Université catholique de Louvain
École polytechnique de Louvain

Systèmes informatiques II (INGI1113)

Projet 2 - Programmation d'un PIC

– 20 octobre 2013 –

Travail du groupe G35 :

Derval Guillaume 68911100 Gégo Anthony 28581100

1 Introduction

En guise de deuxième projet pour le cours de Systèmes informatiques II, nous avons été invités à réaliser un réveil à l'aide d'une carte programmable Olimex PIC-Maxi-Web, équipée d'un microcontrôleur PIC18F97J60. Le réveil devait être en mesure d'afficher l'heure au format 24h, de permettre à l'utilisateur de modifier l'heure et de paramétrer une alarme.

2 Utilisation des compteurs

Un compteur (counter/timer) est un circuit électronique digital dont l'entrée est un signal d'horloge (un signal rectangulaire d'une fréquence donnée) et dont la sortie est l'équivalent binaire d'impulsions compté depuis la mise sous tension ou la remise à zéro du compteur. Dans notre cas, ces compteurs sont parties intégrantes du PIC et sont manipulables au travers du jeu d'instructions de ce dernier. De plus, une fois que le compteur a dépassé sa capacité, il génère une interruption.

Pour mesurer le temps et afficher l'heure, nous avons besoin de recourir à l'utilisation d'un timer. Nous en disposons de deux de 16 bits (représentés sur deux entiers non signés de 8 bits), un premier (timer 0) qui utilise la fréquence d'horloge principale du PIC (environ 40Mhz), et un second (timer 1) qui utilise une fréquence d'horloge provenant d'un cristal oscillant à exactement 32.768 kHz. Il est ainsi clair que le second aura atteint sa capacité au bout de deux secondes précisément. Il nous a cependant été demandé de travailler avec le premier timer.

Dès lors, nous avons tenté de déterminer le nombre d'interruptions nécessaires au timer 0 (initialisé à 0) pour rattraper une interruption de timer 1. Afin de ne pas provoquer trop d'interruptions inutiles, nous avons fixé l'échelle du timer 0 à 1/64. Ainsi, le timer ne sera incrémenté qu'après 64 impulsions. Ensuite, pour obtenir une bonne moyenne, nous avons fixé l'échelle du timer 0 à 1/16. Nous avons calculé qu'il fallait en moyenne 1532 interruptions en 16 secondes, soit 95.75 interruptions par secondes. Choisir une échelle plus élevée pour le timer 0 aurait certes réduit le nombres d'interruptions à gérer mais également réduit la qualité de l'estimation.

Notre programme maintient ainsi en mémoire le nombre d'interruptions du timer 0 depuis minuit, et la divise par 95.75 pour obtenir le nombre de secondes écoulées depuis. Bien évidemment, pour réduire les erreurs de précision, cette division a lieu sur la somme et non sur chaque incrément.

3 Gestion des évènements

Un évènement se produit lorsque l'utilisateur appuie sur un bouton. En tant qu'entrée/sortie, ceci provoque une interruption.

Afin de laisser la priorité à l'incrémentation de la variable associée au compteur, les interruptions liées aux boutons ont définies avec une basse priorité. Ainsi, lors de l'exécution d'une de ces interruptions, une interruption du compteur pourra mettre en pause l'exécution des opérations liées à un évènement utilisateur pour les reprendre après avoir exécuté l'incrémentation.

Un bref descriptif des fonctionnalités représentées par ces boutons est disponible ci-dessous.

4 Mise à jour de l'écran

Un problème majeure s'est posé à nous quant à la mise à jour de l'écran. Il n'existe effectivement pas de système d'interruption pour cette opération. L'écran doit être rafraîchi au moins toutes les secondes et par principe de garder les fonctions liées au interruptions de haute priorité courtes, nous n'avons pas préféré inclure tout ce code lors de l'incrémentation du timer 0. Le programme réalise donc de l'attente active pour la mise à jour de l'écran au prix de la garantie d'un compteur précis.

L'utilisation des interruptions a effectivement comme avantage de ne pas utiliser pour rien les cycles de calculs car elles sont matérielles.

5 Modes d'exécution

Notre programme repose sur 4 principaux modes d'exécution :

— Mode 0 : L'heure est affichée et actualisée toutes les secondes. C'est le mode par défaut.
— Mode 1 : Une alarme peut être définie ; l'utilisateur spécifie l'heure à laquelle il souhaite voir le réveil sonner à l'aide des boutons. Une fois l'heure spécifiée, ce mode est quitté.
— Mode 2 : L'heure du réveil peut être définie ; l'utilisateur spécifie l'heure à l'aide des boutons. Une fois l'heure spécifié, ce mode est quitté et l'heure est mise à jour.
— Mode 3 : L'alarme a été déclenchée. Une led clignote toutes les secondes pendant 30 secondes. L'utilisateur peut l'arrêter manuellement. Une fois l'alarme arrêtée, le mode 0 est rétabli et l'alarme est désactivée.

Pour le mode d'édition, un vecteur de taille 3 fait office de mémoire pour les heures, minutes et secondes spécifiées.

6 Manuel d'utilisation

6.1 Démarrage du réveil

Au démarrage du réveil, l'heure est initialisée à minuit.

6.2 Configuration de l'heure

Une fois le réveil démarré ou si l'heure courante est affichée, vous pouvez à tout moment configurer l'heure courante en appuyant sur le bouton 2. Un nouvel écran s'affiche à vous.

Le champ (heures, minutes ou secondes) dont vous effectuez la modification se met à clignoter. Pour incrémenter ce champ, appuyez sur le bouton 1. Pour passer au champ suivant, appuyez sur le bouton 2. Une fois le dernier champ atteint, vous devrez confirmer votre saisie via le bouton 2 également.

6.3 Configuration de l'alarme

De la même manière que pour la configuration de l'heure, vous pouvez configurer l'alarme en appuyant sur le bouton 1 lorsque le réveil est démarré oui si l'heure courante est affichée.

6.4 Arrêt de l'alarme

Lorsque l'alarme se déclenche, deux possibilités s'offre à vous. Vous pouvez soit arrêter l'alarme par vous-même en appuyant sur un des deux boutons, ou soit attendre 30 secondes que l'alarme s'arrête par elle-même. Attention, lorsque l'alarme est désactivée, elle n'est plus en mémoire, il faut alors reconfigurer l'alarme.

7 Conclusion

Ce projet nous a permis de maîtriser la notion d'interruptions et de la mettre en pratique ainsi que de se confronter à des notions de programmation de plus bas niveau telles que la manipulation de compteurs.

Le programme principal est `clock.c`, et peut être construit via la commande `make clock`. Le programme `test.c` est celui utilisé pour calculer le nombre d'interruptions du timer 0 équivalente à une interruption du timer 1. Il peut être construit via la commande `make test`. Les deux programmes peuvent être construit en tapant simplement `make`.