# UNIT – 4

Syllabus: - I/O Organization: I/O Interface – PCI Bus, SCSI Bus, USB, Data Transfer: Serial, Parallel, Synchronous, Asynchronous Modes of Data Transfer, Direct Memory Access (DMA), I/O Processor.

**Input Output Organization:**

The input-output subsystems of computer referred to as I/O provides an efficient mode of communication between the central system and the outside world. The most familiar means of entering information into a computer is through a typewriter-like keyboard that allows a person to enter alphanumeric information directly.

Input and output devices attached to the computer are also called peripherals. Among the common peripherals are keyboards, display units, printers. Peripherals that provide auxiliary storage for the system are magnetic disks and tapes. Peripherals are electromechanical and electromagnetic devices of some complexity.

**ASCII Alphanumeric Characters:**

Input and output devices that communicate with people and the computer are usually involved in the transfer of alphanumeric information to and from the devices and the computer. ASCII (American Standard Code for Information Interchange). It uses 7 bits to code 128 characters as shown in table 1. The seven bits of the code are designated by b1 through b7, with b7 being the most significant bit. The letter A, for example is represented in ASCII as 1000001(column 100, row 001). The ASCII code contains 94 characters that can be printed and 34 nonprinting characters used for various control functions. The printing characters consists of 26 uppercase letters A through Z, 26 lowercase letters, 10 numerals 0 through 9 and 32 special printable characters such as %,* and $.



**Table 1: American Standard Code for Information Interchange (ASCII)**

| $b_4b_3b_2b_1$ | $b_7b_6b_5$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **000** | **001** | **010** | **011** | **100** | **101** | **110** | **111** |
| 0000 | NUL | DLE | SP | 0 | @ | P | . | P |
| 0001 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0010 | STX | DC2 | “ | 2 | B | R | b | r |
| 0011 | ETX | DC3 | # | 3 | C | S | c | s |
| 0100 | EOT | DC4 | $ | 4 | D | T | d | t |
| 0101 | ENQ | NAK | % | 5 | E | U | e | U |
| 0110 | ACK | SYN | & | 6 | F | V | f | v |
| 0111 | BEL | ETB | ‘ | 7 | G | W | g | W |
| 1000 | BS | CAN | ( | 8 | H | X | h | x |
| 1001 | HT | EM | ) | 9 | I | Y | i | y |
| 1010 | LF | SUB | * | : | J | Z | j | z |
| 1011 | VT | ESC | + | ; | K | [ | k | { |
| 1100 | FF | FS | , | < | L | \ | l | \| |
| 1101 | CR | GS | - | = | M | ] | m | } |
| 1110 | SO | RS | . | > | N | ^ | n | ~ |
| 1111 | SI | US | / | ? | O | - | o | DEL |

**Input Output Interface:**

I/O interface provides a method for transferring information between internal storage and external storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the CPU. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

The major differences are:
Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore a conversion of signal values may be required.
The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
Data codes and formats in peripherals differ from the word format in the CPU and memory.
The operation modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.
To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.

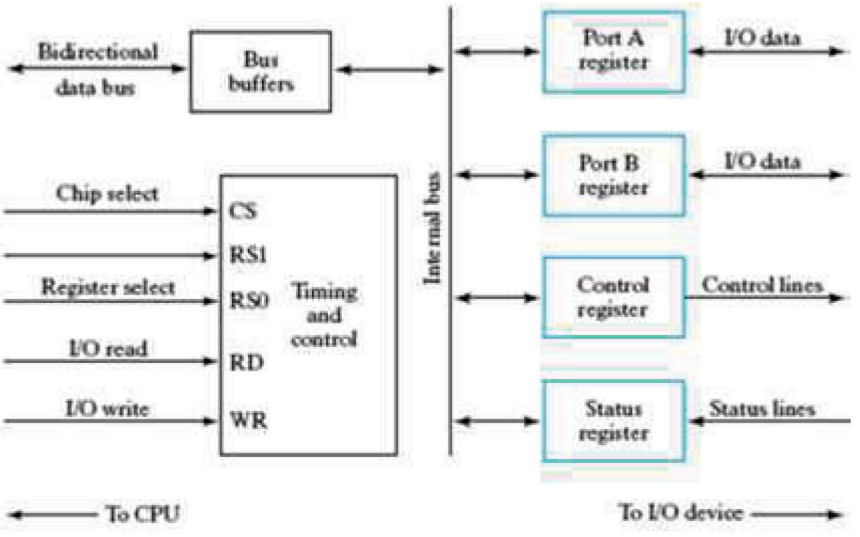
**Example of I/O Interface:**

Figure 1 shows the block diagram of I/O interface unit. It consists of 2 data registers called ports, a control register, a status register, bus buffers and timing and control circuits.

