I/O module is the key element for computer system. I/O modules control the peripheral devices. Thus I/O module is required which has two major functions.

- Interface to the processor and memory via the system bus or ***central switch***.
- Interface to one or two peripheral devices by tailored data links

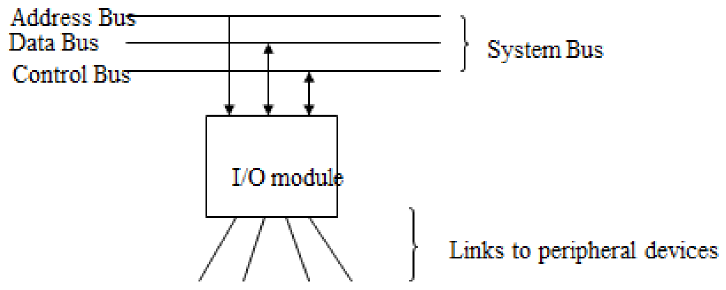

Fig. Generic model of an I/O module

External Devices

External devices provide the means of exchanging data between the external environment and the computer. An external device is attached to computer by a link to an I/O module. The link is used to exchange control, status and data between the I/O module and external device. External device is reffered to as peripheral device or peripheral. It is classified in three categories:

- Human – readable: suitable for communicating with computer user. E.g. Screen, Printer, Keyboard etc.
- Machine – readable: suitable for communicating with equipment. E.g. magnetic disk, tape, sensor, actuator etc.
- Communication: suitable for communicating with remote devices. E.g. Modem, NIC etc.

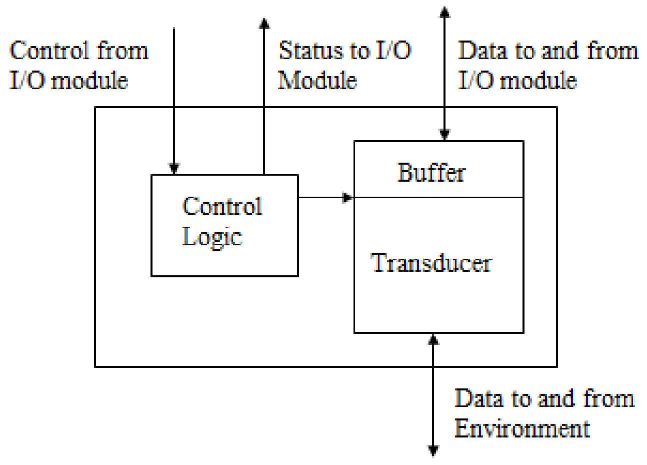

Fig: Block Diagram of an External Device

Interface of I/O device is in the form of –

— Control signals

— Status signal

— Data signal.

Data are in the form of a set of bits to be sent to or received from I/O module. ***Control signals*** determine the function that the device will perform. (e.g. Input the data, output the data, determine head position etc.) ***Status signals*** indicate the state of the device. (e.g. acknowledge signal). ***Control Logic*** controls device operation according to direction from I/O module. ***Transducer*** converts electrical signals into other forms of energy for output and other forms to electrical signals for input. ***Buffer*** hold the data temporarily for transducer or from transducer.

External devices are:

Keyboard/Monitor:

- It is the most common means of computer/user interaction. User provides input through keyboard and it is transmitted to computer and displayed on monitor.

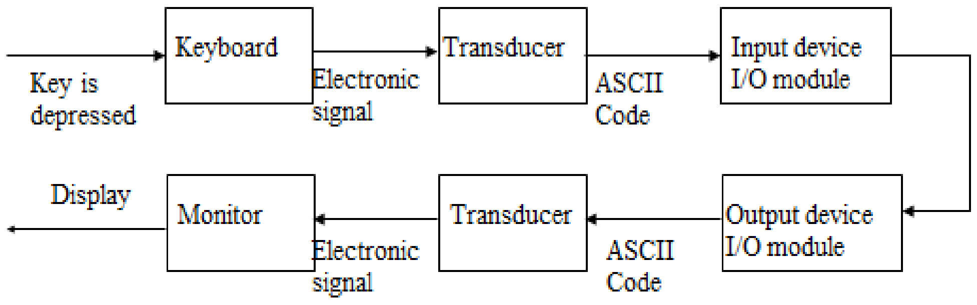

Fig: Working of Keyboard and monitor

Basic unit of exchange is character in the form of code, most common of which is 7-bit code, ASCII (American Standard Code for Information Interchange).Being 7-bit code, it can represent 128 unique characters. Two types of characters: printable and control characters.

Disk Drives

Contains electronics for exchanging data, control and status signals with an I/O module and electronics for containing disk read/write mechanism. Two types of disk:

1. Fixed head
2. Movable head

Fixed head disk transducer is capable of converting magnetic patterns on moving disk and bits in device buffer.

Moving head disk also because the disk arm to move in and out across disk surface.

