2 Utilisation d'un temporisateur sous interruption

Description générale

- **Date du TP :** 4ième semaine ;
- **Durée du TP :** 3h (2semaines) ;
- **Matériels nécessaires** : Carte d'experimentation, microcontrôleur PIC 16f877, programmateur, des composants électroniques ;
- **Logiciels utilisés :** Complilateur C PCW, Simulateur ISIS, IC Writer ;
- **Public cible :** 4ième niveau d'études de technicien supérieur en informatique industrielle ;
- **Partie du cours en rapport avec le TP :** Classification architecturale des systèmes temps reel, les microcontrôleurs.

Objectifs

- Manipuler les temporisateurs (timer) ;
- Programmation du timer en interruption ;
- Se familiariser avec les outils de programmation et de développement ;
- Approfondir les acquis en programmation C des microcontrôleurs PIC ;
- Maitrise des architectures des microcontrôleurs Microchip PIC16.

Pré requis

Notions générales sur :
- L'électronique ;
- Algorithmique ;
- Programmation en C ;
- Les microcontrôleurs.

2.1 Le timer 0 ou RTCC

Présent sur tous les modèles de PIC, le timer 0, qui s'est appelé par le passe RTCC au point que l'on retrouve encore frequemment ce sigle, est le plus simple qui nous soit offert. C'est en effet un banal compteur sur 8 bits dont la source d'horloge peut être interne ou externe et subir ou non une prédivision par un taux programmable entre 1 et 256. " Lorsqu'il déborde, c'est-à-dire lorsqu'il atteint 255, ou FF en hexadécimal, il peut générer une interruption puis il reprend ensuite indéfiniment son cycle de comptage a partir de zéro. On l'utilise généralement comme base de temps dans les programmes, en exploitant la possibilité de générer une interruption lors de son débordement, car il est possible de programmer très Facilement la frequence a laquelle elle se produit. En effet, si l'on utilise comme source d'horloge de ce timer l'horloge interne du microcontrôleur, on peut écrire la relation suivante :

$$Temps = (256 - PRC) * 4 * PRD * 1/FH$$

où :
- $Temps$ est le temps qui s'écoule entre deux interruptions de débordement du timer exprimé en seconde ;
- PRC est la valeur (en décimal) pré chargée dans le registre de comptage du timer. Cette valeur doit être immédiatement rechargé lors de chaque interruption de débordement ;
- PRD est la valeur sélectionnée pour le prédiviseur du timer ;
- FH est la fréquence d'horloge système exprimée en Hz.

2.2 Édition du programme

La figure 2.1 propose un schéma qu'il est possible de réaliser très facilement, et le programme associe est visible dans le listing 2 présenté à la fin de cette manipulation.

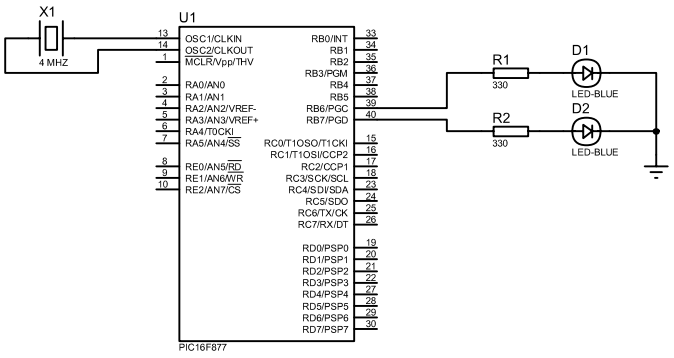

Fig. 2.1 – Schéma de l'utilisation du timer 0 sous interruption

Ses fonctions restent fort simples puisqu'il se contente d'inverser l'état des LED connectées

sur les lignes B7 et B6 du port B toutes les 250 ms, grâce a 1'exploitation du timer 0 sous interruption.

```
//*************************  listing 2 ******************************

#include <16F877.h>
#use delay(clock=4000000)
#fuses NOWDT,XT, PUT, NOPROTECT, BROWNOUT, LVP, NOCPD, NOWRT
int millisec, leds;
#int_TIMER0
void TIMER0_isr()                       // Définition du pgm d'it du timer0
{
   set_timer0 (6);                      // Débordement toutes les ms
   if (millisec++ == 250)
   {
      millisec = 0;
      leds = leds ^ 0xC0 ;              // Inversion de l'état des LED
      output_b(leds); } }

void main()
{
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_4);
enable_interrupts(INT_TIMER0);     // activation de l'interruption de
                                   //débordement du timer 0
enable_interrupts(GLOBAL);             // activation de tout les interruptions
leds = 0x80;
while (true);                          // Le pgm principal ne fait rien
}


//************************fin du Listing*********************************
```

Comme vous pouvez le constater, le programme principal ne fait rien si ce n'est les indispensables initialisations du timer 0 (notez à ce propos le taux de 4 appliqué au prédiviseur) et la validation de l'interruption issue de ce même timer. Tout a lieu en fait dans le programme de traitement des interruptions dont la première fonction est de recharger le registre de comptage du timer comme nous l'avons expliqué ci-dessus. La valeur utilisée a été déterminée avec notre relation, compte tenu du fait que nous avons défini une horloge à 4 MHz, de façon à générer une interruption toutes les millisecondes. Lorsque 250 de ces interruptions ont eu lieu, l'état des LED est inversé et le programme se poursuit sans fin. L'interêt majeur de ce mode de fonctionnement est que le programme principal, ici notre while(true) qui est donc une boucle sans fin, peut executer toutes les tâches que requiert votre application sans avoir à se soucier de celle de comptage du temps qui est gérée par le timer 0 et sa routine d'interruption.

2.3 Travail demandé

1. Compiler le programme avec PCW par la suite vérifier son bon fonctionnement à l'aide de la simulation par ISIS.
2. Expliquer puis commenter le programme présenté dans le listing.
3. Vous pouvez expérimenter à loisir sur cet exemple en modifiant, tant la valeur pré chargée dans le registre de comptage du timer, que le taux du prédiviseur utilisé afin de découvrir toutes les possibilites offertes.