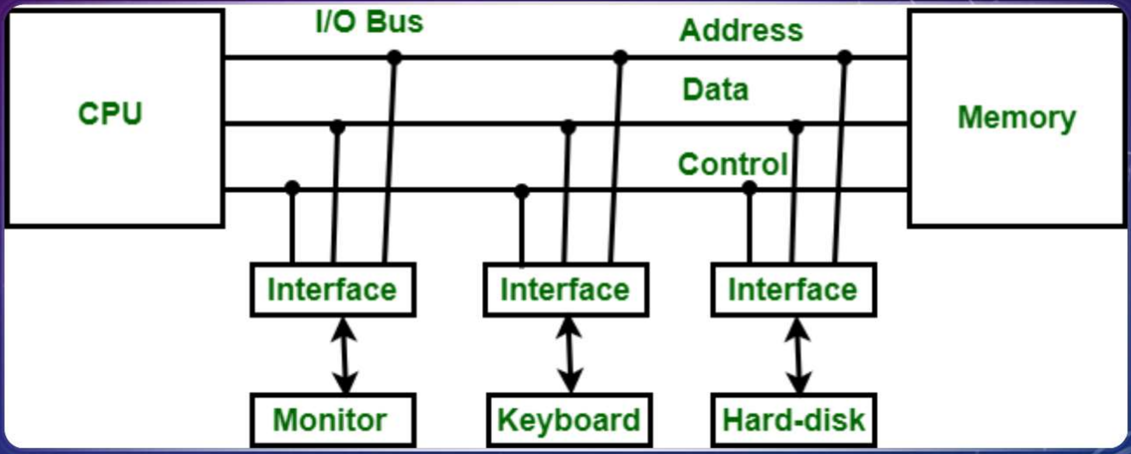
Input Output (I/O) Interface

Prof. Shankar Mali

What Is I/O Interface?

An I/O (Input/Output) Interface is a system that enables communication between a computer's central processing unit (CPU) and peripheral devices such as keyboards, mice, monitors, printers, storage devices, and network interfaces. The interface ensures smooth data transfer between the processor and these external devices.



Functions of I/O Interface?

An I/O interface performs the following key functions:

- Data Communication: Transfers data between the CPU and peripherals.
- Device Control: Sends and receives control signals to manage device operations.
- Data Buffering: Temporarily stores data to handle speed differences between CPU and I/O devices.
- Error Detection and Handling: Ensures reliable data transfer by detecting and correcting errors.

I/O Communication Methods

1. Programmed I/O (PIO)

- Programmed I/O (PIO) is a method where the CPU is responsible for managing data transfer between an I/O device and memory. In this method, the CPU executes instructions to check the status of the device and then transfers data, making it a CPU-intensive process.

2. Interrupt-Driven I/O

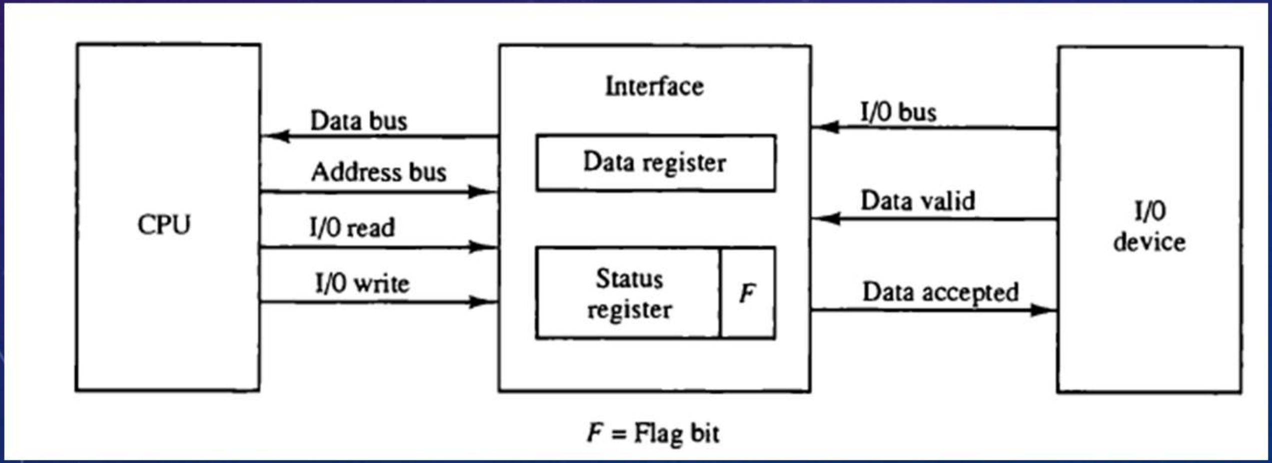
- Interrupt-Driven I/O is a method of handling input/output operations in computer systems where the CPU is interrupted when an I/O device requires attention, rather than continuously polling the device. This approach enhances system efficiency by allowing the CPU to perform other tasks while waiting for I/O operations to complete.

