Yrkes Akademin

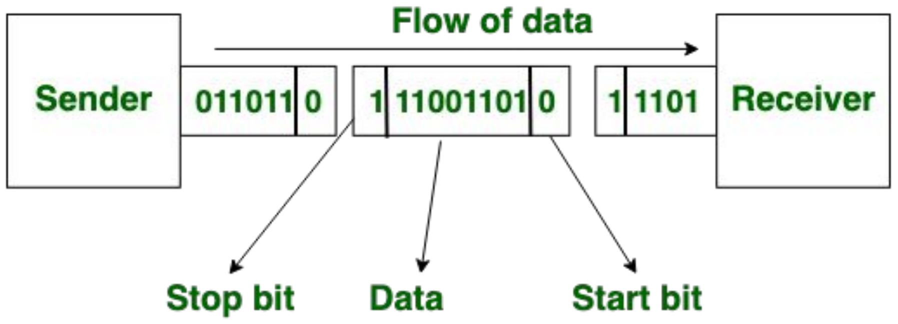
Vi hjälper dig att lyckas!

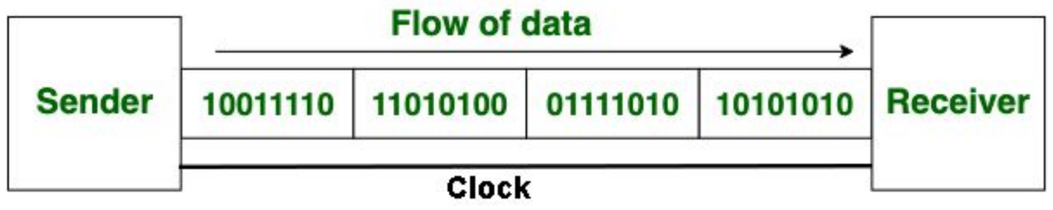
Universal Asynchronous Receiver Transmitter

Communication Protocols

Serial Communication

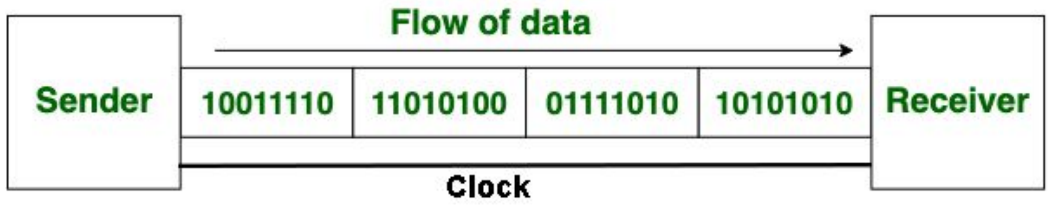
- ❖ Asynchronous Serial Communication
  - ➢ Communication without dedicated clock signal
  - ➢ Both parts need to agree on the speed
  - ➢ No dedicated master or slave.
- ❖ Synchronous Serial Communication
  - ➢ Communication with a dedicated clock
  - ➢ Usually one part is master and another part is slave
  - ➢ Speed is decided by clock. It can theoretically be different in each direction

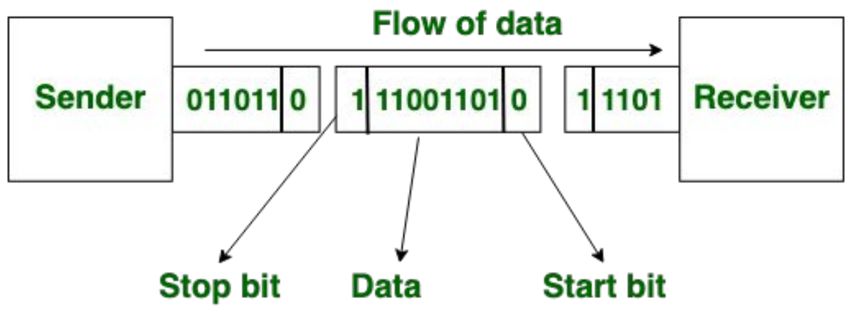




Serial Communication Protocols (Asynchronous vs. Synchronous)