- The interface communicates with the CPU through data bus.
- The chip select (CS) and register select (RS) inputs determine the address assigned to the interface.
- The I/O read and write are two control lines that specify an input or output respectively.
- The four registers communicate directly with the I/O device attached to the interface.
- The I/O data to and from the device can be transferred into either port A or port B. T

The interface may operate with an output device or with an input device or with a device that requires both input and output.

In this example, address bus selects the interface unit through CS and RS1 & RS0 and a particular interface is selected by the circuit (decoder) enabling CS, RS1 and RS0 select one of 4 registers.
The interface registers communicate with the CPU through the bidirectional data bus. The address bus selects the interface unit through the chip select the two register select inputs. A circuit must be provided externally to detect the address assigned to the interface registers. This circuit enables the chip select input when the interface is selected by the address bus. The two register select inputs RS1 and RS0 are usually connected to the two least significant lines of the address bus. These two inputs select one of the four registers in the interface as specified in the table. The content of the selected register is transfer into the CPU via the data bus when the I/O read signal is enabled. The CPU transfers binary information into the selected register via the data bus when the I/O write input is enabled.




| CS | RS1 | RS0 | Register selected |
|---|---|---|---|
| 0 | X | X | None: data bus in high-impedance state |
| 1 | 0 | 0 | Port A register |
| 1 | 0 | 1 | Port B register |
| 1 | 1 | 0 | Control register |
| 1 | 1 | 1 | Status register |

**Figure 1: Example of I/O Interface Unit**

**PCI Bus -** A Peripheral Component Interconnect Bus (PCI bus) connects the CPU and expansion boards such as modem cards, network cards and sound cards. These expansion boards are normally plugged into expansion slots on the motherboard.

**Features:**

It is designed to economically meet the I/O requirements of modern systems. It requires very few chips to implement and support other buses attached to the PCI bus.

It bypasses the standard I/O bus, uses the system bus to increase the bus clock speed and take full advantage of the CPU's data path.

It has an ability to function with a 64 bit data bus.

- It has high bandwidth. The information is transferred across the PCI bus at 33MHz, at the full data width of the CPU.

**PCI Configuration**

Figure 2(a) shows a typical use of PCI in a single processor system. Notice that the processor bus is separate and independent of the PCI bus. The processor connects to the PCI bus through an integrated circuit called a PCI bridge. The memory controller and PCI Bridge provides tight coupling with the processor and delivers data at high speeds. Figure 2(b) shows a typical use of PCI in a multiprocessor system. As shown in the Figure 2(b) in multiprocessor systems one or more PCI configurations may be connected by bridges to the processor's system bus. Again, the use of bridges keeps the PCI independent of the processor speed yet provides the ability to receive and deliver data rapidly.

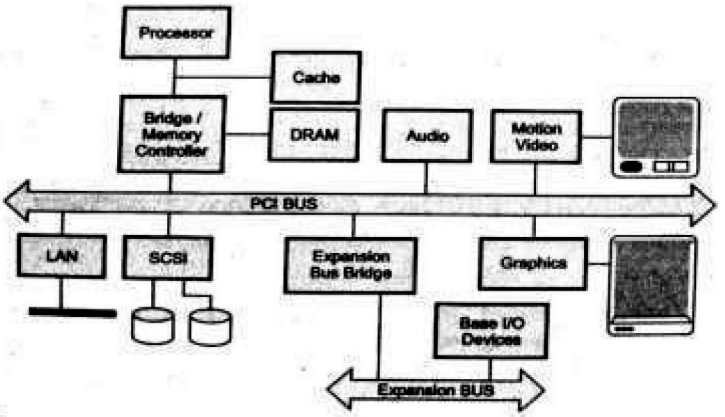



**Figure 2(a): Conceptual design of PCI bus for single processor system**

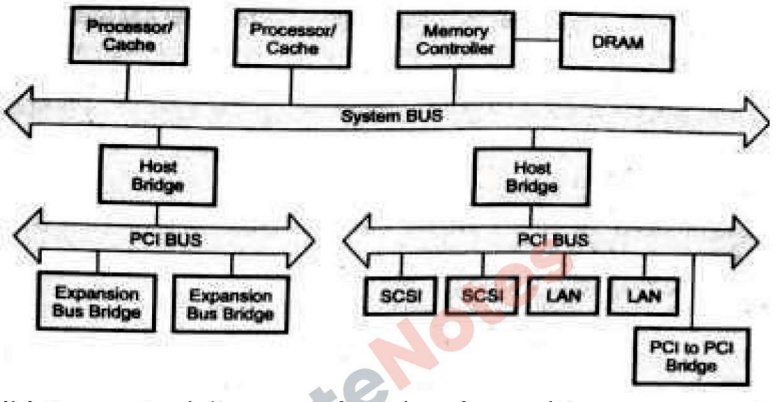



**Figure 2 (b) Conceptual diagram of PCI bus for multiprocessor system**

**Data Transfer**

Every data transfer on the PCI bus is a single transaction consisting of one address phase and one or more data phases. All the events during read cycle are synchronized with the falling edge of the clock cycle.

**SCSI Bus**

The acronym SCSI stands for small computer system interface. It was adopted as a standard by the American National Standard Institute (ANSI) in 1986. This bus connects I/O devices such as hard disk units and printers to personal computers. SCSI was originally designed to transfer data a byte at a time at rates up to 5 MB/s. The Figure 3 shows the SCSI I/O bus.

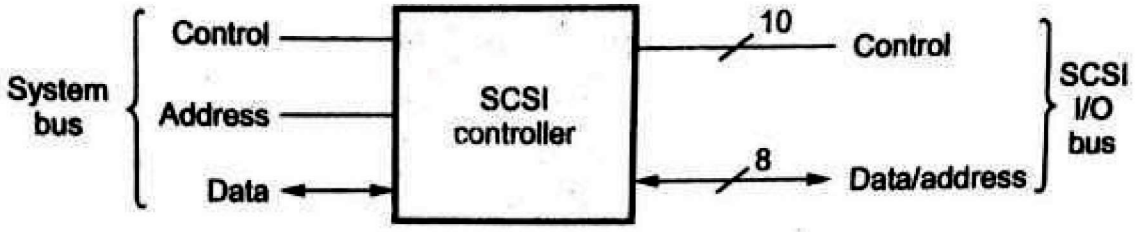



**Figure 3: SCSI I/O bus**

SCSI is smaller and simpler bus and its data sub bus is only 8-bits wide. Its data bus is also used to transfer addresses. Ten additional lines provide all the necessary control functions. Recent extensions to the original SCSI standard have wider data buses (16 and 32 bits), more control features and higher data transfer rates.

**SCSI Standard**

SCSI-1 defines the basics of the first SCSI buses, including cable length signaling characteristics, commands and transfer modes.

- Devices corresponding to the SCSI-1 standard use only a narrow (8- bit) bus, With a 5 MB/s maximum transfer rate.
- Only single-ended transmission was supported, with passive termination.
- There were also difficulties associated with the standard gaining universal acceptance due to the fact that many manufacturers implemented different subsets of its features.
- Devices that adhere to the SCSI-l standard in most cases can be used with host adapters and other devices that use the higher transfer rates of the more advanced SCSI-2 protocols, but they will still function at their original slow speed.

**SCSI-2** is an extensive enhancement over SCSI-1. In addition, the standard defines the following significant new features as additions to the original SCSI-1 specification:
1.**Fast SCSI:** This higher-speed transfer protocol doubles the speed of the bus to 10 MHz
2.**Wide SCSI:** The width of the original SCSI bus was increased to 16 (or even 32) bits. This permits more data throughput at a given signaling speed.
3.**More Devices per Bus**: On buses that are running with Wide SCSI, 16 devices are supported.
**SCSI-3 -** SCSI-3 specification defines the mechanical, electrical and protocol layers of the interface. Data transfers of 8 bits at 20Mbps over a 50 pin connector and 16 bits at 40Mbps over a 68 pin connector. The number of devices on the bus increased to 16 (for Fast-10), Fast-20 allows 8 devices maximum with a number of other combinations

**Universal Serial Bus (USB)**
USB gives fast and flexible interface for connecting all kinds of peripherals. USB is playing a key role in fast growing consumer areas like digital imaging, PC telephony, and multimedia; games, etc. The presence of USB in most new PCs and its plug-n- play capability, means that PCs and peripherals (such as CD ROM drives, tape and floppy drives, scanners, printers, video devices, digital cameras, digital speakers, telephones, modems, key boards, mice, digital joysticks and others) will automatically configure and work together, With high degree of reliability, in this exciting new application areas.

**USB Features**

- **Simple cables**
- **One interface for many devices**
- **Automatic configuration**
- **No user setting**

**Data Transfer: Serial, Parallel**

The transfer of data between two units may be done in two ways: parallel or serial. In **serial data transmission**, each bit in the message is sent in sequence one at a time. This method requires the use of one pair of conductors or one conductor and a common ground. Serial transmission is slower but is less expensive since it requires only one pair of conductors.
In **parallel data transmission**, each bit of the message has its own path and the total message is transmitted at the same time. This means that an n bit message must be transmitted through n separate conductor paths. Parallel transmission is faster but requires many wires. It is used for short distances and where speed is important.
Serial transmission can be synchronous or asynchronous.
In **synchronous transmission**, the two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses. In long distant serial transmission, each unit is driven by a separate clock of the same frequency. Synchronization signals are transmitted periodically between the two units to keep their clocks in step with each other.

In **asynchronous transmission**, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted. This is in contrast to synchronous transmission, where bits must be transmitted continuously to keep the clock frequency in both units synchronized with each other .

**Asynchronous serial transmission:**

With this technique, each character consists of three parts: a start bit, the character bits, and stop bits. The convention is that the transmitter rests at the state when no characters are transmitted. The first bit, called the start bit, is always a 0 and is used to indicate the beginning of a character. The last bit called the stop bit is always a 1. A transmitted character can be detected by the receiver from knowledge of the transmission rules:

1. When a character is not being sent, the line is kept in the 1 state.
2. The initiation of a character transmission is detected from the start bit, which is always 0.
3. The character bits always follow the start bit.

After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1 state for at least one bit time. Using these rules, the receiver can detect the start bit when the line goes from 1 to 0. A clock in the receiver examines the line at proper bit times. The receiver knows the transfer rate of the bits and the number of character bits to accept. After the character bits are transmitted, one or two stop bits are sent. The stop bits are always in the 1 state and frame the end of the character to signify the idle or wait state.

**Synchronous Modes of Data Transfer**

In a synchronous data transfer, all devices derive timing information from a common clock signal. The Figure 4 shows the timing diagram for synchronous input/output transfer At time $t_o$, the processor places the device address on the address lines of System bus and sets the control lines to perform the input operation. During time $t_0$- $t_1$, addressed device gets, the address and it recognizes that an input operation is requested. At time $t_1$, address device places its data on the data bus. At the end of bus cycle that is at time $t_2$ the processor reads the data lines and loads the data from data bus into its input buffer.

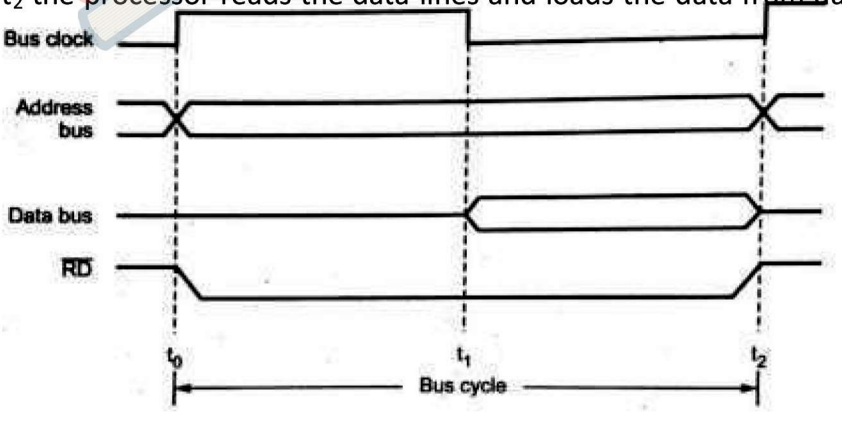



**Figure 4: Timing diagram for synchronous input transfer**

The timing diagrams shown in Figure 5 do not consider the propagation delay so they represent ideal timings for data transfer that takes place on lines. In practice the propagation delay, delays the timings at which signals actually change state. This is illustrated in Figure 6. The Figure 6 shows the signals as seen by i the master and the other as seen by the slave.

