

# Electronics Serial Communication-UART

Terry Sturtevant

Wilfrid Laurier University

November 16, 2016

# Serial Communication -UART

# Serial Communication -UART

- Universal Asynchronous Receiver Transmitter

# Serial Communication -UART

- Universal Asynchronous Receiver Transmitter
- Simplest form of serial communication

# Serial Communication -UART

- Universal Asynchronous Receiver Transmitter
- Simplest form of serial communication
- Between 2 devices

# Serial Communication -UART

- Universal Asynchronous Receiver Transmitter
- Simplest form of serial communication
- Between 2 devices
- Uses 2 signals (and Ground), Rx and Tx

## Serial Communication -UART

- Universal Asynchronous Receiver Transmitter
- Simplest form of serial communication
- Between 2 devices
- Uses 2 signals (and Ground), Rx and Tx
- Asynchronous, so both must agree on baud rate

# Communication parameters



# Communication parameters

- 1 Start bit at “0” level

# Communication parameters

- 1 Start bit at “0” level
- LSB transmitted first

## Communication parameters

- 1 Start bit at “0” level
- LSB transmitted first
- Can have odd, even, or no parity bit

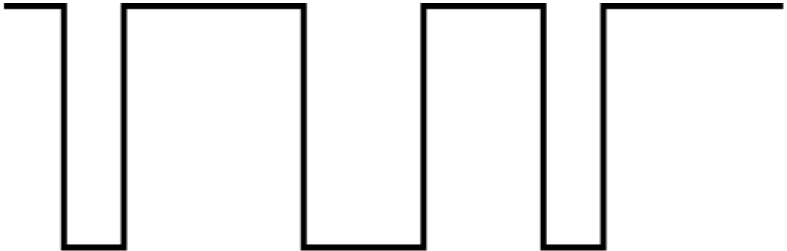
## Communication parameters

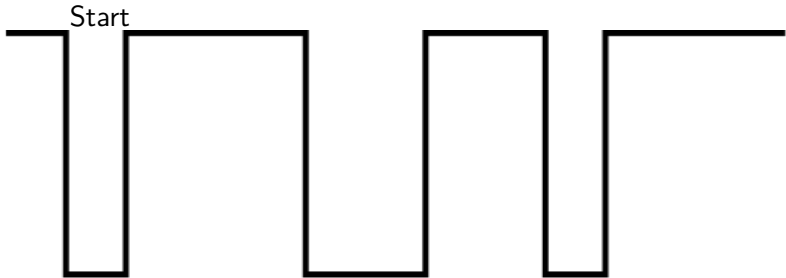
- 1 Start bit at “0” level
- LSB transmitted first
- Can have odd, even, or no parity bit
- 1 or 2 Stop bits at “1” level

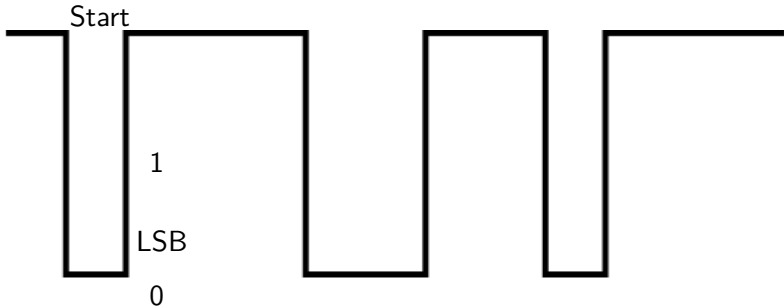
## Communication parameters

- 1 Start bit at “0” level
- LSB transmitted first
- Can have odd, even, or no parity bit
- 1 or 2 Stop bits at “1” level

