# I/O Interface (Interrupt and DMA Mode)

The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected to any computer system. These communication links are used to resolve the differences between CPU and peripheral. There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.

## Mode of Transfer:

The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes. Data transfer to and from the peripherals may be done in any of the three possible ways:

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

**Now let's discuss each mode one by one.**

1. **Programmed I/O:** It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices. **Example of Programmed I/O:** In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory. In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility. This is discussed below.
2. **Interrupt- initiated I/O:** Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.
   - The I/O transfer rate is limited by the speed with which the processor can test and service a device.

- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.
- **Terms:**
    - Hardware Interrupts: Interrupts present in the hardware pins.
    - Software Interrupts: These are the instructions used in the program whenever the required functionality is needed.
    - Vectored interrupts: These interrupts are associated with the static vector address.
    - Non-vectored interrupts: These interrupts are associated with the dynamic vector address.
    - Maskable Interrupts: These interrupts can be enabled or disabled explicitly.
    - Non-maskable interrupts: These are always in the enabled state. we cannot disable them.
    - External interrupts: Generated by external devices such as I/O.
    - Internal interrupts: These devices are generated by the internal components of the processor such as power failure, error instruction, temperature sensor, etc.
    - Synchronous interrupts: These interrupts are controlled by the fixed time interval. All the interval interrupts are called as synchronous interrupts.
    - Asynchronous interrupts: These are initiated based on the feedback of previous instructions. All the external interrupts are called as asynchronous interrupts.

3. **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.
    - Buses grant request time.
    - Transfer the entire block of data at transfer rate of device because the device is usually slow than the speed at which the data can be transferred to CPU.
    - Release the control of the bus back to CPU so, total time taken to transfer the N bytes = Bus grant request time + (N) * (memory transfer rate) + Bus release control time.
    - Buffer the byte into the buffer
    - Inform the CPU that the device has 1 byte to transfer (i.e. bus grant request)
    - Transfer the byte (at system bus speed)
    - Release the control of the bus back to CPU.

## Advantages:

**Standardization:** I/O interfaces provide a standard way of communicating with external devices. This means that different devices can be connected to a computer using the same interface, which makes it easier to swap out devices and reduces the need for specialized hardware.

**Modularity:** With I/O interfaces, different devices can be added or removed from a computer without affecting the other components. This makes it easier to upgrade or replace a faulty device without affecting the rest of the system.

**Efficiency:** I/O interfaces can transfer data between the computer and the external devices at high speeds, which allows for faster data transfer and processing times.

**Compatibility:** I/O interfaces are designed to be compatible with a wide range of devices, which means that users can choose from a variety of devices that are compatible with their computer's I/O interface.

## Disadvantages:

**Cost:** I/O interfaces can be expensive, especially if specialized hardware is required to connect a particular device to a computer system.

**Complexity:** Some I/O interfaces can be complex to configure and require specialized knowledge to set up and maintain. This can be a disadvantage for users who are not familiar with the technical aspects of computer hardware.

**Compatibility issues:** While I/O interfaces are designed to be compatible with a wide range of devices, there can still be compatibility issues with certain devices. In some cases, device drivers may need to be installed to ensure proper functionality.