3. Direct Memory Access (DMA)

- **Direct Memory Access (DMA)** is an I/O technique that allows peripheral devices to transfer data directly to or from main memory without continuous CPU intervention. This improves system efficiency by freeing up the CPU for other tasks.

Working of Programmed I/O (PIO)

- CPU sends a request to the I/O device.
- CPU continuously checks (polls) the device status (busy or ready).
- Once the device is ready, the CPU transfers data to or from the device.
- CPU waits until the operation completes before continuing.
- Since the CPU is actively involved in the transfer, it is blocked from executing other tasks, making PIO inefficient for large data transfers.



Working of Interrupt Driven I/O

1. CPU Initiates I/O Request: The CPU sends a request to an I/O device and continues executing other processes.
2. I/O Device Works Independently: The device performs the requested operation without CPU intervention.
3. Interrupt Signal to CPU: Once the operation is completed, the device sends an interrupt signal to the CPU.
4. CPU Handles Interrupt:
   - The CPU pauses the current task.
   - It executes the interrupt service routine (ISR) to process the I/O request.
   - After handling the interrupt, the CPU resumes its previous task.

Working of DMA

1. CPU Requests Data Transfer: The CPU initiates the transfer by providing details (source, destination, data size) to the DMA controller (DMAC).
2. DMA Controller Handles Transfer:
   - The DMAC takes control of the system bus.
   - It transfers data directly between memory and the I/O device.
3. Interrupt to CPU: Once the transfer is complete, the DMA controller sends an interrupt to inform the CPU.
4. CPU Resumes Control: The CPU processes the data or moves to the next task.

Serial I/O Communication Interface

- **Characteristics:**
- Data Transmission: Sends data one bit at a time sequentially over a single wire or channel.
- Cabling: Requires fewer wires, typically 2 to 4 (e.g., Tx, Rx, GND, and sometimes a clock line).
- Speed: Modern serial interfaces can achieve very high speeds due to advancements in technology (e.g., USB 3.0, PCIe).
- Distance: Ideal for long-distance communication due to minimal signal degradation.
- Complexity: Simpler hardware design, fewer pins required on microcontrollers and processors.
- **Examples:**
- USB (Universal Serial Bus): Common for connecting peripherals like keyboards, mice, printers, etc.
- UART (Universal Asynchronous Receiver/Transmitter): Often used in embedded systems for serial communication.
- I2C (Inter-Integrated Circuit): Suitable for communication between integrated circuits.
- SPI (Serial Peripheral Interface): Widely used in embedded systems for high-speed data transfer.
- RS-232: An older standard used for serial communication with modems and other peripherals.