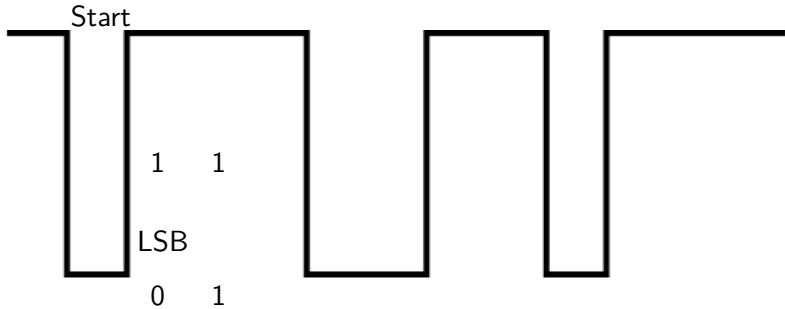
Since start and stop bits are opposite, new characters can always be detected.

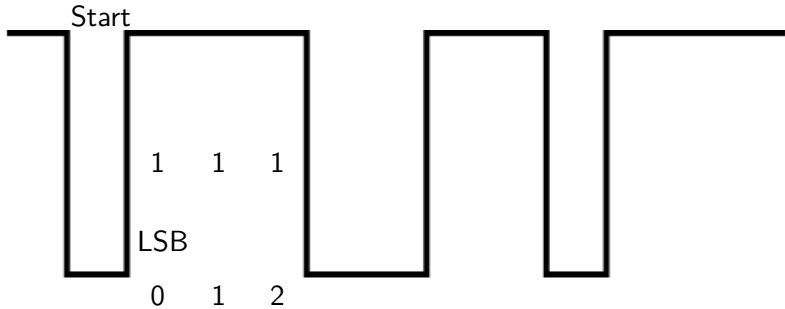


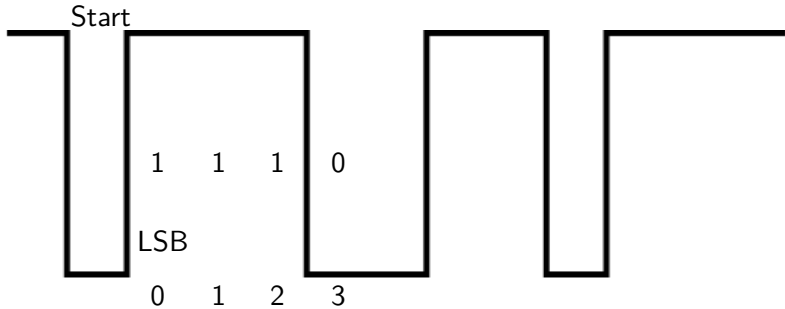


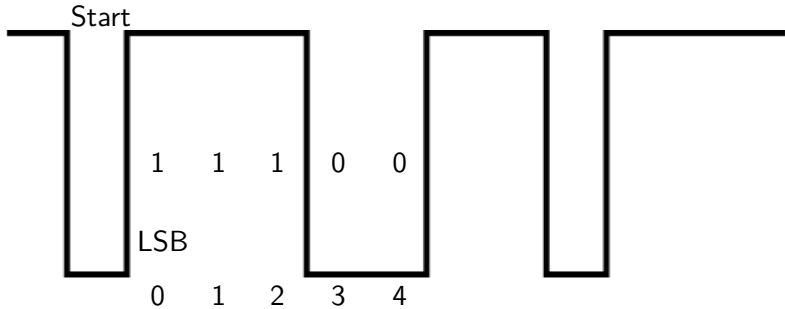


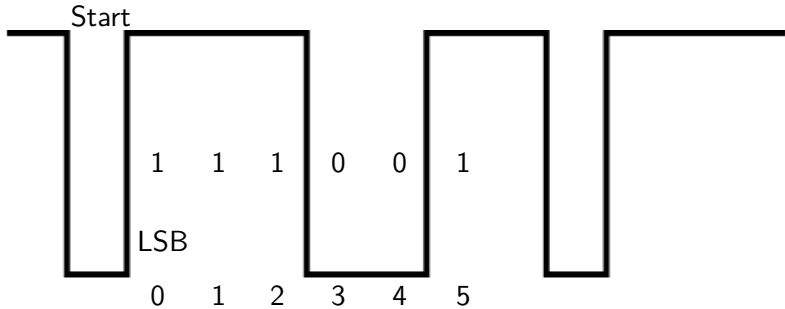


