IUT
CERGY-PONTOISE

Périphériques

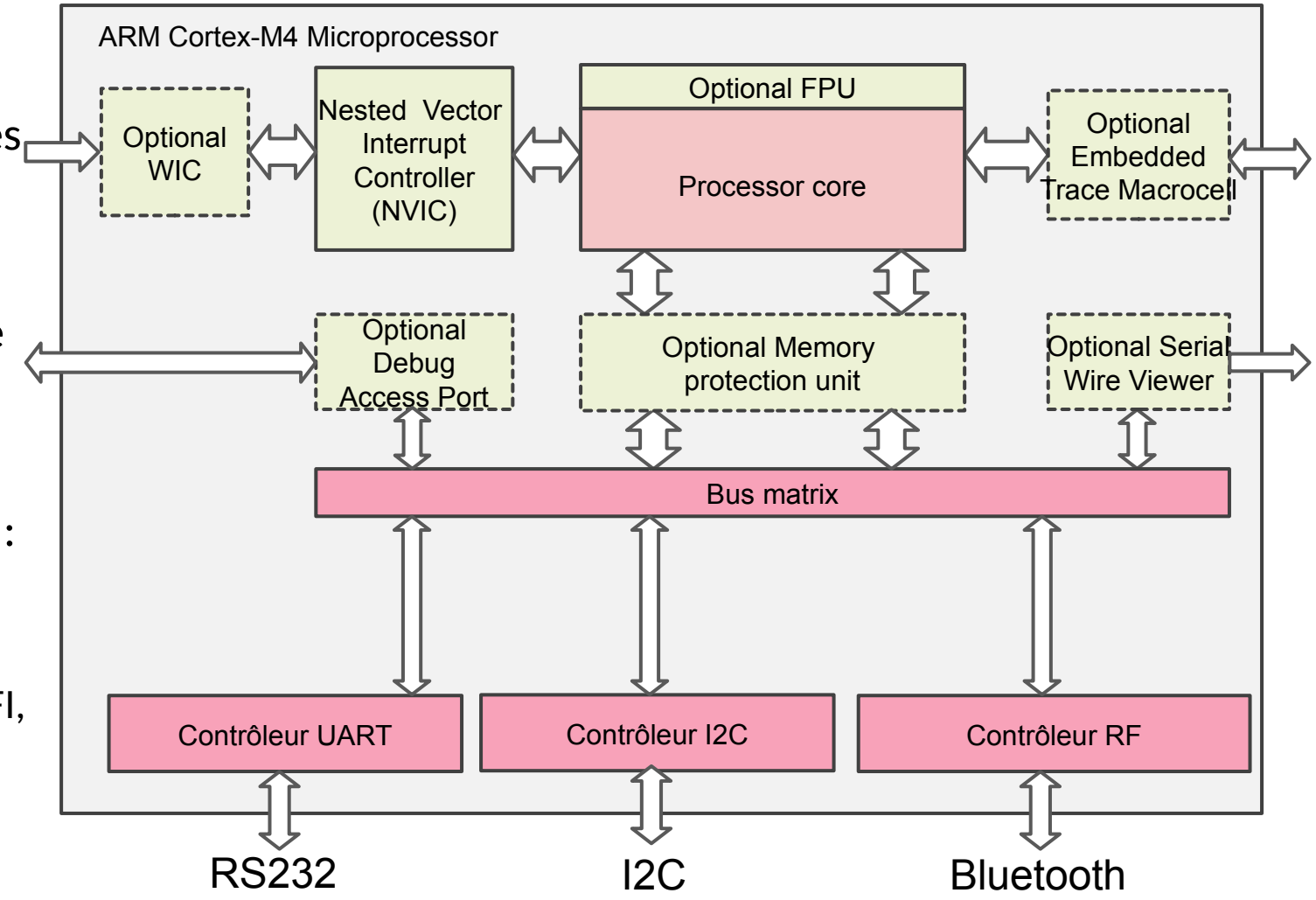
BUS
UART

Sommaire

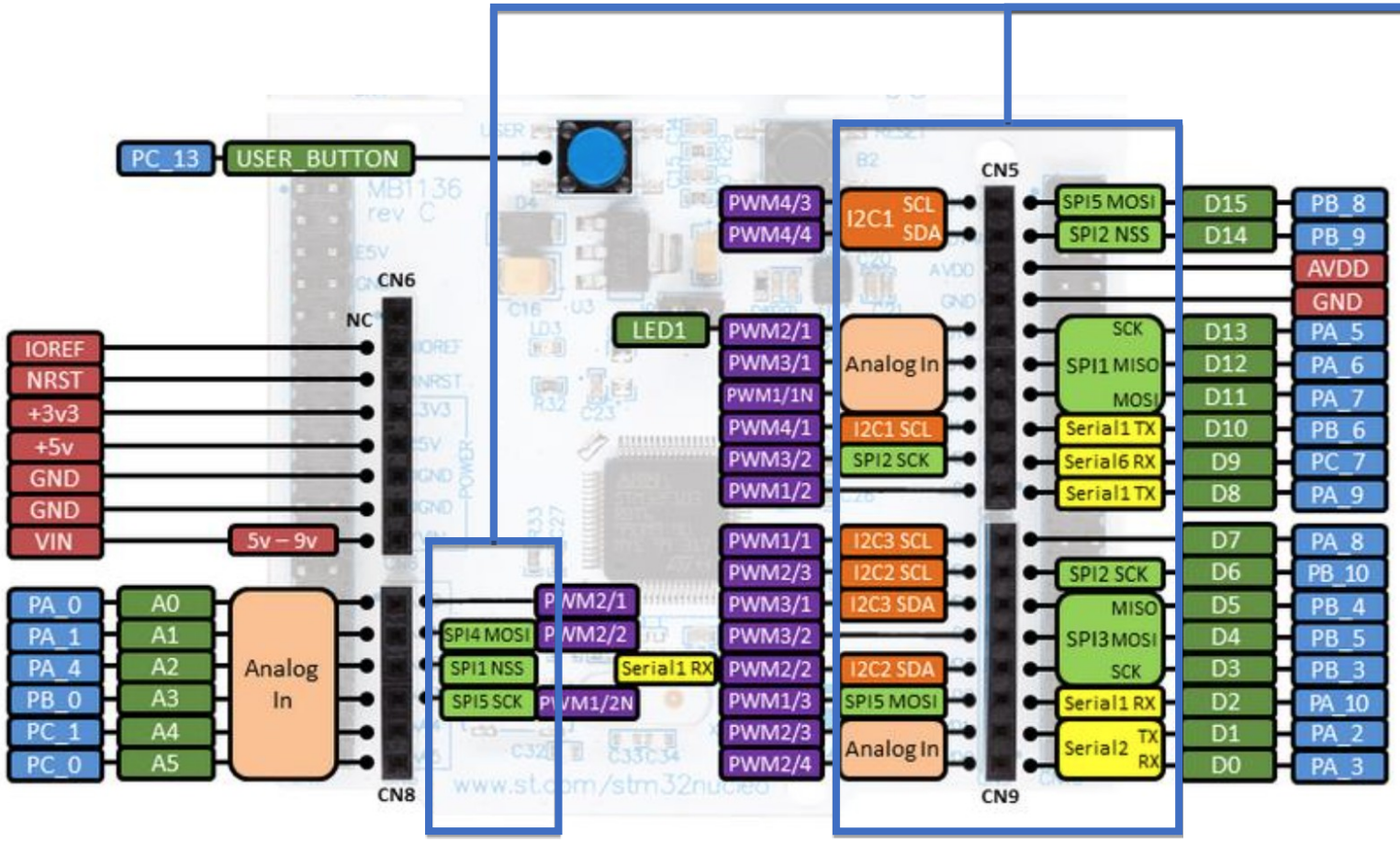
- **Généralités sur les bus**
- **Partie 1 : UART**
  - Fonctionnalité
  - Registres de configurations
  - Exemple de programmation
- **Partie 3 : I2C**
  - Fonctionnalité
  - Registres de configuration
  - Exemple de programmation
- **Partie 2 : SPI**
  - Fonctionnalité
  - Registres de configuration
  - Exemple de programmation

Mode de communication

- Aujourd'hui, il existe plusieurs modes de communication entre un processeur et un/des périphérique(s)
- Généralement, un périphérique communique avec le processeur via un bus système
- Le périphérique de communication peut être :
