

# Unité de transmission/réception série asynchrone – version sans bit de parité

ASR 2024 - V1

# Plan de la présentation

## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

# Plan de la présentation

## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

# UART : *Universal Asynchronous Receiver Transmitter*

- contrôleur d'entrée/sortie pour le mode RS232
- RS232 : liaison série point à point utilisée pour connecter des modems, des souris, ...
- remplacé par le bus USB
- Plusieurs modèles :
  - 8250 : registres de 8 bits
  - 16450 : registres de 16 bits
  - 16550 : registres de 16 bits + utilisation de FIFOs

# Plan de la présentation

## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

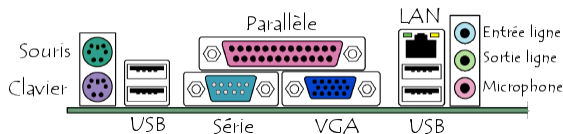
- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

# La liaison RS232, ça me dit quelque chose ...

- Jamais entendu parlé ?



- Où ça se branche ?

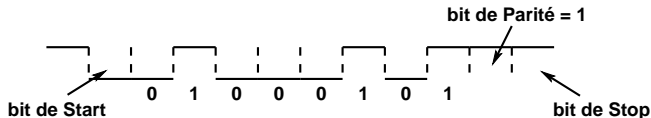


- Nexys 4 ? => Port USB

# La liaison RS232, qu'est-ce que c'est ?

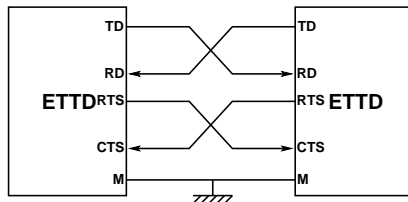
- Transmission série bidirectionnelle (half-duplex ou full-duplex)
- Transmission asynchrone → délimiteurs
  - 1 bit de start
  - 1, 1.5, ou 2 bits de stop
- Détection d'erreur : bit de parité

## Exemple de transmission RS232



# La liaison RS232, comment ça marche ?

- Transmission orientée bit
- Resynchronisation du récepteur à la réception du bit de Start
- Contrôle de flux :
  - logiciel : caractère XON (0x11) = prêt à recevoir, caractère XOFF (0x13) = demande de suspension de la réception
  - matériel : 2 fils supplémentaires : l'émetteur désire émettre (RTS - Ready to Send), le récepteur est prêt à recevoir (CTS - Clear to Send)

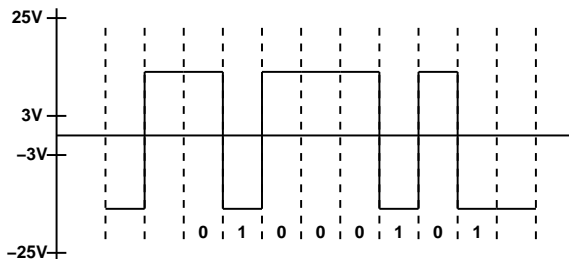




# La liaison RS232, les signaux transmis

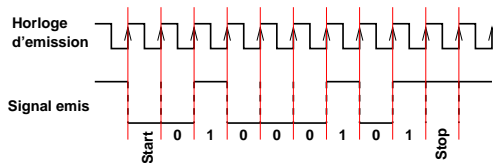
## Niveaux électriques

- 1 logique → tension comprise entre -3V et -25V
- 0 logique → tension comprise entre +3V et +25V
- ex : transmission de 45 (en hexadécimal) avec 1 bit de stop et sans parité

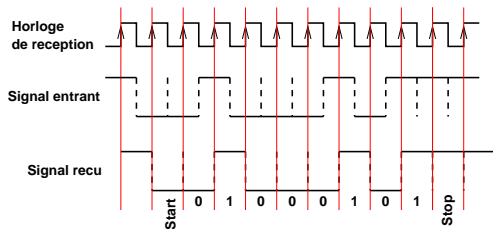


# La liaison RS232, gestion de l'asynchronisme

- chez l'émetteur, fréquence d'émission 9,6 kHz :

