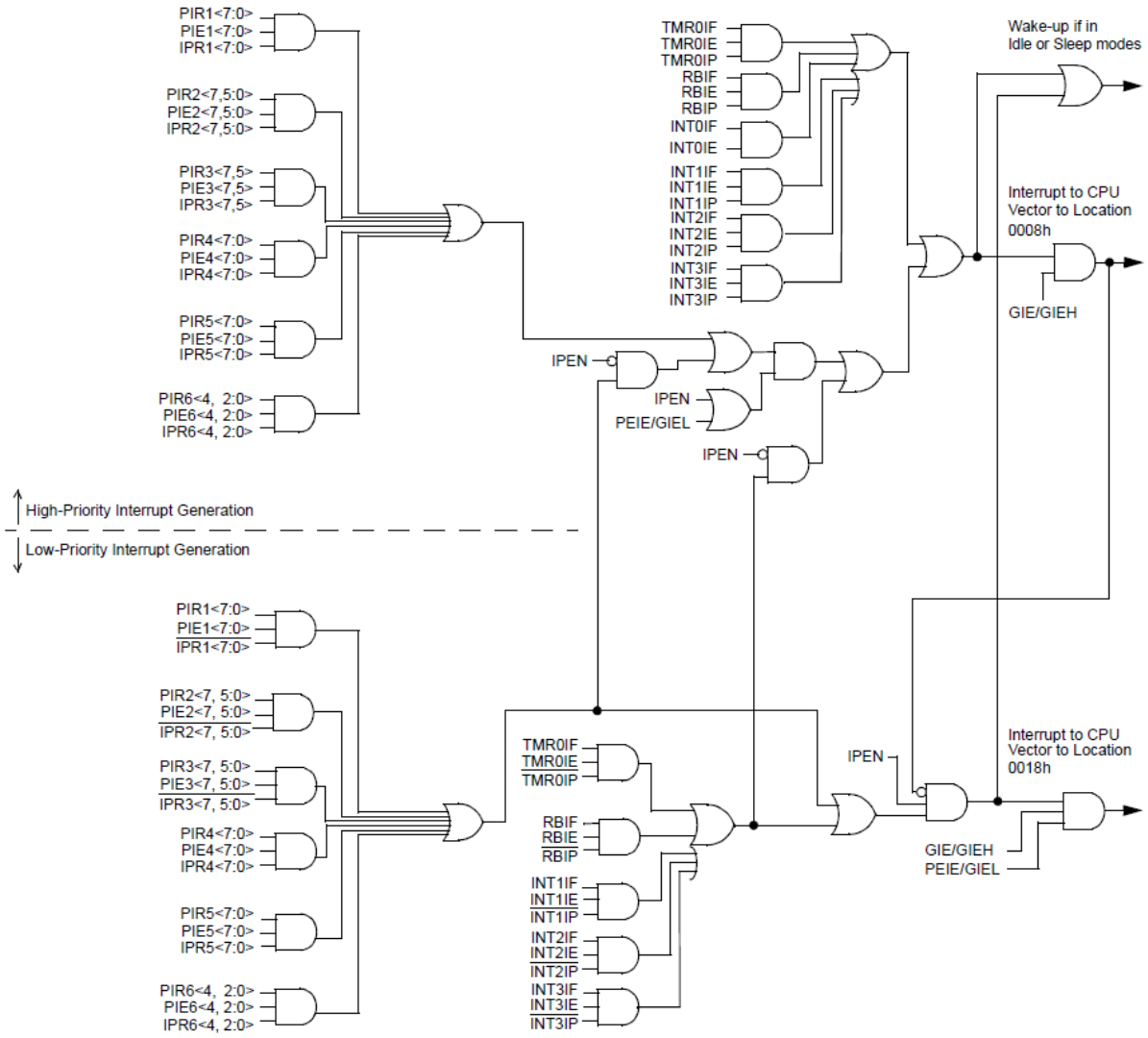

Figure 1 – Schéma des interruptions du PIC18F87k22, tiré de la datasheet *Microchip*

