Unit - III

Input Output Ports

Ports: The connection point acts as an interface between the computer and external devices like printers, modems, etc.

There are two types of ports :

1. **Internal Port:** It connects the system's motherboard to internal devices like hard disk, CD drive, internal Bluetooth, etc.
2. **External Port:** It connects the system's motherboard to external devices like a mouse, printer, USB, etc.

Some important types of ports are as per follows:

1. **Serial Port :**

- Used for external modems and older computer mouse
- Two versions-9pin,25pin
- Data travels at 115 kilobits per second

2. **Parallel Port :**

- Used for scanners and printers
- 25 pin model

3. **Universal Serial Bus (or USB) Port :**

- It can connect all kinds of external USB devices such as external hard disks, printers, scanners, mouse, keyboards, etc.
- Data travels at 12 megabits per second.

4**. Firewire Port :** FireWire is Apple Computer's interface standard for enabling high speed communication using serial bus. It is also called IEEE 1394 and used mostly for audio and video devices like digital camcorders.

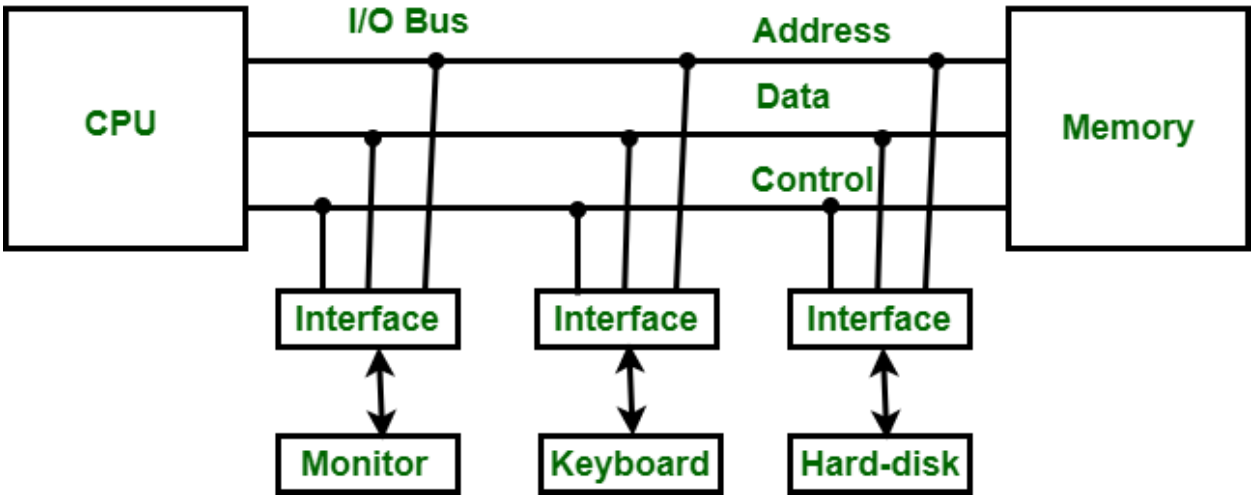
- Transfers large amounts of data at a very fast speed.
- Connects camcorders and video equipment to the computer.
- Data travels at 400 to 800 megabits per second.

5. **Ethernet Port :**

- Connects to a network and high-speed Internet.
- Data travels at 10 megabits to 1000 megabits per second depending upon the network bandwidth.

Input-Output Interface

I/O is used as an method which helps in transferring of information between the internal storage devices i.e. memory and the external peripheral device . A peripheral device is that which provide input and output for the computer, it is also called Input-Output devices. For Example: A keyboard and mouse provide Input to the computer are called input devices while a monitor and printer that provide output to the computer are called output devices. Just like the external hard-drives, there is also availability of some peripheral devices which are able to provide both input and output.



In micro-computer base system, the only purpose of peripheral devices is just to provide **special communication links** for the interfacing them with the CPU. To resolve the differences between peripheral devices and CPU, there is a special need for communication links.
The major differences are as follows:

1. The nature of peripheral devices is electromagnetic and electro-mechanical. The nature of the CPU is electronic. There is a lot of difference in the mode of operation of both peripheral devices and CPU.
2. There is also a synchronization mechanism because the data transfer rate of peripheral devices are slow than CPU.
3. In peripheral devices, data code and formats are differ from the format in the CPU and memory.
4. The operating mode of peripheral devices are different and each may be controlled so as not to disturb the operation of other peripheral devices connected to CPU.

