4.1 ACCESSING I/O DEVICES

A simple arrangement to connect I/O devices to a computer is to use a single bus structure. It consists of three sets of lines to carry

- Address
- Data
- Control Signals.

When the processor places a particular address on address lines, the devices that recognize this address responds to the command issued on the control lines. The processor request either a read or write operation and the requested data are transferred over the data lines.

memory mapped I/O: When I/O devices & memory share the same address space, the arrangement is called **memory mapped I/O.**

Single Bus Structure

Eg:-



Move DATAIN, R_0 → Reads the data from DATAIN then into processor register R_0.
Move R_0, DATAOUT → Send the contents of register Ro to location DATAOUT.
DATAIN → Input buffer associated with keyboard.
DATAOUT → Output data buffer of a display unit / printer

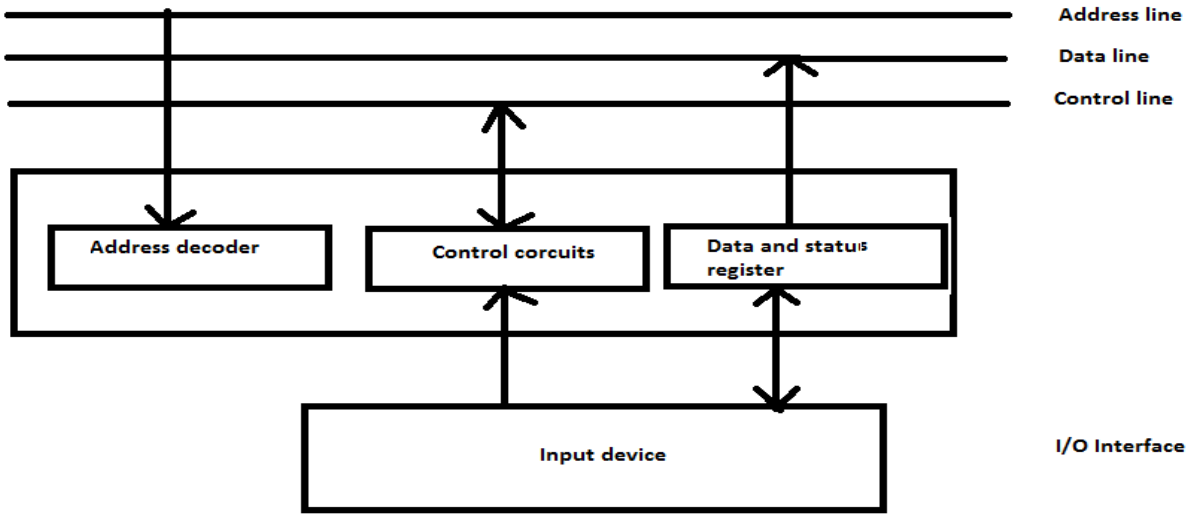
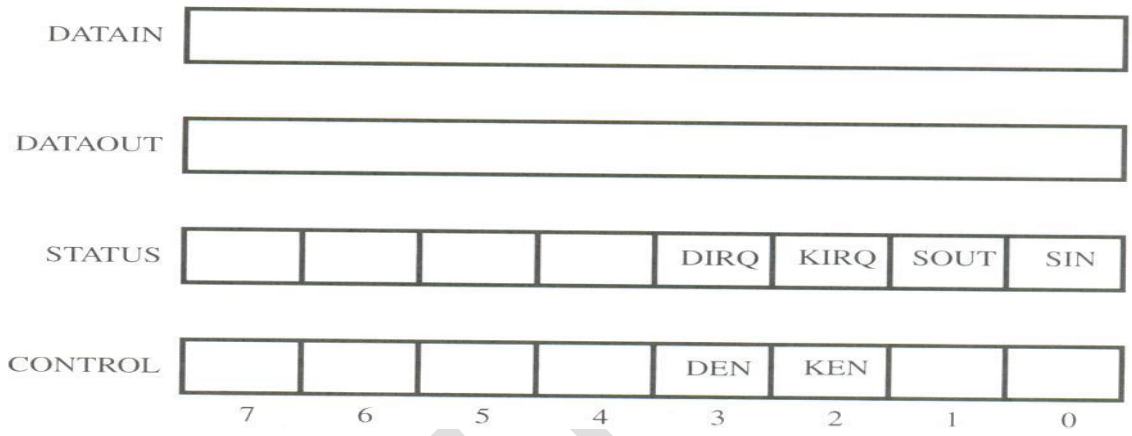


Fig: I/O Interface for an input device

Address Decoder:
It enables the device to recognize its address when the address appears on address lines.
Data register →It holds the data being transferred to or from the processor.
Status register →It contains information Relevant to the operation of the I/O devices.
The address decoder, data & status registers and the control circuitry required to co-ordinate I/O transfers constitute the device‟s I/F circuit.

For an input device, SIN status flag is used SIN = 1, when a character is entered at the keyboard.

For an output device, SOUT status flag is used SIN = 0, once the char is read by processor.



DIR Q Interrupt Request for display.
KIR Q Interrupt Request for keyboard.
KEN keyboard enable.
DEN Display Enable.
SIN, SOUT status flags.
The data from the keyboard are made available in the DATAIN register & the data sent to the display are stored in DATAOUT register.

Program:

```
WAIT K   Move # Line, Ro
         Test Bit #0, STATUS
         Branch = 0 WAIT K
         Move DATAIN, R1
WAIT D   Test Bit #1, STATUS
         Branch = 0 WAIT D
         Move R1, DATAOUT
         Move R1, (Ro)+
         Compare #OD, R1
         Branch = 0 WAIT K
         Move #DOA, DATAOUT
         Call PROCESS
```

Explanation
This program, reads a line of characters from the keyboard & stores it in a memory buffer starting at locations LINE. Then it calls the subroutine “PROCESS” to process the input line. As

each character is read, it is echoed back to the display. Register Ro is used as a updated using Auto – increment mode so that successive characters are stored in successive memory location. Each character is checked to see if there is carriage return (CR), char, which has the ASCII code 0D(hex). If it is, a line feed character (on) is sent to move the cursor one line down on the display & subroutine PROCESS is called. Otherwise, the program loops back to wait for another character from the keyboard.

PROGRAM CONTROLLED I/O

Here the processor repeatedly checks a status flag to achieve the required synchronization between Processor & I/O device.(ie) the processor polls the device.

There are 2 mechanisms to handle I/o operations. They are,

➔ Interrupt Driven I/O

➔ DMA (Synchronization is achieved by having I/O device send special over the bus where is ready for data transfer operation)

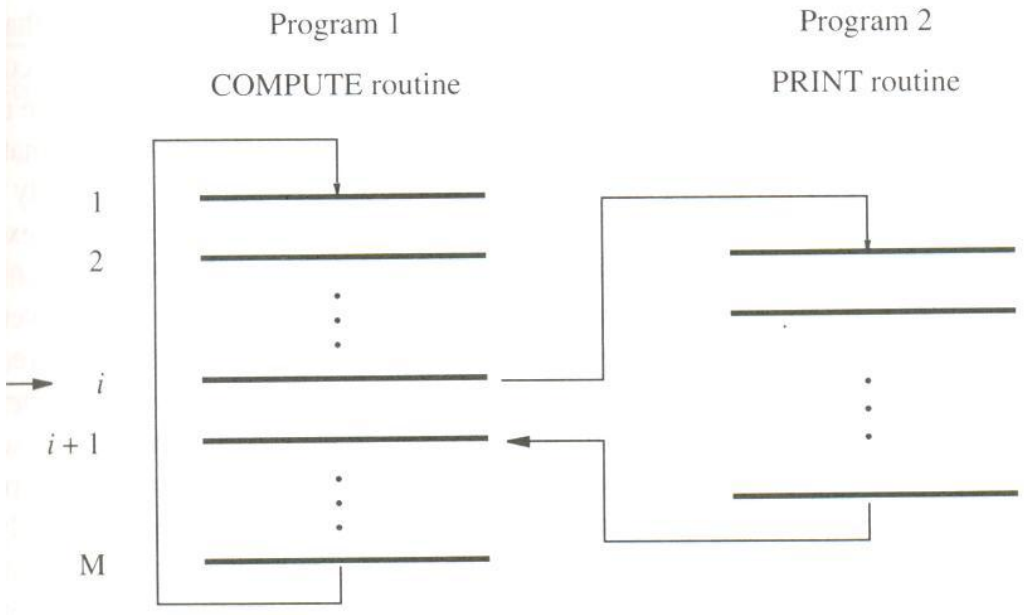
DMA:

Synchronization is achieved by having I/O device send special over the bus where is ready for data transfer operation) .It is a technique used for high speed I/O device. Here, the input device transfer data directly to or from the memory without continuous involvement by the processor

4.2 INTERRUPTS

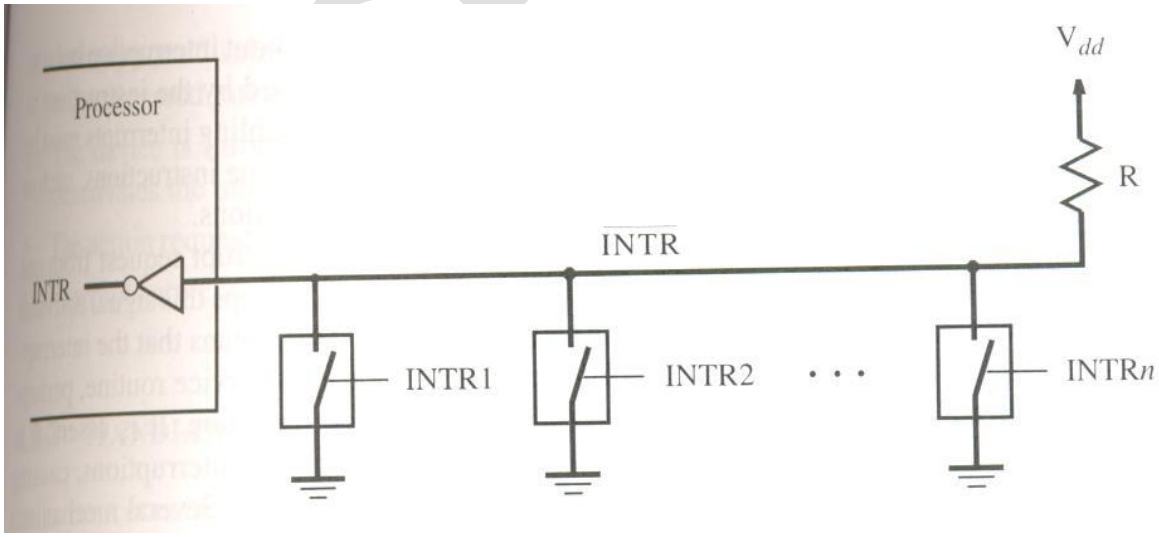
- When a program enters a wait loop, it will repeatedly check the device status. During this period, the processor will not perform any function.
- The Interrupt request line will send a hardware signal called the interrupt signal to the processor.
- On receiving this signal, the processor will perform the useful function during the waiting period.
- The routine executed in response to an interrupt request is called **Interrupt Service Routine.**
- The interrupt resembles the subroutine calls.

