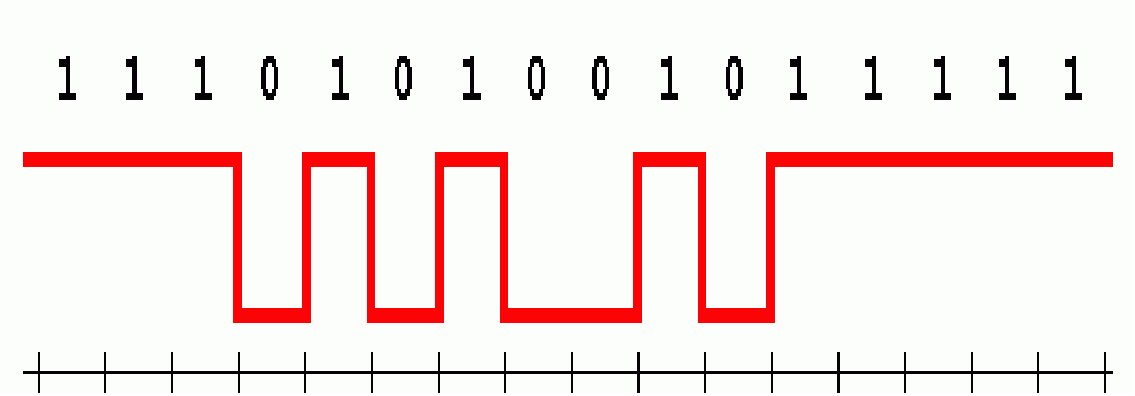
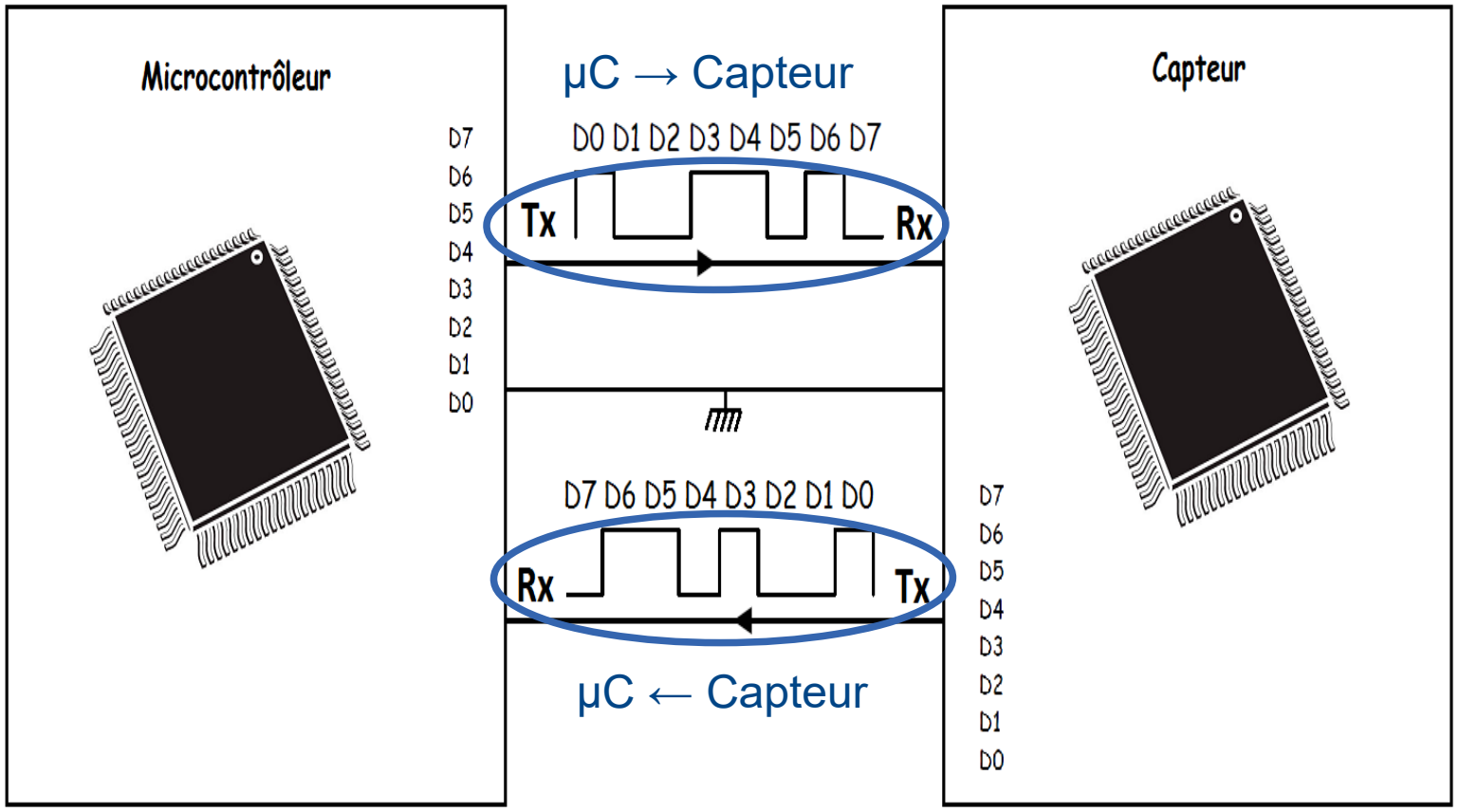
# Microcontrôleurs