There is a special need of the additional hardware to resolve the differences between CPU and peripheral devices to supervise and synchronize all input and output devices.

Functions of Input-Output Interface:

1. It is used to synchronize the operating speed of CPU with respect to input-output devices.
2. It selects the input-output device which is appropriate for the interpretation of the input-output signal.
3. It is capable of providing signals like control and timing signals.
4. In this data buffering can be possible through data bus.
5. There are various error detectors.
6. It converts serial data into parallel data and vice-versa.
7. It also convert digital data into analog signal and vice-versa.

Memory mapped I/O and Isolated I/O

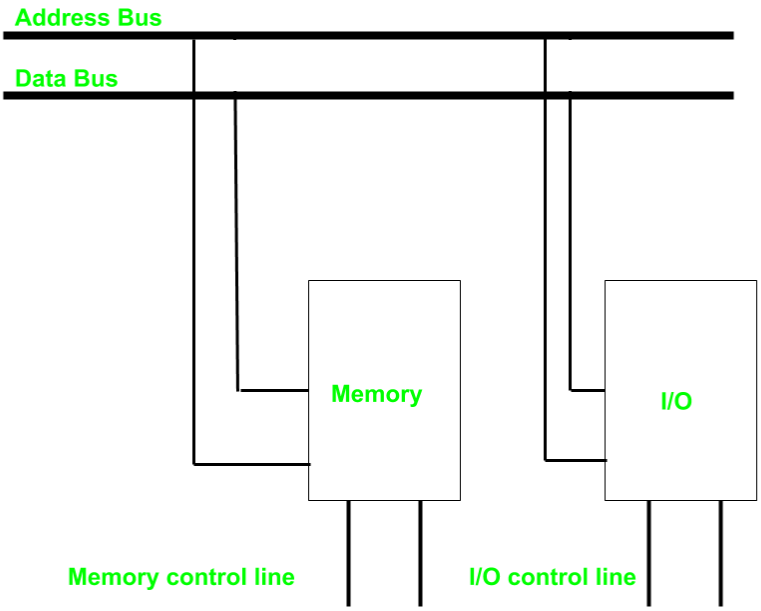
As a CPU needs to communicate with the various memory and input-output devices (I/O) as we know data between the processor and these devices flow with the help of the system bus. There are three ways in which system bus can be allotted to them :

1. Separate set of address, control and data bus to I/O and memory.
2. Have common bus (data and address) for I/O and memory but separate control lines.
3. Have common bus (data, address, and control) for I/O and memory.

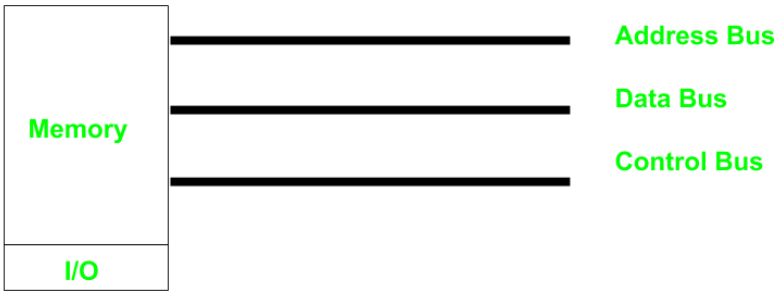
In first case it is simple because both have different set of address space and instruction but require more buses.

Isolated I/O –

Then we have Isolated I/O in which we Have common bus(data and address) for I/O and memory but separate read and write control lines for I/O. So when CPU decode instruction then if data is for I/O then it places the address on the address line and set I/O read or write control line on due to which data transfer occurs between CPU and I/O. As the address space of memory and I/O is isolated and the name is so. The address for I/O here is called ports. Here we have different read-write instruction for both I/O and memory.



