Menakshi

COA Unit-5 Notes

Peripheral Devices:-

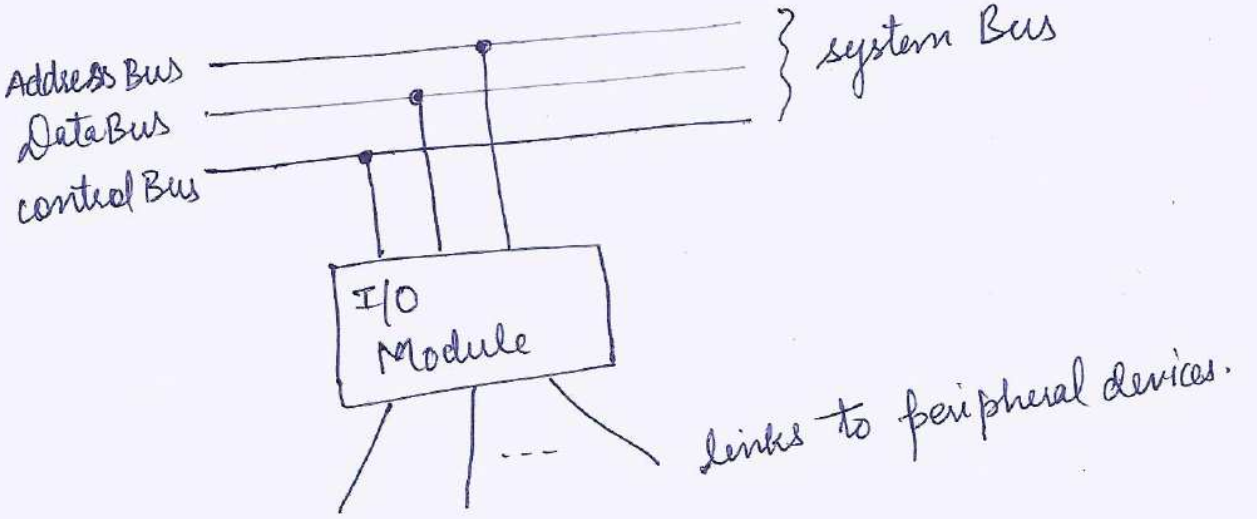
An external device connected to an I/O module is often referred to as a peripheral device or simply a peripheral.

3 categories of external devices:-

(1) Human Readable Device:
suitable for communicating with the computer user (e.g. VDT (Video Display Terminal), Printer, Keyboard, etc).

(2) Machine Readable Devices:-
suitable for communicating with equipment. eg. magnetic disk, sensor etc.

(3) communication Device:
suitable for communicating remote devices.
Eg- The remote device can be the human readable device such as terminal, a machine readable device or another computer.

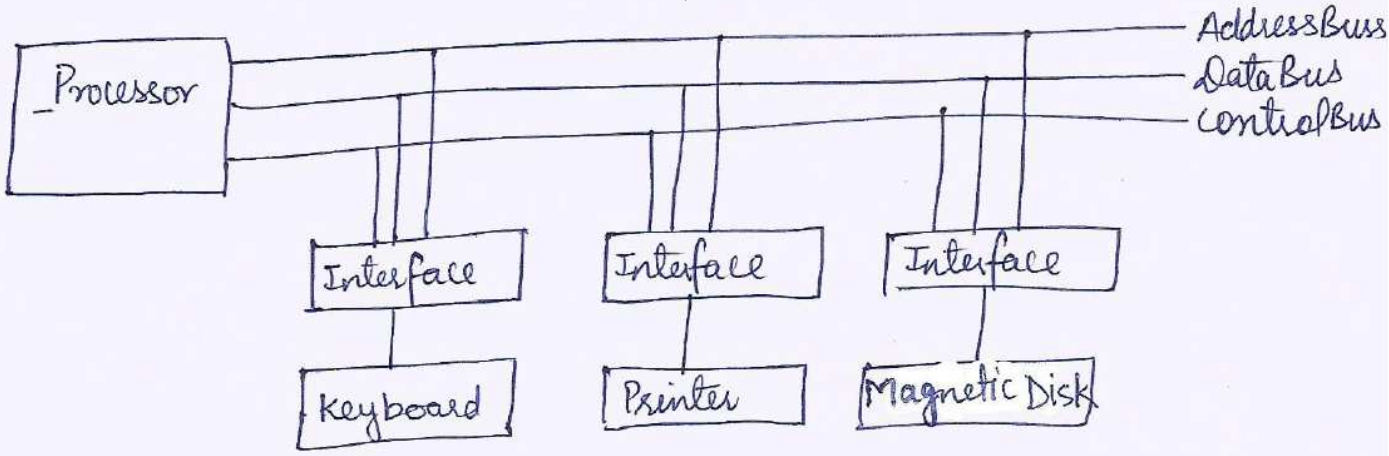


Generic Model of an I/O Module

I/O Interface:

⇒ I/O interface provide a method for transferring information between CPU and external devices.

⇒ The input output interface resolves the differences that exist between the CPU and each external device.



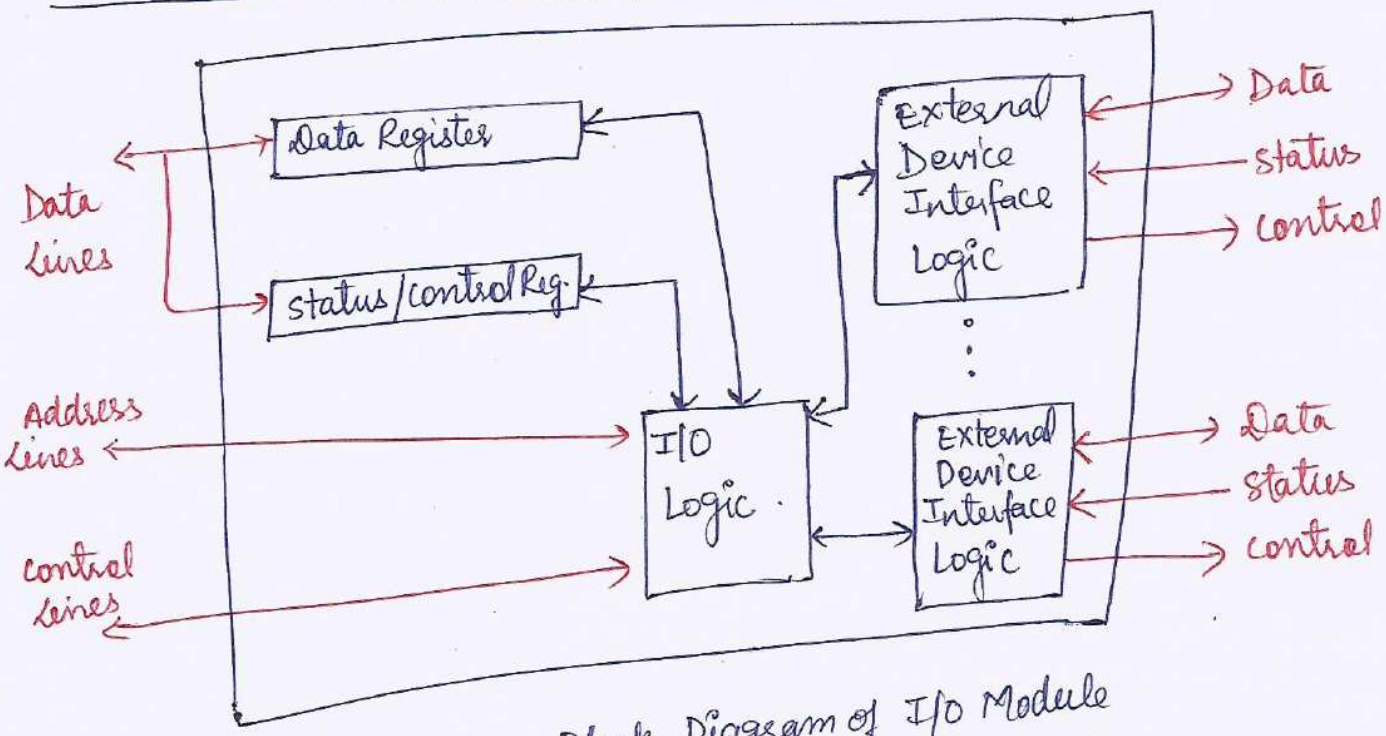
I/O Bus and Interface Module

Functions of I/O Module:

→ Major functions for an I/O Module fall into following categories.

- control and timing
- Processor communication
- Device Communication
- Data Buffering
- Error Detection.

