Raspberry Pi 3 Liaison Série sur PI3

1 Présentation

Le BCM2837 sur le Raspberry Pi 3 possède 2 UART. (**ttyAMA0** et **ttyS0**)

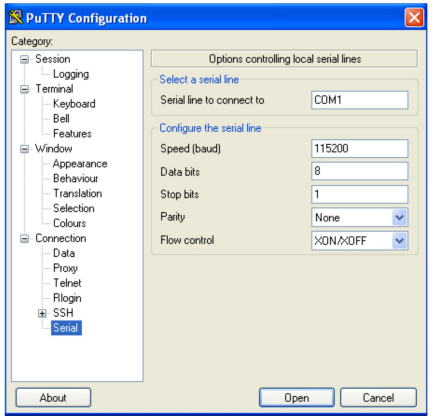
Le **ttyAMA0** se réfère maintenant au port série qui est connecté au bluetooth et **ttyS0** se réfère au miniUART.

Malheureusement, il y a un certain nombre de conséquences :

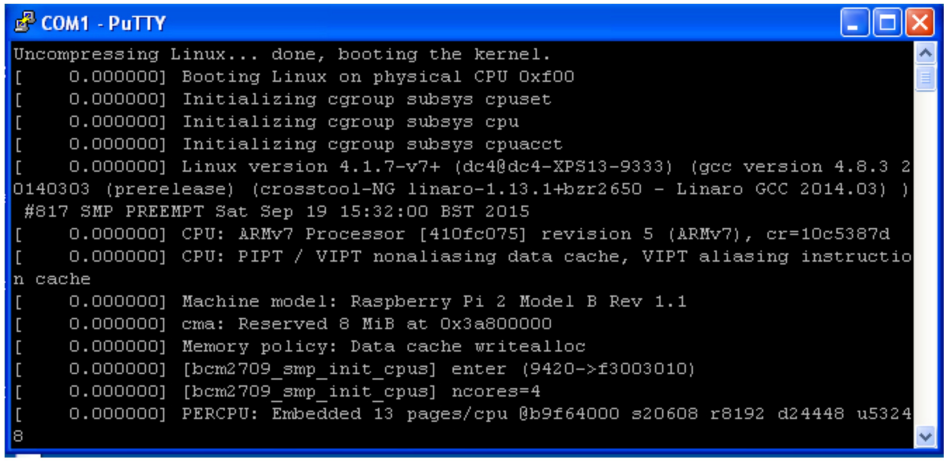
- Le mini UART est un faible UART de débit secondaire destiné à être utilisé en tant que console.
- Le mini UART présente les caractéristiques suivantes:
  - 7 ou 8 bits par caractère.
  - 1 bit de start et 1 bit de stop.
  - **Pas de parité**.

Il n'y a pas de support pour la parité **donc toute configuration avec parité ne sera pas prise en compte.**

Connecter la carte sur le port com du PC
ouvrir PuTTY et configuer la liaison de la façon suivante:



Mettre sous tension le Rapberry Pi 3



Par défaut l'UART du Raspberry Pi sert de port de débug pour Linux.

2 Libérer l'UART du mode debug

Empêcher l'émission de messages du Kernel et l'activation du mode debugging sur l'UART

```
pi@raspberrypi /dev $ sudo nano /boot/cmdline.txt

dwc_otg.lpm_enable=0 ~~console=ttyAMA0,115200 kgdboc=ttyAMA0,115200~~ console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

devient

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

3 Configuration en ligne de commande

Stty permet de connaître la configuration actuelle de la liaison

```
pi@myraspberry ~/C $ stty -F /dev/ttyS0
speed 9600 baud; line = 0;
-brkint -imaxbel
```

sur l'écran ci-dessus on peut voir que la vitesse de transmission est 9600 bauds, pour modifier la vitesse de transmission utiliser la commande suivante :

```
pi@raspberrypi:/etc $ sudo stty -F /dev/ttyS0 57600

pi@raspberrypi:~ $ stty -F /dev/ttyS0 115200 cs7 -parenb -parodd
        -ixon
```

La liaison série fonctionnera à 115200 bauds, chaque donnée sera codée sur 7 bits, sans parité, et il n'y aura pas de contrôle de flux xon/xoff (contrôle de flux logiciel). (Attention pas de caractères accentués sur 7bits !!!.)

4 Utilisation en ligne de commande

Sous Linux, "tout est fichier". L'envoi ou la réception de données sur le RPi se fait en lisant ou en écrivant dans /dev/ttyS0.

Par exemple pour envoyer le contenu du fichier achat.c sur la sortie série, la commande pourrait être la suivante:

```
pi@myraspberry ~/C $ cat achat.c > /dev/ttyS0
```

Autre exemple : avec la commande echo

```
pi@myraspberry ~/C $ echo 'bonjour le monde' > /dev/ttyS0
```

Pour afficher le flux de données reçues :


```
pi@raspberrypi:~ $ cat /dev/ttyS0

pi@raspberrypi:~ $ hexdump -C < /dev/ttyS0
00000000  61 61 61 61 0a 41 41 41  41 41 41 0a 61 61 61 61  |aaaa.AAAAAA.aaaa|
00000010  61 61 61 61 61 61 61 61  61 61 61 0a 2e 2e 2e 2e  |aaaaaaaaaaa.....|
00000020  2e 71 2e 2e 2e 2e 2e 2e  2e 2e 71 2e 2e 2e 2e 2e  |.q........q.....|
```

5 Programmation en C

En langage C sous LINUX, tous les périphériques sont considérés comme des fichiers. On y accède donc comme pour un fichier en mode bas niveau.
En C, nous utiliserons **open read write close**
pour en savoir plus : man 2 open man 2 read etc etc

et pour configurer la liaison série j'utilise la fonction system() car je n'ai pas réussi à faire fonctionner sans bug la configuration avec termios !!!.

```c
#include <stdio.h>  /* Standard input/output definitions */
#include <string.h> /* String function definitions */
#include <fcntl.h>  /* File control definitions */
#include <errno.h>

int main(int argc, char **argv) {

    int retour,fd;
    char recu[256];

    memset(recu, 0, 256); // efface le buffer
```

```
    fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY);
    if ( fd == -1 ){
         printf("pb ouverture: %s\n",strerror(errno));
         return(errno);
    }

    system("clear");   // on efface l'écran et on configure
    system("stty -F /dev/ttyS0 115200 cs8 -parenb -parodd
-ixon");

    write(fd, "Bienvenue sur Raspberry PI3 \n", 29); // envoi
    read(fd, recu, 255); // reception d'un message
    write(fd, "Message envoyé : ", 18); // envoi
    write(fd, recu, 255); // envoi du message reçu
    printf("Message recu:\n%s\n", recu);

    retour=close(fd);
    if ( retour == -1 ){
         printf("pb fermeture: %s\n",strerror(errno));
         return(errno);
    }
    return 0;
}
```


Fiche 16b Raspberry Pi3 Liaison série sur PI3