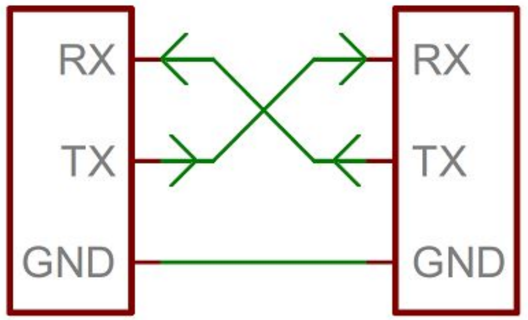
SYNCHRONOUS TRANSMISSION	ASYNCHRONOUS TRANSMISSION
Data is sent in form of blocks or frames	Data is sent in form of byte or character
A dedicated clock is required to synchronize the communication	Both parts need to agree on speed (baud rate/bit rate) Some UART Standard baud rates: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200 bps
Fast. By starting and stopping the clock signal, there is no need to have start/stop bits. Blocks of data can be sent without having any gap between the bytes.	Slow. E.g. in RS-232, up to 115200 bps
Examples: SPI, I2C, PCI Express and etc.	Examples: UART, RS-232, RS-485, CAN, and etc.

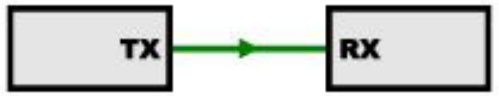


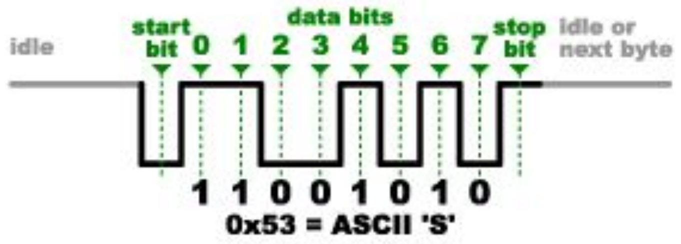


Universal Asynchronous Receiver Transmitter (UART)

- Universal Asynchronous Receiver Transmitter (UART)
  - UART defines the data link layer of the OSI model and it handles:
    - Conversion between serial and parallel data
    - Generating frames with start and stop bits
    - Generating parity bit, if needed (for data integrity)
    - Baud-Rate Generation, normally: 2400-115200 bps
    - Generating Interrupts for:
      - Transmit Complete
      - Receive Complete
    - TTL levels: <0.8V for LOW and >2.0V for HIGH
  - Supports simplex, half-duplex and full-duplex

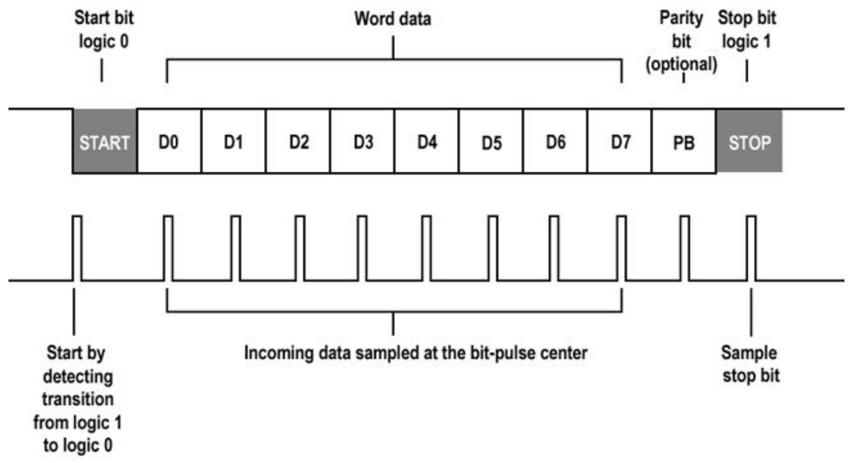






Universal Asynchronous Receiver Transmitter (UART)