  - Une interface de communication série
  - Une interface de communication parallèle
  - Une interface de communication RF (BLE, WIFI, 4G, etc.)



Les bus du STM32



La carte Nucléo STM32 possède :

- ❑ 3 UARTS
- ❑ 3 I2C
- ❑ 5 SPI

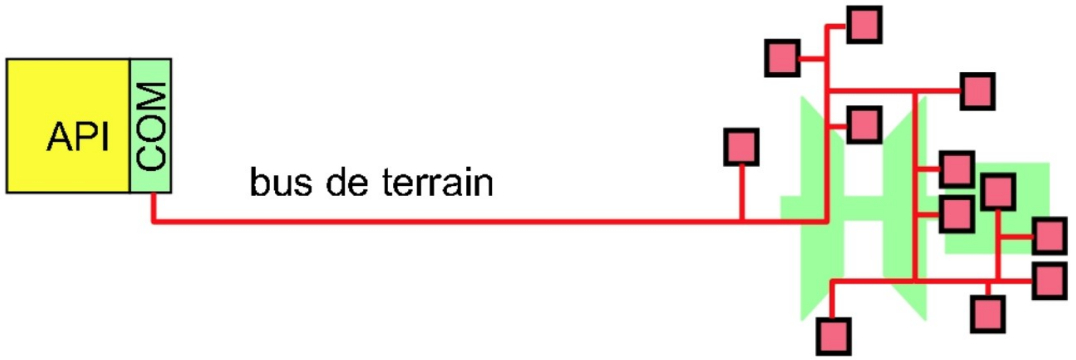
Mode de communication

- Un bus est un protocole, composé de règles qui gouvernent les modes de communication entres plusieurs dispositifs connectés.

- Un bus est composé de signaux de donnée et de contrôle

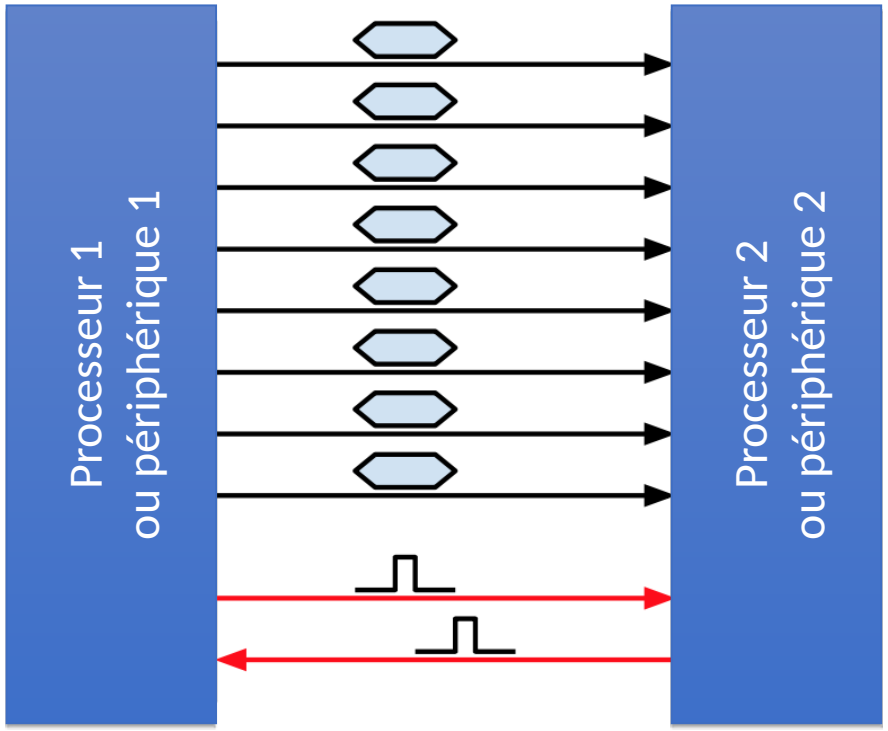
- Un bus peut être asynchrone ou synchrone
  - Dans un bus synchrone, l'horloge est distribuée avec les données
  - Dans un bus asynchrone, l'horloge est récupérée à travers les données envoyées

