ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

MicroProcesseur - Microcontrôleur
seconde année

Enoncés de TD

Exercice 0 - Préambule :

1. Expliquez ce qu'est l'attente active. Quel intérêt y a-t-il à gérer la conversion par interruption plutôt que par attente active ?

2. Comme chaque demande d'interruption peut provenir de plusieurs interfaces, quel est l'élément qui gère ces demandes et les transmet au processeur ?

3. Comment peut-on interdire la prise en compte de nouvelles interruptions pendant l'exécution d'une routine de service d'interruption ?

4. Que se passe-t-il si une demande d'interruption est faite alors que le processeur a masqué les interruptions ?

5. Si une interruption est traité alors que le programme principal est exécuté en mode utilisateur, quelle sera la pile utilisée durant la routine de service de l'interruption ? Pourquoi ?

Exercice 1 – Priorités :

1. Quel registre numéro x des registres NVIC_IPLR faut-il utiliser pour configurer la priorité des interruption du convertisseur analogique-numérique ADC1 ?

2. Donnez la valeur de ce registre qui permet d'avoir un niveau de priorité de 64 ?

3. Quel registre permet d'autoriser les interruptions du

convertisseur analogique-numérique ADC1?

4. Donnez la valeur de ce registre qui autorise les interruptions provenant du convertisseur analogique numérique ADC1.

5. Le registre BASEPRI contient successivement les valeurs 64, 0 et 128. En comparant ces valeurs à celle du registre NVIC_ICPRx précédemment définie, dites pour chacune des trois valeurs, si les interruptions du convertisseur seront ou non prises immédiatement en compte.

6. Le registre PRIMASK contient la valeur 1, les interruptions du convertisseurs ADC1 peuvent-elle traitées ?

Exercice 2 – Convertisseur Analogique-Numérique :

On veut utiliser pour numériser les signaux produits par une instrument sur 8 voies reliés aux canaux 0 à 7 du convertisseur ADC1 au STM32. Les conversions seront faites sur des canaux réguliers.

Dans chacune des questions suivantes on écrira~:

une procédure *init_adc()* d'initialisation du convertisseur permettant son utilisation par interruption, sa calibration à la mise sous tension du convertisseur et une base de temps d'échantillonage de 1,5 cycles.

une procédure de service d'interruption *isr_adc()* appelée lorsque le convertisseur produit une interruption et dont les fonctions sont détaillées dans chaque question

un programme principal *main()* qui lance le convertisseur en utilisant ces fonctions.

1. Le convertisseur effectue une unique conversion du canal 1 et le résultat est copié dans la variable globale Converted_Data.

2. Le convertisseur effectue en continu la conversion du canal 0 et le résultat est copié dans la variable globale Converted_Data.

3. Le convertisseur effectue une unique conversion des canaux allant du 7 au 0 et le résultat est copié dans la variable

globale Converted_Channels de dimension 8.

4. Le convertisseur effectue en continu la conversion des canaux allant du 7 au 0 et le résultat est copié dans la variable globale Converted_Channels de dimension 8.

Exercice 3 – Compteur-Timer :

On dispose d'un STM32 avec un quartz 72 Mhz qu'on veut utiliser pour numériser périodiquement les signaux produits par un instrument sur 8 voies reliés aux canaux 0 à 7 du convertisseur intégré au STM32. Les conversions seront faites sur des canaux réguliers.

Dans chacune des questions suivantes on écrira :

une procédure *init_adc()* d'initialisation du convertisseur permettant son utilisation par interruption, sa calibration à la mise sous tension du convertisseur et une base de temps d'échantillonage de 1,5 cycles.

une procédure de service d'interruption *isr_adc()* appelée lorsque le convertisseur produit une interruption et dont les fonctions sont détaillées dans chaque question

une procédure *init_t2()* d'initialisation du compteur-timer T2 permettant son utilisation par interruption

une procédure de service d'interruption *isr_t2()* du compteur-timer T2 pour générer les temporisations nécessaires

un programme principal *main()* qui lance le convertisseur en utilisant ces fonctions.

1. Le convertisseur effectue une unique conversion du canal 1 après une attente de 1 ms, et le résultat est copié dans la variable globale Converted_Data.

2. Le convertisseur effectue la conversion du canal 2 toutes les 10 ms, et le résultat est copié dans la variable globale Converted_Data.

3. Le convertisseur effectue toutes les 5 ms la conversion des canaux allant du 7 au 0 et le résultat est copié dans la

variable globale Converted_Channels de dimension 8.