PERIPHERAL/COMPUTER CONNECTIONS

Introduction

- The CPU performs arithmetic and logic operations and controls the operation of the entire system.
- Some tasks are completely controlled by CPU, but for some others, it merely initiates a sequence of events which are controlled elsewhere, such as in a peripheral device.
- The peripheral devices permit communication of information and storage of information.

Introduction

- Peripherals normally communicate through the CPU.
- Some may communicate themselves and memory bypassing CPU.
- The number and types of peripheral devices depend on the main applications for which the computer system is intended.

Components of a digital computer

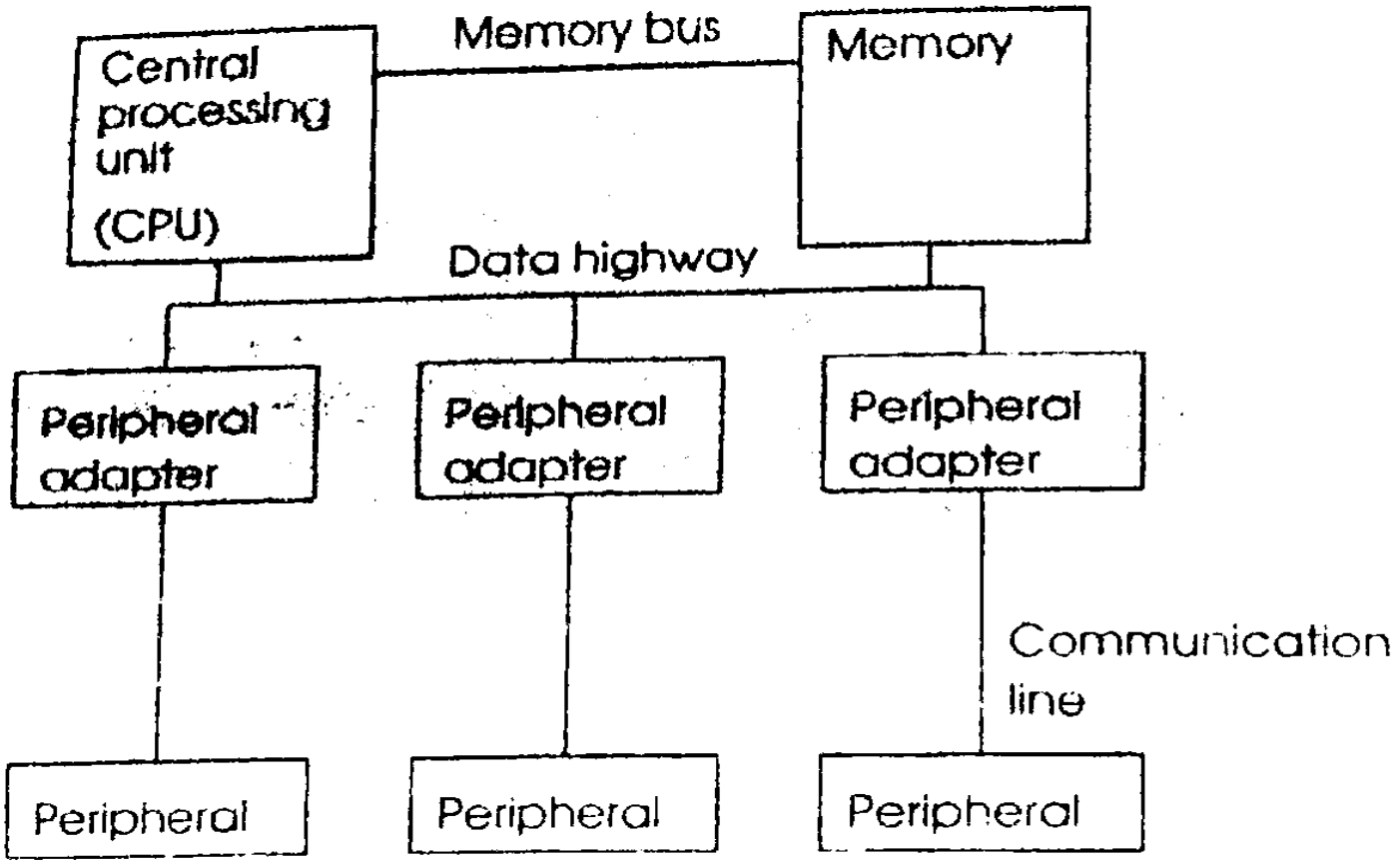


Figure 2.1 *Components of a digital computer*

Data Highway

- Data (including programs) are moved around the computer on a set of wires forming a data highway (Bus).
- Address Bus, Control Bus, Data Bus

Data Highway: Address Bus

- It contains address information.
- The number of lines available for this determines the maximum amount of physical memory that can be addressed.
- If there are n lines , then 2^n locations can be addressed.
- 8086 has 20 address lines and thus it can address 2^{20}=1048576 (1 Mega bytes) memory locations.

Data Highway: Data Bus

- It contains the information or program instructions required by the CPU.
- The speed of operation of a computer depend on the rate at which data can be made available and a good way to improve the speed is to add more data lines to transfer more data at a time.
- Small systems use 8-bit data bus. They are relatively slow but of low cost.
- 8086 uses 16-bit and 80386 uses 32-bit bus.

Data Highway: Control Bus

- It contains control signals.
- Control signals are required to maintain orderly flow of data along the bus. It is necessary to indicate whether to **read** or **write** data during a transfer, Timing signals. It also indicates whether a transfer is to **memory** or a **peripheral adaptor**.