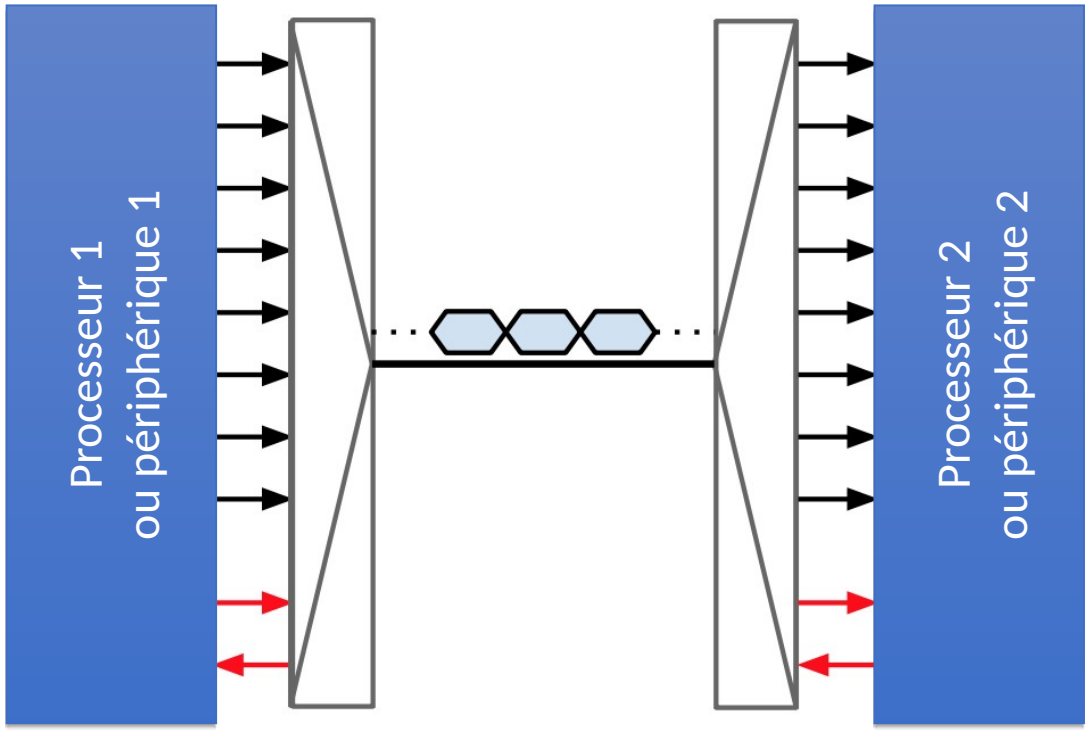
- Suivant le protocole, plusieurs appareils peuvent être connectés sur le même support physique
  - Bus de terrain



Mode de communication

- **Parallèle**
  - Donnée sur N fils en //
- **Série**
  - Donnée sérialisée/désérialisée sur 1 fil





Mode de communication

Parallèle

- Bus de N fils en parallèle
- Tous les signaux doivent arriver en même temps (synchrone)
- Limité par le nombre d'IO du processeur
- Simplicité de mise en œuvre
- Débit important
- Courte distance
- Tend à être abandonné

Série

- Bus de 2 à 3 fils en parallèle
- Signaux asynchrone
- Permet plus de bus en parallèle
- Débit important
- Longue distance
- Complexité dépend du protocole
- Beaucoup de nvx protocoles (USB3.0, USBC, Thunderbolt, etc.)

