I/O Module

Chapter 7 Page-228—254 Book of William Stalling

I/O Module

1. Each module interfaces to the system bus and controls one or more peripheral devices.

2. An I/O module is not simply a set of mechanical connectors that wire a device into the system bus. Rather, the I/O module contains logic for performing a communication function between the peripheral and the bus.

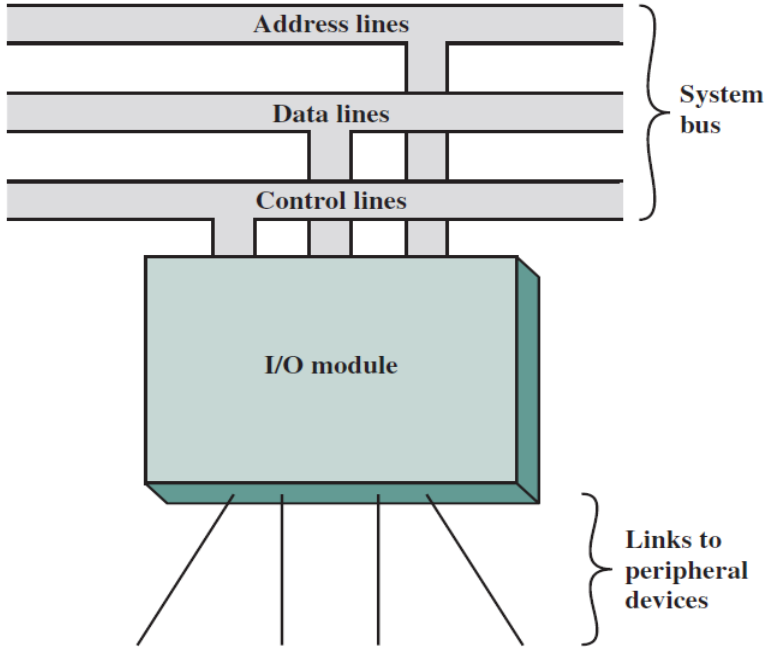


Figure 7.1 Generic Model of an I/O Module

Why an I/O Module is Required?

1. There are a wide variety of peripherals with various methods of operation.
2. The data transfer rate of peripherals is often much slower than that of the memory or processor. On the other hand, the data transfer rate of some peripherals is faster than that of the memory or processor.
3. Peripherals often use different data formats and word lengths than the computer to which they are attached.
4. This module has two major functions:

   —Interface to the processor and memory via the system bus

   —Interface to one or more peripheral devices

Function of an I/O Module

1. Control and timing
2. Processor communication
3. Device communication
4. Data buffering
5. Error detection

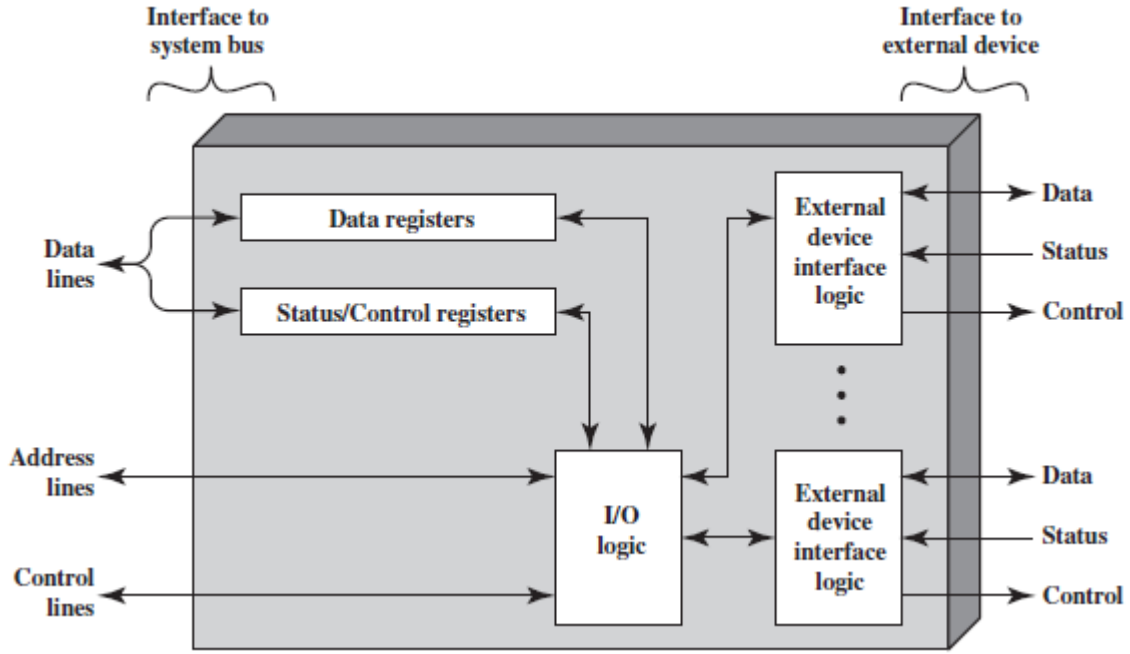
I/O Modules' Function: Control and Timing