Block Diagram of I/O Module



Block Diagram of I/O Module

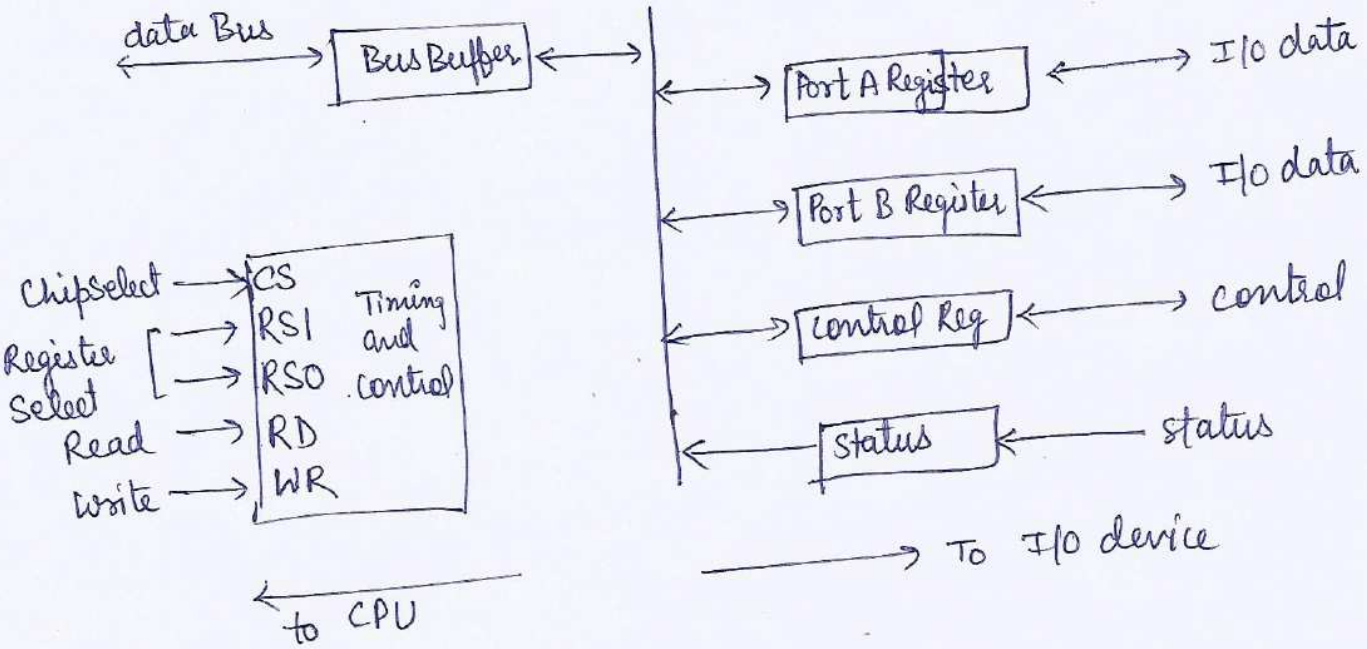
I/O Port: A connection point that acts as an interface b/w the computer and external devices like mouse, keyboard, printer, modem etc.

Ports are of two types:-

Internal Port: It connects motherboard to internal device like Hard disk drive, internal modem etc.

External Port: It connects the motherboard to external device like, mouse, printer, and keyboard etc.

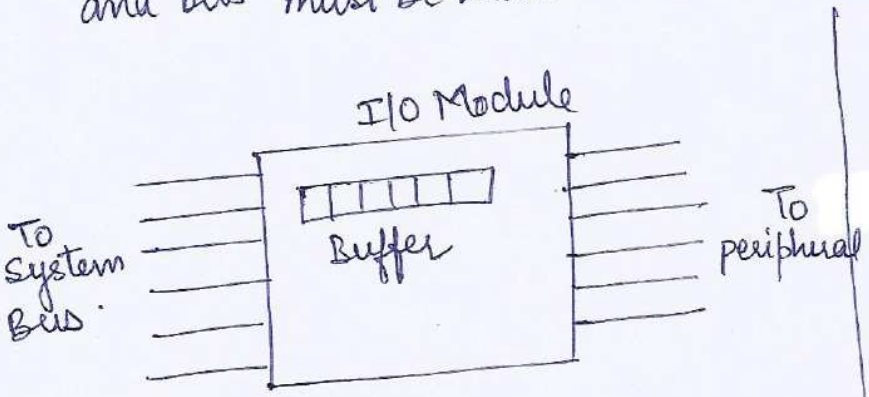
Example of Interface:



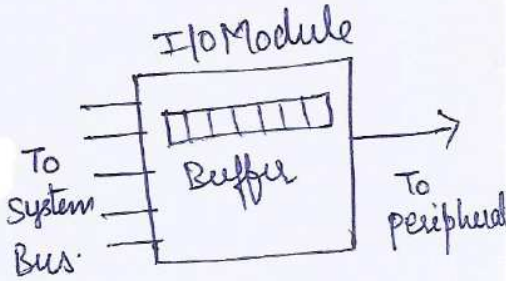
Interfaces
- Serial Interface
- Parallel Interface

* In parallel interface, there are multiple connecting I/O module and the peripheral, and multiple bits are transmitted simultaneously over the data bus. (all of bits of data word transmitted simultaneously)

* In serial interface, there is only one line used to transmit data, and bits must be transmitted one at a time.



(a) parallel I/O.



(b) serial I/O.

I/O Systems:

User programs never directly interact with I/O devices. All I/O operations are performed with the help of system calls.

There are 4 types of I/O commands:-

→ Control
→ Test Status
→ Read
→ Write

Modes of Data Transfer (I/O Techniques)

- Programmed I/O
- Interrupt Driven I/O
- Direct Memory Access

Programmed I/O

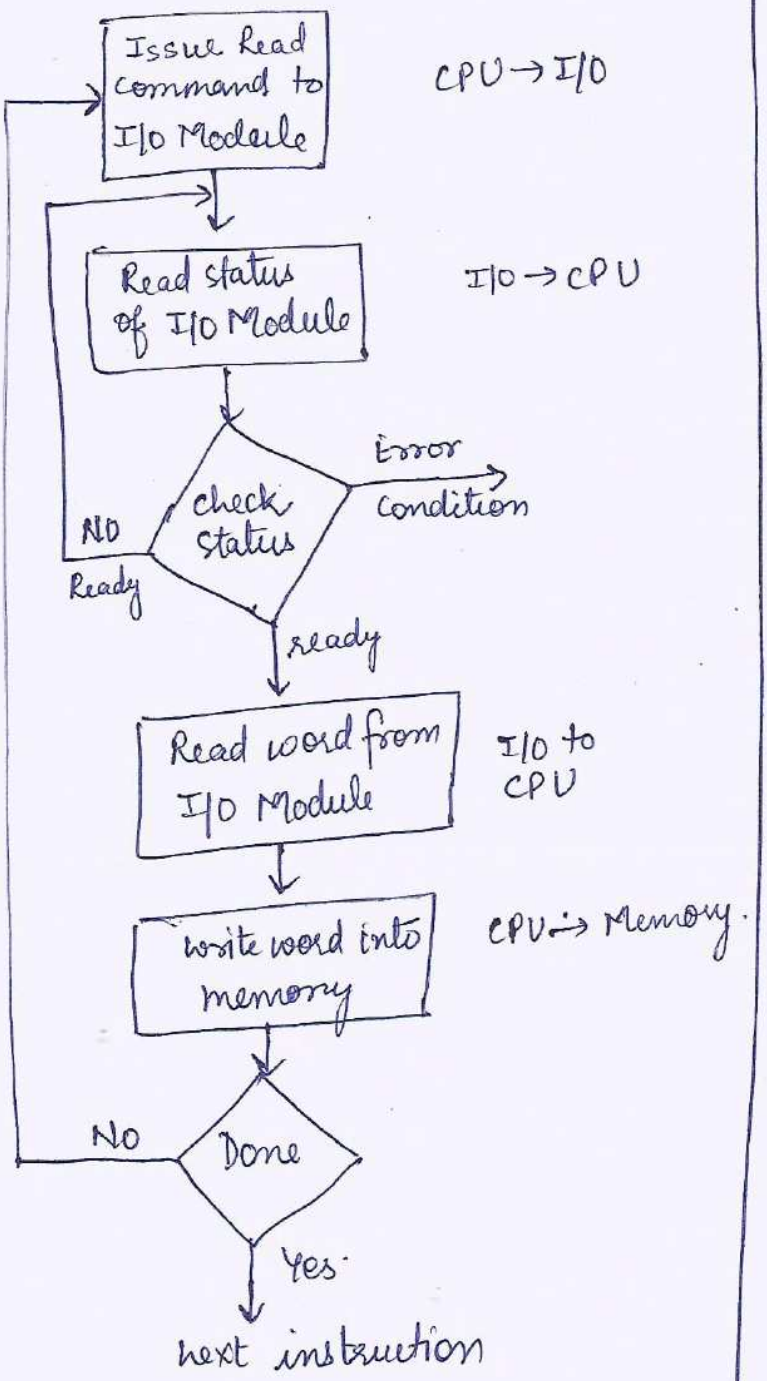
- Input-output instructions written in computer programm.
- Each data item transfer is initiated by an instruction in the program
- Transferring data under program control requires a constant monitoring of peripherals by CPU. (once initiated when to transfer)

