Les liaisons série démythifiées

Version : A
2017-03-15

33 pages

Table des matières

1 Révisions de ce document

Version	Modifications
A	Première version.

2 Généralités

2.1 But de ce document

Malgré les connexions à grand débit permises par USB, les liaisons série conventionnelles (RS232) sont encore utilisées dans le milieu industriel; leur vitesse, ridicule selon les standards du jour, est souvent suffisante pour transmettre de petits messages, à une cadence faible. Leur souplesse est sans pareil.

La gestion des liaisons série asynchrones[1] est souvent confiée à un circuit intégré dédié nommé U(S)ART, ACIA ou autre, mais si un tel circuit n'est pas présent dans l'ordinateur, cette gestion est suffisamment rustique pour qu'il soit relativement simple d'implémenter cette gestion en logiciel, ce qui n'est pas le cas par exemple avec l'USB (autre protocole série), où l'emploi d'un circuit intégré dédié est quasiment indispensable.

Les liaisons série ont la réputation d'être difficiles à mettre en œuvre, mais c'est essentiellement dû à la méconnaissance de quelques principes de base, que nous allons tenter de clarifier ici.

Dans ce document, on utilisera plutôt la convention de parler de **liaison série**, de **ports série**,… plutôt que de liaison RS232,...

On emploie généralement l'appellation « RS232 », mais RS232 est une norme, qui définit précisément de quoi elle parle ; dans la pratique, on parle souvent de RS232 pour évoquer des aspects non couverts par cette norme, en particulier le format de transfert des octets.

Cette norme existe en plusieurs versions (RS232-C, EIA232-D,…) et aussi selon plusieurs appellations, en fonction de l'organisme de normalisation local.

Les normes ne sont pas librement accessibles (156 US$...), mais c'est sans grande importance, car ce qu'il faut prendre en compte, c'est l'implémentation de cette norme par les constructeurs de matériels informatiques, et il faut être conscient de certaines libertés prises par ces constructeurs vis-à-vis de ces normes.

L'appellation de « liaison série » retenue n'est, en toute rigueur, pas suffisamment précise, car tel M.Jourdain, vous utilisez d'autres liaisons série peut-être sans le savoir : USB, SATA, FireWire, Ethernet…

Cependant, l'appellation « port série » sans autre précision désigne de manière non ambiguë les fameux ports *COM* des PC sous MSDOS et sous Windows.

Ce document n'a pas vocation à être une référence : il cherche tout au plus à capitaliser l'expérience de son auteur.

2.2 A qui s'adresse ce document ?

Ce document est destiné à des personnes ayant un minimum de connaissances en informatique, mais il fera hurler de rire les ingénieurs en télécommunications.

Certains chapitres ne diront quelque chose qu'à ceux ayant des notions d'électronique. Les autres peuvent les ignorer.

---

1 Les seules évoquées dans ce document. Les liaisons série synchrones ne se trouvent qu'en milieu professionnel ; elles sont caractérisées par la présence de signaux d'horloge pour l'émission et/ou la réception.

2.3 Sur la forme de ce document

Les liens soulignés sont cliquables, ainsi que les numéros de page, chapitres et acronymes, qui ne sont pas spécialement mis en évidence par un soulignement.

Les liens suffixés par le symbole « » [1] sont des URL externes à ce document (Web) ; les autres liens renvoient dans le présent document.

Le document comporte de nombreuses redites, pour la plupart volontaires, ce qui rend la lecture agaçante, mais qui peuvent aider à mémoriser certains termes ou concepts.

2.4 Réserves

L'auteur a dit tout ce qu'il savait sur le sujet (et même plus). Il est donc inutile de le contacter pour lui demander de régler un problème, il ne pourra que vous relire (mal) ce document.

Il va sans dire que l'auteur dégage toute responsabilité en ce qui concerne toutes conséquences directes ou indirectes de la lecture de ce document. Vous êtes supposés être des professionnels, ou des amateurs responsables.

Si on doit vous dire explicitement qu'il ne faut pas mettre le chat dans le micro-ondes, alors cessez immédiatement la lecture de ce document, et détruisez en par le feu toutes les copies en votre possession.

3 Principes de base

3.3 Liaison canonique

La chose la plus importante à savoir pour mettre en œuvre une liaison RS232, est que les liaisons de ce type ont été prévues initialement (dans les années 60) pour connecter un terminal à un modem. Point barre.

Une fois ce principe assimilé, vous êtes parés pour effectuer les connexions les plus hétérodoxes.

Ce principe de base implique que la liaison est dissymétrique (terminal d'un côté, modem de l'autre), que l'on a deux équipements reliés **et pas plus**, et que de nombreux fils sont nécessaires pour permettre le fonctionnement du modem.

Dans ces conditions, la connexion est très simple : il suffit de relier les deux équipements par un câble contenant 25 fils, en **point à point** (la broche numéro 1 d'un connecteur est reliée à la numéro 1 de l'autre connecteur, la numéro 2 à la numéro 2, etc.).

Il n'est pas nécessaire de recourir à des croisements de fils, à des courts-circuits entre signaux, à des diodes ou autres bidouilles plus ou moins orthodoxes.

Tout autre type de liaison que Terminal/Modem ressort du bricolage. C'est pourtant dans la pratique les connexions que nous rencontrons essentiellement.

Ce document a pour but de vous aider à comprendre cette liaison canonique, afin de faire les bricolages adéquats **en toute connaissance de cause**, plutôt que par la technique essais/erreurs.

C'est l'occasion d'introduire deux termes relativement peu utilisés, pas assez en tous cas,

1 Emprunté à Wikipédia, sous licence Creative Commons.

mais qui sont *stratégiques* dans ce document ; il s'agit de « DTE » et « DCE », qui désignent respectivement le dispositif assimilé à un terminal, et le dispositif prenant la place du modem.

Pour simplifier, le DTE est l'ordinateur, et le DCE est le périphérique.

Le DCE peut être virtuel, par exemple dans le cas de la connexion DTE/DTE, non prévue initialement, il n'y a a priori pas de DCE/modem, mais le null-modem indispensable permet à chaque DTE/terminal de voir l'autre DTE/terminal comme un DCE/modem.

3.4 Échange de politesses[1]

Le contrôle de flux est une notion qu'il va être indispensable d'évoquer dans les tous prochains chapitres : il convient donc de la présenter brièvement.

Même aux vitesses modestes généralement pratiquées avec les liaisons série, cette vitesse peut être encore trop grande pour certains récepteurs.

Par exemple, les vénérables imprimantes à aiguilles ne peuvent souvent pas supporter un flot continu de texte à plus de 1 200 bits/s.

Même si l'imprimante peut suivre ce débit infernal, elle peut avoir besoin de souffler un peu pour passer à la page suivante.

Le besoin d'un dialogue entre les deux interlocuteurs sur la liaison série se fait donc sentir :

Émetteur	Récepteur
Es-tu prêt à recevoir mes données ?	
	Oui. Tu peux envoyer tes données.
J'envoie mes données.	
J'envoie toujours des données.	
	Je n'arrive plus à suivre. Je ne suis plus prêt à recevoir tes données.
Je suspends l'envoi de mes données, tant que tu n'es pas prêt à les recevoir à nouveau.	
	(Je traite les données reçues qui se sont accumulées).
	J'ai rattrapé mon retard. Je suis à nouveau prêt à recevoir tes données.
Je continue l'envoi de mes données depuis l'endroit où je m'étais arrêté.	

Il y a deux possibilités : soit le contrôle de flux matériel, soit le contrôle de flux logiciel. Ces deux possibilités sont détaillées plus loin dans ce document.

1 Tentative de traduction de « *handshake* ».

4 L'aspect logiciel

4.1 Le format des données

Une liaison série a pour principe de transmettre les bits les uns après les autres. Monsieur de La Palice n'aurait pas mieux dit.

Les bits sont émis à une vitesse qui est une convention adoptée par les deux interlocuteurs. La vitesse maximum étant déterminée par le matériel. La vitesse est généralement mesurée en bauds, ou en bits par seconde. De la vitesse, on en déduit le temps de transmission d'un bit. Par exemple, à 9 600 bits par seconde, un bit dure 1 / 9 600 s, soit à peu près 0,1 ms.

Les bits sont regroupés par paquets d'une taille pratique pour l'usage que l'on veut en faire, en général 8 bits car l'octet est l'unité de manipulation de données des micro-ordinateurs.

On peut se contenter de 7 bits, si on ne transmet que du texte purement ASCII (non accentué).

A l'époque des téléscripteurs[1], les caractères étaient codés sur 5 bits. Le micro-contrôleur 8051 possède un mode 9 bits. En cherchant bien, on devrait pouvoir trouver des longueurs encore plus exotiques.

On voit donc que la taille de l'unité d'information est assez arbitraire : c'est ce que les américains désignaient par « *byte* », avant que ce mot ne devienne dans la pratique synonyme d'« octet »[2].