I/O modules

- The five major functions of I/O module are:

— Control and timing

— Processor communication

— Device communication

— Data buffering

— Error Detection

In any period of time processor may communicate with one or more external devices depending upon need of program. The internal resources must be shared among number of activities, including data I/O. thus control and timing is required for I/O for the coordination between internal resources and external device. E.g. The control of the transfer of data from an external device to the processor might involve the following steps:

1. The processor interrogates the I/O module to check the status of the attached device.
2. The I/O module returns the device status.
3. If the device is ready to transmit, the processor requests the transfer of data, by means of command to the I/O module.
4. The I/O module obtains the unit of data from the external device.
5. The data are transferred from the I/O module to the processor.

Processor communication involves the following activities:

Command decoding – I/O module accepts command from processor sent as control signals. READ SECTOR, WRITE SECTOR, SEEK track number, SCAN record ID etc.

Data – Data exchange takes place between processor and I/O module through I/O module.

Status reporting – Peripherals are slow in operation. Thus status of I/O module should be reported. BUSY and READY signals.

Address recognition – Each I/O device has an address, so I/O module recognizes the unique address for each peripheral it controls.

Device communication involves the commands, status information and data.

Data transfer rate of main memory and processor is high relative to external device. Data from main memory are sent to I/O module and are buffered to send them in peripherals at its rate. I/O module must be able to operate at both device and memory speeds .I/O module is also responsible for error detection. It also should report the error to the processor. Error includes mechanical and electrical malfunctions reported by device (e.g. paper jam). Another error consists changes in pattern as it is transmitted form device to I/O module. Error detecting code also deals with transmission errors.

I/O Module Structure

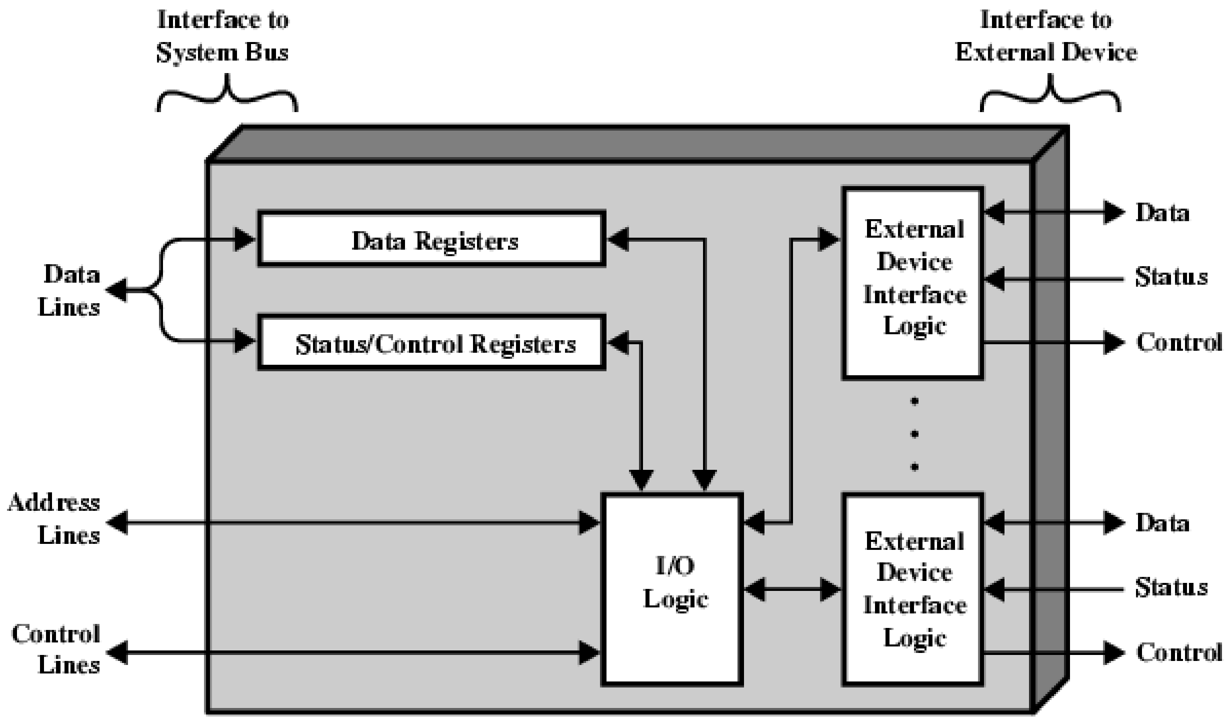

Fig: Block Diagram of I/O module

The I/O module connects the rest of the computers through the set of signal lines (**Data, address and control lines**). Data transferred to and from CPU are buffered at ***data registers***. ***Status register*** provides current status information and control register accept control information from CPU. ***I/O logic*** interacts with CPU through address and control lines. Control lines are used to issue commands to I/O e. The module must also recognize the unique address associated with devices it controls.

I/O module functions to allow the processor to view a wide range of devices. It hides the details of an external device so that a processor can function in terms of simple *read* and *write* commands.

An I/O module that takes on most of the detailed processing burden, presenting a high level interface to the processor, is usually referred to as an **I/O channel** or **I/O processor**. An I/O module that is primitive and requires detailed control is referred as I/O controller or device controller. I/O controllers are seen on microcomputers and I/O channels in mainframe.

Three techniques are possible for I/O operations.

