

# Communication série



GIF-1001 Ordinateurs: Structure et Applications  
Jean-François Lalonde

# Série vs parallèle

- Série: on envoie un bit à la fois
- Parallèle: on envoie plusieurs bits à la fois
- Lequel de ces deux types est le plus rapide?
- De quel type sont les bus suivants:
  - PCI-Express (pour les périphériques rapides dans vos PCs)?
  - USB?
  - Thunderbolt (mélange entre PCIe et DisplayPort)?

# Série vs parallèle

- La majorité des bus à l'intérieur des ordinateurs modernes adoptent des protocoles de type **série**.
- Pourquoi?
  - Des signaux électriques qui partent en même temps d'un transmetteur n'atteignent pas le receveur en même temps si les fils qui propagent les signaux n'ont pas la même longueur.
  - Il est très difficile de s'assurer que les traces (fils sur une carte électronique) aient tous la **même** longueur!
  - Il est possible que des bits d'un fil arrivent décalés par rapport au bits des autres fils (surtout à haute fréquence)

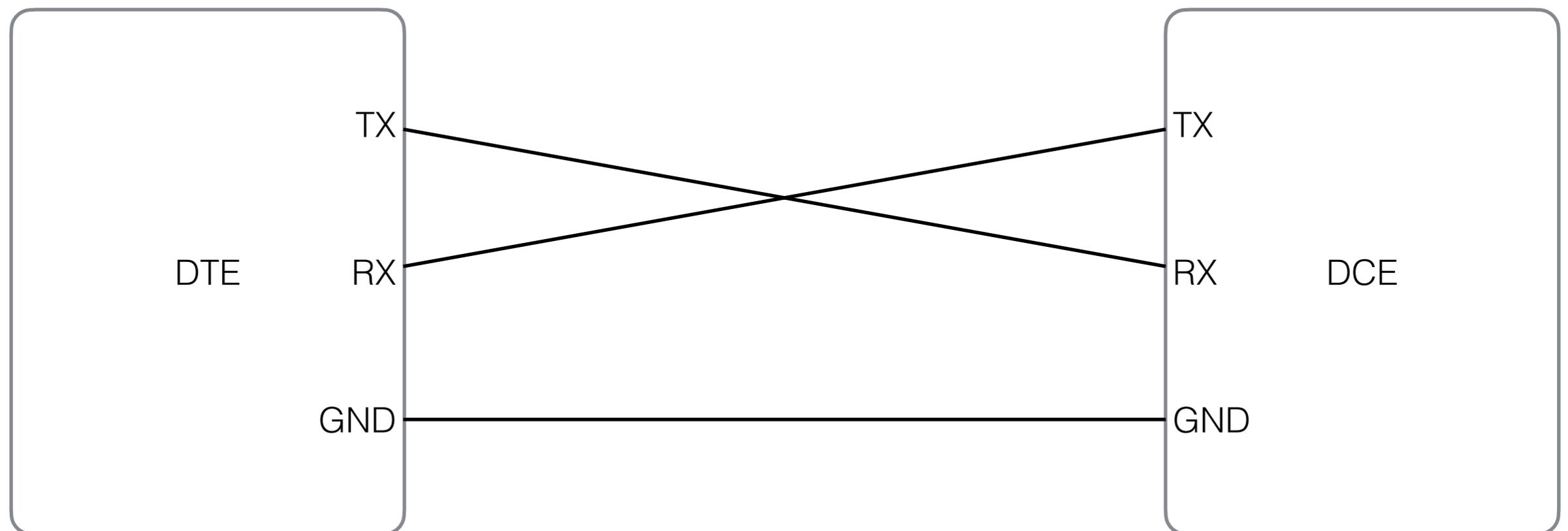
# Protocole série RS-232

- Introduit en 1960 (!), mais présent dans les ordinateurs personnels jusqu'au début des années 2000.
- Pour sa simplicité, le protocole est toujours utilisé pour des applications industrielles.
- La base pour comprendre des protocoles plus modernes, comme le USB.