Fig:Transfer of control through the use of interrupts



- The processor first completes the execution of instruction i Then it loads the PC(Program Counter) with the address of the first instruction of the ISR.
- After the execution of ISR, the processor has to come back to instruction i + 1.
- Therefore, when an interrupt occurs, the current contents of PC which point to i +1 is put in temporary storage in a known location.
- A return from interrupt instruction at the end of ISR reloads the PC from that temporary storage location, causing the execution to resume at instruction i+1.
- When the processor is handling the interrupts, it must inform the device that its request has been recognized so that it remove its interrupt requests signal.
- This may be accomplished by a special control signal called the **interrupt acknowledge signal.**
- The task of saving and restoring the information can be done automatically by the processor.
- The processor saves only the contents of **program counter & status register** (ie) it saves only the minimal amount of information to maintain the integrity of the program execution.
- Saving registers also increases the delay between the time an interrupt request is received and the start of the execution of the ISR. This delay is called the **Interrupt Latency.**
- Generally, the long interrupt latency in unacceptable.
- The concept of interrupts is used in Operating System and in Control Applications, where processing of certain routines must be accurately timed relative to external events. This application is also called as **real-time processing**.

4.3 INTERRUPT HARDWARE



- A single interrupt request line may be used to serve „n" devices. All devices are connected to the line via switches to ground.
- To request an interrupt, a device closes its associated switch, the voltage on INTR line drops to 0(zero).

- If all the interrupt request signals (INTR1 to INTRn) are inactive, all switches are open and the voltage on INTR line is equal to Vdd.
- When a device requests an interrupts, the value of INTR is the logical OR of the requests from individual devices.
  (ie) **INTR = INTR1+……+INTRn**
- **INTR** □□It is used to name the INTR signal on common line it is active in the low voltage state.
- **Open collector** (bipolar ckt) or **Open drain** (MOS circuits) is used to drive INTR line.
- The Output of the Open collector (or) Open drain control is equal to a switch to the ground that is open when gates input is in „0" state and closed when the gates input is in "1" state.
- Resistor "R" is called a **pull-up resistor** because it pulls the line voltage upto the high voltage state when the switches are open.

4.4 ENABLING AND DISABLING INTERRUPTS

The arrival of an interrupt request from an external device causes the processor to suspend the execution of one program & start the execution of another because the interrupt may alter the sequence of events to be executed. INTR is active during the execution of **Interrupt Service Routine**. There are 3 mechanisms to solve the problem of infinite loop which occurs due to successive interruptions of active INTR signals.

1. The first possibility is to have processor hardware ignore the interrupt request line until the execution of the first instruction of the interrupt service routine has been completed .then ,by using an interrupt disable instruction as first instruction in interrupt service routine, the programmer can assure that no further interrupts will occur until an interrupt –enable instruction is executed. Typically interrupt enable instruction will be the last instruction in ISR.
2.

- Handling interrupt request from a single device. The following are the typical scenario.
  - ➔ The device raises an interrupt request.
  - ➔ The processor interrupts the program currently being executed.
  - ➔ Interrupts are disabled by changing the control bits is PS (Processor Status register)
  - ➔ The device is informed that its request has been recognized & in response, it deactivates the INTR signal.
  - ➔ The actions are enabled & execution of the interrupted program is resumed.

Edge-triggered:

The processor has a special interrupt request line for which the interrupt handling circuit responds only to the leading edge of the signal. Such a line said to be edge-triggered.

4.5 HANDLING MULTIPLE DEVICES

When several devices requests interrupt at the same time, it raises some questions. They are.

- ➢ How can the processor recognize the device requesting an interrupt?
- ➢ Given that the different devices are likely to require different ISR, how can the processor obtain the starting address of the appropriate routines in each case?
- ➢ Should a device be allowed to interrupt the processor while another interrupt is being serviced?

SACET-CSE

- How should two or more simultaneous interrupt requests be handled?

Polling Scheme:

- ➔ If two devices have activated the interrupt request line, the ISR for the selected device (first device) will be completed & then the second request can be serviced.
- ➔ The simplest way to identify the interrupting device is to have the ISR polls all the encountered with the IRQ bit set is the device to be serviced
- ➔ IRQ (Interrupt Request) -> when a device raises an interrupt requests, the status register IRQ is set to 1.

Merit:

It is easy to implement.

Demerit:

The time spent for interrogating the IRQ bits of all the devices that may not be requesting any service.

Vectored Interrupt:

Here the device requesting an interrupt may identify itself to the processor by sending a special code over the bus & then the processor start executing the ISR. The code supplied by the processor indicates the starting address of the ISR for the device. The code length ranges from 4 to 8 bits. The location pointed to by the interrupting device is used to store the staring address to ISR. The processor reads this address, called the interrupt vector & loads into PC. The interrupt vector also includes a new value for the Processor Status Register. When the processor is ready to receive the interrupt vector code, it activate the interrupt acknowledge (INTA) line.

Interrupt Nesting:

Multiple Priority Scheme:

- In multiple level priority scheme, we assign a priority level to the processor that can be changed under program control.
- The priority level of the processor is the priority of the program that is currently being executed.
- The processor accepts interrupts only from devices that have priorities higher than its own.
- At the time the execution of an ISR for some device is started, the priority of the processor is raised to that of the device.
- The action disables interrupts from devices at the same level of priority or lower.

Privileged Instruction:

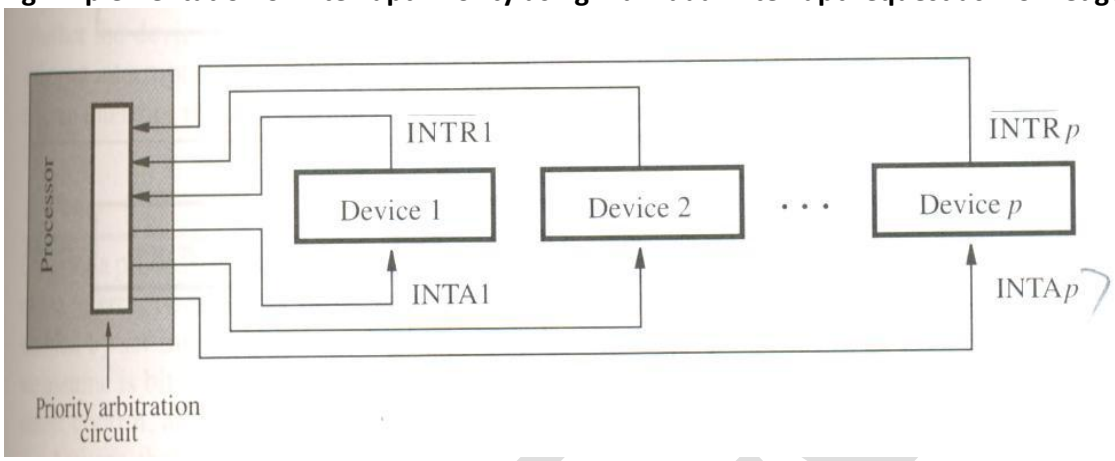
- The processor priority is usually encoded in a few bits of the Processor Status word. It can also be changed by program instruction & then it is write into PS. These instructions are called **privileged instruction.** This can be executed only when the processor is in supervisor mode.
- The processor is in supervisor mode only when executing OS routines.
- It switches to the user mode before beginning to execute application program.

Privileged Exception:

- User program cannot accidently or intentionally change the priority of the processor & disrupts the system operation.
- An attempt to execute a privileged instruction while in user mode, leads to a special type of interrupt called the privileged exception.