— Programmed I/O

— Interrupt driven I/O

— Direct Memory Access (DMA)

In programmed I/O, the processor executes a program that gives it direct control of the I/O operation, including sensing I/O device status, sending a read or write command, and transferring the data. The execution of I/O related instructions are performed by issuing a command to the appropriate I/O module. I/O module performs the requested action and set the appropriate bits in the I/O status register. The processor periodically checks the status of the I/O module until it finds that the operation is complete.

I/O commands: For I/O related instruction processor issues an address specifying I/O module and an I/O command. Four types of I/O command is received:

— Control →It activates peripherals to perform required action. eg. Rewind, forward, pause in an magnetic disk etc.

— Test → Checks the status conditions associated with an I/O module and its peripherals. Processor checks for the available peripherals. It also checks the most recent operation is completed or not..

— Read → I/O module reads data from peripherals and place it into an internal buffer. The processor then obtains data by requesting I/O module to place it on data bus.

— Write → I/O module takes data from data bus and transmits the data to peripheral.

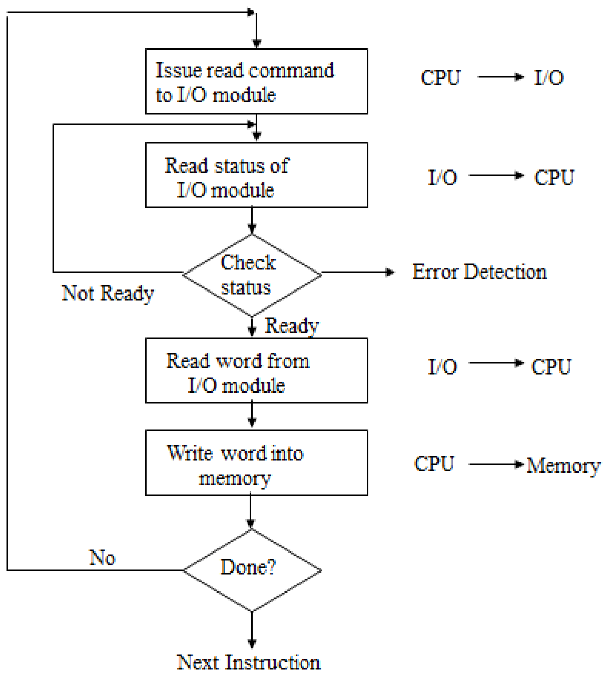

Fig. Programmed I/O operations.

Above programmed I/O reads data from peripheral into memory. Data are read in one word (16 bits) at a time. For each word that is read, the processor must check status the word is available in I/O module. Thus it makes the processor bupportes in any sy all the time.

I/O instructions

In programmed I/O there is a relation between the I/O related instructions that processor fetches from memory and I/O commands that processor issues to an I/O module to execute the instructions. Instructions are mapped into I/O commands that are they have one to one relationship.

There are many devices connected through I/O module to system. Each device has its own unique address. When processor issues command, the command contains the address of desired device. I/O module must interpret the address.

When processor, memory and I/O share a common bus, two modes of addressing are possible: Memory mapped and Isolated. Memory mapped I/O has a single address space for memory locations and I/O devices. The processor treats the status and data registers of I/O modules as memory locations and uses the same instructions to access both memory and I/O devices. E.g. for 10 address lines a combined total of 2^{10} =1024 memory locations and I/O address can be supported in any combination.

Memory-mapped I/O has a single read line and single write line on the bus. The bus may be equipped with memory read and write plus I/P and O/P command lines. Thus, command line specifies whether the address refers to memory location or an I/O device.

Isolated I/O supports for both I/O address space and memory address space i.e. for 10 address lines it has 1024 memory locations and 1024 I/O addresses. E.g. reads 1 byte of data from keyboard to accumulator of processor.

e.g. 516

7	6	5	4	3	2	1 0

Keyboard Input Register

517

7	6	5	4	3	2	1 0

Keyboard Input Status Control Register

Address	Instruction	Operand	Comment
200	Load AC	"1"	Load Accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	571	Get status byte
	Branch if sign=0	202	Loop until ready
	Load AC	516	Load data byte

Fig: Memory Mapped I/O

Address	Instruction	Operand	Comment
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch not ready	201	Loop until complete
	In	5	Load data byte

Fig: Isolated I/O

For 10-bit address 512(0-511) bit is used for memory and 512 for I/O(512-1023). Memory mapped can give large set of instructions but uses memory address space.

Interrupt Driven I/O

To reduce the time spent on I/O operations for periodically checking the status of I/O device in programmed I/O, the CPU can use an interrupt – driven I/O approach. For this

— CPU issues read command to I/O module.

— I/O module gets data from peripheral while CPU does other work.

— I/O module interrupts CPU to exchange data.

— CPU requests data.

— I/O module transfers data.

— CPU issues I/O command to the module.

CPU recognizes and responds to interrupt at the end of instruction execution cycle. Interrupt technique is used to support a wide variety of devices.