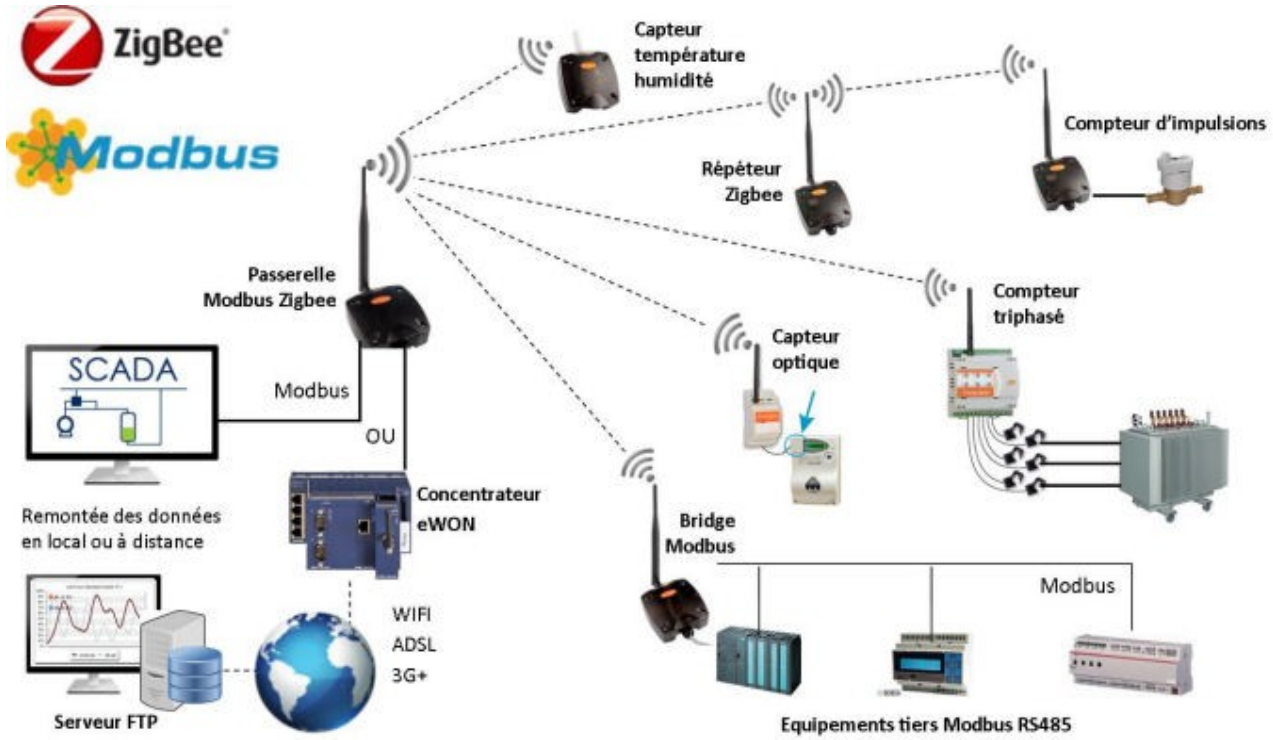
Horloge de transmission

- La transmission se fait :
  - Soit en **synchronisme** avec **une horloge de référence** commune au 2 systèmes et transmise sur une ligne supplémentaire :
    - Exemple : liaison SSP du PIC (Synchronous Serial Port).
  - Soit de façon indépendante sans horloge de référence :
    - La vitesse de transmission doit être identique sur une même ligne qui relie les circuits d'émission et de réception. Par contre elle n'est pas forcément la même sur les 2 lignes :
      - Exemple liaison USART du PIC (Asynchronous Synchronous Receiver Transmitter)

Bus de communication

- Pas de bus universel pour toutes les configurations de communication

- Comment choisir ?
  - Le choix dépend :
    - De la distance entres les équipements,
    - Le nombre d'équipements,
    - Du support physique de transmission,
    - Du débits,
    - De l'immunité au bruit,
    - De la consommation,
    - Du choix de modulation,
    - Du coût,
    - Etc….