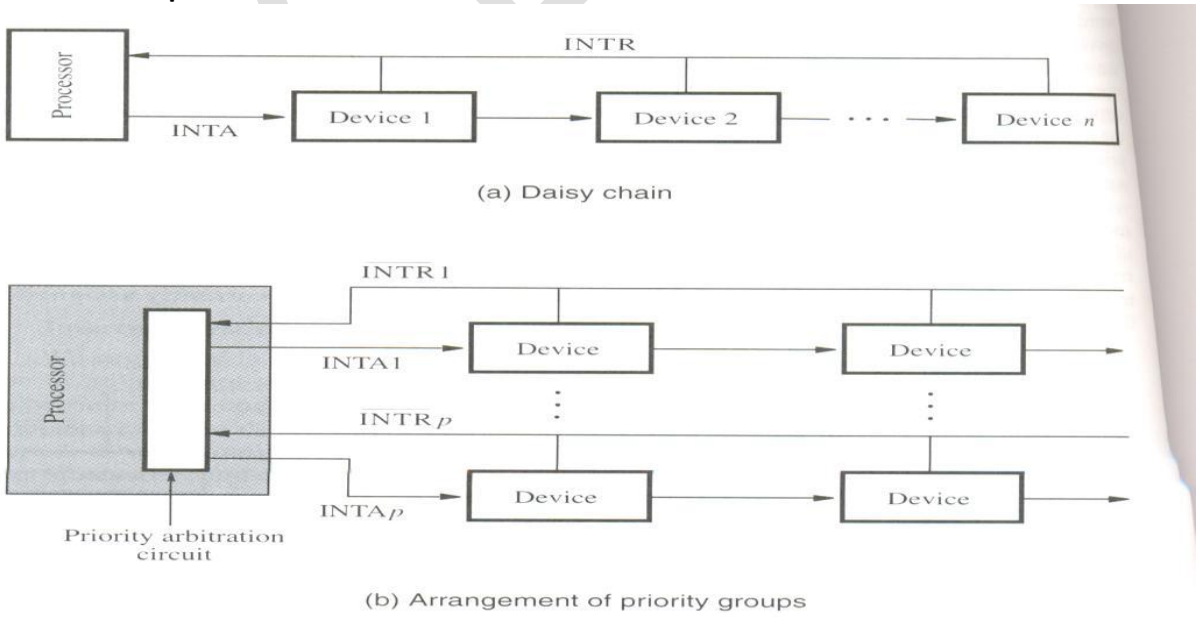
SACET-CSE

Fig: Implementation of Interrupt Priority using individual Interrupt request acknowledge lines



- Each of the interrupt request line is assigned a different priority level.
- Interrupt request received over these lines are sent to a priority arbitration circuit in the processor.
- A request is accepted only if it has a higher priority level than that currently assigned to the processor

Simultaneous Requests



- The interrupt request line INTR is common to all devices. The interrupt acknowledge line INTA is connected in a daisy chain fashion such that INTA signal propagates serially through the devices.
- When several devices raise an interrupt request, the INTR is activated & the processor responds by setting INTA line to 1. this signal is received by device
- Device1 passes the signal on to device2 only if it does not require any service.
- If devices1 has a pending request for interrupt blocks that INTA signal & proceeds to put its identification code on the data lines.
- Therefore, the device that is electrically closest to the processor has the highest priority

Merits:

- It requires fewer wires than the individual connections.

Arrangement of Priority Groups:

- Here the devices are organized in groups & each group is connected at a different priority level. Within a group, devices are connected in a daisy chain.

Controlling Device Requests:

KEN → Keyboard Interrupt Enable
DEN → Display Interrupt Enable
KIRQ / DIRQ → Keyboard / Display unit requesting an interrupt.

- There are two mechanism for controlling interrupt requests.
- At the devices end, an interrupt enable bit in a control register determines whether the device is allowed to generate an interrupt requests.
- At the processor end, either an interrupt enable bit in the PS (Processor Status) or a priority structure determines whether a given interrupt requests will be accepted.

Initiating the Interrupt Process:

- Load the starting address of ISR in location INTVEC (vectored interrupt).
- Load the address LINE in a memory location PNTR. The ISR will use this location as a pointer to store the i/p characters in the memory.
- Enable the keyboard interrupts by setting bit 2 in register CONTROL to 1.
- Enable interrupts in the processor by setting to 1, the IE bit in the processor status register PS.

Exception of ISR:

- Read the input characters from the keyboard input data register. This will cause the interface circuits to remove its interrupt requests.
- Store the characters in a memory location pointed to by PNTR & increment PNTR.
- When the end of line is reached, disable keyboard interrupt & inform program main.
- Return from interrupt.

Exceptions:

- An interrupt is an event that causes the execution of one program to be suspended and the execution of another program to begin.
- The Exception is used to refer to any event that causes an interruption.
-

Kinds of exception:

- Recovery from errors
- Debugging
- Privileged Exception

SACET-CSE

Recovery From Errors:

- Computers have error-checking code in Main Memory , which allows detection of errors in the stored data.
- If an error occurs, the control hardware detects it informs the processor by raising an interrupt.
- The processor also interrupts the program, if it detects an error or an unusual condition while executing the instance (ie) it suspends the program being executed and starts an execution service routine.
- This routine takes appropriate action to recover from the error.

. **Debugging:**

System software has a program called debugger, which helps to find errors in a program. The debugger uses exceptions to provide two important facilities .They are

- ❖ Trace
- ❖ Breakpoint

Trace Mode:

When processor is in trace mode , an exception occurs after execution of every instance using the debugging program as the exception service routine. The debugging program examine the contents of registers, memory location etc. On return from the debugging program the next instance in the program being debugged is executed The trace exception is disabled during the execution of the debugging program.

Break point:

Here the program being debugged is interrupted only at specific points selected by the user. An instance called the Trap (or) software interrupt is usually provided for this purpose. While debugging the user may interrupt the program execution after instance I When the program is executed and reaches that point it examine the memory and register contents.

Privileged Exception:

To protect the OS of a computer from being corrupted by user program certain instance can be executed only when the processor is in supervisor mode. These are called privileged exceptions.When the processor is in user mode, it will not execute instance (ie) when the processor is in supervisor mode , it will execute instance.

4.6 DIRECT MEMORY ACCESS

A special control unit may be provided to allow the transfer of large block of data at high speed directly between the external device and main memory , without continuous intervention by the processor. This approach is called **DMA.** DMA transfers are performed by a control circuit called the **DMA Controller**. To initiate the transfer of a block of words , the processor sends,

- ➢ Starting address
- ➢ Number of words in the block
- ➢ Direction of transfer.

When a block of data is transferred , the DMA controller increment the memory address for successive words and keep track of number of words and it also informs the processor by raising an interrupt signal. While DMA control is taking place, the program requested the transfer cannot continue and the processor can be used to execute another program. After DMA transfer is completed, the processor returns to the program that requested the transfer.

SACET-CSE

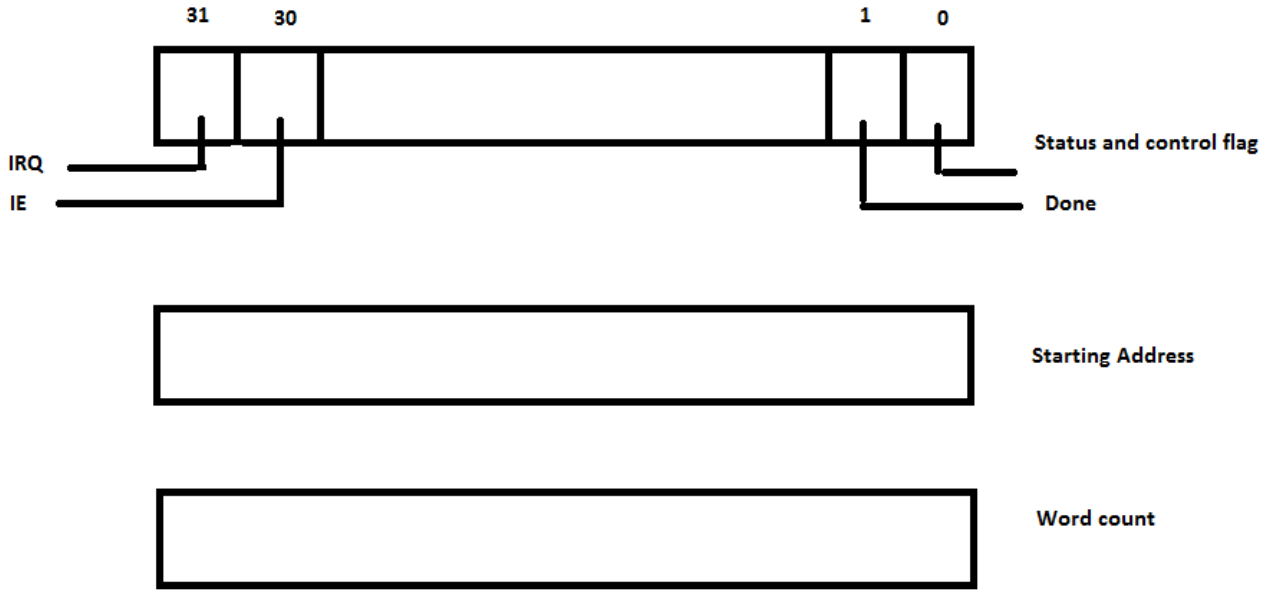


Fig:Registes in a DMA Interface

R/W → Determines the direction of transfer . When

R/W =1, DMA controller read data from memory to I/O device.

R/W =0, DMA controller perform write operation.

Done Flag=1, the controller has completed transferring a block of data and is ready to receive another command.

IE=1, it causes the controller to raise an interrupt (interrupt Enabled) after it has completed transferring the block of data.

IRQ=1, it indicates that the controller has requested an interrupt.

