TP n°4 bis
Liaison série asynchrone

(deuxième partie)

Objectifs :
Création des fonctions d'émission de caractère et de chaîne de caractères.
Réception série asynchrone avec interruption.
Algorithme d'un interpréteur de commande simplifié.

1/ Création d'un jeux de fonctions pour l'émission série

- Ecrire une fonction `serial_init()` qui initialise le module USART du pic en mode série asynchrone pour la transmission et la réception 8 bits à 9600 Baud. Cette fonction n'autorisera pas les interruptions.
- Ecrire une fonction `serial_putchar()` qui émet un caractère passé en paramètre sur la liaison série.
- Ecrire une fonction `serial_puts()` qui émet une chaîne de caractères passé en paramètre sur la liaison série. De manière à pouvoir accepter des chaînes constantes pour écrire par exemple `serial_puts("Bonjour\r");` , le prototype de cette fonction sera :

```
void serial_puts(const char s[]);
```

Testez vos fonctions avec un programme main() qui les appelle pour émettre une phrase lorsqu'on appuie sur le bouton S3.

NB: Hyperterminal saute une ligne et revient à la première position sur réception du caractère \r (CR) (et pas \n qui correspond ici à LF (0X0A)).

2/ Création d'un interpréteur de commande simplifié

Mise au point de la réception série en interruption *:*

- Ajouter au programme précédent la réception d'une chaîne de caractères dans un programme d'interruption avec indication au programme principal qu'une commande complète est reçue (réception d'un CR (0x0D)) par la mise à 1 d'un variable globale. (Utilisez l'algorithme étudié en cours). Pour l'instant , dans le programme principal on se limitera à l'inversion de l'état d'une LED lorsque la variable globale passe à 1.

Pour mettre au point ce programme, on ne peut pas placer de point d'arrêt dans le programme d'interruption (c'est une limitation de l'icd2). Vous pouvez par contre :
- Allumer des LED pour indiquer un passage dans le programme d'IT.
- Observer l'évolution du tableau contenant les caractères reçus dans une "Watch Window".
- Réémettre la commande reçue dans le programme principal avec la fonction serial_puts() .

Attention à la configuration de l'émulateur de terminal : il faut de la touche "entrée" provoque l'émission du caractère CR (0x0D) uniquement (et pas CR suivi de LF (0x0A)).

TP n°4 bis
Liaison série asynchrone

(deuxième partie)

LA PARTIE PRECEDENTE DOIT FONCTIONNER PARFAITEMENT AVANT DE CONTINUER. EN PARTICULIER LE TABLEAU RECU DOIT CORRESPONDRE EXACTEMENT AUX CARACTERES EMIS ET LA DETECTION DE COMMANDE COMPLETE DOIT FONCTIONNER.

Analyse et dialogue simplifié :

On souhaite implanter les commandes suivantes :
`ON` allume RB1, `OFF` éteint RB1, `?` ou `HELP` affiche l'aide, `V` affiche la version.

Le dialogue peut ressembler à l'exemple ci dessous :

```
Connected to picdem Tp4b v1.0
>?
Usage :
     ON  : allume RB1
     OFF : éteint RB1
     V : affiche la version
     HELP ou ? : aide
>ON
>OFF
>V
Tp4b v1.0
>XYZ
?
>
```

- Implanter ces commandes dans le programme principal.
(On pourra utiliser la fonction strcmp qui compare deux chaîne de caractères)

Commandes avec paramètres simples

Ajouter une commande `LEDS` qui prend un paramètre 0 ou 1.
`LEDS 1` : allume les quatre LED.
`LEDS 0` : éteint les quatre LED.

Commandes avec retour d'information variable

En utilisant les fonctions de la librairie biosdem et la fonction standard `sprintf()`, ajouter une commande `CONV` qui provoque la conversion de la tension sur RA0 et qui renvoie la valeur trouvé en décimal.
Exemple :

```
>CONV
234
```