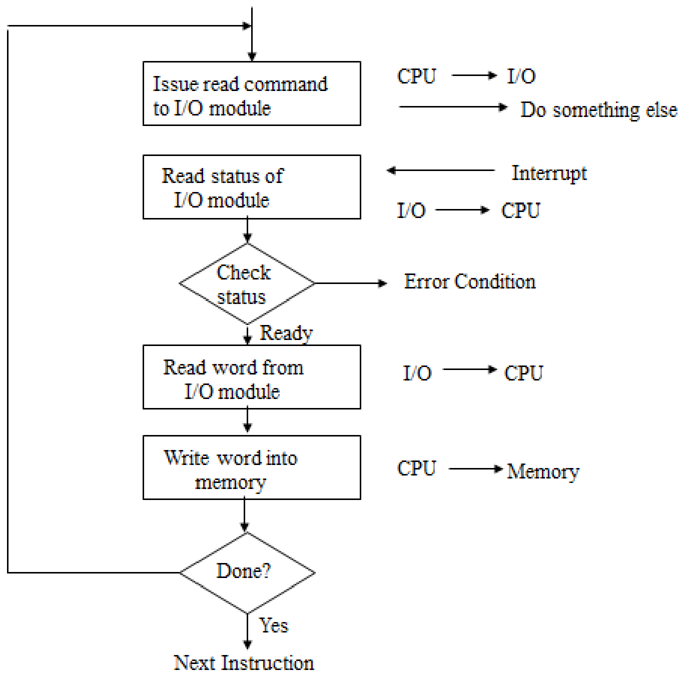

Fig. Interrupt driven I/O

Interrupt Processing:

The occurrence of interrupt triggers a number of events in processor hardware and software. Figure below shows the sequence:

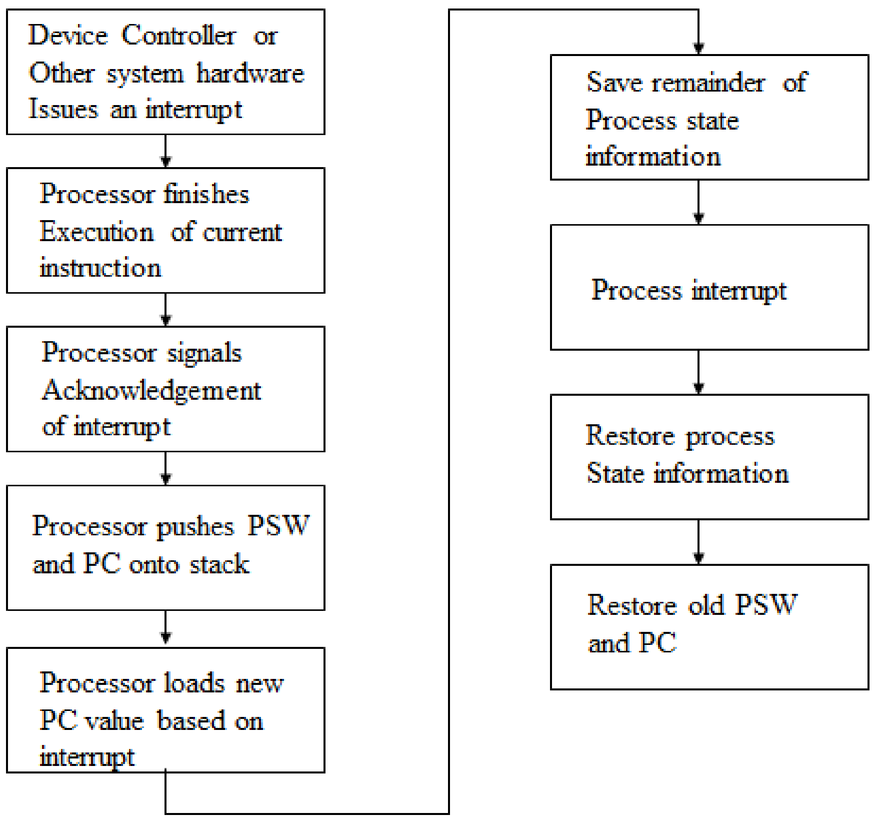
Device Controller or Other system hardware Issues an interrupt
Processor finishes Execution of current instruction
Processor signals Acknowledgement of interrupt
Processor pushes PSW and PC onto stack
Processor loads new PC value based on interrupt
Save remainder of Process state information
Process interrupt
Restore process State information
Restore old PSW and PC

Fig: Simple Interrupt Processing

1. Device issues an interrupt signal to the processor.
2. Processor first finishes the current execution and then responds to the interrupt.
3. Processor tests for interrupt; if there is one it sends an acknowledgement signal to device. This allows device to remove its interrupt signal.
4. The processor now processes the interrupt routine. For this purpose it saves the information of current program. Status of processor is saved in PSW ant the location of next instruction in PC. These are then pushed to control stack.
5. Processor now loads the PC with entry location of interrupt-handling program. If there is more than one interrupt-handling routine processor must decide which one to invoke. Once PC is loaded, processor proceeds to the next instruction cycle, which begins with instruction fetch. Instruction fetch is determined by PC, thus control is transferred to interrupt handler program. The execution results in following operation.
6. Here, the PC and PSW relating to the interrupted program have been saved to system stack. The content of the processor needs to be saved as it may be used by interrupt handler. Interrupt handler will begin by saving the contents of all of registers on the stack.
7. The interrupt handler next processes the interrupt. This includes an examination of status information relating to the I/O operation or other event that causes interrupt.
8. When interrupt processor is complete, the saved registers value are retrieved from stack and restored to registers.
9. Finally PSW and PC values are stored.

