h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Laboratoire de systèmes numériques
F. Vannel, C. Abegg

Interruptions Vectorisées, General Purpose Input Output (GPIO) et TIMER sur la carte myLab2

Labo PRO
Semestre hiver 2016-2017

Introduction

Ce laboratoire va vous permettre d'utiliser la carte d'extension **myLab2** avec la carte processeur **LPCXpresso** ainsi que mettre en oeuvre les interruptions des GPIO et du TIMER. Vous utiliserez les LEDs, les interrupteurs, les boutons poussoirs ainsi que le joystick.

1 Complément d'informations sur l'utilisation des interruptions pour le micro-contrôleur LPC1769

1.1 Routines d'interruption

Les définitions des noms des routines d'interruptions se trouvent dans le fichier

`cr_startup_lpc175x_6x.c`. Libre à vous de renommer le nom de ces routines si vous le souhaitez. Pour rappel les interruptions du GPIO sont partagées avec EINT3.

Il convient de noter que dans le code de la routine d'interruption, il est nécessaire de terminer celle-ci par une quittance.

- Quittance de la source d'interruption auprès du périphérique. Plusieurs périphériques nécessitent de quittancer manuellement la source de l'interruption (par exemple, les interruptions externes, le TIMER), d'autre le feront automatiquement (par exemple, le convertisseur A/D). Pour savoir s'il est nécessaire ou non de quittancer, il faut pour cela se référer à la documentation technique du micro-contrôleur. Pour le GPIO, le registre `IOxIntClr` permet de réaliser cette tâche.

En cas de manquement à cette règle, la conséquence fâcheuse sera que votre routine d'interruption sera immédiatement rappelée, alors que ceci n'aurait pas du avoir lieu. Le déroulement de votre programme sera faux et vous ne pourrez plus jamais retourner dans le programme principal.

1.2 Initialisation des interruptions

Pour utiliser les interruptions, il faudra initialiser le gestionnaire d'interruption. Pour ce faire, il conviendra dans la procédure d'initialisation, de faire plusieurs actions :

1. Déclarer les périphériques à même de générer des interruptions et leur associer une priorité. Le registre `ISER0` permet d'effectuer ceci. Pour la gestion des priorités, ceci pourra être omis et nous utiliserons les priorités de bases.
2. Assigner le nom de la routine d'interruption à appeler au vecteur correspondant. Ceci vous est déjà fait au début du fichier `cr_startup_lpc175x_6x.c`.
3. Activer les interruptions pour chaque périphériques. Par exemple avec le registre du GPIO `IOxIntEnX`

2 Exercices Interruptions GPIO

Exercice 1: Compteur 8 bits manuel

En reprenant votre travail du laboratoire précédent, modifiez le compteur 8 bits manuel afin d'incrémenter (pression sur bouton A) ou décrémenter (pression sur bouton B) sa valeur via une interruption. Vous devez définir l'interruption sur l'appui des boutons A et B. Dans un premier temps, écrivez la fonction d'interruption pour incrémenter le compteur. Une fois celle-ci validée, complétez votre code pour décrémenter le compteur.

Rappel : Structurez votre code par l'utilisation de fonctions et librairies. N'oubliez pas au début de chaque exercice de bien définir les connexions entre la carte **myLab2** et la carte **LPCXpresso**. Une bonne solution consiste à créer une procédure `init_GPIO()` dans la librairie de la carte **myLab2**. La fonction `init_Interruption()` peut aussi être créée. Ces fonctions seront appelées avant la boucle `while(1)`. Utilisez les librairies déjà créées pour contrôler l'affichage.

Il est suggéré de garder l'affichage des LEDs dans la boucle `while(1)` et de ne modifier que la valeur du compteur dans la routine d'interruption.

IMPORTANT : La plupart des registres que vous allez vouloir utiliser dans ce laboratoire ont été déclarés pour vous. Si celà n'est pas fait, à vous de les rajouter dans le fichier `config_LPC1769.h` situé dans votre projet.

3 Exercices Interruptions TIMER

Exercice 2: Chenillard simple avec TIMER

Nous souhaitons, à l'aide du périphérique TIMER0, réaliser un chenillard lumineux sur 8 LEDs. Le chenillard, similaire à celui déjà réalisé, doit allumer au démarrage la LED0 de la carte myLab2. Toute les 600 ms, la LED allumée doit être décalée d'une position vers la gauche. Dès que la dernière LED (LED7) a été allumée, la séquence recommence à la première (LED0). La base de temps pour le **TIMER** devra être de **1 ms**.