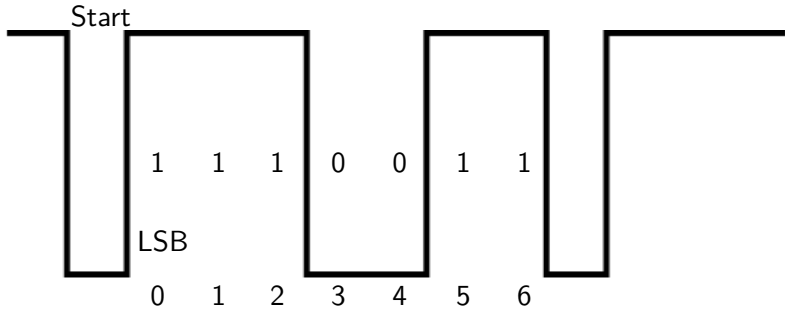


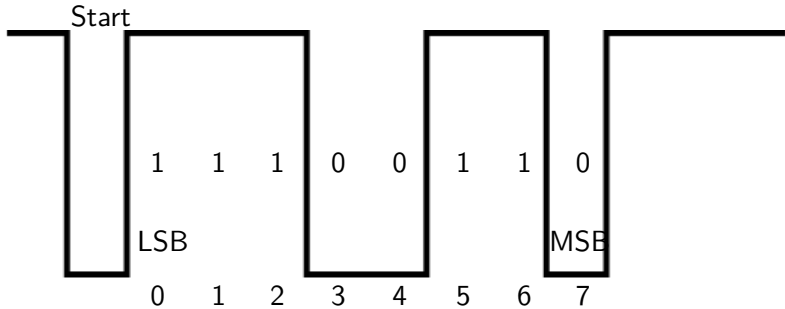


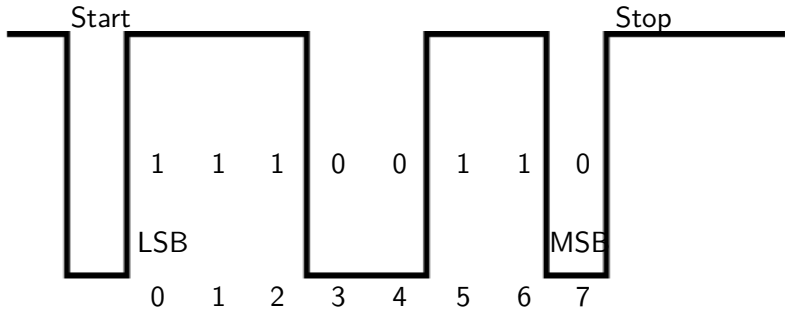




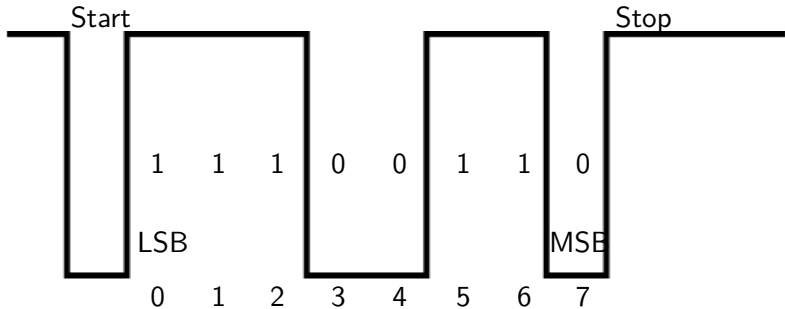




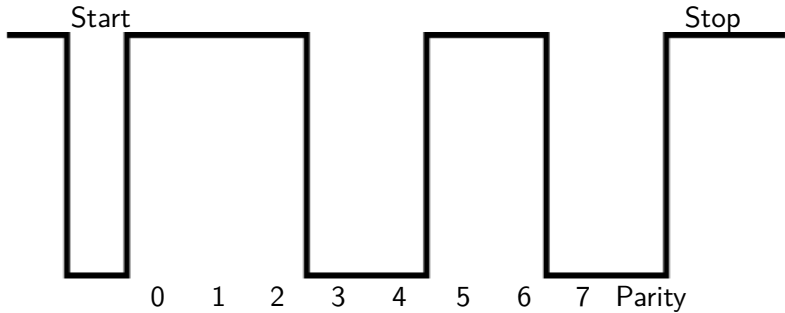


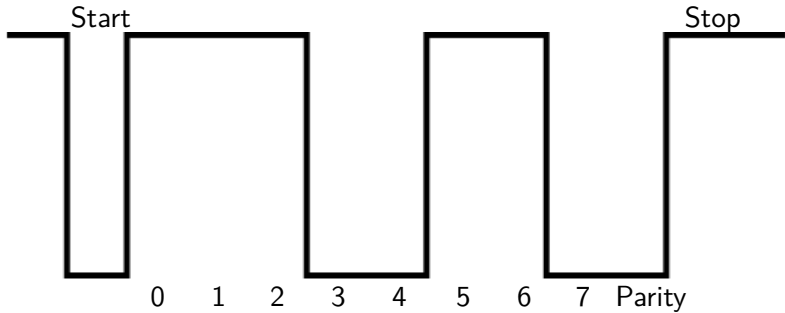




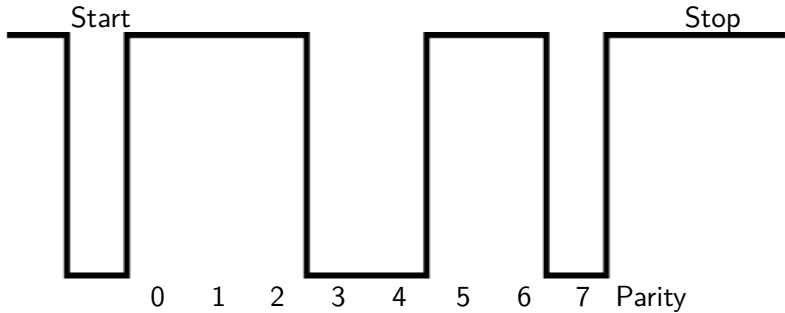


UART no parity - 01100111





UART even parity



UART odd parity

# Baud rate calculation

## Baud rate calculation

- Baud rate is the number of bits possible in a second

## Baud rate calculation

- Baud rate is the number of bits possible in a second
- e.g. 9600 baud  $\rightarrow$  1 bit takes  $\frac{1}{9600}$  second