Peripheral Adapter Registers

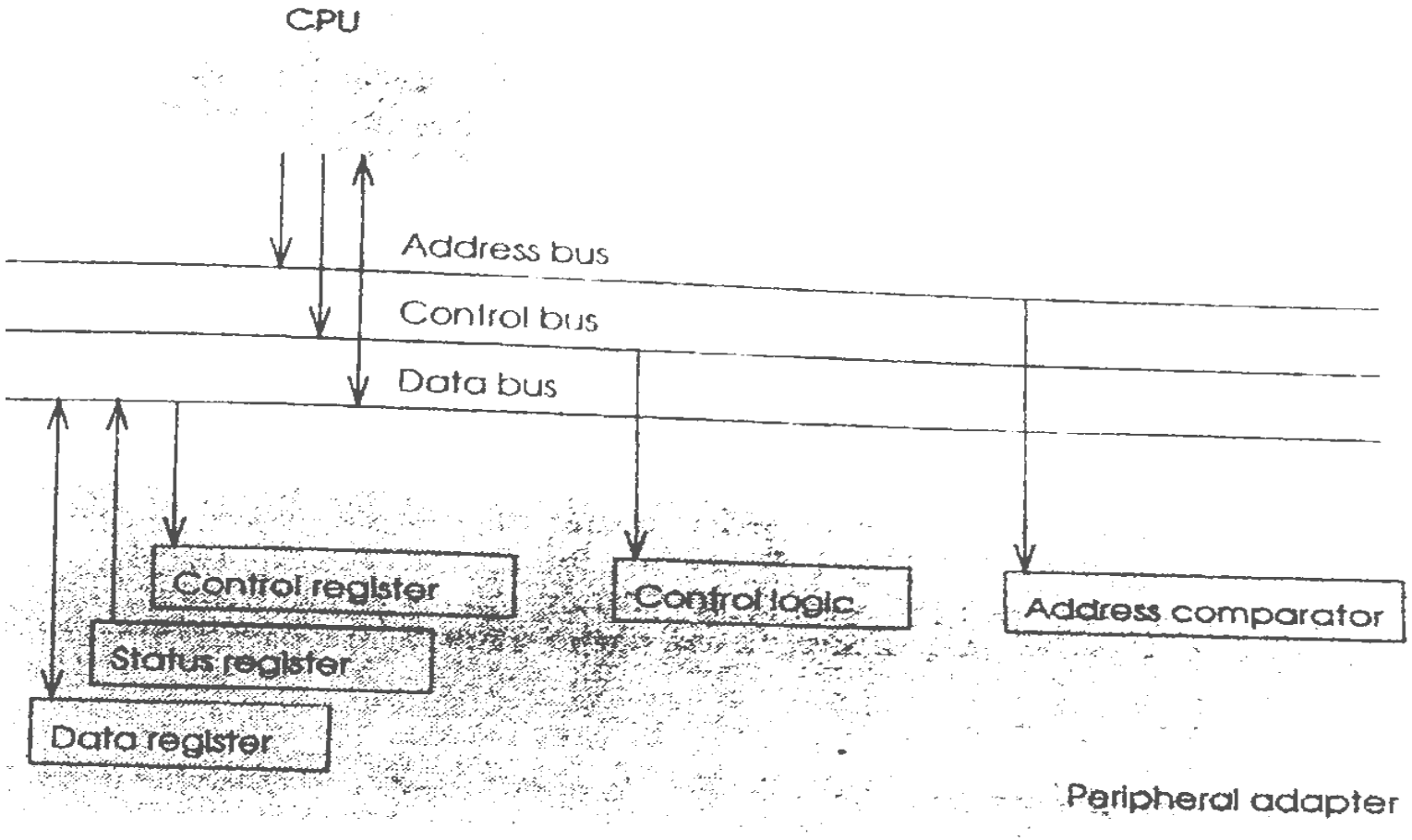


Figure 2.2 *Connection of peripheral adapter to highway*

Peripheral Adapter Registers

- Peripheral adaptors are directly connected to the buses.
- PA contains:
  - Control register
  - Status register
  - Data register
  - Control Logic
  - Address Comparator.

Peripheral Adapter Registers

Address Comparator

- It is necessary to distinguish between adapters used for different peripherals. So a value is allocated to the adaptor (*Often set with switches or jumpers*).
- An adaptor may recognize a single address or a small group of address.
- It compares the address on the address bus with the peripheral adapter's address.
- If the address on the bus matches that of the adapter then the control lines are interpreted by the ***control logic*** to perform the required function, typically to read from or write to a ***register*** (may be data register) connected to a data bus.

Peripheral Adapter Registers

- **Control Register**
  - Stores values written to it to control the operation of the adaptor.
- **Status Register**
  - Can be read by the CPU to determine the status of the device (e.g. whether it is ready for use, busy, switched on/off etc.)
  - Each piece of information stored in the control and status registers usually needs only a single bit and several such bits are stored in each register, each bit is often known as a *flag*.
- **Data Register**
  - Used to hold temporarily a value to be transferred to or from the peripheral so that it is not necessary to synchronize the computer with the peripheral.

Computer Input/Output Operation

- Programmed I/O
- Interrupt
- Block Data Transfer
- Direct Memory Access

Programmed I/O

- The programmed I/O was the simplest type of I/O technique for the exchanges of data or any types of communication between the processor and the external devices.
- With programmed I/O, data are exchanged between the processor and the I/O module.
- The instruction set for the CPU typically contains instructions such as
  - Input data to the CPU from the peripheral
  - Output data from the CPU to the peripheral
  - Set individual bits in the peripheral adapter's control register
  - Test individual bits in the peripheral adapter's status register
- If the peripheral is ready, CPU puts the data into data register.
- This will automatically set a flag to indicate that data is available for the peripheral.
- If the peripheral is not ready
  - Program must wait and try again

Programmed I/O: Software Polling

- If a number of peripherals are active at the same time, it is necessary to test each one for readiness.
- The technique of testing a number of peripherals in turn is known as ***software polling.***

Processor instructions for communicating with peripherals are often given obvious mnemonics such as "IN" and "OUT"

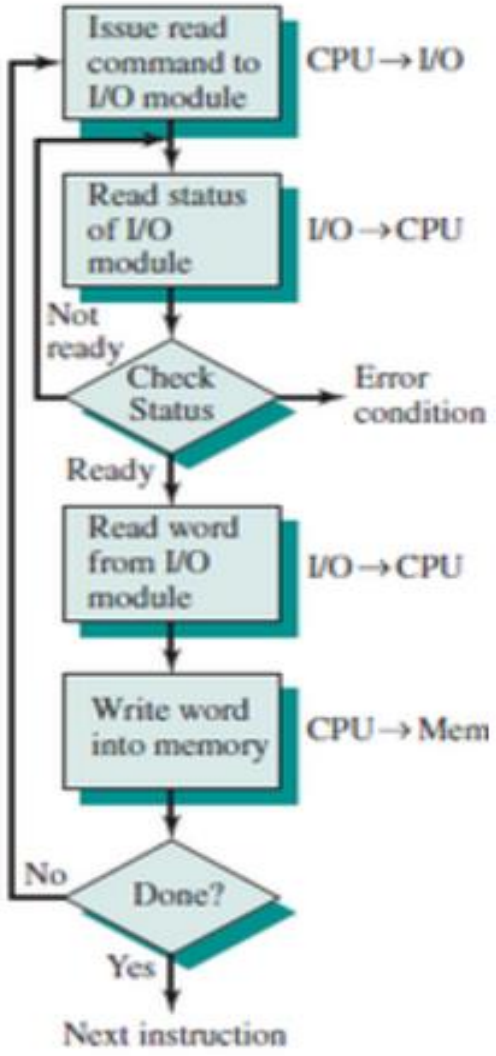
Example:

```
OUT   DATA,AL

IN    AL,STATUS
```

Programmed I/O: Data Transfer

- 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.



Interrupts

- Improve on Programmed I/O
  - Not require the program to sit idle while waiting for a peripheral to become ready
- More effective use of the processor, including running programs at the same time as I/O is being performed
- When a peripheral is able to transfer data
  - It sets its ready flag in its status register and asserts a control line connected to the CPU
- This control line is interpreted by the CPU as an *interrupt request* (IREQ)
- CPU then stops what it is doing , stores enough information to be able to resume later and starts executing an interrupt routine
- This routine deals with the device and then uses the information stored at the beginning of the interrupt to return the CPU to executing the interrupted program – without the program being aware of what has taken place

Interrupts

- 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.

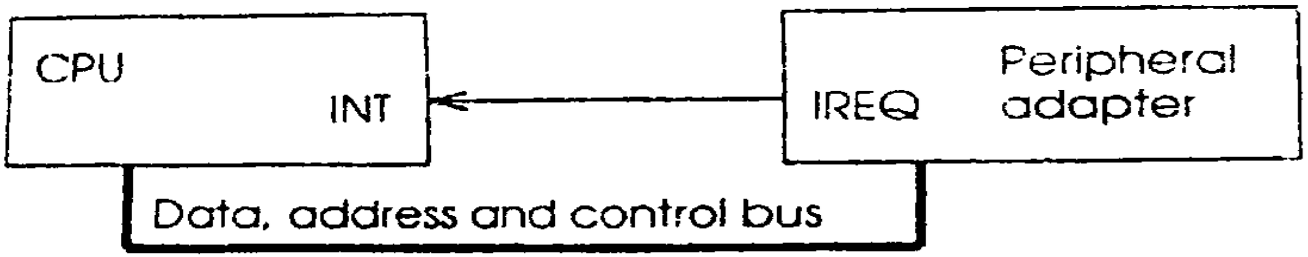


Figure 2.5 *Interrupt connection*

Priority Interrupts

- Whenever a peripheral is ready to transfer data , it is necessary to service its interrupt within a reasonable time
  - Otherwise subsequent data may be lost
- Faster peripheral require faster servicing
- If two or more peripherals are ready at the same time, it is better to service the faster one first since the slow one can be made to wait a little while
- Multiple interrupt requests can be resolved by using a priority interrupt scheme
- A hardware priority encoder arbitrates between requests and sends a single value- that represents the highest priority device- to the CPU
- The interrupt request is formed by performing a logical OR of the peripherals IREQ lines
- Interrupt requests are assumed to remain asserted until reset by instructions in the service routine.
- This is not the most efficient technique