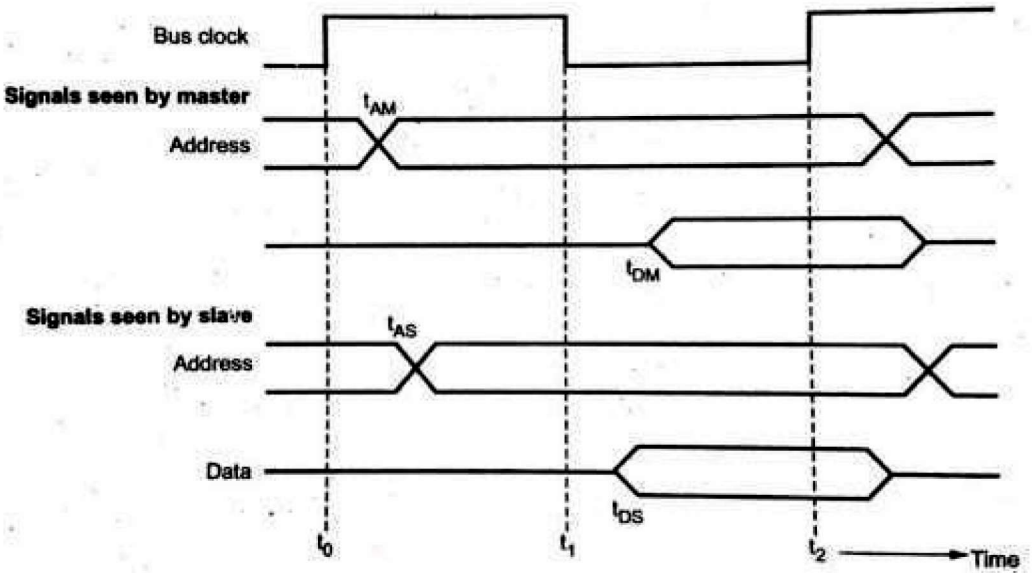



**Figure 5: Timing diagram for synchronous output**

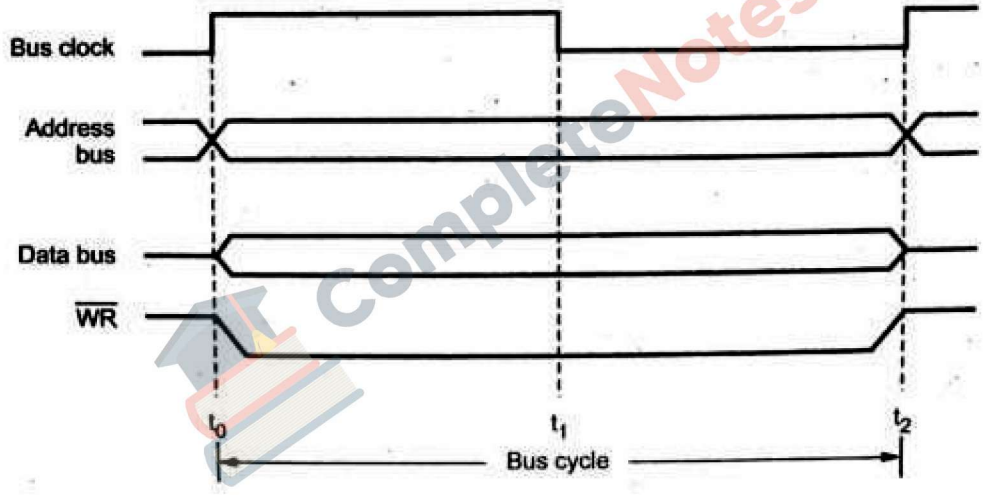



**Figure 6: Timing diagram for synchronous input transfer considering propagation delay;**

The master sends address signals on the rising edge at the beginning of $t_0$ but due to the delay in the bus driver circuit address signals appear on the address bus after time delay of $t_{AM}$. After some more delay, at $t_{As}$, the address signals reach the slave. The slave decodes the address and at $t_1$ sends the requested data. Here again, the data signals do not appear on the data bus until $t_{DS}$. Data signals reach to the master at time $t_{DM}$. At $t_2$, the master reads and loads the data into its input buffer; The time $t_2$ - $t_{DM}$ is known as data setup time for master's input buffer. The data must continue to be valid after $t_2$ for a period equal to the hold time of the master's input buffer.

**Limitations:-** In this mode the transfer has to be completed within one clock cycle, the clock period, $t_z—t_o$, must be selected to accommodate the longest delays on the bus and the slowest device interface. This forces all devices to operate at the speed of the slowest device.

**Serial Communication:**

In parallel communication number of lines required to transfer data depend on the number of bits to be transferred. For example, to transfer a byte of data, 8 lines are required and all 8 bits are transferred

simultaneously. Thus for transmitting data over a long distance, using parallel communication is impractical due to the increase in cost of cabling.

**Table 2 : Difference between Serial and Parallel Data Transfer**

| S. No. | Serial Data Transfer | Parallel Data Transfer |
|---|---|---|
| 1 | It transfer data one bit at a time | It can transmit more than one data bit at a time. |
| 2 | Lower data transfer rate | Faster data transfer rate |
| 3 | Needs less number of wires to connect devices in the system. | It needs more number of wires to connect devices in the system. |
| 4 | Well suited for long distances, because fewer wires are used as compared to a parallel bus. | The inter connection penalty increases as distances increase. Thus not suitable for long distance communication. |

**Asynchronous Serial Communication**

Figure 8 shows the transmission format for asynchronous serial transmission. Asynchronous formats are character oriented. In this, the bits of a character or data word are sent at a constant rate, but characters can come at any rate (asynchronously) as long as they do not overlap. When no characters are being sent, a line stays high at logic 1 called mark, logic 0 is called space. The beginning of a character is indicated by a start bit which is always low. This is used to synchronize the transmitter and receiver. After the start bit, the data bits are sent with least significant bit first, followed by one or more stop bits (active high). The stop bits indicate the end of character.

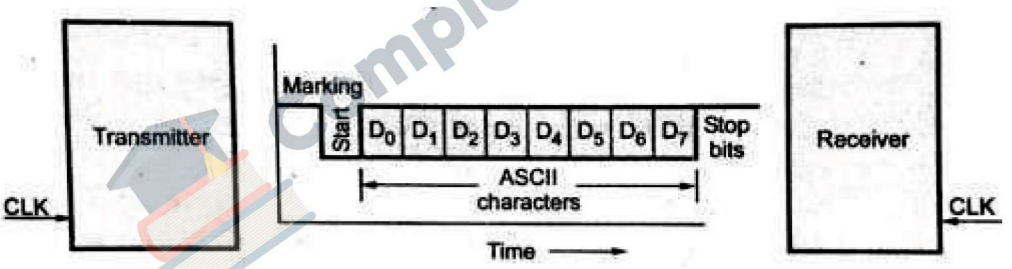



**Figure 7: Transmission format for asynchronous transmission**

The data rate can be expressed as bits/sec. or characters/sec. The term bits/sec is also called the baud rate. The asynchronous format is generally used in low-speed transmission (less than 20 Kbits/ sec).

