Input/output Module

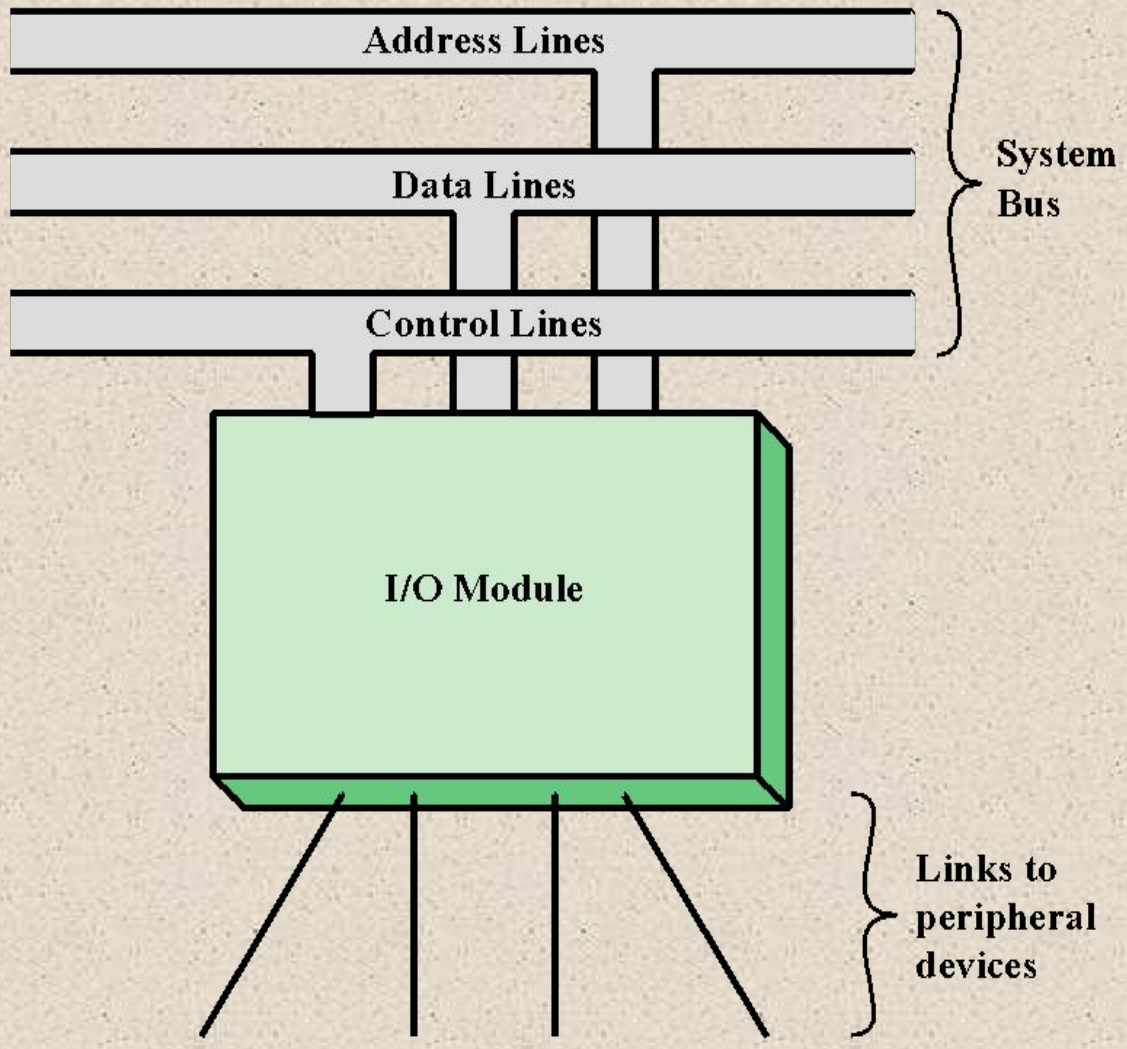


Figure 7.1 Generic Model of an I/O Module

External Devices

- Provide a means of exchanging data between the external environment and the computer
- Attach to the 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 the external device
- *Peripheral device*
  - An external device connected to an I/O module

Three categories:

- Human readable
  - Suitable for communicating with the computer user
  - Video display terminals (VDTs), printers
- Machine readable
  - Suitable for communicating with equipment
  - Magnetic disk and tape systems, sensors and actuators
- Communication
  - Suitable for communicating with remote devices such as a terminal, a machine readable device, or another computer

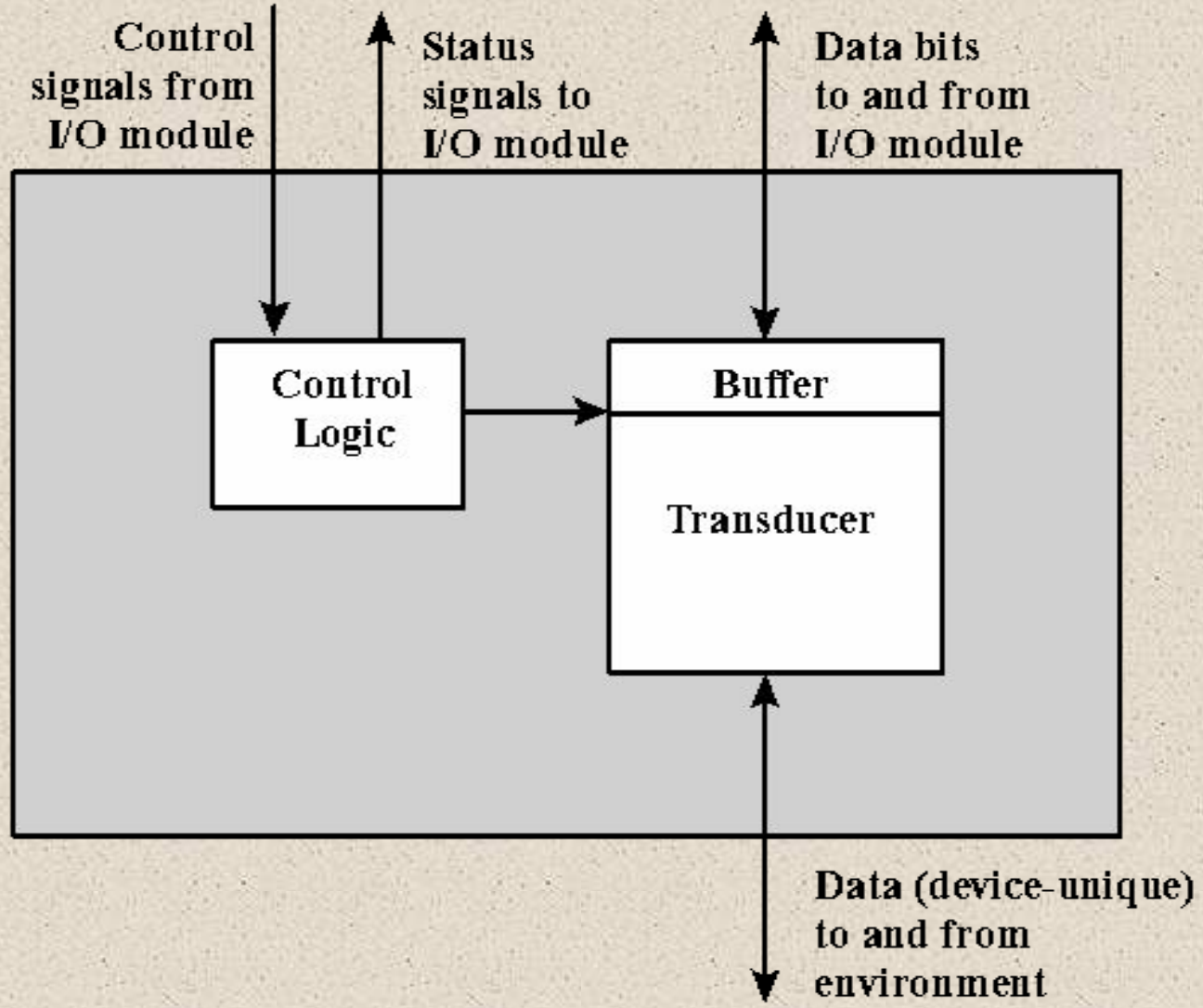


Figure 7.2 Block Diagram of an External Device

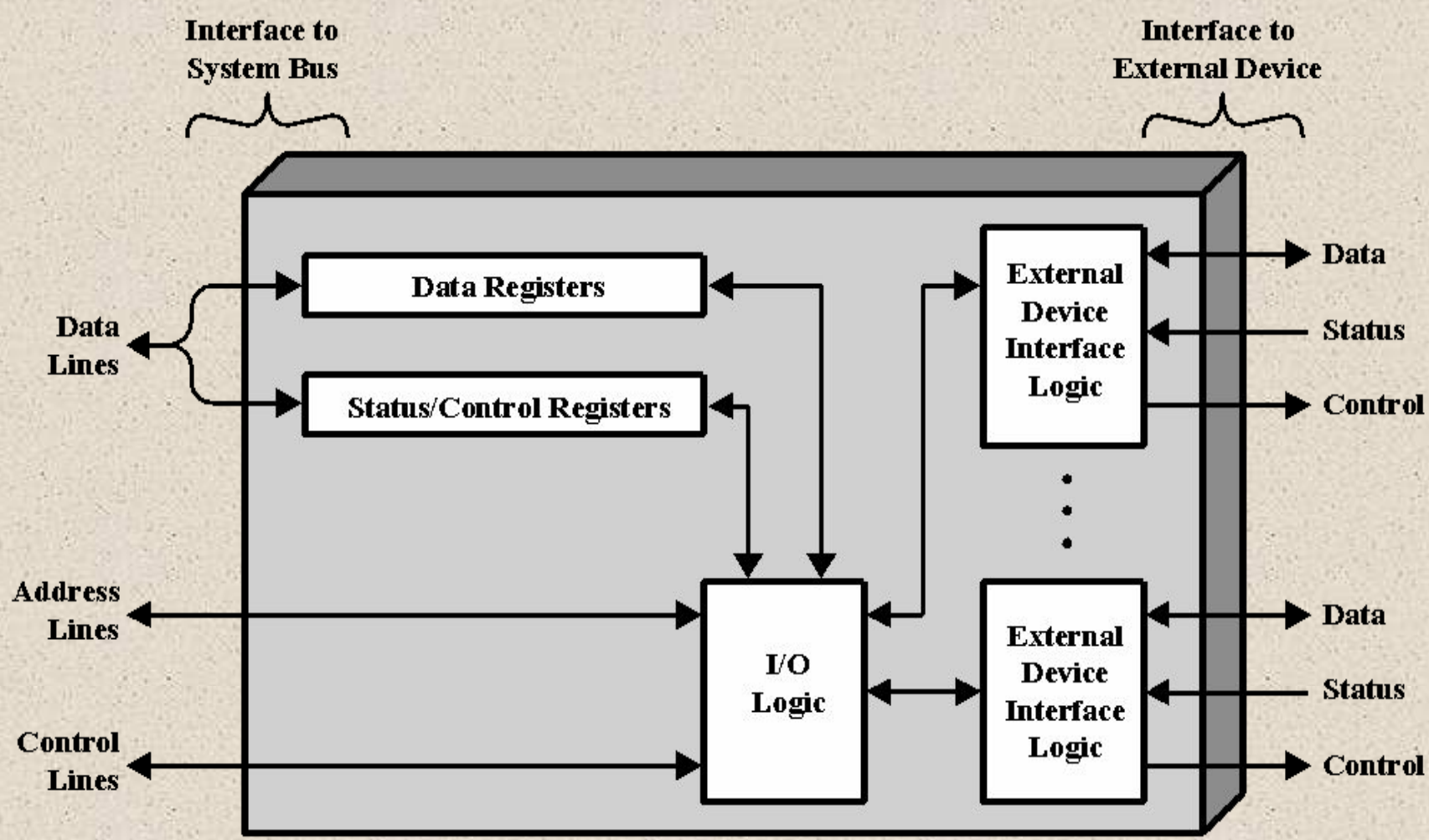


Figure 7.3 Block Diagram of an I/O Module

Keyboard/Monitor

Most common means of computer/user interaction

User provides input through the keyboard

The monitor displays data provided by the computer

International Reference Alphabet (IRA)