Dans la vraie vie, le choix est limité à 7 ou 8 bits, mais on utilise presque toujours 8 bits, si bien que par la suite on parlera de transmission d'« octets », en gardant à l'esprit que cet « octet » n'est qu'un cas typique.

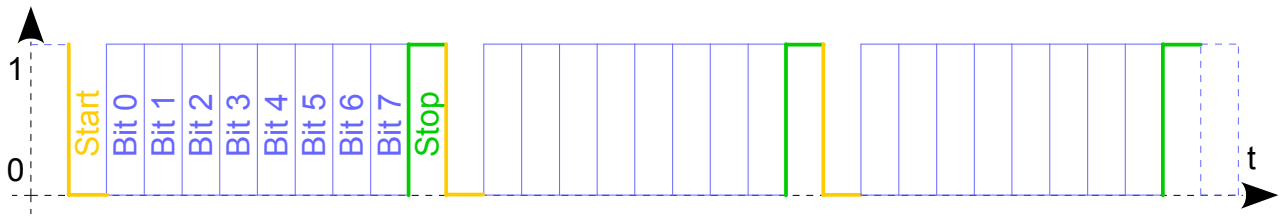
Une fois le principe d'envoyer les bits les uns à la suite des autres adopté, se pose alors la question « comment déterminer où commencent et où finissent les octets ? ».

La réponse en mode asynchrone, le seul mode qui nous intéresse ici car de loin le plus courant, est d'encadrer les bits utiles de l'octet par 2 bits *balises* dont la valeur est toujours la même, ce qui rend assez facilement repérable le début et la fin de l'octet. Ces bits sont utilisés par les circuits intégrés dédiés à la gestion de la réception série pour se synchroniser[3].

Le niveau de repos (en l'absence de toute émission) de la ligne étant conventionnellement « 1 » logique[4], le bit signalant le début de l'octet a naturellement la valeur « 0 » logique. Ce bit est désigné « bit de start ». Le bit signalant la fin de l'octet prend la valeur « 1 » logique, signifiant le retour au repos de la ligne. Ce bit est nommé « bit de stop ».

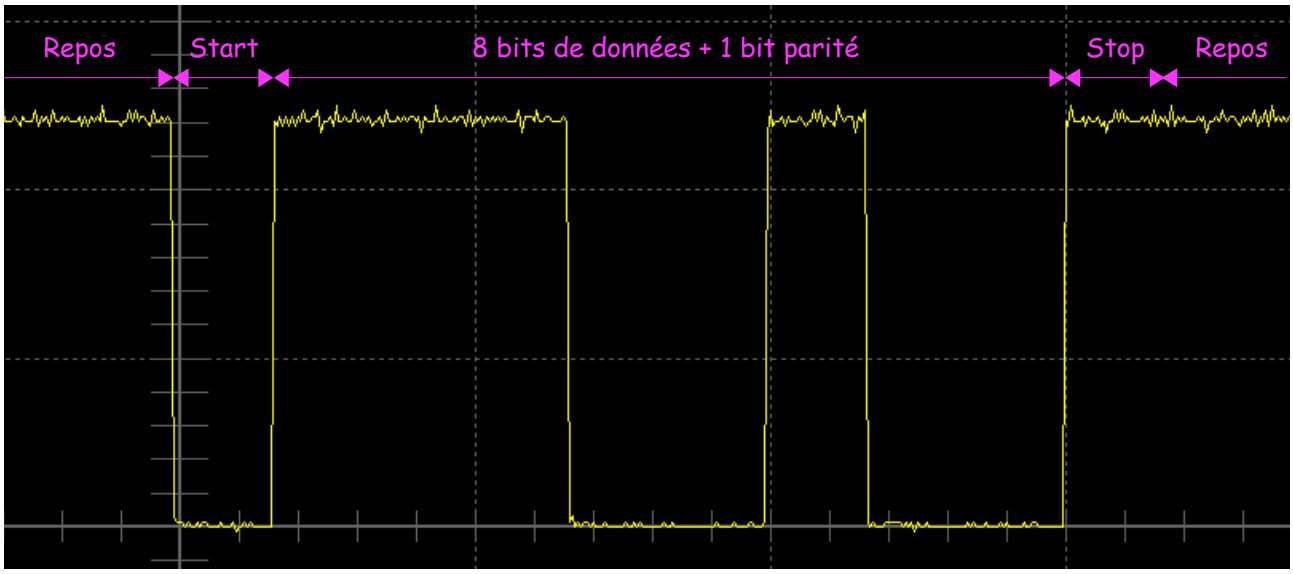
---

1 Pas encore complètement morts...
2 Ethymologiquement, un octet ne peut être constitué que de 8 bits.
3 Le mode « asynchrone » est caractérisé par l'absence d'un signal d'horloge, mais émetteur et récepteur doivent quand même se synchroniser au début de chaque octet.
4 On est obligé de préciser « logique », car observés à l'oscilloscope, les signaux présents sur la liaison RS232 proprement dite sont inversés, c'est-à-dire que la valeur logique « 1 » est codée par une tension négative (typiquement -12 V), alors que la valeur logique « 0 » est codée par une tension positive (+12 V). Pour plus de détails, voir le chapitre « Logique positive / négative », page 29.



Trois octets quelconques, délimités par Start et Stop

Illustration 1



Avec certains appareils anciens et lents, il était nécessaire d'utiliser 2 bits de stop, voire 1,5 bit de stop. Il est surprenant de parler de 1,5 bit, étant donné qu'un bit représente la plus petite unité d'information, mais il suffit de remarquer qu'en fait on demande que la ligne reste au repos pendant une **durée** égale à 150 % de la durée de transmission d'un bit de donnée.

A noter, que le nombre de bits de stop correspond à la durée **minimum** à respecter entre deux octets. Il n'y a pas de durée maximum : il peut s'écouler sans problème trois jours entre le bit de stop d'un octet, et le bit de start du suivant... Certains protocoles imposent cependant que les octets soient collés les uns aux autres, c'est-à-dire qu'un bit de stop doit immédiatement être suivi du bit de start de l'octet suivant, et ce, jusqu'à la fin du message. Ce genre de protocole se situe au-dessus du protocole série asynchrone, et donc peut imposer des règles supplémentaires, plus contraignantes.

Comportement surprenant : imaginez une liaison série dont les deux interlocuteurs sont paramétrés de la même façon, sauf[1] sur le nombre de bits de stop, un interlocuteur émet 1 bit de stop, et l'autre 2.

On pourrait croire que, vu que le paramétrage est différent, la communication ne va pas fonctionner. En fait, elle fonctionne à moitié : les octets émis avec 2 bits de stop sont parfaitement reçus par celui qui est configuré pour 1 seul bit de stop.

Il suffit de relire le paragraphe un peu plus haut : le nombre de bits de stop paramétré est un **minimum** ; l'interlocuteur paramétré pour 1 bit de stop en attend **au moins 1**, et il en reçoit 2 : tout va bien. Dans l'autre sens, celui qui attend 2 bits de stop, en reçoit 1 seul : il

1 A la suite d'une erreur de paramétrage, bien entendu. Paramétrer consciemment les deux extrémités de façons différentes, c'est chercher les problèmes, qui viendront pourtant bien tous seuls...

rejette tous les octets reçus, car il les considère mal formés.

Sauf cas exceptionnel dûment justifié, on n'utilisera qu'un seul bit de stop. De nombreux appareils et logiciels ne permettent même pas de paramétrer ce nombre de bits de stop : un seul bit de stop étant implicite.

Le dernier sujet à décrire concernant l'aspect logiciel des liaisons série asynchrones, est la parité.

La parité est un contrôle rudimentaire, assez similaire à la preuve par neuf : elle permet de détecter facilement certaines erreurs de transmission (bits déformés par des parasites de tous genres), mais ce n'est pas une preuve absolue : elle permet de détecter une erreur sur un octet, mais pas deux.

Le contrôle par parité consiste à **ajouter un bit** dont la valeur est fonction du nombre de bits à 1 de l'octet émis. En réception, le bit de parité est recalculé selon la même règle qu'à l'émission, et le résultat du calcul est comparé au bit de parité reçu. Si la même valeur de parité est trouvée, l'octet est accepté. Sinon, il est rejeté, ou marqué comme erroné par le circuit intégré dédié à la gestion de la liaison série.

L'usage de la parité est facultatif. Il est seulement nécessaire que les deux interlocuteurs de la liaisons respectent la même convention, à savoir :

Anglais	Français	Commentaire
None	Sans	Aucun bit de contrôle de parité n'est ajouté. Seuls les 7 ou 8 bits de données sont transmis.
Even	Paire	Un bit est ajouté aux 7 ou 8 bits de données, de façon à ce que le nombre de bits à 1 soit pair (respectivement impair).
Odd	Impaire	
Mark	*Mark*	Bit supplémentaire, de valeur **constante** (mark=1, space=0), donc a priori inutile. Il n'y a pas d'usage connu[1] pour ce paramétrage.
Space	*Space*	

4.2 Break !

Si vous n'avez jamais entendu parler de *break* (dans le contexte des liaisons série) ou de *framing error*, c'est que vous n'êtes a priori pas concernés par ce chapitre : vous pouvez le sauter sans trop d'états d'âme.

Nous avons vu que, soit ligne est au repos, soit elle transmet des octets qui possèdent au moins un bit à l'état actif (le bit de start), et au moins un bit à l'état repos (le bit de stop).

Tant que des octets sont à transmettre, la ligne n'est jamais à l'état actif plus longtemps que la durée d'un octet.

La condition de *break* consiste à créer volontairement une condition particulière qui ne corresponde pas à l'envoi d'un octet et qui puisse être détecté par le récepteur.

On rappelle que l'émission d'un octet sur la liaison série consiste à, dans l'ordre :

1. Mettre la ligne dans l'état actif, en envoyant un bit de start (valeur logique : 0),
2. Envoyer les 8 bits de l'octets (éventuellement seulement 7 bits si on se contente d'ASCII pur), de valeurs absolument quelconques,

1 Ce paramétrage semble être un fossile datant de l'âge de pierre. Aujourd'hui, il ne semble plus être utilisé que pour des *bidouilles*.

3. Envoyer optionnellement un bit de contrôle de parité, dont la valeur est fonction de l'octet qui vient d'être envoyé,
4. Enfin, placer la ligne dans l'état « repos », en envoyant un bit de stop (valeur logique : 1).

Le bit de start est obligatoire, pour signaler le début de *quelque chose*, « quelque chose » étant généralement un octet, mais peut aussi être un *break*.

Les 8 bits suivants sont absolument quelconques : aucune valeur n'est réservée dans le cas général.

Le bit de parité, s'il existe, soit possède une valeur correcte et dans ce cas l'octet est accepté comme un octet quelconque, soit possède une valeur incorrecte et l'octet est rejeté comme erroné.

Enfin, si le bit de stop est absent, c'est-à-dire si la ligne ne passe pas à l'état repos dans le temps escompté, une erreur de réception est également détectée, dite « erreur d'encadrement » (*framing error*).

Un *break* n'est qu'une « erreur d'encadrement » volontairement produite, destinée à signaler une condition plus ou moins exceptionnelle (interruption du message transmis, par exemple, ou bien la fin du message)

On distingue parfois un *break* d'une *framing error*, mais la différence n'est pas très claire. Il semble que le caractère intentionnel du *break* se manifeste par sa durée : la ligne est positionnée à l'état actif pour une durée relativement longue (de l'ordre du temps de transmission de 3 caractères au minimum). Il est possible, mais sans garantie, qu'un circuit récepteur distingue un *break* par le fait que tous les bits de données sont à l'état actif, y compris l'éventuel bit de parité[1].

Si le logiciel utilisé en réception[2] distingue le *break* de la *framing error*, il est possible qu'une *framing error* soit signalée avant le *break*, puisque le *break* **est** semble-t-il une longue *framing error*.

Pour résumer, le *break* est un non-caractère destiné à signaler un événement conventionnel.

4.3 Contrôle de flux logiciel

Le contrôle de flux logiciel est peu utilisé, car son emploi est pratiquement limité aux échanges de texte (par opposition aux échanges binaires, où les octets peuvent prendre des valeurs quelconques), et demande un certaine intelligence de la part des deux interlocuteurs.

Le principe est que le destinataire accepte a priori les données, jusqu'à ce qu'il envoie un avertissement demandant à l'émetteur de suspendre l'émission.

L'avertissement a généralement la forme du caractère de contrôle XOFF/^S[3].

Le trafic peut reprendre dès que le récepteur donne son feu vert en envoyant le caractère de contrôle XON/^Q.

---

1 8 bits à 1, avec une parité à 1 est cependant une combinaison parfaitement légale pour l'octet 0xff, si la parité configurée est impaire.

2 Par exemple, la fonction de bas niveau `ClearCommError()` de l'API Win32 distingue les deux cas.

3 La notation ^X indique un caractère ASCII non imprimable, obtenu par pression de la touche « Ctrl » du clavier, et de la touche « X ».

Le protocole XON/XOFF est standard, et peut donc être employé par divers équipements ou logiciels.

Si le même logiciel est utilisé par les deux interlocuteurs, il peut implémenter un contrôle de flux spécifique, qui est alors transparent : l'utilisateur n'a pas à s'en préoccuper, à part éventuellement paramétrer le logiciel de communication de la même façon des deux côtés.

5 L'aspect matériel

La liaison RS232 standard utilisait initialement un connecteur dit « DB-25 », qui comme sa dénomination le laisse supposer, possède 25 contacts.

Nous verrons que ces 25 contacts ne sont pas toujours utiles, et qu'en général on peut se contenter de beaucoup moins, jusqu'à 3 contacts, voire 2 dans le cas dégénéré où la communication est unidirectionnelle.

Ce connecteur 25 points reste cependant beaucoup employé sur les matériels professionnels, quitte à n'utiliser qu'une infime partie de ses possibilités.

Un avantage secondaire de ce connecteur, est sa robustesse mécanique.

Pour illustrer les différents câblages, nous n'utiliserons que des connecteurs 9 points, dits « DE-9 » ou de manière erronée, « ~~DB-9~~ », aujourd'hui plus courants que les 25 points ; les professionnels auront vite fait de faire la correspondance avec les connecteurs 25 points si nécessaire.

Sur ce connecteur, les signaux ont tous les mêmes caractéristiques électriques. Il faut cependant distinguer trois types de signaux :

1. les sorties, où le connecteur **fournit** une tension,
2. les entrées, où il faut **appliquer** une tension.
3. la borne de référence de tension, celle par rapport à laquelle les autres tensions sont mesurées[1] : la masse, qui est malencontreusement nommée GND (***ground*** = terre), ou SG (***signal ground***), car cette borne ne doit en principe **pas** être reliée à la terre.

Il faut un fil distinct par signal utilisé, plus un fil commun à tous les signaux (GND/SG).

En raccordant deux équipements par une liaison série, les principes à respecter sont de :

1. Connecter la borne unique de masse (GND/SG) d'un équipement à la borne de masse de l'autre équipement, de façon à définir une référence de tension commune,
2. **Ne pas** relier les bornes de **terre** entre elles ; éventuellement la terre peut être reliée d'un seul côté pour assurer le blindage du câble,
3. Connecter les sorties d'un équipement, aux entrées de l'autre équipement, et réciproquement,
4. **Ne pas** connecter les bornes inutilisées, aussi bien entrées que sorties[2].

1 Une tension n'est pas une mesure absolue. On mesure une tension par rapport à un point de référence. On parle de **D**ifférence **D**e **P**otentiel (DDP) entre le point mesuré, et le point de référence. Souvent, la référence est implicite. On dit qu'un conducteur est porté à 230 V, implicitement par rapport au conducteur neutre, ou à 12 V, implicitement par rapport à la masse.

2 Ce qui peut être un réflexe d'électronicien habitué aux entrées MOS.

5.1 Les niveaux de tensions RS232

Tout ceci peut commencer à paraître compliqué, mais s'éclaircira à l'usage.

Une bonne nouvelle cependant : la spécification des lignes série impose qu'aucun court-circuit entre broches de ce connecteur ne peut avoir de conséquences irréversibles, même si le court-circuit est permanent. Un port RS232 est quasiment indestructible, tant que vous ne le branchez pas directement sur 230 Volts.

Les circuits utilisés en interface des ports RS232 sont en général très robustes et tolérants, mais certains constructeurs de matériels bon marché jouent sur cette tolérance.

En particulier, certains PC portables (du temps béni où ils étaient encore équipés de ports série) et certains adaptateurs USB/Série ne fournissent pas des niveaux de tension conformes à la norme RS232.

La norme prévoit qu'un signal appliqué à une borne d'entrée est considéré en réception :

- comme la valeur logique « 0 » s'il a une valeur comprise entre +3 V et +15 V,
- comme la valeur logique « 1 » s'il a une valeur comprise entre -3 V et -15 V.

A noter, que les signaux électriques sur la liaison série sont en **logique négative**, c'est-à-dire qu'un signal ayant une tension négative par rapport à la masse GND/SG est considérée comme « 1 » logique, et qu'une tension positive est un « 0 » logique.

Une tension d'une valeur absolue supérieure à 15 V peut avoir des effets indésirables, potentiellement destructeurs.

Une tension de valeur intermédiaire, inférieure à 3 V en valeur absolue, ne donne pas de résultat garanti. En toute rigueur, la valeur logique reconnue par le circuit d'entrée peut aussi bien être « 0 » que « 1 ». On peut en déduire qu'un signal d'entrée ne doit **jamais** être relié à la masse GND/SG, dont la tension, par définition de la masse, est de 0 V.

Dans la pratique, les circuits fournissent généralement un signal de +12 V ou -12 V. Même après une longueur acceptable de câble de mauvaise qualité qui dégradera le signal, le récepteur verra un signal dont la valeur absolue sera largement supérieure aux 3 V critiques.

Cependant, certains constructeurs vendant des adaptateurs sans marque *à pas cher* font des économies en utilisant des tensions qu'ils ont sous la main, à savoir 0 V et +5 V.

La tension de +5 V est dans la plage autorisée par la norme (+3 V à +15 V), mais est très proche de la limite basse ; une longueur importante de câble de mauvaise qualité peut dégrader le signal jusqu'à obtenir une valeur inférieure à la valeur minimum de +3 V.

Par contre, la valeur de 0 V adoptée pour représenter un « 1 » logique est franchement **hors norme**.

Cependant, ces adaptateurs peuvent arriver à fonctionner dans des conditions favorables : courte longueur de câble (ne dépassant pas quelques mètres), pas de parasites industriels et circuits d'entrée dont la tolérance excède la norme (ce qui est souvent le cas).

A savoir, que les niveaux de tensions sont aussi dégradés par des outils d'observation (bien pratiques pourtant) du genre « boîte à LED ».

Équivalences d'appellations des niveaux logiques ; selon les sources, on utilise plusieurs termes différents pour désigner l'état des signaux :

« 0 » logique	+12 V	Space	Active	Negated	ON
« 1 » logique	-12 V	Mark	Idle	Asserted	OFF

5.2 Contrôle de flux matériel

La norme RS232 impose un connecteur 25 points, dont 24 sont connectés. Un fil est utilisé pour l'émission des données, un autre fil pour la réception, un troisième est une référence commune à tous les signaux.

A quoi peuvent donc servir les 21 autres broches des connecteurs Sub-D 25 points, historiquement utilisés pour les liaisons RS232 ?

Ces signaux, en partie du moins, sont utilisés pour le contrôle de flux matériel.

Ce protocole matériel, couramment dénommé RTS/CTS, est géré à l'aide de quelques-uns des 24 fils de la liaison.

Ces signaux sont parfois utilisés, non pas en tant que contrôle de flux proprement dit, c'est-à-dire pour suspendre le trafic juste le temps nécessaire, mais simplement pour signaler que l'appareil distant est connecté et est sous tension, et donc a priori prêt à recevoir.

5.3 Connecteurs

5.3.1 DB-25

Le connecteur 25 points fait partie de la norme RS232. C'est donc le connecteur de référence.

Historiquement, les équipements étaient équipés de connecteurs femelles, ce qui peut se comprendre car les signaux de sorties sont alors moins exposés avec ce type de connecteur.

C'était compter sans IBM, qui a choisi d'utiliser un connecteur DB-25 femelle pour le port d'**imprimante parallèle** de ses PC, là où tout le monde utilisait un connecteur Centronics femelle[1]…

Du coup, IBM a adopté le connecteur DB-25 **mâle** pour son port série. L'inconvénient apparent de ce connecteur est d'exposer les signaux de sortie à toutes sortes de courts-circuits involontaires, mais la norme RS232 précise que les courts-circuits entre broches quelconques et d'une durée indéterminée ne doivent pas avoir de conséquences dramatiques.

5.3.2 DE-9

Le connecteur DB-25 est spécifié par la norme, mais il est prévu pour implanter **tous** les signaux RS232.

Dans l'immense majorité des cas, un nombre très réduit de ces signaux est vraiment nécessaire ; dans le cas le plus simple, on pourrait se contenter de 3 fils, voire 2.

IBM a donc fini par imposer un connecteur plus petit que le connecteur officiel : le DE-9,

1 Syndrome *Not Invented Here*, ou bien économies de bouts de chandelles ?

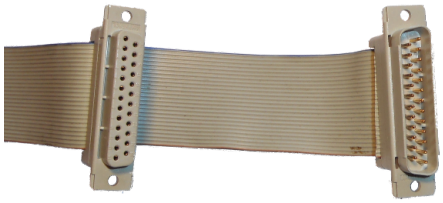
qui véhicule jusqu'à 9 signaux, ce qui est généralement plus que nécessaire, et est largement suffisant pour poser des casse-têtes de câblage[1]...

5.3.3 RJ45

On trouve de plus en plus des connecteurs RJ45 ; il ne semble pas y avoir de consensus concernant l'affectation des signaux aux différentes broches. Souvent, le constructeur fournit un équipement équipé de RJ45, et les câbles qui vont avec.

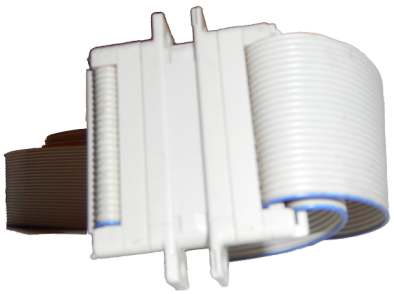
5.4 Câbles

Le câble indispensable est une nappe prolongatrice de quelques mètres, munie à chaque extrémité d'un connecteur mâle et d'un connecteur femelle espacés de quelques centimètres.



Un tel câble peut facilement être fait soi-même. Normalement, il faudrait utiliser un outil spécifique pour sertir les connecteurs, outil dont seul un usage professionnel peut justifier l'achat. Heureusement, pour sertir ces connecteurs on peut utiliser un petit étau, et mettre un caoutchouc dur sur chacun des mors pour protéger le connecteur. Le câble doit être **bien perpendiculaire** au connecteur. Serrer doucement l'étau, jusqu'au déclic du connecteur.

Faire attention à connecter les broches numéro 1 de tous les connecteurs entre elles. En général, un fil extérieur de la nappe est rouge : c'est celui à faire correspondre à la broche 1. Il doit être possible de refermer un connecteur mâle sur un connecteur femelle, sans avoir à vriller la nappe.



S'il est nécessaire de recouper la nappe, utiliser une paire de ciseaux assez forts, mais coupant très bien, ou alors un massicot ; si la coupure n'est pas parfaitement nette, il y a risque de court-circuit entre deux fils adjacents.

Le résultat obtenu est bien plus propre et plus sûr qu'un câble soudé.

Éviter de réutiliser des **câbles de provenance inconnue** : en général, ils ont été adaptés à un besoin précis ; c'est à cause de bricolages avec ce genre de câble, qu'établir une

1 L'historique des connecteurs DB-25 et DE-9 n'est qu'un délire paranoïaque de l'auteur, sans la moindre preuve.

liaison série est considéré comme un travail digne d'Hercule.

5.5 Terre !

Le câblage de la terre n'est pas ce qu'il y a de plus simple. Ce chapitre ne prétend pas donner de solution universelle, ni même de solution tout court.

Les règles de câblage de la terre dépassent largement le cadre de ce document.

On n'évoque ici que quelques principes généraux.

Pour commencer, il faut un peu clarifier les choses, entre la terre et la masse.
La confusion vient en partie de ce qu'en anglais (toute la littérature technique est en anglais), la masse est appelée « *Ground* », qui signifie littéralement « sol ».

La langue française est parfois confuse, en distinguant dans le domaine électrique « masse électrique » qui est le potentiel de référence, et « masse mécanique », qui n'est peut-être pas la terre, mais qui doit généralement être reliée à la terre pour raisons de sécurité.

La masse est le potentiel de référence pour les signaux ; cette référence doit être unique pour les deux interlocuteurs de la liaison RS232, c'est-à-dire, entre le terminal et le modem.

La masse a un rôle de référence de potentiel, tandis que la terre a généralement[1] un rôle de sécurité.

Si les deux équipements sont à l'intérieur d'un même bâtiment, les deux équipements sont reliés à la même terre par la prise 230 V : il faut en principe éviter de relier une nouvelle fois ces terres via la liaison RS232, ce qui aurait pour effet de créer une boucle, dans laquelle un courant parasite ne demande qu'à circuler.

Pour résumer : si les deux équipements RS232 sont à quelques mètres l'un de l'autre, dans le même bâtiment, il ne devrait pas y avoir de problèmes liés à la terre. Sinon, le plus sûr est d'établir une liaison utilisant une fibre optique[2], qui apporte une isolation parfaite, et est insensible à tous parasites.

Dans le cas général, masse électrique et terre sont distinctes, mais il arrive assez souvent que les deux soient reliées, par un câblage éventuellement amovible, pour tenter d'éliminer des parasites[3].

AVERTISSEMENT : le raccordement à la terre a un but de sécurité. Vous n'êtes pas dispensés de vous conformer à la norme NF C 15-100. Votre sécurité ainsi que celle de votre entourage sont en jeu.

Si la carcasse ou le chassis d'un équipement est prévu[4] pour être relié à la terre, vous ne devez **en aucun cas** rompre cette connexion.