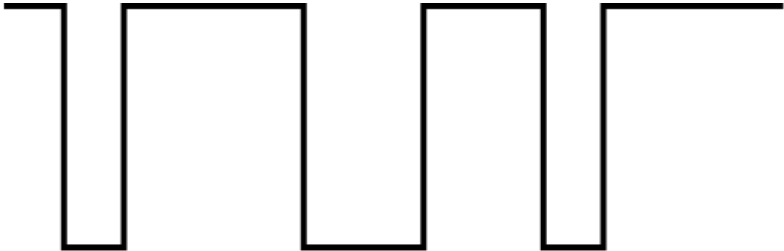
## Baud rate calculation

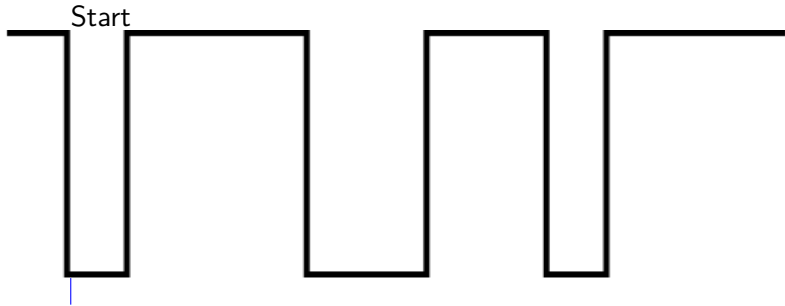
- Baud rate is the number of bits possible in a second
- e.g. 9600 baud  $\rightarrow$  1 bit takes  $\frac{1}{9600}$  second
- After start bit is detected, wait time for  $1\frac{1}{2}$  bit to test for first data bit and then after every 1 bit interval

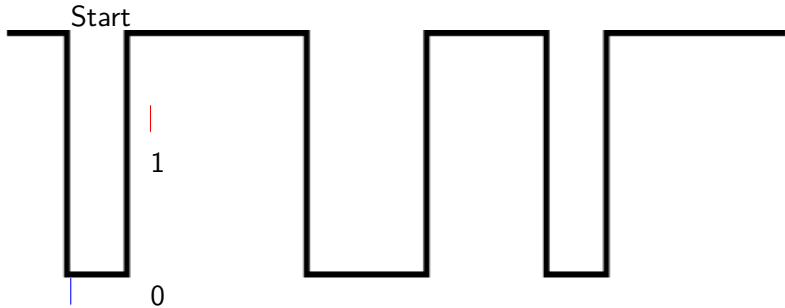


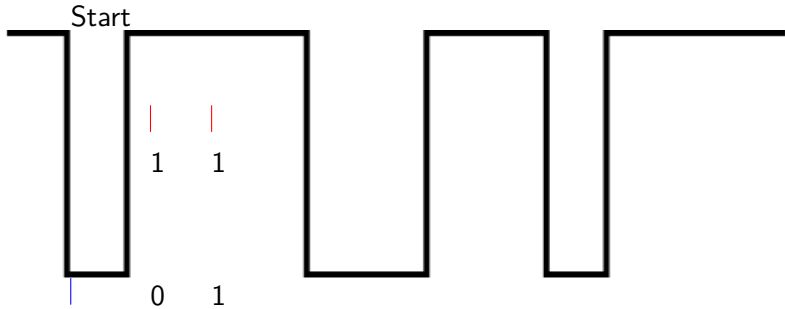
## Baud rate calculation

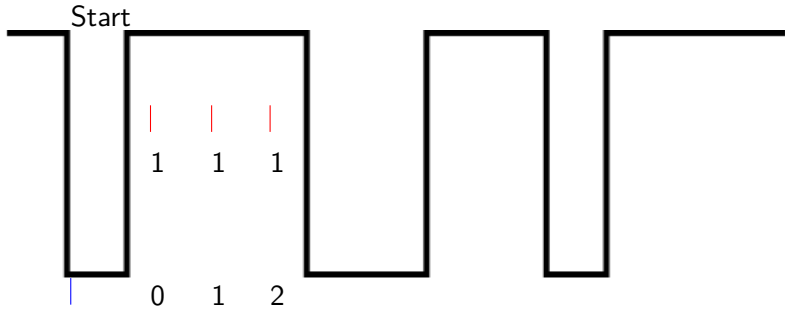
- Baud rate is the number of bits possible in a second
- e.g. 9600 baud  $\rightarrow$  1 bit takes  $\frac{1}{9600}$  second
- After start bit is detected, wait time for  $1\frac{1}{2}$  bit to test for first data bit and then after every 1 bit interval
- Resetting at the start bit allows some clock variation

