Peripherals Devices

Peripheral Device is defined as the device which provides input/output functions for a computer and serves as an auxiliary computer device without computing-intensive functionality. Generally peripheral devices, however, are not essential for the computer to perform its basic tasks, they can be thought of as an enhancement to the user's experience. A peripheral device is a device that is connected to a computer system but is not part of the core computer system architecture. Generally, more people use the term peripheral more loosely to refer to a device external to the computer case.

Classification of Peripheral devices:

It is generally classified into 4 basic categories which are given below:

1. Input Devices:

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

Example:

Keyboard, mouse, scanner, microphone etc.

Keyboard: A keyboard is an input device that allows users to enter text and commands into a computer system.

Mouse: A mouse is an input device that allows users to control the cursor on a computer screen.

Scanner: A scanner is an input device that allows users to convert physical documents and images into digital files.

Microphone: A microphone is an input device that allows users to record audio.

2. Output Devices:

An output device is generally reverse of the input process and generally translating 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.

Example:

Monitors, headphones, printers etc.

Monitor: A monitor is an output device that displays visual information from a computer system.

Printer: A printer is an output device that produces physical copies of documents or images.

Speaker: A speaker is an output device that produces audio.

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 requirement devices and also provides better compatibility.

Example:

Hard disk, magnetic tape, Flash memory etc.

Hard Drive: A hard drive is a storage device that stores data and files on a computer system.

USB Drive: A USB drive is a small, portable storage device that connects to a computer system to provide additional storage space.

Memory Card: A memory card is a small, portable storage device that is commonly used in digital cameras and smartphone.

External Hard Drive: An external hard drive is a storage device that connects to a computer system to provide additional storage space.

4. Communication Devices:

Communication devices are used to connect a computer system to other devices or networks. Examples of communication devices include:

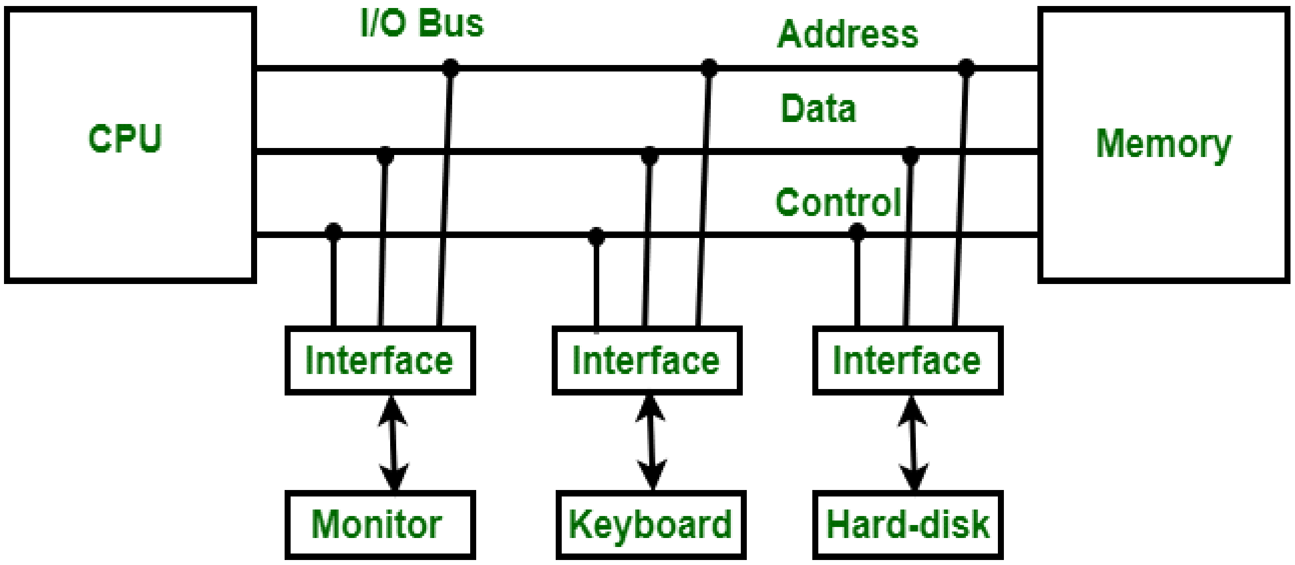
Modem: A modem is a communication device that allows a computer system to connect to the internet.

Network Card: A network card is a communication device that allows a computer system to connect to a network.

Router: A router is a communication device that allows multiple devices to connect to a network.

Introduction to Input-Output Interface

Input-Output Interface 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.



