UNIT-3

Input / output Device

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 3 basic categories which are given below:

- **Input Devices:**
  The input devices 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.
  Example: keyboard, mouse, scanner etc.
- **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.
- **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 provide better compatibility. Example: hard disk, magnetic tape, flash memory etc.

Advantage of Peripherals Devices:
Peripherals devices provides more feature 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.

Interrupt

In **computer architecture**, an **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

Our processor repeatedly keeps on checking for the SIN and SOUT bits for the synchronization in the program. Hence, the processor does not do any things useful other than running the infinite loop. To avoid this situation input/output device can have the concept of interrupts . When the input/output device is ready it could signal the processor on a separate line called interrupt request line. On receiving the interrupt the processor reads the input output device and hence removing the infinite loop waiting mechanism.

Type of Interrupts

1. **Hardware Interrupts:** If the signal for the processor is from external device or hardware is called hardware interrupts. **Example:** from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts.

Hardware interrupts can be classified into two types

- **Maskable Interrupt:** The hardware interrupts which can be delayed or disabled when a much highest priority interrupt has occurred to the processor.

- **Non Maskable Interrupt:** The hardware which cannot be delayed or disabled and should process by the processor immediately.

2. **Software Interrupts:** Software interrupt can also divided in to two types they are

- **Normal Interrupts:** the interrupts which are caused by the software instructions are called software instructions.

- **Exception:** unplanned interrupts while executing a program is called Exception. For example: while executing a program if we got a value which should be divided by zero is called a exception.

Need for Interrupts

The operating system is a reactive program. When you give some input it will perform computations and produces output but meanwhile you can interact with the system by interrupting the running process or you can stop and start another process. This reactiveness is due to the interrupts. Modern operating systems are interrupt driven

Interrupt Service Routine and its working

The routine that gets executed when an interrupt request is made is called as interrupt service routine.

- **Step 1:** When the interrupt occurs the processor is currently executing **i'th** instruction and the program counter will be currently pointing to **(i + 1)th** instruction.

- **Step 2:** When the interrupt occurs the program counter value is stored on the processes stack.

- **Step 3:** The program counter is now loaded with the address of interrupt service routine.

- **Step 4:** Once the interrupt service routine is completed the address on the processes stack is pop and place back in the program counter.

- **Step 5:** Execution resumes from **(i + 1)th** line of COMPUTE routine.

I/O Interface & Mode of 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. Type of I/O interface:

Non programmed I/O interface: In Non programmed I/O interface, configuration is pre-defined during the design time. E.g. buffer, latch.

Programmed I/O interface: In programmed I/O interface, configuration is defined in the command word register.

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 Data 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)
4. I/O Channels and Processors

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.

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.

   Note: Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the processor to transfer data between memory and the I/O module, and any data transfer must transversea path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.

   - The processor is tied up in managing an I/O transfer a number of instructions must be executed for each I/O transfer.
   - The I/O transfer rate is limited by the speed with which the processor can test and service a device.

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.

Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.

Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

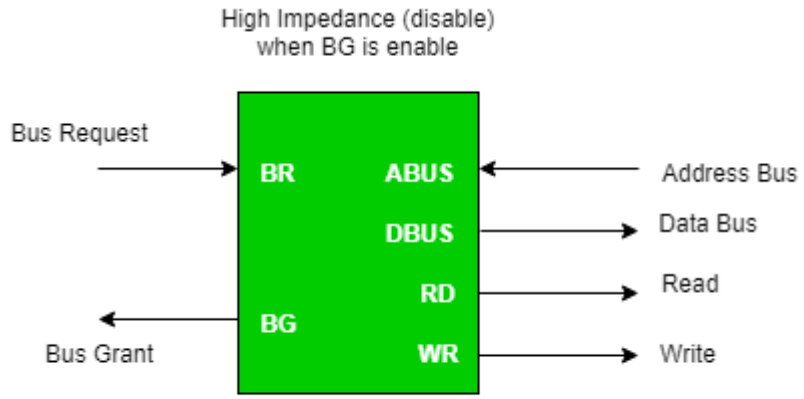
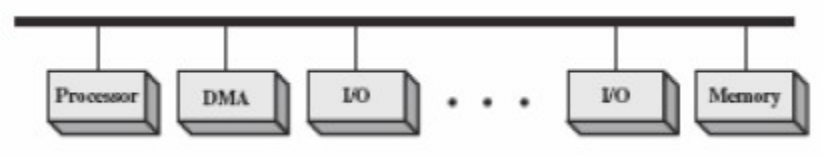