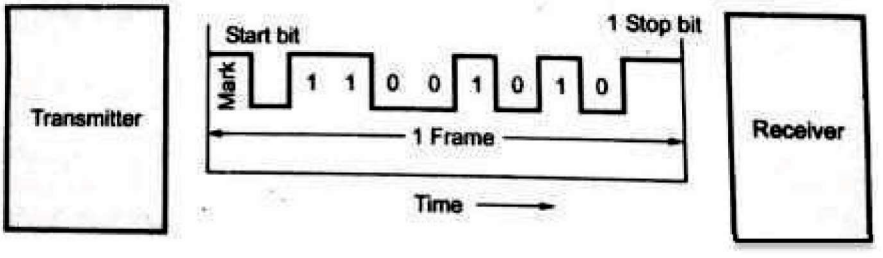


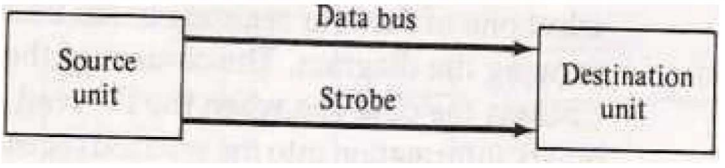

**Figure 8 : Asynchronous com with m byte CAH**

**Asynchronous Modes of Data Transfer**

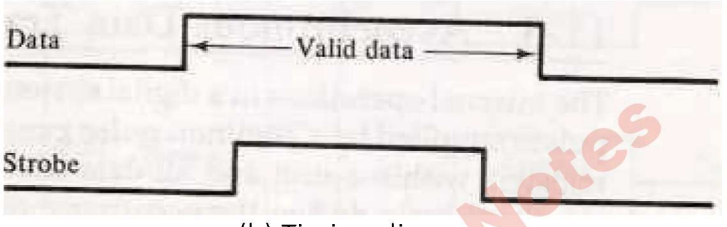
Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. One way of achieving this is by means of a strobe.

Pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.
The unit receiving the data item responds with another control signal to acknowledge receipt of the data.
This type of agreement between two independent units is referred to as handshaking.
The strobe pulse method and the handshaking method of asynchronous data transfer are not restricted to I/O transfers.

**Strobe Control -** The strobe control method of asynchronous data transfer employ a single control line to time each transfer. The strobe may be activated by either the source or the destination unit. Figure (a) shows a source initiated transfer. The, data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word. The strobe is a single line that informs the destination unit when a valid data word is available in the bus.



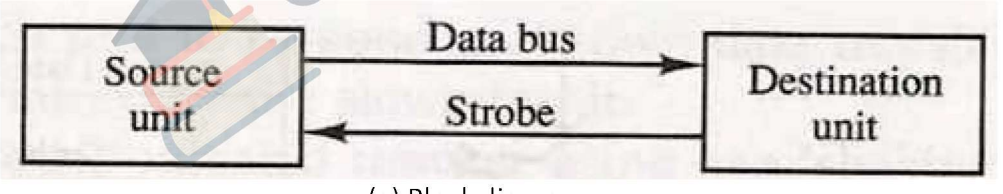

(a) Block diagram




(b) Timing diagram

**Figure 9: Source initiated strobe for data transfer.**

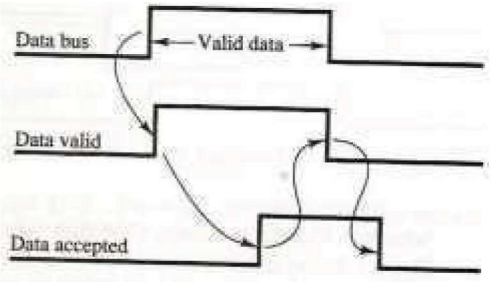
Figure 10 shows a data transfer initiated by the destination unit. In this case the destination unit activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain in the bus long enough for the destination unit to accept it.




(a) Block diagram

**Figure 10: Destination initiated strobe for data transfer**

**Handshaking -** The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus. The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer. One control line is in the same direction as the data flow in the bus from the source to the destination. It is used by the source unit 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. It is used by the destination unit to inform the source whether it can accept data. The sequence of control during the transfer depends on the unit that initiates the transfer.