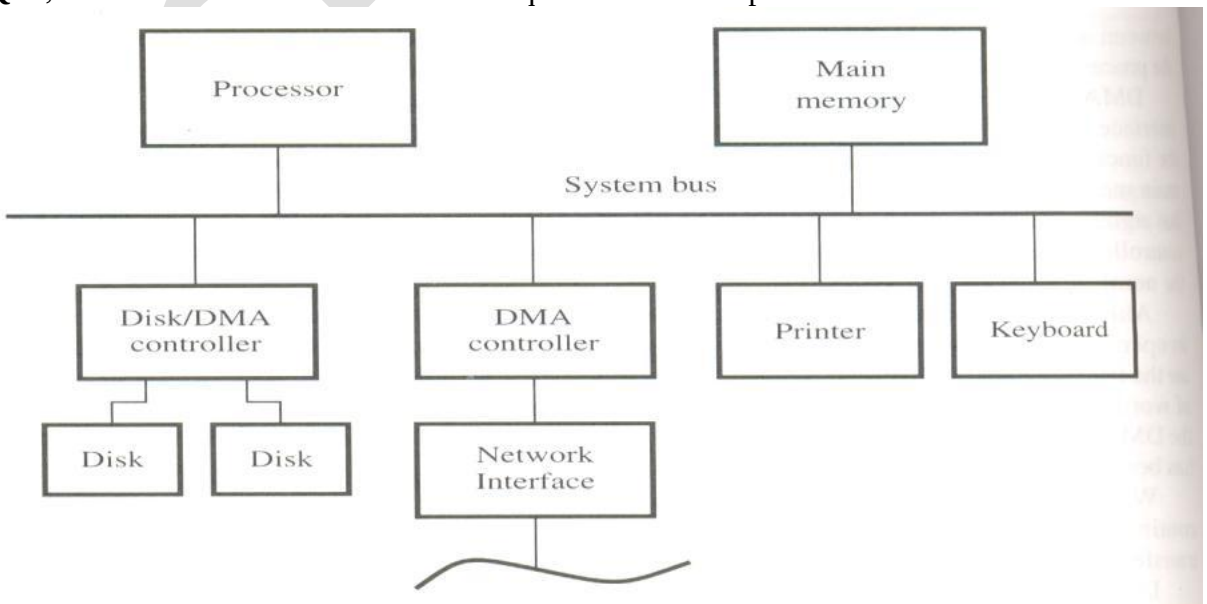


Fig: Use of DMA controllers in a computer system

- A DMA controller connects a high speed network to the computer bus . The disk controller with two disks, also has DMA capability and it provides two DMA channels.

- To start a DMA transfer of a block of data from main memory to one of the disks, the program write s the address and the word count information. Into the registers of the corresponding channel of the disk controller.
- When DMA transfer is completed, it will be recorded in status and control registers of the DMA channel (ie) **Done bit=IRQ=IE=1**.

Cycle Stealing:

- Requests by DMA devices for using the bus are having higher priority than processor requests . Top priority is given to high speed peripherals such as ,
  - Disk
  - High speed Network Interface and Graphics display device.
- Since the processor originates most memory access cycles, the DMA controller can be said to steal the memory cycles from the processor. This interviewing technique is called **Cycle stealing.**

Burst Mode:

- The DMA controller may be given exclusive access to the main memory to transfer a block of data without interruption. This is known as **Burst/Block Mode**

Bus Master:

- The device that is allowed to initiate data transfers on the bus at any given time is called the bus master.

Bus Arbitration:

- It is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it.

Types:

There are 2 approaches to bus arbitration. They are,

- Centralized arbitration (A single bus arbiter performs arbitration)
- Distributed arbitration (all devices participate in the selection of next bus master).

Centralized Arbitration:

Here the processor is the bus master and it may grants bus mastership to one of its DMA controller. A DMA controller indicates that it needs to become the bus master by activating the Bus Request line (BR) which is an open drain line. The signal on BR is the logical OR of the bus request from all devices connected to it. When BR is activated the processor activates the Bus Grant Signal (BGI) and indicated the DMA controller that they may use the bus when it becomes free. This signal is connected to all devices using a daisy chain arrangement. If DMA requests the bus, it blocks the propagation of Grant Signal to other devices and it indicates to all devices that it is using the bus by activating open collector line, Bus Busy (BBSY).

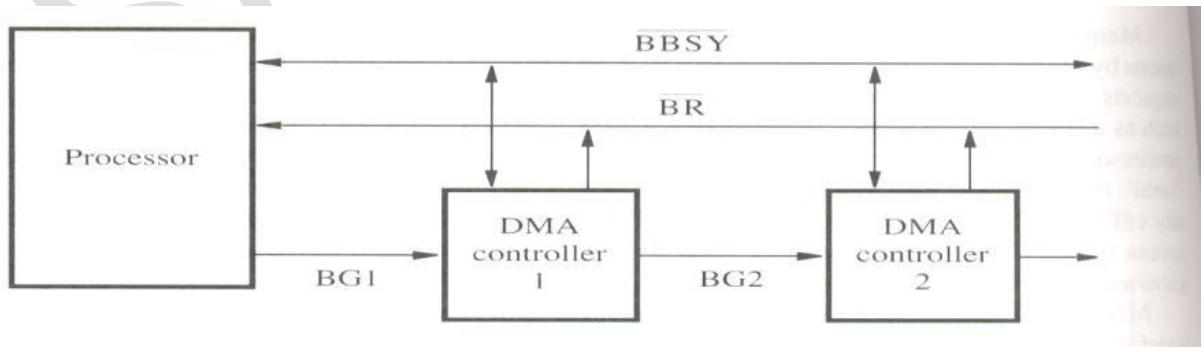


Fig:A simple arrangement for bus arbitration using a daisy chain

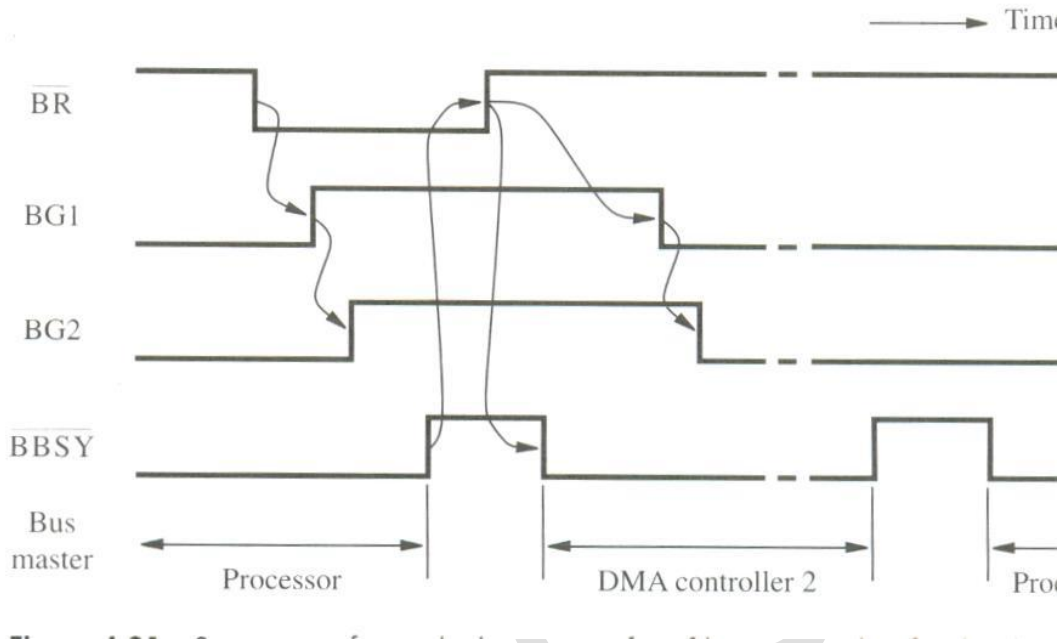


Fig: Sequence of signals during transfer of bus mastership for the devices

The timing diagram shows the sequence of events for the devices connected to the processor is shown. DMA controller 2 requests and acquires bus mastership and later releases the bus. During its tenture as bus master, it may perform one or more data transfer. After it releases the bus, the processor resources bus mastership.

Distributed Arbitration:

It means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process.

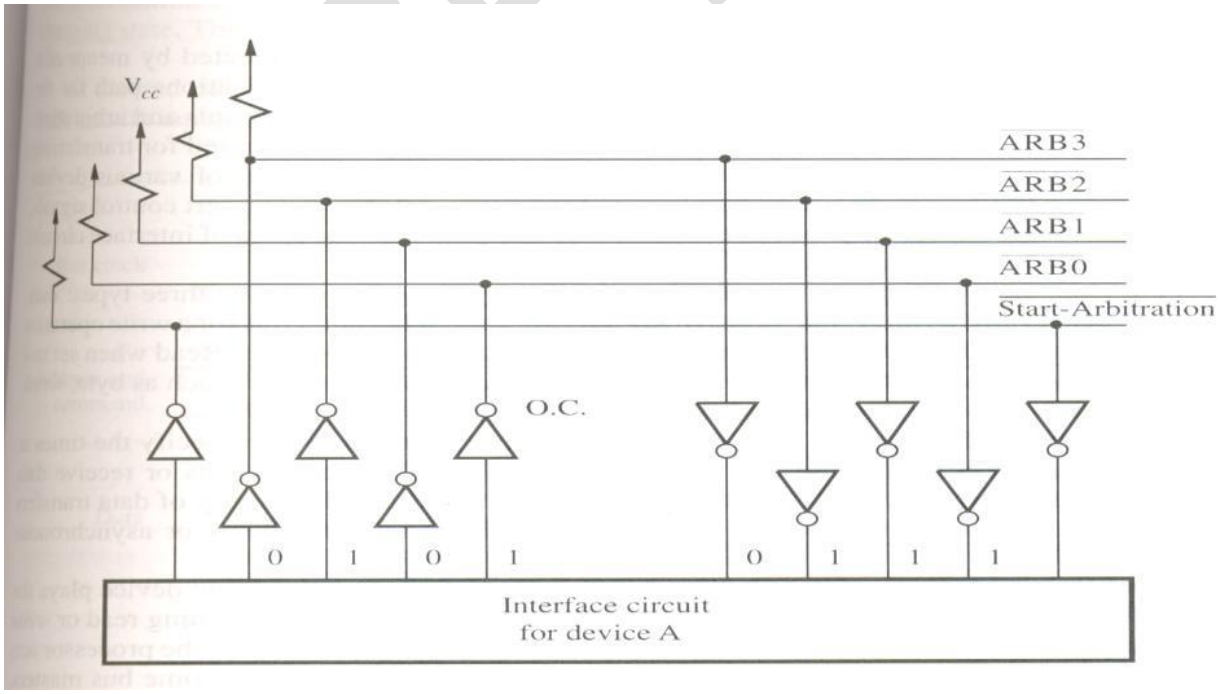


Fig:A distributed arbitration scheme

- Each device on the bus is assigned a 4 bit id.
- When one or more devices request the bus, they assert the Start-Arbitration signal & place their 4 bit ID number on four open collector lines, ARB0 to ARB3.
- A winner is selected as a result of the interaction among the signals transmitted over these lines.
- The net outcome is that the code on the four lines represents the request that has the highest ID number.
- The drivers are of open collector type. Hence, if the i/p to one driver is equal to 1, the i/p to another driver connected to the same bus line is equal to „0"(ie. bus the is in low-voltage state).

