Unit 7
Input Output Organization

I/O plays a crucial role in any modern computer system. Therefore, a clear understanding and appreciation of the fundamentals of I/O operations, devices, and interfaces are of great importance.

I/O subsystem

The input-output subsystem of a computer, referred as I/O, provides an efficient mode of communication between the central system and outside environment. Data and programs must be entered into the computer memory for processing and result of computations must be must be recorded or displayed for the user.

Peripheral devices

Input or output devices attached to the computer are called *peripherals*. Keyboards, display units and printers are most common peripheral devices. Magnetic disks, tapes are also peripherals which provide auxiliary storage for the system.

Input Devices

- Keyboard and mouse
- Touch screen
- Light pen
- Auxiliary storage
- Card reader
- Optical and magnetic character readers
- Data acquisition equipments

Output Devices

- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Digital incremental Plotters
- Auxiliary storage

Not all input comes from people and not all intended for people. In various real time processes as machine tooling, assembly line procedures and chemical & industrial processes, various processes communicate with each other providing input and/or outputs to other processes.

- I/O organization of a computer is a function of size of the computer and the devices connected to it. In other words, amount of hardware computer possesses to communicate with no. of peripheral units, differentiate between small and large system.
- IO devices communicating with people and computer usually transfer alphanumeric information using ASCII binary encoding.

Input-Output Interface

Input-output interface provides a method for transferring information between internal storage and external I/O devices. It resolves the *differences* between the computer and peripheral devices. The major differences are:

- Peripherals are **electromechanical and electromagnetic** devices and manner of operation is different from that of CPU which is **electronic** component.
- **Data transfer rate** of peripherals is slower than that of CPU. So some synchronization mechanism may be needed.
- **Data codes and formats** in peripherals differ from the word format in CPU and memory.
- **Operating modes of peripherals** are different from each other and each must be controlled so as not to disturb other.

To resolve these differences, computer system usually include special hardware unit between CPU and peripherals to supervise and synchronize I/O transfers, which are called **Interface units** since they interface processor bus and peripherals.

I/O Bus and Interface Modules

Peripherals connected to a computer need special communication link to interface with CPU. This special link is called **I/O bus**. Fig below clears the idea:

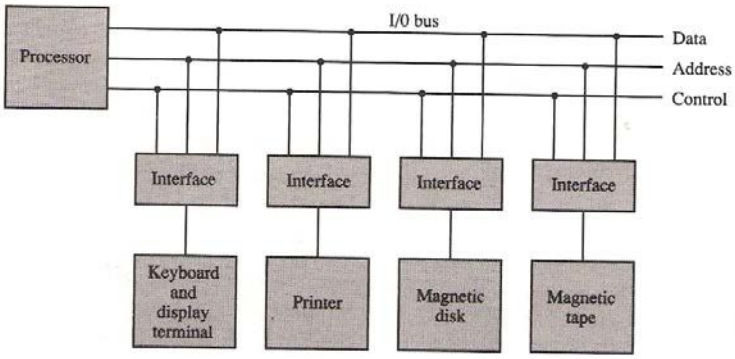


Fig: Connection of I/O bus to I/O devices

- I/O bus from the processor is attached to all peripheral interfaces.
- I/O bus consists of **Data lines**, **address** and **control lines**.
- To communicate with a particular device, the processor places a device address on the address lines. Each peripheral has an **interface module** associated with its interface.

Functions of an interface are as below:

- Decodes the device address (device code)
- Decodes the I/O commands (operation or function code) in control lines.
- Provides signals for the peripheral controller
- Synchronizes the data flow
- Supervises the transfer rate between peripheral and CPU or Memory

I/O commands

The function code provided by processor in control line is called *I/O command*. The interpretation of command depends on the peripheral that the processor is addressing. There are **4 types** of commands that an interface may receive:

a) Control command: Issued to activate the peripheral and to inform it what to do? E.g. a magnetic tape unit may be instructed to backspace tape by one record.
b) Status command: Used to check the various status conditions of the interface before a transfer is initiated.
c) Data input command: Causes the interface to read the data from the peripheral and places it into the interface buffer. [**HEY!** Processor checks if data are available using status command and then issues a data input command. The interface places the data on data lines, where they are accepted by the processor]

d) Data output command: Causes the interface to read the data from the bus and saves it into the interface buffer.