# Protocole série RS-232

DTE (*Data Terminal Equipment*)  
par exemple: ordinateur

DCE (*Data Communication Equipment*)  
par exemple: modem

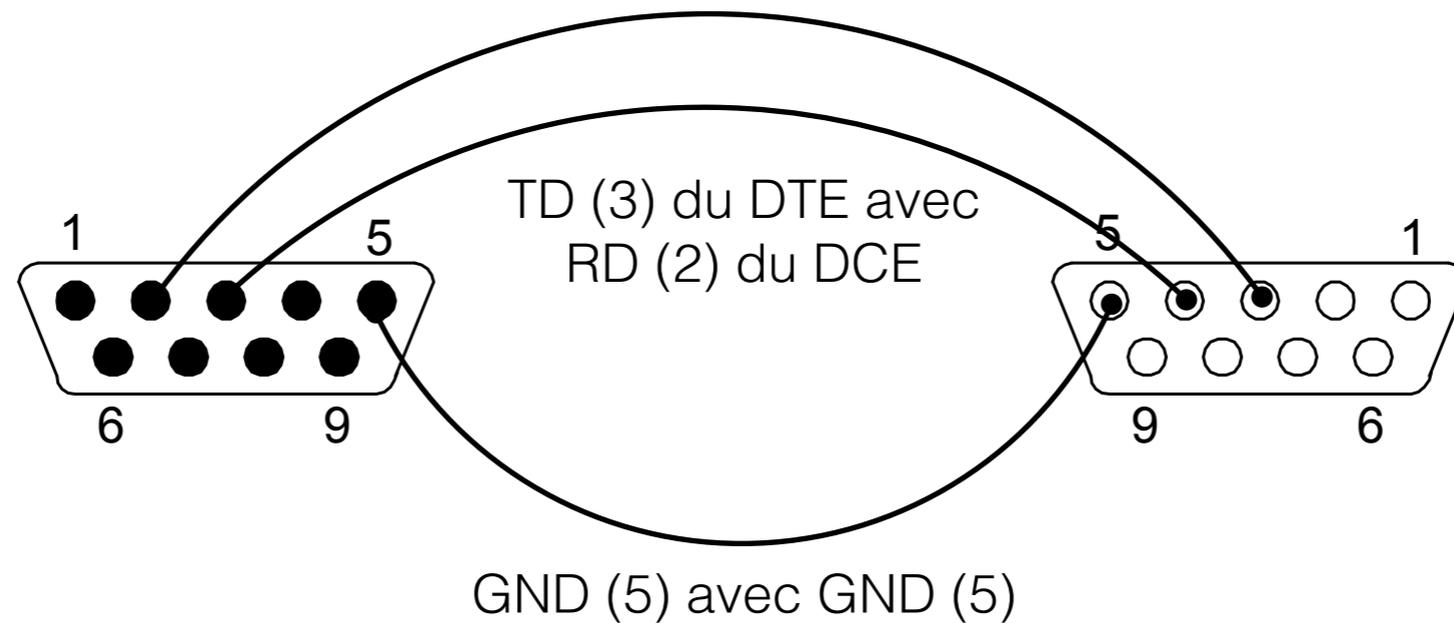


# Matériel et connecteur

DTE (*Data Terminal Equipment*)  
par exemple: ordinateur

RD (2) du DTE avec  
TD (3) du DCE

DCE (*Data Communication Equipment*)  
par exemple: modem

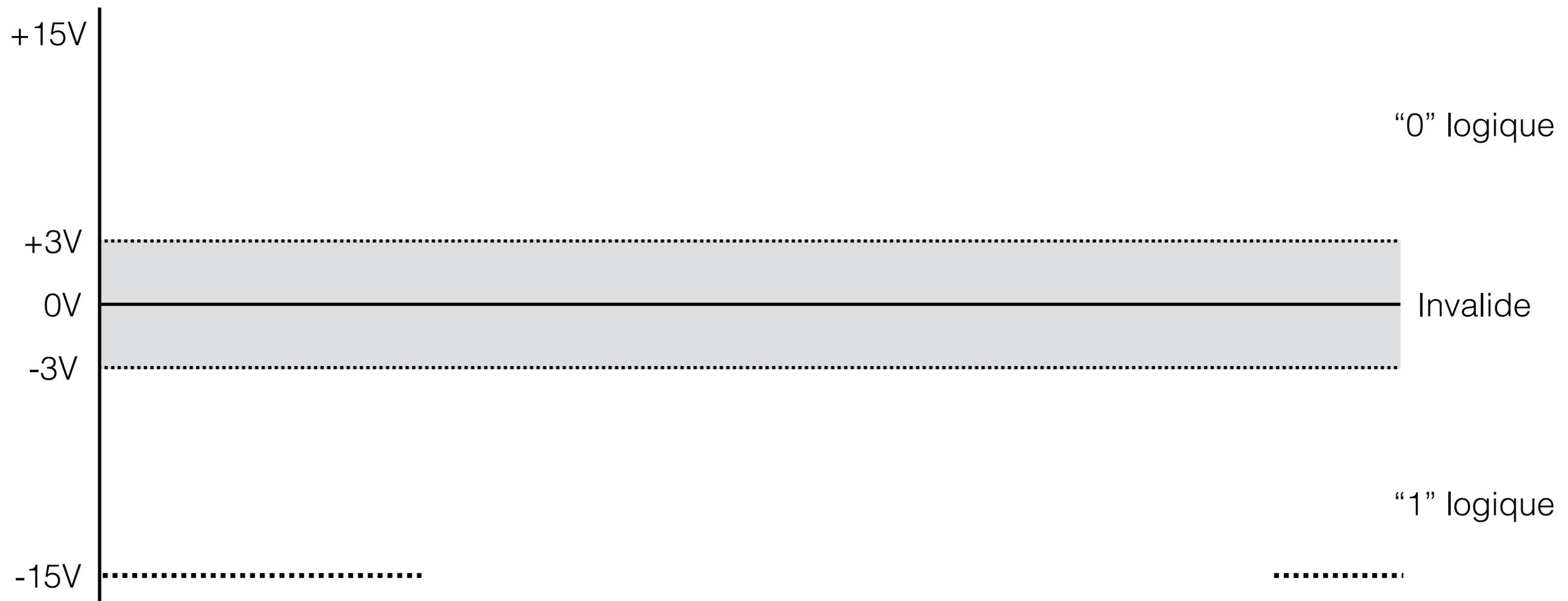


# Protocole de communication

- Le principal protocole de communication utilisé sur le port série est le RS-232. Cette spécification détermine:
  - Les caractéristiques des signaux électriques transmis (voltages, vitesse, transitions, longueurs de fils, etc.).
  - Le connecteur utilisé.
  - Les fonctions de chaque partie du port.

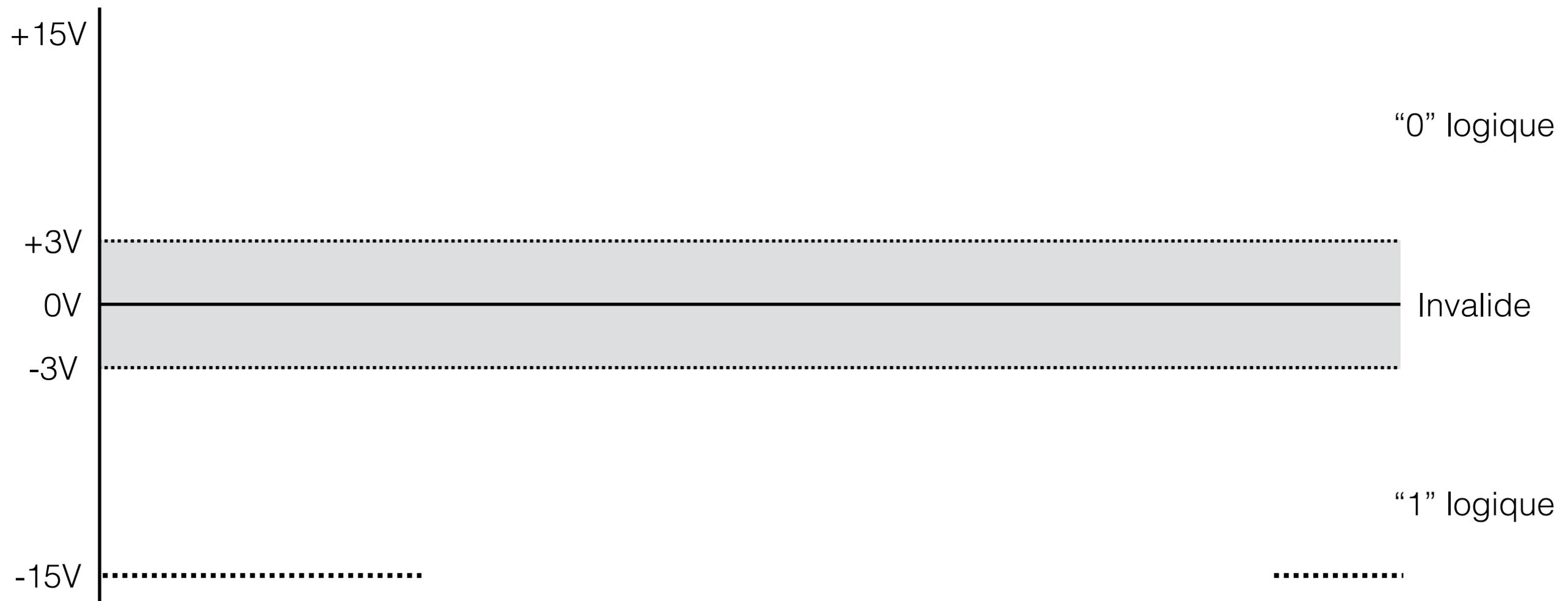
# Signaux

- Le signal transmis sur les pins RD et TD va de -15V à +15V:
  - entre +3V et +15V, il est interprété comme un **0** logique;
  - entre -3V et -15V, il est interprété comme un **1** logique;
  - entre -3V et +3V, le signal est considéré invalide.
- Lorsqu'on a rien à envoyer, on envoie un **1** en continu