Example:

- Assume two devices A & B have their ID 5 (0101), 6(0110) and their code is 0111.
- Each devices compares the pattern on the arbitration line to its own ID starting from MSB.
- If it detects a difference at any bit position, it disables the drivers at that bit position. It does this by placing „0" at the i/p of these drivers.
- In our eg. „A" detects a difference in line ARB1, hence it disables the drivers on lines ARB1 & ARB0.
- This causes the pattern on the arbitration line to change to 0110 which means that „B" has won the contention.

4.7 Buses

A bus protocol is the set of rules that govern the behavior of various devices connected to the bus ie, when to place information in the bus, assert control signals etc. The bus lines used for transferring data is grouped into 3 types. They are,

- Address line
- Data line
- Control line.

Control signals:Specifies that whether read / write operation has to performed. It also carries timing information. (ie) they specify the time at which the processor & I/O devices place the data on the bus & receive the data from the bus.

- During data transfer operation, one device plays the role of a „**Master**".
- **Master** device initiates the data transfer by issuing read / write command on the bus. Hence it is also called as „**Initiator**".
- The device addressed by the master is called as **Slave / Target**

Types of Buses:

There are 2 types of buses. They are,

- Synchronous Bus
- Asynchronous Bus.

Synchronous Bus:-

- In synchronous bus, all devices derive timing information from a common clock line.
- Equally spaced pulses on this line define equal time.
- During a „**bus cycle**", one data transfer on take place.
- The „**crossing points**" indicate the tone at which the patterns change.
- A „**signal line'** in an indeterminate / high impedance state is represented by an intermediate half way between the low to high signal levels.

SACET-CSE

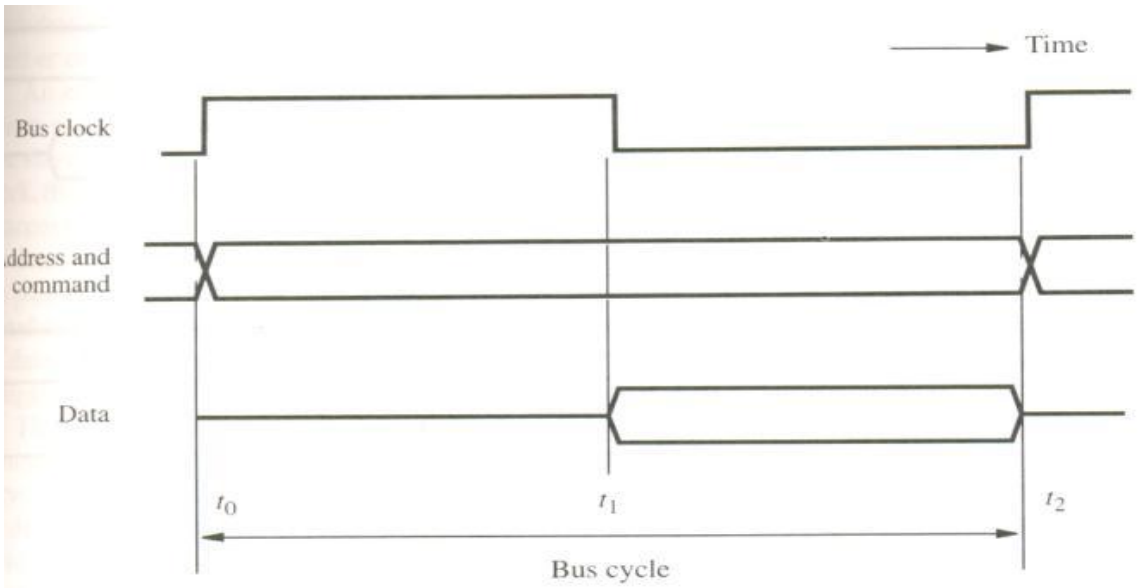


Fig:Timing of an input transfer of a synchronous bus.

- In this case, the command will indicate an input operation & specify the length of the operand to be read.
- The clock pulse width t1 – t0 must be longer than the maximum delay between devices connected to the bus.
- The clock pulse width should be long to allow the devices to decode the address & control signals so that the addressed device can respond at time t1.
- The slaves take no action or place any data on the bus before t1.

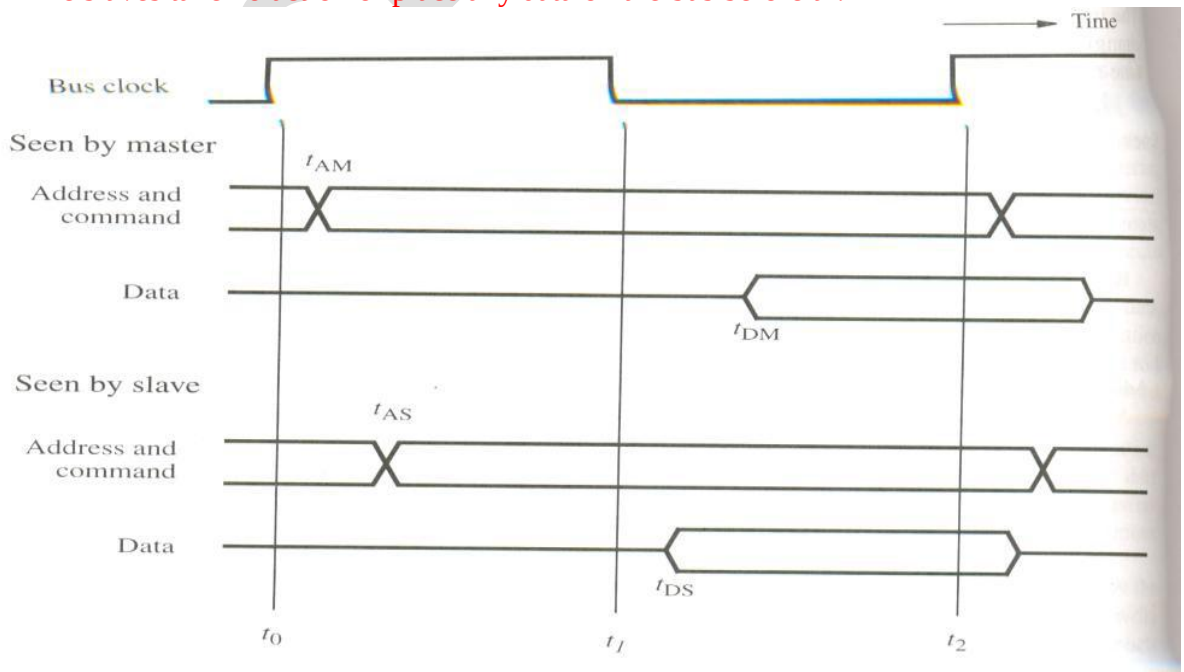


Fig:A detailed timing diagram for the input transfer

- The picture shows two views of the signal except the clock.

- One view shows the signal seen by the master & the other is seen by the salve.
- The master sends the address & command signals on the rising edge at the beginning of clock period (t0). These signals do not actually appear on the bus until tam.
- Some times later, at tAS the signals reach the slave.
- The slave decodes the address & at t1, it sends the requested data.
- At t2, the master loads the data into its i/p buffer.
- Hence the period t2, tDM is the setup time for the masters i/p buffer.
- The data must be continued to be valid after t2, for a period equal to the hold time of that buffers.

Demerits:

- The device does not respond.
- The error will not be detected

Multiple Cycle Transfer:-

- During, clock cycle1, the master sends address & cmd infn/. On the bus‟ requesting a „read‟ operation.
- The slave receives this information & decodes it.
- At the active edge of the clock (ie) the beginning of clock cycel2, it makes accession to respond immediately.
- The data become ready & are placed in the bus at clock cycle3.
- At the same times, the slave asserts a control signal called „slave-ready‟.
- The master which has been waiting for this signal, strobes, the data to its i/p buffer at the end of clock cycle3.
- The bus transfer operation is now complete & the master sends a new address to start a new transfer in clock cycle4.
- The „slave-ready‟ signal is an acknowledgement form the slave to the master confirming that valid data has been sent.