Liaison Série et I2C

# Liaison Série et I2C - Sommaire

- **Liaison point à point**



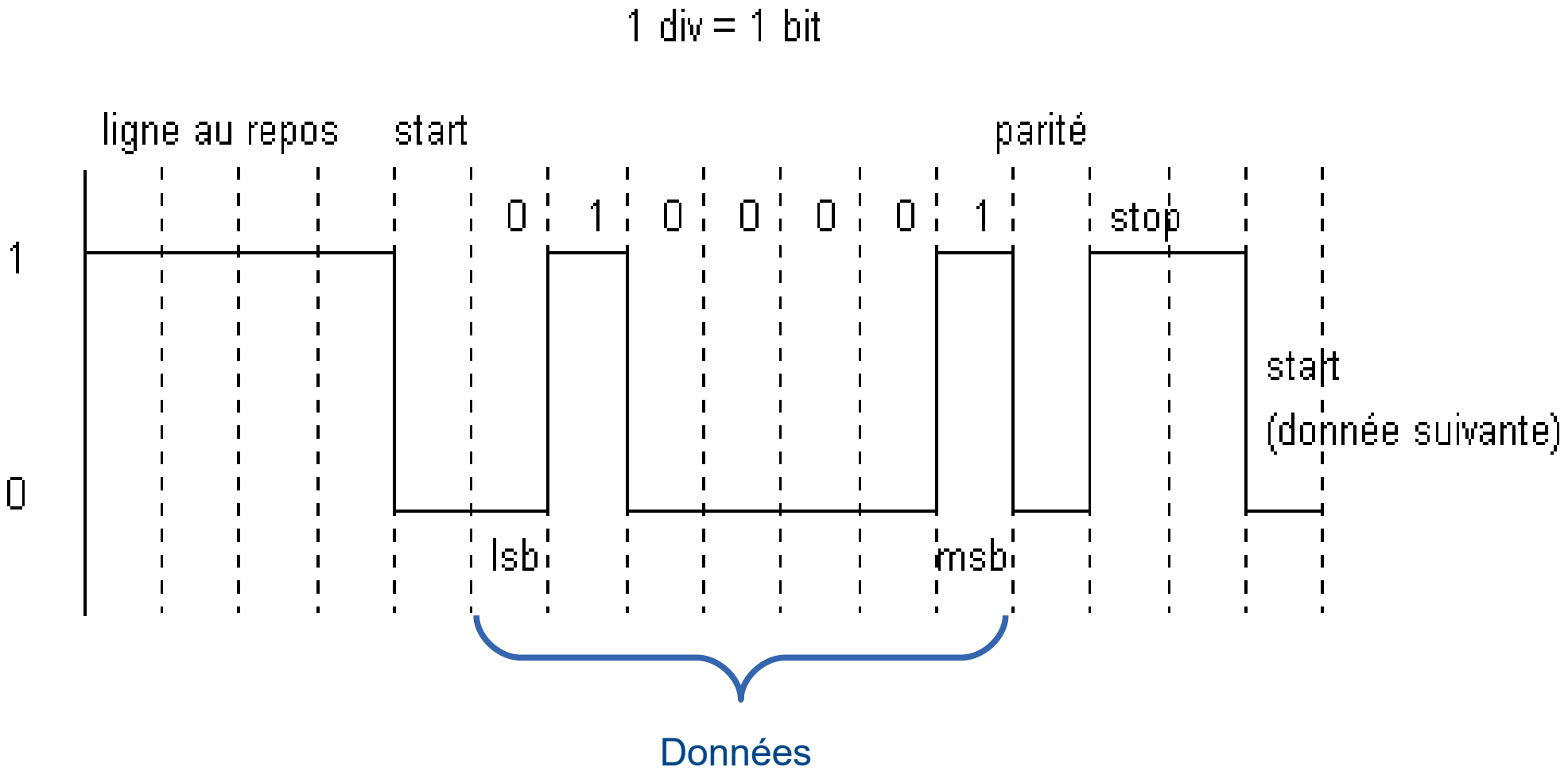
La transmission se fait en émettant les bits de données les uns après les autres.

La liaison série est soit :

- Asynchrone. Un signal de synchronisation est généré par l'émetteur au début d'une séquence de bits données plus ou moins longue.

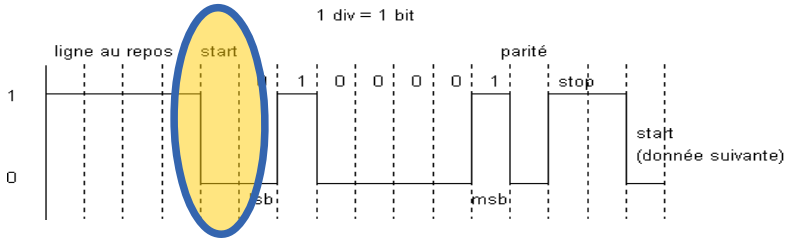
- Synchrone. Fil d'horloge supplémentaire permettant de synchroniser le récepteur.

- **Liaison Série Asynchrone**




→ La ligne de transmission est à « 1 » au repos

- **Bit de Start**

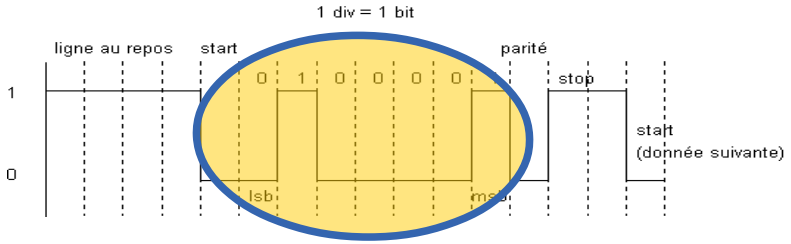



**→ La transmission commence par un bit de Start. Il vaut « 0 » et sa durée est fixée par le baudrate (vitesse de transmission) sélectionné.**

## Baudrate standard :

| | |
|---|---|
| 110 bauds | 14400 bauds |
| 300 bauds | 19200 bauds |
| 600 bauds | 28800 bauds |
| 1200 bauds | 38400 bauds |
| 2400 bauds | 56000 bauds |
| 4800 bauds | 57600 bauds |
| 9600 bauds | 115200 bauds |

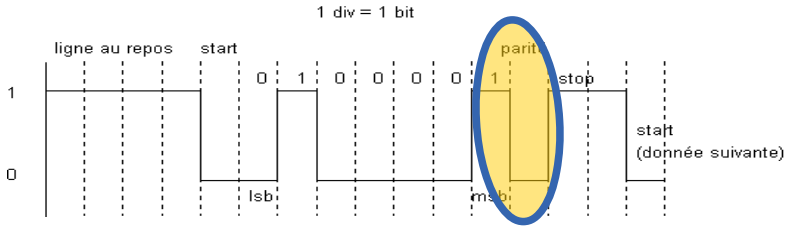
- **Data**




**→ Le bit de poids faible est envoyé en premier. Le temps d'émission de chaque bit est fixé par le baudrate.**

**→ Le nombre de bit émis par trame est le plus souvent de 8 bits (1 octet), mais il existe d'autres variantes.**

- **Parité**




**Le but est de détecter des erreurs de transmission.**

**→ Parité « Paire » :**

**on compte les bits à "1" en comprenant le bit de parité. Leur nombre doit être pair.**

**→ Parité « Impaire » :**

**on compte les bits à "1" en comprenant le bit de parité. Leur nombre doit être Impair.**

**En résumé, il existe trois possibilités de convention.**

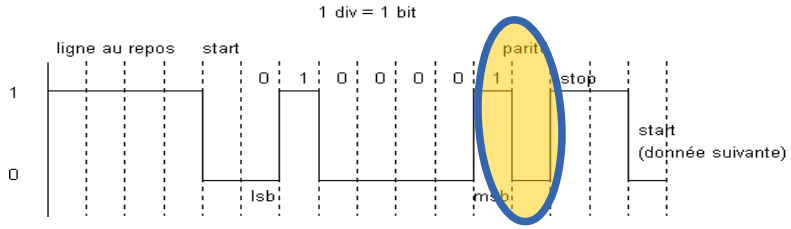
**- PARITE PAIRE : "EVEN" en anglais.**

**- PARITE IMPAIRE : "ODD" en anglais.**

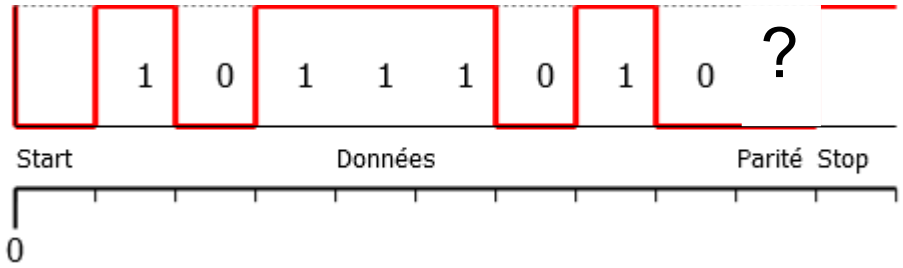
**- PAS DE BIT DE PARITE : "NONE" en anglais**

