**Synchronous**

Synchronous serial communication describes a serial communication protocol in which "data is sent in a continuous stream at a constant rate."[1]

Synchronous communication requires that the clocks in the transmitting and receiving devices are synchronized – running at the same rate – so the receiver can sample the signal at the same time intervals used by the transmitter. No start or stop bits are required. For this reason "synchronous communication permits more information to be passed over a circuit per unit time"[2] than asynchronous serial communication. Over time the transmitting and receiving clocks will tend to drift apart, requiring resynchronization.

**Asyncronous**

Asynchronous serial communication describes a serial communication protocol in which a start signal is sent prior to each byte, character or code word and a stop signal is sent after each code word, so as to make the communication asynchronous. The start signal serves to prepare the receiving mechanism for the reception and registration of a symbol and the stop signal serves to bring the receiving mechanism to rest in preparation for the reception of the next symbol.

**serial port**

In computing, a serial port is a serial communication physical interface through which information transfers in or out one bit at a time (in contrast to a parallel port).[1] Throughout most of the history of personal computers, data was transferred through serial ports to devices such as modems, terminals and various peripherals.

**Paralell comunication**

In telecommunication and computer science, parallel communication is a method of conveying multiple binary digits (bits) simultaneously. It contrasts with serial communication, which conveys only a single bit at a time; this distinction is one way of characterizing a communications link.

The basic difference between a parallel and a serial communication channel is the number of electrical conductors used at the physical layer to convey bits. Parallel communication implies more than one such conductor. For example, an 8-bit parallel channel will convey eight bits (or a byte) simultaneously, whereas a serial channel would convey those same bits sequentially, one at a time. If both channels operated at the same clock speed, the parallel channel would be eight times faster. A parallel channel may have additional conductors for other signals, such as a clock signal to pace the flow of data, a signal to control the direction of data flow, and handshaking signals.

**polling**

Polling, or polled operation, in computer science, refers to actively sampling the status of an external device by a client program as a synchronous activity. Polling is most often used in terms of input/output (I/O), and is also referred to as polled I/O or software-driven I/O.

**maskeble non-maskable**

the interrupt which can be ignored by the processor ,while performing its operations are called maskable interrupts. generally maskable interrupts are the interrupts that comes from the pheripheral devices.

where as the non maskable interrupts are the interrupts which cannot be ignored. generally these type of interrupts are specified to be software interrupts.

**mapped-io and prot- io**

Memory-mapped I/O (MMIO) and port-mapped I/O (PMIO) (which is also called isolated I/O) are two complementary methods of performing input/output between the CPU and peripheral devices in a computer. An alternative approach is using dedicated I/O processors—commonly known as channels on mainframe computers—that execute their own instructions.

Memory-mapped I/O (not to be confused with memory-mapped file I/O) uses the same address bus to address both memory and I/O devices – the memory and registers of the I/O devices are mapped to (associated with) address values. So when an address is accessed by the CPU, it may refer to a portion of physical RAM, but it can also refer to memory of the I/O device. Thus, the CPU instructions used to access the memory can also be used for accessing devices. Each I/O device monitors the CPU's address bus and responds to any CPU access of an address assigned to that device, connecting the data bus to the desired device's hardware register. To accommodate the I/O devices, areas of the addresses used by the CPU must be reserved for I/O and must not be available for normal physical memory. The reservation might be temporary—the Commodore 64 could bank switch between its I/O devices and regular memory—or permanent.

Port-mapped I/O often uses a special class of CPU instructions designed specifically for performing I/O, such as the IN and OUT instructions found on microprocessors based on the x86 and x86-64 architectures. These two instructions can copy one, two or four bytes (outb, outw, outl) between the EAX register or one of that register's subdivisions on the CPU and a specified I/O port which is assigned to an I/O device. I/O devices have a separate address space from general memory, either accomplished by an extra "I/O" pin on the CPU's physical interface, or an entire bus dedicated to I/O. Because the address space for I/O is isolated from that for main memory, this is sometimes referred to as isolated I/O.

Direct memory access (DMA) is a feature of computerized systems that allows certain hardware subsystems to access main system memory independently of the central processing unit (CPU).

**DMA**

Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done. This feature is useful any time the CPU cannot keep up with the rate of data transfer, or where the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer. Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards and sound cards. DMA is also used for intra-chip data transfer in multi-core processors. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without DMA channels. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel.