INFO2 : INFORMATIQUE EMBARQUEE
TP5 : Mise en œuvre de la liaison série asynchrone RS232

1 Objectifs

Malgré son âge avancé, la **liaison série asynchrone RS232** est encore très utilisée pour faire communiquer deux équipements entre eux. Par exemple, un PC émulant un **terminal série** et un microcontrôleur peuvent facilement dialoguer grâce à leurs liaisons RS232. Le PC peut ainsi récupérer des données stockées dans le microcontrôleur ou initialiser certains des paramètres de l'application gérée par le microcontrôleur. En outre, quand le microcontrôleur est muni d'un programme *bootloader*, on peut grâce à la liaison série actualiser son *firmware* sans nécessité d'un programmateur comme le PICkit3.

Remarque : L'**UART**[1] d'un microcontrôleur respecte rarement les niveaux électriques requis par la norme RS232. Un circuit adaptateur comme le MAX232 est donc souvent présent entre le PC et le microcontrôleur.

Dans ce TP, nous allons d'abord configurer l'UART du PIC18F4520.

Ensuite nous enverrons un caractère vers le terminal série. Comme prolongement, nous utiliserons des fonctions d'affichage évoluées (**`printf`**) afin d'envoyer des chaînes de caractères.

Par la suite, nous lirons un caractère en provenance du terminal série. En fonction de sa valeur nous déclencherons telle ou telle action.

Nous afficherons ensuite le résultat d'une conversion analogique-numérique sur le terminal série.

Enfin, nous encapsulerons les trames série dans des trames Bluetooth. Pour information, il est aussi possible d'encapsuler facilement des trames série dans des trames USB (grâce aux puces FTDI) mais cela ne sera pas étudié ici.

2 Projet TP5A : Envoi d'un caractère

2.1 Cahier des charges

L'UART du PIC (dont l'oscillateur est à 8 MHz) doit être configurée en émission / réception avec les paramètres suivants : « 19200, 8, N, 1 » (pas de contrôle de flux). Ces paramètres seront conservés pour toute la durée du TP.

Ensuite, le caractère 'A' doit être envoyé sur l'écran d'un PC faisant fonctionner un émulateur de terminal.

2.2 Validation du travail

- Vérifiez que la prise DB9 du PC et celle de la carte EasyPIC v7 sont reliées ensemble par un câble série.
- Côté PC, le logiciel Tera Term Pro permet d'émuler un terminal. Lancez-le et configurez-le comme il faut : Serial Port : COM1 puis Setup / Serial port….
- Ecrivez votre programme et testez-le.

3 Projet TP5B : Envoi d'une chaîne de caractères

3.1 Cahier des charges

Sur l'écran du terminal série, on doit voir écrit :

```
Première ligne de texte.
Deuxième ligne de texte.
Troisième ligne de texte.
```

On utilisera pour cela une fonction évoluée d'affichage dont le prototype est dans **`stdio.h`**. (Lire la suite avant de commencer !)

[1] Voir le cours pour les acronymes.

3.2 Préparation

Dans l'énoncé du TP4 à la partie intitulée « Présentation de fprintf() », nous avons vu que :

- La fonction de **stdio.h** qui permet d'envoyer une chaîne de caractères vers le flux standard de sortie **stdout** est **printf()**. (**fprintf** permettait d'envoyer vers **_H_USER** qui correspondait à un écran LCD.)
- le flux standard de sortie **stdout** pour un PIC18F (et bon nombre de microcontrôleurs) est son UART (en général la première s'il en a plusieurs).

Remarque : Pour information, **printf()** travaille en faisant des appels successifs à **_usart_putc()** tout comme au TP4 **fprintf()** travaillait en faisant des appels successifs à **_user_putc()**. Attention : A la différence de **_user_putc()**, **_usart_putc()** est déjà écrite, il ne faut pas le faire.

3.3 Validation du travail

- Effectuez la partie du TP4 intitulée : « Réglage du « modèle de mémoire », obligatoire dès que l'on utilise **printf**, **fprintf**, etc.
- Ecrivez votre programme et testez-le. Le retour à la ligne est-il géré ? Si non, rectifiez le problème.

4 Projet TP5C : Lecture d'un caractère et écho

4.1 Cahier des charges

Sur l'écran du terminal série, on doit voir écrit :

Bonjour, ici le terminal. J'attends votre saisie.

Ensuite, tout caractère frappé sur le clavier du terminal doit être reproduit sur l'écran du terminal, c'est ce qu'on appelle l'écho. (Une frappe sur la touche Entrée doit produire un vrai retour à la ligne.)

Ensuite, il faut aussi s'arranger pour que la frappe du caractère '0' allume la led connectée à RD0 (les autres sont éteintes), la frappe du caractère '1' allume la led connectée à RD1 (les autres sont éteintes), …, la frappe du caractère '7' allume la led connectée à RD7 (les autres sont éteintes). Les autres caractères ne déclenchent aucune action, ils sont juste « échoés ». Une structure **switch case** est conseillée.

4.2 Validation du travail

- Ecrivez votre programme et testez-le.

5 Projet TP5D : Affichage d'une tension

5.1 Cahier des charges

Toutes les 500 ms environ, on voit écrit sur l'écran du terminal série la valeur de la tension disponible sur le potentiomètre P1 connecté à RA3, suivi de l'unité de cette tension.

5.2 Préparation

Pour fabriquer une représentation sous forme de caractères d'un nombre réel, vous aurez besoin de la fonction **ftoa()** déjà présentée au TP4. Celle-ci est implémentée dans le fichier **INFO2_ftoa.c** et son prototype est dans **INFO2_xlcd.h** (disque Ressources en haut à droite du Bureau, répertoire **LCD**).

- Faites une copie de ces 2 fichiers dans votre répertoire de projet.
- Dans votre fichier C, après les **#include** du début, ajoutez : **#include "INFO2_xlcd.h"**.
- Déclarez **INFO2_ftoa.c** comme faisant partie des *Source Files* du projet : clic droit sur Source Files puis Add Existing Item….

5.3 Validation du travail

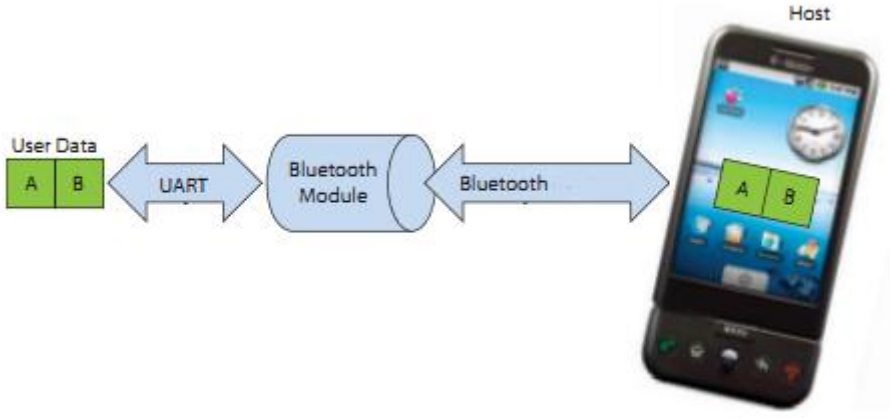
- Ecrivez votre programme et testez-le.

6 Envoi de l'information de tension vers un périphérique Bluetooth

6.1 Présentation

La liaison série asynchrone est plutôt ancienne, mais, comme vous venez de vous en rendre compte, elle est assez simple à prendre en main. Aujourd'hui, il y a cela dit de moins en moins de dispositifs mettant en œuvre une vraie RS232 (aux niveaux mécanique et électrique) mais son protocole d'échange des données (bit de *start*, etc.) est encore très utilisé.

En effet, à la manière de la figure suivante, il est possible d'encapsuler une trame série (venant d'une UART) dans des trames de protocoles beaucoup plus d'actualité, mais plus complexes, comme Bluetooth[2]. Pour le concepteur, tout ceci est transparent, il n'a absolument pas besoin de connaître la norme Bluetooth, seulement les fondamentaux de la liaison série asynchrone. En effet, parmi tous les profils que le Bluetooth peut faire fonctionner, il en est un qui s'appelle ***Serial Port Profile*** (SPP).



Nous n'allons rien changer au programme du projet précédent. Simplement, au lieu de connecter RC6 (Tx) et RC7 (Rx) du PIC à l'ensemble MAX232-prise DB9, nous allons les relier aux pattes Rx et Tx d'un module Bluetooth qui se chargera :

- A l'émission : de la mise en forme des trames série pour le Bluetooth.
- A la réception : du décodage des informations Bluetooth pour en fabriquer des trames série.

Un smartphone sera l'autre extrémité Bluetooth. Sur celui-ci on fera fonctionner un terminal Bluetooth qui effectue le travail inverse du module Bluetooth.

Le module Bluetooth utilisé s'appelle « Easy BlueTooth ». Il est fabriqué par MikroElektronika, le fabricant de la carte d'évaluation EasyPIC v7. L'essentiel de ce module est constitué du composant RN41 de chez Microchip (moins de 20 €). Au niveau connectique, le module est équipé d'un connecteur lui permettant d'être enfiché facilement sur un connecteur du port C de la carte d'évaluation EasyPIC v7.

Remarque : On trouve des modules Bluetooth comme le HC-06 à 4 € sur eBay ou Amazon.

Le module est configuré ici pour être en « esclave », c'est-à-dire qu'il ne peut prendre l'initiative d'établir une liaison. C'est le smartphone qui sera « maître », c'est-à-dire que c'est lui qui va essayer de se connecter au module.

6.2 Procédure

6.2.1 Côté PIC

- Vérifiez que c'est le programme d'affichage de la tension dans un terminal qui est présent sur le PIC.
- Eteignez la carte d'évaluation EasyPic v7.
- Débranchez le câble série.
- Vérifiez sur le module « Easy BlueTooth » :
  - Ensemble de microswitches « SW1 » : microswitch 1 (« P7 ») vers « RX » et microswitch 4 (« P6 ») vers « TX » et rien d'autre ! Cela est adapté à notre PIC pour lesquelles RC6 et RC7 sont les broches de l'UART.
  - Connecteur J1 « SUPPLY SELECT » : Cavalier à cheval sur « 5V » et la broche du milieu.
  - Connecteur « Pull Up » : Un cavalier seulement sur « PI03 ». Le fait de ne pas avoir de cavalier sur « PI07 » permet au module d'avoir une liaison à 19200 bds, ce qui était le cas du PIC dans le projet d'affichage de tension.

[2] L'encapsulation par USB est aussi possible. C'est ce que font les puces FTDI.

Ainsi les deux communiqueront correctement. S'il y avait un cavalier sur « PI07 », c'est une liaison à 9600 bds, ce qui n'est pas intéressant pour nous.

- Placez le module sur le connecteur « CN10 » du port C de la carte d'évaluation EasyPIC v7. Ce connecteur est du côté droit de la carte.
- Au niveau de la carte d'évaluation EasyPIC v7 :
  - Cavaliers J3 et J4 : Retirés totalement. Cela empêche RC6 et RC7 d'aller vers le MAX232 et donc évite des conflits avec le module « Easy BlueTooth ».
- Appelez l'enseignant pour vérifier tous les points précédents avant de rallumer la carte d'évaluation EasyPIC v7.

6.2.2 Côté smartphone

- Procurez-vous un smartphone doté du Bluetooth et équipé de l'application Bluetooth Terminal. Dans les Paramètres Bluetooth du smartphone, faites un scan pour voir le module INFO2-BT-xx (xx est écrit au marqueur sur le module) dans Appareils Disponibles. (Parfois c'est l'adresse MAC du module qui est affichée plutôt que son nom.)
- Cliquez sur ce nom pour l'appairer à votre smartphone. Le code de sécurité à saisir est **mikroe**.
- Quand l'appairage a fonctionné, lancez l'application Bluetooth Terminal. Connectez-vous au module Bluetooth. Si tout va bien, vous devriez voir la tension qui s'affiche sur l'écran du smartphone !
- Reprenez le projet TP5C pour voir s'il fonctionne avec votre téléphone.

7 Evolutions possibles de ce TP

- Tester la communication Bluetooth dans l'autre sens : Le smartphone envoie des informations au PIC, par exemple un numéro de LED du port D à allumer.
- Ecrire soi-même puis tester les fonctions que l'on trouve dans **usart.h** :
  - Emission : **void WriteUSART(char data), char BusyUSART(void)**
  - Réception : **char DataRdyUSART(void), char ReadUSART(void), void getsUSART(char* buffer, unsigned char len)**