Microprocesseur

TP5 : Communication UART

Dans ce TP, nous allons mettre en œuvre un support de communication entre la carte et l'ordinateur. Pour cela nous utiliserons le module UART (Universal Synchronous/Asynchronous Receiver Transmiter) mis en œuvre sur un port USB. Le port USB n'est pas compatible avec le protocole UART, nous communiquons avec un composant qui dialogue en USB avec le PC, et en UART avec le microcontrôleur. Ce composant est courrament appelé *convertisseur USB/UART* par abus de langage, concrètement il s'agit d'un périphérique de communication UART qui est ajouté au PC par le bus USB, tout comme nous pourrions connecter un disque dur ou une carte Wi-Fi. Vu du PC, il ne s'agit donc que d'un port UART. La simplicité de ce protocole permettra de ne s'attarder que sur la gestion logicielle de la communication. Cette fois, vous aurez besoin de deux câble USB : un pour programmer la carte, un autre pour la communication UART. Notez qu'il n'est pas nécessaire que ces deux câbles soit reliés au même ordinateur.

Côté logiciel, nous utiliserons l'utilitaire *screen* dont l'usage est assez basique :

```
screen <nom_du_port> <vitesse_de_transfert>
```

Sous UNIX, le nom du port est un fichier dans le dossier `/dev` .Le port fourni est vu comme un terminal connecté en USB, son nom sera donc `/dev/ttyUSBx` où *x* représente un entier. Normalement, un seul port sera présent. Si plusieurs ports sont présents, identifiez celui de la carte en vérifiant lequel apparaît/disparaît lorsque l'on connecte/déconnecte la carte.

La vitesse de communication est théoriquement libre, mais il est conseillé de garder les vitesses historiques utilisées (2400bps, 4800bps, 9600bps, 19200bps, 38400bps, 57600bps, 115200bps).

Du coté de la carte, il faut configurer la même vitesse de transfert, associé aux paramètres suivants :

- 1 bit de stop
- pas de parité
- 8 bits par mot
- pas de contrôle de flux (pas d'utilisation des broches CTS/RTS)
- Les autres paramètres doivent être laissés dans leur usage le plus basique.

Tout caractère frappé dans la zone de texte du terminal sera envoyé à la carte par la liaison, mais ne sera pas affiché. Tout caractère émis par le kit sera affiché dans cette même zone de texte.

Pour sortir de l'utilitaire screen, il faut utiliser la séquence *CTRL-a* puis *k*.

1) Transfert de données brutes

a) Configuration de l'UART et envoi basique.

Dans un premier temps, Partez d'un programme qui fait clignoter une LED, cela permet d'identifier si le programme fonctionne quand on n'observe rien sur la liaison. Configurez le périphérique UART5 qui permet les transferts de données avec le module USB. Ce dernier est connecté sur les broches F12 (émission : U5TX) et F13 (réception :U5RX). Remarque : il est nécessaire d'activer les

deux canaux de communication (Rx et Tx) dans le registre U5STA, les autres paramètres devraient vous être accessibles.

Envoyez le caractère 'A' régulièrement de la carte vers le PC (en cadence avec l'animation LED).

b) Envoi de texte (scrutation).

Cette fois-ci, nous allons envoyer des phrases complètes. Dans un premier programme, envoyez régulièrement *'Bonjour\n'*, Dans un second, envoyez régulièrement *'Bonjour le monde\n'*. Ces deux envois nécessitent des précautions très différentes. Pourquoi ?

c) Réception de texte .

Nous allons maintenant nous intéresser à la réception de texte : écrivez un programme qui affiche sur l'écran LCD le texte que la carte a reçu depuis le PC.

Les caractères que vous tapez dans le terminal ne sont pas visibles, car ils sont directement envoyés vers la carte. Pour les visualiser aussi sur le terminal, il faut renvoyer les caractères reçus. Modifiez votre programme pour obtenir ce comportement.

d) Un peu de fun (ou pas).

Pour bien illustrer le fait que ce qui s'affiche sur le terminal ne dépend QUE de ce qu'envoie la carte, vous pouvez vous *amuser* à modifier l'echo des caractères. Vous pouvez utiliser les exemples suivants, ou n'importe quel comportement à votre convenance :

- tous les caractères en envoi sont incrémentés de 1 ('A' devient 'B', 'B' devient 'C', *etc*)
- les 'e' sont changés en 'a' et inversement…
- la casse est modifiée aléatoirement ('x' devient 'X', mais pas tout le temps…)

e) Échange de données.

S'il vous reste un programme C qui fait *Hello World*, vous pouvez également utiliser la commande suivante : `./hello_world > /dev/ttyUSBx`
Cette commande redirige *stdout* vers la lisaison UART, donc vers la carte.

De façon générale, tout programme qui ouvre /dev/ttyUSBx en écriture pourra envoyer des données en effectuant un *fputc*, *fputs* ou *fprintf*. De même si le fichier est ouvert en lecture, les fonctions *fgetc*, *fgets* ou *fscanf* permettront de lire les données venant de la carte. Pour plus de détails, notamment modifier les paramètres de la liaison depuis le programme, consultez `<termios>` ou `<ioctl>` pour un programme en C, ou `pyserial` pour un programme python.