UART (Universal Asynchronous Receiver-Transmitter)

Uart is a hardware communication protocol that facilitates asynchronous serial communication between devices. It is commonly used for low-speed, short-distance data exchange, especially in embedded systems, microcontrollers, and computer peripherals.

Key Characteristics of UART:

1.Asynchronous Communication:

- UART does not use a clock signal for synchronization between the transmitter and receiver.
- Instead, it relies on agreed-upon timing parameters, such as baud rate, start/stop bits, and parity.

2.How UART Works:

- **Transmitter**: Converts parallel data (e.g., 8-bit byte) into a serial bit stream.
- **Receiver**: Converts the serial bit stream back into parallel data.
- A standard UART frame includes:
  - **Start Bit**: Signals the beginning of data transmission.
  - **Data Bits**: Typically 5 to 8 bits.
  - **Parity Bit (optional)**: Provides error-checking.
  - **Stop Bit(s)**: Marks the end of the data packet.

3.Baud Rate:

- Defines the speed of communication, measured in bits per second (bps).
- Both devices must operate at the same baud rate.

4.Simple Wiring:

- UART typically requires just two lines for data exchange:
  - **TX** (Transmitter): Sends data.
  - **RX** (Receiver): Receives data.
- Ground (GND) is also needed to ensure a common reference.

5.Advantages

- Simple and widely use
- Requires minimal wiring
- Effective for short distance communication

6.Disadvantages

- Not suitable for long distances due to lack of error correction and clock synchronization.
- Limited speed compared to synchronous protocols.

Common Applications:

- Communication between microcontrollers and peripherals (e.g., sensors, displays).
- Debugging and console interfaces (e.g., via serial terminals).
- Serial communication adapters (e.g., USB-to-UART).

Comparison with Similar Protocols:

- Unlike **SPI** or **I2C**, UART is asynchronous and doesn't require a shared clock signal.
- It is point-to-point, meaning it supports direct communication between two devices, whereas SPI and I2C can support multiple devices on a single bus.