Information generated by a source need to be encoded into a suitable format for transmission. To transmit the encoded signals generated by the Information-processing equipment over a communication link, assistance is needed. For example, a PC generates a digital signal but needs an additional device to modulate a carrier frequency before it is sent over a telephone line. Encoded data is sent from the generating device to the next device by a bundle of wires, a sort of mini communication link, called an interface.

Because an interface links two devices not necessarily made by the same manufacturer, its characteristics must be defined and standards must be established. Characteristics of an interface include its mechanical specifications (how many wires are used to transport the signal), its electrical specifications (the frequency, amplitude, and phase of the expected signal), and its functional specifications (if multiple wires are used, what does each one do?). These characteristics are all described by several popular standards and are incorporated in the physical layer of the OSI model.

**Data transmission**

The transmission of binary data across a link can be accomplished either in parallel mode or serial mode. In parallel mode, multiple bits are sent with each clock pulse. In serial mode, one bit is sent with each clock pulse. While there is only one way to send parallel data, there are two subclasses of serial transmission: synchronous and asynchronous.
**Parallel Transmission**

Binary data, consisting of 1s and 0s, may be organised into groups of \( n \) bits each. By grouping, we can send data \( n \) bits at a time instead of one. This is called **parallel transmission**.

We use \( n \) wires to send \( n \) bits at one time. That way each bit has its own wire, and all \( n \) bits of one group can be transmitted with each clock pulse from one device to another. The Figure bellow shows how parallel transmission works for \( n = 8 \). Typically, the eight wires are bundled in a cable with a connector at each end.

The advantage of parallel transmission is speed. All else being equal, parallel transmission can increase the transfer speed by a factor of \( n \) over serial transmission. A significant disadvantage of parallel transmission is cost. Parallel transmission requires \( n \) communication lines (wires in the example) just to transmit the data stream. Because this is expensive, parallel transmission is usually limited to short distances.

**Serial Transmission**

**In serial transmission** one bit follows another, so we need only one communication channel rather than \( n \) to transmit data between two communicating devices.

The advantage of serial over parallel transmission is that with only one communication channel, serial transmission reduces the cost of transmission over parallel by roughly a factor of \( n \).

Since communication within devices is parallel, conversion devices are required at the interface between the sender and the line (parallel-to-serial) and between the line and the receiver (serial-to-parallel).
Serial transmission occurs in one of two ways; asynchronous or synchronous.

**Asynchronous Transmission**

Asynchronous transmission is so named because the timing of a signal is unimportant. Instead, information is received and translated by agreed-upon patterns. Patterns are based on grouping the bit stream into bytes. Each group, usually eight bits, is sent along the link as a unit.

To alert the receiver to the arrival of a new group, an extra bit is added to the beginning of each byte. This bit, usually a 0, is called the **start bit**. To let the receiver know that the byte is finished, one or more additional bits are appended to the end of the byte. These bits, usually 1s, are called **stop bits**. By this method, each byte is increased in size to at least 10 bits, of which 8 are information and 2 or more are signals to the receiver. In addition, the transmission of each byte may then be followed by a gap of varying duration. This gap can be represented either by an idle channel or by a stream of additional stop bits.

In asynchronous transmission, we send one start bit (0) at the beginning and one or more stop bits (1s) at the end of each byte. There may be a gap between each byte.

The start and stop bits and the gap alert the receiver to the beginning and end of each byte and allow it to synchronise with the data stream. This **mechanism** is called asynchronous because, at the byte level, sender and receiver do not have to be synchronised. But within each byte, the receiver must still be synchronised with the incoming bit stream. That is, some synchronisation is required, but only for the duration of a single byte. The receiving device resynchronises at the onset of each new byte. When the receiver detects a start bit, it sets a
timer and begins counting bits as they come in. After $n$ bits, the receiver looks for a stop bit. As soon as it detects the stop bit it ignores any received pulses until it detects the next start bit. **Asynchronous here means “asynchronous at the byte level,” but the bits are still synchronised; their durations are the same.**

The following Figure is a schematic illustration of asynchronous transmission. In this example, the start bits are 0s, the stop bits are 1s, and the gap is represented by an idle line rather than by additional stop bits.