**!!! Le transmetteur et le récepteur doivent être accordés sur la même convention !!!**

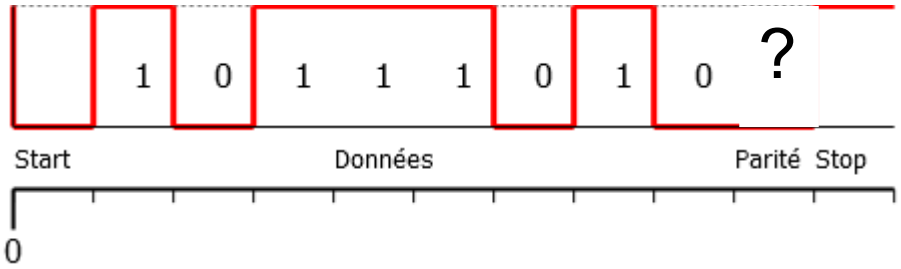
- **Parité**



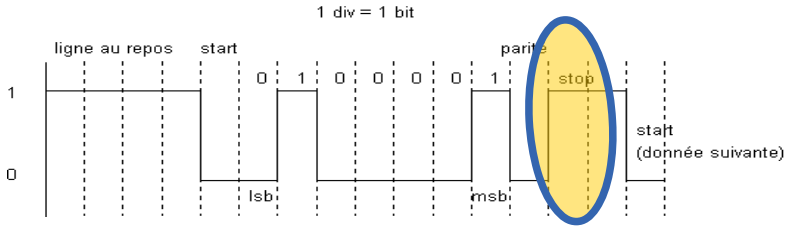

**Parité Paire. Calculer le bit de parité de la transmission suivante :**




- **Parité Impaire. Calculer le bit de parité de la transmission suivante :**



- **Stop**

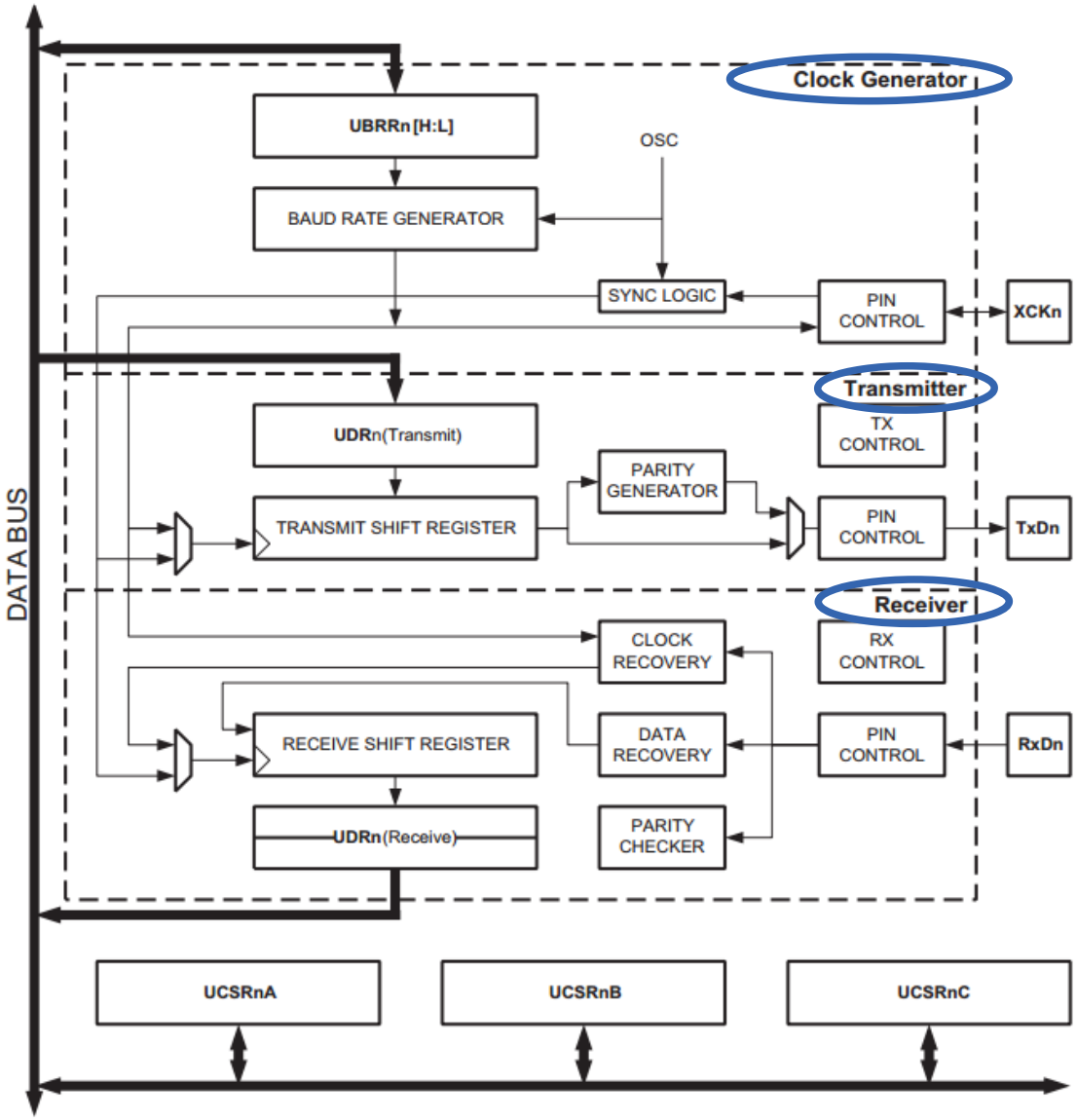



**→ Il sert à mettre fin à la trame et à laisser le temps au récepteur d'enregistrer la donnée reçue.**

**→ La durée du bit de Stop varie entre 0,5 et 2 fois le temps bit.**

**( on parle de 0,5 Stop, 1 Stop ou 2 Stop)**

- **Hardware** ATmega48A/PA/88A/PA/168A/PA/328/P



- **Hardware** ATmega48A/PA/88A/PA/168A/PA/328/P

**Trouver la valeur du registre UBRR0 pour un baudrate de 9600 bauds.**

**Table 20-7. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies (Continued)**

| Baud Rate (bps) | $f_{osc}$ = 16.0000MHz | | | | $f_{osc}$ = 18.4320MHz | | | | $f_{osc}$ = 20.0000MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | U2Xn = 0 | | U2Xn = 1 | | U2Xn = 0 | | U2Xn = 1 | | U2Xn = 0 | | U2Xn = 1 | |
| | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error | UBRRn | Error |
| 2400 | 416 | -0.1% | 832 | 0.0% | 479 | 0.0% | 959 | 0.0% | 520 | 0.0% | 1041 | 0.0% |
| 4800 | 207 | 0.2% | 416 | -0.1% | 239 | 0.0% | 479 | 0.0% | 259 | 0.2% | 520 | 0.0% |
| 9600 | 103 | 0.2% | 207 | 0.2% | 119 | 0.0% | 239 | 0.0% | 129 | 0.2% | 259 | 0.2% |
| 14.4k | 68 | 0.6% | 138 | -0.1% | 79 | 0.0% | 159 | 0.0% | 86 | -0.2% | 173 | -0.2% |