⇒ ~~Once initiated~~

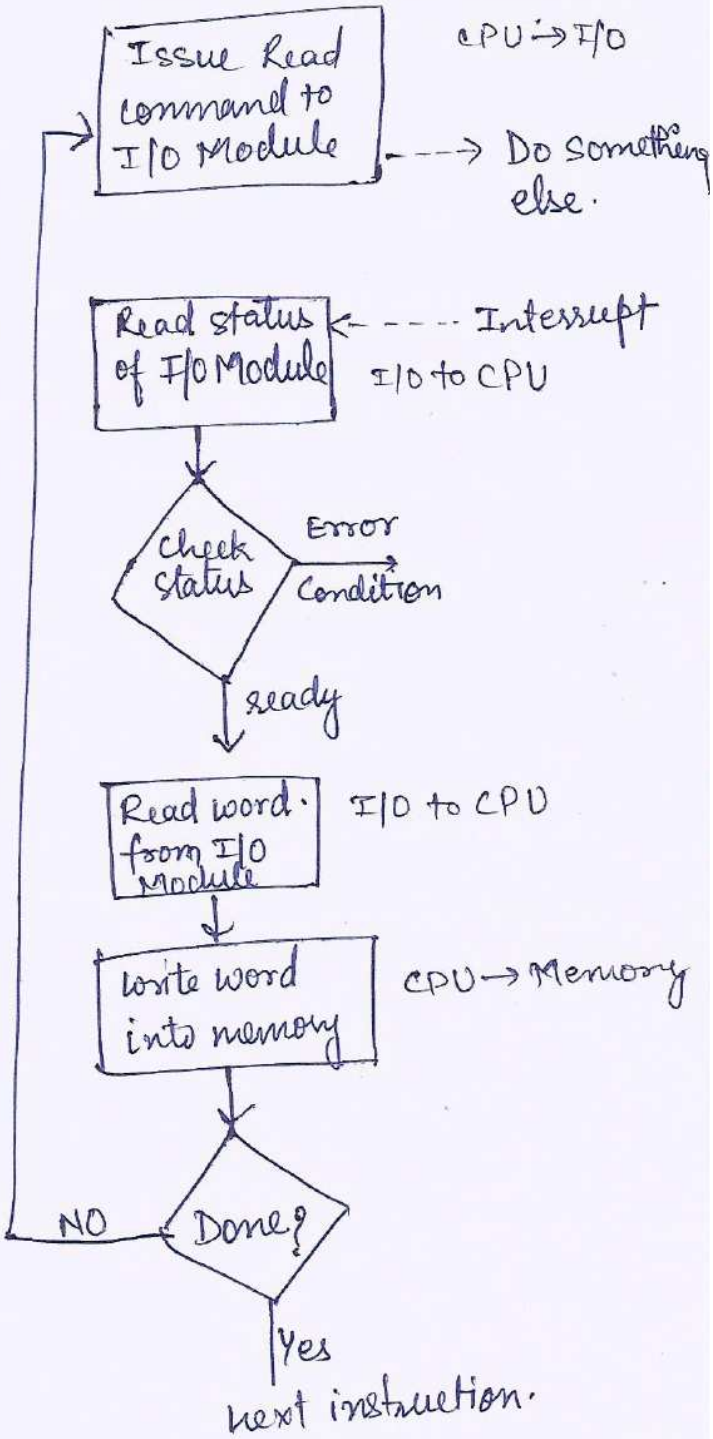
⇒ In this method, the CPU stays in a program loop until I/O unit indicates that it is ready for the data transfer.

⇒ Time consuming

⇒ ~~Less efficient method~~ Less efficiency.



Programmed I/O



Interrupt Driven I/O

Interrupt Driven I/O

→ special commands to inform the interface to issue an interrupt request signal when the data is available from the device. meantime the CPU can proceed to execute another program.

The interface meanwhile keeps monitoring the device.

⇒ when the interface determines the device is ready to data transfer, it generates an interrupt request to the computer.

⇒ upon detecting the interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process I/O transfer, and returns to the task it was actually performing.

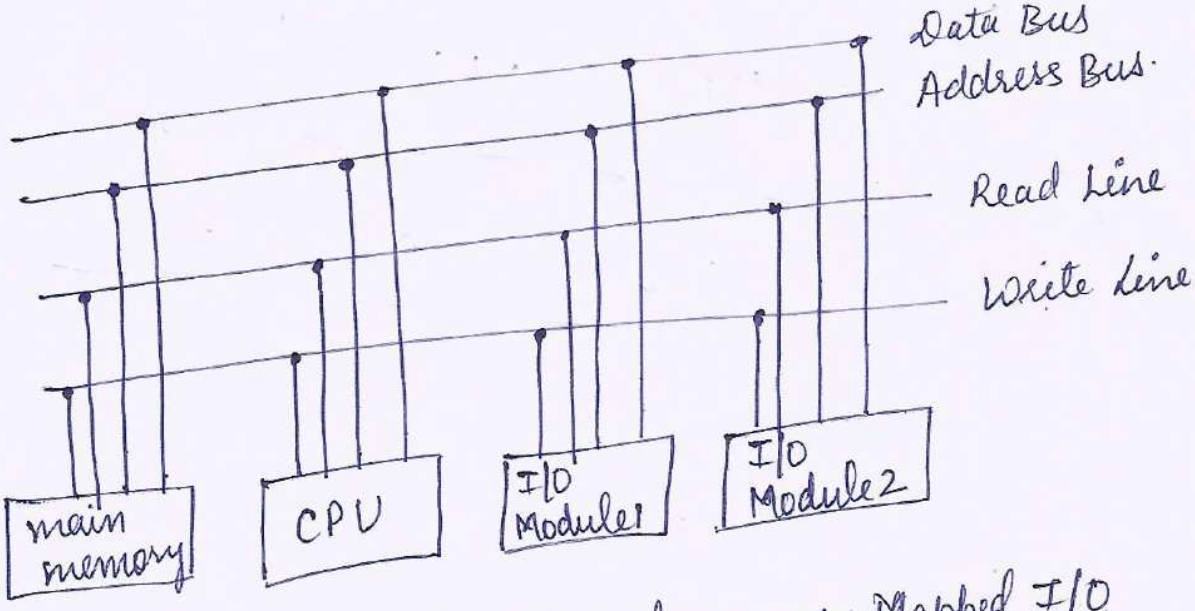
Isolated I/O vs. Memory Mapped I/O

When the processor, main memory and I/O share a common bus, two modes of addressing are possible.

- Memory Mapped I/O
- Isolated I/O (Input-output mapped I/O).

Memory Mapped I/O :-

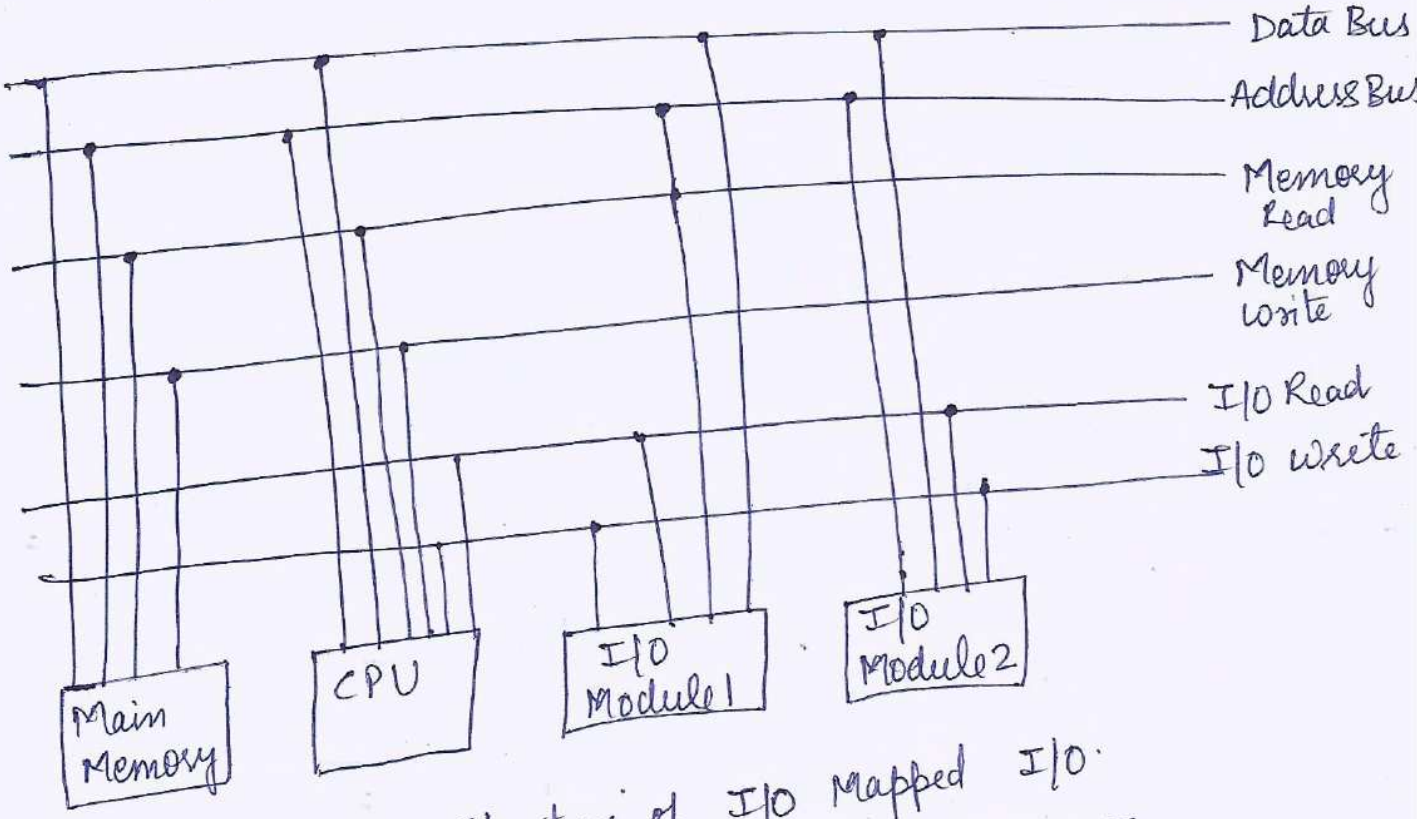
- There is a single address space for memory locations and I/O devices.
- Processor uses same read instructions to access both the memory and Input-output devices.
- No specific I/O-instructions.
- One set of read and write signals



General structure of memory Mapped I/O

Isolated I/O or Input-output ~~I/O~~ Mapped I/O:

- There are separate address spaces for memory or I/O device.
- CPU specifies that the address on lines is for memory or I/O
- Distinct input output commands.
- ~~separ~~ separate read and write lines.