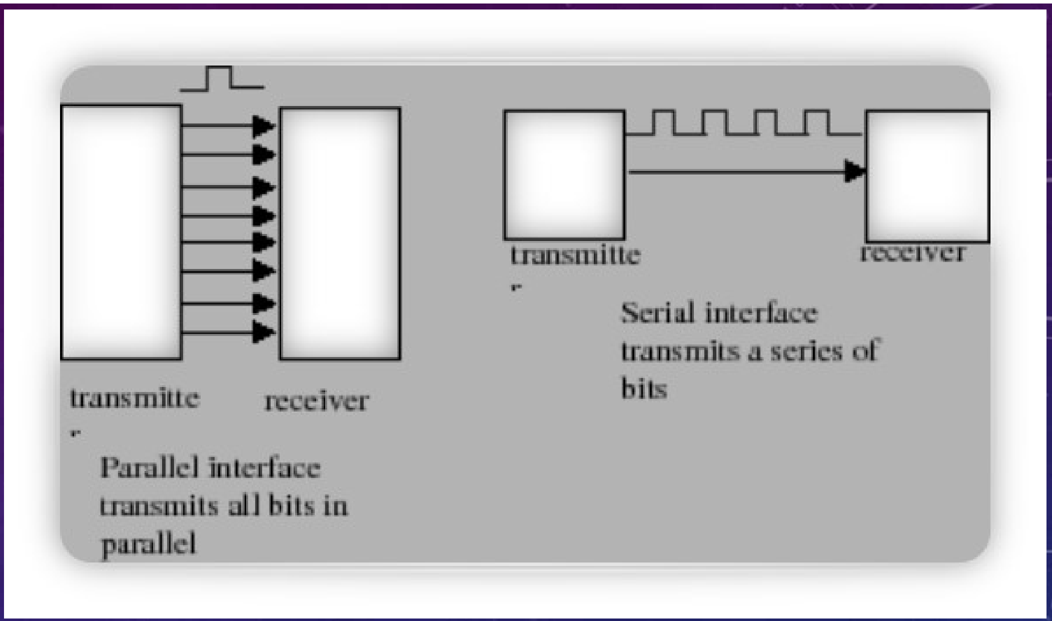
Types of serial communication Interface

- **Asynchronous Serial Communication:**
  In asynchronous communication, data is transmitted one byte at a time, with each byte framed by start and stop bits. This method does not require a shared clock signal between the sender and receiver. Instead, both devices agree on a baud rate for timing. It is commonly used in systems like UART (Universal Asynchronous Receiver/Transmitter).
- **Synchronous Serial Communication:**
  Synchronous communication uses a shared clock signal to synchronize data transfer between devices. Data is sent in a continuous stream without start and stop bits, improving efficiency. Both sender and receiver are synchronized by the clock, enabling higher data rates. Examples include SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit).

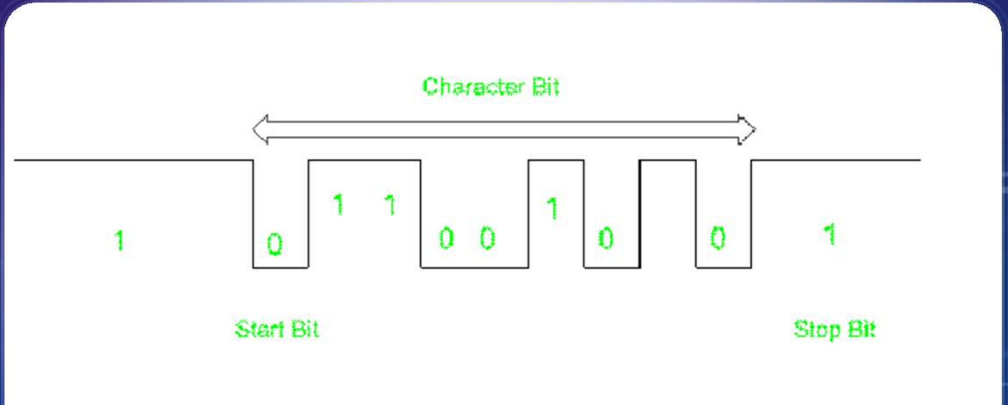
Asynchronous serial communication Interface

Asynchronous communication involves data transfer without the need for a shared clock signal between the sender and receiver. Instead, start and stop bits are used to frame each data byte.

-



transmitter
receiver
Parallel interface transmits all bits in parallel
transmitter
receiver
Serial interface transmits a series of bits



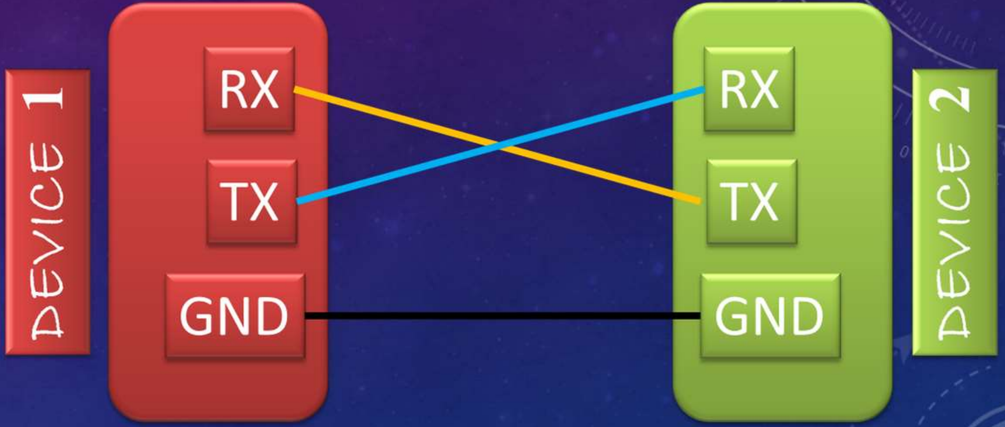
Here earlier state of line was 1 when a character has to be send a 0 is send and character bit are transferred.

Asynchronous serial communication Interface

- **Working Principle**
- Data is transmitted character-by-character, with each character framed by a start bit and one or more stop bits.
- The sender and receiver agree on a common baud rate.
- Idle state is maintained by a constant 'high' signal.
- .