**Asynchronous transmission**

Asynchronous transmission is slower than other forms of transmission because of the addition of control information. But it is cheap and effective, two advantages that make it an attractive choice for situations like low-speed communication. For example, the connection of a terminal to a computer is a natural application for asynchronous transmission. A user types only one character at a time, types extremely slowly in data processing terms, and leaves unpredictable gaps of time between each character.

**Synchronous Transmission**

In **synchronous transmission**, the bit stream is combined into longer "frames," which may contain multiple bytes. Each byte, however, is introduced onto the transmission link without a gap between it and the next one. It is left to the receiver to separate the bit stream into bytes for decoding purposes. In other words, data are transmitted as an unbroken string of 1s and 0s, and the receiver separates that string into the bytes, or characters, it needs to reconstruct the information.

In synchronous transmission, we send bits one after another without start/stop bits or gaps. It is the responsibility of the receiver to group the bits.
The following Figure gives a schematic illustration of synchronous transmission. We have drawn in the divisions between bytes. In reality, those divisions do not exist; the sender puts its data onto the line as one long string. The receiver counts the bits as they arrive and groups them in eight-bit units.

**Synchronous transmission**

The advantage of synchronous transmission is speed. With no extra bits or gaps to introduce at the sending end and remove at the receiving end and, by extension, with fewer bits to move across the link, synchronous transmission is faster than asynchronous transmission. For this reason, it is more useful for high-speed applications like the transmission of data from one computer to another.

**DTE-DCE Interface**

Two terms are important to computer networking: data terminal equipment (DTE) and data circuit-terminating equipment (DCE). There are usually four basic functional units involved in the communication of data: a DTE and DCE on one end and a DCE and DTE on the other end as shown in the Figure below.

**Data Terminal Equipment (DTE)**

Data terminal equipment (DTE) includes any unit that functions either as a source of or as a destination for binary digital data. At the physical layer, it can be a terminal, microcomputer, computer, printer, fax machine, or any other device that generates or consumes digital data. DTEs do not often communicate directly with one another; they generate and consume information but need an intermediary to be able to communicate. Think of a DTE as operating the way your brain does when you talk. Let's say you have an idea that you want to communicate to a friend. Your brain creates the idea but cannot transmit that idea to your friend's brain by itself. Unfortunately or fortunately, we are not a species of mind readers.
Instead, your brain passes the idea to your vocal chords and mouth, which convert it to sound waves that can travel through the air or over a telephone line to your friend's ear and from there to his or her brain, where it is converted back into information. In this model, your brain and your friend's brain are DTEs. Your vocal chords and mouth are your DCE. His or her ear is also a DCE. The air or telephone wire is your transmission medium.

*A DTE is any device that is a source of or destination for binary digital data.*

**Data Circuit-Terminating Equipment (DCE)**

Data circuit-terminating equipment (DCE) includes any functional unit that transmits or receives data in the form of an analog or digital signal through a network. At the physical layer, a DCE takes data generated by a DTE, converts them to an appropriate signal, and then introduces the signal onto the telecommunication link. Commonly used DCEs at this layer include modems (modulator/demodulators). In any network, a DTE generates digital data and passes them to a DCE; the DCE converts the data to a form acceptable to the transmission medium and sends the converted signal to another DCE on the network. The second DCE takes the signal off the line, converts it to a form usable by its DTE, and delivers it.

To make this communication possible, both the sending and receiving DCEs must use the same modulating method (e.g., FSK). The two DTEs do not need to be coordinated with each other, but each must be coordinated with its own DCE and the DCEs must be coordinated so that data translation occurs without loss of integrity.