| 19.2k | 51 | 0.2% | 103 | 0.2% | 59 | 0.0% | 119 | 0.0% | 64 | 0.2% | 129 | 0.2% |
| 28.8k | 34 | -0.8% | 68 | 0.6% | 39 | 0.0% | 79 | 0.0% | 42 | 0.9% | 86 | -0.2% |
| 38.4k | 25 | 0.2% | 51 | 0.2% | 29 | 0.0% | 59 | 0.0% | 32 | -1.4% | 64 | 0.2% |
| 57.6k | 16 | 2.1% | 34 | -0.8% | 19 | 0.0% | 39 | 0.0% | 21 | -1.4% | 42 | 0.9% |
| 76.8k | 12 | 0.2% | 25 | 0.2% | 14 | 0.0% | 29 | 0.0% | 15 | 1.7% | 32 | -1.4% |
| 115.2k | 8 | -3.5% | 16 | 2.1% | 9 | 0.0% | 19 | 0.0% | 10 | -1.4% | 21 | -1.4% |
| 230.4k | 3 | 8.5% | 8 | -3.5% | 4 | 0.0% | 9 | 0.0% | 4 | 8.5% | 10 | -1.4% |
| 250k | 3 | 0.0% | 7 | 0.0% | 4 | -7.8% | 8 | 2.4% | 4 | 0.0% | 9 | 0.0% |
| 0.5M | 1 | 0.0% | 3 | 0.0% | – | – | 4 | -7.8% | – | – | 4 | 0.0% |
| 1M | 0 | 0.0% | 1 | 0.0% | – | – | – | – | – | – | – | – |
| Max. (1) | 1Mbps | | 2Mbps | | 1.152Mbps | | 2.304Mbps | | 1.25Mbps | | 2.5Mbps | |

1. UBRRn = 0, Error = 0.0%

- **Software** (https://www.arduino.cc/en/Reference/Serial)

Serial.available() Retourne le nombre de char disponible à la lecture

Serial.begin(speed, config) Ouvrir la liaison série

Serial.end() Fermer la liaison série

Serial.print(val, format) Écrire une valeur

Serial.println(val, format) Écrire et saut de ligne

Serial.read() Lire

Serial.write(val) Serial.write(str) Serial.write(buf, len) Écrire un octet

Functions

- if (Serial)
- available()
- availableForWrite()
- begin()
- end()
- find()
- findUntil()
- flush()
- parseFloat()
- parseInt()
- peek()
- print()
- println()
- read()
- readBytes()
- readBytesUntil()
- readString()
- readStringUntil()
- setTimeout()
- write()
- serialEvent()