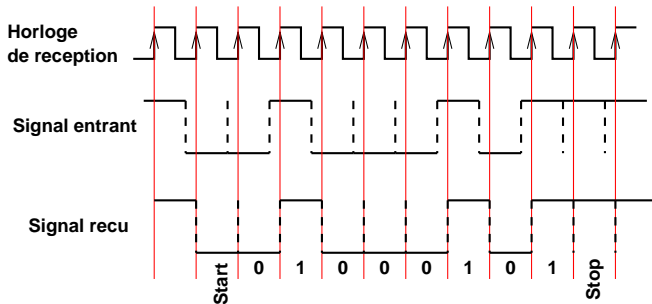


- chez le récepteur :



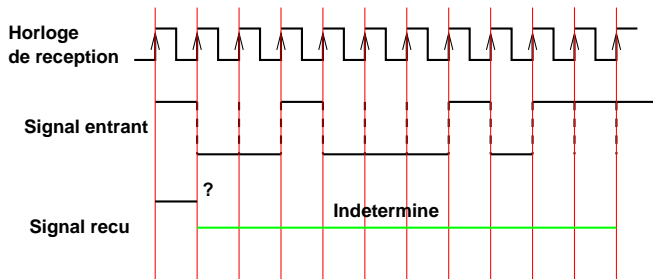
# La liaison RS232, gestion de l'asynchronisme (suite)

■ mais dérive des horloges :



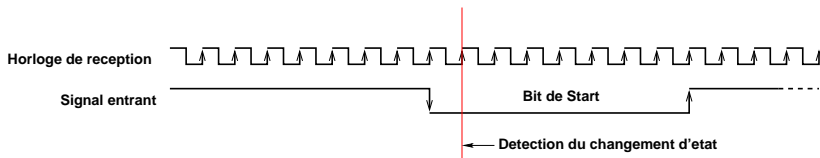
# La liaison RS232, gestion de l'asynchronisme (suite)

## ■ le cas pire :



# La liaison RS232, gestion de l'asynchronisme (fin)

- la solution : détecter au plus tôt le changement de la ligne par surséchantillonnage à la réception



- fréquence de réception = 155 kHz, soit 16x la fréquence d'émission

# Plan de la présentation

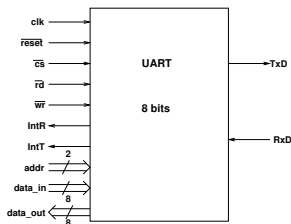
## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

# Schéma global de l'UART



L'interface en VHDL :

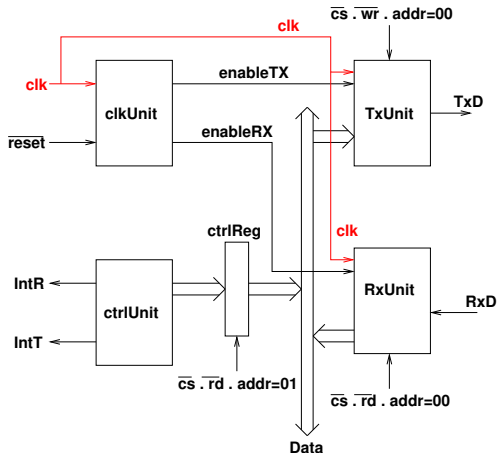
```
entity UART is
  port (
    clk, reset : in  std_logic;
    cs, rd, wr  : in  std_logic;
    RxD        : in  std_logic;
    TxD        : out std_logic;
    IntR       : out std_logic;
    IntT       : out std_logic;
    addr       : in  std_logic_vector(1 downto 0);
    data_in    : in  std_logic_vector(7 downto 0);
    data_out   : out std_logic_vector(7 downto 0));
end UART;
```

# L'UART à réaliser

- Pour simplifier, l'horloge du circuit ( $\equiv$  PC) a une fréquence de 155 kHz (horloge de réception)
  
- Le système se découpe en 4 unités :
  - unité de gestion des horloges d'émission et de réception
  - unité de contrôle qui gère le registre de contrôle (fournie)
  - unité d'émission
  - unité de réception