Priority Interrupts

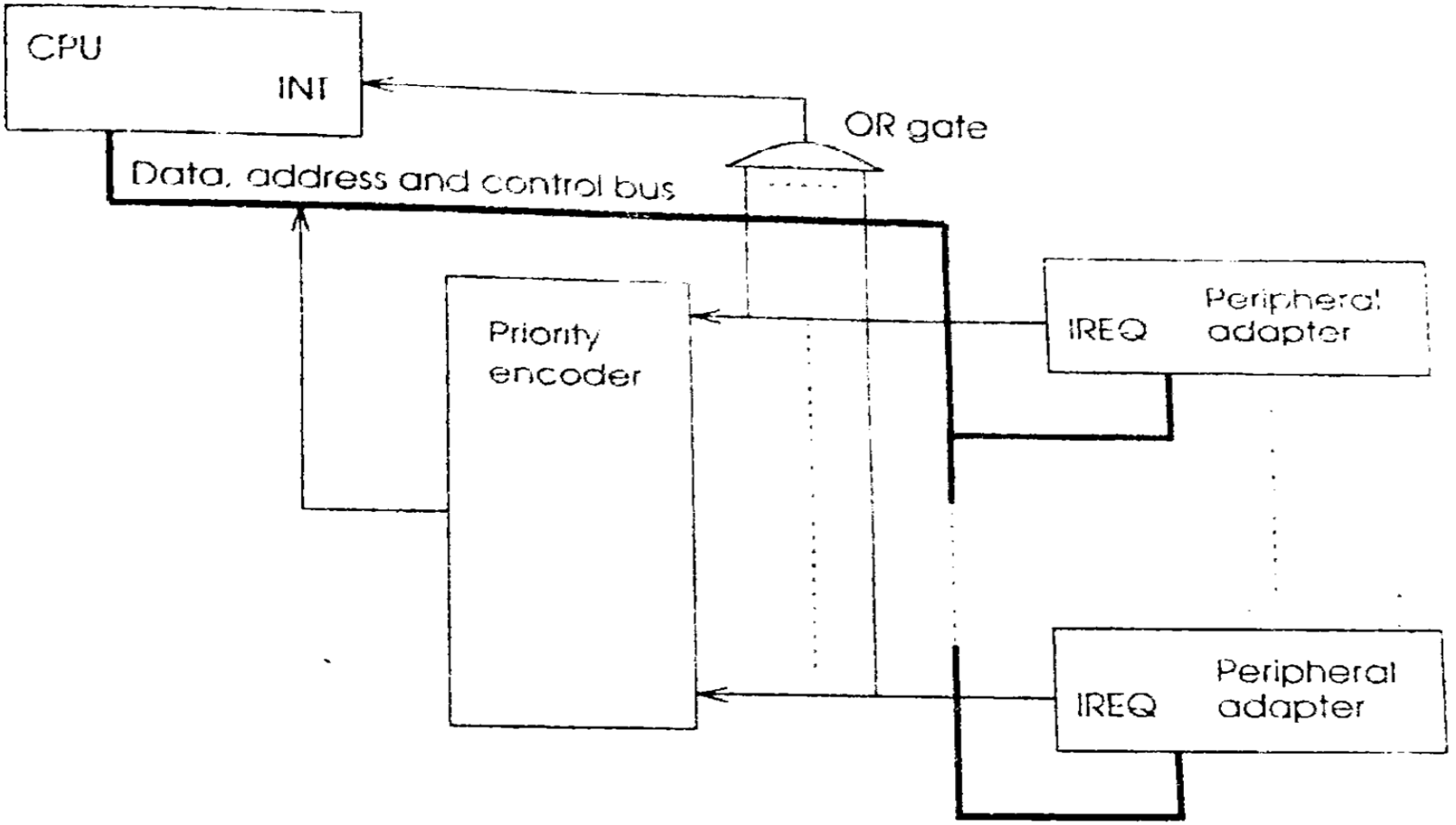


Figure 2.6 *Priority interrupts using a priority encoder*

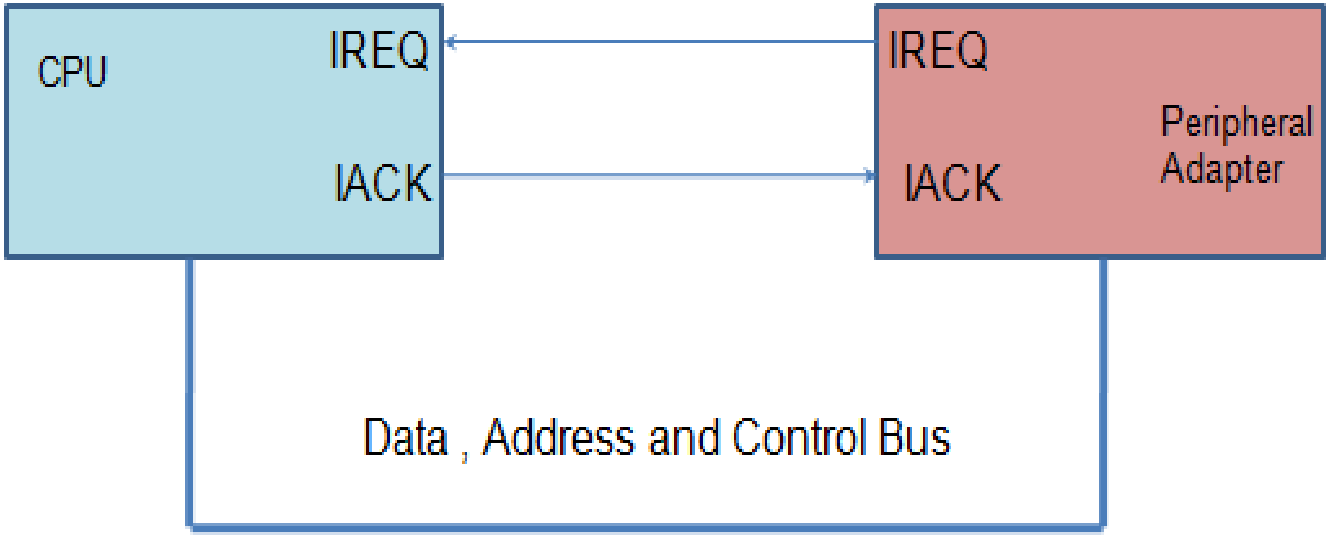
Priority Interrupts: Problem

- Until a request is de-asserted it is not possible for another request to be seen

- This may result
  - Data from a fast peripheral being lost while a service routine is getting around to clearing a low priority interrupt