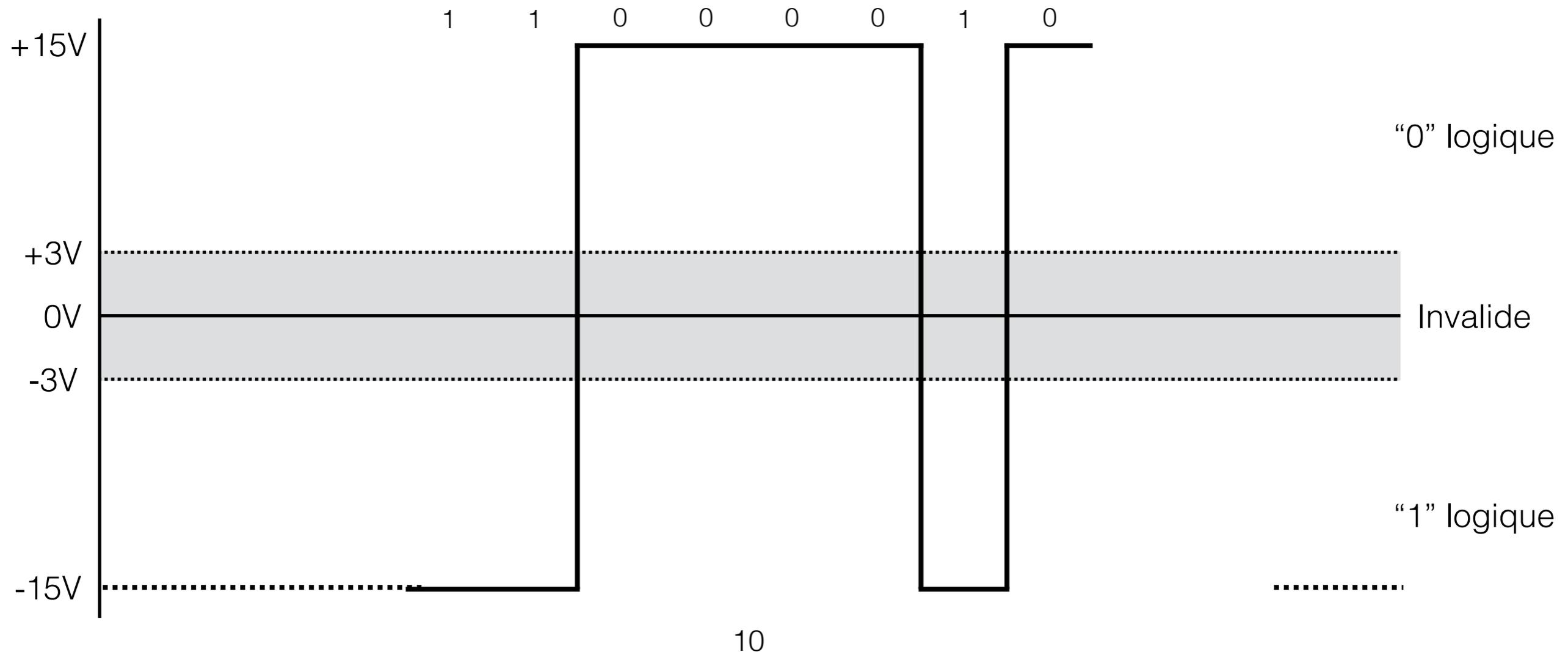
# Signaux

- On transfère un octet (8 bits) à la fois, du bit le **moins** significatif au **plus** significatif (**LSB** au **MSB**—à l'envers!)
  - Par exemple, transmettons l'octet correspondant à la lettre « C » en ASCII
  - $C = 0x43 = 0b01000011$