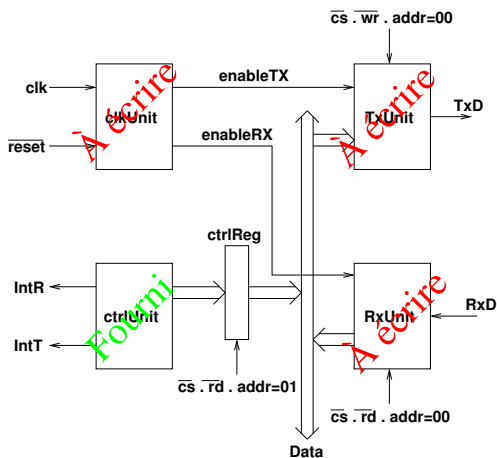


# Schéma interne de l'UART



Attention : toutes les connexions ne sont pas représentées.

# Schéma interne de l'UART



- └ Description de l'UART à réaliser (1ère partie)
- └ Gestion des horloges d'émission et de réception

# Plan de la présentation

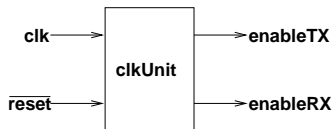
## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

# L'interface de l'unité de gestion des horloges



```
entity clkUnit is
  port (
    clk, reset : in  std_logic;
    enableTX   : out std_logic;
    enableRX   : out std_logic);
end clkUnit;
```

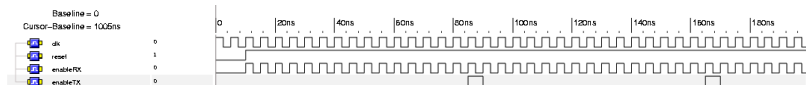
## Implantation de cette unité

### ■ Génère 2 horloges :

- enableRX, horloge de réception, de fréquence 155 kHz
  - pour simplifier, enableRX est synchrone avec l'horloge du circuit :  $\text{enableRX} = \text{clk}$  sauf quand reset
- enableTX, horloge d'émission, de fréquence 9,6 kHz, 16x moins rapide que clk

- └ Description de l'UART à réaliser (1ère partie)
- └ Gestion des horloges d'émission et de réception

# Le comportement de l'unité de gestion des horloges



- tant que reset=0, les horloges enableRX et enableTX n'évoluent pas
- enableRX est (presque) une copie de l'horloge clk
- tous les 16 tops d'horloge clk, enableTX passe à 1 **pour une période** de l'horloge et vaut 0 autrement

└ Description de l'UART à réaliser (1ère partie)

└ L'unité de contrôle de l'UART (fournie)

# Plan de la présentation

## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

## Rôle de l'unité de contrôle (1)

- Fournir les signaux d'interruption IntR et IntT au processeur
  - lorsqu'une donnée a été reçue par l'unité de réception, un front descendant sur IntR indique au processeur qu'il peut demander la lecture de l'octet présent dans le registre de réception
  - lorsque l'unité d'émission est disponible, un front descendant sur IntT indique au processeur qu'il peut émettre un nouvel octet



└ Description de l'UART à réaliser (1ère partie)

└ L'unité de contrôle de l'UART (fournie)

## Rôle de l'unité de contrôle (2)

- Gérer le registre de contrôle ctrlReg en utilisant les signaux fournis par les unités d'émission et de réception
  - le processeur pourra lire ce registre en position addr=01 et en envoyant un signal de lecture (i.e. sélection de l'UART par cs et commande lecture rd)

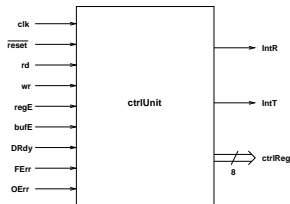
Res	Res	Res	Res	TxRdy	RxRdy	FErr	OErr
-----	-----	-----	-----	-------	-------	------	------

- TxRdy : l'UART est prête à émettre
- RxRdy : l'UART a reçu une donnée valide
- FErr : la trame transmise est erronée
- OErr : le processeur n'a pas lu à temps la donnée reçue

└ Description de l'UART à réaliser (1ère partie)

└ L'unité de contrôle de l'UART (fournie)