Interrupt with acknowledge

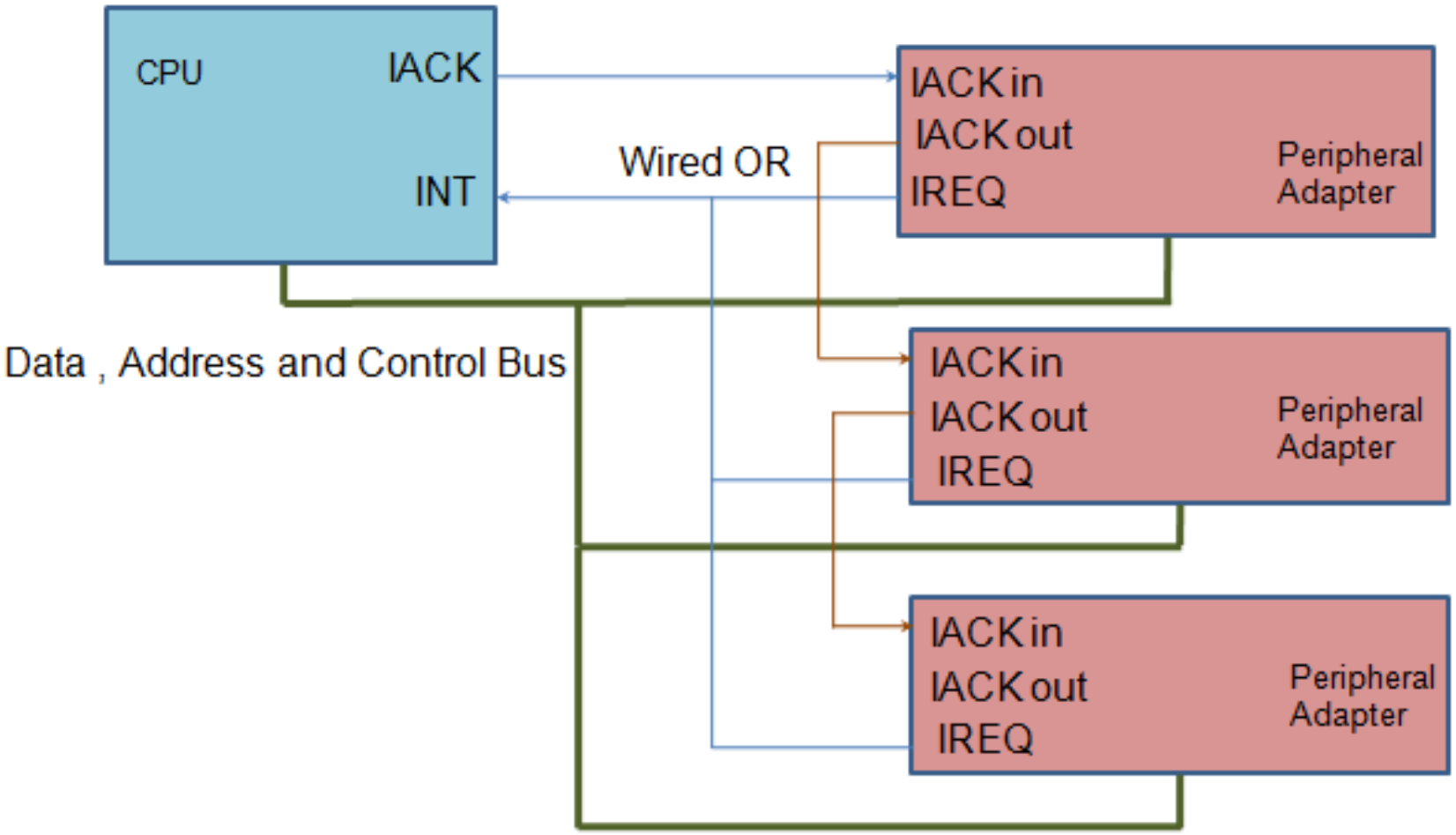
- Most of the computers have a signal generated by the CPU that is returned to the peripheral as soon as the interrupt is detected- *Interrupt acknowledge signal* (IACK).
- This clears the interrupt request from that device and allows other devices to use the interrupt line.



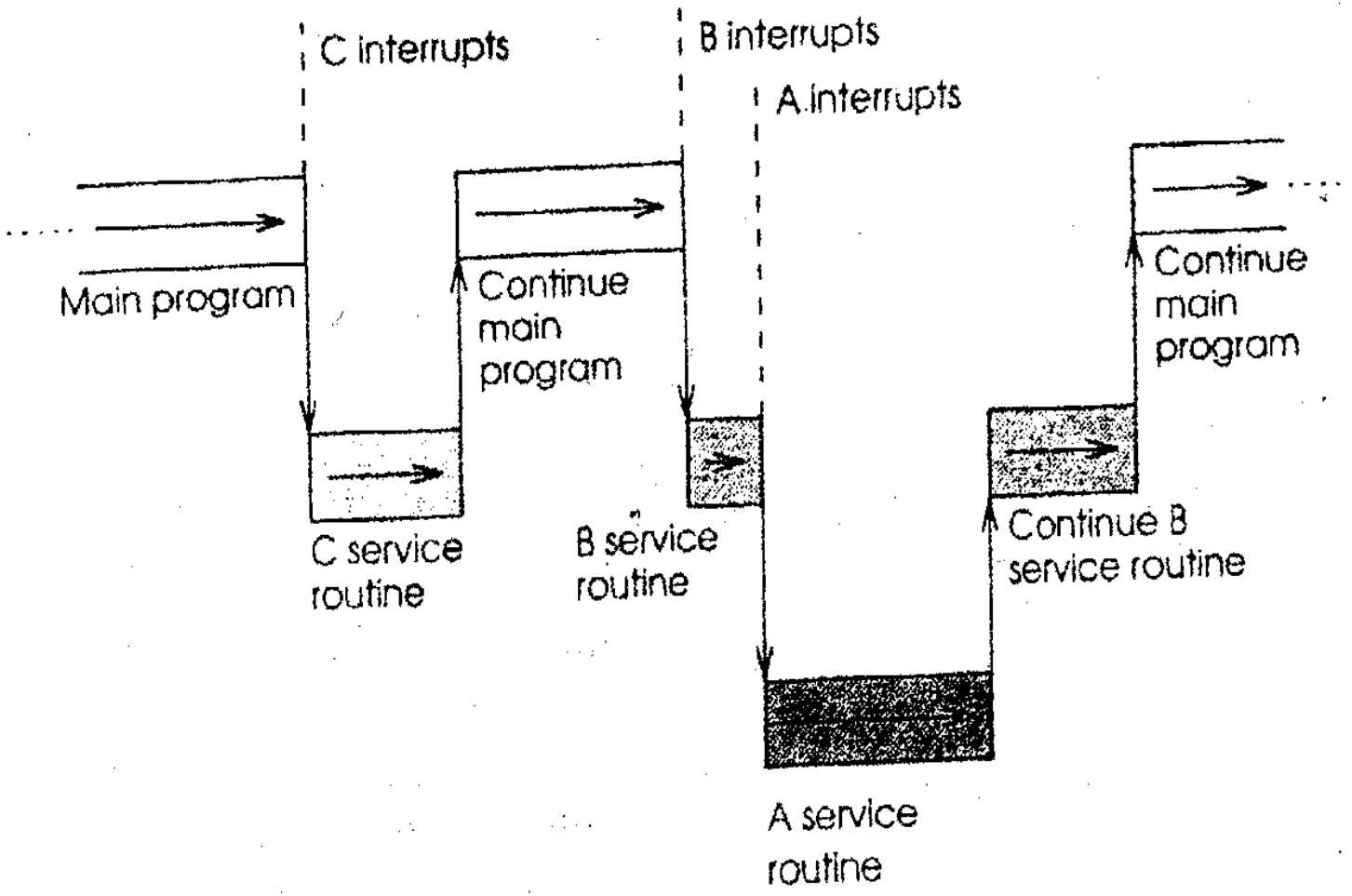
Priority Interrupt Using Daisy chain

- Using an interrupt acknowledge enables the construction of a simpler priority interrupt scheme.
- CPU is able to determine priority not from the interrupt request but by which device the acknowledge is sent to.
- All the interrupt request lines are OR together.
- The CPU IACK is connected directly to the highest priority device so that if more than one request has been made the highest priority device sees it first.
- If it has not made a request, it passes the IACK along to the next device
- This continues down to the lowest priority device.
  - This receives an acknowledgement only if no other device has made a request.