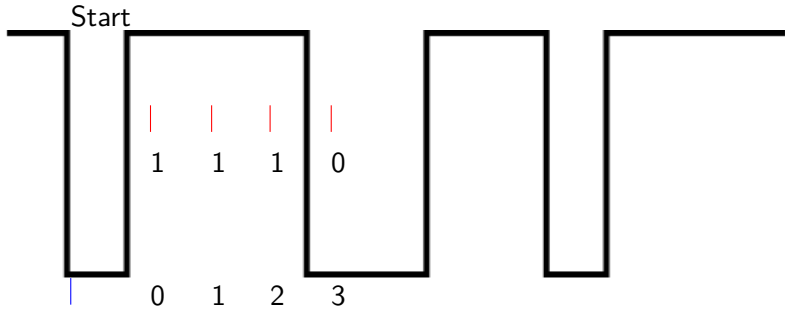


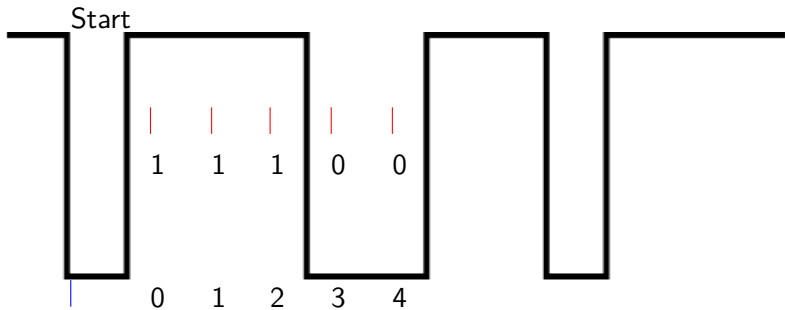




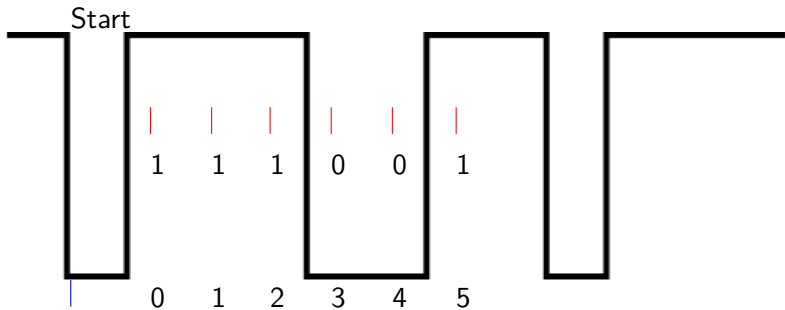


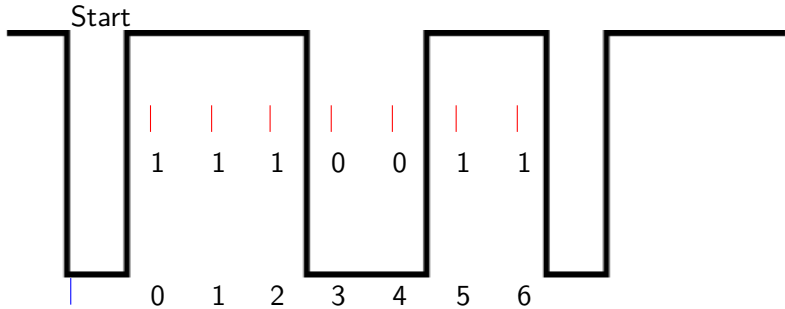


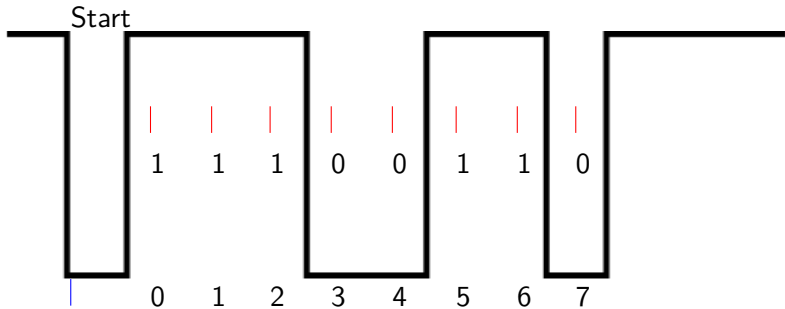


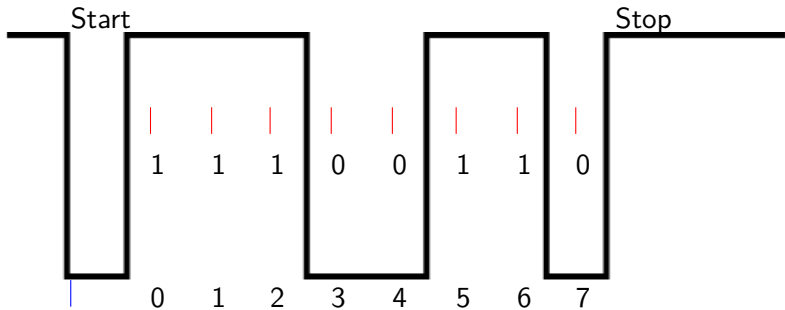


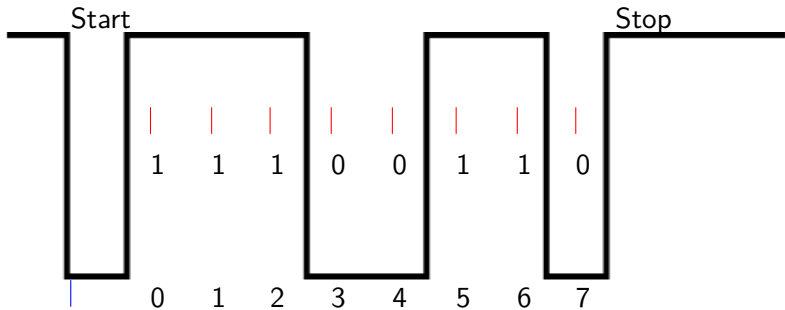












Bit timing

# RS232 communication

## RS232 communication

- Voltages are inverted