Input-Output Interface

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:

- 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.
- There is also a synchronization mechanism because the data transfer rate of peripheral devices are slow than CPU.
- In peripheral devices, data code and formats are differ from the format in the CPU and memory.
- The operating mode of peripheral devices is different and each may be controlled so as not to disturb the operation of other peripheral devices connected to CPU.

Functions of Input-Output Interface:

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

Asynchronous Data Transfer in Computer Organization

The internal operations in an individual unit of a digital system are synchronized using clock pulse. It means clock pulse is given to all registers within a unit. And all data transfer among internal registers occurs simultaneously during the occurrence of the clock pulse. Now, suppose any two units of a digital system are designed independently, such as CPU and I/O interface.

If the registers in the I/O interface share a common clock with CPU registers, then transfer between the two units is said to be synchronous. But in most cases, the internal timing in each unit is independent of each other, so each uses its private clock for its internal registers. In this case, the two units are said to be asynchronous to each other, and if data transfer occurs between them, this data transfer is called **Asynchronous Data Transfer**.

But, the Asynchronous Data Transfer between two independent units requires that control signals be transmitted between the communicating units so that the time can be indicated at which they send data. These two methods can achieve this asynchronous way of data transfer:

- **Strobe control:** A strobe pulse is supplied by one unit to indicate to the other unit when the transfer has to occur.
- **Handshaking:** This method is commonly used to accompany each data item being transferred with a control signal that indicates data in the bus. The unit receiving the data item responds with another signal to acknowledge receipt of the data.

The strobe pulse and handshaking method of asynchronous data transfer is not restricted to I/O transfer. They are used extensively on numerous occasions requiring the transfer of data between two independent units. So, here we consider the transmitting unit as a source and receiving unit as a destination.

For example, the CPU is the source during output or write transfer and the destination unit during input or read transfer.

Therefore, the control sequence during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination.

So, while discussing each data transfer method asynchronously, you can see the control sequence in both terms when it is initiated by source or by destination. In this way, each data transfer method can be further divided into parts, source initiated and destination initiated.

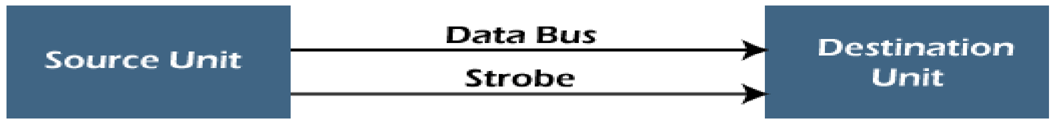
Asynchronous Data Transfer Methods

The asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate when they send the data. Thus, the two methods can achieve the asynchronous way of data transfer.

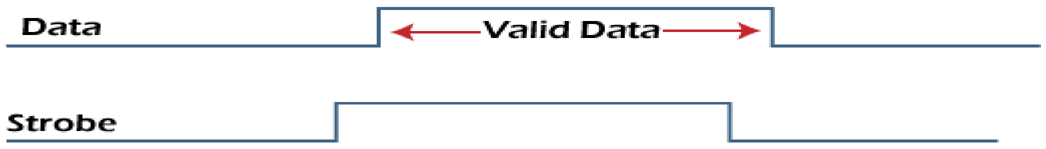
1. Strobe Control Method

The Strobe Control method of asynchronous data transfer employs a single control line to time each transfer. This control line is also known as a strobe, and it may be achieved either by source or destination, depending on which initiate the transfer.

- **Source initiated strobe:** In the below block diagram, you can see that strobe is initiated by source, and as shown in the timing diagram, the source unit first places the data on the data bus.