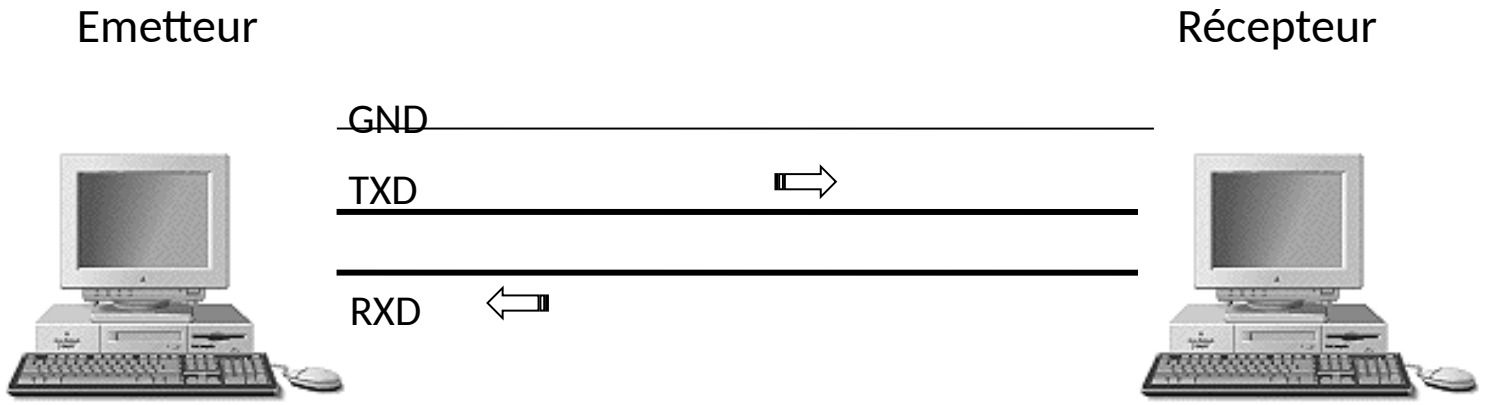
IUT
CERGY-PONTOISE

Liaison série: UART

Bus de communication série

Principe de fonctionnement

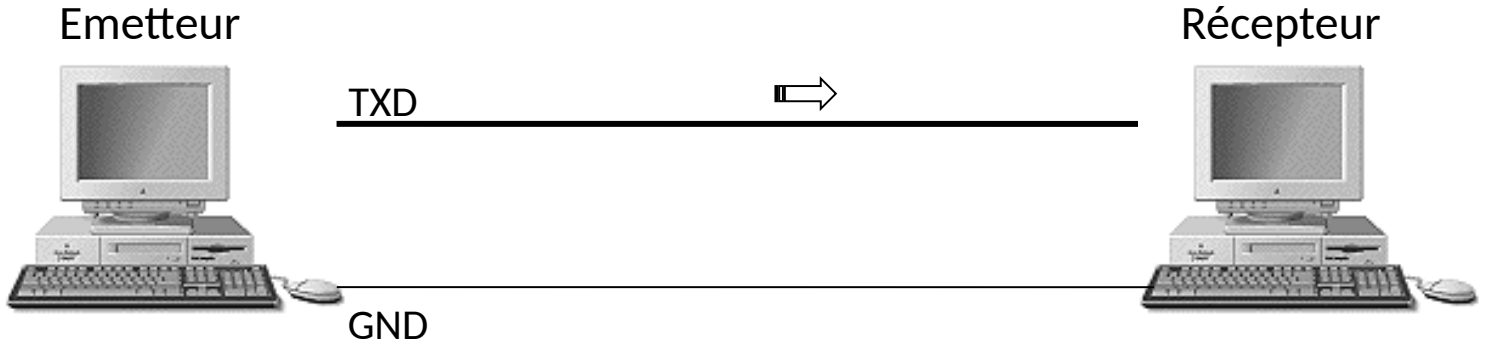
- A la différence des liaisons parallèles la transmission série consiste à transmettre des informations binaires bit par bit sur un fil électrique.



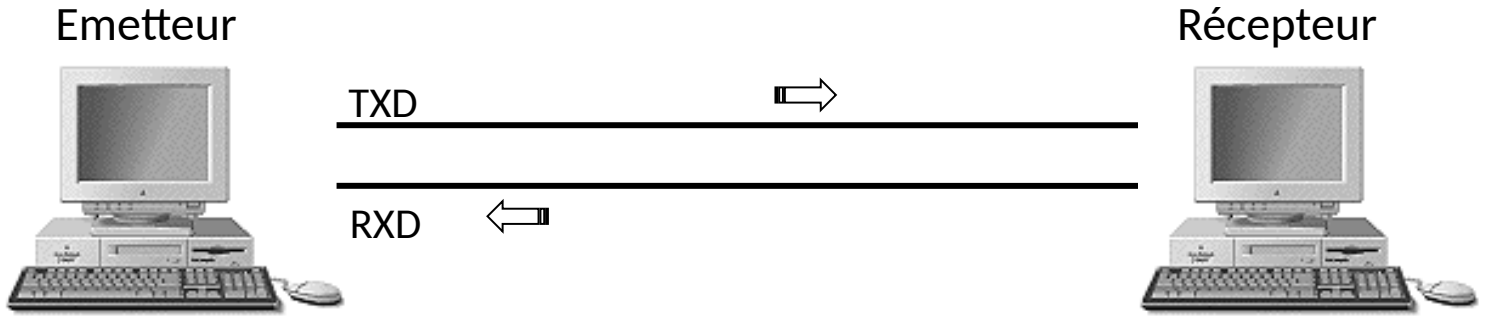
- La communication peut se faire en même temps dans les deux sens : Full-duplex