Priority Interrupt Using Daisy chain



Nested Interrupts



Block Data Transfer

- Processors have instructions for moving blocks of data which may be used for I/O.
- CPU has to synchronize its block moves with the peripheral and is unable to performs any other functions while the block moves is in progress.
- Only suited for fast transfers where the CPU and the peripheral speeds are reasonably well matched.
- It is not commonly used.

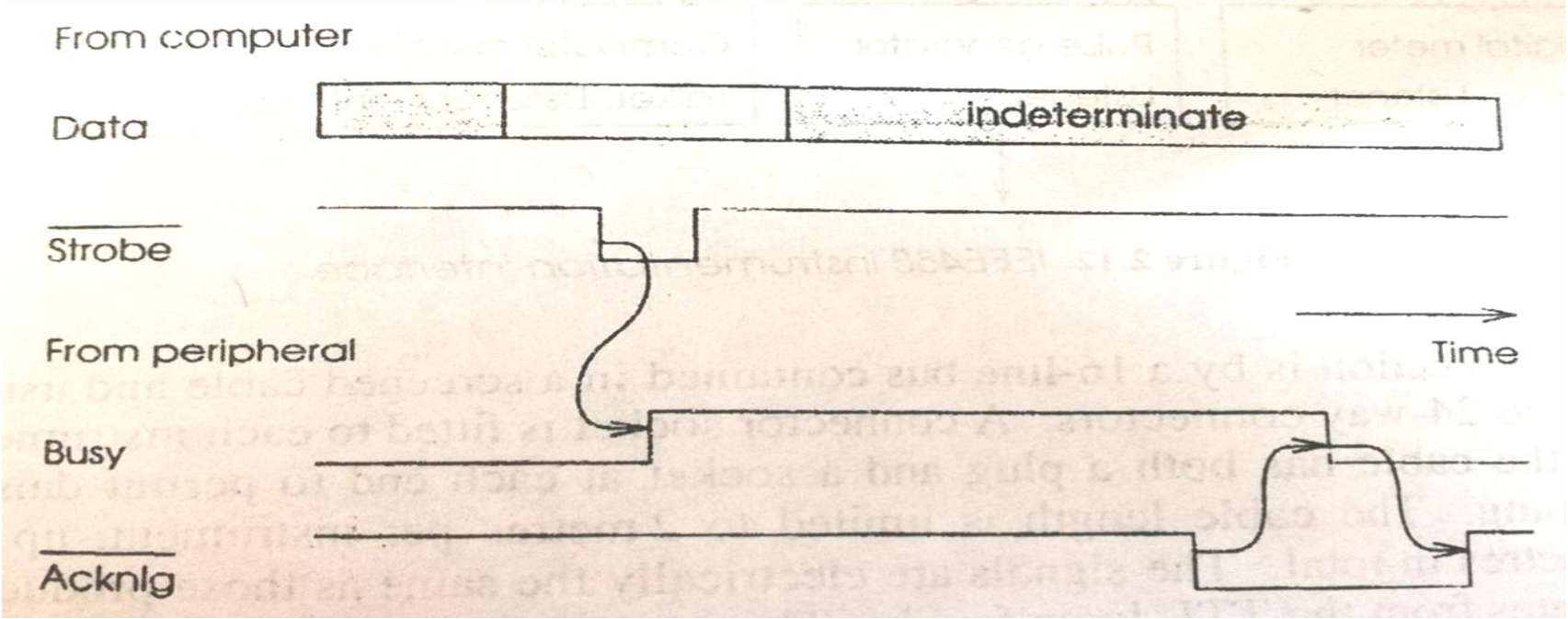
Direct Memory Access (DMA)

- It is better to avoid using the CPU completely for I/O transfer.
- A special hardware called DMA can be used to transfer data between memory and I/O devices.
- When a peripheral indicates that it is ready for data transfer
  - The DMA unit gains the control of the Bus.
  - Places appropriate address and control signals on it to make the transfer.
  - Releases the bus.
- This action of taking over the bus for a period and executing a memory access cycle instead of the CPU doing so is known as *cycle stealing.*

Parallel Interface: Centronics

- Use mainly for connecting printers to computer, this interface provides relatively high speed operation.
- All eight bits containing the output character are transmitted in parallel along twisted pair lines, together with control signals.
- A handshaking scheme is used to control the flow of data. The sequence of events is-
  - Wait until the peripheral is not busy.
  - Place data on the data lines and allow time for the signal levels to settle, at least 500 ns
  - Take the strobe line low for at least 500ns, this will result in the peripheral changing to the “busy” state. 500 ns after the end of this pulse data may be removed from the data lines.
  - When the peripheral has dealt with the data it will set the acknowledge line low to indicate this and will also clear its “busy” state ready for the next cycle.

Parallel Interface: Centronics



Parallel Interface: SCSI

- **SCSI (**Small Computer System Interface**)** is a set of standards for physically connecting and transferring data between computers and peripheral devices.

- Used to connect computers to disk drives and other peripherals (generally those needing a high data rate).

- 8 devices can be supported on the SCSI bus

  - Each device can have 8 logical units
  - Each logical unit contains a unique Logical Unit Number (LUN)
  - Each logical unit can have 256 logical sub units
    - Thus a very large total number of possible devices