Figure - CPU Bus Signals for DMA Transfer

DMA Configurations

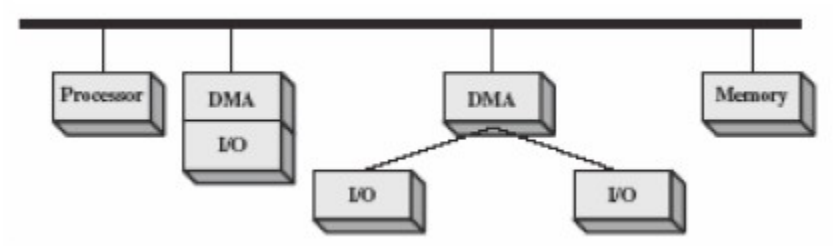
- Single bus – detached DMA module
  - Each transfer uses bus twice – I/O to DMA, DMA to memory
  - Processor suspended twice



- Single bus – integrated DMA module
- Module may support more than one device

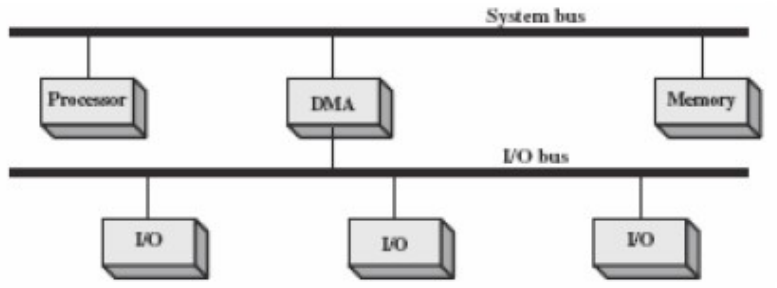
- Each transfer uses bus once – DMA to memory

- Processor suspended once



- Separate I/O bus

- Bus supports all DMA enabled devices
- Each transfer uses bus once – DMA to memory
- Processor suspended once



4.I/O Channels and Processors

The Evolution of the I/O Function

1. Processor directly controls peripheral device
2. Addition of a controller or I/O module – programmed I/O
3. Same as 2 – interrupts added
4. I/O module direct access to memory using DMA
5. I/O module enhanced to become processor like – I/O channel
6. I/O module has local memory of its own – computer like – I/O processor

• More and more the I/O function is performed without processor involvement. The processor is increasingly relieved of I/O related tasks – improved performance.

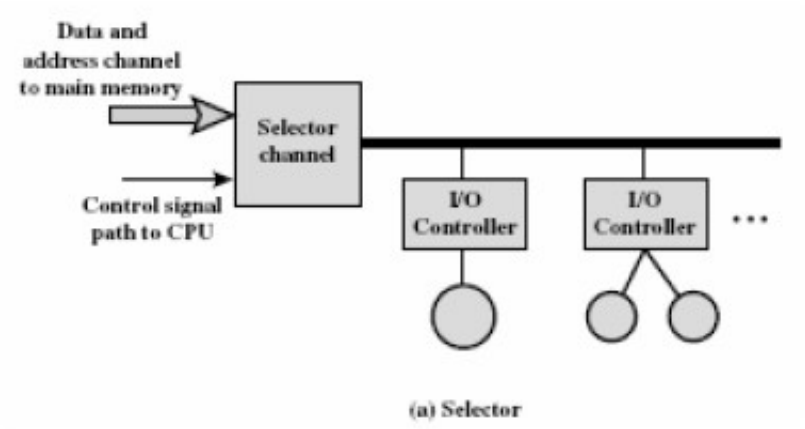
Characteristics of I/O Channels

- Extension of the DMA concept
- Ability to execute I/O instructions – special-purpose processor on I/O channel – complete control over I/O operations
- Processor does not execute I/O instructions itself – processor initiates I/O transfer by instructing the I/O channel to execute a program in memory
- Error condition actions