## RS232 communication

- Voltages are inverted
- $\pm 3 \rightarrow \pm 12$



## RS232 communication

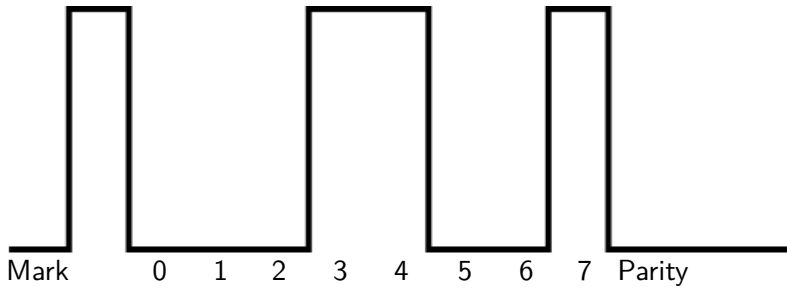
- Voltages are inverted
- $\pm 3 \rightarrow \pm 12$
- Zero is not a valid voltage

## RS232 communication

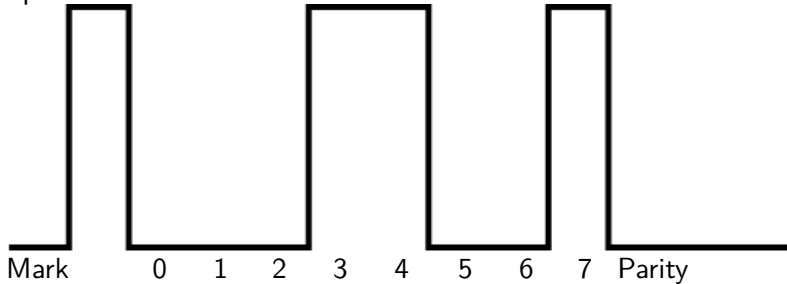
- Voltages are inverted
- $\pm 3 \rightarrow \pm 12$
- Zero is not a valid voltage
- Mark level (inactive/1) is a negative voltage