Design Issues

Two design issues arise in implementing interrupt I/O –

1. How does the processor determine which device issued the interrupt?

2. If multiple interrupts have occurred, how does the processor decide which one to process?

For device identification there is four technique:

1. *Multiple interrupt lines* – Use multiple interrupt lines through interrupt controller (8259C), but impractical.
2. *Software polling* – When the processor detects an interrupt, it branches to an interrupt-service routine whose job is to poll each I/O module to determine which module caused the interrupt. The poll could be in the form of command line .e.g. TEST I/O. Processor issues TEST I/O and places the address of I/O module on address line. The I/O module responds it set the interrupt. I/O module contains an addressable status register. The processor reads the status of I/O module. On identification of correct module, the processor branches to device service routine of specific device. Its main disadvantage is it is time consuming.

3. *Daisy chain (Hardware polling, vectored)* – All I/O module share a common interrupt request line. Interrupt request line is daisy chained through the modules. When the processor senses an interrupt, it sends out an interrupt acknowledge. The requesting module responds by placing a word on the data line. Word is a vector that may be I/O module address or other unique identifier. This is also known as vectored interrupt.

4. *Bus arbitration (vectored)* – An I/O module gets the control of bus before it raises the interrupt request line. Thus only one module can raise interrupt at a time. When the processor detects interrupt, it responds on interrupt acknowledge line.

With multiple lines, the processor picks the interrupt line with the highest priority. With software polling, the order in which modules are polled determines their priority. Same process is for hardware polling. Bus arbitration employs a priority scheme by using bus controller or bus arbitrator.

Direct Memory Access

Drawbacks of programmed and interrupt-driven I/O are:

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

To overcome these drawbacks, the DMA is used to transfer the data between memory and I/O device with less involvement of CPU. Large amounts of data can be transferred between memory and peripheral, severely impacting CPU performance. DMA involves an additional module on the system bus. DMA takes the control of system from processor. When processor wishes to read and write a block of data, it issues a command to DMA module by sending following information:

– Read/Write request

– Device address

- Starting address of memory block for data read/write
- Amount of data to be transferred

CPU carries on with other work. DMA controller deals with transfer i.e. transfer takes place between DMA and memory without involvement of processor. DMA controller sends interrupt when finished. CPU finalizes the DMA operations and resumes. Thus processor involvement is only at the beginning and end of transfer.

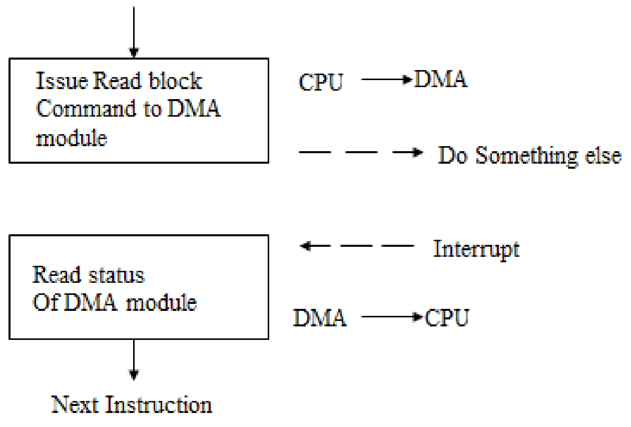
Issue Read block Command to DMA module
CPU → DMA
Do Something else
Read status Of DMA module
Interrupt
DMA → CPU
Next Instruction

Fig: Direct Memory Access

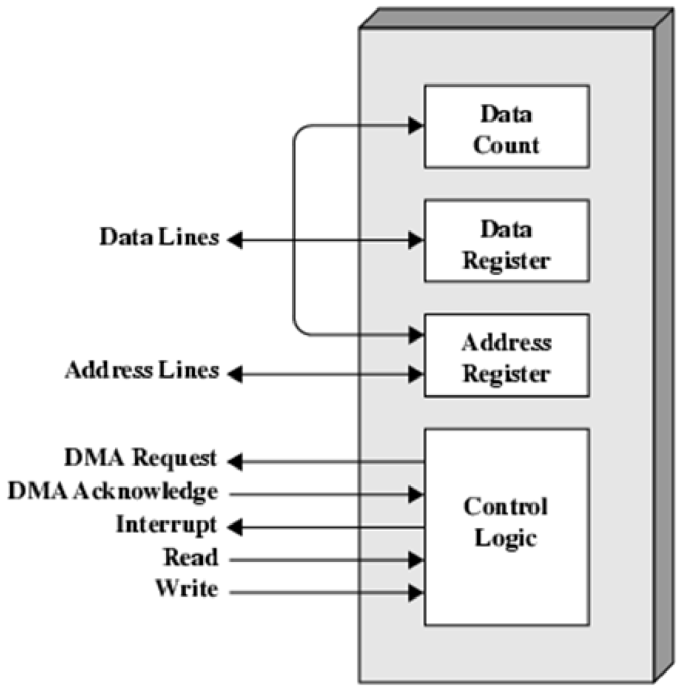

Fig: DMA Block Diagram

DMA modes of data transfer

1. Burst Mode- CPU gives the initialization of no. of bits to DMA as instruction. DMA takes control of system bus. With burst mode, data transfer takes place continuously till all the data are transferred. This mode is needed for fast devices such as magnetic disks

where data transmission cannot be stopped or slowed down. It makes CPU idle for long period.