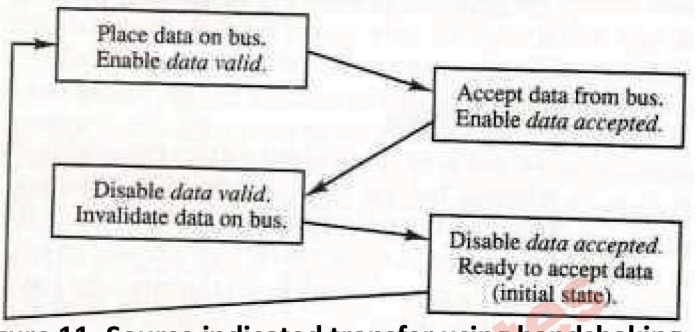




**Figure 11: Source indicated transfer using handshaking.**

**I/O Interfacing Techniques**

I/O devices can be interfaced to a computer system I/O in two Ways, which are filled interfacing techniques, 1. Memory mapped I/O 2. I/O mapped I/O

Memory mapped I/O – In this technique, the total memory address space is partitioned and part of this space is devoted to I/O addressing as shown in Fig. 19. When this technique is used, a memory reference instruction that causes data to be fetched from or stored at address specified, automatically becomes an I/O instruction if that address is made the address of an I/O port.

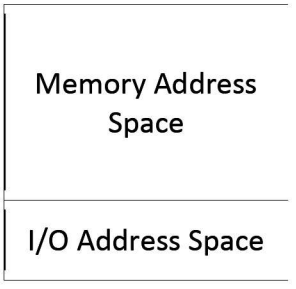



**Figure 12: Address space**

**Advantage -** The usual memory related instructions are used for I/O related operations. The special I/O instructions are not required.

**Disadvantage -** The memory address space is reduced.

**I/O mapped I/O -** If we do not want to reduce the memory address space, we allot a different I/O address space, apart from total memory space which is called I/O mapped I/O technique as shown in Figure 17.

**Advantage -** The advantage is that the full memory address space is available.

**Disadvantage -** The memory related instructions do not work, Therefore, processor can only use this mode if it has special instructions for No related operations such as I/O read, I/O write.

**Table 3: Memory Mapped I/O, I/O Mapped I/O Comparison**

| S. No. | Memory Mapped I/O | I/O Mapped I/O |
|---|---|---|
| 1 | Memory and I/O share the entire address range of processor | Processor provides separate address range for memory and I/O devices. |
| 2 | Usually, processor provides more address lines for accessing memory. Therefore more decoding is required control signals. | Usually, processor provides less address I/O. Therefore less decoding is required. |
| 3 | Memory control signals are used to control read and write I/O operations. | I/O control signals are used to control read and write I/O operations. |

**Direct Memory Access**

In software control data transfer, processor executes a series of instructions to carry out data transfer. For each instruction execution fetch, decode and execute phases are required. Figure 21 gives the flowchart to transfer data from memory to I/O device. Thus to carry out these tasks processor requires considerable time. So this method of data transfer is not suitable for large data transfers such as data transfer from magnetic disk or optical disk to memory.

Drawbacks in programmed I/O and interrupt driven I/O

- The I/O transfer rate is limited by the speed with which the CPU can test and service a device.
- The time that the CPU spends testing I/O device status and executing a number of instructions for I/O data transfers can often be better spent on other processing tasks.
- To overcome above drawbacks an alternative technique, hardware controlled data transfer can be used.



**DMA Block Diagram**

For performing the DMA operation, the basic blocks required in a DMA channel controller are shown in Figure 13. DMA controller communicates with the CPU via the data bus and control lines. The registers in DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (Register select) inputs. The RD (Read) and WR (write) inputs are bidirectional.

When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write the DMA registers RD and WR signals are input signals for DMA. When BG=1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR signals (RD and WR are now output signals for DMA). DMA consists of data count, data register, address register and control-logic.

Data counter register stores the number which gives the number data transfer to be done in one DMA cycle. It is automatically decremented after each word transfer. Data register acts as buffer whereas address register initially holds the starting address of the device. Actually, it stores the address of the next word to be transferred. It is automatically incremented or decremented after each word transfer. After each transfer, data counter is tested for zero. When the data count reaches zero, the DMA transfer halts. The DMA controller is normally provided with an interrupts capability in Which case it sends an interrupt to processor to signal the end of the I/O data transfer.