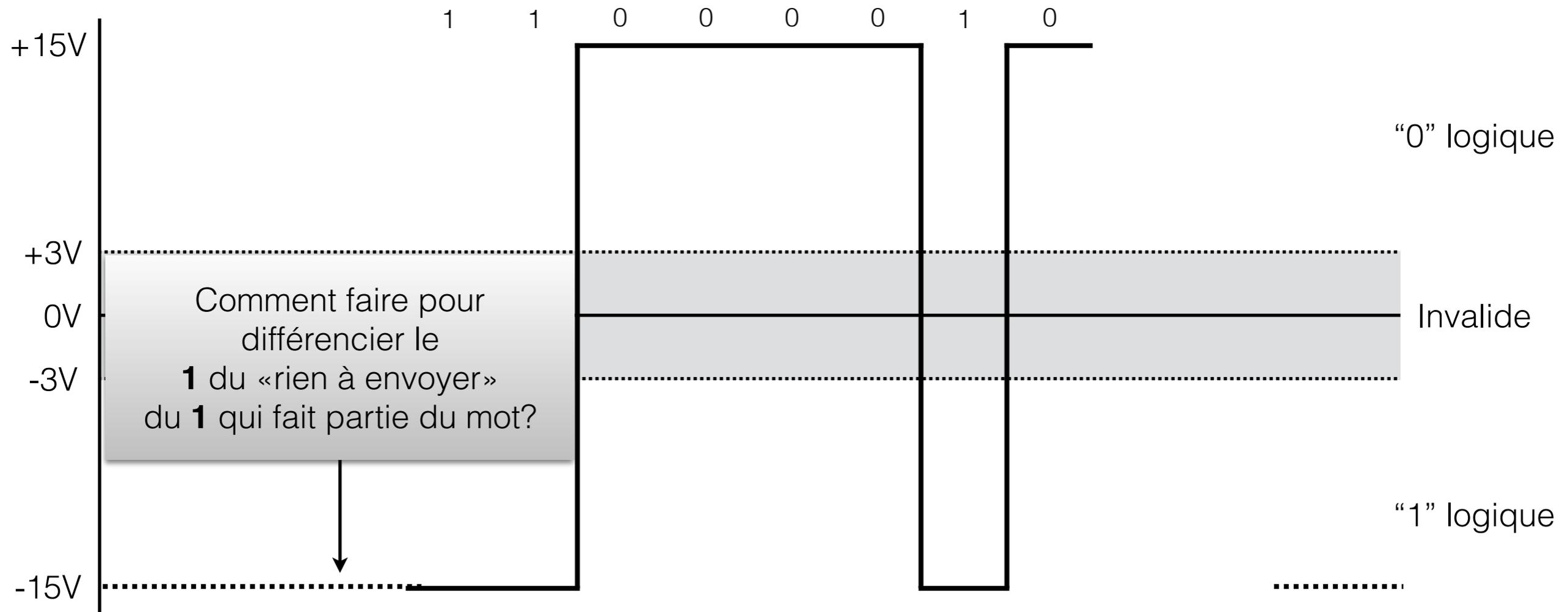
# Signaux

- On transfère un octet (8 bits) à la fois, du bit le **moins** significatif au **plus** significatif (**LSB** au **MSB**—à l'envers!)
  - Par exemple, transmettons l'octet correspondant à la lettre « C » en ASCII
  - $C = 0x43 = 0b01000011$



# Signaux: point de départ

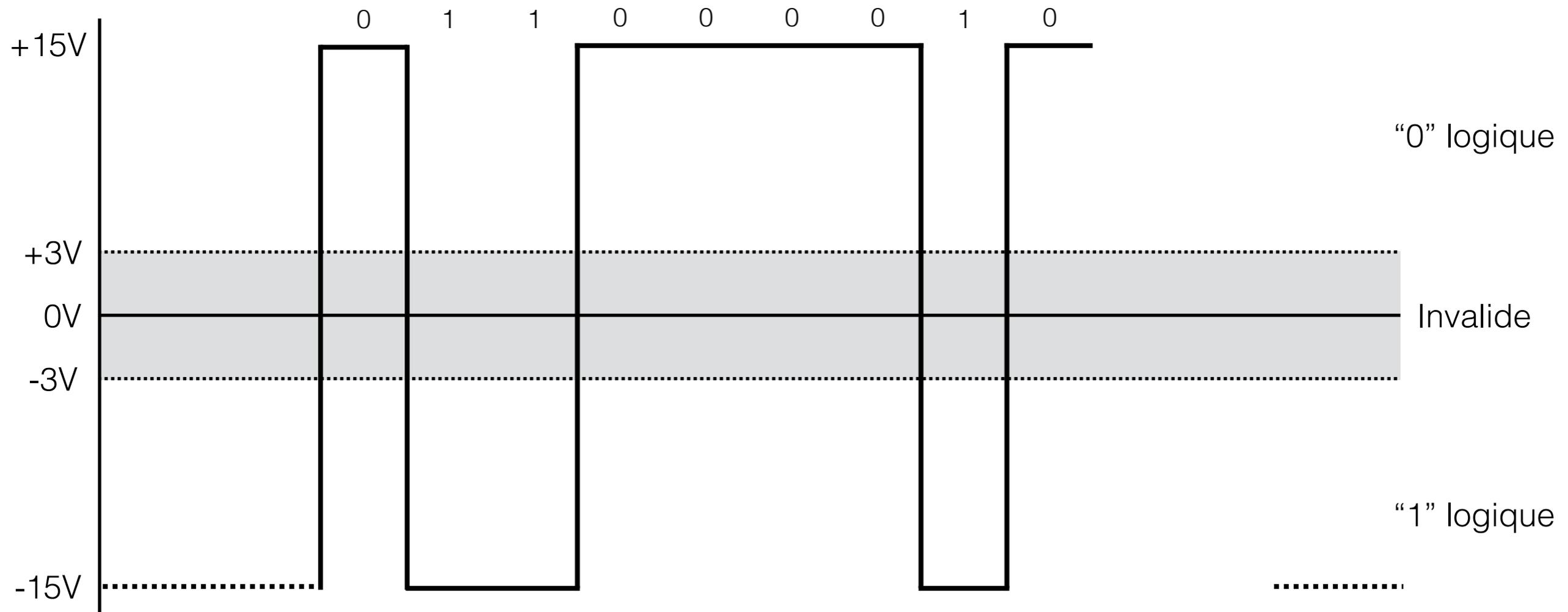
Comment faire pour savoir qu'on commence à envoyer un mot?



