Universal Asynchronous RX/TX (UART)

- `MX_USART2_UART_INIT()`
- P2P protocol (only between 2 devices)
- can send 7bits/8bits at a time with ***LSB first***
- uses single START bit and 1+ STOP bits
- *asynchronous* without a clock; both sides need to agree on transfer rate ⇒ **baud rate**
- RTS & CTS → handshake
- Adds an optional form of error detection (parity)
  - even parity - 1 if data has an odd number of '1'
  - odd parity - 1 if data has an even number of '1'
  - mark parity - always send a 1
  - space parity - always send a 0
- flow control
  - supported in both hardware and software

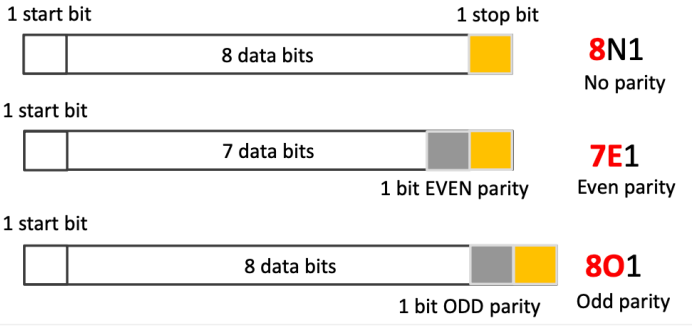
USB -> UART Adaptor

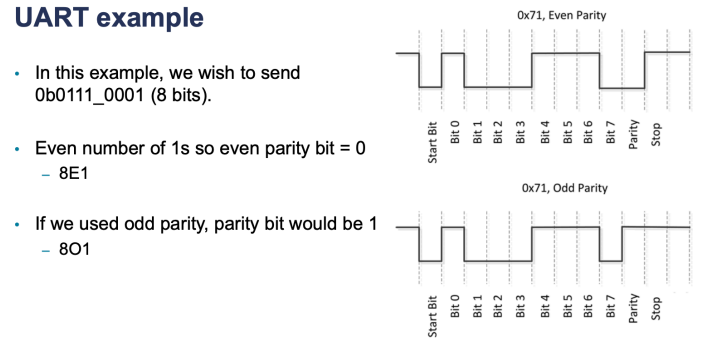
When you plug something into a USB port on your computer that appears as a serial port (COM port on Windows or /dev/tty on Linux/Mac), what's typically happening is:

1. The physical connection to your computer is USB
2. Inside the device is a USB-to-UART bridge chip (like FTDI FT232, CP2102, CH340)
3. This bridge chip:
   - Connects to your computer via USB protocol
   - Converts the USB signals to UART signals for the internal device components
   - Makes the device appear as a traditional serial port to your operating system

UART Data Frames

- Common frame formats
  - 8N1 - data = 8 bits, N = no parity, 1 stop bit
  - 7E1 - data = 7 bits, E = even parity, 1 stop bit



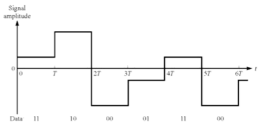


UART Modes

- **Simplex** one direction only
  - RX cannot send info back to TX (print terminal)
- **Full duplex** both dev can send and receive at the same time (like SPI)
- **Half duplex** only one dev can be sending or receiving at a time (like I2C)

Bit rate VS Baud rate

- baud rate = symbols per second
- some systems (i.e. modems) transmit symbols



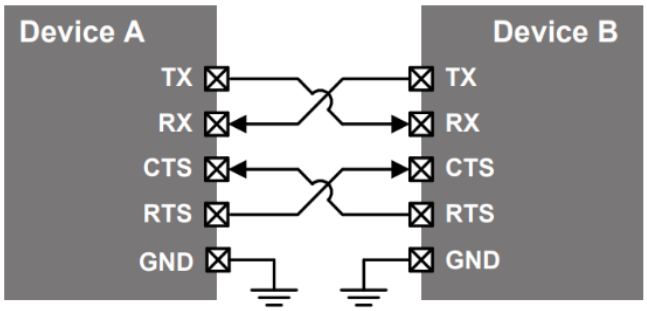
Four symbols (00, 01, 10, 11)
Each symbol = 2 bits