Principe de fonctionnement

- Communication Half Duplex



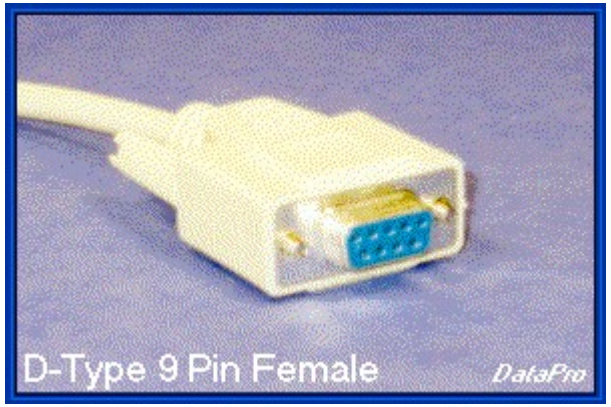
- Communication Full-duplex

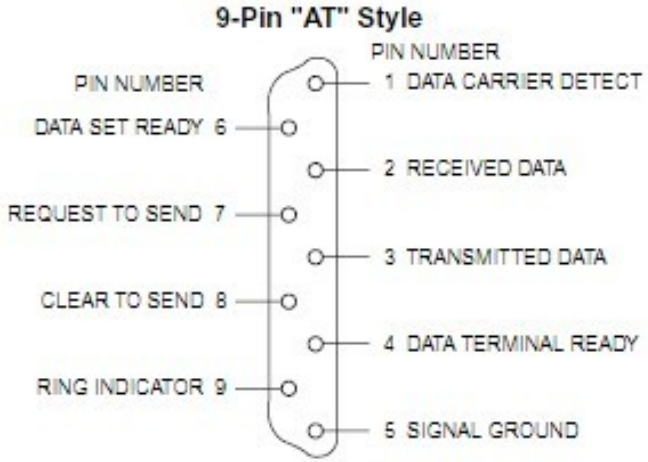


UART – RS232

UART – RS232 : Universal Asynchronous Receiver/ Transmitter

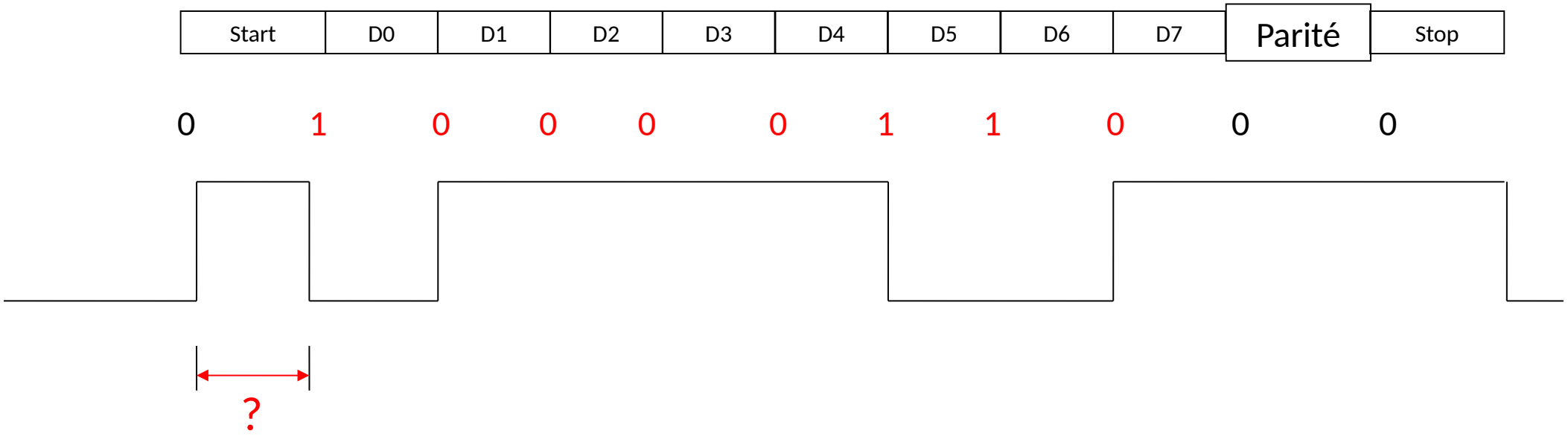
- Débit <1Mbit/s
- Protocole Full-Duplex
- Synchronisation via des signaux de handshake : CTS et RTS
- Liaison point à point
- Trame composée d'un bit de start, de 7 bits de données et 1 bit d stop
- La parité est optionnelle : 1 bit supplémentaire
- 3 fils : TX, RX et GND
- UART : niveaux entre 0 et 5V
- RS232 : niveau 1 entre -25 et -5v et niveau 0 entre 5 et 25v





UART – RS232

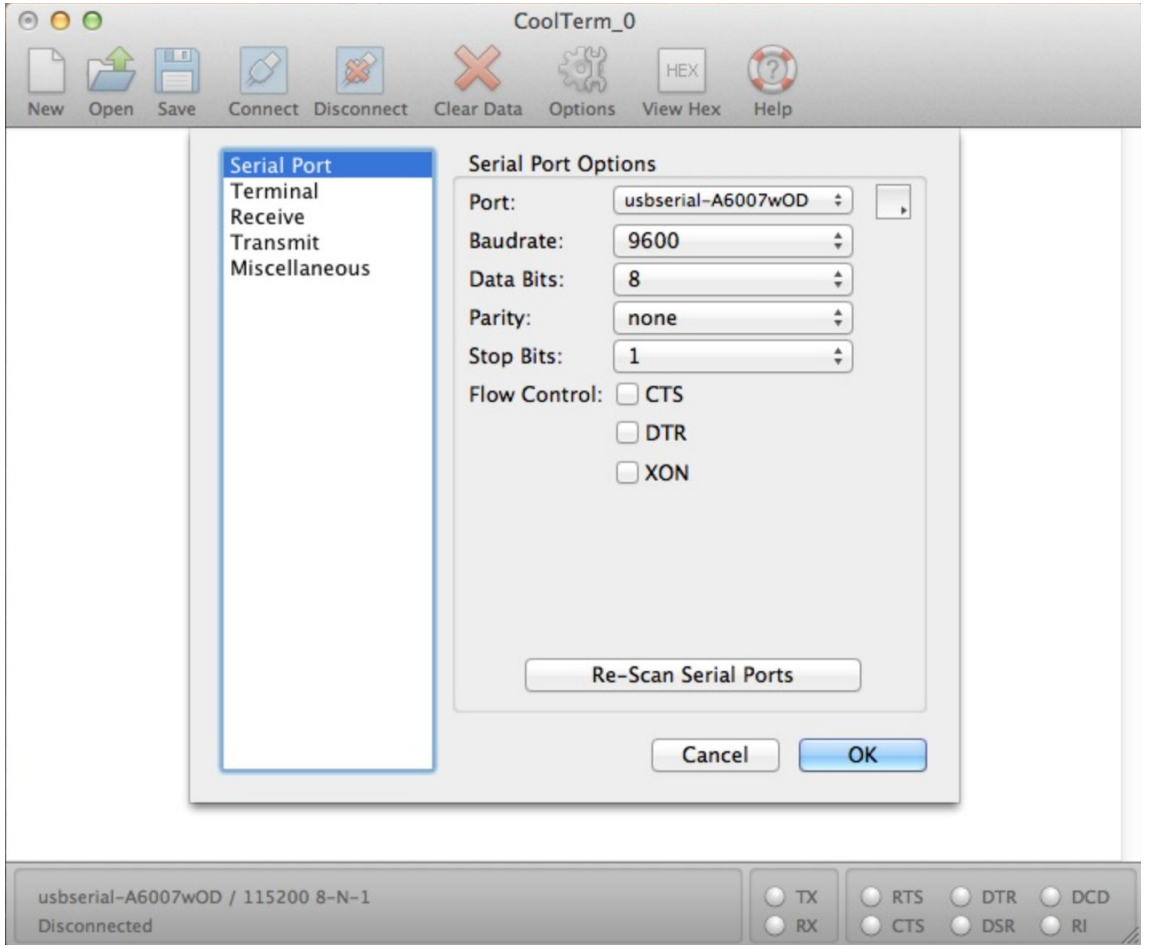
Exemple d'une trame



Période d'un bit : débit binaire en baud

UART – RS232

- Vitesse de transmission
  - 1200, 2400, 4800, 9600, 14400, 19200, 28800, 33600, 56000, 115000 …
  - La vitesse doit être identique pour l'émetteur et le récepteur. Elle est prédéfinie par configuration