*A DCE is any device that transmits or receives data in the form of an analog or digital signal through a network.*

**Standards**

Over the years, many standards have been developed to define the connection between a DTE and a DCE (see Figure 6.7). Though their solutions differ, each standard provides a model for the mechanical, electrical, and functional characteristics of the connection.

**DTE-DCE interface**
Of the organisations involved in DTE-DCE interface standards, the most active are the Electronic Industries Association (EIA) and the International Telecommunication Union-Telecommunication Standards Committee (ITU-T). The EIA (previously RS) standards are called, appropriately enough, EIA-232, EIA-442, EIA-449, and so on. The ITU-T standards are called the V series and the X series.

*The EIA and the ITU-T have been involved in developing DTE-DCE interface standards. The EIA standards are called EIA-232, EIA-442, EIA-449, and so on. The ITU-T standards are called the V series and the X series.*

**EIA-232 Interface (RS-232)**

One important interface standard developed by the EIA is the **EIA-232**, (previously RS-232, Recommended Standard) which defines the mechanical, electrical, and functional characteristics of the interface between a DTE and a DCE. Originally issued in 1962 as the RS-232 standard (recommended standard), the EIA-232 has been revised several times. The most recent version, EIA-232-D, defines not only the type of connectors to be used but also the specific cable and plugs and the functionality of each pin.

**EIA-232 (previously called RS-232) defines the mechanical, electrical, and functional characteristics of the interface between a DTE and a DCE.**

**Mechanical Specification**

The mechanical specification of the EIA-232 standard defines the interface as a 25-wire cable with a male and a female DB-25 pin connector attached to either end. The length of the cable may not exceed 15 meters (about 50 feet).

A **DB-25** connector is a plug with 25 pins or receptacles, each of which is attached to a single wire with a specific function. With this design, the EIA has created the possibility of 25 separate interactions between a DTE and a DCE. Fewer are actually used in current practice, but the standard allows for future inclusion of functionality.

The EIA-232 calls for a 25-wire cable terminated at one end by a male connector and at the other end by a female connector. The term *male connector* refers to a plug with each wire in the cable connecting to a pin. The term *female connector* refers to a receptacle with each wire in the cable connecting to a metal tube, or sheath. In the DB-25 connector, these pins and tubes are arranged in two rows, with 13 on the top and 12 on the bottom.

Another implementation of EIA-232 uses a 9-wire cable with a male and a female DB-9 pin connector attached to either end.
Electrical Specification
The electrical specification of the standard defines the voltage levels and the type of signal to be transmitted in either direction between the DTE and the DCE,

Sending the Data
The electrical specification for sending data is shown in the Figure bellow. EIA-232 states that all data must be transmitted as logical 1s and 0s (called mark and space) using NRZ-L encoding, with 0 defined as a positive voltage and 1 defined as a negative voltage. However, rather than defining a single range bounded by highest and lowest amplitudes, EIA-232 defines two distinct ranges, one for positive voltages and one for negative. A receiver recognizes and accepts as an intentional signal any voltage that falls within these ranges, but no voltages that fall outside the ranges. To be recognized as data, the amplitude of a signal must fall between 3 and 15 volts or between -3 and -15 volts. By allowing valid signals to fall within two 12-volt ranges, EIA-232 makes it unlikely that degradation of a signal by noise will affect its recognisability. In other words, as long as a pulse falls within one of the acceptable ranges, the precision of that pulse is unimportant.

Electrical specification for sending data in EIA-232
The above Figure shows a square wave degraded by noise into a curve. The amplitude of the fourth bit is lower than intended (compared to that of the second bit), and rather than staying at one single voltage, it covers a range of many voltages. If the receiver were looking only for a fixed voltage, the degradation of this pulse would have made it unrecoverable. The bit also would have been unrecoverable if the receiver were looking only for pulses that held a single voltage for their entire duration.
Control and Timing

Only 4 wires out of the 25 available in an EIA-232 interface are used for data functions. The remaining 21 are reserved for functions like control, timing, grounding, and testing. The electrical specifications for these other wires are similar to those governing data transmission, but simpler. Any of the other functions is considered ON if it transmits a voltage of at least +3 and OFF if it transmits a voltage with a value less than -3 volts.