Structure of I/O Mapped I/O.

Comparison between programmed and Interrupt driven I/O

Programmed I/O	Interrupt Driven I/O
→ Can be implemented without any additional hardware	→ Additional hardware required for interrupt handling
→ based on busy waiting CPU keeps checking the status of I/O device.	→ CPU, switches to another task. without waiting
→ Low efficiency	→ higher efficiency than programmed I/O
→ Only one activity can be handled using programmed I/O	→ Multiple I/O activity can be handled in overlapped fashion here

Comparison between Input-output Mapped (Isolated) and Memory Mapped I/O

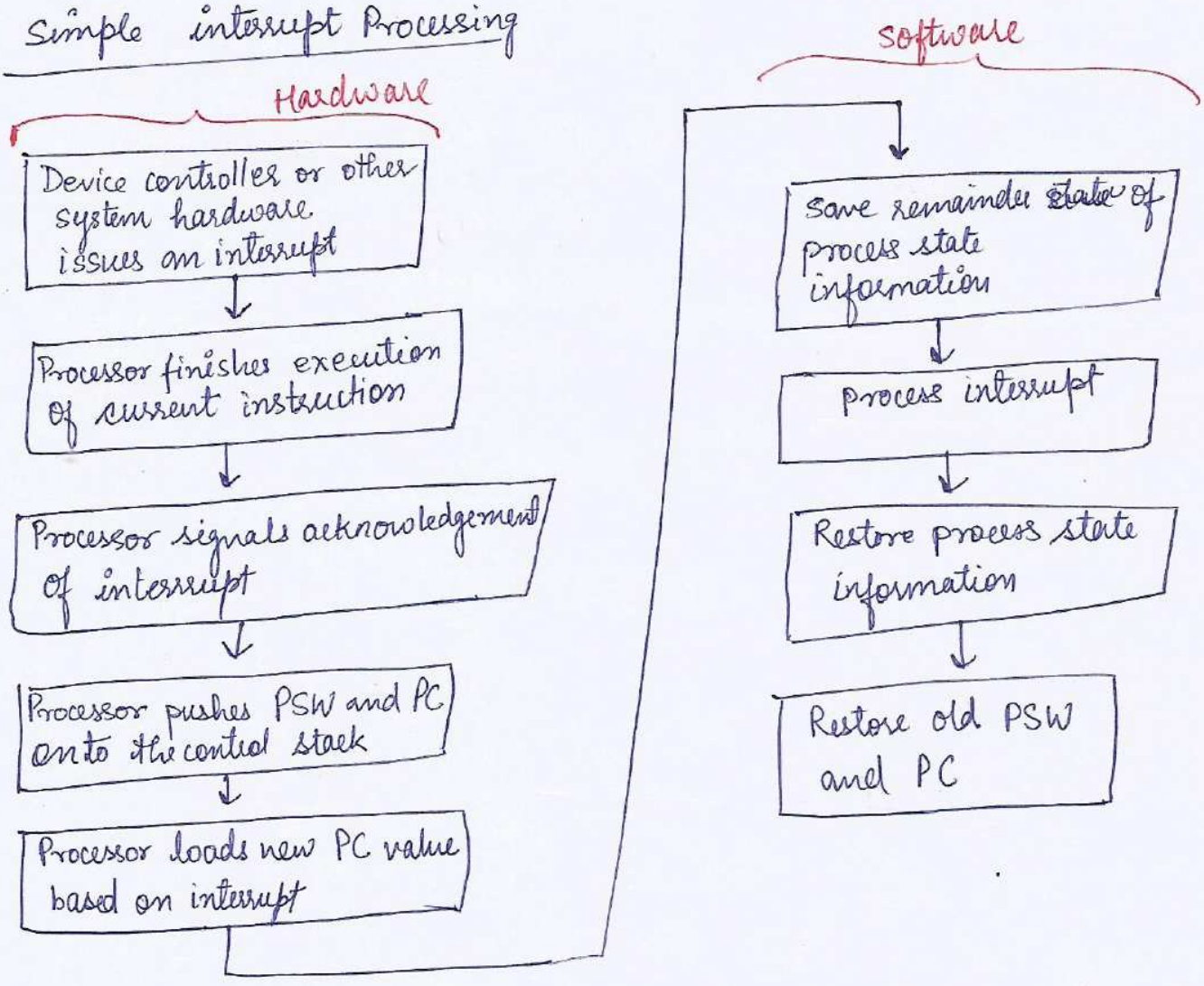
Memory Mapped I/O	Peripheral Mapped I/O or Input-output Mapped I/O
– Same address space for memory and I/O devices.	→ two (separate) address spaces for memory and I/O devices.
– uses same instructions for both I/O and memory operations	→ separate commands for memory operations and different commands for I/O.
→ one set of read and write signals for memory and I/O	→ separate read and write signals for memory and I/O
– Memory Mapped I/O is less efficient	→ More efficient.

Interrupt Processing :

The basic method of interrupting the CPU is done by activating a control line that connects the interrupt source to the CPU.

CPU ⇒ * recognizes the presence of interrupt
* executes a specific interrupt handling program.
* determines the source of interrupt
* determines the address (branch address) of the interrupt handling program.

Simple interrupt Processing



Design issues:

Two design issues in implementing Interrupt I/O.

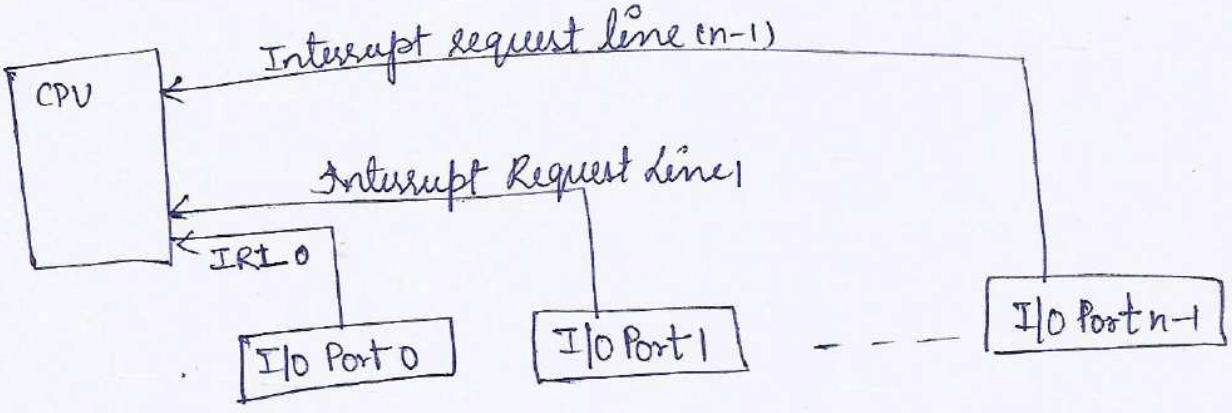
(1) → To determine which device issued the interrupt from multiple devices.

(2) if multiple devices interrupts have occurred, how does the the processor decide which one to process.

Issues: Device identification

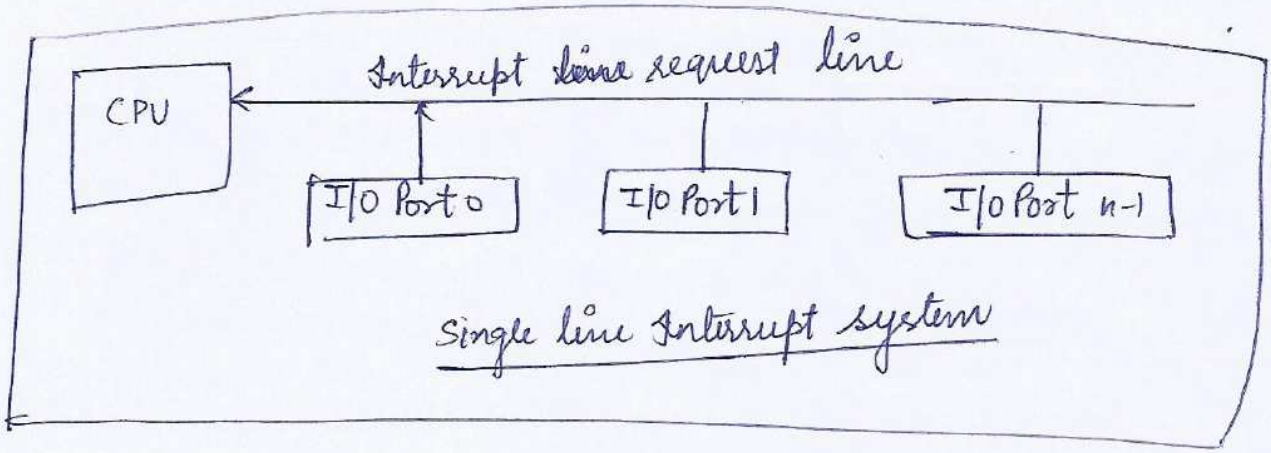
4 Techniques exist.
- → Multiple interrupt lines
- → Software Poll
- → Daisy chain (Hardware Poll, vectored)
- → Bus Arbitration (vectored)

Multiple Interrupt Lines:-



⇒ immediate recognition of device.
⇒ separate Interrupt request line
⇒ impractical approach.

multiple Interrupt line system



Single line Interrupt system

Software Poll:

⇒ when processor detects the interrupt, it branches to an interrupt service-routine whose job is to poll each I/o module to determine which module caused the interrupt.