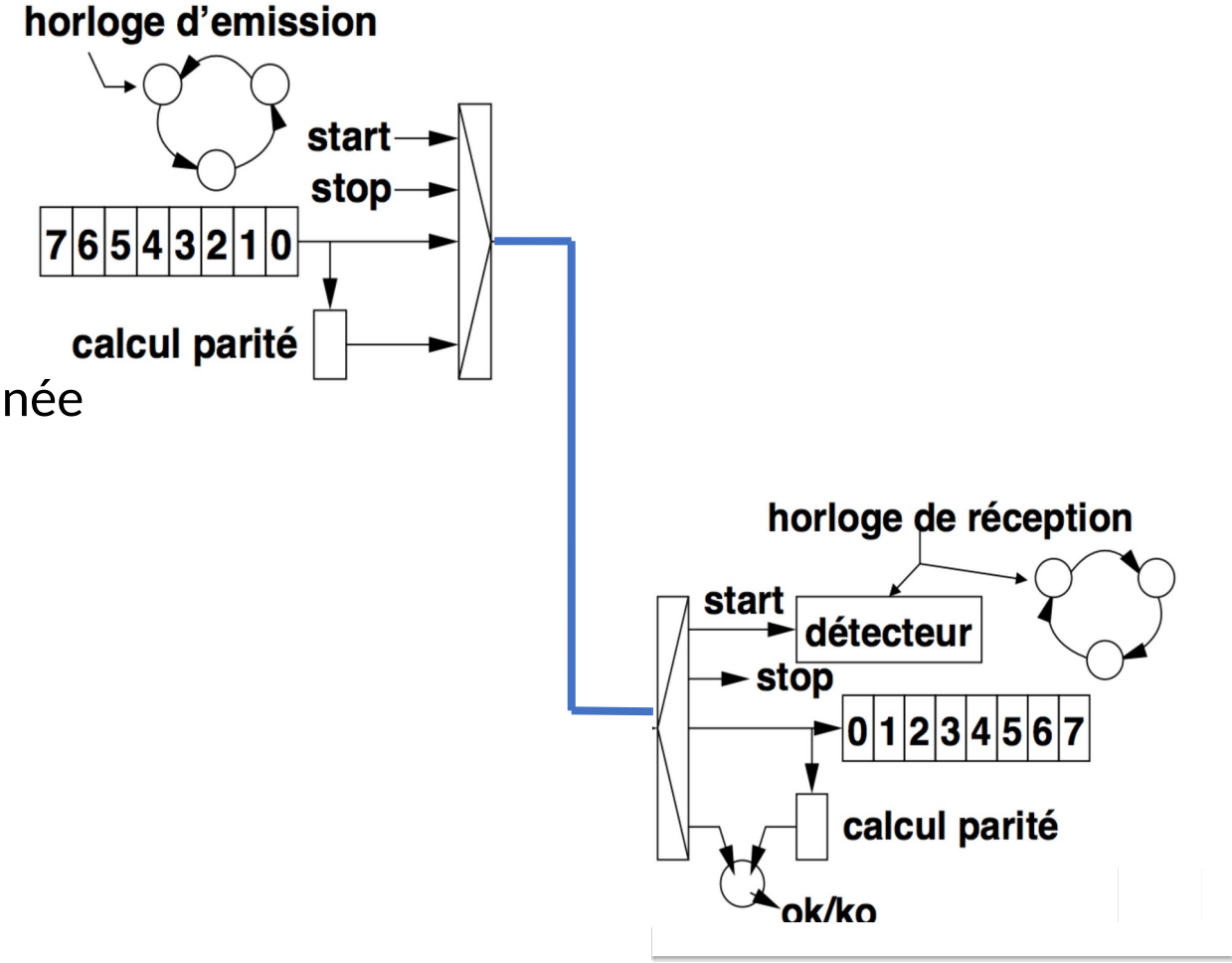
UART – RS232

Emetteur

- La donnée est remplie dans un registre à décalage à chargement parallèle
- Principe de la sérialisation
- Un bit supplémentaire est rajoutée pour signer la donnée (bit à 1 pour les données paires)

Récepteur

- La donnée lue sur la ligne après la détection du bit de start est remplie dans un registre à décalage, à chargement série
- Les données sont disponibles ensuite en parallèle
- Principe de la sérialisation