Examples:

1. **UART (Universal Asynchronous Receiver/Transmitter):**
   - Used in embedded systems, microcontrollers, and serial ports.
   - Commonly used for debugging and low-speed data transfer.



2. **RS-232 (Recommended Standard 232):**
   - An older standard for serial communication, often used in modems and industrial equipment.
   - Limited to shorter distances and lower speeds.

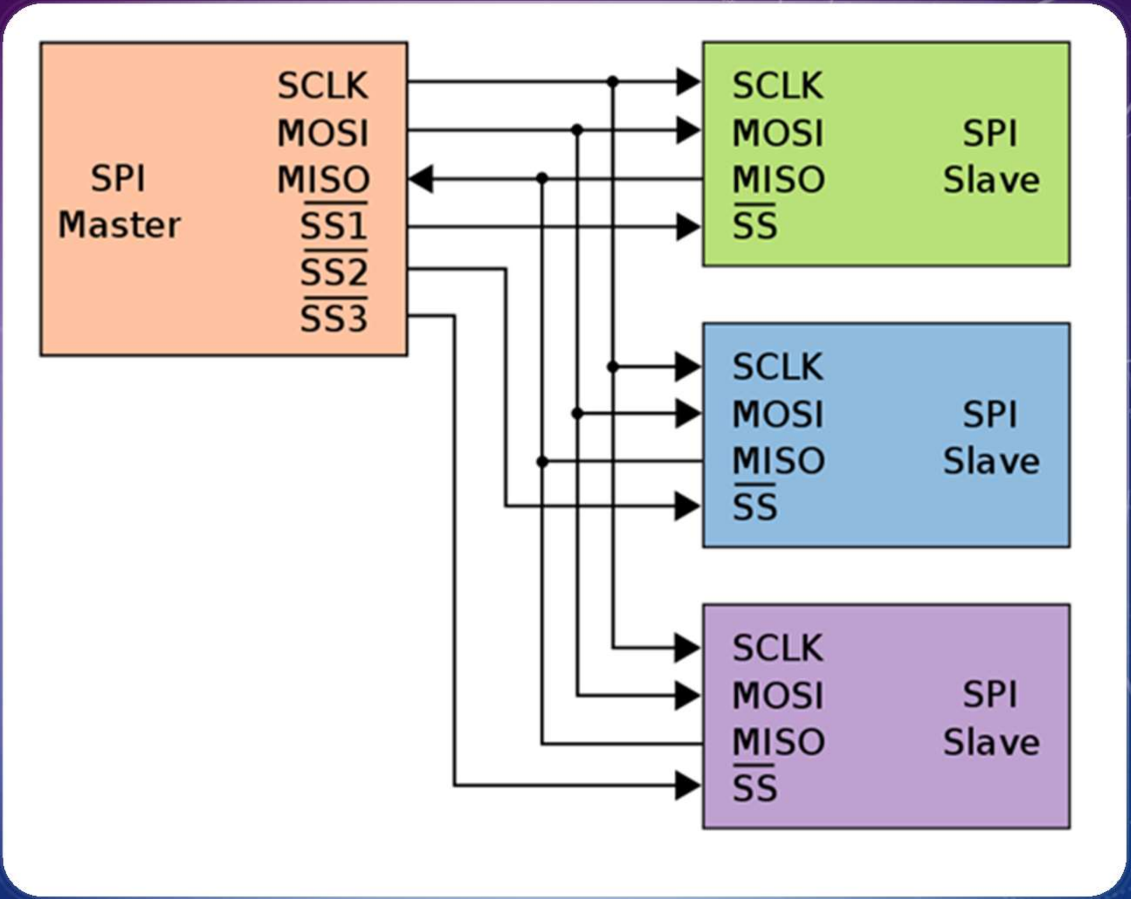
Asynchronous serial communication Interface

- Advantages
- Simple implementation.
- Cost-effective for low-speed communication.
- No need for clock synchronization.

- Disadvantages
- Lower data transfer rate due to start/stop bit overhead.
- Not suitable for high-speed data transfer.

Synchronous Serial communication

- **Data Transfer:** Data is transferred in sync with a clock signal, allowing higher speeds and accurate data sampling.
- **Synchronization:** The clock signal is shared between the transmitter and receiver.
- A synchronous serial interface always pairs its data line(s) with a clock signal, and all devices on a synchronous serial bus share a **common clock**. This makes for a more straightforward, even faster serial transfer, but it also requires at least one extra wire between communicating devices. In simple terms which **interface required SCK, SCLK pin the synchronous**.
- In synchronous method, for example in **SPI there are 4 pins used**, **2 pins** are for **MISO & MOSI** (Master In Slave Out & Master Out Slave In), **1 pin** for fixed **Clock** (from single master device) and **1 Data pin** for each slave device. Here SS1, SS2, SS3 are data pins for 3 slave devices.