Memory Mapped I/O –
In this case every bus in common due to which the same set of instructions work for memory and I/O. Hence we manipulate I/O same as memory and both have same address space, due to which addressing capability of memory become less because some part is occupied by the I/O.



Differences between memory mapped I/O and isolated I/O –

Isolated I/O	Memory Mapped I/O
Memory and I/O have separate address space	Both have same address space
All address can be used by the memory	Due to addition of I/O addressable memory become less for memory

Isolated I/O	Memory Mapped I/O
Separate instruction control read and write operation in I/O and Memory	Same instructions can control both I/O and Memory
In this I/O address are called ports.	Normal memory address are for both
More efficient due to separate buses	Lesser efficient
Larger in size due to more buses	Smaller in size
It is complex due to separate logic is used to control both.	Simpler logic is used as I/O is also treated as memory only.

Advantages of Memory-Mapped I/O:

Faster I/O Operations: Memory-mapped I/O allows the CPU to access I/O devices at the same speed as it accesses memory. This means that I/O operations can be performed much faster compared to isolated I/O.
Simplified Programming: Memory-mapped I/O simplifies programming as the same instructions can be used to access memory and I/O devices. This means that software developers do not have to use specialized I/O instructions, which can reduce programming complexity.
Efficient Use of Memory Space: Memory-mapped I/O is more memory-efficient as I/O devices share the same address space as the memory. This means that the same memory address space can be used to access both memory and I/O devices.

Disadvantages of Memory-Mapped I/O:

Limited I/O Address Space: Memory-mapped I/O limits the I/O address space as I/O devices share the same address space as the memory. This means that there may not be enough address space available to address all I/O devices.
Slower Response Time: If an I/O device is slow to respond, it can delay the CPU's access to memory. This can lead to slower overall system performance.

Advantages of Isolated I/O:

Large I/O Address Space: Isolated I/O allows for a larger I/O address space compared to memory-mapped I/O as I/O devices have their own separate address space.
Greater Flexibility: Isolated I/O provides greater flexibility as I/O devices can be added or removed from the system without affecting the memory address space.
Improved Reliability: Isolated I/O provides better reliability as I/O devices do not share the same address space as the memory. This means that if an I/O device fails, it does not affect the memory or other I/O devices.

Disadvantages of Isolated I/O:

Slower I/O Operations: Isolated I/O can result in slower I/O operations compared to memory-mapped I/O as it requires the use of specialized I/O instructions.
More Complex Programming: Isolated I/O requires specialized I/O instructions, which can lead to more complex programming.

Applications:

Memory-mapped I/O applications:

Graphics processing: Memory-mapped I/O is often used in graphics cards to provide fast access to frame buffers and control registers. The graphics data is mapped directly to memory, allowing the CPU to read from and write to the graphics card as if it were accessing regular memory.
Network communication: Network interface cards (NICs) often utilize memory-mapped I/O to transfer data between the network and the system memory. The NIC registers are mapped to specific memory addresses, enabling efficient data transfer and control over network operations.
Direct memory access (DMA): DMA controllers employ memory-mapped I/O to facilitate high-speed data transfers between devices and system memory without CPU intervention. By mapping the DMA controller registers to memory, data can be transferred directly between devices and memory, reducing CPU overhead.

Isolated I/O applications:

Embedded systems: Isolated I/O is commonly used in embedded systems where strict isolation between the CPU and peripherals is necessary. This includes applications such as industrial control systems, robotics, and automotive electronics. Isolation ensures that any faults or malfunctions in peripheral devices do not affect the stability of the entire system.
Microcontrollers: Microcontrollers often rely on isolated I/O to interface with various peripherals, such as sensors, actuators, and displays. Each peripheral is assigned a separate I/O port, allowing the microcontroller to control and communicate with multiple devices independently.

Real-time systems: Isolated I/O is preferred in real-time systems that require precise timing and deterministic behavior. By isolating the I/O operations, these systems can maintain strict control over the timing and synchronization of external events, ensuring reliable and predictable performance.

I/O Data Transfer

The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected to any computer system. These communication links are used to resolve the differences between CPU and peripheral. There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.

Mode of Transfer:
The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes. Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed I/O.
2. Interrupt- initiated I/O.
3. Direct memory access(DMA).