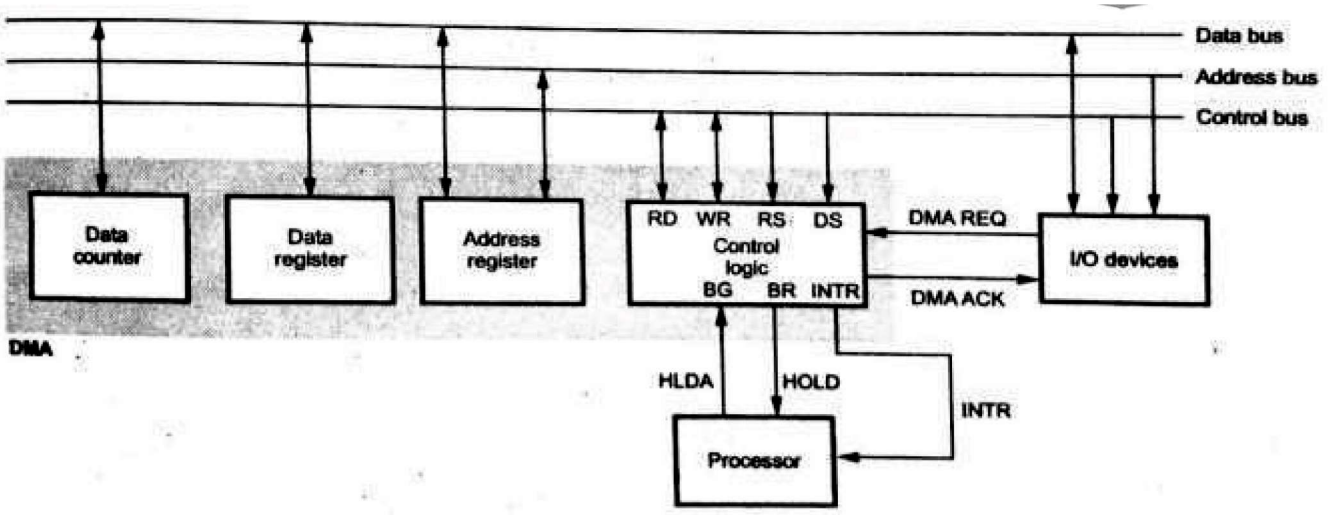



**Figure 13: Typical DMA block Diagram**

**Data Transfer Modes**

DMA controller transfers data in one of the following three modes:
1. Single transfer mode (cycle stealing)
2. Block transfer mode
3. Demand or burst transfer mode

**Single transfer mode (Cycle stealing mode) -** In this mode device can make only one transfer (byte or word). After each transfer DMAC gives the control of all buses to the processor. Due to this processor can have access to the buses on a regular basis. It allows the DMAC to time share the, buses with the processor, hence this mode is most commonly used.

**Block transfer mode –** In this mode device can make number of transfers as programmed in the word count register. After each transfer word count is decremented by 1 and the address is decremented or incremented by 1. The DMA transfer is continued until the word count "role over" from zero to FFFFH, a Terminal Count (TC) or an external END of Process (EOP) is encountered. Block transfer mode is used when the DMAC needs to transfer a block of data.

**Demand transfer mode -** In this mode the device is programmed to continue making transfers until a TC or external W is encountered or until DREQ goes inactive.

**I/O Processor:**
The I/O processor (IOP) has an ability to execute I/O instructions and it can have complete control over me /0 operations. The I/0 instructions are stored in main memory. When I/0 transfer is required, the CPU initiates an I/0 transfer by instructing the I/0 channel to execute an I/O program stored in the main memory. The I/0 program specifies the device or devices, the area of memory storage, priority and actions to be taken for certain error conditions.
**Block Diagram of IOP:** Figure 14 shows the block diagram of computer system with an I/O processor. The CPU and I/O processor work independently and communicate with each other using centrally located memory and DMA. The CPU does the processing of needed in the solution-of computational tasks and IOP does the data transfer between various peripheral devices and the memory unit.

CPU and IOP Communication - The communication between CPU and IOP may be different for different processor and IOP configurations. However, in most of cases the memory based control blocks are used to store the information about the task to be performed. The processor uses these blocks to leave information in it for the other processor. The memory control block is linked, i.e. the address of the next memory based control blocks is available in the previous memory based control block.

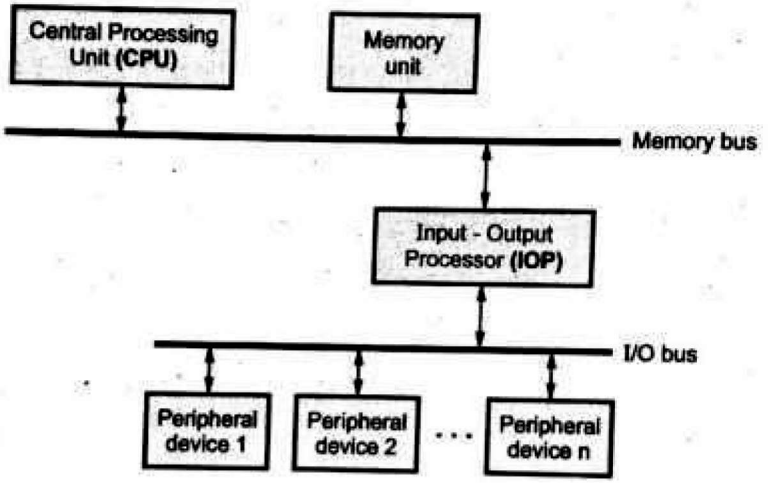



**Figure 14: Block diagram of computer with I/O processor**