I/O Bus versus Memory Bus

In addition to communicating with I/O, processor also has to work with memory unit. Like I/O bus, memory bus contains data, address and read/write control lines. 3 physical organizations, the computer buses can be used to communicate with memory and I/O:

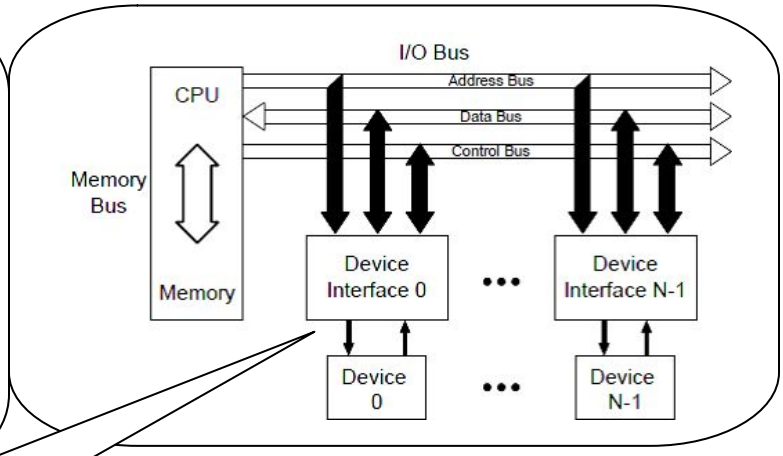
a) Use two separate buses, one for memory and other for I/O: Computer has independent sets of data, address and control buses, one for accessing memory and other for I/O. usually employed in a computer that has separate IOP (Input Output Processor).
b) Use one common bus for both memory and I/O having separate control lines
c) Use one common bus for memory and I/O with common control lines

Isolated I/O versus Memory-Mapped I/O

Question: *Differentiate between isolated I/O and memory-mapped I/O.*

Isolated I/O Configuration

- Two functionally and physically separate buses
  - Memory bus
  - I/O bus
- Each consists of the same three main groupings of wires
  - Address (example might be 6 wires, or up to 2^6 = 64 devices)
  - Data
  - Control

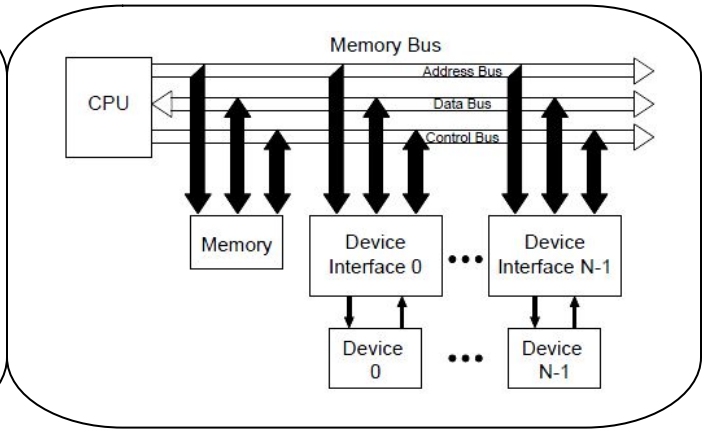


- Each device interface has a port which consists of a minimum of:
  - Control register
  - Status register
  - Data-in (Read) register (or buffer)
  - Data-out (Write) register (or buffer)

- CPU can execute instructions to manipulate the I/O bus separate from the memory bus
- Now only used in very high performance systems

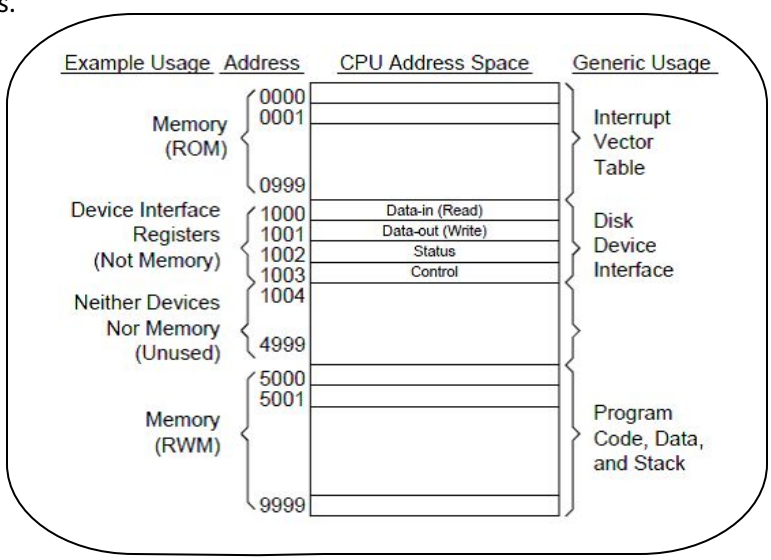
Memory-Mapped I/O configuration

- The memory bus is the *only* bus in the system
- Device interfaces assigned to the *address space* of the CPU or processing element
- Most common way of interfacing devices to computer systems
- CPU can manipulate I/O data residing in interface registers with same instructions that are used to access memory words.
- Typically, a segment of total address space is reserved for interface registers.



In this case, Memory space is not only ordinary system memory. It can refer to all the addresses that the programmer may specify. These addresses correspond to all the possible valid addresses that the CPU may place on its memory bus address lines.

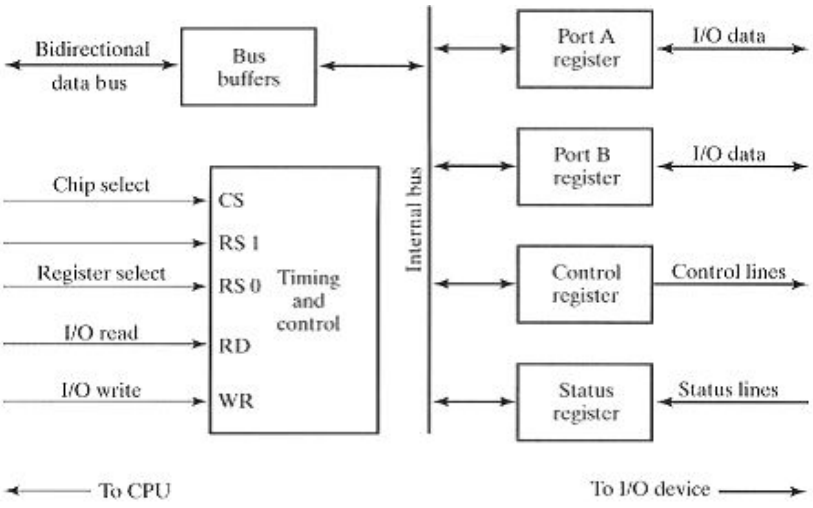
- Diagram shows a hypothetical example of a 10,000 byte memory space
  - Shows the principal regions of the memory Space of a computer system
- Random Access Memory (RAM) includes both:
  →ROM: Read-only memory (0000-0999)
  →RWM: Read-write memory (5000-9999)
- Unused memory space
  →No devices connected to these addresses
  →If CPU tries to access, causes a hardware or *bus* error



I/O interface Unit (an example)

I/O interface unit is shown in the block diagram below, it consists:

- Two data registers called *ports*.
- A control register
- A status register
- Bus buffers
- Timing and control circuits



- Interface communicates with CPU through data bus
- Chip select (CS) and Register select (RS) inputs determine the address assigned to the interface.
- I/O read and write are two control lines that specify input and output.
- 4 registers directly communicates with the I/O device attached to the interface.

Fig: I/O interface unit

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

- Address bus selects the interface unit through CS and RS1 & RS0.
- Particular interface is selected by the circuit (decoder) enabling CS.
- RS1 and RS0 select one of 4 registers.

Modes of I/O transfer (Types of I/O)

Binary information received from an external device is usually stored in memory for later processing. CPU merely executes I/O instructions and may accept data from memory unit (which in fact is ultimate source or destination). Data transfer between the central computer and I/O devices may be handled in one 3 modes:

- Programmed I/O
- Interrupt-initiated I/O
- Direct memory access (DMA)

Programmed I/O

Programmed I/O operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant monitoring of the peripheral by the CPU. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. It is up to the **programmed instructions** executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device. In programmed I/O method, I/O device does not have direct access to memory. Transfer from peripheral to memory/ CPU requires the execution of several I/O instructions by CPU.