- **Software**
  - **Mise en œuvre (les étapes)**
    - **Ouvrir la liaison série en configurant le bon mode (vitesse, nombre de bit de data, parité, nombre de bit de stop).**
    - **Utiliser les fonctions de lecture et d'écriture.**
  - **Exemple 1 :**

```
void setup() {
  Serial.begin(9600,SERIAL_7N1);     // initialize serial:
  Serial.print("Exemple");
  Serial.print(1);
  Serial.println("Liaison Serie sans IT");
}

void loop() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    Serial.write(inChar);
  }

}
```

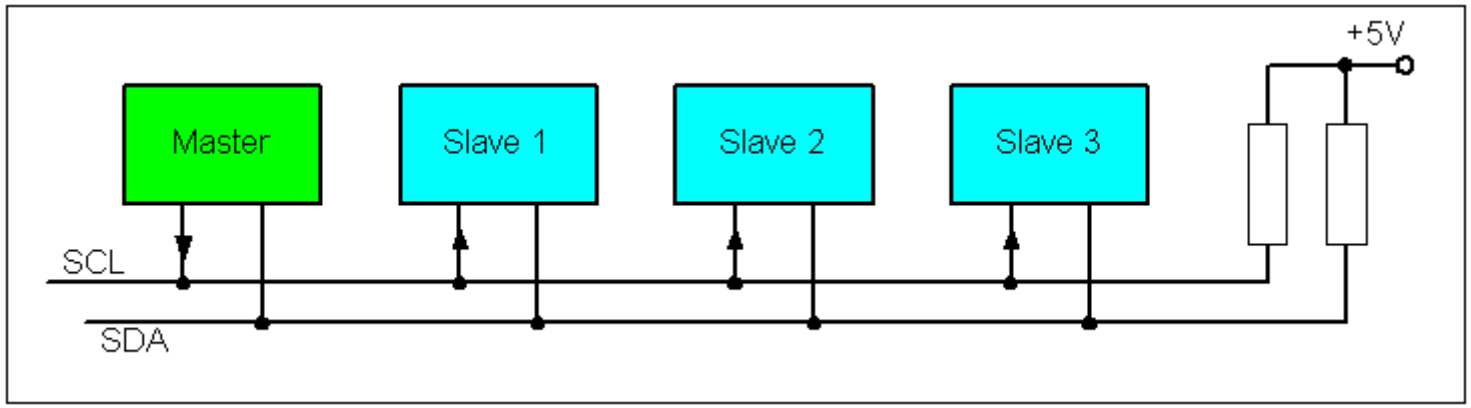
- **Mise en œuvre** (https://www.arduino.cc/en/Reference/softwareSerial)
  - **Déclaration d'un objet de type « *SoftwareSerial* » avec la désignation de Rx, Tx**
  - **Utilisation classique**
  - **!!!! Attention aux limitations**

→ Baudrate max : 57600 bauds
→ Incompatible avec certaines pins
→ Ressource processeur importante
→ uniquement configurable en mode 8N1
(8 bits, No parity, 1 Stop)