Two type of I/O channels

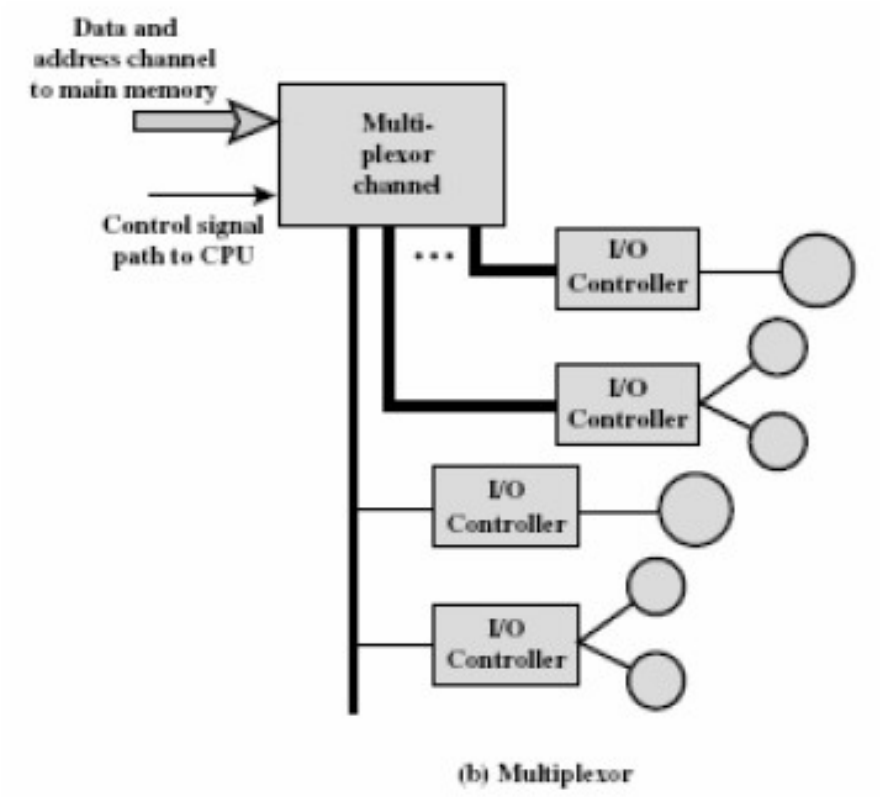
Selector channel

- Controls multiple high-speed devices
- Dedicated to the transfer of data with one of the devices
- Each device handled by a controller, or I/O module
- I/O channel controls these I/O controllers



Multiplexor channel

- Can handle multiple devices at the same time
- Byte multiplexor – used for low-speed devices
- Block multiplexor – interleaves blocks of data from several devices.



Synchronous & Asynchronous Transmission

In **synchronous** data transfer: sender and receiver use same clock signal.

- Supports high data transfer rate
- Need clock signal between the sender and receiver.
- Requires master/ slave configuration

There are many serial data transfer protocol. The protocol serial data transfer can be grouped into two types: synchronous and asynchronous . For synchronous data transfer, both the sender and receiver access the data according to the same clock. Therefore , a special line for the clock signal is required. A master (or one of the senders) should provide the clock signal to all the receivers in the synchronous data transfer

Examples of Synchronous Transmission

- Chatrooms
- Video conferencing
- Telephonic conversations
- Face-to-face interactions

