TR54 – Partiel

Durée prévue : 1h, Documents autorisés

On considère une interface permettant de réceptionner des données issues d'un périphérique. Cette interface est capable de générer une requête d'interruption niveau lorsque des données sont réceptionnées.

On propose d'écrire une partie du pilote de réception de donnée.

Fonctionnement de l'interface :
lorsqu'une donnée arrive, celle-ci est stocké dans une pile FIFO (First In First Out) de l'interface.
les bits 0 à 3 du registre d'état appelé *SR* rendent compte du nombre de données présentes dans la pile FIFO de l'interface. Ce champs est accessible en lecture/écriture afin de permettre de vider artificiellement la FIFO ('flush')
L'interface génère une requête d'interruption tant que le compteur FIFO n'est pas à 0. Les interruptions dues à la FIFO non vide peuvent être masquées par la mise à 0 du bit 4 du registre *SR* et autorisées par la mise à 1 de ce même bit.
Les données placées dans la FIFO sont accessibles par lecture d'un registre appelé *DATA*. Les données sont de type octet.

Le système d'exploitation est eCos. La routine isr et la routine dsr ont été enregistrées à l'initialisation. De même l'interface a été configurée à l'initialisation.

Adresses des registres
DATA : 0xEFFFFFF0
SR : 0xEFFFFFF4
vecteur d'interruption : 4

Routine de lecture
On considère ici que l'interface est initialisée par une routine d'ouverture dont on ne s'occupe pas ici. Les interruptions de l'interface sont bloquées par le système et peuvent être débloquées (fonctions mask(vector), unmask(vector)), elles sont par ailleurs également bloquées au niveau de l'interface (bit4 du registre *SR* à 1).

Le prototype de la fonction appelée par l'application utilisateur est la suivante
*unsigned int read(unsigned int len, char * data)*
avec
len : nombre d'octets demandé
data : adresses où doivent être stockées les données
la fonction renvoie le nombre données effectivement copiée

La politique choisie pour le fonctionnement de cette fonction est un fonctionnement bloquant. Cela signifie que lorsqu'une application émet une requête d'E/S en appelant la fonction read, elle reste bloquée (attente passive) jusqu'à ce que le *len* données demandée soient effectivement arrivées.

Routines de traitement des interruptions
Les routines isr et dsr ont été enregistrées auprès du système. On rappelle les prototypes de ces deux fonctions :
cyg_uint32 isr(cyg_vector_t vector, cyg_addrword_t data)
void dsr(cyg_vector_t vector, cyg_ucount32 count, cyg_addrword_t data)
vector : numéro du vecteur d'interruption, data : paramètre utilisateur déclaré à l'initialisation qui ne sera pas utilisé ici, count : nombre de fois ou la dsr a été appelée avant d'être lancée.

Travail demandé :
Expliquez brièvement comment la réception de donnée est gérée à l'aide des 3 fonctions précédente : *read*, *isr* et *dsr*
Donnez le code C expliqué des 3 routines *read, isr* et *dsr* permettant à l'application de récupérer les données demandées.