Microprocesseur

TP2 : PIT / Interruptions

Dans ce TP, nous allons nous intéresser à une meilleure maîtrise du temps et à l'utilisation des interruptions.

1) Le module PIT

Le module PIT (Periodic Interval Timer) est un compteur matériel qui permet de compter le nombre de cycles d'horloge écoulés pour fournir une fréquence précise. Il dépend des périphériques systèmes et reçoit l'horloge en permanence. Il n''est donc pas nécessaire d'intervenir sur le PMC pour s'en servir.

Vous vous référencerez directement à la datasheet et aux fichiers de déclaration pour exploiter ce module (p77). Quelque chose (que vous devrez préciser) se produit dès que ce compteur atteint ou dépasse la valeur maximale qui a été entrée à l'étape de configuration. Le compteur repart alors immédiatement de zéro pour garder une bonne périodicité du système. L'ensemble des registres qui permettent de gérer ce compteur sont présentés à partir de la page 80.

a) to read, or not to read...

Cette première partie consiste à extraire les informations de la documentation pour pouvoir se servir du module PIT par la suite.
Quel champ de quel registre représente le compteur périodique ?
Comment définir la valeur maximale jusqu'à laquelle le compteur va s'incrémenter ?
Trois phénomènes se produisent sur les registres lorsque le compteur atteint ou dépasse cette valeur maximale. Lesquels ?
Soit F la fréquence des évènements produits par le module PIT. Donnez la relation entre F, MCK et PIV (valeur contenue dans le champ PIV du registre PIMR).
Quelle est la différence entre les registres PIIR et PIVR ?

b) utilisation en scrutation

Reprenez le projet vierge utilisé au TP1 pour cette partie Ajoutez la configuration du PIT et modifiez la fonction d'attente pour qu'elle dure précisément 0,1s. (garanti par algorithme, non par mesure approximative).
Donnez l'algorithme et surtout la configuration du PIT.

Passez à une période de 1s. Pensez à vérifiez (chronométrer) le comportement que vous obtenez en pratique. Pourquoi devez-vous modifier votre algorithme ?
Décrivez les modifications que vous avez apportées.

2) Utilisation en interruptions

La partie la plus intéressante du PIT, est de s'en servir pour déclencher une fonction d'interruption

périodique. Pour contrôler cette interruption, il faut cependant configurer le module AIC (Advanced Interrupt Controler). Contrairement aux PICs, les microcontrôleurs AT91 peuvent gérer plusieurs fonctions d'interruption différentes. La structure liée à chacune de ces fonctions est appelée *vecteur* (votre cours vous donnera plus de détails sur le fonctionnement des interruptions en général). La gestion de ces vecteurs est le rôle du module AIC.

a) méthode d'écriture d'une fonction d'interruption.

L'écriture du code exécuté par une interruption se fait sous la forme d'une fonction C standard, mais avec quelques restrictions:

- une interruption est lancée par un événement extérieur au processeur (périphérique ou intervention de l'utilisateur) une fonction d'interruption ne reçoit donc aucun argument et ne renvoie aucun résultat. Elle doit prendre ses arguments dans des variables globales, et renvoyer des résultats par le même moyen. Sa déclaration est donc *void fonction (void)*.
- Une interruption prend la main sur le processeur et bloque le programme principal : elle doit donc être **rapide**. Il faut limiter au maximum l'usage de boucles et **ne jamais** faire d'attente d'aucune sorte. Lorsque l'attente d'un évènement est nécessaire, la bonne méthode consiste à s'assurer qu'une autre interruption se déclenchera sur cet événement et rendre la main. L'évènement sera traité dans cette autre interruption.

b) Configuration du module AIC

La configuration du module d'interruptions se fait via son adresse de base AT91C_BASE_AIC. Le vecteur d'interruption associé à chaque module est égal à son identifiant (note 2, page 164). Le module PIT fait partie des composants liés au système, donc son vecteur d'interruption correspond à l'identifiant AT91C_ID_SYS.

Quels sont les deux registres du module AIC qui permettent d'autoriser et d'inhiber les interruptions vecteur par vecteur ? (la liste des registres est donnée page 164, ne pas confondre *autoriser* une interruption et *déclencher* une interruption).
Une bonne partie des interruptions est déjà gérée dans le fichier assembleur Cstartup.s79, il ne sera donc pas nécessaire de maitriser entièrement le module AIC. Seuls les deux registres précédents ainsi que les tableaux de registres SMR[32] et SVR[32] seront utiles dans le cadre de ces TPs. Pour un vecteur *i* donné, quel est le rôle de SMR[*i*] et SVR[*i*] ?

Pour configurer un vecteur d'interruption, il faut écrire dans un registre (que vous aurez à déterminer) l'adresse de la fonction à exécuter lors des déclenchements (la fonction d'interruption). Pour cela, la syntaxe C est très pratique. En effet, il s'agit du nom de la fonction, sans les parenthèses. Ainsi, pour la fonction *void fonction_interruption(void)*, cette adresse sera *fonction_interruption*. Par contre, *fonction_interruption* est de type « pointeur sur fonction », qu'il faudra convertir en entier non signé pour éviter un avertissement ou une erreur lors de la compilation. L'affectation pour le vecteur *i*, ressemble donc à la ligne suivante :

AT91C_BASE_AIC->*registre_qui_va_bien*[*i*] = (unsigned int)*fonction_interruption*;

A chaque fois que c'est possible, il est conseillé de configurer les interruptions en déclenchement sur niveau haut plutôt que sur front montant. Cela signifie que tant que le bit qui génère la demande d'interruption est actif, le processeur déclenchera l'interruption.

c) Mise en œuvre

Pour vérifier le bon fonctionnement des interruptions, nous vous proposons d'effectuer simultanément deux types d'animations.

- La première, en scrutation, avec une boucle infinie et une fonction d'attente basée sur une boucle *for*.
- La deuxième gérée en interruptions sur le PIT.

Ouvrez le projet *TP2_IT* sur la page dédiée aux TPs. Ce projet contient déjà une animation (LED1 clignote rapidement sans utiliser le PIT). Ajoutez la configuration du PIT et des interruptions pour faire clignoter LED2 de façon indépendante (0,25s allumée, puis 0,25s éteinte). Vous détaillerez les configurations des modules PIO, PIT et AIC ainsi que l'algorithme de la fonction d'interruption.
Pour valider ce programme, les deux animations doivent être visibles en parallèle, sans que le code de chacune ne se préoccupe de l'autre.

3) Passage d'évènements IT/scrutation

a) dans un sens...

Ajoutez la ligne suivante à la fin de la fonction d'attente :

```
AT91C_BASE_PMC->PMC_SCDR = 1;
```

(la ligne est déjà présente en commentaire)

Retrouvez dans la datasheet ce que produit cette affectation.

Qu'observez-vous par rapport au comportement des LEDs ? Expliquez ce comportement.

b) … puis dans l'autre

Exceptionnellement, configurez l'interruption du PIT en déclenchement sur front montant.
Retirez la ligne de la question précédente et ajoutez la ligne suivante à la fin de la fonction d'attente :

```
AT91C_BASE_AIC->AIC_ISCR =  1 <<  AT91C_ID_SYS;
```

Expliquez l'utilité de cette ligne. Pourquoi passer en déclenchement sur front ?
Que se passe-t-il ? Proposez une explication.