- Basic unit of exchange is the character
  - Associated with each character is a code
  - Each character in this code is represented by a unique 7-bit binary code
    - 128 different characters can be represented
- Characters are of two types:
  - Printable
    - Alphabetic, numeric, and special characters that can be printed on paper or displayed on a screen
  - Control
    - Have to do with controlling the printing or displaying of characters
    - Example is carriage return
    - Other control characters are concerned with communications procedures

Keyboard Codes

- When the user depresses a key it generates an electronic signal that is interpreted by the transducer in the keyboard and translated into the bit pattern of the corresponding IRA code
- This bit pattern is transmitted to the I/O module in the computer
- On output, IRA code characters are transmitted to an external device from the I/O module
- The transducer interprets the code and sends the required electronic signals to the output device either to display the indicated character or perform the requested control function

The major functions for an I/O module fall into the following categories:

Control and timing
- Coordinates the flow of traffic between internal resources and external devices

Processor communication
- Involves command decoding, data, status reporting, address recognition

Device communication
- Involves commands, status information, and data

Data buffering
- Performs the needed buffering operation to balance device and memory speeds

Error detection
- Detects and reports transmission errors

Programmed I/O

Three techniques are possible for I/O operations:

- Programmed I/O
  - Data are exchanged between the processor and the I/O module
  - Processor executes a program that gives it direct control of the I/O operation
  - When the processor issues a command it must wait until the I/O operation is complete
  - If the processor is faster than the I/O module this is wasteful of processor time
- Interrupt-driven I/O
  - Processor issues an I/O command, continues to execute other instructions and is interrupted by the I/O module when the latter has completed its work
- Direct memory access (DMA)
  - The I/O module and main memory exchange data directly without processor involvement

Table 7.1
I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

I/O Commands

- There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:

1) Control

   - used to activate a peripheral and tell it what to do

2) Test

   - used to test various status conditions associated with an I/O module and its peripherals

3) Read

   - causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer

4) Write

   - causes the I/O module to take an item of data from the data bus and subsequently transmit that data item to the peripheral