La terre ne doit pas être improvisée (connexion à une conduite d'eau **interdite**[5]) : elle doit

1 On peut rencontrer des récepteurs de radios alimentés par piles, ne présentant pas de problème de sécurité électrique, et cependant reliés à la terre pour améliorer l'efficacité de l'antenne.
2 Ce n'est alors plus du RS232.
3 L'élimination de parasites est du ressort de la magie noire ; il faut une certaine expérience pour arriver à un résultat probant.
4 Certains appareils, dits à **double isolation** identifiés par le symbole ⧈, ne doivent **pas** être reliés à la terre. Par exemple, un sèche-cheveu grand public, ou un convecteur électrique.
5 Il est interdit d'utiliser une conduite d'eau comme terre, mais les conduites d'eau métalliques doivent être reliées à la terre.

être établie par un électricien, ce qui est le cas dans les constructions modernes et dans les constructions anciennes mises aux normes.

Dans les liaisons série, les appellations courantes sont :

Terre (protection)	PG, Protective Ground
	FG, Frame Ground
	Shield, Shield Ground
Masse (référence)	SG, Signal Ground
	GND, Ground
	Common Ground

Remarque : la broche numéro 1 du connecteur DB-25 est prévue pour être connectée à la terre de protection, alors qu'aucune broche n'est prévue à cette effet avec le connecteur DE-9. Le cas échéant, la terre est dans ce cas directement raccordée au corps métallique du connecteur DE-9.

5.6 Limites RS232

Les possibilités des liaisons série RS232 ne sont pas illimitées.

Certaines de ces limites peuvent être dépassées dans la pratique, en jouant sur la grande tolérance du récepteur.

D'autres limites peuvent partiellement être dépassées par bricolage, mais n'ont en général pas de vocation universelle.

5.6.1 Distance

Selon Wikipédia, la norme RS232 ne donne pas de distance maximale de câble, mais limite la capacité[1] du câble, cette capacité étant fonction de la nature du câble.

Cependant, une règle non écrite courante veut qu'à 9 600 bauds, le fonctionnement d'une liaison RS232 est assuré jusqu'à 15 mètres.

Si la vitesse est réduite de moitié, la distance peut être doublée :

Bauds	m
19 200	7,50
9 600	15
4 800	30
2 400	60
1 200	120
600	240
300	480

1 Capacité = condensateur parasite formé par les conducteurs et l'isolant du câble. La capacité est exprimée en Farads, ou plus couramment en ses sous-multiples nanofarads et picofarads.

De nombreuses sources citent 15 m à 19 200 bauds. En adoptant cette hypothèse, les longueurs données par le tableau précédent sont alors à multiplier par deux.

5.6.2 Vitesse

Il n'y a pas de vitesse limite basse ; on a utilisé couramment 75 bauds[1].

Les vitesses qu'il est généralement possible de paramétrer sont 300, 1 200, 2 400, 4 800, 9 600, 19 200,115 200.

On trouve aussi pour d'anciens équipements, 150, 110, et 75 bauds. L'auteur a aussi rencontré une fois dans sa vie professionnelle, la vitesse de 200 bauds.

A noter, qu'en toute rigueur le « baud » n'est pas équivalent au « bit/seconde ». Le baud est une unité de modulation, plus qu'une unité de débit. En modulant 4 bits à la fois, un modem 2 400 bauds peut assurer un débit de 4 x 2 400 = 9 600 bits/seconde.

Les plus anciens d'entre nous se rappellent qu'à l'époque héroïque d'Internet on pouvait obtenir un débit colossal de 56 000 bits/seconde en utilisant une ligne téléphonique filaire limitée par principe à ~2 400 bauds.

Le débit d'information est mesuré sans ambiguïté en « bits/seconde », mais l'usage a entériné le terme « baud » dans le contexte des liaisons série.

5.6.3 Nombre d'équipements

Le nombre d'équipements connectés à une ligne RS232 est de deux, ni plus, ni moins.

De plus, la connexion est asymétrique : les deux équipements n'ont pas le même rôle, ni le même câblage (DTE, et DCE).

Connecter 2 DTE entre eux, par exemple 2 PC, n'est pas prévu. On peut cependant tricher un peu avec un bricolage raisonnablement propre, en intercalant un null-modem dans la liaison, faisant apparaître ainsi un des DTE comme un DCE virtuel.

5.7 Adaptateurs USB/Série

Les PC ne sont maintenant plus équipés de ports série. La solution est d'utiliser un adaptateur USB/Série.

Il est malheureusement difficile d'avoir les caractéristiques de ces adaptateurs.

Les adaptateurs anonymes (*NONAME*) sont souvent fabriqués à l'économie : ils ne gèrent pas certains signaux courants, et plus grave, ils ne respectent pas les niveaux de tension RS232. Pour des cas assez simples, ces adaptateurs peuvent suffire, mais il ne font qu'ajouter une source supplémentaire de problèmes potentiels.

La prudence consiste donc à acheter une marque connue, si possible germanique. Ce n'est pas forcément plus cher[2], et dans tous les cas, cela revient moins cher qu'un adaptateur incomplet avec lequel on perd des heures.

5.8 Null-modem

Rappelons nous que RS232 a été initialement prévue pour connecter un DTE/terminal, à un DCE/modem. On retrouve, bien entendu, la même architecture de l'autre côté de la

1 Entre autres, dans le Minitel (mais ce n'était pas du RS232).
2 12 € en 2015, marque de distribution allemande.

liaison modem, en miroir.

Un null-modem est un adaptateur se substituant fonctionnellement aux deux modems.

C'est l'outil idéal pour connecter deux DTE, sans que ça ne dégénère en bricolage immonde.

La fonctions essentielle du null-modem est de croiser les fils d'émission et de réception, et certains signaux de protocole matériel.

Par dérogation exceptionnelle à notre principe qu'« un adaptateur ne doit assurer qu'une fonction », un null-modem, de par sa structure symétrique, peut être doté de deux connecteurs de même genre. En général, ces connecteurs ont 25 points, ce qui a l'effet secondaire souhaitable de rendre ces adaptateurs suffisamment gros pour qu'ils puissent être identifiés clairement. Pour un usage amateur, pour lequel les connecteurs 9 points sont plus courants, on peut trouver des null-modems 9 points, et si nécessaire, on peut en faire un assez facilement.

5.9 Adaptateurs

L'idéal est de se munir d'adaptateurs n'effectuant qu'**une seule fonction** connue :

1. 9 broches mâle, vers 25 broches femelle,
2. 9 broches femelle, vers 25 broches mâle,
3. Null-modem, mâle-femelle.

A priori un changeur de genre est inutile si l'on dispose d'un câble en nappe à 4 connecteurs (un connecteur mâle et un connecteur femelle à chaque extréminté de la nappe).

5.10 Boîte à LED

Si vous avez besoin d'établir des liaisons série à titre professionnel, la question ne se pose même pas : une boîte à LED a toutes les chances d'être amortie lors de sa première utilisation.

Ces boîtiers sont munis pour chaque signal intéressant, soit d'une LED bicolore (rouge et verte), soit de deux LED distinctes, éventuellement de la même couleur.

Dans les deux cas, l'idée est de représenter l'état du signal, soit « 0 » logique, soit « 1 » logique. Un signal non connecté à une sortie est caractérisé par l'extinction des LED.

Les boîtes étant souvent prévues pour être insérées dans une liaison 25 points, il faut aussi sans doute prévoir un adaptateur 9 points femelle / 25 points mâle, et un adaptateur 9 points mâle / 25 points femelle.

Illustration 2

© 2014 sintech adapter shop

Illustration 3

© 2014 allputer.com.

Il faut cependant être averti des limitations, en général peu gênantes, de ces appareils :

- Pour illuminer les LED, ils consomment un peu de courant prélevé sur le signal de sortie, et donc dégradent un peu les signaux. En général, la dégradation est suffisamment faible pour que les signaux restent conformes aux spécifications.
- Ils sont pratiquement inutilisables avec certains ports série bas de gamme où les signaux sont en 0-5 V ; la boîte à LED ne permet de distinguer que le « 0 » logique représenté par 5 V, le « 1 » logique est indiscernable du signal déconnecté, c'est-à-dire LED éteinte(s). De plus, si le 5 V provoque l'illumination d'une LED, la dégradation du signal sera telle que le sa tension risque de s'écrouler jusque dans la zone d'incertitude de -3 V/+3 V.

5.10.1 Conseils d'achat

Ce genre de boîte est généralement commercialisé sous le nom de « testeur RS232 ».

- Les signaux minimum à visualiser sont : TD, RD, RTS, CTS, DCD, DTR, DSR.
- Les noms de ces signaux ne sont significatifs que si la boîte à LED est connectée **directement**[1] à l'équipement, ou éventuellement avec une rallonge de type « nappe » (point-à-point).
- La boîte doit **impérativement** être équipée d'un connecteur mâle d'un côté, et femelle de l'autre. Il est ainsi possible de l'intercaler provisoirement dans une ligne, sans avoir à recourir à un changeur de genre.
- Pour un usage professionnel, préférez des connecteurs 25 points, ceux-ci équipant encore de nombreux appareils.
- Pour un usage amateur, préférer des connecteurs 9 points.
- Une telle boîte ne devrait pas coûter beaucoup plus de 15 €[2].