2. Cycle Stealing Mode- DMA controller transfers one byte of data then returns control of system buses to CPU. CPU performs an instruction, and then DMA controller transfers a data value. DMA steals the cycle of CPU.

3. Transparent Mode- DMA controller transfers data only when the CPU is performing operations that don't use system buses. Thus CPU is not idle in this case. But if CPU is using bus all the time, DMA could not take the control of system bus. This mode requires hardware to detect the state of CPU.

CPU and IOP Communication

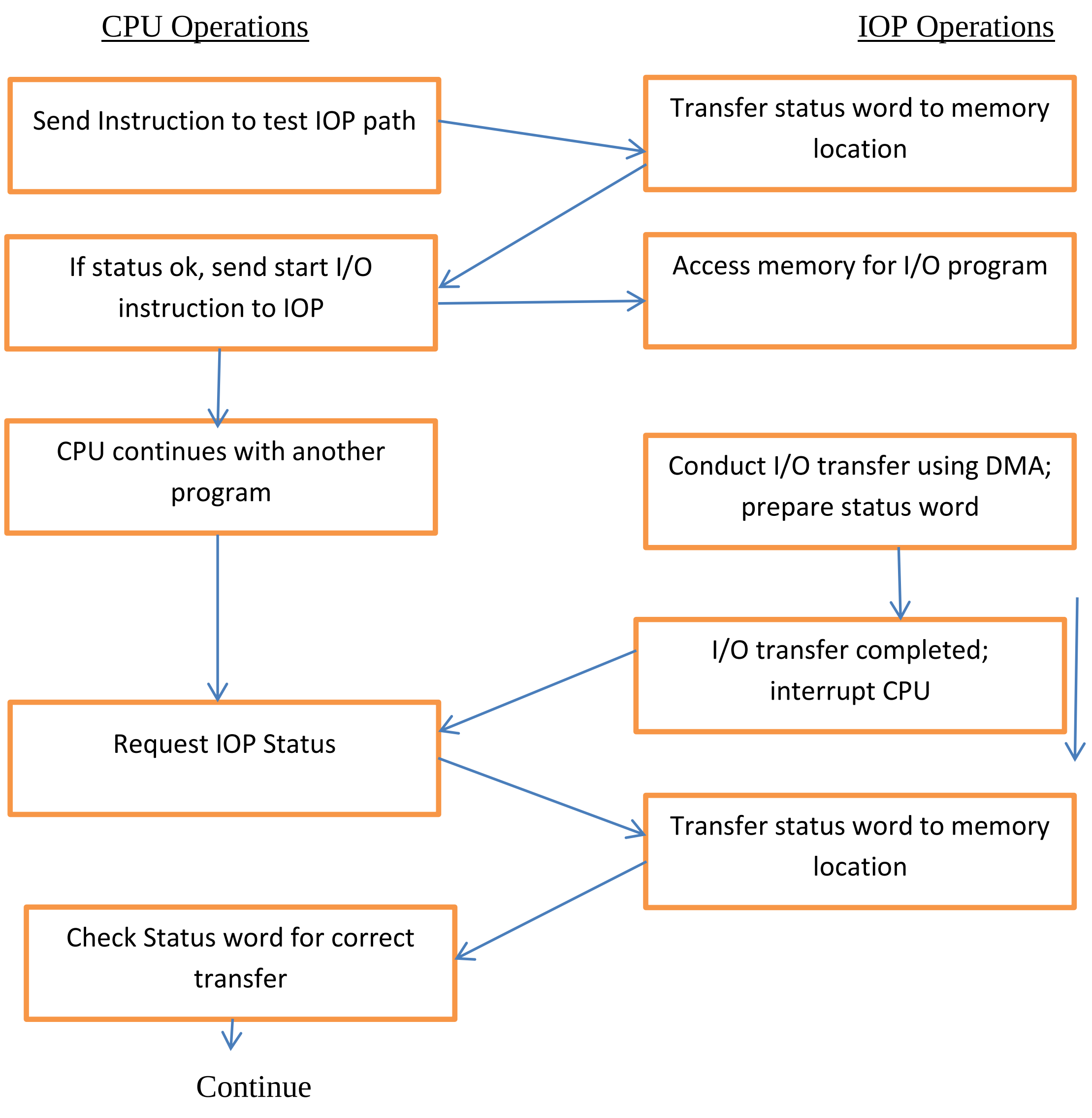
CPU Operations
IOP Operations
Send Instruction to test IOP path
Transfer status word to memory location
If status ok, send start I/O instruction to IOP
Access memory for I/O program
CPU continues with another program
Conduct I/O transfer using DMA; prepare status word
I/O transfer completed; interrupt CPU
Request IOP Status
Transfer status word to memory location
Check Status word for correct transfer
Continue

Fig: CPU-IOP Communication

I/O Channels and Processors:

I/O channels are the module that behaves like a processor with specialized instruction set. CPU directs the processor to execute a I/O program in memory. I/O processor fetches an executes instruction without any involvement of CPU. CPU is interrupted only when I/O activities are completed.