Asynchronous Transmission

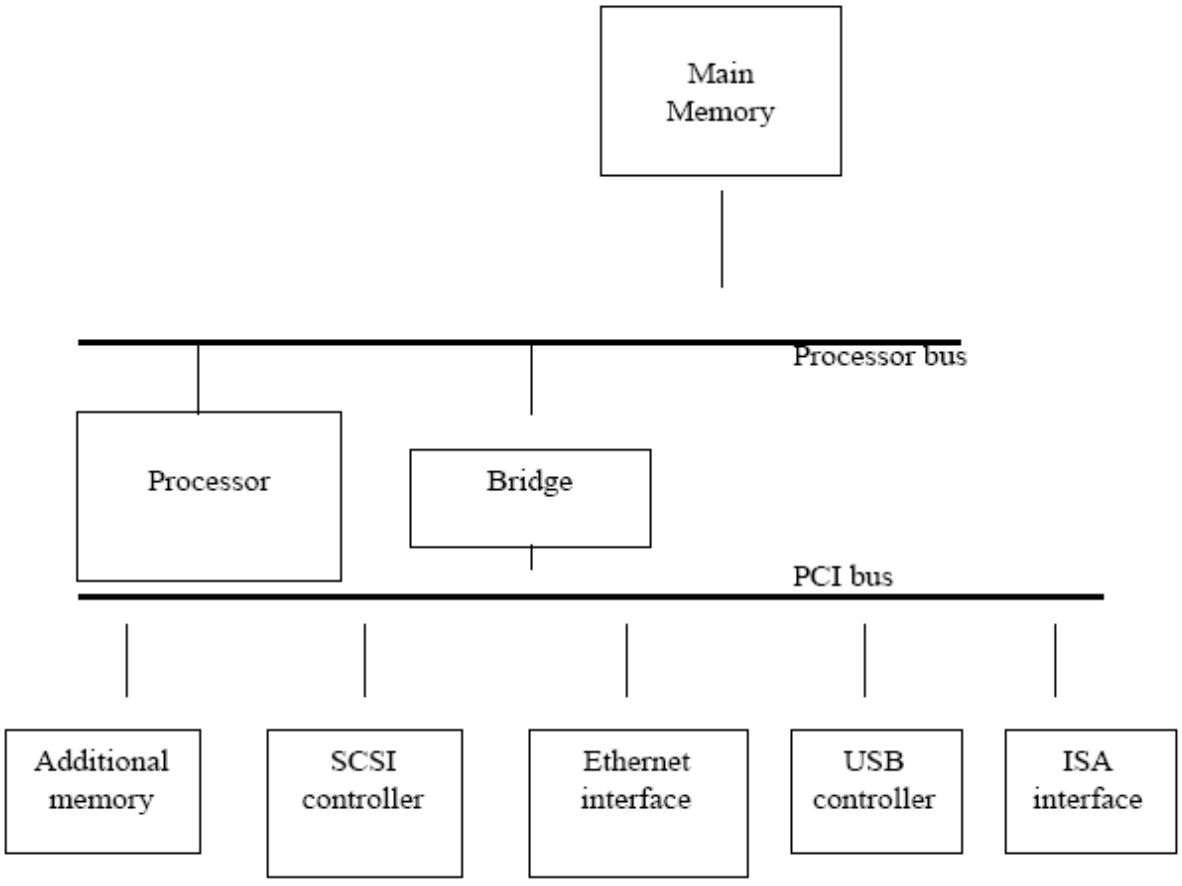
In asynchronous data transfer, there is no common clock signal between the sender and receiver. Therefore, the sender and the receiver first need to agree on a data transfer speed. This speed usually does not change after the data transfer starts. Both the sender and receiver set up their own internal circuit to make sure that the data accessing is follow that agreement. However just like some watches run faster than others, computer clocks also differ in accuracy y. Although the difference is very small, it can accumulate fast and eventually cause errors in data transfer. This problem is solved by adding synchronous bits at the front, middle or end of the data. Since the synchronization is done periodically , the receiver can correct the clock accumulation error. The synchronization information may be added to every byte of data or every frame of data. Sending these extra synchronization bits may account for up to 50 % data transfer overhead and hence slows down the actual data transfer. It sends the data in a constant current of bytes. The size of a character transmitted is 8 bits, with a parity bit added both

at the beginning and at the end, making it a total of 10 bits. It doesn't need a clock for integration—rather, it utilizes the parity bits to tell the receiver how to translate the data.

Examples of Asynchronous Transmission

- Emails
- Forums
- Letters, radio, television

Standard communication interface



PCI(Peripheral Component Interconnect bus)(1992)
The PCI bus is a good example of a system bus that grew out of the need for standardization. It supports the functions found on a processor bus bit in a standardized format that is independent of any particular processor. Devices connected to the PCI bus appear to the processor as if they were connected

directly to the processor bus. They are assigned addresses in the memory address space of the processor.

SCSI bus (Small Computer system interface)
The acronym SCSI stands for Small Computer System Interface. It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X3.131 . In the original specifications of the standard, devices such as disks are connected to a computer via a 50-wire cable, which can be up to 25 meters in length and can transfer data at rates up to 5 megabytes/s.
The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years. SCSI-2 and SCSI-3 have been defined, and each has several options. A SCSI bus may have eight data lines, in which case it is called a narrow bus and transfers data one byte at a time.
Alternatively, a wide SCSI bus has 16 data lines and transfers data 16 bits at a time. There are also several options for the electrical signaling scheme used.