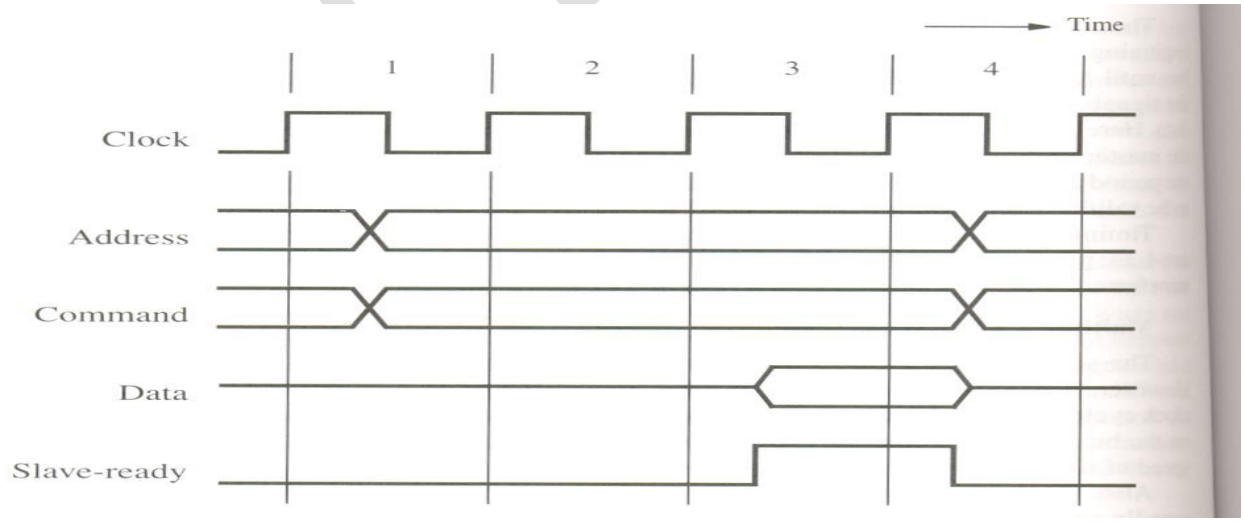
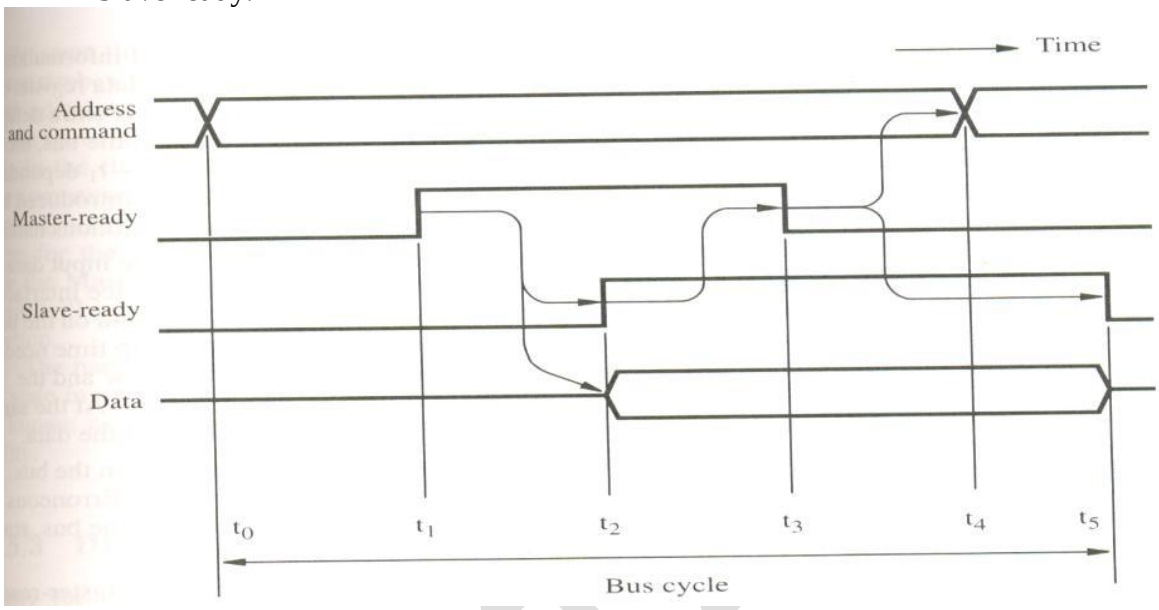


Fig:An input transfer using multiple clock cycles

Asynchronous Bus:-

An alternate scheme for controlling data transfer on. The bus is based on the use of "**handshake**" between **Master** & the **Slave**. The common clock is replaced by two timing control lines. They are

- Master–ready
- Slave ready.



The handshake protocol proceed as follows:

At **t0**→ The master places the address and command information on the bus and all devices on the bus begin to decode the information

At **t1** → The master sets the Master ready line to 1 to inform the I/O devices that the address and command information is ready.

- The delay t1 – t0 is intended to allow for any skew that may occurs on the bus.
- The skew occurs when two signals simultaneously transmitted from one source arrive at the destination at different time.
- Thus to guarantee that the Master ready signal does not arrive at any device a head of the address and command information the delay t1 – t0 should be larger than the maximum possible bus skew.

At **t_2** → The selected slave having decoded the address and command information performs the required i/p operation by placing the data from its data register on the data lines. At the same time, it sets the "slave – Ready" signal to 1.

At **t_3** → The slave ready signal arrives at the master indicating that the i/p data are available on the bus.

At **t_4** → The master removes the address and command information on the bus. The delay between t_3 and t_4 is again intended to allow for bus skew. Erroneous addressing may take place if the address, as seen by some device on the bus, starts to change while the master – ready signal is still equal to 1.

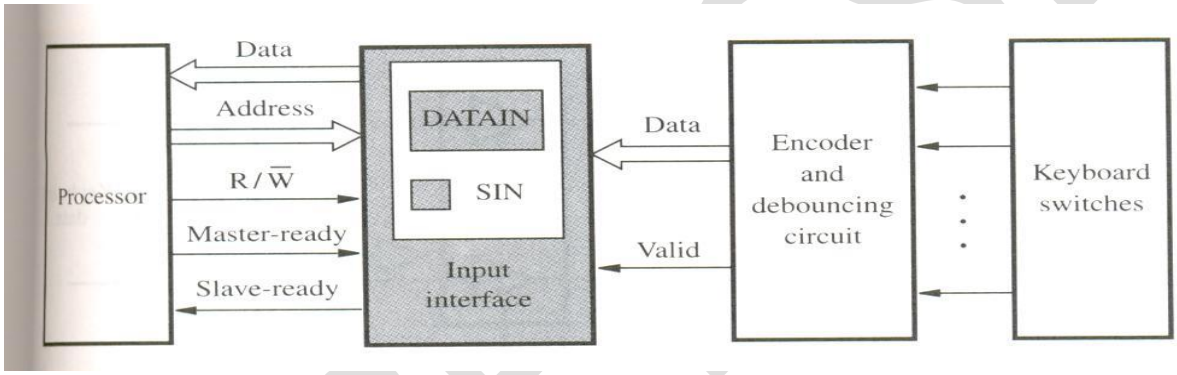
At **t_5** → When the device interface receives the 1 to 0 transitions of the Master – ready signal. It removes the data and the slave – ready signal from the bus. This completes the i/p transfer.

- In this diagram, the master place the output data on the data lines and at the same time it transmits the address and command information.
- The selected slave strobes the data to its o/p buffer when it receives the Master-ready signal and it indicates this by setting the slave – ready signal to 1.
- At time t0 to t1 and from t3 to t4, the Master compensates for bus.
- A change of state is one signal is followed by a change is the other signal. Hence this scheme is called as Full Handshake.
- It provides the higher degree of flexibility and reliability.

4.8 INTERFACE CIRCUITS

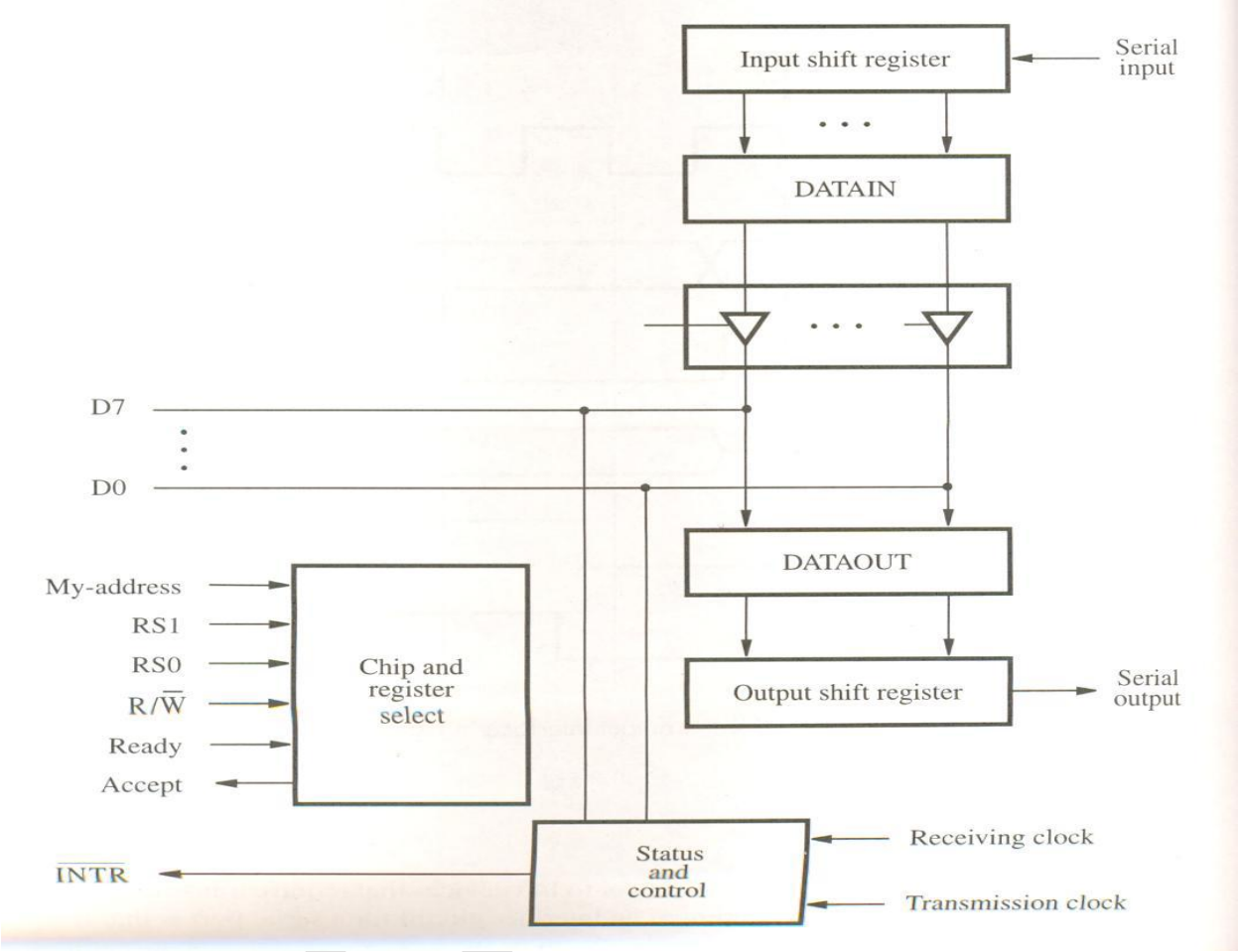
The interface circuits are of two types. They are

- Parallel Port
- Serial Port



- The output of the encoder consists of the bits that represent the encoded character and one signal called **vali**d, whi**c**h indicates the key is pressed.
- The information is sent to the interface circuits, which contains a data register, DATAIN and a status flag SIN.
- When a key is pressed, the Valid signal changes from 0 to1,causing the ASCII code to be loaded into DATAIN and SIN set to 1.
- The status flag SIN set to 0 when the processor reads the contents of the DATAIN register.
- The interface circuit is connected to the asynchronous bus on which transfers are controlled using the Handshake signals Master ready and Slave-ready.