For I/O processors, module has its own local memory. With this large set of I/O devices can be controlled with minimal involvement of CPU. It controls the communication with interactive terminals.

Characteristics of I/O channels

I/O channel executes I/O instructions that completely control I/O operation. CPU does not executes I/O instructions instead all instructions are stored in main memory that can be executed by I/O processor. CPU only initiates I/O channels to perform transfer. The instruction specify device or devices, area or areas of memory storage, priority and actions to be taken.

Two types of I/O channels:

1. Selector- A selector controls multiple high-speed devices but one at a time. I/O channel selects one device and data transfer takes place.
2. Multiplexer-A multiplexer channel can handle multiple I/O devices at same time. For high-speed device, a block multiplexer interleaves block of data from several devices.

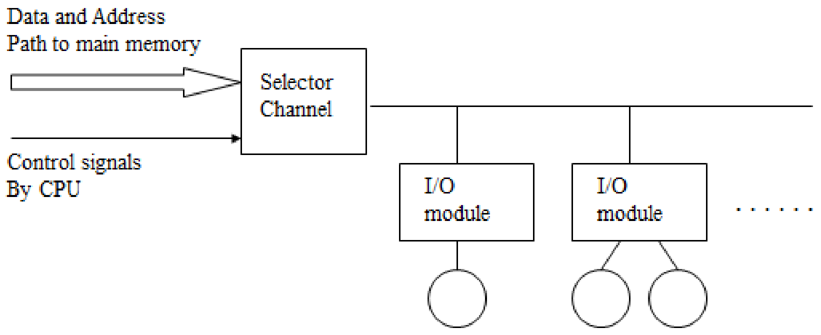

Fig. Selector Channel Architecture

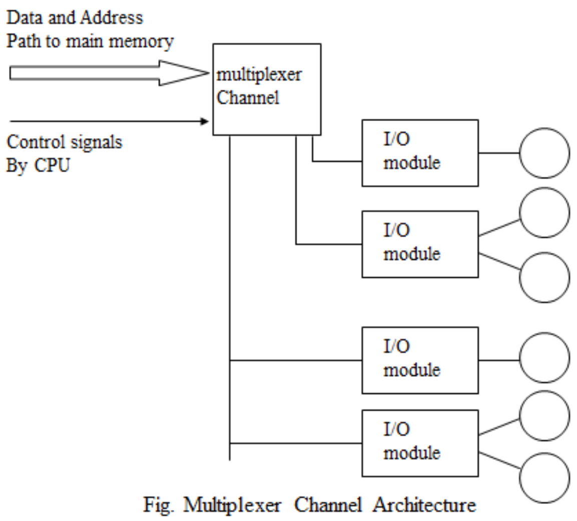

Fig. Multiplexer Channel Architecture

The External Interface

Two types of I/O module configurations –

1. *Point-to-point configuration* – provides the dedicated line between the I/O module and the external device. (keyboard)

2. *Multipoint configuration* – use the external buses (like PCI) and is used to support external mass storage device (disk and tape drives) and multimedia devices (CD ROMs, audio, video etc.).

Two types of external buses –

1. Fire Wire Serial Bus
2. InfiniBand

Fire Wire Serial Bus- Fire wire interface has advantages over older I/O interfaces. It is very high speed, low cost and easy to implement. Fire wire supports other electronic devices also. Fire wire uses a serial transmission rather than parallel. Fire wire provides a single I/O interface with a simple connector that can handle numerous devices through a single port so that mouse, printer, external disk drives, sound and LAN hookups can be replaced with single connector. Parallel interface requires shielding, compatibility, large set of wires, large area and expensive.

Fire wire Configurations:

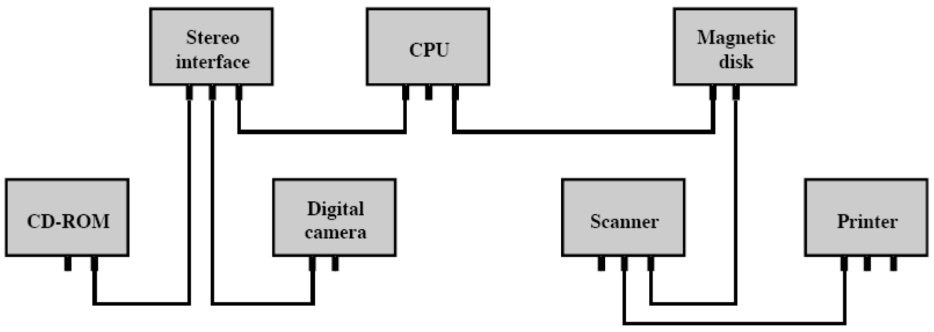

Figure 7.17 Simple FireWire Configuration

It uses a Daisy-Chain configuration with up to 63 devises connected off a single port. 1022 fire wire buses can be interconnected using bridges to support many peripherals.