⇒ Time consuming

⇒ Priority is can be implemented by defining polling sequence.

Vectored Interrupt using Daisy chaining: (Hardware Poll)

⇒ All I/o module shares common interrupt request line

→ The interrupt acknowledgement line is daisychained through the modules.

– when a processor senses the interrupt, it sends out an interrupt acknowledge propagating through a series of I/o module until it gets a requesting module.

– The requesting module responds by placing a word on data lines. This word is referred to as vector.

Bus Arbitration Techniques:- (vectored interrupts)

→ with bus arbitration technique, an I/O module first gain control of bus before it can raise the interrupt request line.

→ Thus only one module can raise the line at a time.

Types of Interrupts:

Program interrupts

These are generated by some condition that occurs as a result of an instruction execution such as:

- → Arithmetic Overflow
- → Division by zero
- → Execution of illegal machine instruction
- → Segment limit violation
- → Execution of privileged instruction

Timer Interrupts

These are generated within the processor. This allows operating system to perform certain operations on regular basis.

Input-output interrupts:

These are generated for initiation or completion of I/O operations I/O failure or I/O error too can generate an interrupt.

Hardware Failure Interrupts

These are generated by a failure; such as power failure or memory parity error.

Interrupts vs. Exceptions:

An interrupt is generated by a signal from hardware, and it may occur at random times during execution of a program.

An exception is generated from software, and it is provoked by the execution of an instruction.

Hardware and Software interrupts

→ When microprocessor receive interrupts signals through pins (hardware) of microprocessor. These are known as Hardware interrupts

→ Software interrupts are those which are inserted in the between the program which means these are mnemonics of the microprocessor.

Vectored and non vectored interrupts

- In vectored interrupts, the source (or device) that interrupts supplies the branch information to the computer.
- In a non-vectored interrupt, the branch address is assigned to a fixed location in the memory.

Maskable and non-maskable:

- Maskable interrupts are those which can be disabled or ignored by the microprocessor
- Non-maskable: which can not be disabled or ignored by the microprocessor.

Types of interrupts:

4 categories

→ Program interrupts
→ Timer interrupts
→ Input/output interrupts
→ Hardware failure

Direct memory Access :-

→ Both interrupt driven and programmed I/O require involvement of CPU for data transfer.

→ CPU can be better utilized for program execution.

→ I/O activities are slow and involvement of CPU will not have a desired effect on the system performance.

→ DMA increases the speed of I/O transfer by eliminating the role played by CPU in I/O operations.

→ when large amount of data is stored/transferred from CPU, a DMA module can be used.

DMA operates in the following ways:

when I/O is requested, the CPU instructs the DMA module about the operation, by providing information:

- → Source address
- → Target address
- → Read / write
- → Number of words to be read or written.

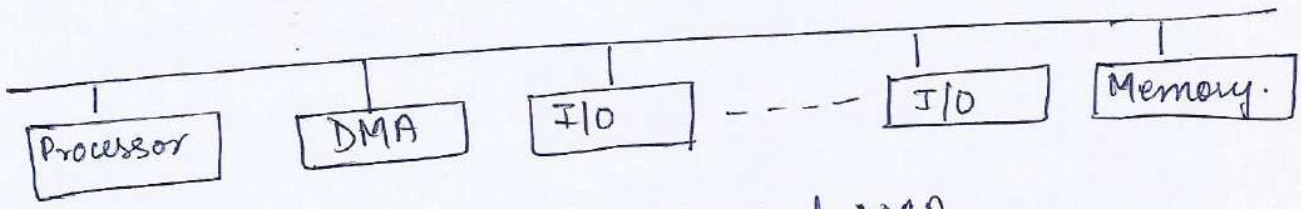
⇨ CPU loads the DMA registers 'Data count' and 'Address Register' with number of bytes to be transferred and starting address memory location respectively.

when the DMA controller is ready to transmit or receive data, it activates the DMA request line to CPU. CPU wait for the next break point. CPU now leaves the control of system bus and activates DMA acknowledge.

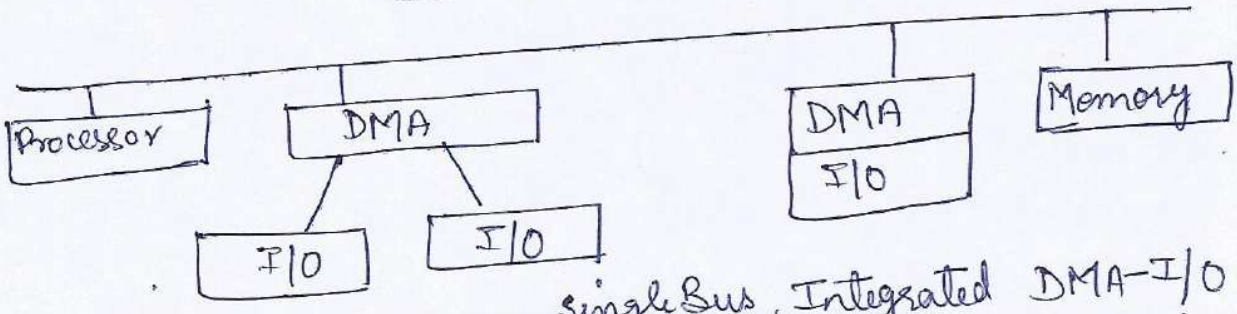
→ The DMA controller transfers the data directly from or to main memory. After, a word is transferred. Data count (or word count register) and Address register are updated.

→ If the data count register has not reached zero but the I/O device is not ready to send or receive the data, the DMA controller releases the system bus to CPU by deactivating DMA request line.

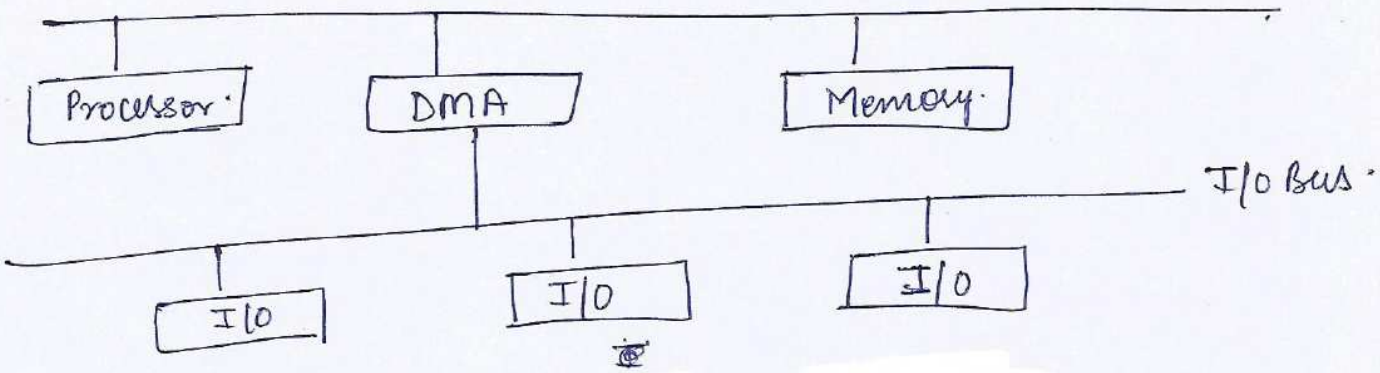
– if data count register is decremented to zero, DMA controller finally relinquishs the control of system bus. It may also send an interrupt signal to CPU to indicate the end of data transfer.



Single Bus Detached DMA

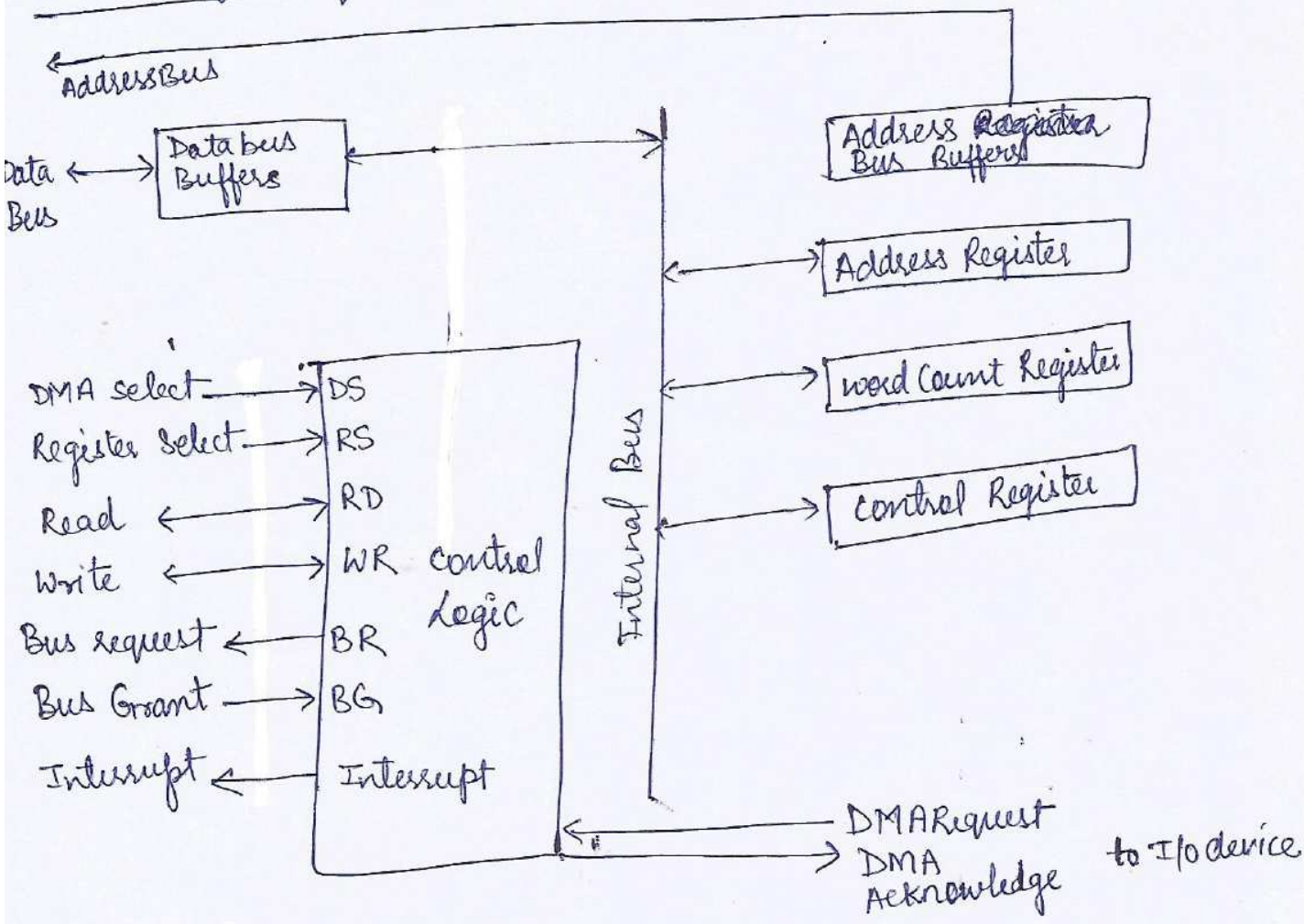


Single Bus, Integrated DMA-I/O



I/O Bus.

Block Diagram of DMA



→ Bus Request (BR) input is used ~~to~~ by the DMA controller to request to CPU to leave the control of system bus.

→ The CPU activates the 'Bus Grant' (BG) output to inform the DMA controller.

→ When DMA terminates the transfer, it disable the bus request line. CPU disables the bus grant; takes control of buses and returns it to its normal operation.

DMA Data Transfer modes:

→ DMA block transfer
→ Cycle stealing mode
→ Transparent DMA.

(1) DMA Block Transfer:-

→ In DMA block transfer technique, a block of data of arbitrary length can be transferred in a single burst.
→ This DMA mode is needed for secondary memories like disk drivers, where data transmission can not be stopped or slowed without loss of data.

(2) cycle steading Mode:

In cycle stealing mode, the DMA controller is allowed to transfer one word at a time, after which it must return the control of the bus to the CPU.

* DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.

DMA module transfers a word and returns the control to the processor Note that this is not an interrupt; the processor does not save a context and do something else.
Rather, processor pauses for one bus cycle. The overall effect is to cause the processor to execute more slowly. Nevertheless, for a multiple-word I/O transfer, DMA is far more efficient than interrupt driven or programmed I/O.

(3) Transparent DMA:

In transparent DMA, DMA is allowed to steal only those cycles when the CPU is not using the system bus.

CPU does not require to have control of system bus during Decode instruction or execute instruction phase.

* DMA transfering data during transparent DMA does not have any adverse effect on CPU performance.

Input - output channels:

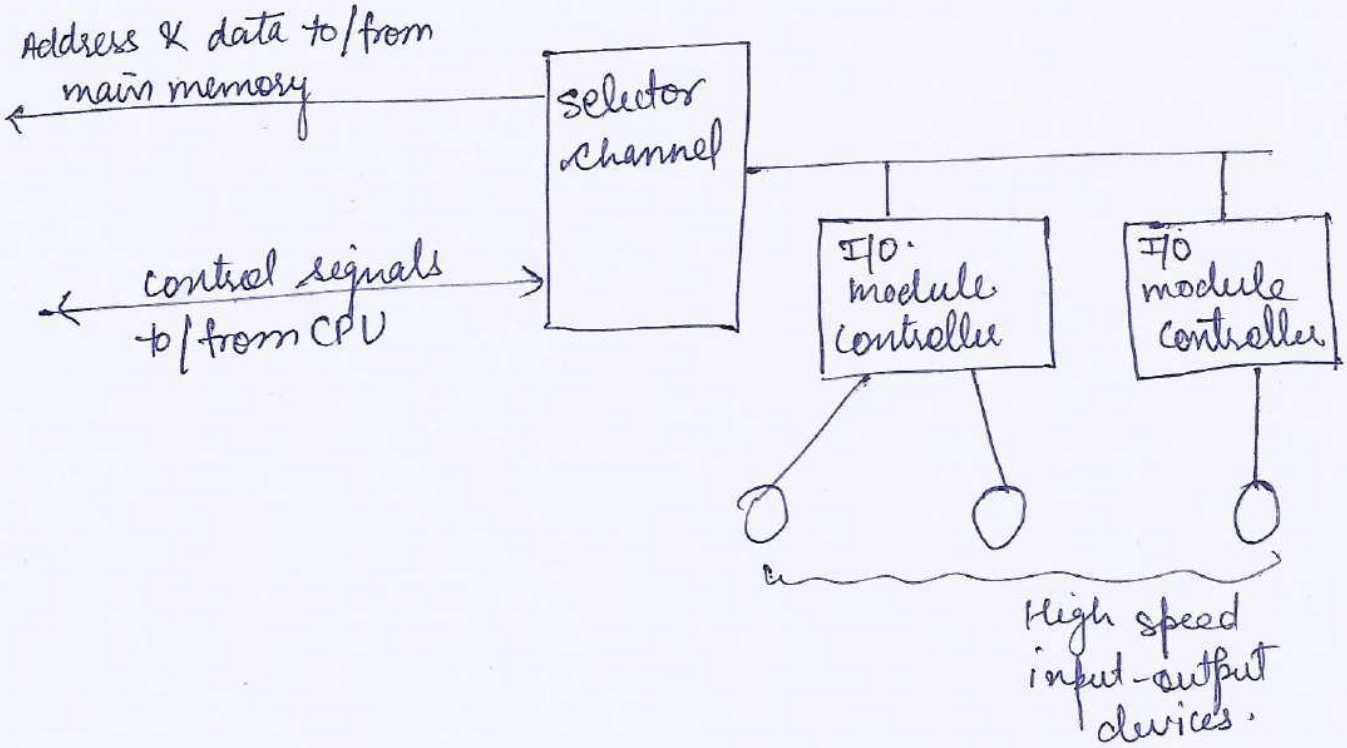
Input-output channels or input-output processors, virtually retrieves CPU from all I/O related activities. It has complete control over device. It can also communicate with CPU.

* I/O processor is a processor like CPU with limited number of instructions. An I/O processor has the ability to execute the I/O instructions It can also carry out the work of data conversion required by the input-output device.

There are two types of I/O channels:

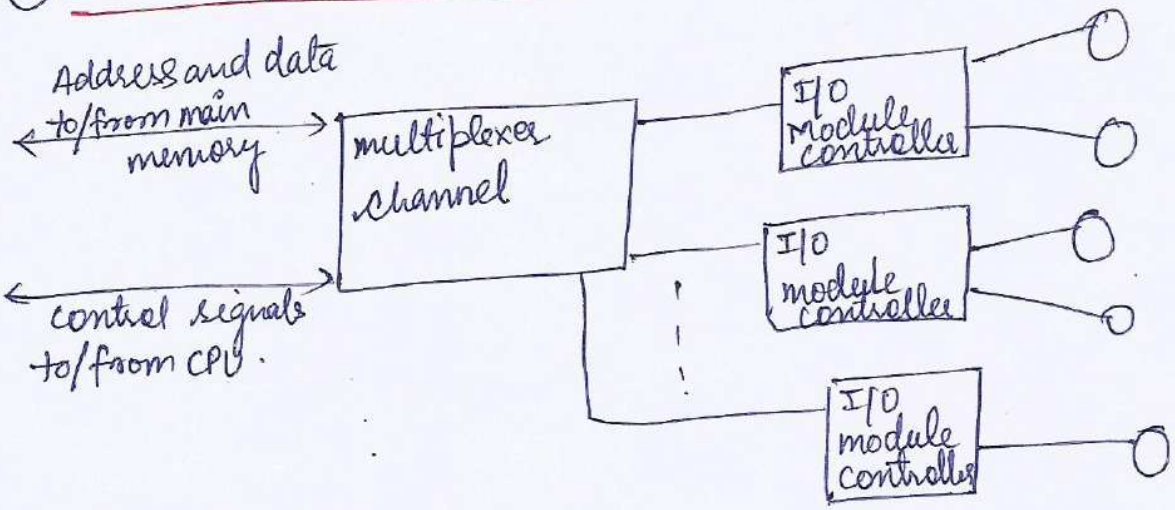
1. Selector channel
2. Multiplexer channel.

1. Selector channel:



- A selector channel can handle multiple high speed devices. Only one device is selected at a time for communication.
- A selector channel is useful in high speed transfer of data in burst mode.

(2) A multiplexer channel:



A multiplexer channel can handle I/O with a number of devices at the same time. If the device is slow then byte, multiplexing is used.

Example There are three devices which need to send individual bytes are

B1, B2, ----- B5 → data from 1st device.
X1, X2, ----- X5 → data from 2nd device.
Y1, Y2 ----- Y5 → data from 3rd device.