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 operational and ready to transmit, the processor requests the transfer of data, by means of a command to the I/O module.
4. The I/O module obtains a unit of data (e.g., 8 or 16 bits) from the external device.
5. The data are transferred from the I/O module to the processor

I/O Modules’ Function: Processor and Device Communication

1. Command decoding
2. Data
3. Status Reporting
4. Address Recognition

I/O Module Structure



Programmed I/O

1. Data are exchanged between the processor and the I/O module.
2. The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.
3. When the processor issues a command to the I/O module, it must wait until the I/O operation is complete. If the processor is faster than the I/O module, this is waste of processor time.

Programmed I/O

1. When the processor encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module.
2. The I/O module will perform the requested action and then set the appropriate bits in the I/O status register and takes no further action to alert the processor.
3. It is the responsibility of the processor to periodically check the status of the I/O module until it finds that the operation is complete

Shortcomings of Programmed I/O

1. The processor has to wait a long time for the I/O module to be ready for either reception or transmission of data.
2. The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded.

Interrupt driven I/O

1. The processor issues an I/O command to a module and then go on to do some other useful work.
2. The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor. The processor then executes the data transfer as before, and then resumes its former processing.

Interrupt driven I/O: I/O Module Perspective

1. For input, the I/O module receives a READ command from the processor.
2. The I/O module then proceeds to read data in from an associated peripheral. Once the data are in the module’s data register, the module signals an interrupt to the processor over a control line.
3. The module then waits until its data are requested by the processor. When the request is made, the module places its data on the data bus and is then ready for another I/O operation

Interrupt Driven I/O: Processor Perspective

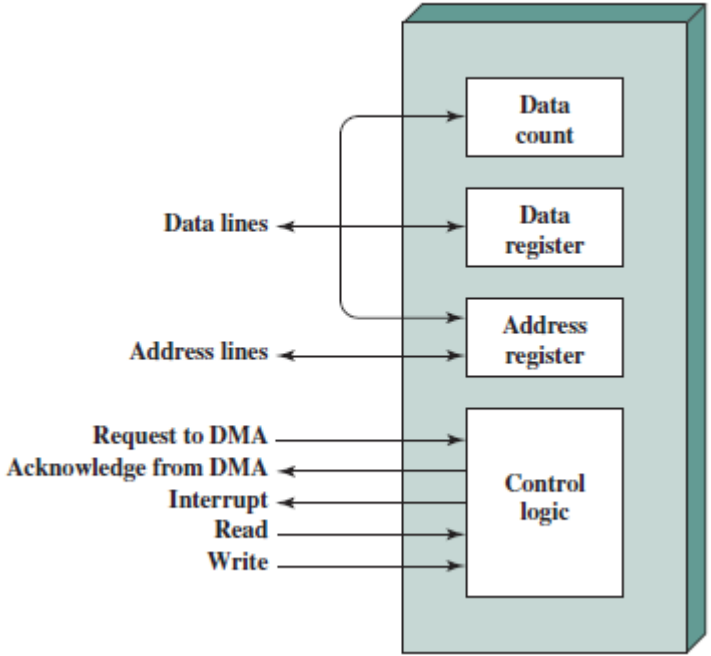
1. The processor issues a READ command. It then goes off and does something else.
2. At the end of each instruction cycle, the processor checks for interrupts. When the interrupt from the I/O module occurs, the processor saves the context (e.g., program counter and processor registers) of the current program and processes the interrupt.
3. The processor reads the word of data from the I/O module and stores it in memory. It then restores the context of the program it was working on (or some other program) and resumes execution

DMA Function

1. DMA involves an additional module on the system bus.
2. It mimics the processor.
3. The DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily. The latter technique is more common and is referred to as cycle stealing, because the DMA module in effect steals a bus cycle.