## Interface de l'unité de contrôle



```
entity ctrlUnit is
  port (
    clk, reset      : in  std_logic;
    rd, wr          : in  std_logic;
    DRdy, FErr, OErr : in  std_logic;
    BufE, RegE      : in  std_logic;
    IntR            : out std_logic;
    IntT            : out std_logic;
    ctrlReg         : out std_logic_vector(7 downto 0));
end ctrlUnit;
```

# Plan de la présentation

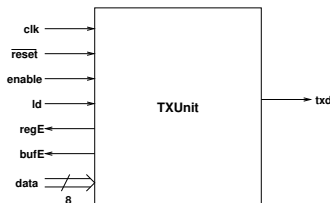
## 1 Introduction

- L'UART
- La liaison RS232

## 2 Description de l'UART à réaliser (1ère partie)

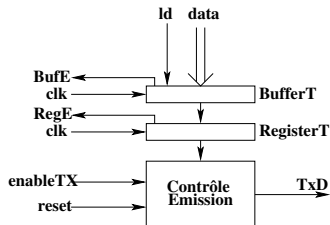
- Description du projet
- Gestion des horloges d'émission et de réception
- L'unité de contrôle de l'UART (fournie)
- L'unité d'émission

# Interface de l'unité d'émission



```
entity TXUnit is
  port (
    clk, reset : in std_logic;
    enable      : in std_logic;
    ld          : in std_logic;
    txd         : out std_logic;
    regE        : out std_logic;
    bufE        : out std_logic;
    data        : in std_logic_vector(7 downto 0)
  );
end TXUnit;
```

# Composants de l'unité d'émission

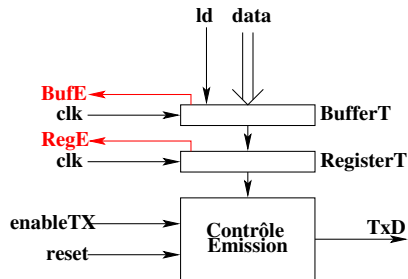


L'émetteur se compose de :

- un **registre BufferT** qui permet de mettre en attente une nouvelle donnée lorsqu'une donnée est en train d'être émise. Le signal **BufE** (*Buffer Enable*) permet de savoir si le buffer est vide ( $\text{BufE}=1$ ) ou pas ( $\text{BufE}=0$ )
- un **registre RegisterT**, registre d'émission utilisé par le circuit contrôle d'émission au cours de la transmission de la donnée. Le signal **RegE** (*Register Enable*) permet de savoir si ce registre est vide ( $\text{RegE}=1$ ) ou pas ( $\text{RegE}=0$ )
- un **circuit de contrôle d'émission** qui positionne les bits à émettre sur la ligne **TxD** à chaque top de **enableTX**

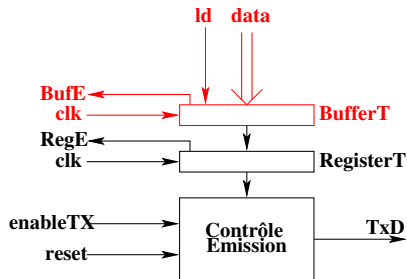
# Fonctionnement de l'unité d'émission

- 1** initialement le buffer et le registre sont libres (bufE=1 et regE=1)



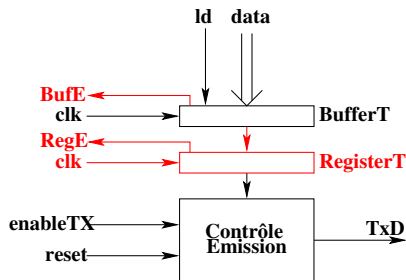
# Fonctionnement de l'unité d'émission

- 1 initialement le buffer et le registre sont libres (bufE=1 et regE=1)
- 2 chargement d'une donnée dans le buffer (bufE=0 et regE=1)