5.10.2 Étalonnage de la boîte à LED.

Il est curieux de parler d'étalonnage pour un matériel ne possédant aucun réglage.

En fait, comme il y a 99 % de chances que l'appareil soit fourni sans notice, ou que celle-ci

1 Pas de câbles ou d'adaptateurs susceptibles de croiser des signaux.
2 En 2016.

soit perdue, il s'agit de déterminer à quoi il faut s'attendre.

L'étalonnage consiste à identifier clairement les LED, et leur état : allumé, éteint, couleur.

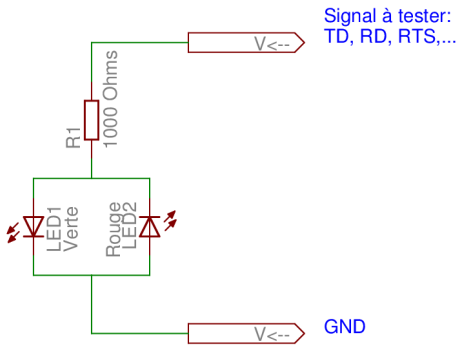
L'étalonnage se fera préférentiellement avec une liaison déjà opérationnelle, avec contrôle de flux matériel.

5.10.3 DIY* testeur RS232 minimal

Quiconque a accès à un fer à souder peut réaliser un testeur minimum, permettant de tester un signal à la fois.

Attention : les deux LED doivent **impérativement** être montées tête-bêche. Elles se protègent ainsi mutuellement contre les tensions trop élevées (>5 V).

Éviter dans le cas présent, les LED blanches et bleues, car leur tension de seuil élevée n'est pas adaptée à la plage assez large des tensions susceptibles d'être rencontrées.



La LED verte s'illumine quand un signal de protocole (RTS, DTR,...) est inactif, et la rouge quand il est actif.

LED	Tension	Etat logique	TD/RD	RTS, CTS,...
Verte	+12 V	0	Break[1]	Inactif
Rouge	-12 V	1	Repos	Actif

5.10.4 Détermination du type d'équipement DTE ou DCE.

Connecter la boîte au port série de l'équipement, soit directement, soit par l'intermédiaire d'un câble en nappe, de câbles et/ou adaptateurs dont on a ainsi la **certitude** qu'aucun croisement de signal n'est effectué.

Les LED allumées sont caractéristiques du type d'équipement. Dans le cas présent, on ne s'intéresse qu'au fait que les LED sont allumées ou éteintes. L'état logique représenté par la couleur de la LED n'est pas significatif : l'information utile est « le signal est une sortie », si la LED est allumée.

* *Do It Yourself* : Faites-le vous-même.

1 Les signaux d'émission de donnéesTD et de réception de données RD peuvent être indéfiniment à l'état repos, et momentanément dans l'état *break*. Pendant la transmission de données, les 2 LED semblent allumées simultanément : en fait, elles clignotent alternativement à grande vitesse.

Seules les LED allumées ont valeur probante : celles éteintes doivent être ignorées.

Une seule LED allumée suffit à déterminer le type d'équipement. Si plusieurs LED sont allumées, elles doivent être cohérentes, et détecter toutes le même type d'équipement.

LED allumée	Type
TD	DTE
DTR	
RTS	
RD	DCE
DSR	
CTS	
DCD	

Une, et une seule des LED TD ou RD devrait s'allumer.

Pour s'en convaincre, il suffit de connecter une boîte à LED à un équipement dont le type est connu, par exemple un PC est toujours un DTE.

5.12 Outils divers

Un multimètre *à pas cher* avec la fonction Ohmmètre peut être utile. Une sonnette (Ohmmètre qui fait *bip*, quand il mesure une très faible résistance) serait « *un plus* »[1]. La précision de ce genre d'outil n'est absolument pas critique. La sonnette permet de reconstituer le schéma interne d'un adaptateur. La fonction voltmètre n'est utilisable que sur des signaux stables, par exemple il est inutile de tenter de l'utiliser pendant qu'une émission est en cours.

Dans le chapitre des outils, on peut ajouter ces deux phrases de test :

- PORTEZ CE VIEUX WHISKY AU JUGE BLOND QUI FUME
- THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ces deux phrases ont la caractéristique d'employer toutes les lettres de l'alphabet ASCII.

Si ces phrases sont transmises sans erreur, c'est le signe d'un bon paramétrage.

A noter, que ces phrases ne contiennent que des majuscules, ce qui n'est pas très grave, et ne contiennent que des caractères non accentués, ce qui d'un côté présente un avantage, étant donné les multiples possibilités d'encodage des caractères accentués, mais d'un autre côté présente l'inconvénient de ne pas tester le 8ème bit de l'octet, l'ASCII pur ne nécessitant que 7 bits.

6 Connexions envisageables

Les connexions possibles se résument à un nombre fini de cas :

- Connexion d'un DTE à un DCE,
- Connexion d'un DTE à un autre DTE.

1 © RH.

Et pour ces deux types de connexion, il faut envisager les variantes concernant les signaux de contrôle de flux matériel :

- Les deux équipements gèrent les signaux de protocole,
- Aucun des deux équipements ne gère les signaux de protocole,
- Un des équipements gère les signaux de protocole, mais pas l'autre.

Ce qui en théorie permet 6 combinaisons, mais le cas DTE/DTE peut se ramener au cas canonique DTE/DCE très simplement, avec un null-modem, ce qui nous conduit à seulement 4 cas possibles, chacun de ces cas possédant deux variantes.

La connexion d'un DCE à un autre DCE est laissée à titre d'exercice au lecteur[1].

Par « l'équipement gère les signaux de protocole », il faut entendre que l'équipement a **besoin** de ces signaux, et pas seulement que ces signaux sont présents parce que *tout le monde le fait*.

La gestion des signaux de protocole est souvent paramétrable, au moins du côté du DTE. Dans ce cas, et si le DCE n'a pas besoin de ces signaux, alors le plus simple est de ne pas les gérer.

7 Avant de commencer

Le moins vous aurez à improviser pour établir votre connexion, le mieux cela se passera.

Munissez-vous des caractéristiques des équipements que vous voulez relier :

1. Type de connecteur :
   - DB-25,
   - DE-9[2],
   - RJ45
2. Genre du connecteur :
   - Mâle,
   - Femelle.
3. Le type de l'équipement :
   - DTE,
   - DCE.
4. Les caractéristiques logicielles de la liaison :
   - Nombre de bits (a priori : 8),
   - Parité éventuelle,
   - Nombre de bits de stop (dans le doute : 1).
5. Le contrôle de flux utilisé par l'équipement :

1 C'est littéralement un « cas d'école », que vous n'avez pratiquement aucune chance de rencontrer dans la vraie vie.
2 Souvent improprement appelé ~~DB-9~~.

- Aucun,
- Matériel (RTS/CTS),
- Logiciel (XON/XOFF).

Sur un PC, les ports séries de la carte-mère sont maintenant des connecteurs 9 points **mâles**, identifiés par un symbole « IOIOI ». Ce symbole est en général difficilement lisible.

A ne pas confondre avec un connecteur VGA, qui a la même forme, mais possède 15 contacts sur 3 rangées, et de surcroît est femelle.

Munissez vous de câbles et d'adaptateurs dont le câblage est connu : nappes, adaptateurs moulés.

Si vous ne possédez pas tous les éléments, il faut les reconstituer, **avant** de mettre en œuvre la liaison en totalité.

Certains équipements exotiques peuvent être équipés de connexions sensées vous faciliter le travail, mais qui en fait le complique, parce que **non standard**.

Cependant, sauf excellente raison, on supposera que la connexion est raisonnablement standard. Une bonne raison peut être la documentation du constructeur, ou un connecteur qui n'est ni DE-9, ni DB-25. On n'envisagera ce cas qu'en toute dernière extrémité, car même dans les cas simples, il existe quantités de raisons pour lesquelles ça ne fonctionne pas du premier coup.

8 Ça ne marche pas

8.1 Symptômes caractéristiques

Q : On ne reçoit que des caractères sans signification.

R : La vitesse est probablement mal paramétrée.

Q : La moitié des caractères reçus sont défectueux.

R : La parité est mal paramétrée.

Q : Les caractères ne sortent pas du DTE.

R : Le contrôle de flux matériel est activé dans le DTE, mais le DCE n'y répond pas.

Soit :

- Paramétrer le DTE pour ne pas utiliser le contrôle de flux matériel.
- Paramétrer le DCE pour gérer le contrôle de flux matériel.
- Etablir par câblage la connexion des signaux de contrôle de flux, entre DTE et DCE (RTS du DCE sur CTS du DCE,...)