Now let's discuss each mode one by one.

1. **Programmed I/O:** It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices.
   Example of Programmed I/O: In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory. In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility. This is discussed below.
2. **Interrupt- initiated I/O:** Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing,

branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

- The I/O transfer rate is limited by the speed with which the processor can test and service a device.
- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.
- Terms:
    - Hardware Interrupts: Interrupts present in the hardware pins.
    - Software Interrupts: These are the instructions used in the program whenever the required functionality is needed.
    - Vectored interrupts: These interrupts are associated with the static vector address.
    - Non-vectored interrupts: These interrupts are associated with the dynamic vector address.
    - Maskable Interrupts: These interrupts can be enabled or disabled explicitly.
    - Non-maskable interrupts: These are always in the enabled state. we cannot disable them.
    - External interrupts: Generated by external devices such as I/O.
    - Internal interrupts: These devices are generated by the internal components of the processor such as power failure, error instruction, temperature sensor, etc.
    - Synchronous interrupts: These interrupts are controlled by the fixed time interval. All the interval interrupts are called as synchronous interrupts.
    - Asynchronous interrupts: These are initiated based on the feedback of previous instructions. All the external interrupts are called as asynchronous interrupts.

3. **Direct Memory Access**: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.
    1. Bus grant request time.
    2. Transfer the entire block of data at transfer rate of device because the device is usually slow than the speed at which the data can be transferred to CPU.
    3. Release the control of the bus back to CPU So, total time taken to transfer the N bytes = Bus grant request time + (N) * (memory transfer rate) + Bus release control time.
4. Buffer the byte into the buffer
5. Inform the CPU that the device has 1 byte to transfer (i.e. bus grant request)
6. Transfer the byte (at system bus speed)
7. Release the control of the bus back to CPU.

Advantages:

Standardization: I/O interfaces provide a standard way of communicating with external devices. This means that different devices can be connected to a computer using the same interface, which makes it easier to swap out devices and reduces the need for specialized hardware.
Modularity: With I/O interfaces, different devices can be added or removed from a computer without affecting the other components. This makes it easier to upgrade or replace a faulty device without affecting the rest of the system.
Efficiency: I/O interfaces can transfer data between the computer and the external devices at high speeds, which allows for faster data transfer and processing times.
Compatibility: I/O interfaces are designed to be compatible with a wide range of devices, which means that users can choose from a variety of devices that are compatible with their computer's I/O interface.

Input/Output Controller

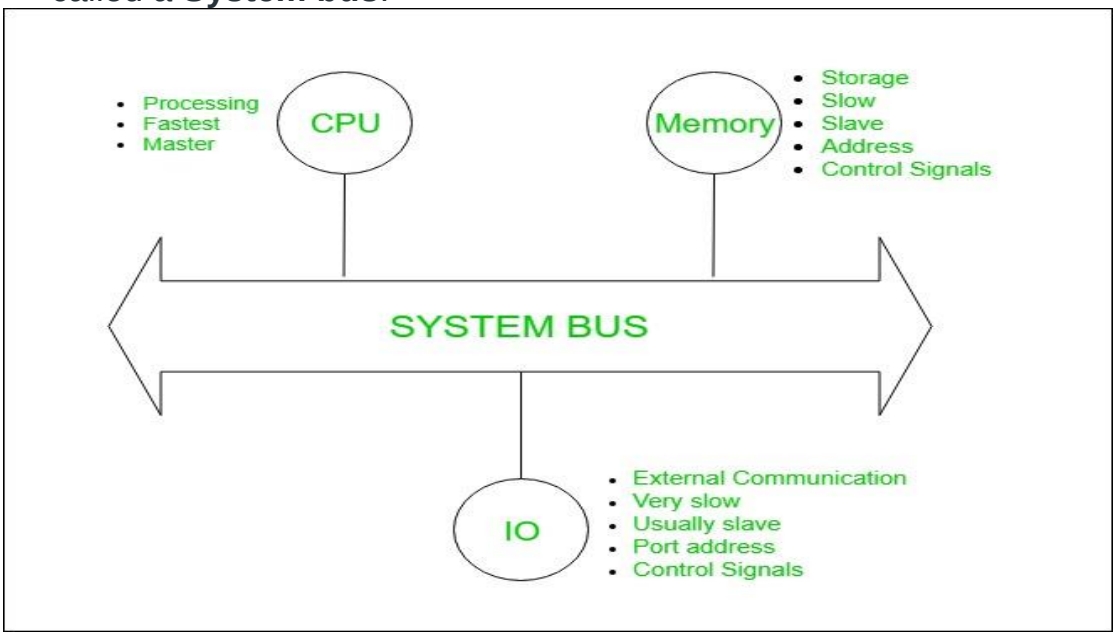
Input/Output Controller is a component that attaches with each device and is used to accept input and provide output to these devices. Applications access I/O devices with the help of these I/O controllers. Thus I/O controller is a peripheral device that enables the main processor to transfer data between the host system and I/O devices. The I/O controller is a special purpose processor and are autonomous in nature. Autonomous means that I/O controllers carry out operations on I/O devices while the main CPU continues to execute programs. The CPU controls the activities of an I/O controller by writing into and reading from I/O ports. I/0 controllers have certain registers to store data and control signals. **These registers are:**