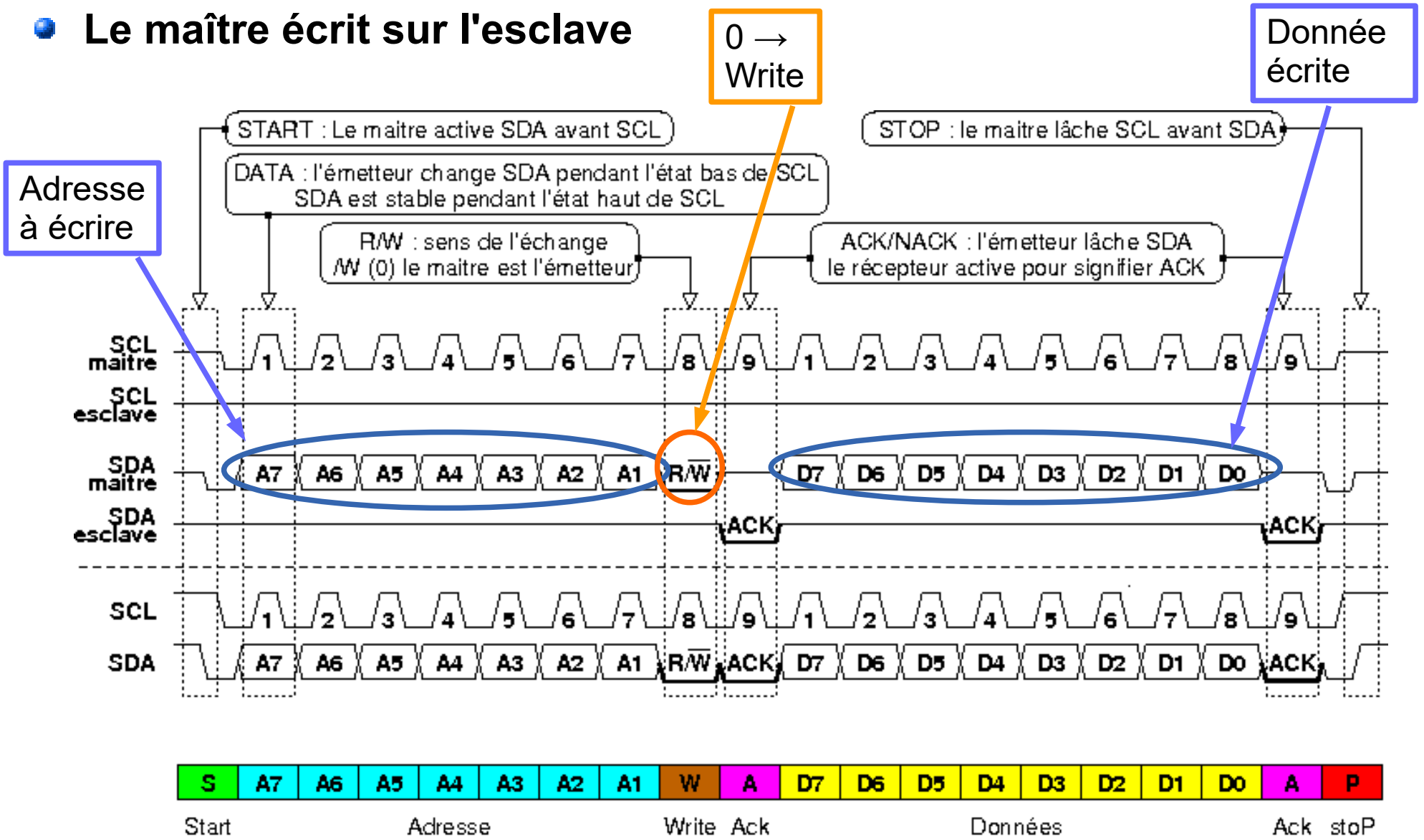
```
#define TX 10
#define RX 11

#include <SoftwareSerial.h>
SoftwareSerial mySerial(RX, TX); // RX, TX

void setup() {
    pinMode(TX,OUTPUT);
    pinMode(RX,INPUT);
    mySerial.begin(9600);
    mySerial.print("SoftwareSerial.h");
    mySerial.print('V');
    mySerial.println(1);
}
```

# Liaison multipoint

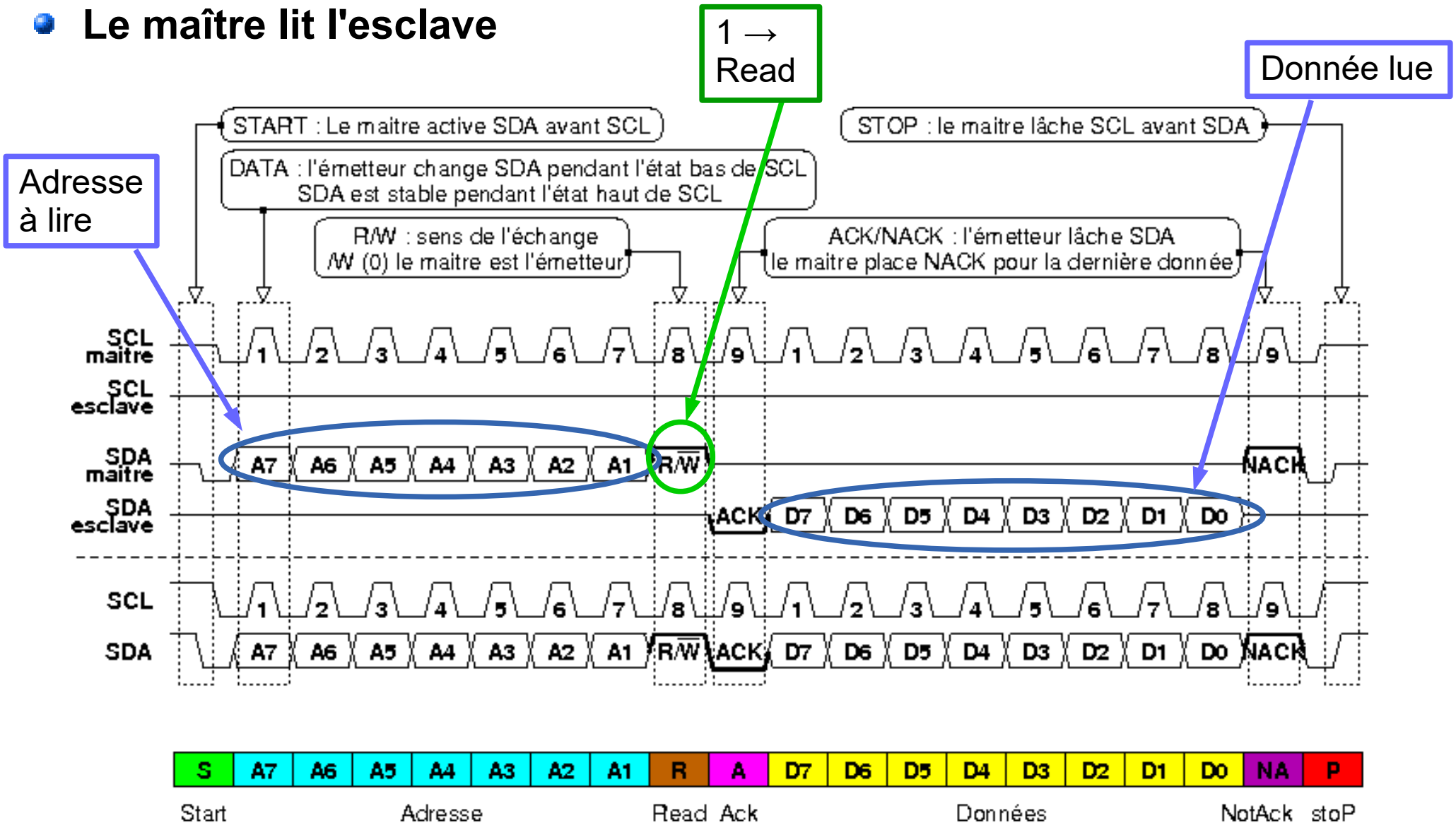
- Un maître ↔ plusieurs esclaves.
- Une ligne d'horloge SCL et une ligne de Données SDA. Niveau « 1 » au repos.
- Le maître gère les échanges de données.
- Résistance de pull-up à rajouter sur le bus (valeur à sélectionner soigneusement).
- Vitesse de transfert entre 100 kb/s et 5Mb/s.
- Chaque esclave possède une adresse unique.