Synchronous I/O Interface

- **3.1 Definition**
- **Synchronous communication involves data transfer with a continuous, synchronized clock signal shared between the sender and receiver. Data is sent in a steady stream without start and stop bits, enabling faster and more efficient communication.**
- **3.2 Working Principle**
- **Data is synchronized with the clock pulses, ensuring precise timing of data transfer.**
- **Both devices share a common clock line to maintain synchronization, reducing the chance of miscommunication.**
- **Data transfer is continuous, and the clock signal helps align data bits correctly at the receiving end. Typically involves a master-slave configuration, where the master generates the clock signal.**

Advantages

- Higher data transfer rates due to the absence of start/stop bits.
- Efficient and ideal for high-speed communication and real-time data transfer.
- Suitable for communication between devices with matched speeds.
- Supports full-duplex communication, allowing simultaneous data transmission in both directions.

Disadvantages

- More complex hardware requirements due to the need for precise clock synchronization.
- Limited to shorter distances unless additional technologies (e.g., repeaters) are used.
- Master-slave configuration may limit the flexibility of the system.

Synchronous Serial communication

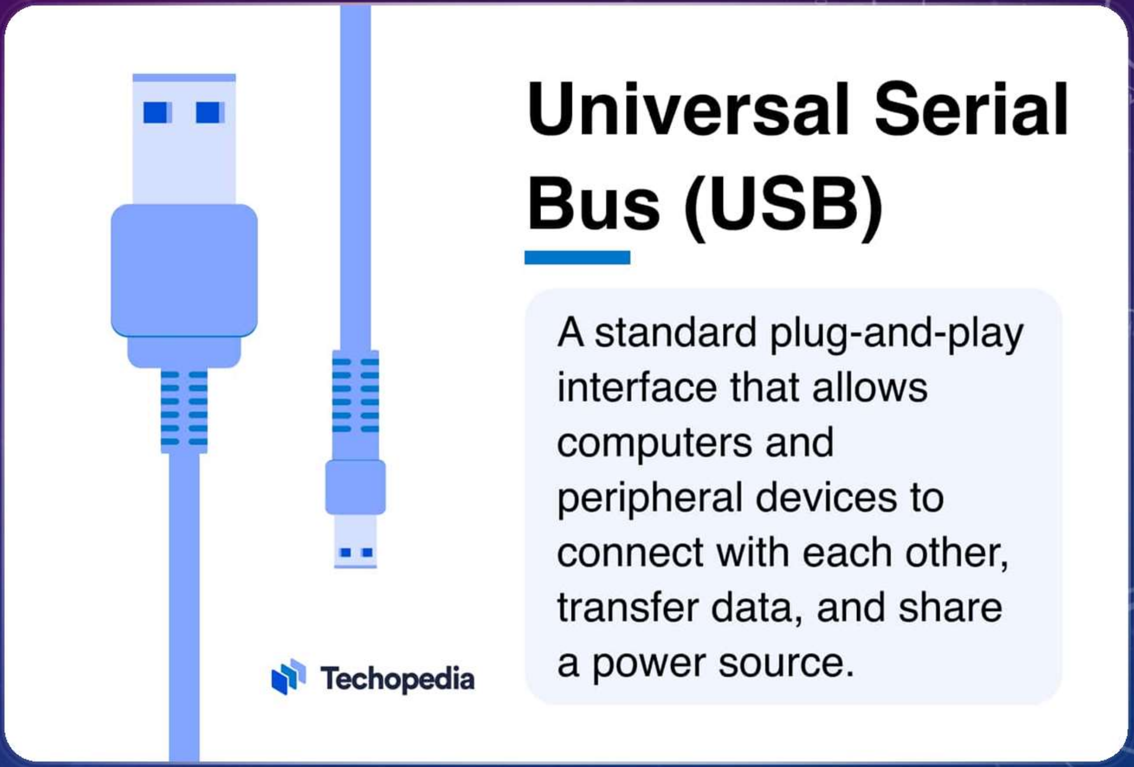
Examples:

1. **I2C (Inter-Integrated Circuit):**
   - **Bus Type:** Two-wire (SDA - Serial Data, SCL - Serial Clock).
   - **Use Case:** Communication between microcontrollers and peripheral devices like sensors and displays.
   - **Features:** Supports multiple devices with addressing, slower speeds compared to SPI.
2. **SPI (Serial Peripheral Interface):**
   - **Bus Type:** Four-wire (MISO, MOSI, SCK, SS/CS).
   - **Use Case:** High-speed communication with sensors, memory devices, and display modules.
   - **Features:** Full-duplex communication, high-speed, simple protocol but requires more lines than I2C.
3. **CAN (Controller Area Network):**
   - **Use Case:** Widely used in automotive and industrial systems for robust communication.