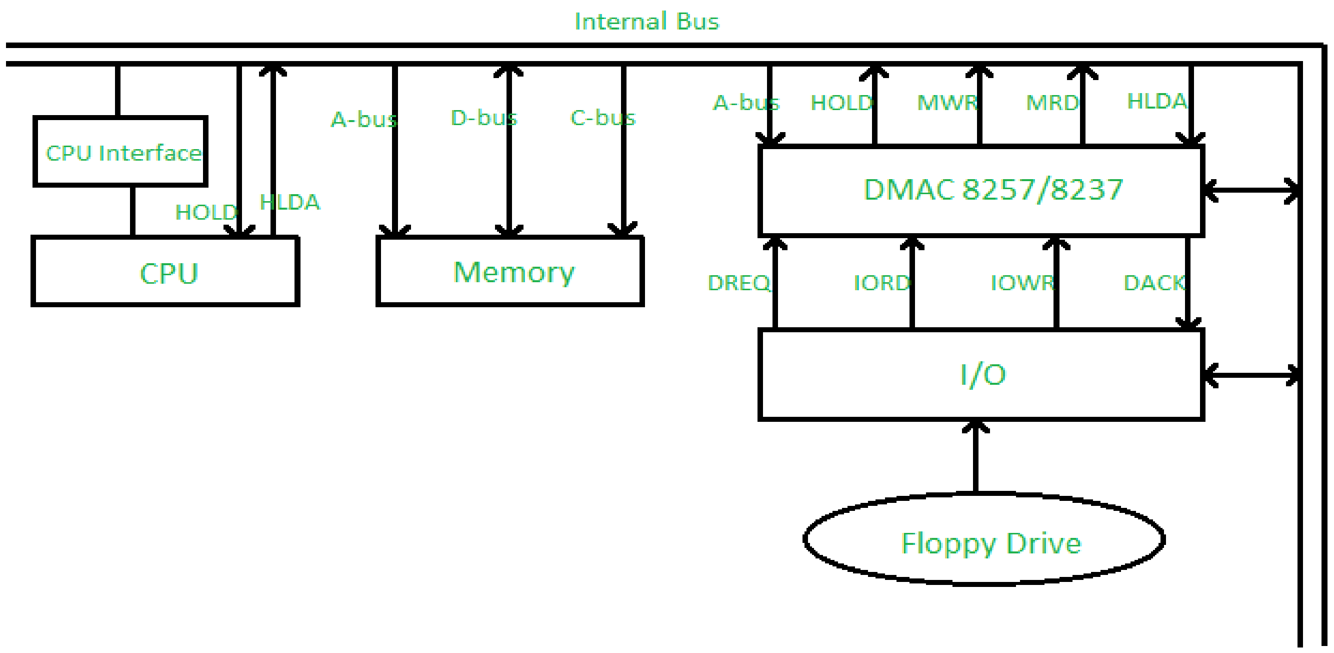
**Security risks:** I/O interfaces can be a security risk if they are not properly configured or secured. Hackers can exploit vulnerabilities in I/O interfaces to gain unauthorized access to a computer system or steal data.

## Direct memory access with DMA controller 8257/8237

Suppose any device which is connected to input-output port wants to transfer data to memory, first of all it will send input-output port address and control signal, input-output read to input-output port, then it will send memory address and memory write signal to memory where data has to be transferred. In normal input-output technique the processor becomes busy in checking whether any input-output operation is completed or not for next input-output operation, therefore this technique is slow.

This problem of slow data transfer between input-output port and memory or between two memories is avoided by implementing Direct Memory Access (DMA) technique. This is faster as the microprocessor/computer is bypassed and the control of address bus and data bus is given to the DMA controller.

| | |
|---|---|
| HOLD | – Hold Signal |
| HLDA | – Hold Acknowledgment |
| DREQ | – DMA Request |
| DACK | – DMA Acknowledgment |

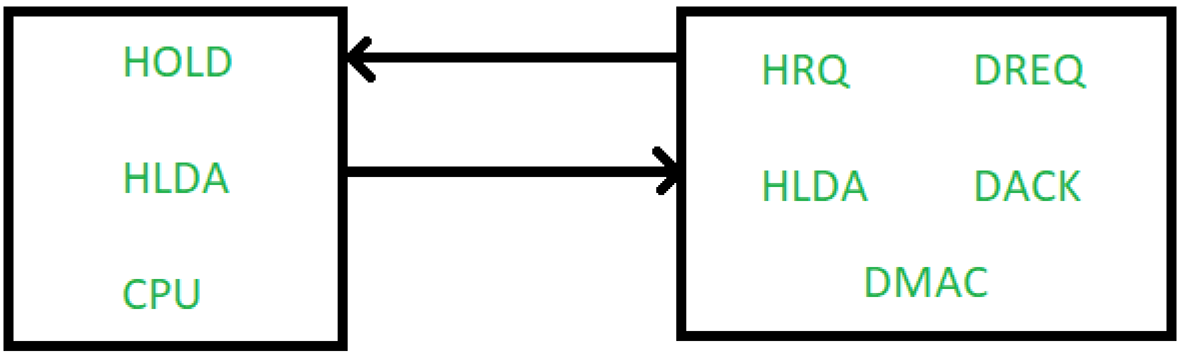


Suppose a floppy drive that is connected at input-output port wants to transfer data to memory, the following steps are performed:
**Step-1:** First of all the floppy drive will send a DMA request (DREQ) to the DMAC, it means the floppy drive wants its DMA service.