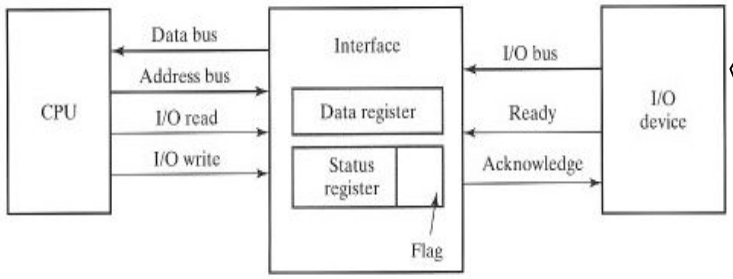
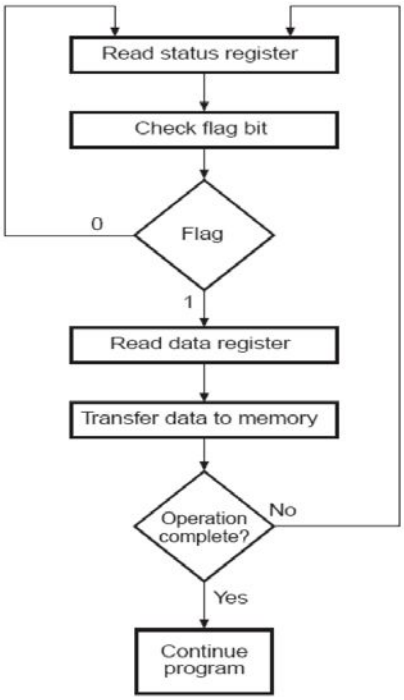


Diagram shows data transfer from I/O device to CPU. Device transfers bytes of data one at a time as they are available. When a byte of data is available, the device places it in the I/O bus and enables its data valid line. The interface accepts the byte into its data register and enables the data accepted line. The interface sets a bit in the status register that we will refer to as an F or "flag" bit.

Now for programmed I/O, a program is written for the computer to check the flag bit to determine if I/O device has put byte of data in data register of interface.



Flowchart of the program that must be written to the CPU is shown here. The transfer of each byte (assuming device is sending sequence of bytes) requires 3 instructions:

a) Read status register
b) Check F bit. If not set branch to a) and if set branch to c).
c) Read data register

Interrupt-initiated I/O

Since polling (constantly monitoring the flag F) takes valuable CPU time, alternative for CPU is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the **interrupt facility**. While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set. The CPU deviates from what it is doing to take care of the input or output transfer. After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.

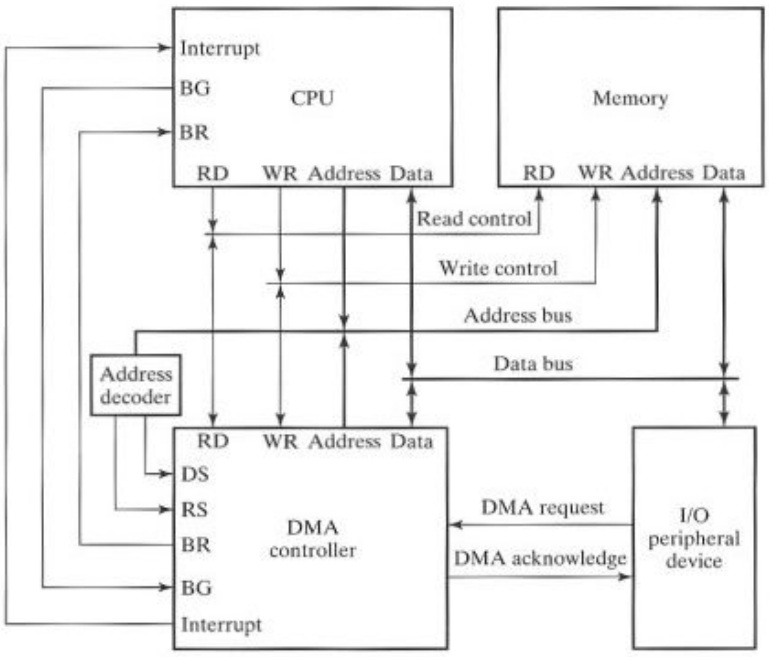
The CPU **responds to the interrupt signal** by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.