- The data format and signaling
  - In the idle time the signal is HIGH
  - Transmission start with a LOW as the start bit
  - Then 6,7,8 or 9 data bits
  - Then optionally a parity bit (**E**ven, **O**dd, **N**one)
  - Then one or two stop bits (HIGH)



- Notation: *Speed/number of bits - type of parity - number of stop bits*
  - E.g. 9600/8-N-1
- Teensyduino: Teensy 3.5 has 6 UARTs (Serial1, Serial2, ... Serial6)
- Arduino: ESP32 has two UARTs (Serial1 and Serial2)

Universal Asynchronous Receiver Transmitter (UART)

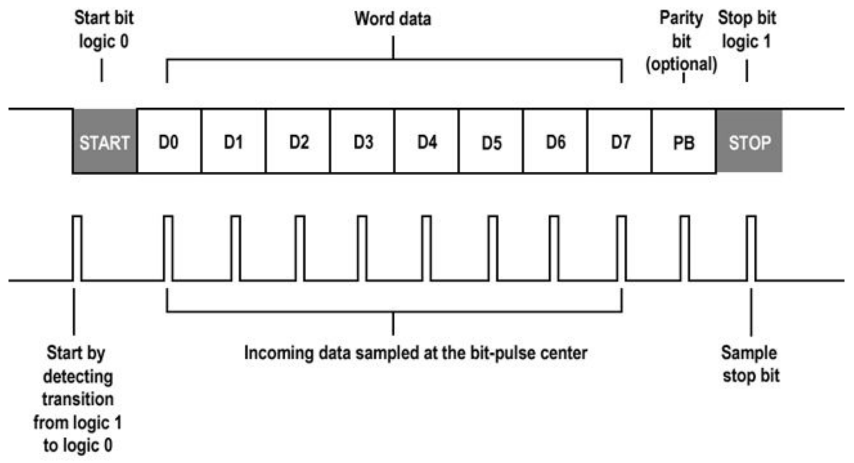
❖ Synchronization and Sampling Point

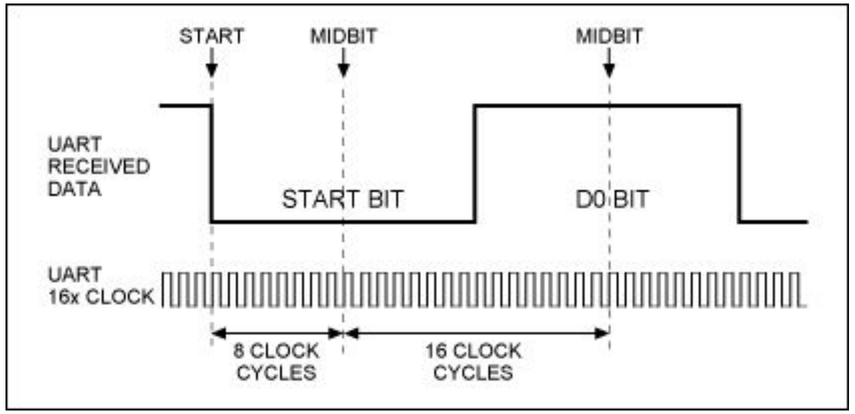
➢ There is no clock line

➢ The receiver clock should be synchronized with the transmitter clock in order to sample the bits properly in correct times

➢ The receiver clock which is 16 times faster than the agreed baud rate will be synchronized on the falling edge of start bit

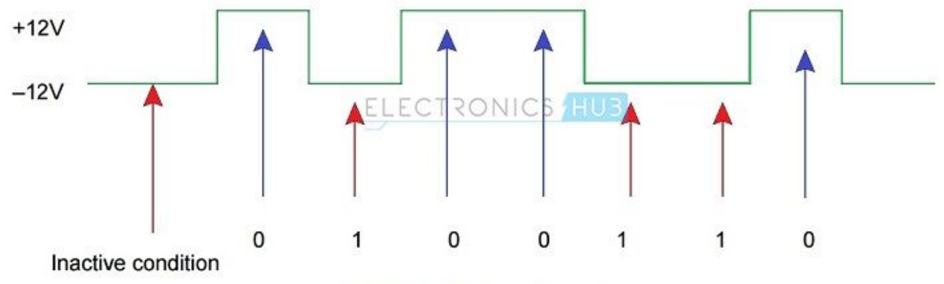
➢ The data line gets sampled in the middle of the bit; every 8 cycles of the internal clock of the receiver (8 / 16)