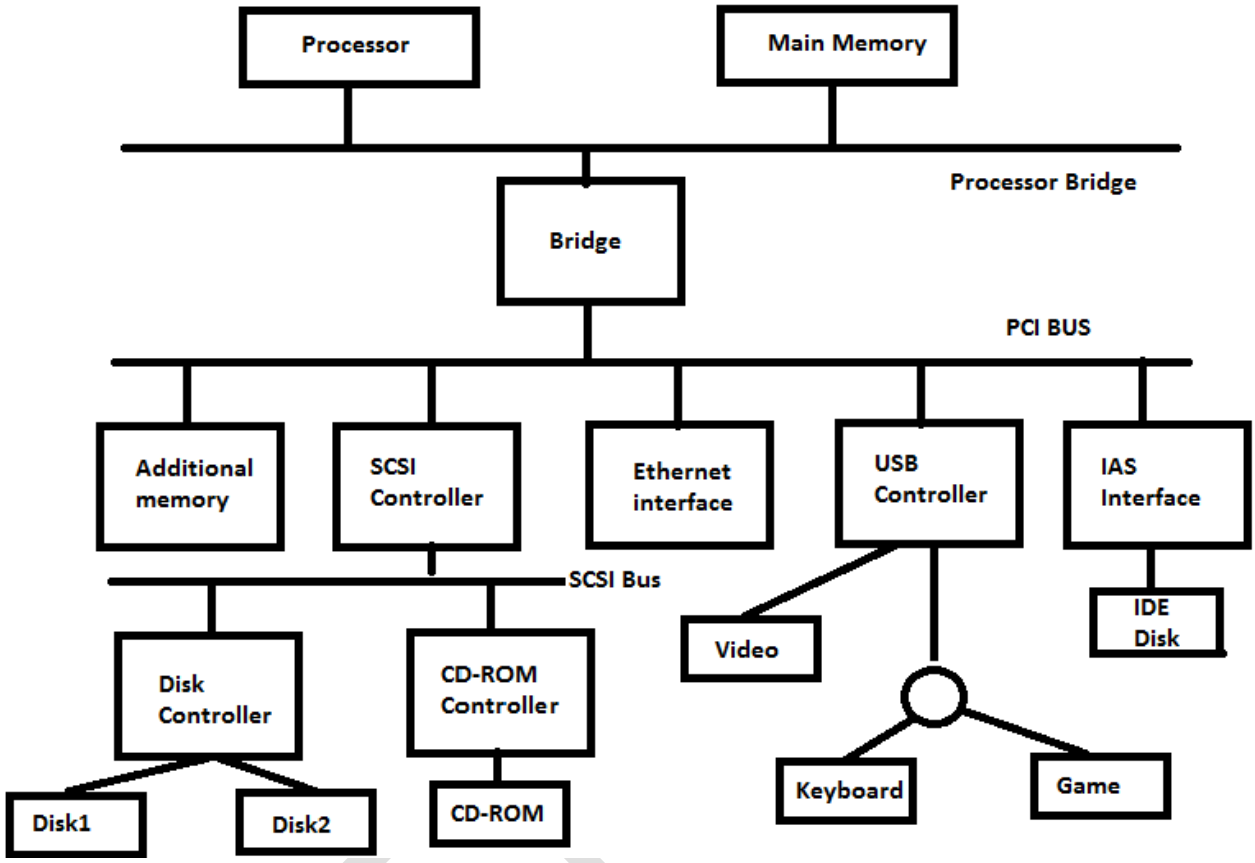
Serial Port:



A serial port used to connect the processor to I/O device that requires transmission one bit at a time. It is capable of communicating in a bit serial fashion on the device side and in a bit parallel fashion on the bus side.

4.9 STANDARD I/O INTERFACE

- A standard I/O Interface is required to fit the I/O device with an Interface circuit.
- The processor bus is the bus defined by the signals on the processor chip itself.
- The devices that require a very high speed connection to the processor such as the main memory, may be connected directly to this bus.
- The bridge connects two buses, which translates the signals and protocols of one bus into another.
- The bridge circuit introduces a small delay in data transfer between processor and the devices.



Example of a Computer System using different Interface Standards

USB – Universal Serial Bus

USB supports 3 speed of operation. They are,

- Low speed (1.5Mb/s)
- Full speed (12Mb/s)
- High speed (480Mb/s)

The USB has been designed to meet the key objectives. They are,

1. It provide a simple, low cost & easy to use interconnection structure that overcomes the difficulties due to the limited number of I/O ports available on a computer.
2. It accommodate a wide range of data transfer characteristics for I/O devices including telephone & Internet connections.
3. Enhance user convenience through **'Plug & Play'** mode of operation.

Merits of USB:-

USB helps to add many devices to a computer system at any time without opening the computer box.

Device Characteristics:-

The kinds of devices that may be connected to a computer cover a wide range of functionality. The speed, volume & timing constrains associated with data transfer to & from devices varies significantly.

Plug & Play

The main objective of USB is that it provides a plug & play capability. The plug & play feature enhances the connection of new device at any time, while the system is operation. The system should

- Detect the existence of the new device automatically.
- Identify the appropriate device driver software.
- Establish the appropriate addresses.
- Establish the logical connection for communication.

USB Architecture:-

1. USB has a serial bus format which satisfies the low-cost & flexibility requirements.
2. Clock & data information are encoded together & transmitted as a single signal.
3. There are no limitations on clock frequency or distance arising form data skew, & hence it is possible to provide a high data transfer bandwidth by using a high clock frequency.
4. To accommodate a large number of devices that can be added / removed at any time, the USB has the tree structure.

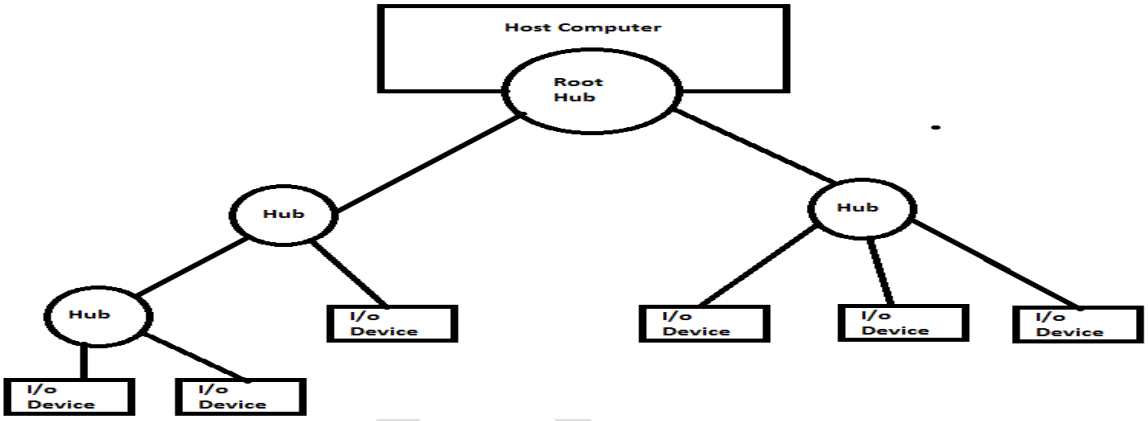


Fig:USB Tree Structure

Each node of the tree has a device called "**hub**", which acts as an intermediate control point between host & I/O devices. At the root of the tree, the "root hub" connects the entire tree to the host computer. The leaves of the tree are the I/O devices being served.

PCI

1. PCI is developed as a low cost bus that is truly processor independent.
2. It supports high speed disk, graphics and video devices.
3. PCI has plug and play capability for connecting I/O devices.
4. To connect new devices, the user simply connects the device interface board to the bus

Data Transfer

1. The data are transferred between cache and main memory is the bursts of several words and they are stored in successive memory locations.
2. When the processor specifies an address and request a "read" operation from memory, the memory responds by sending a sequence of data words starting at that address.
3. During write operation, the processor sends the address followed by sequence of data words to be written in successive memory locations.
4. PCI supports read and write operation.
5. A read / write operation involving a single word is treated as a burst of length one.

PCI has three address spaces. They are

- Memory address space
- I/O address space
- Configuration address space

I/O address space → It is intended for use with processor

Configuration space → It is intended to give PCI, its plug and play capability.

