LINGI1113 - SYSTÈMES INFORMATIQUE 2

Projet PIC

Professeur :

Marc LOBELLE

Étudiants : (Groupe 7)

Julien COLMONTS 41630800

Vincent VAN OUYTSEL 19890900

Programme :

SINF13BA

Année académique 2012-2013

1 Introduction

Le projet PIC est le second projet à réaliser dans le cadre du cours de systèmes informatiques de $3^{ième}$ Bac. Nous devions réaliser un réveil doté d'une alarme sur une "machine nue". Le matériel mis à disposition est un *PIC MAXI-WEB* de la société *OLIMEX*. Dans ce rapport, nous allons d'abord présenter les programmes écrits par nos soins. Ensuite, nous détaillerons les problèmes que nous avons rencontrés au long de ce projet.

Les différents documents utilisés pour ce projet sont téléchargeables aux adresses suivantes :

- La datasheet du PIC MAXI-WEB : http://ww1.microchip.com/downloads/en/DeviceDoc/39762f.pdf
- Le schéma de la carte PIC MAXI-WEB : https://www.olimex.com/Products/PIC/Development/PIC-MAXI-WEB/resources/PIC-MAXI-WEB_sch.pdf

2 Test de la vitesse d'horloge

La première partie de ce projet consistait à calculer la fréquence exacte de l'horloge du PIC. Ce calcul pouvait être effectué de manière très simple. En effet, le *PIC-MAXI-WEB* possède un temporisateur dont l'oscillateur interne peut être facilement réglé pour lancer une interruption après très exactement une seconde. Il nous a alors suffit de programmer le lancement simultané de deux temporisateurs (Timer0 et Timer1, démarrage à une instruction près), le Timer1 étant celui qui overflow après une seconde. Une fois l'interruption déclenchée, il suffit de regarder la valeur contenue dans l'autre temporisateur. Le Timer0 était réglé de manière à ne pas générer d'interruption. Pour être certain qu'il n'overflow pas pendant la seconde qui s'écoule, il fallait programmer le prescaler du Timer0 sur sa valeur maximum. L'incrémentation du temporisateur ne se fait alors que toutes les 256 instructions. On introduit donc un peu d'incertitude car, lors de l'overflow du Timer1, dans le pire cas, on manquera 255 incrémentations du Timer0. On sait, grâce à la documentation, que la fréquence de l'horloge est de 25MHz. Nous pouvons donc signaler que l'erreur est faible ($\approx 0,001$ MHz). Il faut également multiplier la valeur obtenue par 4 car une instruction se déroule sur quatre coups d'horloge. Les commentaires présents dans le code expliquent l'initialisa-

tion des différents registres pas à pas. Un calcul simple permet alors d'obtenir la fréquence du PIC :

$$Freq_{PIC} = Value_{TMP_0} \times 256 \times 4$$

Après avoir effectué ce test, nous avons mesuré que la fréquence du PIC est de 25,350 MHz. On peut donc conclure que notre programme calcule relativement précisément la fréquence de l'horloge.

Si vous souhaitez effectuer le test, le programme se situe dans le répertoire PIC_CHIPFREQ. Nous avons utilisé les fichiers de tests donnés pour commencer le projet et avons modifié le contenu de TEST.C pour effectuer les opérations demandées. Pour compiler le programme de test, il suffit donc d'utiliser la commande MAKE. Après avoir flashé le programme sur la carte, l'écran LCD donne les informations voulues.

3 Réveil

Documentation pour l'utilisateur

Les fonctionnalités du réveil étaient principalement dictées dans l'énoncé du projet. Lors de la mise sous tension du système, l'horloge s'initialise à 00h00m00s (comme un réveil traditionnel). Deux boutons sont présents sur la carte :

- Le bouton 1 : il s'agit du bouton du dessous.
- Le bouton 2 : il s'agit du bouton placé le plus haut sur la carte.

Au démarrage, le réveil affiche un état standard. On peut y voir l'heure défiler. L'alarme est désactivée par défaut, l'heure de celle-ci est initialisée à minuit.
En appuyant sur le bouton 1 sur cet état standard, on peut activer/désactiver l'alarme. L'affichage se met alors à jour, indique "Alarm ON" suivi de l'heure sur laquelle il est réglé.
Pour ouvrir le menu, il suffit d'appuyer sur le bouton 2. Le réveil propose alors un premier choix, celui de régler l'heure courante. Ré-appuyer sur le bouton 2 affiche l'option suivante du menu :

régler l'heure de l'alarme. Pour chacune de ces options, une pression sur le bouton 1 permet d'accéder au paramétrage de l'heure que l'on souhaite modifier. Tandis qu'un troisième appuis sur le boutons 2 nous ramène à l'état initial.

Pour modifier une heure, -une fois entré dans le menu adéquat- il suffit d'appuyer sur le bouton 1 pour incrémenter l'unité de temps entre crochet (heure ou minute) d'une unité. Pour valider votre réglage, appuyez sur le bouton 2.