Then on a byte multiplexer channel may send the ~~data~~ bytes as B1 X1 Y1 B2 X2 Y2 B3 X3 Y3 B4 X4 Y4 ------.

Data Transfer:

- Synchronous
- Asynchronous

Synchronous: If registers in the interface share a common clock with CPU registers, the transfer between two units is said to be synchronous.

Asynchronous: The internal timing in each unit is independent from the other in that each uses its own private clock for internal registers, in this case the two units are said to be asynchronous.

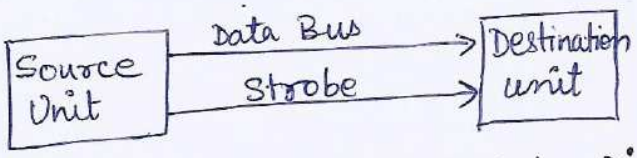
Asynchronous Data Transfer:

→ two independent units transmit control signals to indicate the time at which data is being transmitted.

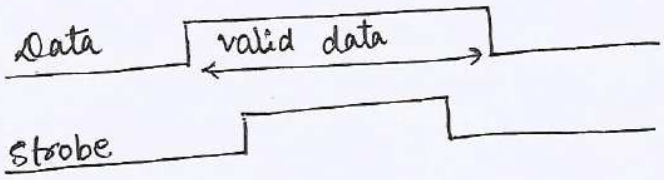
→ Two methods
- Strobe
- Handshaking

Strobed Method (one way) One of communicating device supplies all the control signals.

- Strobe can be activated by either source or destination



Source iniated Strobe Diagram



Timing Diagram

The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

* The source unit first places the data on the databus. After a brief delay to ensure that the data settle a steady value, the source activates the strobe pulse.

* In destination initiated data transfer, destination unit activates the strobe pulse, informing the source to provide the data.

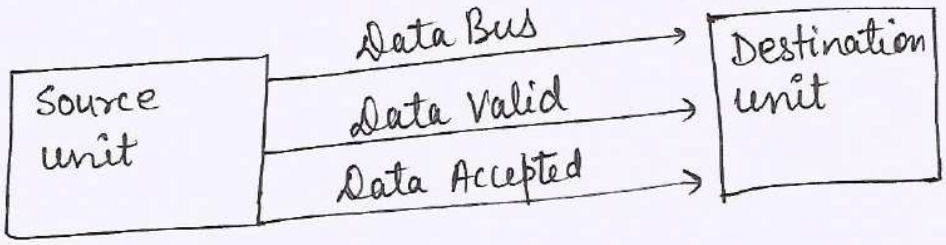
Disadvantage: In strobe method, the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item.

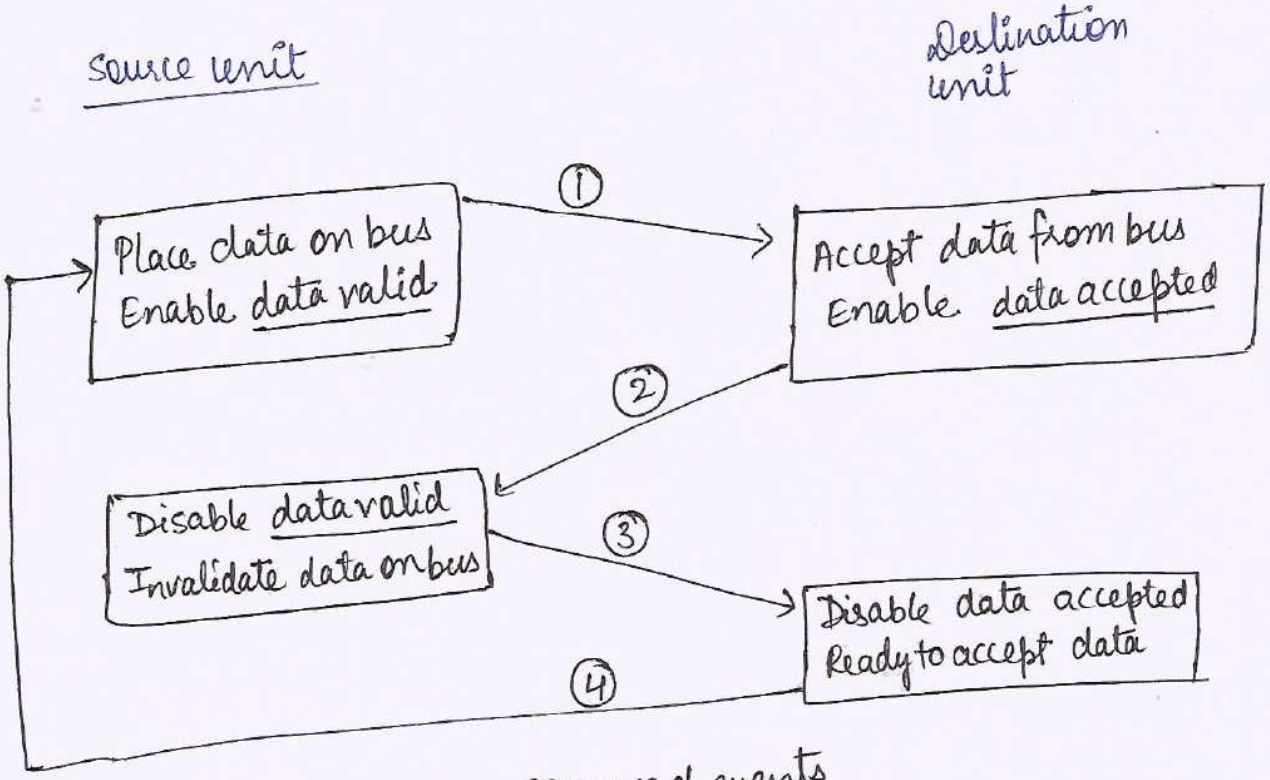
Handshaking: (Two way)

* The disadvantage of oneway control is that it does not verify that the data transfer has been completed successfully. Data may be lost.

* Control of data transfer is based on the use of handshake between processor and the device selected for input-output.

Block Diagram





sequence of events

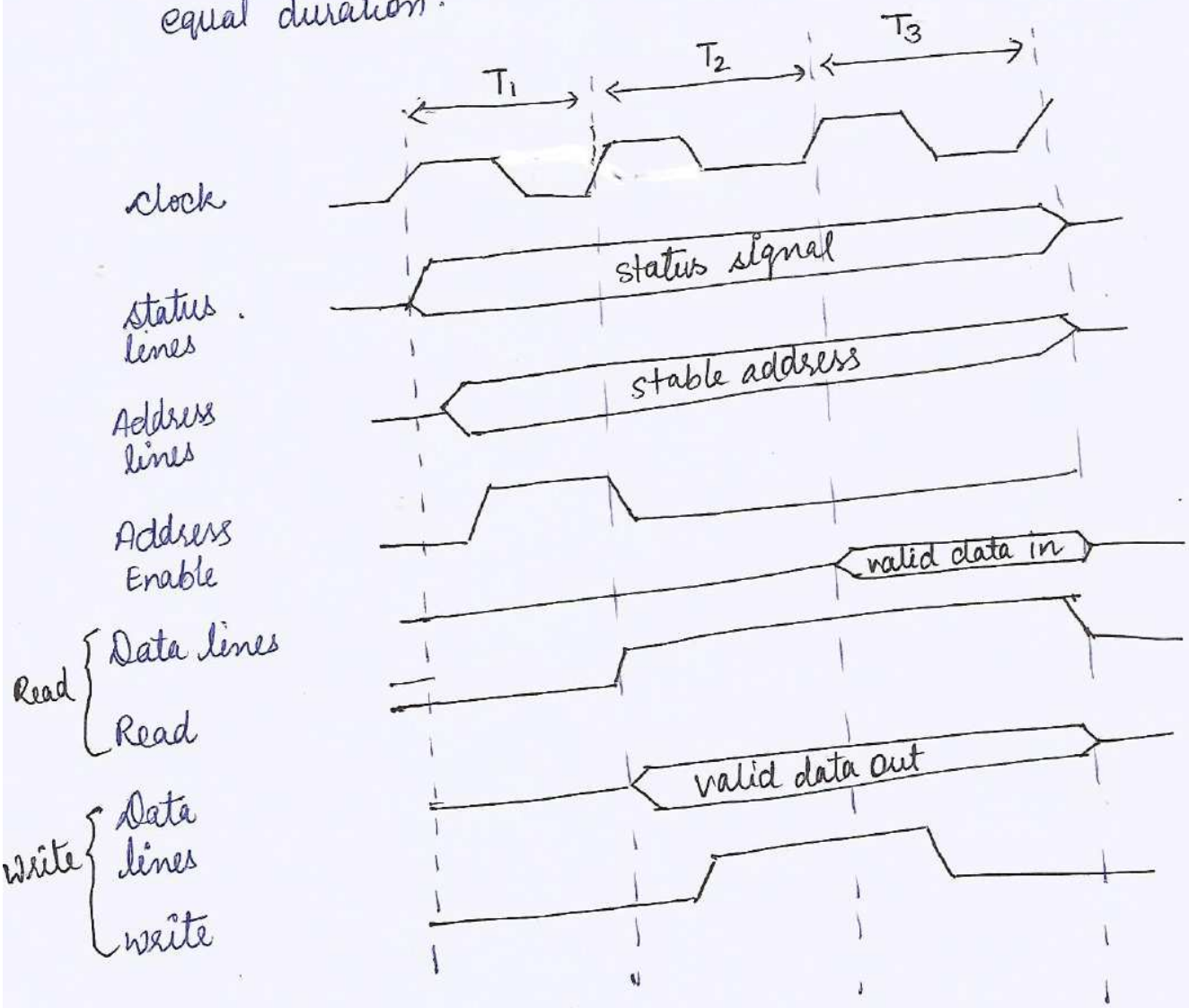
Synchronous Bus:

→ include a clock in the control lines
→ A fixed protocol for communication that is relative to the clock.