Q : Un testeur de liaison montre que le signal TD est allumé, mais RD est éteint.

R : Les deux équipements sont probablement deux DTE. Intercaler un null-modem.

Q : Le récepteur constate que les caractères sont généralement reçus correctement, mais on perd parfois quelques caractères consécutifs.

R : L'émetteur va trop vite pour le récepteur. Il faut utiliser un contrôle de flux.

Q : Les caractères accentués sont transformés en « Ã ».

R : Ce n'est pas un problème de la liaison série : c'est un problème d'encodage. Le texte émis est encodé en UTF-8, et le récepteur le décode comme de l'ISO-8859-1.

9 Annexes

9.1 Référence des signaux

Ne pas perdre de vue que la liaison est dissymétrique, et est prévue pour connecter en point à point un équipement désigné par « DTE » à un autre équipement désigné par « DCE ».

DTE	**D**ata **T**erminal **E**quipment	Le mot-clef à retenir est « Terminal ». Un PC est **toujours** un DTE.
DCE	**D**ata **C**ommunication **E**quipment	Le mot-clef à retenir est « Communication ». C'est un modem, ou un appareil que l'on pourrait mettre à la place d'un modem.

Le nommage des signaux, ainsi que le sens du signal, sont relatifs au terminal / DTE. Le signal TD « Transmission de données » est à comprendre comme « Transmission de données **par le DTE** ».

Ce point de vue est à inverser si on se place du point de vue du modem/DCE. Le même signal TD est à comprendre comme « Réception de données par le DCE ».

Le signal TD est une sortie du DTE, mais est une entrée du DCE.

Nom	Signification	Sens	Traduction	DE-9	DB-25	Commentaire
RD, RX, RXD	**R**eceive **D**ata	DTE←DCE	Réception de données	2	3	C'est sur cette broche que le terminal/DTE émet les données, et que le modem/DCE les **reçoit**.
TD, TX, TXD	**T**ransmit **D**ata	DTE→DCE	Émission de données	3	2	C'est le signal symétrique de TD : le terminal/DTE reçoit les données sur cette broche, mais le modem/DCE **émet** sur cette broche.
RTS	**R**equest **T**o **S**end	DTE→DCE	Demande pour émettre	7	4	Si le DTE utilise le contrôle de flux matériel, il active le signal RTS, et attend que le modem/DCE réponde en activant CTS.
CTS	**C**lear **T**o **S**end	DTE←DCE	Prêt à émettre	8	5	

Nom	Signification	Sens	Traduction	DE-9	DB-25	Commentaire
DCD, CD	**D**ata **C**arrier **D**etect		Porteuse de données détectée	1	8	
DTR	**D**ata **T**erminal **R**eady	DTE→DCE	Terminal de données prêt	4	20	Indique que le terminal/DTE est prêt à recevoir des données.
DSR	**D**ata **S**et **R**eady	DTE←DCE		6	6	
GND, SG	**Grou**nd, Signal Ground	DTE−DCE	Masse électrique	5	7	Ce n'est **pas** la terre de protection. Ce n'est ni une entrée, ni une sortie : c'est le potentiel de référence des autres signaux.
	Protective Ground	DTE−DCE		N/A	1	La terre n'est pas une des 9 broches du connecteur DE-9. Certaines variantes de ce connecteur permettent de repiquer la terre directement sur le corps métallique du connecteur. A priori, il faut éviter de relier la terre des deux côtés du câble.

9.2 Encodage des caractères

Le protocole série transporte des octets de manière transparente, sans se préoccuper de ce qu'ils peuvent signifier.

Il est donc apte à transporter du binaire, ou du texte encodé de manière quelconque.

Il faut quand même avoir quelques notions de ce qui existe, pour situer l'origine du problème. Si d'un côté de la liaison série, on transmet le mot « série », et que de l'autre on reçoit « sÃ©rie », est-ce dû à un problème de configuration du nombre de bits de stop, ou bien est-ce plutôt la parité ? Ni l'un ni l'autre : le texte a été émis en UTF-8, et reçu en ISO-8859-1.

Il faut distinguer « octet » et « caractère ». Un octet, par définition, est un ensemble de 8 bits. Un caractère est une unité d'affichage[1].

Un octet permet d'encoder 256 caractères différents, alors qu'il faut pouvoir encoder au minimum 3000 caractères en chinois.

On voit tout de suite que dès qu'on s'écarte de l'alphabet latin, un octet est insuffisant pour désigner un caractère.

Les encodages les plus courants utilisés en français, sont :

9.2.1 ASCII

On dit généralement que tout fichier texte est ASCII, mais c'est souvent faux. ASCII signifie ***American*** *Standard Character Information Interchange*.

Le mot important est « American » : la langue de Steinbeck ne possède pas de caractères accentués, ni d'une manière générale ce qu'on appelle les diacritiques (« ç », « œ »,…).

L'ASCII a été étendu de nombreuses fois, de façon à pouvoir représenter les caractères nationaux, ainsi que de nombreux symboles plus ou moins utiles.

Bien entendu, ces extensions sont joyeusement incompatibles entre elles. Si vous ne savez pas quelle extension de l'ASCII a été utilisée pour écrire un texte, le résultat risque d'être aussi décourageant à lire qu'un SMS de votre ado.

La bonne nouvelle, c'est que la norme connue sous le nom d'UNICODE semble s'imposer. UNICODE affecte un numéro unique à chaque caractère, l'ASCII étant un sous-ensemble de l'UNICODE (les 127 premiers caractères).

La mauvaise nouvelle, c'est qu'il y a quand même plusieurs façons d'encoder le numéro affecté à un caractère, au choix : UTF-8, UTF-16LE, UTF-16BE, parmi les plus communes.

On croit avoir tout dit quand on précise qu'un fichier est codé en ASCII...

9.2.2 ISO-8859-1

Code normalisé sur 8 bits, extension de l'ASCII avec lequel il est compatible. Il permet d'encoder les caractères spécifiquement français, à l'exception notable de « œ ou æ », et « €[2] ». Ce code est utilisé, entre autres, par Windows.

---

1 Définition approximative, non authoritative.
2 Le code ISO-8859-15 est une variante du ISO-8859-1 qui possède le symbole « Euro ».

9.2.3 UTF-8

Encodage normalisé de l'UNICODE. L'énorme avantage de cet encodage est d'être compatible avec l'ASCII : un texte purement ASCII (sans diactritiques) est identique au même texte encodé en UTF-8. L'inconvénient est que les caractères sont encodés par un nombre variable d'octets. Cependant, la plupart des programmes sachant manipuler une extension 8 bits quelconque de l'ASCII (entre autres : ISO-8859-1) savent manipuler sans modification des chaînes UTF-8, à part pour les opérations prenant en compte la casse des caractères, du genre « conversion des caractères en minuscules ».

L'essentiel des caractères français est encodé sur un octet. Les caractères diacritiques sont encodés sur deux octets. Des caractères plus exotiques (chinois, copte,...) peuvent nécessiter trois ou quatre octets.

UTF-8 est très utilisé sur le Web, et dans le monde Unix/Linux.

9.2.4 UCS-2

Encodage normalisé de l'UNICODE, promu essentiellement par Microsoft, où tous les caractères sont encodés sur 16 bits. Le premier octet de la plupart des caractères utilisés en français vaut 0. L'avantage de cet encodage, est que **tous** les caractères ont la même taille : 16 bits. Les inconvénients sont que la taille du texte est doublée (pour le français), et que ce code n'est pas directement compatible avec l'ISO-8859-1, ou l'ASCII : il faut en passer par des outils de conversion comme « `iconv`[1] ».

Microsoft ajoute à la confusion, en désignant cet encodage par « Unicode », alors que ce n'est **qu'un** encodage, parmi de nombreuses possibilités.

9.2.5 UTF-16

Microsoft a maintenant évolué vers l'UTF-16 qui est compatible avec l'UCS-2, mais les caractères sont de taille variable : ils peuvent utiliser un ou deux mots de 16 bits (de manière similaire à l'UTF-8, mais avec des octets).

9.3 Logique positive / négative

Nous avons évoqué le fait que les signaux RS232 étaient en logique négative.

L'utilisateur moyen n'a pas à prendre ce détail en considération, jusqu'à ce qu'il veuille observer électriquement la liaison, soit avec un oscilloscope, un voltmètre, où même de simples LED.

Un bit peut prendre deux valeurs : 0, ou 1.

Ces deux valeurs peuvent être associées à des états considérés conventionnellement comme complémentaires :

- vrai ou faux,
- interrupteur ouvert ou fermé,
- lampe allumée ou éteinte,
- couleur noire ou blanche,

1 Réservé aux bienheureux Linuxiens. Les Windowsiens peuvent se racheter en utilisant `iconv` sous Cygwin https://www.cygwin.com/.