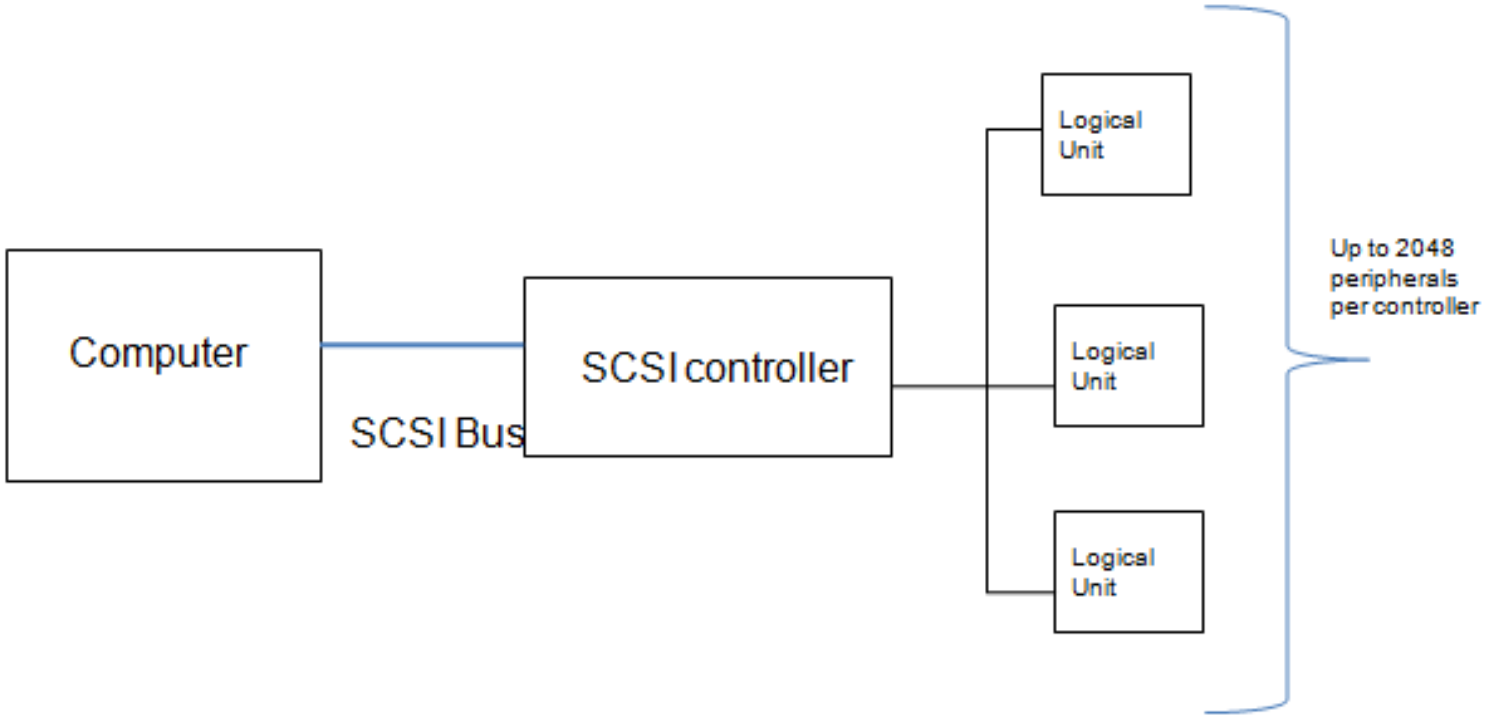
Parallel Interface: SCSI

- Each device can be either an **initiator**, a **target** or both.
  - Initiator:
    - A SCSI device that requests an operation to be performed by another SCSI device.
    - Initiate a SCSI session.
    - Does not provide any Logical Unit Number.
  - Target:
    - A SCSI device that performs an operation as requested by an initiator.
    - Waits for initiators' commands and provides required input/output data transfers.
    - Provide one or more Logical Unit Number.

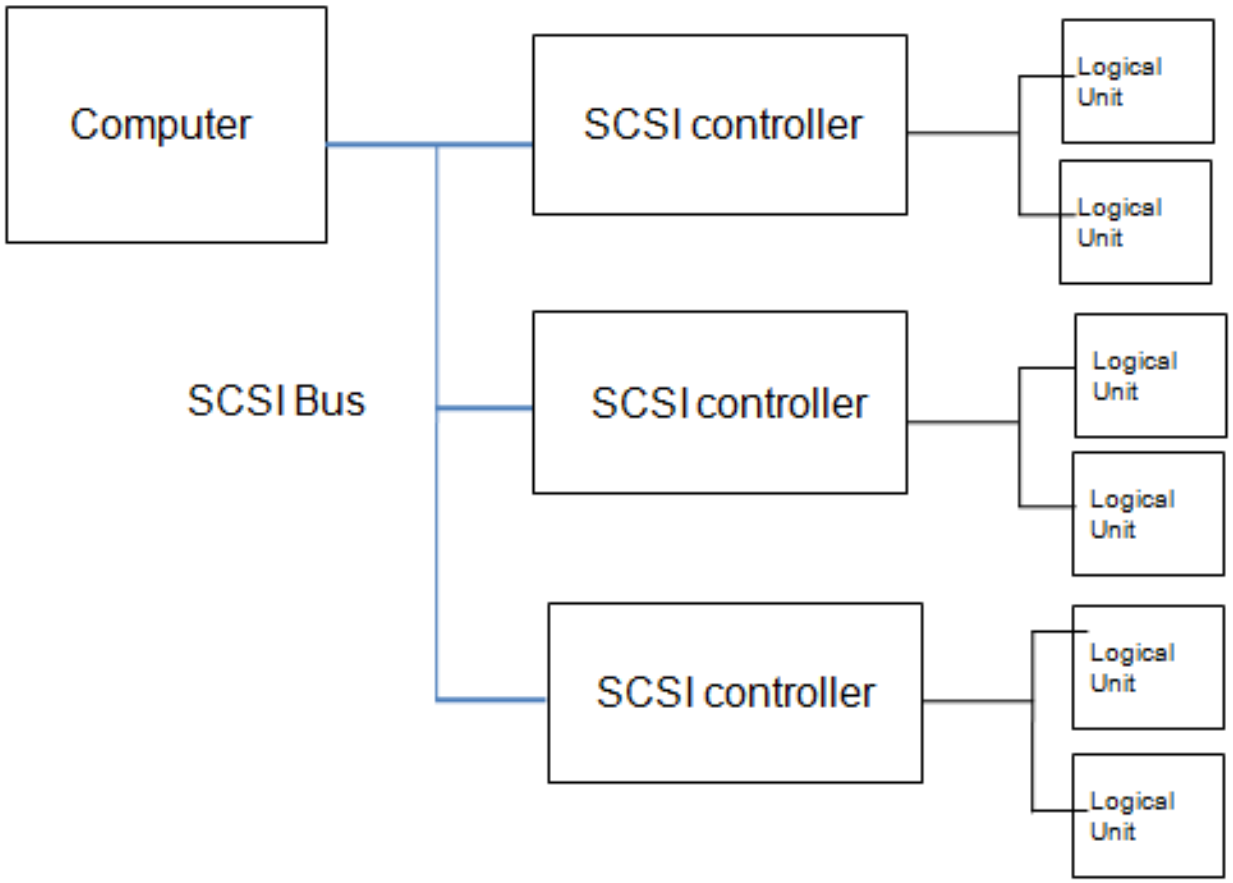
SCSI

- To be useful SCSI bus must have at least one initiator and one target
  - Can have multiple initiator and/or target
- Data is transferred over an 8 bit bidirectional bus
  - Optional parity line to check the error of transmission
- When an end user sends a request, the operating system sends the SCSI commands to the SCSI controller, which then sends it to the storage device.
- SCSI controller (embedded on motherboard) is a host bus adapter or a chip that allows a SCSI storage device to communicate with operating system across a SCSI bus.