# Signaux: point de départ

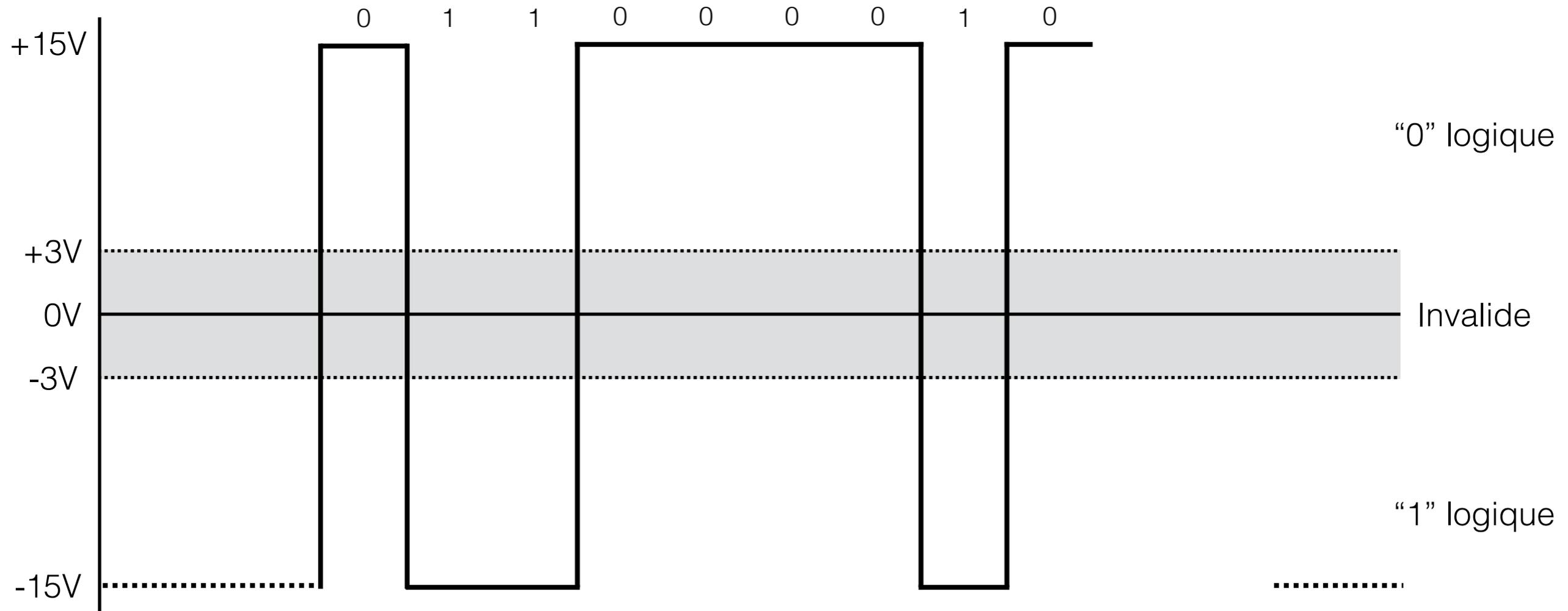
Comment faire pour savoir qu'on commence à envoyer un mot?

On envoie un **bit de départ** (*start bit*) qui met le signal à 0.



# Signaux: erreur de transmission

Comment faire pour savoir s'il y a eu une erreur de transmission?