# Fonctionnement de l'unité d'émission

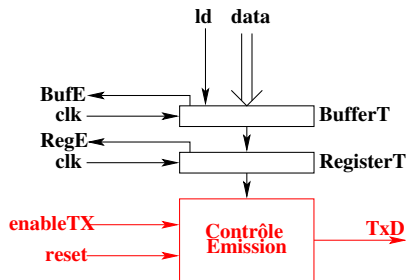
- 1 initialement le buffer et le registre sont libres (bufE=1 et regE=1)
- 2 chargement d'une donnée dans le buffer (bufE=0 et regE=1)
- 3 déchargement du buffer dans le registre d'émission (bufE=1 et regE=0), le buffer est libre → **le processeur peut demander à émettre un nouveau caractère**





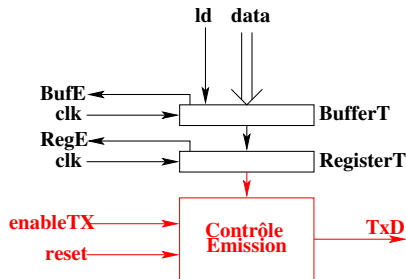
# Fonctionnement de l'unité d'émission

- 1 initialement le buffer et le registre sont libres (bufE=1 et regE=1)
- 2 chargement d'une donnée dans le buffer (bufE=0 et regE=1)
- 3 déchargement du buffer dans le registre d'émission (bufE=1 et regE=0), le buffer est libre → **le processeur peut demander à émettre un nouveau caractère**
- 4 envoi du bit de start quand enableTx vaut 1



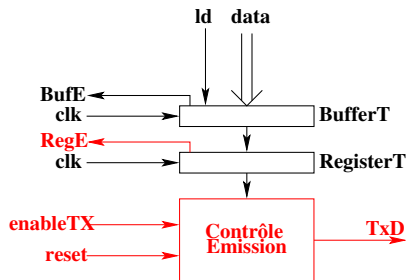
# Fonctionnement de l'unité d'émission

- 1 initialement le buffer et le registre sont libres (bufE=1 et regE=1)
- 2 chargement d'une donnée dans le buffer (bufE=0 et regE=1)
- 3 déchargement du buffer dans le registre d'émission (bufE=1 et regE=0), le buffer est libre → **le processeur peut demander à émettre un nouveau caractère**
- 4 envoi du bit de start quand enableTx vaut 1
- 5 envois successifs des données quand enableTx vaut 1



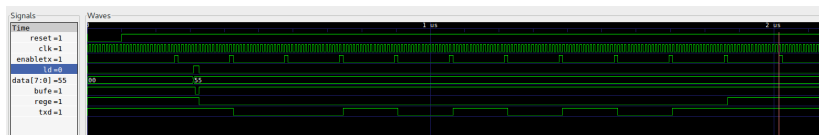
# Fonctionnement de l'unité d'émission

- 1 initialement le buffer et le registre sont libres (bufE=1 et regE=1)
- 2 chargement d'une donnée dans le buffer (bufE=0 et regE=1)
- 3 déchargement du buffer dans le registre d'émission (bufE=1 et regE=0), le buffer est libre → **le processeur peut demander à émettre un nouveau caractère**
- 4 envoi du bit de start quand enableTx vaut 1
- 5 envois successifs des données quand enableTx vaut 1
- 6 envoi du bit de stop quand enableTx vaut 1, le registre est libre à nouveau (bufE=1 et regE=1)



## Test de l'unité d'émission

### ■ Emission d'un unique caractère 0x55 ('U') :

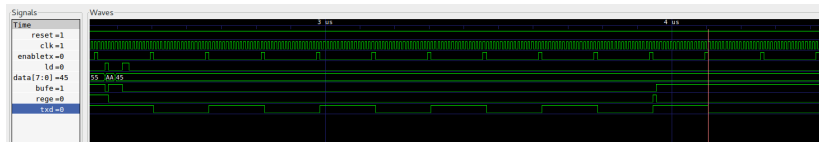


le curseur rouge délimite la fin de la transmission du caractère (fin de l'envoi du bit de stop).

## Test de l'unité d'émission

Emission de 2 caractères à la suite :

- Emission du caractère 0xAA et acquisition du caractère 0x45 ('E') :



le curseur rouge délimite le bit de stop de la transmission de 0xAA et le bit de start de celle de 0x45