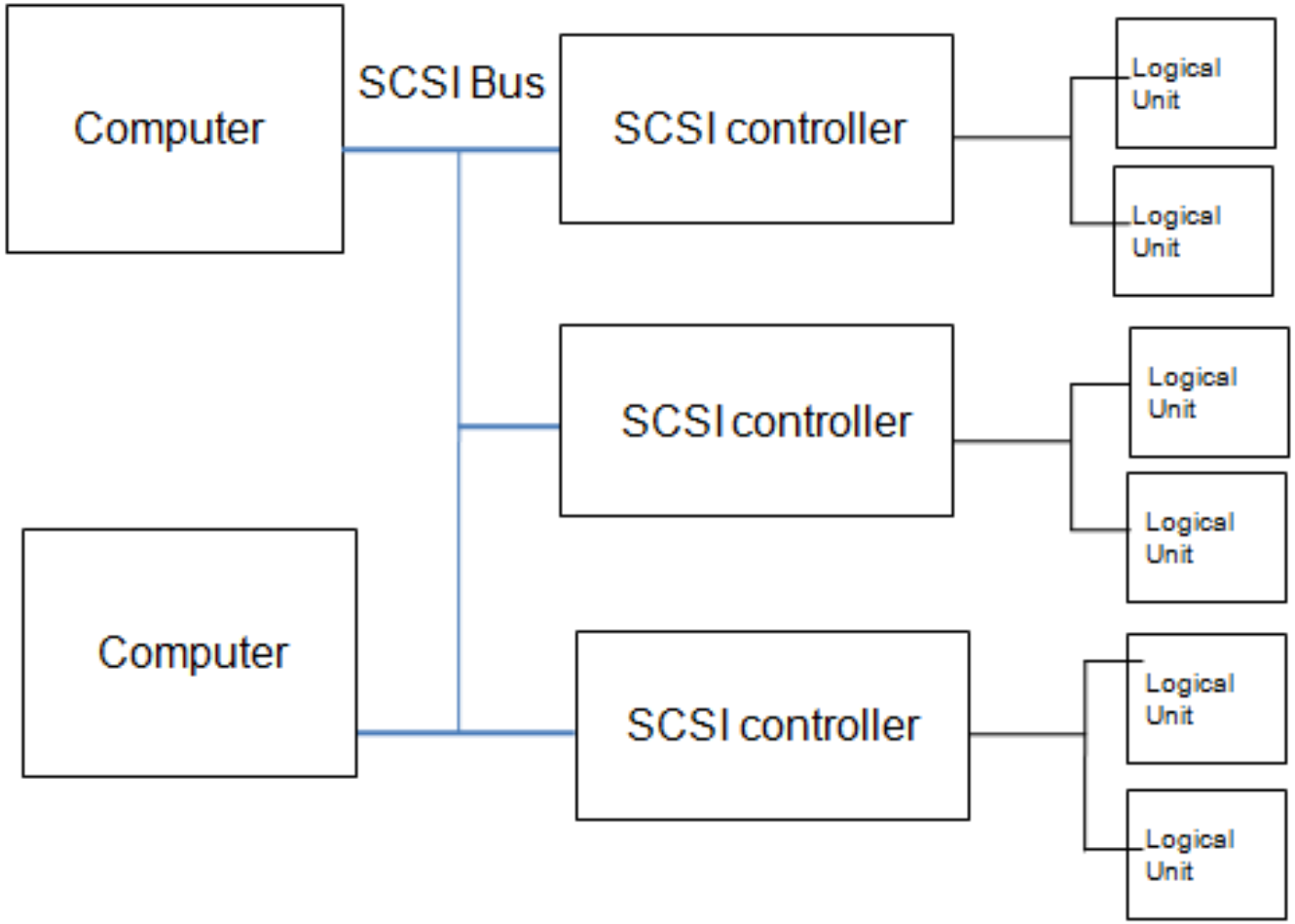
SCSI: Single initiator, Single Target



SCSI: Single Initiator, Multiple Targets



SCSI: Multiple Initiator, Multiple Targets



Protocol For Using SCSI Bus

- Phase 1 : Arbitration
- Phase 2 : Selection
- Phase 3 : Information Transfer
- Phase 4 : Bus free
- Phase 5 : Reselection

- ✓ Several commands can be received before the first is completed
- ✓ Devices can queue requests

Protocol For Using Bus

- **Phase 1: Arbitration**

  - Allows one SCSI device to gain control of the SCSI bus so that it can initiate or resume an I/O process.
  - The SCSI device shall first wait for the BUS FREE phase to occur

- **Phase 2: Selection**

  - The SELECTION phase allows an initiator to select a target for the purpose of initiating some target function.

Protocol For Using Bus

- **Phase 3: Information transfer**

  - The COMMAND, DATA, STATUS, and MESSAGE phases are combined to transfer data or control information.

- **Phase 4: Bus Free**

  - BUS FREE phase is entered when a target release the BUSY signal
  - The BUS FREE phase indicates that there is no current I/O process and the SCSI bus is available for a new connection.

Protocol For Using Bus

- **Phase 5: Reselection**

  - When the target is ready to respond to the given command then it can re-establish contact with the initiator in Reselection phase.
  - Connection that started by the initiator but was suspended by the target.

SCSI: Mathematical Problem-1

Initiator	Target	Arbitration time of initiator (sec)	Task time of target given by initiator (sec)
2	1	4	8.4
1	3	5.1	5
1	2	6.4	5.8
1	3	8	3.5
2	2	8.2	4.9

Suppose, 2 computers are sharing a common SCSI bus among them. 3 SCSI controllers are connected with the same bus. Now solve the following scenario and find out the value of t at which all the controllers and bus will become free after executing all the tasks given by the all the computers.

Consider ,data transfer rate between all components = 1.7 sec
And arbitration +selection/reselection time= 2 sec

SCSI: Mathematical Problem-2

Initiator	Target	Arbitration time of initiator (sec)	Task time of target given by initiator (sec)
2	3	1	10
1	2	3	8
1	3	6	4
3	1	8	20
3	3	12	9

Suppose, 3 computers are sharing a common SCSI bus among them. 3 SCSI controllers are connected with the same bus. Now solve the following scenario and find out the value of t at which all the controllers and bus will become free after executing all the tasks given by the all the computers.

Consider ,data transfer rate between all components = 2 sec
And arbitration +selection/reselection time= 1 sec

Serial Interface

- Synchronous Transmission (Self Study)
- Asynchronous Transmission (Self Study)

The RS232 Standard

- A standard used for serial data transmission.
- Communication is assumed to be between Data Terminal Equipment (DTE) and Data Communication Equipment (DCE).
- DTE can be a computer or peripheral, DCE is a non- existent device such as a modem.
- Standard telephone line can be used due to its availability.