UART – RS232

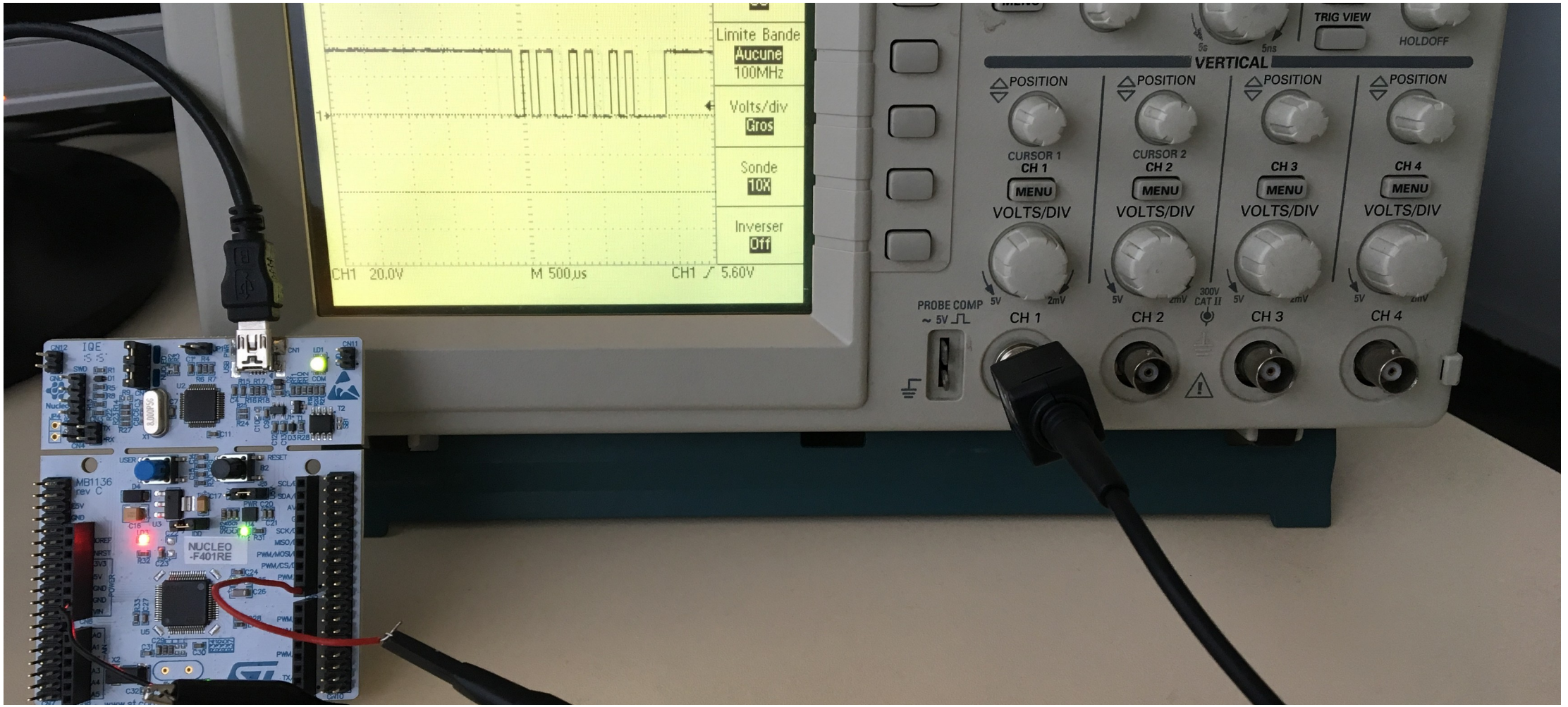
Exemple sous MBED

```
2  #include "mbed.h"
3
4  Serial uart(D8,D2);
5  DigitalOut myled(LED1);
6
7  int main()
8  {
9      int i=48;
10     uart.baud(9600);
11     while(1)
12         {
13         uart.printf("%c\n",i);
14         if (i>100) i=48;
15         myled = 1;
16         wait(0.5);
17         myled = 0;
18         wait(0.5);
19         i++;
20         }
21 }
22
```

	Serial (PinName tx, PinName rx, const char *name=NULL, int baud=MBED_CONF_PLATFORM_DEFAULT_SERIAL_BAUD_RATE) Create a **Serial** port, connected to the specified transmit and receive pins.
	Serial (PinName tx, PinName rx, int baud) Create a **Serial** port, connected to the specified transmit and receive pins, with the specified baud.
void	**baud** (int baudrate) Set the baud rate of the serial port.
void	**format** (int bits=8, Parity parity=SerialBase::None, int stop_bits=1) Set the transmission format used by the serial port.
int	**readable** () Determine if there is a character available to read.
int	**writeable** () Determine if there is space available to write a character.
void	**attach** (Callback< void()> func, IrqType type=RxIrq) Attach a function to call whenever a serial interrupt is generated.
template<typename T >	**MBED_DEPRECATED_SINCE** ("mbed-os-5.1","The attach function does not support cv-qualifiers. Replaced by ""attach(callback(obj, method), type).") void attach(T *obj Attach a member function to call whenever a serial interrupt is generated.
void	**set_flow_control** (Flow type, PinName flow1=NC, PinName flow2=NC) Set the flow control type on the serial port.
int	**write** (const uint8_t *buffer, int length, const event_callback_t &callback, int event=SERIAL_EVENT_TX_COMPLETE) Begin asynchronous write using 8bit buffer.
int	**write** (const uint16_t *buffer, int length, const event_callback_t &callback, int event=SERIAL_EVENT_TX_COMPLETE) Begin asynchronous write using 16bit buffer.
void	**abort_write** () Abort the on-going write transfer.
int	**read** (uint8_t *buffer, int length, const event_callback_t &callback, int event=SERIAL_EVENT_RX_COMPLETE, unsigned char char_match=SERIAL_RESERVED_CHAR_MATCH) Begin asynchronous reading using 8bit buffer.
int	**read** (uint16_t *buffer, int length, const event_callback_t &callback, int event=SERIAL_EVENT_RX_COMPLETE, unsigned char char_match=SERIAL_RESERVED_CHAR_MATCH) Begin asynchronous reading using 16bit buffer.

IUT CERGY-PONTOISE

UART – RS232



UART – STM32 — Vitesse de transmission

- Débit en bits (*bitrate*)
  - Nombres de bits transférés par unité de temps.
  - Exemple : 1000bps (ou 1kbps) = 1000 bits envoyés par seconde.

- Débit en bauds (*baudrate*)
  - Nombre de changement de signaux ou symboles par unité de temps.
  - Un symbole peut être un niveau de tension (par ex., on a 4 niveaux : 0V, 1V, 2V, 4V ; *etc.*).
  - Un symbole peut aussi être une fréquence, ou un changement de phase.

Si on a un signal de type carré avec seulement deux niveaux de tension, alors débit en bauds = débit en bits.

Pour aller plus loin :
https://www.electronicdesign.com/technologies/communications/article/21802272/whats-the-difference-between-bit-rate-and-baud-rate