ESME Sudria 4 mai 2022 – Classe 2S
Cours de P. FOUBET

TD du cours

Initiation à µClinux

sur STM32F429I DISCx

===

Tests avec un module FTDI

USB-série TTL

1°) Remarques préliminaires

Certains convertisseurs sont dits « TTL » par opposition aux convertisseurs à la norme RS232. Les tensions et les niveaux logiques sont différents et incompatibles, ne les confondez pas ou vous risquez de griller vos appareils !
L'appellation « TTL » est aussi un abus de langage. Elle indique juste que les signaux sont en basse tension et que la polarité est positive. Cela recouvre aussi bien les signaux en 3,3V qu'en 5V car les circuits en technologie TTL (les plus répandus dans les années 80) utilisent des tensions entre 0,8V et 2,4V.
Mais aujourd'hui tous les circuits sont en technologie CMOS et les tensions montent vraiment à 3,3V et 5V : ils ne sont pas compatibles entre eux !
Donc **ATTENTION**, il y a un risque pour vos circuits si vous branchez une sortie 5V sur une entrée 3,3V, et l'inverse a toutes les chances de mal fonctionner.

Les tests de cet exemple ont été faits avec un convertisseur assemblé à base du contrôleur PL2303 fabriqué par Prolific Technology Inc.

Pour faire le test nous relierons les deux pins TXD et RXD avec un cavalier afin de permettre, sur la même machine, d'envoyer et de recevoir les données (image ci-contre).

2°) Détection de l'interface parmi les « devices »

Lorsque vous branchez un appareil sur un port USB, comme ce dongle série, le noyau devrait en principe le détecter rapidement et vous pouvez vous en assurer en listant les périphériques connectés au bus USB :

```
# lsusb
...
Bus 002 Device 053: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
```

ou en consultant la liste des messages du noyau Linux :

```
# dmesg | tail
...
[1553557.312632] usb 2-4: new full-speed USB device number 53 using xhci_hcd
[1553557.461257] usb 2-4: New USB device found, idVendor=067b, idProduct=2303,
bcdDevice= 3.00
[1553557.461261] usb 2-4: New USB device strings: Mfr=1, Product=2,
SerialNumber=0
[1553557.461262] usb 2-4: Product: USB-Serial Controller
[1553557.461264] usb 2-4: Manufacturer: Prolific Technology Inc.
[1553557.461919] pl2303 2-4:1.0: pl2303 converter detected
[1553557.462539] usb 2-4: pl2303 converter now attached to ttyUSB0
```

Le convertisseur Prolific PL2303 a bien été reconnu et son interface est ***/dev/ttyUSB0*** *:*

```
# ls -l /dev/ttyUSB0
crw-rw---- 1 root dialout 188, 0 mai    5 16:54 /dev/ttyUSB0
```

Si l'on veut faire ces tests en tant qu'utilisateur non administrateur (**invite** par exemple), il faut qu'il fasse partie du groupe **dialout**, sinon on sera obligé de faire un *chmod* à chaque fois que l'on mettra en place le dongle, ce qui n'est pas pratique. Pour vérifier qu'il en fait bien partie :

```
# groups invite
invite : invite dialout …
```

Sinon il faut l'ajouter :
```
# adduser invite dialout
```

3°) Paramétrage de l'interface

La commande étant longue et comme on sera amené à la modifier, on se fabrique un shell script Set_tty dans lequel on met (on peut aussi utiliser l'option -F à la place de la redirection, cf : man stty) :

```
#! /bin/sh
stty 115200 cs8 raw -echo -echoe -echonl -echok -echoprt -echoctl -parenb < /dev/ttyUSB0
```

Ici la vitesse de transmission (115200) est exprimée en bauds.
Rappel : Le débit binaire brut (bits par seconde) est le produit de la rapidité de modulation (bauds) par le logarithme en base 2 du nombre d'états possibles de l'élément de signal.

Pour rendre le script exécutable :
$ chmod a+x Set_tty

Pour initialiser l'interface :
$ ./Set_tty

Pour vérifier :
```
$ stty < /dev/ttyUSB0
speed 115200 baud; line = 0;
min = 1; time = 0;
-brkint -icrnl -imaxbel
-opost
-isig -icanon -echo -echoe -echok -echoctl
```

4°) Enfin les tests !!