- **Features:** Supports multi-master communication, prioritization of messages, and error handling

Universal Serial Bus (USB)

- Versions: USB 2.0 (480 Mbps), USB 3.0 (5 Gbps), USB 3.1 (10 Gbps), USB 3.2 (20 Gbps), USB4 (up to 40 Gbps).
- Use Cases: Connecting peripherals like keyboards, mice, storage devices, cameras, and more.
- Features: Supports data transfer, power delivery, and device communication.

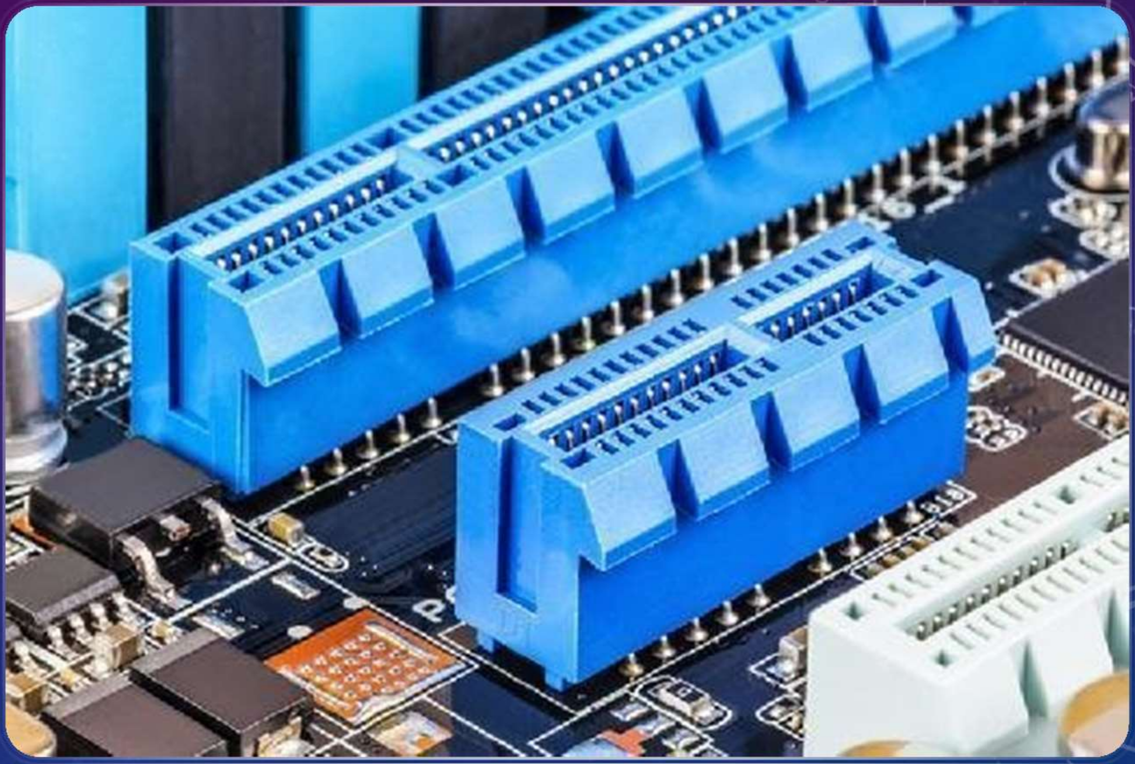


Universal Serial Bus (USB)

A standard plug-and-play interface that allows computers and peripheral devices to connect with each other, transfer data, and share a power source.