In this case, if baud rate = 100, we are transmitting 2 x 100 bits per second

- if transmission is down one bit at a time, baud rate = bit rate
- *baud rate on TX & RX must be the same during transmission*
  - otherwise garbage data
- 9600 and multiples are common (x1, x2, x4, x6, x12)
- **USART** Universal Synchronous/Async Receiver-Transmitter adds another wire for clock

UART Flow Control (Handshake)

- hardware flow control uses RTS (request to send) and CTS (clear to send)
  - RTS & CTS are cross connected
  - dev keeps RTS high to indicate it can receive info
  - otherwise set RTS to low to signal sender to stop transmitting
- RTS & CTS are not mandatory
  - can be emulated with ASCII char
  - dev can send 0×13 (XOFF) to tell sender to stop; to resume, send 0×11 (XON)



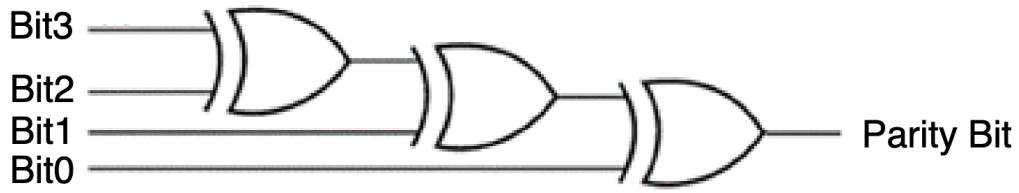
COM Port

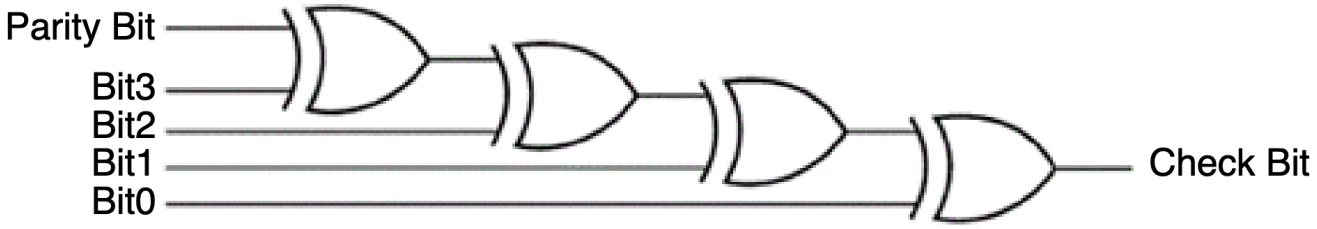
- Another name for serial port
- Since UART is p2p, PC needs to know which port to monitor

Parity

Even parity (Invert for Odd Parity)

parity





Protocols that use UART

- UART protocol defines 'how' to send and receive bits; at the analog level, there are implementations such as
  - RS-232, RS-422, RS-423, RS-485
  - all together referred to as serial protocols

	UART	RS-232	RS-422	RS485
Logic 0 (V)	~0	+3 to +15	-2 to -6	-2 to -6
Logic 1 (V)	+3.3 or +5	-3 to -15	+2 to +6	+2 to +6
Connections	Point to point only	Point to point only	Multidrop	Multidrop
Communication mode	Three possible modes	Full-duplex	Full-duplex	Half-duplex
Operation mode	Single-ended	Single-ended	Differential	Differential

Differential Signalling

- to improve signal integrity, some protocols send original signal and complement
  - complement to reduce EM interference
  - HDMI, SATA, PCIe, USB