DMA Function

1. A read or write request using the read or write control line between the processor and the DMA.
2. The address of the I/O device communicated on the data lines
3. The starting location in memory to read from or write to, communicated on the data lines and stored in the address register.
4. The number of words to be read or written communicated via the data lines and stored in the data count register.

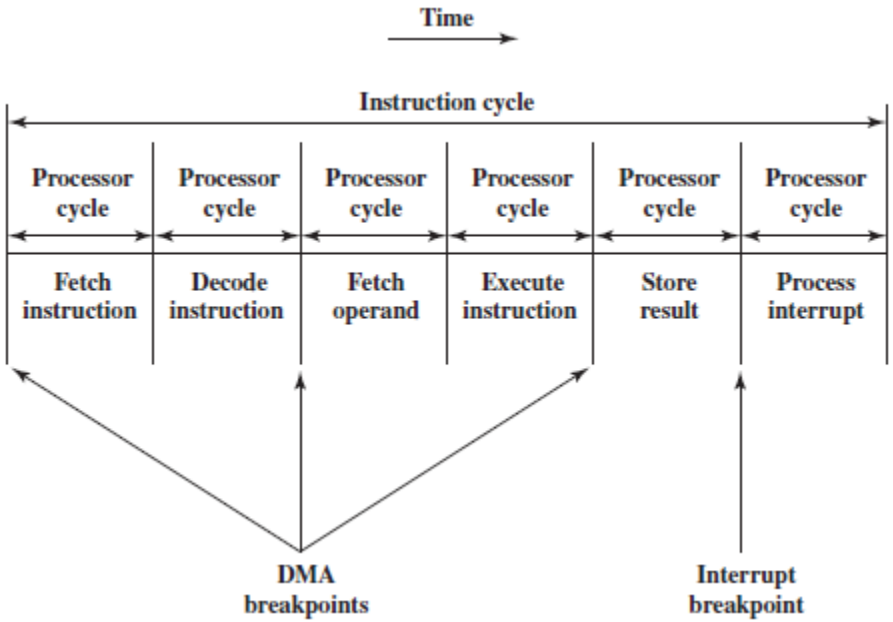


DMA Function

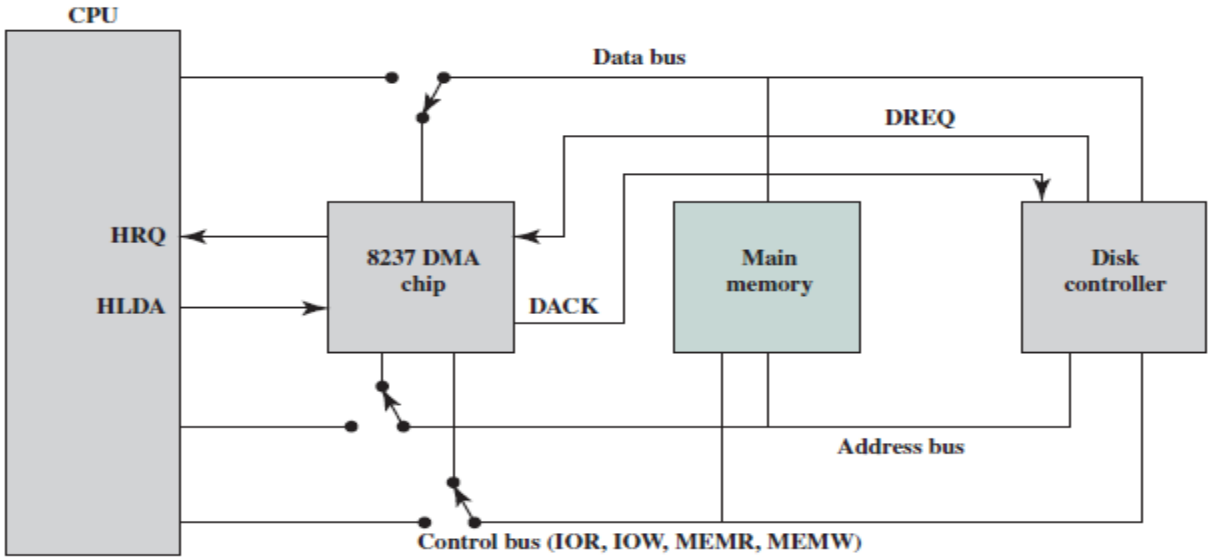
5. The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the processor.

6. When the transfer is complete, the DMA module sends an interrupt signal to the processor.

DMA and Interrupt Breakpoints during an Instruction Cycle



Intel 8237A DMA Controller



DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request