- **Le maître écrit sur l'esclave**



- **Le maître lit l'esclave**



- **Hardware** ATmega48A/PA/88A/PA/168A/PA/328/P
  - **Configuration de la vitesse de transfert**

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot (\text{PrescalerValue})}$$

- TWBR = Value of the TWI Bit Rate Register.
- *PrescalerValue* = Value of the prescaler, see Table 22-7 on page 232.

Note: Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load. See Table 29-14 on page 308 for value of pull-up resistor.

### 22.9.3 TWSR – TWI Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| (0xB9) | TWS7 | TWS6 | TWS5 | TWS4 | TWS3 | – | TWPS1 | TWPS0 | TWSR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |

**Table 22-7. TWI Bit Rate Prescaler**

| TWPS1 | TWPS0 | Prescaler Value |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 4 |
| 1 | 0 | 16 |
| 1 | 1 | 64 |

- **Software** (https://www.arduino.cc/en/reference/wire)

Mode esclave

Wire.begin(address)

Mode maître

Wire.begin()

Functions

Ouvre le port I2C - begin()

Demande de données à l'esclave - requestFrom()

Démarre une communication I2C - beginTransmission()

Termine une communication I2C - endTransmission()

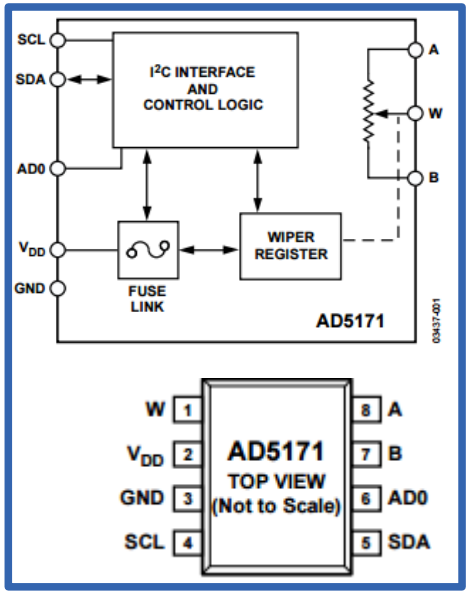
Écrire - write()

Donnée(s) disponible(s) ? - available()

Lecture d'une donnée (octet) - read()

- onReceive()
- onRequest()

- **Software**
  - **Exemple : AD5171 potentiomètre digital**




```
#include <Wire.h>
void setup(){
  Wire.begin(); // join i2c bus (address optional for master)
}

byte val = 0;
void loop(){
  Wire.beginTransmission(44); // transmit to device #44 (0x2c)
  // device address is specified in datasheet
  Wire.write(byte(0x00));            // sends instruction byte
  Wire.write(val);              // sends potentiometer value byte
  Wire.endTransmission();     // stop transmitting
  val++;        // increment value
  if (val == 64){ // if reached 64th position (max)
     val = 0;    // start over from lowest value
  }
  delay(500);
}
```