On commence par se fabriquer un fichier (nommons le **meghasard**) contenant 1 Mo de données aléatoires :
```
$ dd if=/dev/urandom of=meghasard bs=1000000 count=1
$ ls -l meghasard
-rw-r--r-- 1 invite invite 1000000 mai    5 09:59 meghasard
```

Dans un premier terminal, on lance une tâche qui attend les données venant du device ***ttyUSB0,*** et qui va les stocker dans le fichier ***meghasard2.*** Le *block size* est plus grand pour être certain qu'elle va la récupérer en un seul bloc :
$ dd if=`/dev/ttyUSB0` of=meghasard2 bs=2000000

Dans un second terminal on envoie les données du fichier **meghasard** :
$ dd if=meghasard of=`/dev/ttyUSB0` bs=1000000 ; sync

1+0 enregistrements lus
1+0 enregistrements écrits
1000000 octets (1,0 MB, 977 KiB) copiés, 90,1219 s, 11,1 kB/s

Une fois que l'on a obtenu ce résultat, on interrompt la tâche de réception (Ctrl-C) du premier terminal, et pour bien vérifier que le transfert est fiable, on compare les deux fichiers :

```
$ cmp meghasard meghasard2
```

Le transfert s'est fait avec une bande passante de 11,1 kB/s ce qui correspond à peu près à notre paramétrage de 115200 bauds (1 octet correspond à 10 bits : 1 start + 8 data + 1 stop).

On va maintenant tester la vitesse maximale de transfert en modifiant le script **Set_tty** dans lequel on va remplacer 115200 par le maximum autorisé par la commande stty : **921600.**

Pour ré-initialiser l'interface :
$ ./Set_tty

Pour vérifier :

```
$ stty < /dev/ttyUSB0
speed 921600 baud; line = 0;
min = 1; time = 0;
-brkint -icrnl -imaxbel
-opost
-isig -icanon -echo -echoe -echok -echoctl
```

On recommence le test en commençant par lancer la tâche de réception dans le premier terminal. Puis on envoie dans le second terminal :

```
$ dd if=meghasard of=/dev/ttyUSB0 bs=1000000 ; sync
1+0 enregistrements lus
1+0 enregistrements écrits
1000000 octets (1,0 MB, 977 KiB) copiés, 11,8031 s, 84,7 kB/s
```

On vérifie comme tout à l'heure que les deux fichiers sont identiques.

On obtient une valeur indicative de 84,7 kB/s, qui correspond à 91,96 % du débit demandé, ce qui n'est pas si mal sur un noyau Linux standard, sachant que le PL2303 est donné théoriquement pour 12M bauds mais avec des drivers spéciaux !!

5°) Connexion à la carte STM32F429I

Suivant la documentation de la carte (http://stm32f4-discovery.net/2014/04/library-04-connect-stm32f429-discovery-to-computer-with-usart/), voici le tableau concernant les ports USART :

	Pins pack 1		Pins pack 2		Pins pack 3		
U(S)ARTx	**TX**	**RX**	**TX**	**RX**	**TX**	**RX**	**APB**
USART1	PA9	PA10	PB6	PB7			2
USART2	PA2	PA3	PD5	PD6			1
USART3	PB10	PB11	PC10	PC11	PD8	PD9	1
UART4	PA0	PA1	PC10	PC11			1
UART5	PC12	PD2					1
USART6	PC6	PC7	PG14	PG9			2
UART7	PE8	PE7	PF7	PF6			1
UART8	PE1	PE0					1

Les connecteurs PC10 et PC11 correspondent respectivement aux lignes TX et RX du troisième port série de la carte (USART3).
Un troisième câble reliera le connecteur masse (GND) du dongle à un de ceux de la carte.

Et, **très important**, le choix PC10/PC11 doit être celui qui a été mis dans le code du noyau :
```
$ vi arch/arm/mach-stm32/iomux.c
```

et vérifier que l'on a bien (sinon le rajouter) :
```
#if defined(CONFIG_STM32_USART3)
                gpio_dsc.port = 2;
                gpio_dsc.pin  = 10;
                stm32f2_gpio_config(&gpio_dsc, STM32F2_GPIO_ROLE_USART3);

                gpio_dsc.port = 2;
                gpio_dsc.pin  = 11;
                stm32f2_gpio_config(&gpio_dsc, STM32F2_GPIO_ROLE_USART3);
#endif
```

et il faut que dans la configuration du noyau l'interface USART3 soit activée :
```
CONFIG_STM32_USART3=y
```
Enfin, il ne faudra pas oublier de lancer le login sur l'interface ttyS2 (fichier inittab du rootfs) :
```
ttyS2::respawn:/bin/login -f root
```

Le câblage correspondant aux paramètres ci-dessus est donc le suivant :

```
Côté dongle USB                                  Côté STM32F429I
TXD    <----------------------------------------->   PC11
RXD    <----------------------------------------->   PC10
GND    <----------------------------------------->    GND
```

Ne pas connecter les pins d'alimentation : c'est inutile puisque le dongle est alimenté par le PC et la carte par son câble USB. Et cela évite les erreurs et les risques de griller la carte !

Dernière remarque : la liaison série ttyS0 est initialisée par U-Boot à 115200 bauds. Mais pas celle-ci ! La vitesse de transfert par défaut est de **9600 bauds**.

Pour se connecter à la carte via le dongle USB câblé :
```
$ picocom -b 9600 /dev/ttyUSB0
```