Lorsque l'alarme sonne, l'utilisateur est prévenu par un message sur l'écran (l'heure actuelle continue de s'afficher sur la ligne inférieur de l'écran) et les LED rouges clignotent pendant 30 secondes, à raison d'une oscillation par seconde. Une pression sur n'importe quel bouton permet d'arrêter la sonnerie. Si l'utilisateur ne modifie ni l'heure courante, ni l'heure d'alarme, le réveil est programmé pour sonner une et unique fois par jour. Il vous est donc possible de réactiver l'alarme dans la même minute que celle où le réveil doit sonner sans qu'il ne déclenche à nouveau.

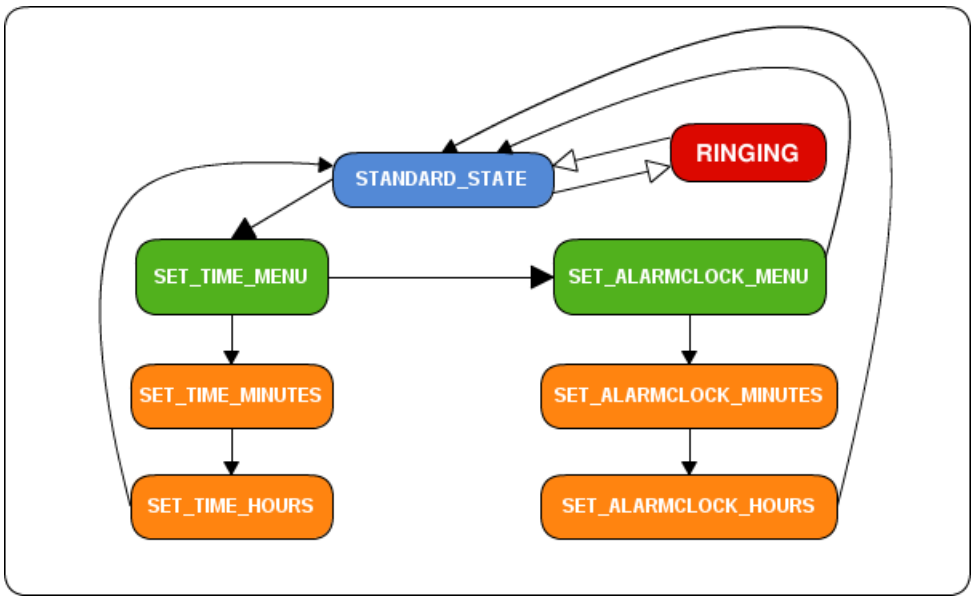
Documentation pour l'installateur

Notre programme principal ce trouve dans le sous-répertoire PIC_ALARMCLOCK de l'archive "COLMONTS_VANOUYTSEL.TAR.GZIP". Il Pour réaliser le programme du réveil, nous sommes à nouveau parti des fichiers donnés comme test avec le projet. Le seul fichier modifié est TEST.C. Le makefile est donc toujours correct pour compiler l'application à porter sur le PIC. L'utilisation de la commande MAKE est suffisante pour créer le fichier à envoyer.

Documentation pour le programmeur

Notre programme a été réalisé sur une base très simple. En effet, la majorité du travail consistait à comprendre le fonctionnement de base du PIC. Le programme en lui-même était assez simple à réaliser. Nous avons donc basé notre application sur huit états différents, permettant de savoir ce qu'il faut afficher sur l'écran LCD mais également quelles fonctionnalités possèdent les boutons à

chaque instant. Voici un diagramme des transitions entre ces différents états :



L'élément en bleu est l'état d'affichage standard. Les états en vert sont les états de transition dans le menu principal. Les états en orange concernent les modifications d'heure. Et enfin, l'état en rouge est celui qui s'active lorsque l'alarme "sonne".

Au niveau des interruptions, il nous semblait évident que le comptage du temps qui passe est bien évidemment l'élément primordial durant toute l'exécution. Les interruptions de haute priorité sont donc utilisée uniquement pour l'overflow du Timer0. Pour avoir le plus de précision possible, nous avons décidé de compter par milliseconde. Le nombre d'incrémentations du Timer0 est d'environ 7000 pour compter cette unité de temps. Comme la taille de celui-ci est de 65536 (2^{16}), il n'est pas nécessaire d'utiliser le prescaler.

Le second type d'interruptions, celles de basse priorité, sont utilisées pour tout ce qui concerne les boutons. Nous avons considéré qu'il était plus important d'interrompre la routine qui va modifier un état ou l'heure au profit du comptage du temps que de les mettre à un même niveau de priorité.

Au niveau de l'affichage LCD, celui-ci se met à jour en permanence dans la boucle du programme principal. Nous évitons ainsi tout affichage dans une interruption (problème de durée de cette opération).