6. PCI Bridge provides a separate physical connection to main memory.
7. The master maintains the address information on the bus until data transfer is completed.
8. At any time, only one device acts as **bus master**.
9. A master is called "initiator" in PCI which is either processor or **DMA.**
10. The addressed device that responds to read and write commands is called a **target.**
11. A complete transfer operation on the bus, involving an address and bust of data is called a "**transaction'.**

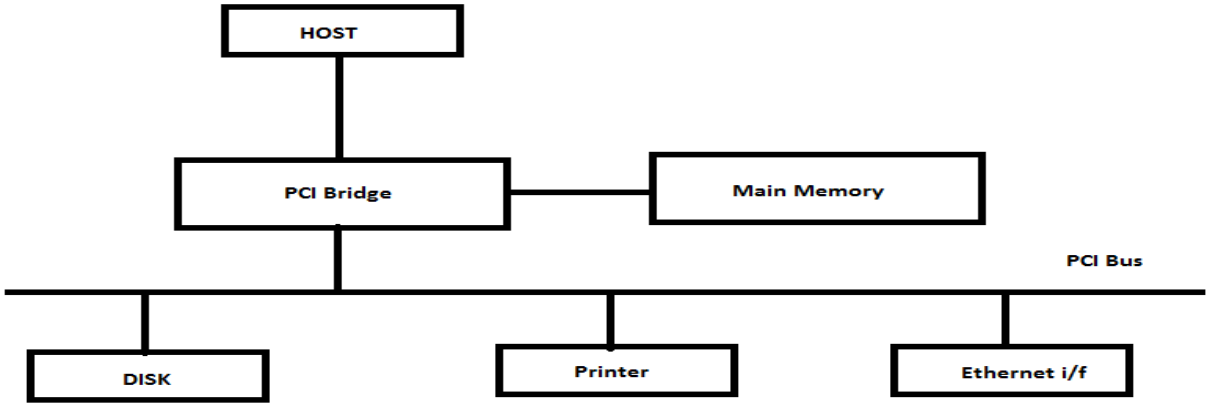


Fig:Use of a PCI bus in a Computer system

Data Transfer Signals on PCI Bus

Name Function

CLK → 33 MHZ / 66 MHZ clock

FRAME # → Sent by the indicator to indicate the duration of transaction

AD → 32 address / data line

C/BE # → 4 command / byte Enable Lines

IRDY, TRDYA→ Initiator Ready, Target Ready Signals

DEVSEL # → A response from the device indicating that it has recognized its address and is ready for data transfer transaction.

IDSEL # → Initialization Device Select

Individual word transfers are called "**phases'**.

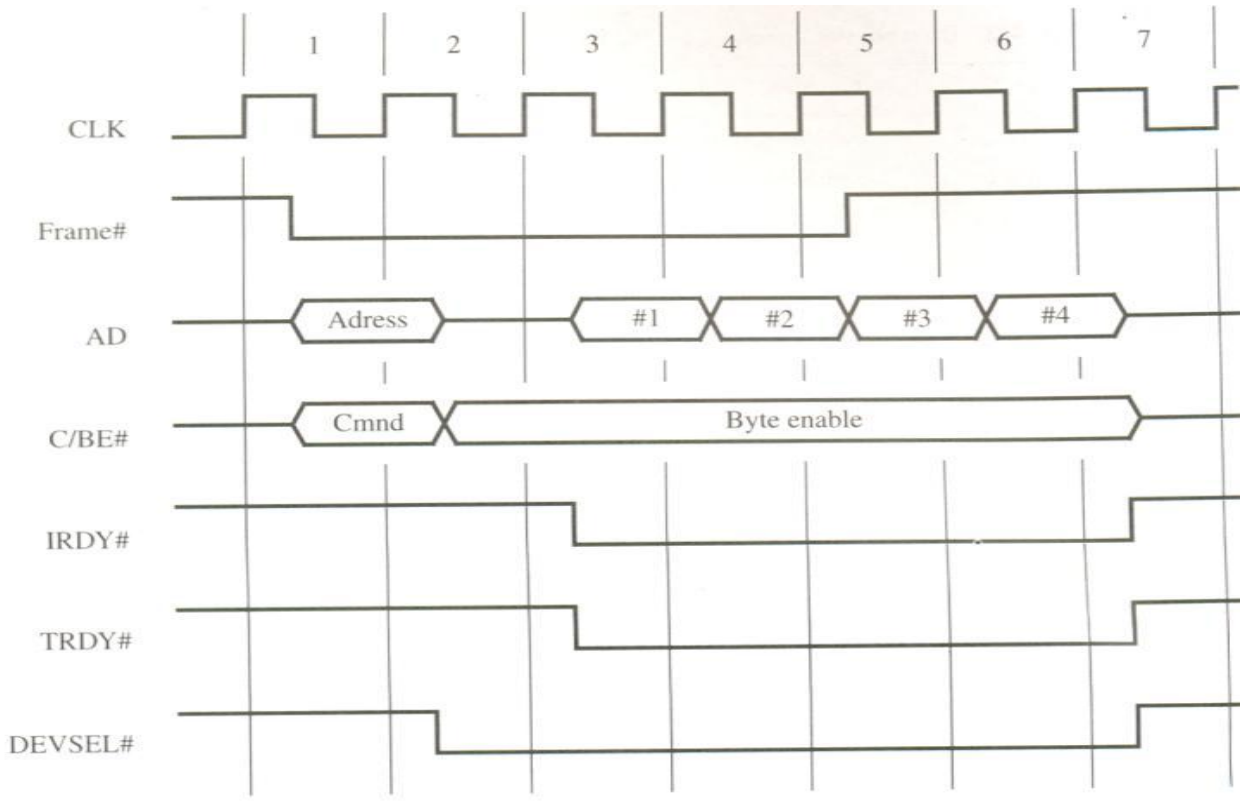


Fig :Read operation an PCI Bus

1. In Clock cycle1, the processor asserts FRAME # to indicate the beginning of a transaction ; it sends the address on AD lines and command on C/BE # Lines.
2. Clock cycle2 is used to turn the AD Bus lines around ; the processor ; The processor removes the address and disconnects its drives from AD lines.
3. The selected target enable its drivers on AD lines and fetches the requested data to be placed on the bus.
4. It asserts DEVSEL # and maintains it in asserted state until the end of the transaction.
5. C/BE # is used to send a bus command in clock cycle and it is used for different purpose during the rest of the transaction.
6. During clock cycle 3, the initiator asserts IRDY #, to indicate that it is ready to receive data.
7. If the target has data ready to send then it asserts TRDY #. In our eg, the target sends 3 more words of data in clock cycle 4 to 6.
8. The indicator uses FRAME # to indicate the duration of the burst, since it read 4 words, the initiator negates FRAME # during clock cycle 5.
9. After sending the 4th word, the target disconnects its drivers and negates DEVSEL # during clock cycle 7.

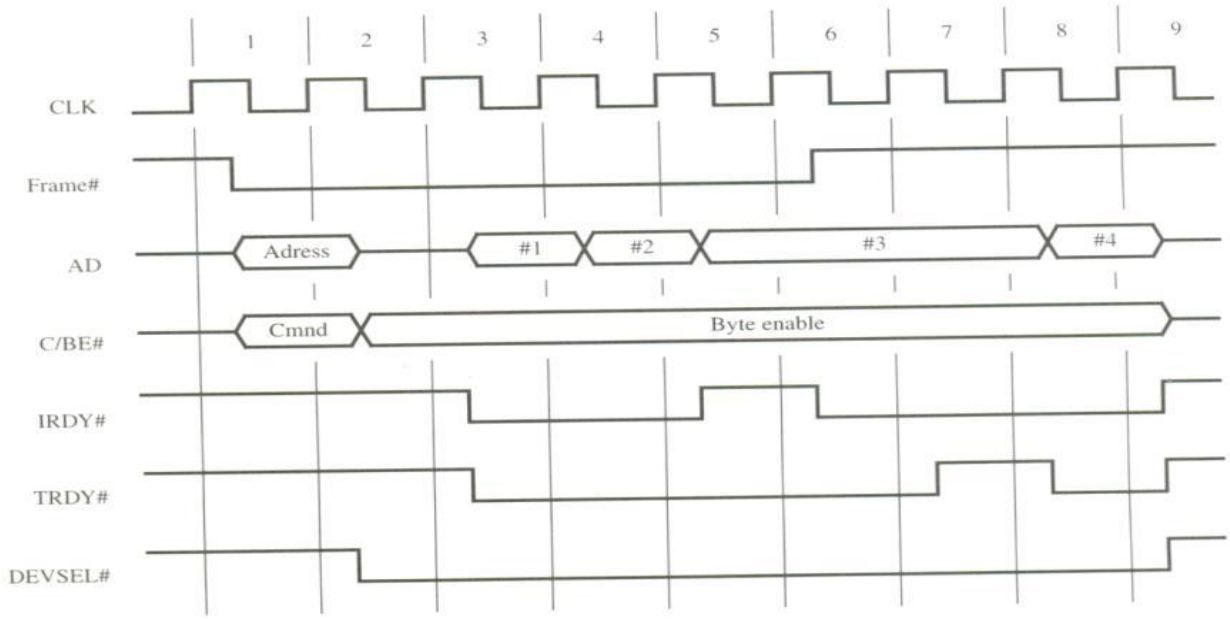


Fig: A read operation showing the role of IRDY# / TRY#

1. It indicates the pause in the middle of the transaction.
2. The first and words are transferred and the target sends the 3rd word in cycle 5.
3. But the indicator is not able to receive it. Hence it negates IRDY#.
4. In response the target maintains 3rd data on AD line until IRDY is asserted again.
5. In cycle 6, the indicator asserts IRDY. But the target is not ready to transfer the fourth word immediately, hence it negates TRDY in cycle 7. Hence it sends the 4th word and asserts TRDY# at cycle 8.

Device Configuration

1. The PCI has a configuration ROM memory that stores information about that device.
2. The configuration ROM‟s of all devices are accessible in the configuration address space.
3. The initialization s/w read these ROM‟s whenever the S/M is powered up or reset
4. In each case, it determines whether the device is a printer, keyboard, Ethernet interface or disk controller.
5. Devices are assigned address during initialization process and each device has an w/p signal called IDSEL # (Initialization device select) which has 21 address lines (AD) (AD to AD31).
6. During configuration operation, the address is applied to AD i/p of the device and the corresponding AD line is set to and all other lines are set to 0.