Notions sur le protocole de communication USB

Je donne ici quelques indications non exhaustives sur le protocole de communication des prises USB. Le but est de permettre de déboguer des scripts de tous langages quand on veut communiquer avec des périphériques. Cela permet d'aller « à la pêche » d'infos qui peuvent être utiles.

On va voir comment les périphériques et l'ordinateur ou contrôleur comme un arduino, communiquent.

Les octets qui sont véhiculés sur ces ports de communication ne le sont pas en série asynchrones comme par exemple sur le RS232 bien connu des anciens OMs, mais en NRZI (Non Remize à Zéro Inversé). En gros cela signifie qu'un bit dans un octet à 0 ne provoque pas de changement d'état par rapport au bit précédent, tandis qu'un bit à 1 provoque un changement d'état, mais le tout avec le « inversé » est d'obtenir une diminution de la bande passe du signal transmis (donc de grande importance en communication VHF), et de synchronisation des horloges réception et réception.

Les signaux électriques sont transmis sur deux fils plus la masse et en niveaux TTL , donc 5V, qui sont symétriques, ce qui veut dire que quand D- est à 0, D+ est à 1 et inversement. Ainsi on obtient une meilleure insensibilité aux parasites électriques par rapport au RS232 qui lui utilise un seul fil pour la réception et un autre fil pour l'émission. En RS232, si on a deux fils (TX et RX), pour chaque sens on reste en deux fils (donnée + masse) alors qu'en symétrique, on a deux fils (D-, D+, GND) ce qui permet cette meilleurs insibilisation en symétrique.

La fiabilité de l’USB2 repose sur l’apport de la diminution de la bande passante du au codage NRZI, et des fils symétriques de donnée. La longueur de cable USB peut être très longue, ce qui limite est la perte de niveaux des signaux due au cable. On peut aller en théorie jusqu’à une vingtaine de mètres. Si on met des HUBS intermédiaires pour regonfler le signal, on peut aller beaucoup plus loin. Cela peut jouer sur la vitesse en bauds de communication. Un cable court permet naturellement une vitesse en bauds la plus élevée. Avec un disque dur USB par exemple, cela peut avoir ne grande importance.

Une ligne USB, c’est à dire le signal qui circule sur les fils D- et D+ est bidirectionnelle, alors qu’en asynchrone classique RS232, ce n’est pas le cas.

Un bus avec des adresses:

Comme sur une prise USB on peut avoir plusieurs périphériques, pour les identifier les uns par rapport aux autres, on leur attribue une adresse sur 7 bits, soit 8 bits avec la parité. On a donc 128 adresses possibles.

Tout le monde sur le bus USB utilise la techniques des jetons qui circulent sur tous les périphériques, tous le monde peut recevoir des jetons qui ne leur sont pas destinés. Un jetons comprend une trame d’octets, dont le premier est l’octet d’adresse. L’arduino ou l’ordinateur envoi le jeton vers le premier périphérique, celui-ci le reçoit, voit si son adresse est dans ce jeton. Si c’est le cas, il utilise les données du jeton. Si l’adresse du jeton n’est pas la sienne, il le transmet au périphérique suivant et ainsi de suite.

Normalement, le bus USB prévoyait que chacun des périphériques avait deux prises, une sur le périphérique qui le précède, et l’autre vers le périphérique suivant. On pouvait ainsi chaîner les périphériques les uns derrières les autres. Mais de fait, les périphériques n’ont qu’une prise, on utilise alors un « Hub » USB bien connu de tout le monde. Il a deux fonctions:

permettre le branchement de plusieurs périphériques sur une même prise USb, et apporter une alimentation 5V conséquente pour tous les périphériques qui peuvent être branchés sur lui. La valeur max théorique sur un bus USB est de 0,5A, ce qui est loin d'être respecté. D'où l'intérêt d'un hub USB.

Repérer un périphérique dans le répertoire /dev sous linux:

Sous linux, la commande lsusb permet d'obtenir des informations sur ce qui est connecté sur l'ordinateur en USB.

Avec les droits root, # lsusb donne par exemple:

Bus 002 Device 002: ID 18a5:0411 Verbatim, Ltd
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

On voit ici plusieurs colonnes: la colonne « Bus », « Device », « ID » et des mots compréhensibles. Ces derniers apportent une information immédiate et claire sur le périphérique concerné. La colonne « ID comporte 4 octets (codés en octal), puis le caractère « : » et enfin un autre octet.

Le premier octet est un codage conventionnel standard sur le fabriquant, et le deuxième octet se rapporte au modèle du produit chez ce fabriquant.

Le nombre après « Bus » est une première adresse en rapport avec la prise USB. Le nombre après « Device » est une adresse attribuée au périphérique branché sur cette prise par le système linux.

Interprétation de ces informations:

Si on va dans /dev/bus/usb/001, on voit la présence d'un fichier: 002 il est en rapport avec la chaîne « VIA Labs, Inc. Hub ».

Souvenez vous que sous linux « tout est fichier », même un périphérique. Cela signifie que la manière de traiter un fichier classique et un périphérique est la même: on lit ou on écrit. On peut comprendre que la routine de lecture et écriture des fichiers est la plus utilisée par le système linux. Mais se greffe à ce traitement la routine dite « pilote » qui permet de gérer tout cela.

Pour revenir à notre exemple: le Bus 001 est: /dev/bus/usb/001, donc un répertoire, le Device 002 est /dev/bus/usb/001/002 soit un fichier.

Avec la commande sous root: # cat /dev/bus/usb/001/002 on peut voire les données issues du périphérique en question.

Exemple: Acquisition récepteur GPS

En utilisant la commande sous root lsusb, j'obtien entre autres la ligne ci-dessous:

Bus 004 Device 016: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP2102/CP2109 UART Bridge Controller [CP210x family]

Je l'ai déterminée en lançant # lsusb avec et sans le récepteur GPS branché. On lit ainsi que le récepteur est détecté sur le BUS 004, le device 016. Ce qui fait qu'on le trouve dans le fichier périphérique: /dev/bus/usb/004/016.

La commande # dmesg nous indique aussi qu'un périphérique associé est créé en /dev/USB0.

Si on installe la commande # minicom qui est un terminal tty virtuel, sur le modem /dev/ttyUSB0 et à une vitesse 9600 bauds

et 8 bits sans parité, on voit défiler les données NMEA. Mais si on met comme modem /dev/bus/usb/004/016, on botient la même chose.

Voilà, on est au terme de cet article technique linux. Avec les raspberry on peut d’amuser à découvrir l’intérieur d’un OS linux plus en profondeur