Tâche 1

1. Configurez les interruptions pour réagir sur le bouton central de la carte PICEBS2 (actif haut).
2. Écrivez le code de la routine d'interruption permettant de changer l'état de toutes les leds connectées sur le PORTH.
3. Testez le bon fonctionnement de votre programme
4. Mesurez la consommation de courant moyenne de la carte. Cette mesure doit se faire entre les points de test blanc et jaune de la carte PICEBS2 sachant qu'une résistance de 1 ohm est placée entre ces deux points de test.
5. Faites-en sorte de mettre le processeur en mode *sleep* lorsqu'il ne traite pas les boutons.
6. Effectuez à nouveau la mesure de courant. Que constatez-vous lors de cette nouvelle mesure ? Vous attendiez-vous à cela ?

   .....

   .....

7. Touchez avec votre doigt les pins du microcontrôleur et mesurez le courant. Que constatez-vous et pourquoi ?

   .....

   .....

Partie 2 – Latence des interruptions

Explication théorique

Lors d'une interruption acceptée par le processeur, celui-ci doit terminer l'instruction en cours de traitement puis charger la première instruction de la routine d'interruption. Le temps nécessaire pour cela s'appelle le temps de latence (*interrupt latency time*). Il se mesure en nombre de cycles CPU. Passé ce temps, le processeur exécutera l'instruction se trouvant à l'adresse `0x08` (*interrupt vector*) si la notion de priorité n'est pas activée ou s'il s'agit d'une interruption haute priorité.

Dans cette deuxième partie de labo, nous allons mettre en avant les effets de la latence des interruptions ainsi que ses conséquences.

Tâche 2

On désire générer un signal carré sur la pin `RH0` du processeur. Ce signal doit avoir la fréquence la plus élevée possible. Dans le même temps, on désire compter le nombre d'impulsions effectuées sur le bouton `BT0/RB0` (bouton de gauche) de la carte. Il ne faut pas en perdre une seule. Cette valeur (8 bits) sera placée en mémoire et pourra être visualisée à l'aide du *debugger*.

1. Écrivez le code pour générer le signal carré le plus rapide possible sur la pin `RH0`.
2. Visualisez votre signal à l'oscilloscope et vérifiez qu'il soit bien carré. Quelle est la fréquence que vous obtenez ?

   .....

3. Écrivez le code nécessaire à la gestion des interruptions sur le bouton. Vous pouvez reprendre les résultats de la tâche numéro 1. N'oubliez pas d'effectuer la sauvegarde des registres.
4. Que constatez-vous sur le signal à l'oscilloscope lorsque vous appuyez sur le bouton. Comment l'expliquez-vous ?

.....

.....

.....

Tâche 4 – First steps in C programming

For this last task we will program interrupts using the C programming language (even though we might not have seen the theory about it). The task to be done is to measure the speed of an interrupt on RB0, which corresponds to the speed of a fan that produces one pulse per revolution. The connection schematic will be provided in class.

For this task, you have to create a new C project as follows :

1. Create a new projet in MPLAB as usual but instead of assembler (**`mpasm`**), choose the **`xc8`** compiler.
2. Download the configuration source files (**`picebs.h`** and **`picebs.c`**) for the PICEBS2 board and copy them to the projet folder (there are located here http://sin.begincoding.net/documentation/C/)
3. Right-click on the projet and choose *Add exisiting item* and select the **`picebs2.h`** file
4. Add a new main source file by selecting *File -> New file*
5. Add the following base code (which does nothing).

```
#include "picebs2.h"

void interrupt myInterruptRoutine(void)
{

}

void main()
{

    while (1)
    {

    }
}
```

As in Java, the main function is the entry point of your program. Here, the function **`myInterruptRoutine`** will be located at address 0x8 in the code memory (the interrupt vector). After having checked that everything compiles, complete this code to count the number of interrupts and display the result on the bar graph. The speed should be displayed in arbitrary time units.