Serial Communication (RS-232)

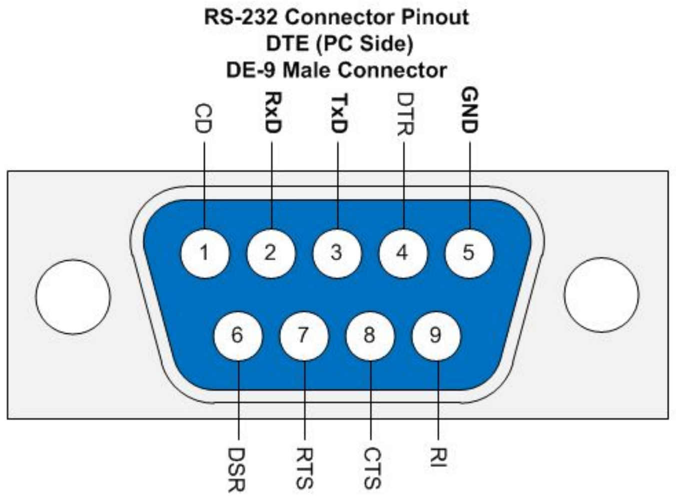
- RS-232 (Recommended Standard 232)
  - Is a standard interface used for serial communication
  - Signaling with ±12v (±3v up to ±15v)
  - Maximum Distance: 15 m
  - Maximum Bit Rate: 1 mbps
  - Communication Mode: Full duplex
  - Number of Devices: peer-to-peer (P2P)
  - Supports hardware handshaking
    - There are some signals to control data flow
  - Was used in old PCs for connecting the peripheral devices
    - But It is still used industries; E.g. PLC, CNC machines, etc.

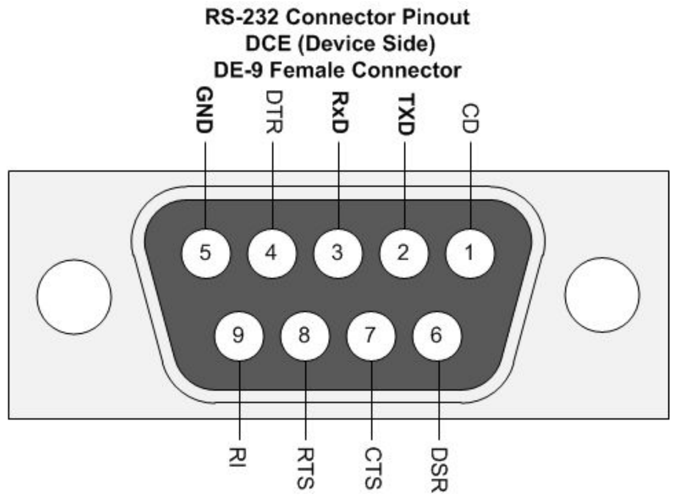


RS232 Voltage Levels

Serial Communication (RS-232)

- Pinout of the interface
  - Data Signals:
    - TxD - Transmit Data
    - RxD - Receive Data
    - GND - Signal Ground
  - Modem Control
    - RI - Ring Indicator
    - DTR - Data Terminal Ready
    - DCD - Data Carrier Detect
    - DSR - Data Set Ready
  - Flow Control
    - RTS - Request To Send
    - CTS - Clear To Send