1. The **Control register** contains that show the functioning of devices i.e. one bit shows whether the device communicates in half duplex or full duplex mode, another bit shows party checking and third bit shows the word length of data.
2. The S**tatus register** holds bits that show the status of I/O command. These bits indicate the success, busy or failure status of a command.
3. **Input registers** contain the input read by the end user.
4. The **output registers** hold the output written by the host. The program counter holds the address of next instruction to be executed by the processor.

System Bus Design

The electrically conducting path along which data is transmitted inside any digital electronic device. A Computer bus consists of a set of parallel conductors, which may be conventional wires, copper tracks on a PRINTED CIRCUIT BOARD, or microscopic aluminum trails on the surface of a silicon chip. Each wire carries just one bit, so the number of wires determines the most significant data WORD the bus can transmit: a bus with eight wires can carry only 8-bit data words and hence defines the device as an 8-bit device.

- The bus is a communication channel.
- The characteristic of the bus is shared transmission media.
- The limitation of a bus is only one transmission at a time.
- A bus used to communicate between the major components of a computer is called a **System bus**.



Asynchronous Data Transfer

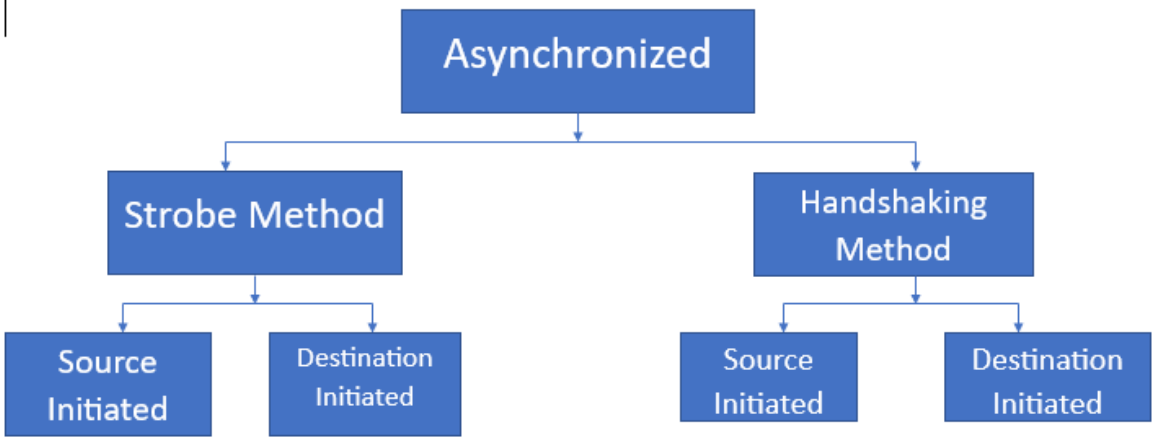
Asynchronous data transfer enables computers to send and receive data without having to wait for a real-time response. With this technique, data is conveyed in discrete units known as packets that may be handled separately. This article will explain what asynchronous data transfer is, its primary terminologies, advantages and disadvantages, and some frequently asked questions.