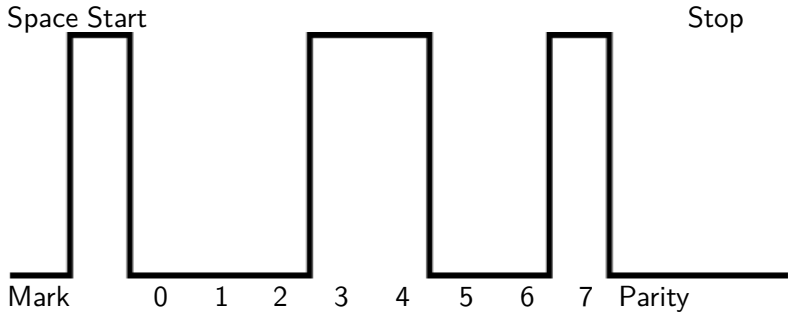
## RS232 communication

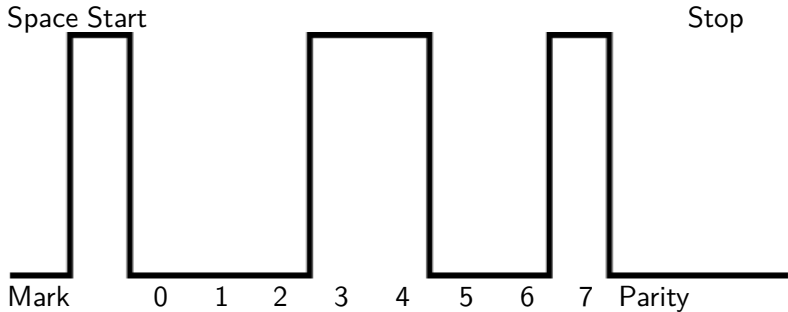
- Voltages are inverted
- $\pm 3 \rightarrow \pm 12$
- Zero is not a valid voltage
- Mark level (inactive/1) is a negative voltage
- Space level (active/0) is a positive voltage



Space Start







RS232 levels

# PySerial



# PySerial

- **ser = serial.Serial()**  
open port

# PySerial

- **ser = serial.Serial()**  
open port
- **ser.baudrate = 19200**  
set baudrate

# PySerial

- **ser = serial.Serial()**  
open port
- **ser.baudrate = 19200**  
set baudrate
- **ser.port = '//dev//ttyAMA0'**  
set port

# PySerial

- **ser = serial.Serial()**  
open port
- **ser.baudrate = 19200**  
set baudrate
- **ser.port = '//dev//ttyAMA0'**  
set port
- **ser.is\_open**  
returns *True* if open, *False* if not

# PySerial (continued)

## PySerial (continued)

- **ser.write(b'string')**  
write string

## PySerial (continued)

- **ser.write(b'string')**  
write string
- **ser.read(10)**  
read 10 bytes (or until timeout)

## PySerial (continued)

- **ser.write(b'string')**  
write string
- **ser.read(10)**  
read 10 bytes (or until timeout)
- **ser.readline()**  
read until '\n' received



## PySerial (continued)

- **ser.write(b'string')**  
write string
- **ser.read(10)**  
read 10 bytes (or until timeout)
- **ser.readline()**  
read until '\n' received
- **ser.close()**  
close port

# PySerial sample code

## PySerial sample code

```
import serial
ser = serial.Serial ("/dev/ttyAMA0")
ser.baudrate = 9600
data = ser.read(10)
#data = ser.readline()
ser.write(data)
ser.close()
```