(a) Block Diagram

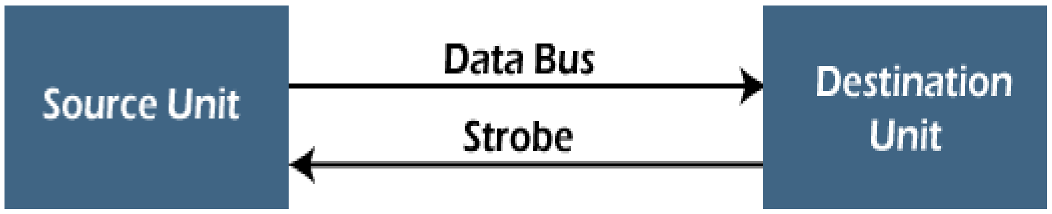


(b) Timing Diagram

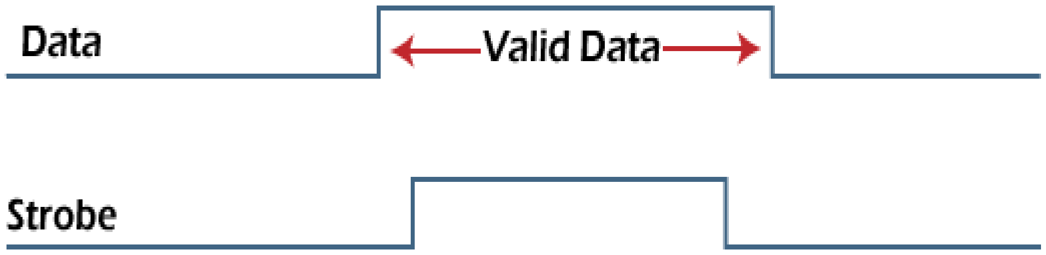
After a brief delay to ensure that the data resolve to a stable value, the source activates a strobe pulse. The information on the data bus and strobe control signal remains in the active state for a sufficient time to allow the destination unit to receive the data.

The destination unit uses a falling edge of strobe control to transfer the contents of a data bus to one of its internal registers. The source removes the data from the data bus after it disables its strobe pulse. Thus, new valid data will be available only after the strobe is enabled again.
In this case, the strobe may be a memory-write control signal from the CPU to a memory unit. The CPU places the word on the data bus and informs the memory unit, which is the destination.

Destination initiated strobe: In the below block diagram, you see that the strobe initiated by destination, and in the timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data.



(a) Block Diagram



(b) Timing Diagram

The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain on the bus long enough for the destination unit to accept it.
The falling edge of the strobe pulse can use again to trigger a destination register. The

destination unit then disables the strobe. Finally, and source removes the data from the data bus after a determined time interval.
In this case, the strobe may be a memory read control from the CPU to a memory unit. The CPU initiates the read operation to inform the memory, which is a source unit, to place the selected word into the data bus.

2. Handshaking Method

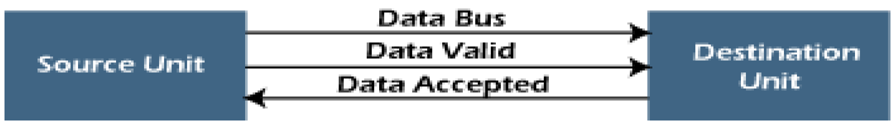
The strobe method has the disadvantage that the source unit that initiates the transfer has no way of knowing whether the destination has received the data that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has placed data on the bus.

So this problem is solved by the handshaking method. The handshaking method introduces a second control signal line that replays the unit that initiates the transfer.

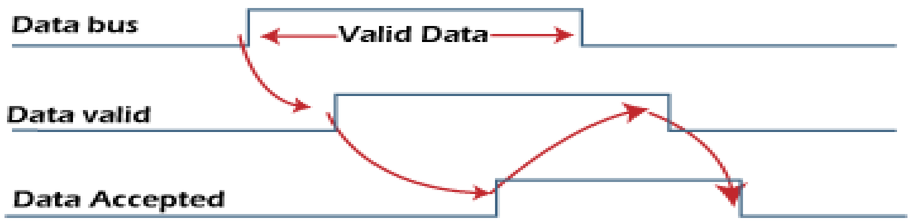
In this method, one control line is in the same direction as the data flow in the bus from the source to the destination. The source unit uses it to inform the destination unit whether there are valid data in the bus.

The other control line is in the other direction from the destination to the source. This is because the destination unit uses it to inform the source whether it can accept data. And in it also, the sequence of control depends on the unit that initiates the transfer. So it means the sequence of control depends on whether the transfer is initiated by source and destination.

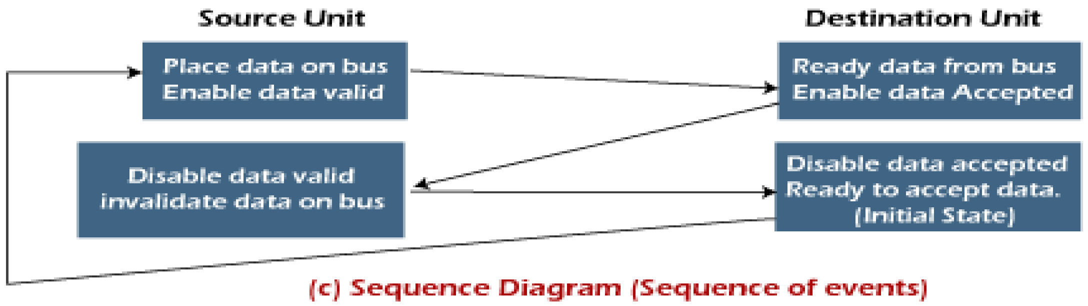
- **Source initiated handshaking:** In the below block diagram, you can see that two handshaking lines are "**data valid**", which is generated by the source unit, and "**data accepted**", generated by the destination
unit.