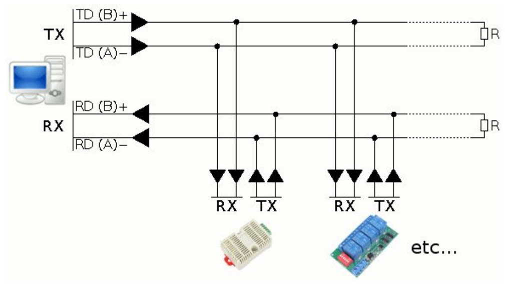


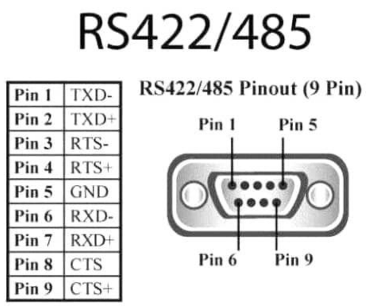


RTS & CTS are used for handshaking and data flow control. In this process, transmitter asks the receiver if it is ready to receive data then receiver checks the buffer if it is not full, then it will give signal to the transmitter that I am ready to receive data.

Serial Communication (RS-485)

- ❖ An interface used for serial communication in long distances, or in noisy conditions
- ❖ Differential signaling: ±7V on top of 0-5V (-7V to +12V)
- ❖ Maximum Distance: 1200 m
- ❖ Maximum Bit Rate: 10 mbps
- ❖ Communication Mode
  - ➢ Simplex/Half Duplex
    - ■ Tx-/Rx- : Transmit/Receive Data, Low
    - ■ Tx+/Rx+ : Transmit/Receive Data, High
  - ➢ Full Duplex
    - ■ Tx- : Transmit Data, Low
    - ■ Tx+: Transmit Data, High
    - ■ Rx- : Receive Data, Low
    - ■ Rx+ : Receive Data, High
- ❖ Number of Devices: 32 transmitters and 32 receivers
  - ➢ Master-slave arrangement in a daisy chain topology is more popular

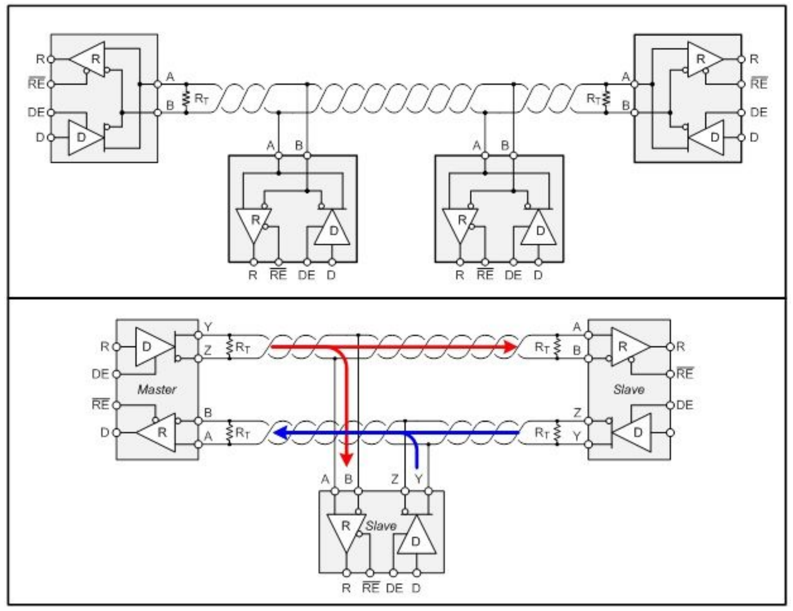



RS422/485

RS422/485 Pinout (9 Pin)

Pin 1	TXD-
Pin 2	TXD+
Pin 3	RTS-
Pin 4	RTS+
Pin 5	GND
Pin 6	RXD-
Pin 7	RXD+
Pin 8	CTS
Pin 9	CTS+

Pin 1
Pin 5
Pin 6
Pin 9



Communication Protocols

❖ Some useful links

- ➢ Serial Communication
- ➢ How does serial communications work on the Arduino?
- ➢ How the UART Serial Communication Protocol Works?
- ➢ Serial Communication Demystified
- ➢ UART Synchronization
- ➢ The Basics Of RS-232 Serial Communications
- ➢ Designing and Installing an RS485 Serial Network
- ➢ What is RS485 and How it's used in Industrial Control Systems?