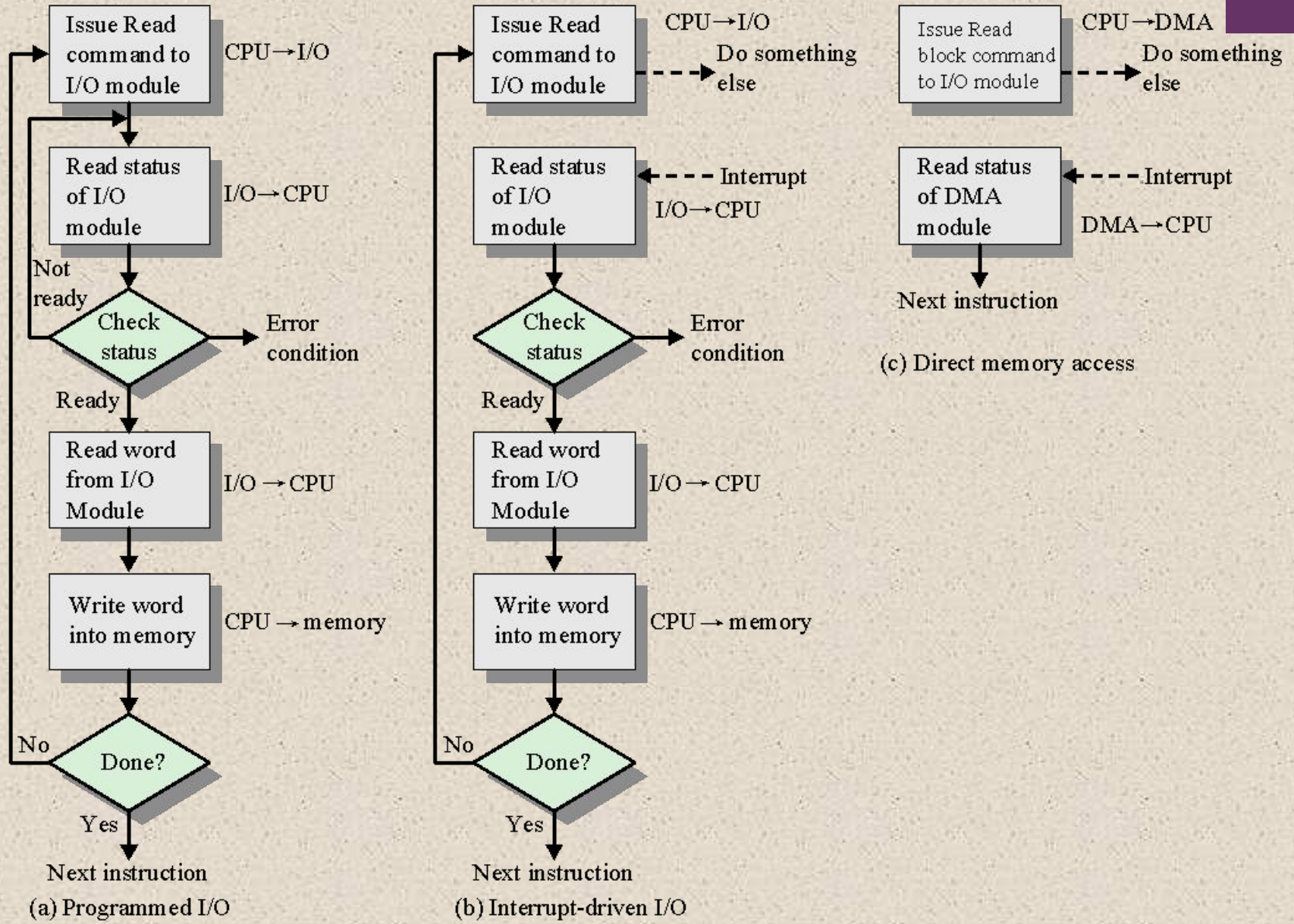


Figure 7.4 Three Techniques for Input of a Block of Data

Programmed I/O:

In this mode the data transfer is initiated by the instructions written in a computer program. An input instruction is required to store the data from the device to the CPU and a store instruction is required to transfer the data from the CPU to the device. Data transfer through this mode requires constant monitoring of the peripheral device by the CPU and also monitor the possibility of new transfer once the transfer has been initiated. Thus CPU stays in a loop until the I/O device indicates that it is ready for data transfer. Thus programmed I/O is a time consuming process that keeps the processor busy needlessly and leads to wastage of the CPU cycles.

This can be overcome by the use of an interrupt facility. This forms the basis for the Interrupt Initiated I/O.

Interrupt Driven I/O:

This mode uses an interrupt facility and special commands to inform the interface to issue the interrupt command when data becomes available and interface is ready for the data transfer. In the meantime CPU keeps on executing other tasks and need not check for the flag. When the flag is set, the interface is informed and an interrupt is initiated. This interrupt causes the CPU to deviate from what it is doing to respond to the I/O transfer. The CPU responds to the signal by storing the return address from the program counter (PC) into the memory stack and then branches to service that processes the I/O request. After the transfer is complete, CPU returns to the previous task it was executing. The branch address of the service can be chosen in two ways known as vectored and non-vectored interrupt. In vectored interrupt, the source that interrupts, supplies the branch information to the CPU while in case of non-vectored interrupt the branch address is assigned to a fixed location in memory.

Direct Memory Access:

The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.

Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

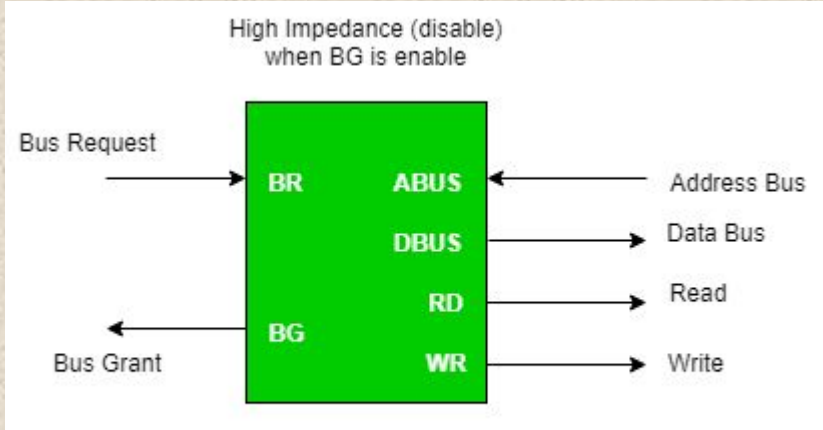


Figure - CPU Bus Signals for DMA Transfer

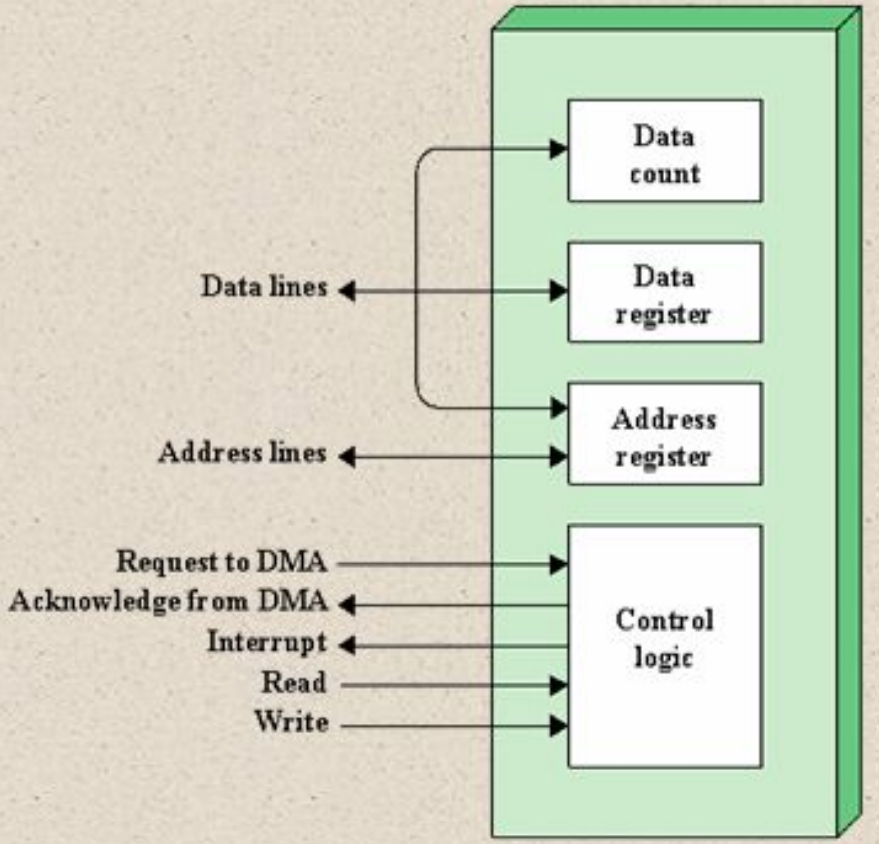
Types of DMA transfer using DMA controller:

Burst Transfer :

DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer.

Cyclic Stealing :

An alternative method in which DMA controller transfers one word at a time after which it must return the control of the buses to the CPU. The CPU delays its operation only for one memory cycle to allow the direct memory l



Block diagram of DMA

Programmed I/O	Interrupt Initiated I/O
Data transfer is initiated by the means of instructions stored in the computer program. Whenever there is a request for I/O transfer the instructions are executed from the program.	The I/O transfer is initiated by the interrupt command issued to the CPU.
The CPU stays in the loop to know if the device is ready for transfer and has to continuously monitor the peripheral device.	There is no need for the CPU to stay in the loop as the interrupt command interrupts the CPU when the device is ready for data transfer.
This leads to the wastage of CPU cycles as CPU remains busy needlessly and thus the efficiency of system gets reduced.	The CPU cycles are not wasted as CPU continues with other work during this time and hence this method is more efficient.
CPU cannot do any work until the transfer is complete as it has to stay in the loop to continuously monitor the peripheral device.	CPU can do any other work until it is interrupted by the command indicating the readiness of device for data transfer
Its module is treated as a slow module.	Its module is faster than programmed I/O module.
It is quite easy to program and understand.	It can be tricky and complicated to understand if one uses low level language.
The performance of the system is severely degraded.	The performance of the system is enhanced to some extent.