Note :

$f_{osc} = 100\,MHz$

$PCLK = \frac{f_{osc}}{4}$

Marche à suivre :

1. Créer la fonction d'initialisation du périphérique TIMER.
   (a) Régler le PRESCALER du TIMER avec la valeur 25000. Ceci permettra d'avoir une base de temps de 1 ms. A savoir $\frac{25000}{PCLK} = 1\,ms$
   (b) Forcer le mode compteur utilisant la base de temps.
   (c) Choisir un MATCH register et l'activer pour qu'il génère une interruption à chaque égalité avec le compteur.
   (d) Régler la valeur du MATCH register pour générer l'interruption après le temps demandé dans l'énoncé. (ceci correspond à la première égalité entre le MATCH register et le compteur du TIMER)
   (e) Mettre en route le TIMER.
2. Compléter la routine d'interruption du `TIMER0` pour, dans un premier temps, contrôler uniquement le chenillard.
   (a) Modifiez et mémorisez l'état du chenillard dans une variable globale.
   (b) Modifier la valeur du MATCH register pour planifier la prochaine interruption du TIMER selon l'énoncé
   (c) Quittancer la source de l'interruption, à savoir le MATCH qui a généré l'appel à cette routine d'interruption.
3. Hormis les initialisations du TIMER et le contenu de la routine d'interruption, le code de l'exercice situé dans la boucle infinie du `main` doit seulement contrôler l'état des 8 LEDs de la carte **myLab2** en fonction de la variable globale modifiée dans la routine d'interruption du TIMER.

Exercice 3: Contrôle d'un chenillard

Créez un chenillard pouvant être piloté par le joystick de la façon suivante :

- Lorsque le bouton central est pressé, la valeur des 8 interrupteurs est lue et chargée dans le chenillard lumineux.
- En déplaçant le joystick sur la gauche le sens de déplacement du chenillard sera dirigé de la droite vers la gauche.
- En déplaçant le joystick sur la droite le sens de déplacement du chenillard sera dirigé de la gauche vers la droite.
- En déplaçant le joystick vers le haut, la vitesse du chenillard est multipliée par 2.
- En déplaçant le joystick vers le bas, la vitesse du chenillard est divisée par 2.

Lorsque le joystick n'est pas manipulé le chenillard doit garder le dernier mode sélectionné, à savoir sens de déplacement et vitesse.

Votre chenillard devra être circulaire, à savoir que lorsque la LED de l'extrémité gauche est allumée et que le sens de déplacement est de la droite vers la gauche, alors la LED de l'extrémité droite devra s'allumer à la prochaine itération. Pour l'autre sens, le principe est similaire.

Au démarrage, libre à vous de choisir un sens de déplacement, une vitesse et un motif pré-allumé.

La vitesse du chenillard doit être gérée par un timer. Le joystick ne disposant pas d'interruptions GPIO, vous devrez utiliser un second timer afin d'évaluer l'état du joystick toutes les 10 ms.

Il vous est conseillé de réaliser cet exercice de façon incrémentale, à savoir commencez par gérer le bouton central, puis le bouton gauche et droite, et à la fin gérez la vitesse avec les boutons haut et bas. Structurez votre code et complétez vos librairies si nécessaire.

Exercice 4: Contrôle d'un chenillard via Click et double click

Vous devez implémenter un système capable de discriminer un click d'un double-click. Un click est défini comme l'appui une seule fois sur un bouton. Un double-click est défini comme l'appui sur le bouton deux fois consécutivement avec une séparation de moins de 500 ms. Si le temps entre les deux appuis sur la bouton est supérieur à cette valeur, la fonctionnalité n'est pas identifiée comme un double-click, mais comme deux clicks simples qui se sont produits deux fois consécutives.

Le résultat de cette discrimination permettra de piloter le chenillard ainsi :

- A chaque click sur le bouton A, le chenillard doit doubler sa vitesse.
- A chaque double-click sur le bouton A, le chenillard doit diviser sa vitesse par deux.
- A chaque click sur le bouton B, le chenillard devra grandir d'une LED (maximum 7).
- A chaque double-click sur le bouton B, le chenillard devra rétrécir d'une LED (minimum 1).

Pour la détection du double-click, utilisez le TIMER1 avec une base de temps adéquate. Vous devez utiliser les interruptions GPIO pour détecter les changements d'états sur les boutons A et B. Les boutons A et B n'étant pas connectés sur des entrées capture, vous devrez utiliser un match register pour discriminer le click du double-click.