Terminologies used in Asynchronous Data Transfer

- **Sender**: The machine or gadget that transfers the data.
- **Receiver**: A device or computer that receives data.
- **Packet**: A discrete unit of transmitted and received data.
- **Buffer**: A short-term location for storing incoming or departing data.

Classification of Asynchronous Data Transfer

- **Strobe Control Method**
- **Handshaking Method**

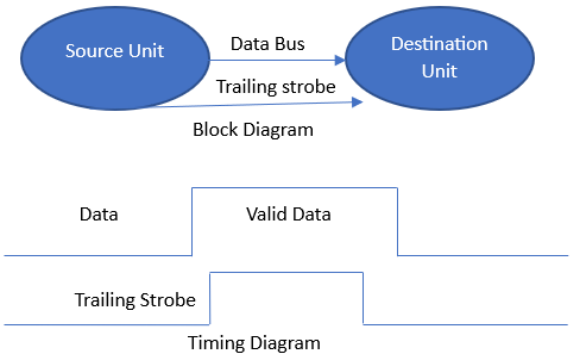


Strobe Control Method For Data Transfer

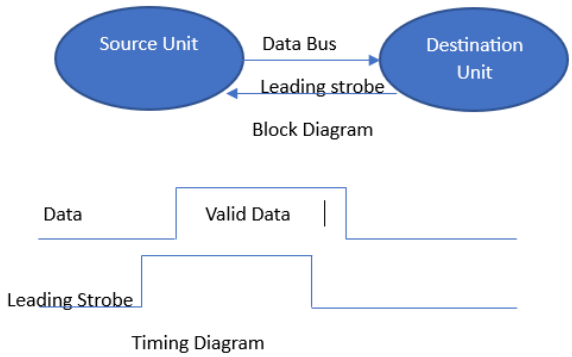
Strobe control is a method used in asynchronous data transfer that synchronizes data flow between two devices. Bits are transmitted one at a time, independently of one another, and without the aid of a clock signal in asynchronous communication. To properly receive the data, the receiving equipment needs to be able to synchronize with the transmitting device.
Strobe control involves sending data along with a different signal known as the strobe signal. The strobe signal alerts the receiving device that the data is valid and ready to be read. The receiving device waits for the strobe signal before reading the data to ensure sure it is synchronized with its clock.

The strobe signal is usually generated by the transmitting device and is sent either before or after the data. If the strobe signal is sent before the data, it is called a leading strobe. If it is sent after the data, it is called a trailing strobe.

Source Initiated Strobe



Destination Initiated Strobe



It is advantageous to utilize strobe control because it enables asynchronous data transfer, which is helpful when the participating devices have dissimilar clock rates or are not synchronized. The time of data transfer is also made more flexible by strobe control since the receiving device doesn't have to synchronize with the

transmitting device's clock; instead, it can wait for the strobe signal before reading the data.

Overall, strobe control, which is frequently employed in a range of electronic devices and systems, is a helpful technique for assuring dependable data flow in asynchronous communication.

Handshaking Method For Data Transfer

During an asynchronous data transfer, two devices manage their communication using **handshaking**. It is guaranteed that the transmitting and receiving devices are prepared to send and receive data. Handshakes are essential in asynchronous communication since there is no clock signal to synchronize the data transfer.
During handshaking, we use two types of signals mostly they are request-to-send (RTS) and clear-to-send (CTS). The receiving device is notified by an RTS signal when the transmitting equipment is ready to provide data. The receiving device responds with a CTS signal when it is ready to accept data.

once data is transmitted to the receiver end. the receiver generates a signal that it has done by sending an acknowledgment (ACK) signal. If the data is not successfully received, the receiving device will notify that a new transmission is necessary via a negative acknowledgment (NAK) signal.
The handshaking procedure guarantees synchronized and dependable data delivery. Additionally, it allows for flow management, preventing the transmitting device from sending the receiving device an excessive amount of data all at once. In order to offer flow control, handshaking signals are utilized to regulate the rate at which data is sent.

The Handshaking Method in asynchronous data transfer is used in different devices for the transfer of data to ensure reliable communication.

