Microprocesseur

TP4 : UART – Réseau d'interruptions

Ce TP a pour objectif de concevoir un système communicant proposant des fonctions du niveau d'un système d'exploitation. Le support sera la liaison série UART. Le principe est d'utiliser une mémoire tampon pour l'envoi et la réception. L'envoi/réception successif des caractères utilise les interruptions pour permettre au programme principal d'effectuer d'autres tâches.

1) Gestion en interruptions

Il est possible de déclencher une interruption sur les bits de contrôle d'envoi et de réception de caractères. Comme lors du TP2, chaque module déclenche le vecteur d'interruption qui correspond à son numéro d'identifiant.

a)Réception en interruption.

Déclarez une chaine de 5 caractères et un entier *nb_ch* en variable globale. *nb_ch* représente le nombre de caractères reçus, la partie initialisation du programme le mettra à zéro. Écrivez une fonction d'interruption qui écrit le dernier caractère reçu à la suite de la chaine. Si on est au cinquième caractère, la fonction laisse *nb_ch* à 5. Lors de la réception du 6ème caractère, la fonction l'écrit en position 0 et remet *nb_ch* à 1.
Configurez le système pour que cette fonction se déclenche lorsqu'un caractère est reçu. Quel est, dans le module USART, le registre qui permet de choisir les bits de statut à l'origine des demandes d'interruptions ?
Pour éviter de perdre des caractères, le programme principal fera un envoi de toute la chaine dès que celle-ci sera pleine en faisant une scrutation sur *nb_ch.*

Au niveau du comportement, on doit donc ne rien voir quand on écrit 4 caractères, puis lors de l'envoi du 5ème caractère, l'ensemble du message apparaît. Ce comportement est cyclique et doit donc recommencer.

b)Réception/Envoi en interruption.

Nous allons maintenant chercher le même comportement en faisant des envois en interruption. Est-il possible d'utiliser une fonction différente pour l'envoi et la réception ?

- si oui : comment ?
- si non : pourquoi ?

L'envoi du premier caractère est provoqué par la réception du 5e caractère, et sera donc envoyé dans le même appel. Comment peut-on provoquer une interruption pour l'envoi des caractères suivants ? Comment terminer ces déclenchements lorsque la chaine complètement envoyée ?

Réalisez le système.

2) Aspect système

L'intérêt de la réception en interruption n'est pas de traiter les données à la volée, mais de mémoriser les données qui arrivent pendant que le programme principal ne peut pas les traiter. De même, l'envoi en interruption permet au programme principal de ne pas s'inquiéter de quand les différents caractères seront transmis. Pour cela, il y a besoin de créer une mémoire tampon (buffer) selon le principe d'une file d'attente FIFO (First In First Out).

Implémentez des files d'attente pour l'envoi et la réception. Vous pouvez par exemple utiliser la méthode des FIFOs circulaires basées sur un tableau de taille fixe. Associez ces structures à des fonction de lecture et d'écriture. Écrivez la fonction d'interruption qui gère la liaison série, ainsi que les fonctions d'envoi et de réception de caractère. Ces fonctions feront appel aux fonctions d'écriture et de lecture sur les FIFOs.

Vous utiliserez cette structure dans un programme qui effectue un écho de ce qui est envoyé, qui affiche « BONJOUR » toutes les secondes (exactement) ET qui reprend en parallèle la question 3c) du premier TP.

- Pour cela : la partie 3c) du TP1 sera faite en scrutation.
- La temporisation de 1s sera faite en interruption sur le PIT pour ne pas être perturbée par les anti-rebonds du joystick.
- La gestion de la liaison série sera effectuée en utilisant les structures FIFO.

Comment gérer le fait que les interruptions ET le programme principal puissent modifier la même variable (structure FIFO) ?

Implémentation d'une FIFO circulaire

Pour implémenter une FIFO circulaire, il y a besoin de deux indices (un en lecture et un en écriture) ainsi que d'un tableau de taille fixe. On notera les deux indices *iw* et *ir* (par exemple). *iw* est le numéro de la case du tableau dans lequel il faut écrire le prochain élément. *ir* est le numéro de la case du tableau dans lequel il faut lire l'élément le plus ancien (First in). A chaque écriture il faut incrémenter *iw*, de même, à chaque lecture, il faut incrémenter *ir*. Ces incréments se font modulo la taille du tableau. Si $ir = iw$ alors, la file d'attente est vide. Si $ir+1 = iw$ modulo la taille du tableau, alors la file d'attente est pleine.