Direct Memory Access (DMA)

- **What is DMA? -** DMA is a sophisticated I/O technique in which a DMA controller replaces the CPU and takes care of the access of both, the I/O device and the memory, for fast data transfers. Using DMA you get the fastest data transfer rates possible.
- Momentum behind DMA: Interrupt driven and programmed I/O require active CPU intervention (All data must pass through CPU). Transfer rate is limited by processor's ability to service the device and hence CPU is tied up managing I/O transfer. Removing CPU form the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.
- Extensively used method to **capture buses** is through special control signals:
    - Bus request (BR): used by DMA controller to request the CPU for buses. When this input is active, CPU terminates the execution the current instruction and places the address bus, data bus and read & write lines into a high impedance state (open circuit).
    - Bus grant (BG): CPU activates BG output to inform DMA that buses are available (in high impedance state). DMA now take control over buses to conduct memory transfers without processor intervention. When DMA terminates the transfer, it disables the BR line and CPU disables BG and returns to normal operation.
- When DMA takes control of bus system, the **transfer with the memory** can be made in following two ways:
    - Burst transfer: A block sequence consisting of a number of memory words is transferred in continuous burst. Needed for fast devices as magnetic disks where data transmission can not be stopped (or slowed down) until whole block is transferred.
    - Cycle stealing: This allows DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow DMA to "steal" one memory cycle.

DMA Transfer

Question*: what is DMA transfer? Explain.*



- CPU communicates with the DMA through address and data buses.
- DMA has its own address which activates RS (Register select) and DS (DMA select) lines.
- When a peripheral device sends a DMA request, the DMA controller activates the BR line, informing CPU to leave buses. The CPU responds with its BG line.
- DMA then puts current value of its address register into the address bus, initiates RD or WR signal, and sends a DMA acknowledge to the peripheral devices.
- When BG=0, RD & WR allow CPU to communicate with internal DMA registers and when BG=1, DMA communicates with RAM through RD & WR lines.

Fig: DMA transfer in a computer system

Input-Output Processor (IOP)

- IOP is a **processor** with direct memory access capability that communicates with I/O devices. In this configuration, the computer system can be divided into a memory unit, and a number of processors comprised of CPU and one or more IOPs.
- IOP is similar to CPU except that it is designed to handle the details of **I/O processing**.
- Unlike DMA controller (which is set up completely by the CPU), IOP can fetch and execute its own instructions. IOP instructions are designed specifically to facilitate I/O transfers.
- Instructions that are read form memory by an IOP are called **commands** to differ them form instructions read by CPU. The *command words* constitute the program for the IOP. The CPU informs the IOP where to find commands in memory when it is time to execute the I/O program.

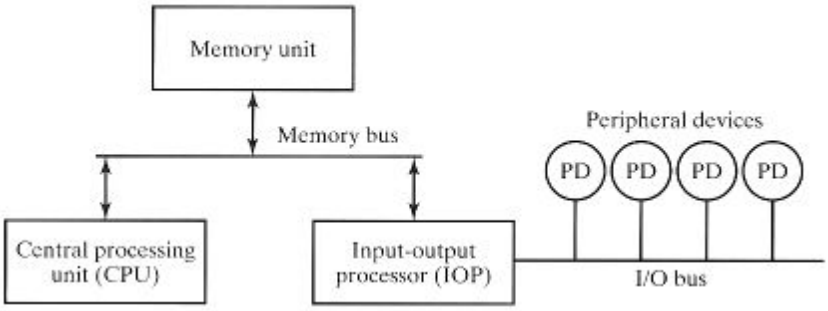


Fig: Block diagram of computer with I/O processor

The memory occupies a central position and can communicate with each processor by means of DMA. CPU is usually assigned the task of initiating the I/O program, from then on; IOP operates independent of the CPU and continues to transfer data from external devices and memory.

CPU-IOP communication

Communication between the CPU and IOP may take different forms depending on the particular computer used. Mostly, memory unit acts as a memory center where each processor leaves information for the other.

Mechanism: CPU sends an instruction to test the IOP path. The IOP responds by inserting a status word in memory for the CPU to check. The bits of the status word indicate the condition of IOP and I/O device ("IOP overload condition", "device busy with another transfer" etc). CPU then checks status word to decide what to do next. If all is in order, CPU sends the instruction to start the I/O transfer. The memory address received with this instruction tells the IOP where to find its program. CPU may continue with another program while the IOP is busy with the I/O program. When IOP terminates the transfer (using DMA), it sends an interrupt request to CPU. The CPU responds by issuing an instruction to read the status from the IOP and IOP then answers by placing the status report into specified memory location. By inspecting the bits in the status word, CPU determines whether the I/O operation was completed satisfactorily and the process is repeated again.