Advantages of Asynchronous Data Transfer

- Because asynchronous data transfer sends data in discrete, independently processable pieces, it enables faster data transfer speeds.
- This method is more effective than synchronous data transfer because there is no need for the receiver to respond.
- Transmission is done by making large files or data sets into smaller packets and sending them in parallel cuts the duration time.

Disadvantages of Asynchronous Data Transfer

- Asynchronous data transfer requires more complex programming and it may be possible that some data may get corrupted or lose data if packets are not received in the correct order or are lost during transmission.
- As we know there will be no real-time communication in asynchronous data transport can be more prone to errors than synchronous data transfer.

Peripheral Device

A **Peripheral Device** is defined as a device that provides input/output functions for a computer and serves as an auxiliary computer device without computing-intensive functionality.

A peripheral device is also called a peripheral, computer peripheral, input-output device, or I/O device.

Classification of Peripheral devices

It is generally classified into 3 basic categories which are given below:

1. Input Devices:

The input device is defined as it converts incoming data and instructions into a pattern of electrical signals in binary code that are comprehensible to a digital computer.

2. Output Devices:

An output device is generally the reverse of the input process and generally translates the digitized signals into a form intelligible to the user. The output device is also performed for sending data from one computer system to another. For some time punched card and paper tape readers were extensively used for input, but these have now been supplanted by more efficient devices.

3. Storage Devices:

Storage devices are used to store data in the system which is required for performing any operation in the system. The storage device is one of the most required devices and also provides better compatibility.

Advantages of Peripherals Devices

Peripherals devices provide more features due to this operation of the system is easy. These are given below:

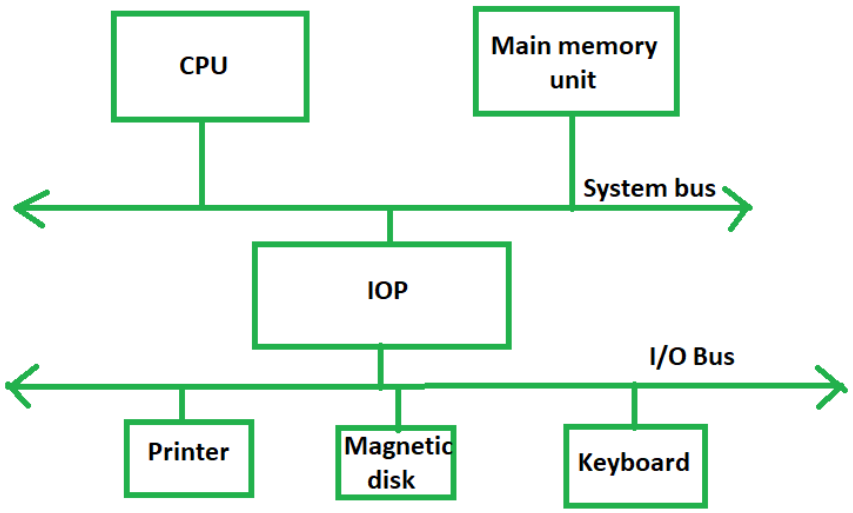
- It is helpful for taking input very easily.
- It is also provided a specific output.
- It has a storage device for storing information or data
- It also improves the efficiency of the system.

Input-Output Processor

The **DMA mode** of data transfer reduces the CPU's overhead in handling I/O operations. It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid the wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU. The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rise to the development of special purpose processors called **Input-Output Processor (IOP) or IO channels**.

The Input-Output Processor (IOP) is just like a CPU that handles the details of I/O operations. It is more equipped with facilities than those available in a typical DMA controller. The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers. In addition to the I/O-related tasks, it can perform other processing tasks like arithmetic, logic, branching, and code translation.

The main memory unit takes a pivotal role. It communicates with the processor by means of DMA.
The Input-Output Processor is a specialized processor which loads and stores data in memory along with the execution of I/O instructions. It acts as an interface between the system and devices. It involves a sequence of events to execute I/O operations and then store the results in memory.