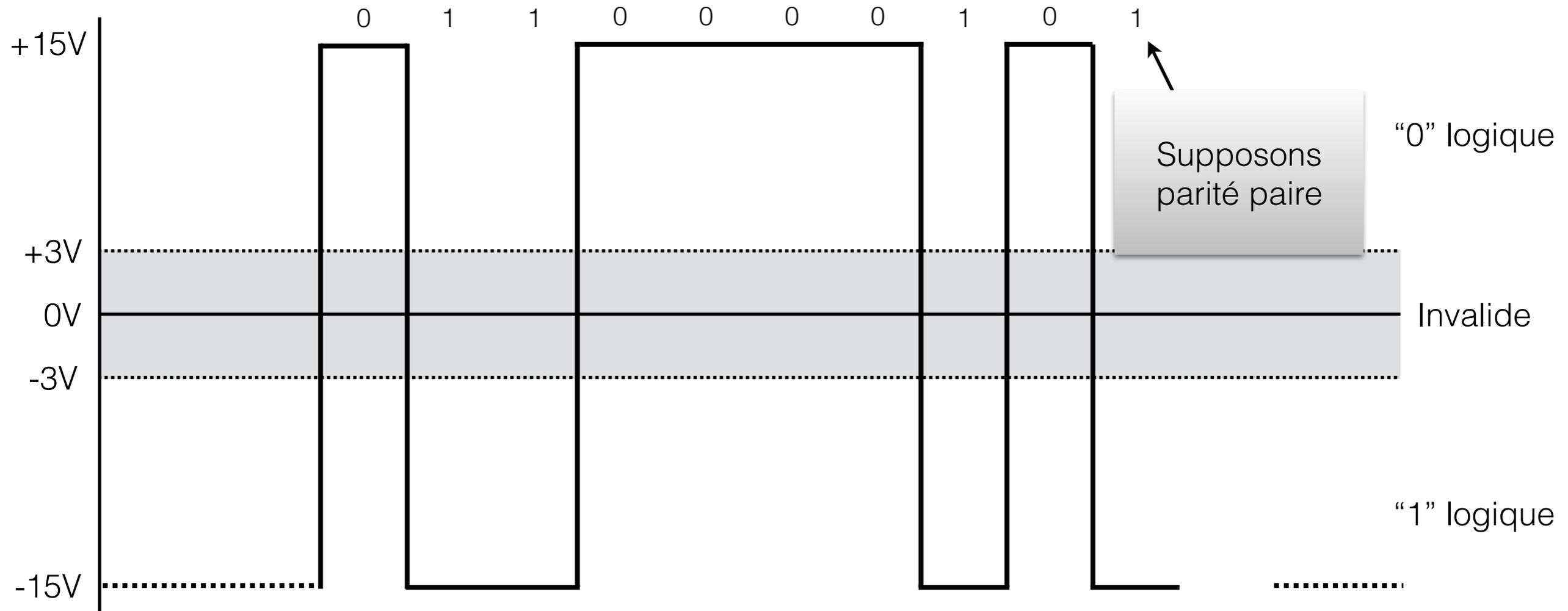
# Bit de parité

- Le bit de parité sert à **vérifier** s'il y a eu des erreurs dans l'octet transmis
  - on ne peut pas corriger l'erreur!
- En transmission, on compte le nombre de fois que **1** apparaît dans l'octet transmis, et on ajuste le bit de parité selon que l'on soit
  - en **parité paire**: le bit de parité est mis à 1 pour que le nombre total de 1 (**incluant** le bit de parité) soit **pair**.
  - en **parité impaire**: le bit de parité est mis à 1 pour que le nombre total de 1 (**incluant** le bit de parité) soit **impair**.
- En réception, on compte le nombre de 1, puis on vérifie si le bit de parité est bon.

# Signaux: bit de parité

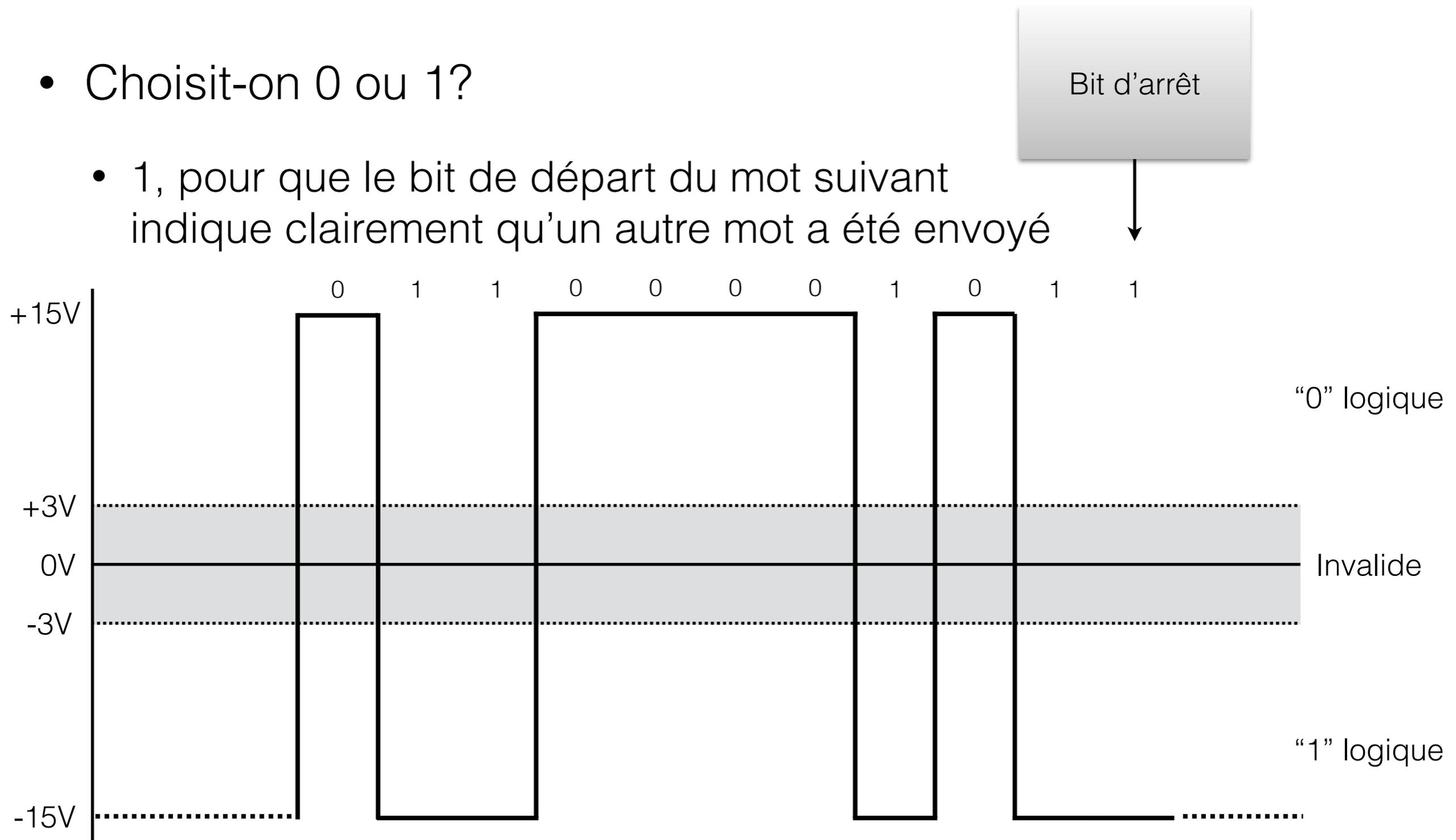
Comment faire pour savoir s'il y a eu une erreur de transmission?