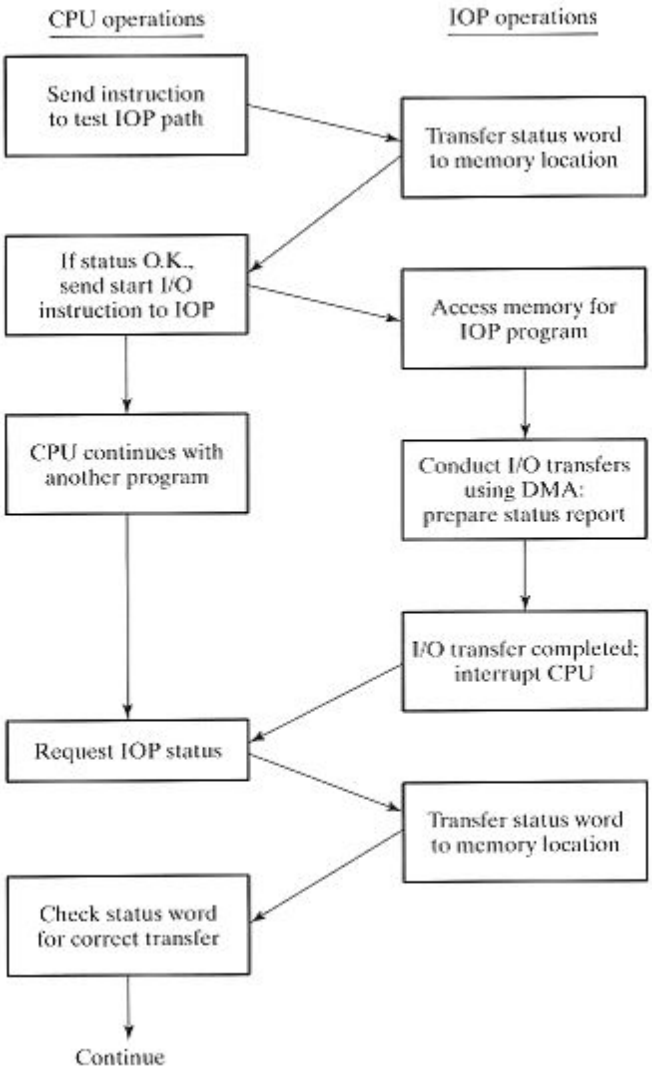


Fig: CPU-IOP communication

Data Communication Processor (DCP)

Data communication processor (DCP) is an I/O processor that distributes and collects data from many remote terminals connected through telephone and other communication lines. It is a specialized I/O processor designed to communicate directly with data communication networks (which may consists of wide variety of devices as printers, displays, sensors etc.). So DCP makes possible to operate efficiently in a time-sharing environment.

Difference between IOP and DCP: Is the way processor communicates with I/O devices.

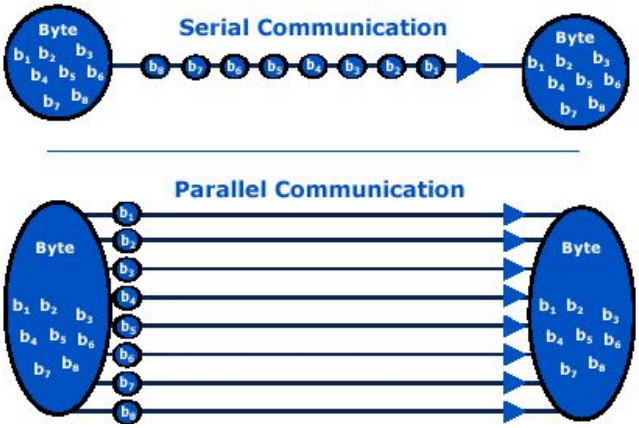
- An I/O processor communicates with the peripherals through a common I/O bus i.e. all peripherals share common bus and use to transfer information to and from I/O processor.
- DCP communicates with each terminal through a single pair of wires. Both data and control information are transferred in serial fashion.

DCP must also communicate with the CPU and memory in the same manner as any I/O processor.

Serial and parallel communication

Serial: Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication.

Parallel: Parallel communication is a method of sending several data signals simultaneously over several parallel channels. It contrasts with serial communication; this distinction is one way of characterizing a communications link.