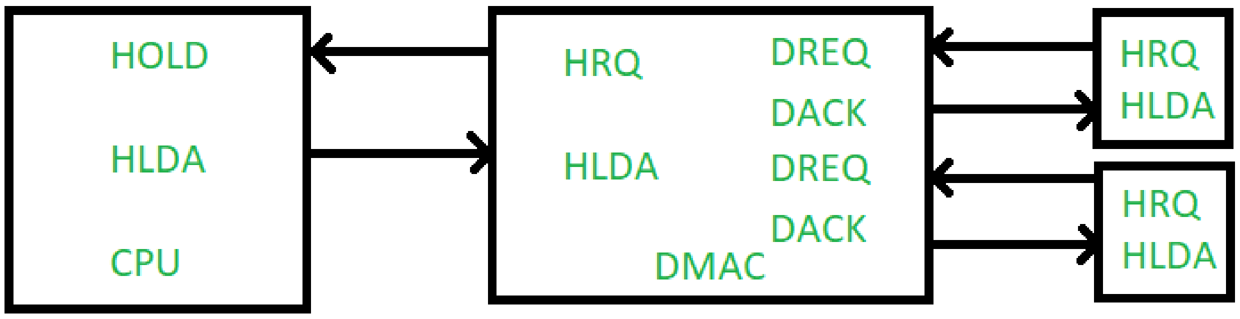
**Step-2:** Now the DMAC will send a HOLD signal to the CPU.

**Step-3:** After accepting the DMA service request from the DMAC, the CPU will send hold acknowledgment (HLDA) to the DMAC, it means the microprocessor has released control of the address bus the data bus to DMAC and the microprocessor/computer is bypassed during DMA service.
**Step-4:** Now the DMAC will send one acknowledgement (DACL) to the floppy drive which is connected at the input-output port. It means the DMAC tells the floppy drive be ready for its DMA service.
**Step-5:** Now with the help of input-output read and memory write signal the data is transferred from the floppy drive to the memory.

**Modes of DMAC:**
**1. Single Mode –** In this only one channel is used, means only a single DMAC is connected to the bus system.



**2. Cascade Mode –** In this multiple channels are used, we can further cascade more number of DMACs.



**Advantages:**

**Improved performance:** DMA improves system performance by freeing up the CPU to perform other tasks while data is being transferred between memory and I/O devices. This allows for faster and more efficient data transfer.

**Reduced CPU overhead:** With DMA, the CPU is not required to be involved in data transfer, which reduces the CPU overhead and allows it to focus on other tasks. This is particularly useful in real-time systems where low latency and fast response times are important.

**Support for high-bandwidth devices:** DMA can support high-bandwidth devices such as graphics cards and network interfaces that require fast data transfer rates.

**Efficient use of system resources:** DMA allows multiple devices to access memory simultaneously, which makes more efficient use of system resources.

**Disadvantages:**

**Complexity:** DMA requires specialized hardware and software to function, which can add to the complexity of a system. This can make it difficult to implement and troubleshoot.

**Security risks:** DMA can be a security risk if not properly configured or secured. Hackers can exploit vulnerabilities in DMA to gain unauthorized access to a computer system or steal data.

**Limited control:** Since the CPU is not involved in data transfer with DMA, it has limited control over the transfer process. This can lead to data corruption or errors if the transfer process is not properly managed.

**Resource conflicts:** DMA can lead to resource conflicts if multiple devices attempt to access memory simultaneously. This can cause system instability and performance issues if not properly managed.