(a) Block Diagram



(b) Timing Diagram



(c) Sequence Diagram (Sequence of events)

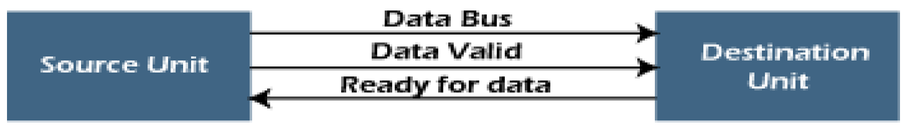
The timing diagram shows the timing relationship of the exchange of signals between the two units. The source initiates a transfer by placing data on the bus and enabling its data valid signal. The destination unit then activates the data accepted signal after it accepts the data from the bus.

The source unit then disables its valid data signal, which invalidates the data on the bus. After this, the destination unit disables its data accepted signal, and the system goes into its initial state. The source unit does not send the next data item until after the destination unit shows readiness to accept new data by disabling the data accepted signal.

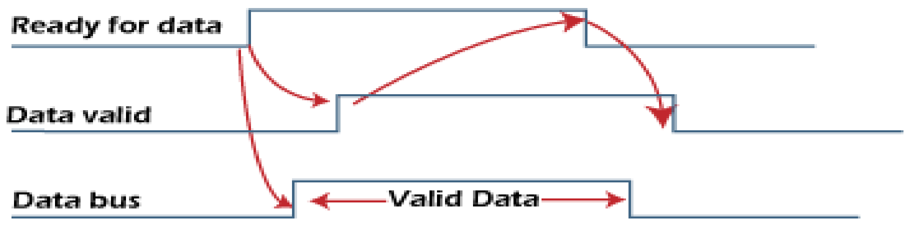
This sequence of events described in its sequence diagram, which shows the above sequence in which the system is present at any given time.

•**Destination initiated handshaking:** In the below block diagram, you see that the two handshaking lines are "**data valid**", generated by the source unit, and "**ready for data**" generated by the destination unit.

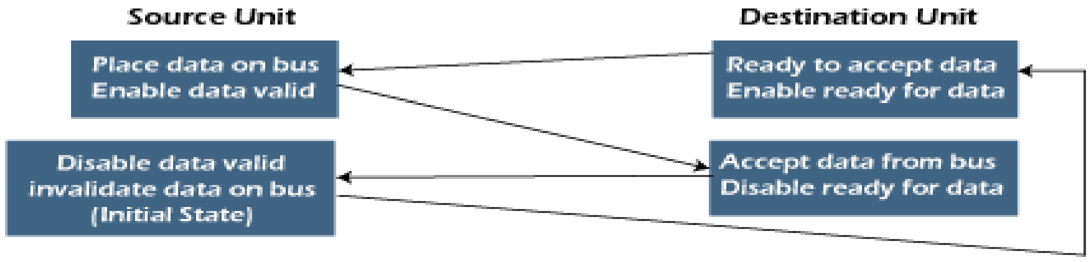
Note that the name of signal data accepted generated by the destination unit has been changed to ready for data to reflect its new meaning.



(a) Block Diagram



(b) Timing Diagram



(c) Sequence Diagram (Sequence of events)

The destination transfer is initiated, so the source unit does not place data on the data bus until it receives a ready data signal from the destination unit. After that, the handshaking process is the same as that of the source initiated.

The sequence of events is shown in its sequence diagram, and the timing relationship between signals is shown in its timing diagram. Therefore, the sequence of events in both cases would be identical.

Advantages of Asynchronous Data Transfer

Asynchronous Data Transfer in computer organization has the following advantages, such as:

- It is more flexible, and devices can exchange information at their own pace. In addition, individual data characters can complete themselves so that even if one packet is corrupted, its predecessors and successors will not be affected.
- It does not require complex processes by the receiving device. Furthermore, it means that inconsistency in data transfer does not result in a big crisis since the device can keep

up with the data stream. It also makes asynchronous transfers suitable for applications where character data is generated irregularly.

Disadvantages of Asynchronous Data Transfer

There are also some disadvantages of using asynchronous data for transfer in computer organization, such as:

The success of these transmissions depends on the start bits and their recognition. Unfortunately, this can be easily susceptible to line interference, causing these bits to be corrupted or distorted.

A large portion of the transmitted data is used to control and identify header bits and thus carries no helpful information related to the transmitted data. This invariably means that more data packets need to be sent.