Fire wire provides hot plugging, that is it can connect or disconnect peripherals without having power supply or reconfiguring the system. It also provides automatic configuration i.e not necessary to set ID’s manually. It should not be strictly daisy chained; it also can be tree like structure. Fire wire provides a three layer protocol:

- Physical →Transmission medium, electrical and signaling characteristics
- Link → Transmission of data in packets
- Transaction →Request-response protocol

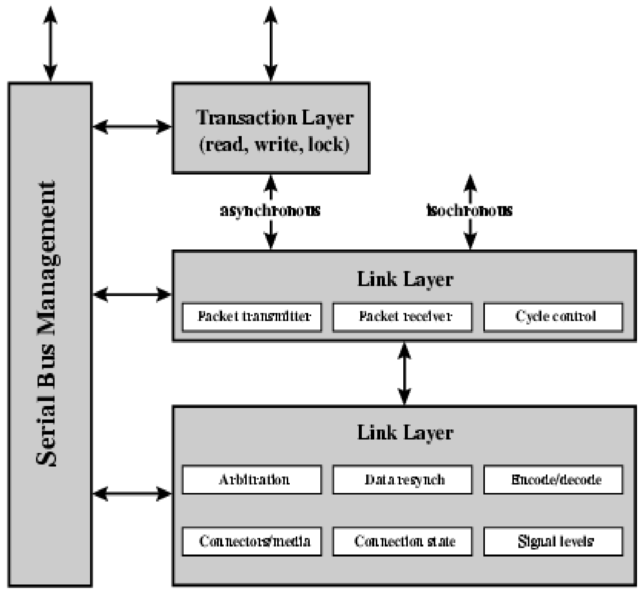

Fig: Fire wire Protocol Stack

Physical Layer

Physical layer specifies several transmission media and their connectors with different physical and data transmission properties. Data rates from 25 to 400Mbps. It converts binary data into electric signals. It also provides two forms of arbitration.

- Fair arbitration
- Urgent arbitration

Link Layer

This layer defines transmission of data in the form of packets. Two transmission types

- Asynchronous- Variable amount of data and several bytes of transaction data are transferred as a packet to an explicit address and then acknowledgement is returned
- Isochronous- Variable amount of data in sequence of fixed size packets are transmitted at regular intervals with no acknowledgement.

InfiniBand

I/O specification designed for high end servers.Version 1 was released in early 2001.It is the architecture and specification for data flow between processor and intelligent I/O devices. Infiniband are intended to replace PCI servers increasing capacity, expandability, and flexibility.

InfiniBand Architecture

Infiniband removes the necessity of basic I/O interface hardware. This allows greater server density and allows for a flexible and scalable data center as independent nodes may be added as needed. Infiniband channel enbales I/O devices to be placed. 17m away from server using copper, 300m using multimode optical fibre and up to 10 km using single mode optical fibre. The key element of this architecture are:

Host Channel Adapter (HCA)- HCA links the server to an infiniband switch. HCA is attached to memory controller that has access on system bus and controls traffic between processor and memory and between HCA and memory. It uses DMA to read and write memory.

Target Channel Adapter (TCA)- TCA connect storage systems routers and other peripheral devices to Infiniband.

Infiniband Switch- A switch provides point to point physical connections to a variety of devices and switches traffic from one link to another. The switch is used for communication between servers and devices through adapters. They manages the linkage without interrupting server's operation.

Links- Link between switch and channel adapter or between two switches.

Subnet- Consists of one or more interconnected switches plus links that connect other devices to switches. Subnet allows administrators to confine broadcast and multicast transmission within the subnet.

Routers- Connects subnets or switch to a network.

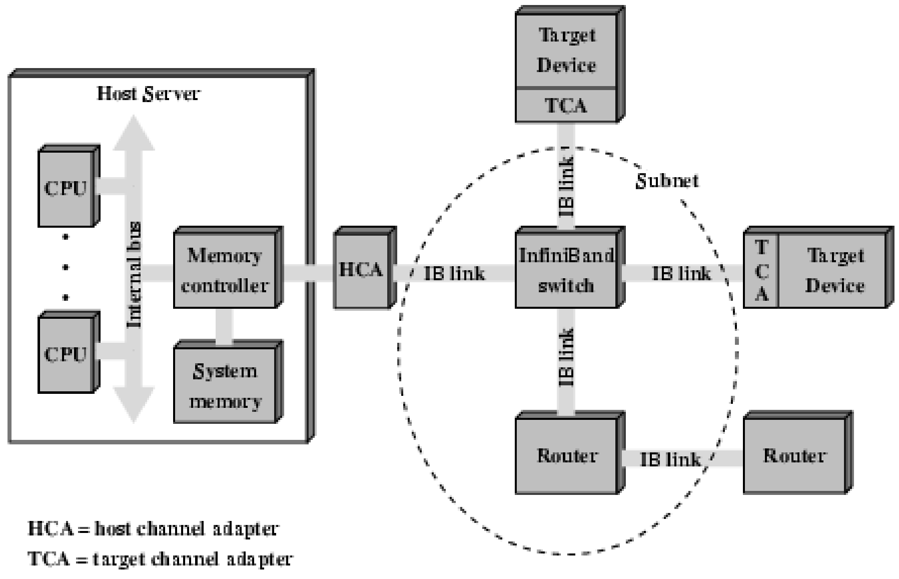

Fig: infiniband switch