- …

La représentation des bits en électronique fait assez naturellement appel à deux tensions.

La technologie dite « TTL » des années 1960/1970 utilisait une seule tension d'alimentation de +5 V, par principe de conception.

Assez *logiquement*, cette tension de +5 V représentait la valeur « 1 », le « 0 » étant représenté par une absence de tension, ou plus exactement une tension de 0 V.

Cette convention, intuitive, est dénommée « logique positive ».

L'électronicien ou l'informaticien n'a pas à se faire de nœuds au cerveau : s'il observe à l'oscilloscope un signal qui commence à 0 V et fait ensuite un bond à +5 V, il observe une transition 0->1.

Y a-t-il une seule raison pour adopter une autre convention, à moins d'avoir l'esprit tordu ?

Les circuits TTL ou assimilés sont assez fragiles : il est hors de question de les exposer au monde extérieur hostile.

On intercale donc entre les circuits TTL et l'extérieur, des circuits faisant l'interface avec les niveaux RS232.

Le rôle de ces circuits d'interface est purement électrique : il faut adapter les niveaux de tension TTL 0-5 V, en niveaux RS232 de +12 V / -12 V, délivrer le courant spécifié par la norme, et d'être insensible à toutes sortes de courts-circuits.

Ces circuits n'ont pas vocation à transformer davantage les signaux : la forme du signal d'entrée est identique à celle du signal de sortie. C'est ce qu'on appelle un circuit tampon (*buffer*).

Les technologies classiques à base de transistors (TTL, MOS) permettent de faire plus facilement un circuit qui inverse son entrée[1], qu'un circuit qui n'inverse pas. C'est sans doute la raison qui a guidé ce choix.

C'est ainsi que le signal se retrouve la *tête en bas* : vu à l'oscilloscope, « 0 » logique est en haut, et « 1 » logique est en bas : le signal est en logique négative.

Comme le signal est inversé en sortant du premier ordinateur, puis inversé à nouveau par un circuit similaire en entrant dans le second ordinateur, le signal est rétabli dans sa logique positive. Il n'y a qu'à l'extérieur de l'ordinateur qu'il est en logique négative.

9.5 Utilisations détournées des liaisons RS232

9.5.1 Alimentation du dispositif connecté

Avec quelques diodes, on peut récupérer quelques milliampères sur les signaux de contrôle de flux RTS,…

Le courant disponible peut être augmenté, en utilisant plusieurs signaux.

Certains micro-modems fonctionnent selon ce principe, sans nécessiter d'alimentation.

1 Accessoirement, les portes logiques inverseuses (NAND, NOR) sont universelles : on pourrait construire un ordinateur (pour le *fun*) rien qu'avec des circuits NAND (en prévoir une grande quantité quand même).

9.5.2 Entrées/Sorties

Nous avons vu que les ports RS232 équipant les PC étaient dotés de signaux destinés à gérer le contrôle de flux.

Certains de ces signaux de contrôle sont des entrées, d'autres des sorties.

Tous ces signaux sont sous le contrôle du logiciel : on peut donc imaginer qu'une application ayant besoin d'un ou deux signaux d'E/S pourrait détourner ces signaux de contrôle.

Avantage : les ports RS232 sont très robustes, il est difficile de les détériorer.

Inconvénient : le niveau de tension des sorties varie énormément d'un PC à l'autre. Pour certaines applications peu regardantes, ceci peut être réglé avec une résistance et une diode Zener.

Pour disposer d'un peu plus d'entrées/sorties, et profiter d'un niveau de tension classique 0-5 V, on peut utiliser un port parallèle au lieu d'un port série. Les inconvénients sont que les ports parallèles sont devenus encore plus rares que les ports série, les entrées sorties sont nettement moins robustes, et il faut sans doute recourir à de la programmation de très bas niveau, avec même un peu d'assembleur.

Si les entrées/sorties ont besoin d'un *timing* précis[1], les adaptateurs USB/Série sont à éviter, car ils introduisent un retard non négligeable dans ce cas, dû au principe de scrutation[2] de l'USB, et de surcroît ce retard est peu déterministe.

9.6 Mise en œuvre de PuTTY

Un logiciel d'émulation de terminal est pratiquement indispensable pour mettre au point une liaison série.

Pour faire simple, ce genre de logiciel permet de paramétrer la liaison série, et envoie sur cette liaison ce qui est frappé au clavier, et affiche ce qui est reçu sur cette même liaison série.

Windows venait en standard avec HyperTerminal jusqu'à Windows XP.

HyperTerminal est relativement simple à mettre en œuvre, abstraction faite des questions sans objet concernant des préfixes téléphoniques régionaux, qui ont pour principal effet de perturber l'utilisateur moyen…

Si vous n'avez pas HyperTerminal, ou si vous n'êtes pas familier avec ce logiciel, PuTTY est un autre logiciel d'émulation de terminal sous Windows, libre (le code source est disponible).

PuTTY a plus de fonctionnalités que la simple émulation de terminal (en particulier, c'est aussi un client SSH, c'est-à-dire qu'il permet d'établir une liaison chiffrée avec un équipement, si celui-ci possède la fonction « serveur SSH »), mais pour cette simple utilisation le fait que le logiciel soit en anglais ne devrait pas poser de problème, à l'aide du présent document.

9.6.1 Téléchargement

Un conseil de prudence élémentaire : d'une manière générale, ne téléchargez de logiciel

1 Exemple typique : le programmateur de PIC JDM, et tous ses descendants.
2 *Polling*.

que depuis le site original ou depuis des sites de téléchargement réputés.

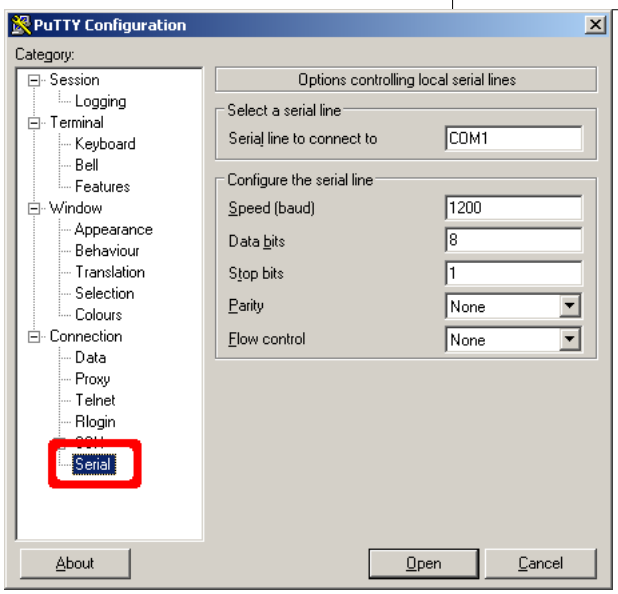
Le site de l'auteur principal est en anglais. Eventuellement, faites-vous aider par Google pour le traduire.

Plusieurs possibilités sont offertes par la page de téléchargement http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html : soit télécharger l'ensemble des fichiers associés emballés dans un *installer* ou dans un fichier ZIP, soit ne télécharger que l'exécutable `putty.exe` proprement dit, car il est autosuffisant.

9.6.2 Utilisation

1. Lancer PuTTY.
2. Développer la branche « Connection ».
3. Sélectionner « Serial » sous « Connection ».
4. Dans le panneau de droite, entrer les caractéristiques du protocole.
   Si possible, commencer avec des caractéristiques suivantes :

Serial line	COM1
Speed	1 200
Data bits	8
Stop bits	1
Parity	None
Flow control	None



1. Cliquer sur « Open ».
2. TODO paramétrer écho

9.7 Survol de RS485

RS485 est une alternative à RS232 : cette norme ne couvre que la partie électrique.

RS485 permet le transfert de bits en série et de signaux tout comme RS232. Il existe dans le commerce des convertisseurs RS232/RS485.

RS485 est matériellement un peu plus compliquée que RS232 : pour chaque signal transporté (TD, RD, RTS,…), il faut maintenant deux fils, et non plus un fil par signal et un fil commun à tous les signaux comme en RS232.

De plus, ces deux fils (par signal) devraient être constitués en une paire torsadée. L'emploi de câbles en nappe est exclu pour des longueurs significatives.

L'avantage de RS485 est de pouvoir couvrir des distances importantes (> 1000 m), de permettre des débits plus importants, et surtout de pouvoir connecter jusqu'à 32 équipements, à condition bien sûr que la couche logicielle ait prévu ce cas.

10 Licences

Ressource	Source	Licence
Illustrations		Domaine publique
RS232 Data Interface a Tutorial on Data Interface and cables		© 2010 DCE
Icônes diverses	Tango	
Symbole « double isolation »	https://upload.wikimedia.org/wikipedia/commons/f/f7/Double_insulation_symbol.svg	By Roman Tworkowski et al (Own work) [Public domain], via Wikimedia Commons