On ajoute un **bit de parité** en fonction du nombre de **1** dans le mot



# Signaux: fin du mot

- On conclut le mot avec un **bit d'arrêt**.
- Choisit-on 0 ou 1?
  - 1, pour que le bit de départ du mot suivant indique clairement qu'un autre mot a été envoyé



# Port série: on « emballe » les données



Bit de départ

Bit d'arrêt

Bit de parité

# Paramètres du port série

- **Baud rate:** La fréquence des bits transmis sur le port série.
  - Les fréquences disponibles sont pré-établies: 300bps, 600bps, 1200bps, ... 19200bps, 38400bps, etc. Défaut = 9600bps
- **Parité:** Le bit de parité sert à vérifier s'il y a eu des erreurs dans l'octet transmis (on ne peut cependant pas *corriger* l'erreur)
  - Notre choix: paire ou impaire
- **Bits d'arrêt:** Nombre de bit d'arrêt (1) qui suivent le byte transmis.
  - Défaut = 1.
- **Nombre de bits par octet:** Nombre de bits transmis par octet.
  - Peut être 5, 6, 7, 8 et 9 (très rare!). Défaut = 8.

# Exercice

- Transmettre le caractère 'j' en ASCII (0x6A) sur le port série.
- On emploie la configuration suivante:
  - mot de 7 bits (LSB en premier);
  - parité paire;
  - 1 bit d'arrêt.
- Questions:
  - Quelle séquence de bits sera-t-elle transmise?
  - Si la vitesse est de 100 bps, combien de temps prendra la transmission de cette séquence de bits?

# Exercice

- Transmettre le caractère 'j' en ASCII (0x6A) sur le port série.
- On emploie la configuration suivante:
  - mot de 7 bits (LSB en premier);
  - parité paire;
  - 1 bit d'arrêt.
- Questions:
  - Quelle séquence de bits sera-t-elle transmise?
  - Si la vitesse est de 100 bps, combien de temps prendra la transmission de cette séquence de bits?

# Exercice

- Transmettre le caractère 'j' en ASCII (0x6A) sur le port série.

- On emploie la configuration suivante:

- mot de 7 bits (LSB en premier);
- parité paire;
- 1 bit d'arrêt.

- Questions:

- Quelle séquence de bits sera-t-elle transmise?

- La séquence sera:

départ	caractère (0x6A sur 7 bits, LSB en premier)							parité	arrêt
0	0	1	0	1	0	1	1	0	1

- Si la vitesse est de 100 bps, combien de temps prendra la transmission de cette séquence de bits?

- $10 \text{ bits} / 100 \text{ bits/s} = .1\text{s}$

# Problème?

Qu'arrive-t-il s'il y a du bruit sur la ligne de communication?

# Mode différentiel

- Les bits transmis sont encodés en mode différentiel. Pourquoi?
  - La différence de tension entre deux signaux propagés sur deux lignes différentes détermine la valeur d'un bit transmis.
  - Le bruit commun sur les deux lignes propageant le signal est éliminé lorsque la différence est effectuée.
  - Lorsque la différence est nulle, le bit est invalide ou une autre information peut être transmise.

# Récapitulation

- Tous les bus modernes emploient le mode de communication **série**
  - On envoie un bit à la fois de façon séquentielle
- On « emballe » le mot à envoyer avec:
  - bit de départ, bit d'arrêt, bit de parité
- Le **mode différentiel** est plus robuste au bruit
  - Nécessite cependant deux lignes
  - Absent dans RS-232, introduit dans son successeur RS-485