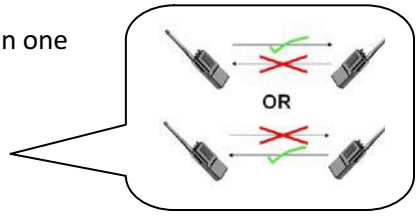


Modes of data transfer

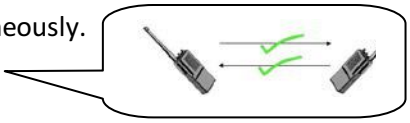
Question: *What are 3 possible modes of transfer data to and from peripherals? Explain.*

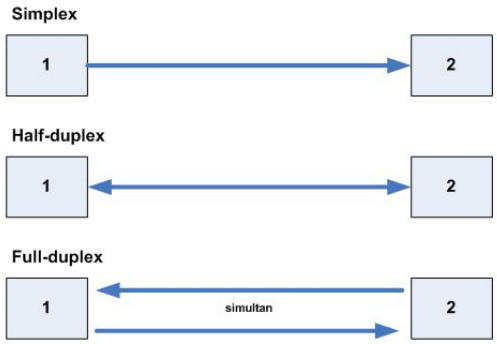
Data can be transmitted in between two points in 3 different modes:

- Simplex:
  - Carries information in one direction only.
  - Seldom used
  - Example: PC to printer, radio and TV broadcasting
- Half-duplex:
  - Capable of transmitting in both directions but only in one direction at a time.
  - Turnaround time: time to switch a half-duplex line from one direction to other.
  - Ex: walkie-talkie" style two-way radio



- Full duplex:
  - Can send and receive data in both directions simultaneously.
  - Example: Telephone, Mobile Phone, etc





Protocol

The orderly transfer of information in a data link is accomplished by means of a *protocol*. A data link control protocol is a set of rules that are followed by interconnecting computers and terminals to ensure the orderly transfer of information.

Purpose of data link protocol:

- To establish and terminate a connection between two stations
- To identify the sender and receiver
- To identify errors
- To handle all control functions

Two major categories according to the message-framing technique used:

- Character-oriented protocol
- Bit-oriented protocol

Character-oriented protocol

It is based on the binary code of the character set (e.g. ASCII). ASCII communication control characters are used for the purpose of routing data, arranging the text in desired format and for the layout of the printed page.

Code	Symbol	Meaning	Function
0010110	SYN	Synchronous idle	Establishes synchronism
0000001	SOH	Start of heading	Heading of block message
0000010	STX	Start of text	Precedes block of text
0000011	ETX	End of text	Terminates block of text
0000100	EOT	End of transmission	Concludes transmission
0000110	ACK	Acknowledge	Affirmative acknowledgement
0010101	NAK	Negative acknowledge	Negative acknowledgement
0000101	ENQ	Inquiry	Inquire if terminal is on
0010111	ETB	End of transmission block	End of block of data
0010000	DLE	Data link escape	Special control character

Table: ASCII communication control characters

Here is the typical example to appreciate the function of the DCP:

SYN	SYN	SOH	Header	STX	Text	ETX	BCC

Fig: message format

Typical message format that might be sent from a terminal to the processor is shown above. It contains following portions:

Code	Symbol	Comments
0001 0110	SYN	First sync character
0001 0110	SYN	Second sync character
0000 0001	SOH	Start of heading
0101 0100	T	Address of terminal is T4
0011 0100	4	
0000 0010	STX	Start of text transmission
0101 0010		
0100 0101	request	Text sent is a request to respond with the balance of
.	balance	account number 1234
.	of account	
.	No. 1234	
1011 0011		
0011 0100		
1000 0011	ETX	End of text transmission
0111 0000	LRC	Longitudinal parity character

Bit-oriented protocol

It allows the transmission of serial bit stream of any length without the implication of character boundaries. Messages are organized in a *frame*. In addition to the information field, a frame contains address, control and error-checking fields.

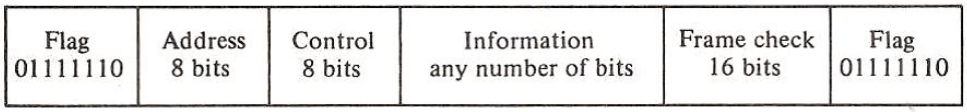


Fig: Frame format for bit-oriented protocol

A frame starts with a 8-bit flag 01111110 followed by an address and control sequence. The information field can be of any length. The frame check field CRC (cyclic redundancy check) detects errors in transmission. The ending flag represents the receiving station.