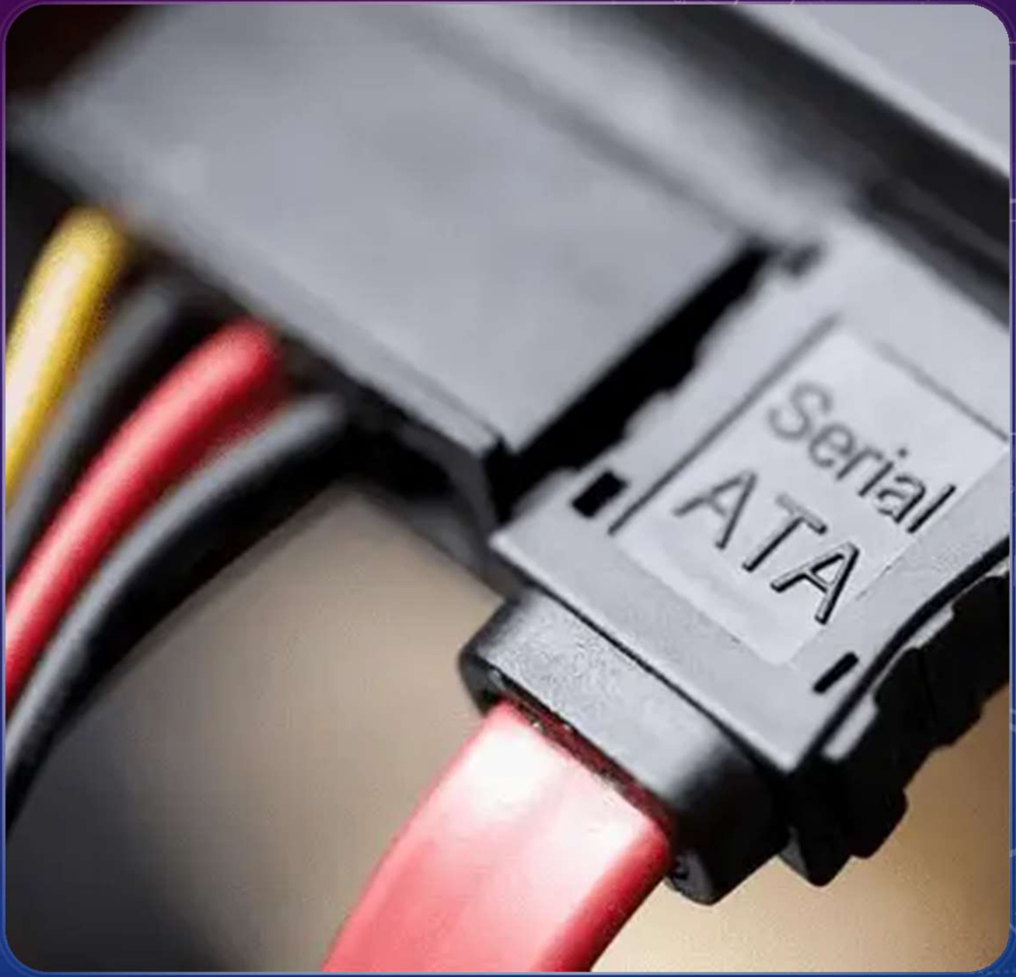
PCIe(Peripheral Component Interconnect Express)

- Speed: PCIe 4.0 (16 Gbps per lane), PCIe 5.0 (32 Gbps per lane), PCIe 6.0 (64 Gbps per lane).
- Use Cases: High-speed communication between a computer's motherboard and components like GPUs, SSDs, network cards.
- Features: Scalable architecture with multiple lanes (e.g., x1, x4, x8, x16), enabling very high bandwidth.



SATA (Serial Advanced Technology Attachment)

- **Speed: SATA III (6 Gbps).**
- **Use Cases: Connecting internal storage devices like hard drives (HDD) and solid-state drives (SSD).**
- **Features: Serial communication replaced older parallel ATA interfaces, improving speed and reliability.**
-



Parallel I/O Communication Interfaces

Parallel I/O (Input/Output) communication interfaces transmit multiple bits of data simultaneously using multiple data lines. These interfaces were widely used in earlier computer systems and devices due to their high data transfer rates at short distances. However, advancements in serial communication technologies have led to a decline in the use of parallel interfaces in many applications.

Key Characteristics of Parallel Communication Interfaces:

- **Multiple Data Lines: Typically 8, 16, 32, or more lines to transfer multiple bits simultaneously.**
- **High Data Transfer Rate: Suitable for short-distance communication where synchronization is manageable.**
- **Synchronization: Requires a separate clock line to synchronize data transfer.**
- **Bulkier Cables: More lines mean thicker and more complex cables and connectors.**
- **Susceptibility to Noise: Signal interference between parallel lines (crosstalk) can affect data integrity, especially at high speeds.**

1. PARALLEL ATA (PATA):

- Use Case: Previously used to connect storage devices like hard drives and optical drives to motherboards.
- Data Width: 16-bit parallel interface.
- Speed: Max 133 MB/s (Ultra ATA/133).
- Replacement: Replaced by SATA (Serial ATA) due to better speed, simplicity, and cable management.

2. SCSI (SMALL COMPUTER SYSTEM INTERFACE):

Parallel I/O (Input/Output) communication interfaces transmit multiple bits of data simultaneously using multiple data lines. These interfaces were widely used in earlier computer systems and devices due to their high data transfer rates at short distances. However, advancements in serial communication technologies have led to a decline in the use of parallel interfaces in many applications.