Advantage: - involves very low logic and can run very fast.
Disadvantage: Every device on the bus must run at the same clock rate.

Synchronous timing

→ The occurrence of events on the bus is determined by a clock
→ A clock is a regular sequence of alternating 1's and 0's of equal duration.



$T_1 \Rightarrow$ processor places address on address lines and may assert various status lines.

$\Rightarrow$ once the address lines have stabilized, the processor issues address enable signal.

For Read operation

$T_2 \Rightarrow$ Processor issues a read command.

$T_3 \Rightarrow$ Memory recognizes the address and places the data on the data lines.

For write operation

$T_2 \Rightarrow$ processor puts the data on data lines.

$\Rightarrow$ once the data on data lines have stablized, the processor issues a write command.

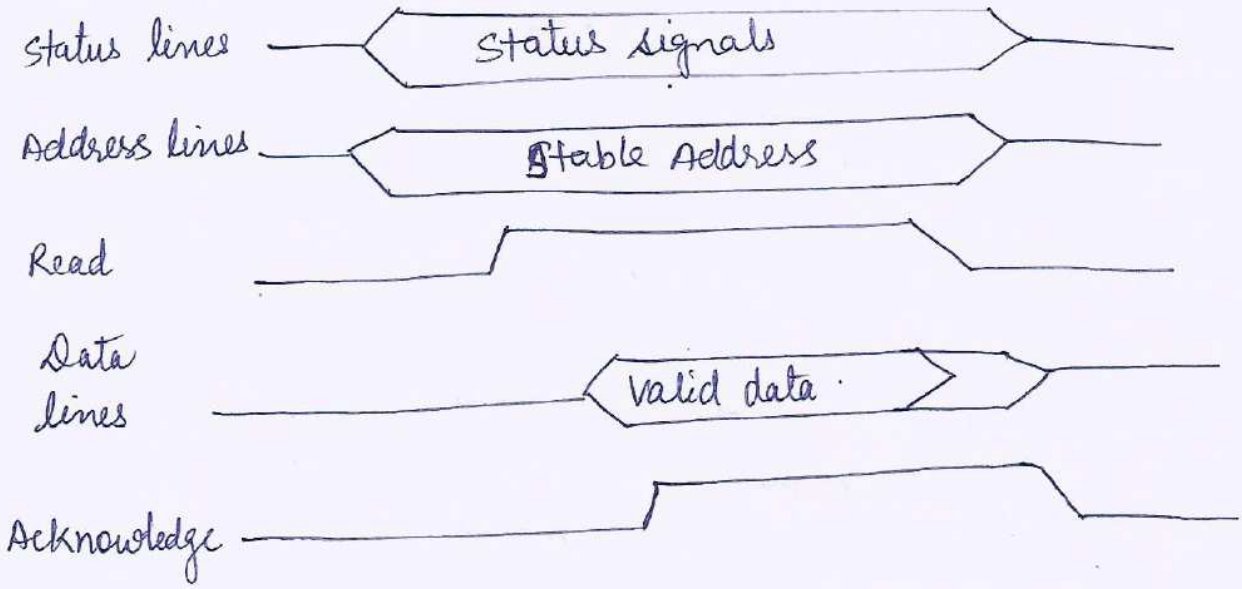
$T_3 \Rightarrow$ Memory copies the information from data lines

Asynchronous Bus:

⇒ No clock

⇒ The occurrence of one event on a bus follows and depends on the occurrence of previous event.

Sequence of events for read operation:



- The processor places address and status signals on the bus.
- After signals have stabilized, the processor issues a read command.
- After address decoding, memory places the data on data lines.
- After stabilized data, memory asserts ~~ack~~ acknowledges to the processor.

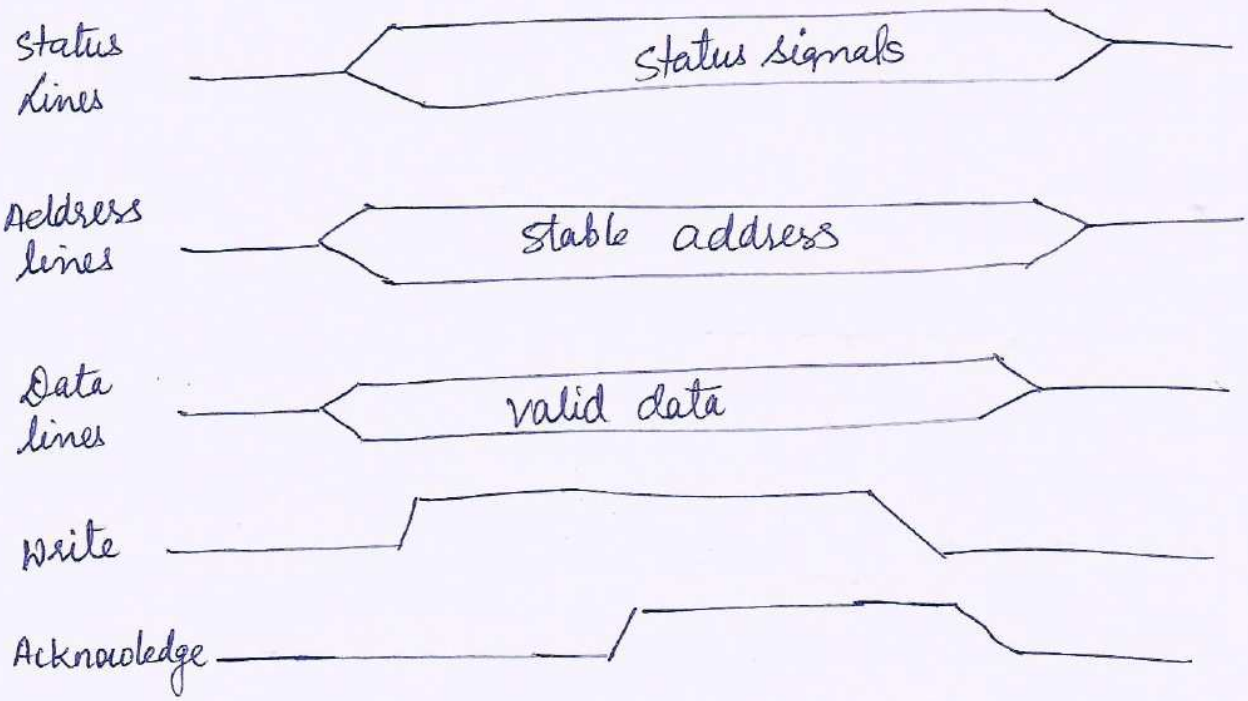
Synchronous Timing

→ Simple to implement and test
→ Less flexible
→ Devices must be on same clock rate

Asynchronous Timing

- Hard to implement and test
- More flexible
- combination of slow and fast devices.

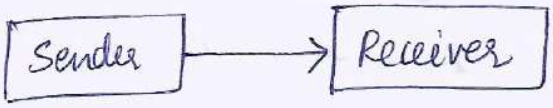
Sequence of events of Write operation:



- The processor places address and status signals on the bees.
- The master places the data lines at the same time
- The processor issues write command.
- → Memory module responds to the write command by copying the data from the data lines and asserting the acknowledge line.
- The master then drops the write signal and memory module drops the acknowledge signal.

Serial communication
or
Serial Transmission

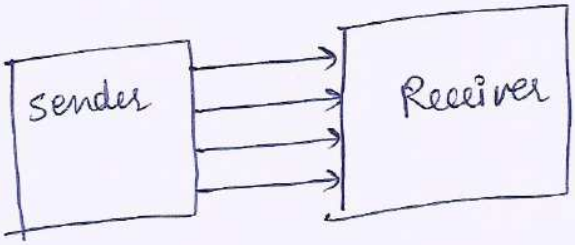
⇒ Serial data transmission occurs by transmitting data one bit at a time in sequential order over a bus.



start bit data bits stop bits ⇒ To know the start and end of transmission.

Parallel Transmission

In parallel transmission several bits can be transmitted at the same time.



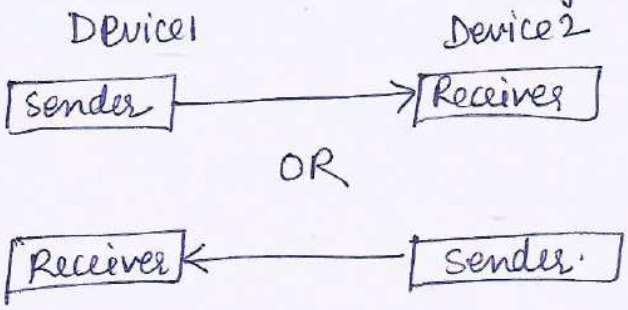
Serial Communication can be
- Simplex
- Half Duplex
- Full Duplex

Simplex Transmission: Signals travels in one direction only.
eg. Keyboard

Half Duplex:

→ capable of sending signals in both the directions, but in only one direction at a time

eg ~~Telephone~~ Walkie Talkie.



Full Duplex: (Bidirectional Transmission)

→ This method allows signal transmission in both the directions simultaneously.

eg Telephone IP service.