Input-Output Processor

Features of an Input-Output Processor

- **Specialized Hardware:** An IOP is equipped with specialized hardware that is optimized for handling input/output operations. This hardware includes input/output ports, DMA controllers, and interrupt controllers.
- **DMA Capability:** An IOP has the capability to perform Direct Memory Access (DMA) operations. DMA allows data to be transferred directly between peripheral devices and memory without going through the CPU, thereby freeing up the CPU for other tasks.
- **Interrupt Handling:** An IOP can handle interrupts from peripheral devices and manage them independently of the CPU. This allows the CPU to focus on executing application programs while the IOP handles interrupts from peripheral devices.
- **Protocol Handling:** An IOP can handle communication protocols for different types of devices such as Ethernet, USB, and SCSI. This allows the IOP to interface with a wide range of devices without requiring additional software support from the CPU.
- **Buffering:** An IOP can buffer data between the CPU and peripheral devices. This allows the IOP to handle large amounts of data without overloading the CPU or the peripheral devices.
- **Command Processing:** An IOP can process commands from peripheral devices independently of the CPU. This allows the CPU to focus on executing application programs while the IOP handles peripheral device commands.
- **Parallel Processing:** An IOP can perform input/output operations in parallel with the CPU. This allows the system to handle multiple tasks simultaneously and improve overall system performance.

Applications of I/O Processors

- **Data Acquisition Systems:** I/O processors can be used in data acquisition systems to acquire and process data from various sensors and input devices. The I/O processor can handle high-speed data transfer and perform real-time processing of the acquired data.
- **Industrial Control Systems:** I/O processors can be used in industrial control systems to interface with various control devices and sensors. The I/O processor can provide precise timing and control signals, and can also perform local processing of the input data.
- **Multimedia Applications:** I/O processors can be used in multimedia applications to handle the input and output of multimedia data, such as audio and video. The I/O processor can perform real-time processing of multimedia data, including decoding, encoding, and compression.
- **Network Communication Systems:** I/O processors can be used in network communication systems to handle the input and output of data packets. The I/O processor can perform packet routing, filtering, and processing, and can also perform encryption and decryption of the data.
- **Storage Systems:** I/O processors can be used in storage systems to handle the input and output of data to and from storage devices. The I/O processor can handle high-speed data transfer and perform data caching and prefetching operations.

Advantages of Input-Output Processor

- The I/O devices can directly access the main memory without the intervention of the processor in I/O processor-based systems.
- It is used to address the problems that arise in the Direct memory access method.
- **Reduced Processor Workload:** With an I/O processor, the main processor doesn't have to deal with I/O operations, allowing it to focus on other tasks. This results in more efficient use of the processor's resources and can lead to faster overall system performance.
- **Improved Data Transfer Rates:** Since the I/O processor can access memory directly, data transfers between I/O devices and memory can be faster and more efficient than with other methods.
- **Increased System Reliability:** By offloading I/O tasks to a dedicated processor, the system can be made more fault-tolerant. For example, if an I/O operation fails, it won't affect other system processes.
- **Scalability:** I/O processor-based systems can be designed to scale easily, allowing for additional I/O processors to be added as needed. This can be particularly useful in large-scale data centers or other environments where the number of I/O devices is constantly changing.
- **Flexibility:** I/O processor-based systems can be designed to handle a wide range of I/O devices and interfaces, providing more flexibility in system design and allowing for better customization to meet specific requirements.

Disadvantages of Input-Output Processor

- **Cost:** I/O processors can add significant costs to a system due to the additional hardware and complexity required. This can be a barrier to adoption, especially for smaller systems.

- **Increased Complexity:** The addition of an I/O processor can increase the overall complexity of a system, making it more difficult to design, build, and maintain. This can also make it harder to diagnose and troubleshoot issues.
- **Limited Performance Gains:** While I/O processors can improve system performance by offloading I/O tasks from the main processor, the gains may not be significant in all cases. In some cases, the additional overhead of the I/O processor may actually slow down the system.
- **Synchronization Issues:** With multiple processors accessing the same memory, synchronization issues can arise, leading to potential data corruption or other errors.
- **Lack of Standardization:** There are many different I/O processor architectures and interfaces available, which can make it difficult to develop standardized software and hardware solutions. This can limit interoperability and make it harder for vendors to develop compatible products.