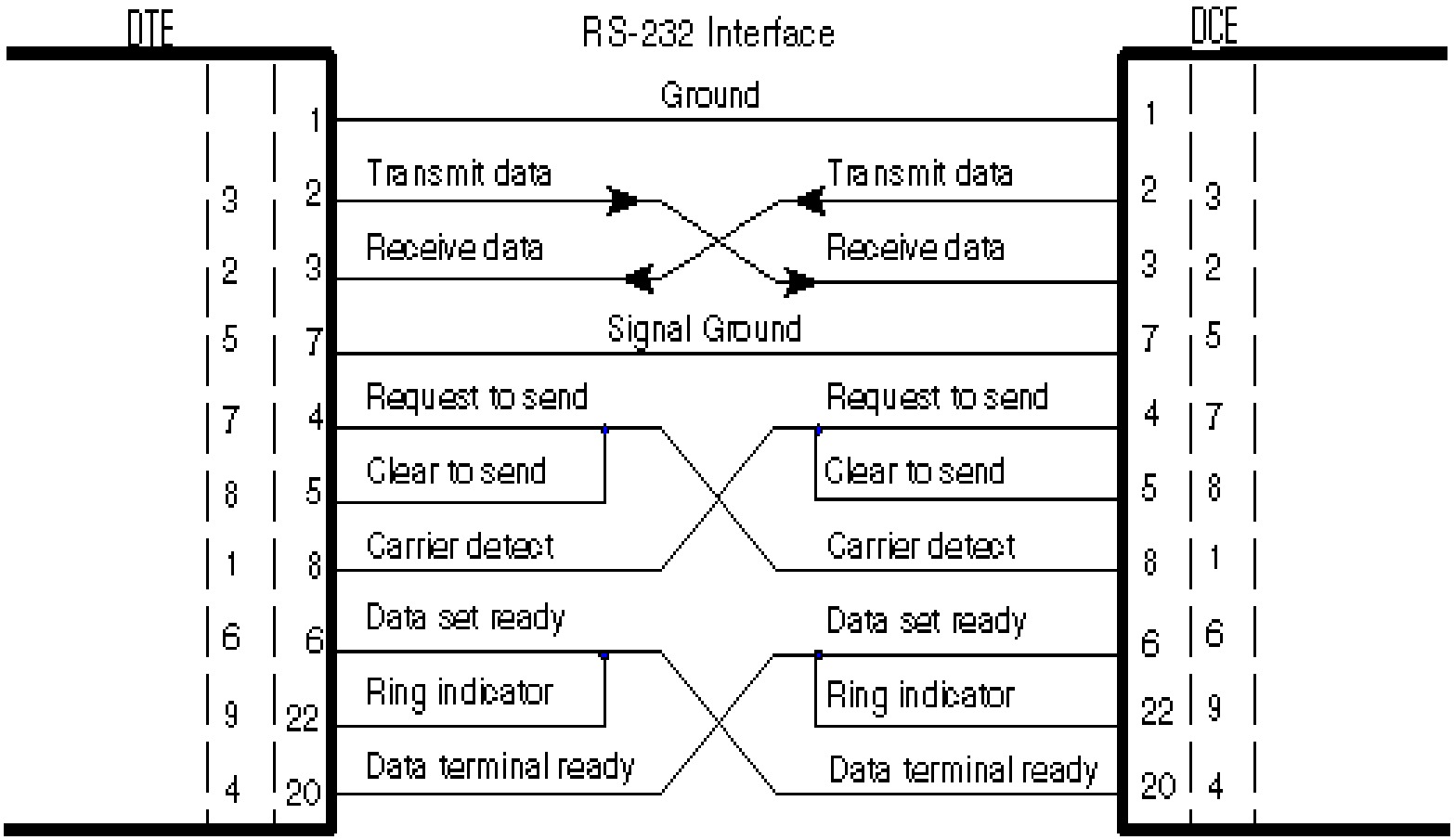
The RS232 Standard

Protocol:

- When the terminal is powered on, it performs a self-checking and asserts DTR signal to tell the modem that it is ready.
- When modem is ready, it sends DSR signal to the terminal.
- Then the modem dials the computer.
- If the computer is available, it sends specified tone.
- When terminal is ready to send, it asserts RTS signal.
- Modem asserts CD signal to indicate that it has established contact with computer.
- When modem is ready, it asserts CTS signal.
- Terminal sends data.
- When all data has been sent, terminal raises, RTS signal high.
- Modem raises CTS signal high.

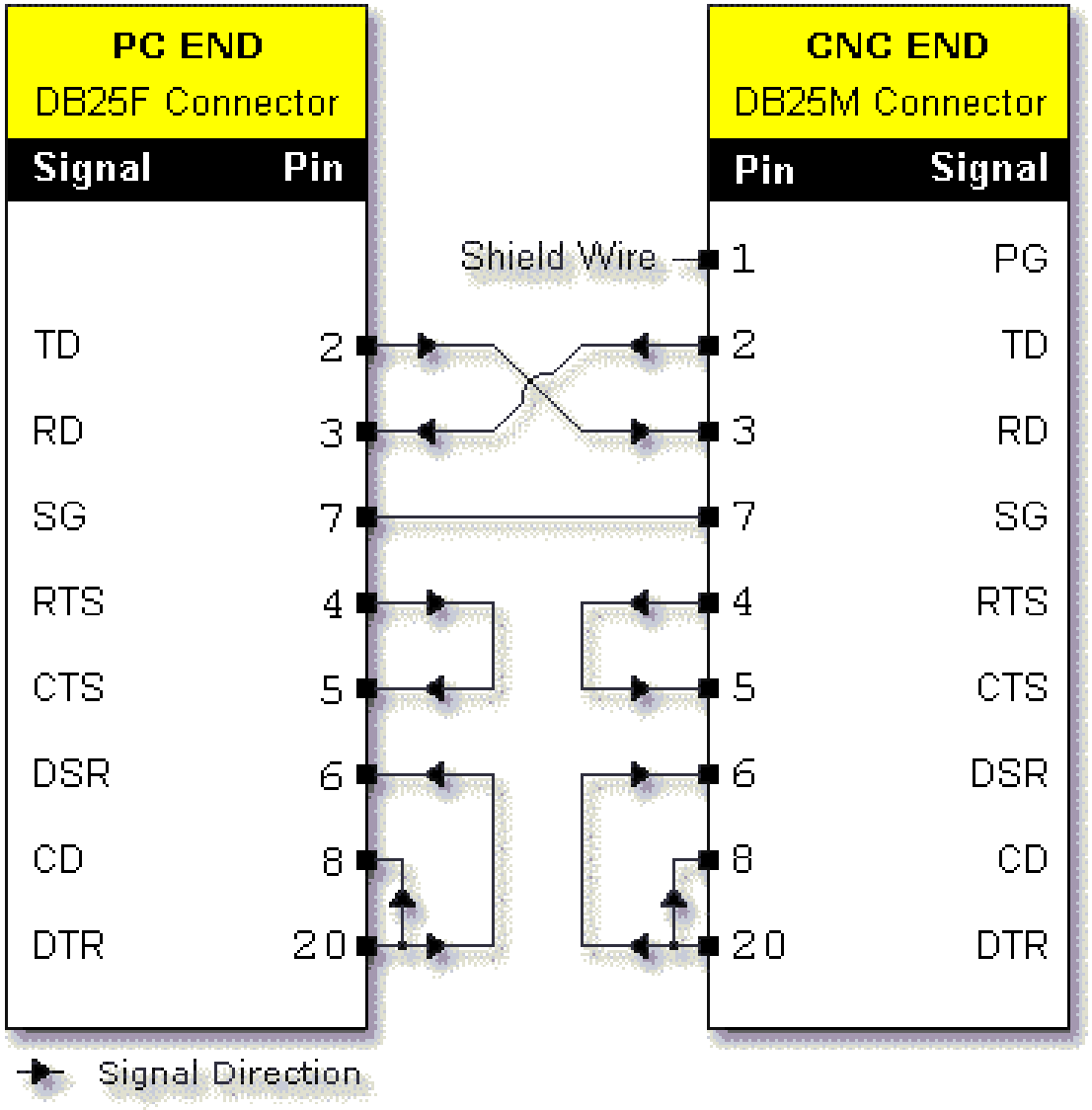
The RS232 Standard

- It is not possible to connect TTL compatible device to RS232
- Solution:
  - MC1488 – convert TTL to RS232
  - MC1489- convert RS232 to TTL


DTE
RS-232 Interface
DCE
Ground
Transmit data
Transmit data
Receive data
Receive data
Signal Ground
Request to send
Request to send
Clear to send
Clear to send
Carrier detect
Carrier detect
Data set ready
Data set ready
Ring indicator
Ring indicator
Data terminal ready
Data terminal ready
1
2
3
4
5
6
7
8
9
20
22
1
2
3
4
5
6
7
8
9
20
22
3
2
5
7
8
1
6
9
4
3
2
7
5
4
7
8
1
6
9
4

Null Modem Connection

- A Null Modem is used to connect two DTE's together
- Any data transmitted from the first computer must be received by the second thus TD is connected to RD. The second computer must have the same set-up thus RD is connected to TD.
- Signal Ground (SG) must also be connected so both grounds are common to each computer.
- The Data Terminal Ready is looped back to Data Set Ready and Carrier Detect on both computers.
- For Request to Send and Clear To Send, as both computers communicate together at the same speed, flow control is not needed thus these two lines are also linked together on each computer. When the computer wishes to send data, it asserts the Request to Send high and as it's hooked together with the Clear to Send, It immediately gets a reply that it is ok to send and does so.


PC END
DB25F Connector
Signal
Pin
TD
2
RD
3
SG
7
RTS
4
CTS
5
DSR
6
CD
8
DTR
20
Shield Wire
CNC END
DB25M Connector
Pin
Signal
1
PG
2
TD
3
RD
7
SG
4
RTS
5
CTS
6
DSR
8
CD
20
DTR
Signal Direction

Assignment

USB Interface

Thank You