3. PCI (PERIPHERAL COMPONENT INTERCONNECT):

- Use Case: Connecting internal hardware components such as network cards, sound cards, and storage controllers.
- Data Width: 32-bit or 64-bit parallel interface.
- Speed: Max 533 MB/s (64-bit, 66 MHz).
- Replacement: Superseded by PCIe (Peripheral Component Interconnect Express), a high-speed serial interface.

4. PARALLEL PRINTER PORT (IEEE 1284):

- Use Case: Traditional interface for connecting printers and other peripherals.
- Data Width: 8-bit parallel communication.
- Speed: Max 4 MB/s.
- Modern Alternative: USB (Universal Serial Bus) offers faster speeds, better reliability, and ease of use.

5. GPIO (GENERAL-PURPOSE INPUT/OUTPUT):

- **Use Case: Often used in microcontrollers and embedded systems for parallel data transfer with sensors, displays, and other modules.**
- **Data Width: Varies, typically 8-bit or 16-bit parallel communication.**
- **Features: Simple, low-cost interface for specific hardware control and prototyping.**

SERIAL VS. PARALLEL COMMUNICATION INTERFACES: WHICH IS BETTER?

- The choice between serial and parallel communication interfaces depends on the specific requirements of the application, including speed, distance, complexity, and cost. Each interface has its strengths and weaknesses, and understanding these can help determine which is better for a particular scenario.
- 1. Data Transmission Method:
- Serial Communication:
  - Transmits data one bit at a time over a single or a few data lines.
  - Examples: USB, PCIe, SATA, Ethernet, SPI, I2C.
- Parallel Communication:
  - Transmits multiple bits simultaneously over multiple data lines (e.g., 8, 16, 32 lines).
  - Examples: PATA, IEEE 1284 (Parallel Printer Port), SCSI, PCI, GPIO.

2. SPEED AND BANDWIDTH:

- **Serial Communication:**
  - Can achieve very high data transfer rates using advanced technologies.
  - Examples: PCIe 6.0 (up to 256 Gbps), USB4 (up to 40 Gbps), Thunderbolt 4 (40 Gbps).
  - Better for long-distance, high-speed applications.
- **Parallel Communication:**
  - Generally faster at short distances due to simultaneous data transmission.
  - Speed limited by issues like crosstalk and signal degradation.
  - Modern applications rarely exceed 320 MB/s (e.g., Ultra-320 SCSI).
- ✅ Winner: Serial Communication — Achieves higher speeds, especially for modern applications.

3. CABLE COMPLEXITY AND COST:

- **Serial Communication:**
  - Requires fewer data lines, resulting in simpler and cheaper cables.
  - Supports differential signaling, reducing noise and improving reliability.
  - Easier to route on PCBs (Printed Circuit Boards).
- **Parallel Communication:**
  - Requires more wires (e.g., 8, 16, 32), making cables bulkier and more expensive.
  - More challenging to manage and prone to crosstalk.
  - Synchronization of parallel data lines adds complexity.
- ✅ Winner: Serial Communication — Simpler, cheaper, and more reliable cabling.

4. TRANSMISSION DISTANCE:

- **Serial Communication:**
  - Supports long-distance communication effectively (e.g., Ethernet up to 100 meters for copper cables, more with fiber optics).
  - Suitable for external peripherals and networking.
- **Parallel Communication:**
  - Limited to short distances due to timing skew and signal degradation.
  - Primarily used for internal connections within a device or a short cable length (e.g., Parallel Printer Cable within a few meters).
- ✅ Winner: Serial Communication — Ideal for both short and long-distance communication.

5. DATA INTEGRITY AND NOISE IMMUNITY

- **Serial Communication:**
  - Uses differential signaling, where two wires carry opposite signals to reduce noise.
  - Advanced error-checking mechanisms like CRC (Cyclic Redundancy Check) and ECC (Error-Correcting Code).
- **Parallel Communication:**
  - Susceptible to crosstalk between data lines, especially at high speeds.
  - Signal degradation and timing issues (e.g., skew) can affect data integrity.
- ✅ Winner: Serial Communication — Superior noise immunity and error handling

6. SYNCHRONIZATION:

- **Serial Communication:**
  - Asynchronous or synchronous modes available.
  - Asynchronous serial communication uses start/stop bits (e.g., UART), while synchronous communication uses a clock (e.g., SPI, I2C).
  - Embedded clocking (e.g., in PCIe) avoids the need for a separate clock line.
- **Parallel Communication:**
  - Requires a dedicated clock line to synchronize all data lines.
  - Timing skew (different arrival times for signals) can be a significant issue.
- ✅ Winner: Serial Communication — Easier synchronization, especially for high-speed applications.