The electrical specification of EIA-232 defines that signals other than data must be sent using OFF less than -3 volts and ON greater than +3 volts.

The following Figure shows one of these signals. The specification for control signals is conceptually reversed from that for data transmission. A positive voltage means ON and a negative voltage means OFF. Also note that OFF is still signified by the transmission of a specific voltage range. An absence of voltage on one of these wires while the system is running means that something is not working properly, and not that the line is turned off.

A final important function of the electrical specification is the definition of bit rate. EIA-232 allows for a maximum bit rate of 20 Kbps, although in practice this often is exceeded.

![Electrical specification for control signals in EIA-232](image)

Functional Specification

Two different implementations of EIA-232 are available: DB-25 and DB-9.

**DB-25 Implementation** EIA-232 defines the functions assigned to each of the 25 pins in the DB-25 connector. The Figure below shows the ordering and functionality of each pin of a male connector. Remember that a female connector will be the mirror image of the male, so that pin 1 in the plug matches tube 1 in the receptacle, and so on.

---

Com 318-07/08 Fall
Functions of pins in EIA-232, DB-25

1. Shield
2. Transmitted data
3. Received data
4. Request to send
5. Clear to send
6. DCE ready
7. Signal ground common return
8. Received line signal detector
9. Reserved (testing)
10. Reserved (testing)
11. Unassigned
12. Secondary received line signal detector
13. Secondary clear to send
14. Secondary transmitted data
15. Transmitter signal element timing (DCE-DTE)
16. Secondary received data
17. Receiver signal element timing (DCE-DTE)
18. Local loopback
19. Secondary request to send
20. DTE ready
21. Remote loopback & signal quality detector
22. Ring detector
23. Data signal rate select
24. Transmitter signal element timing (DTE-DCE)
25. Test mode

DB-9 Implementation Many of the pins on the DB-25 implementation are not necessary in a single asynchronous connection. A simpler 9-pin version of EIA-232 known as DB-9 and shown in the following Figure was developed. Note that there is no pin-to-pin relationship in the two implementations.
This example demonstrates the functioning of EIA-232 in synchronous full-duplex mode over a leased line using only the primary channel. The DCEs here are modems, and the DTEs are computers. There are five distinct steps, from preparation to clearing.

This is a full-duplex model, so both computer/modem systems can transmit data concurrently. In terms of the EIA model, however, one system is still classified as the initiator and the other as the responder.

**Step 1** shows the preparation of the interfaces for transmission. The two grounding circuits, 1 (shield) and 7 (signal ground), are active between both the sending computer/modem combination (left) and the receiving computer/modem combination (right).

**Step 2** ensures that all four devices are ready for transmission. First the sending DTE activates pin 20 and sends a DTE ready message to its DCE. The DCE answers by activating pin 6 and returning a DCE ready message. This same sequence is performed by the remote computer and modem.

**Step 3** sets up the physical connection between the sending and receiving modems. This step can be thought of as the on switch for transmission. It is the first step that involves the network. First, the sending DTE activates pin 4 and sends its DCE a request-to-send message. The DCE transmits a carrier signal to the idle receiving modem. When the receiving modem detects the carrier signal, it activates pin 8, the received line signal detector, telling its computer that a transmission is about to begin. After transmitting the carrier signal, the sending DCE activates pin 5, sending its DTE a clear-to-send message. The remote computer and modem perform the same step.
Step 4 is the data transfer procedure. The initiating computer transfers its data stream to its modem over circuit 2, accompanied by the timing pulse of circuit 24. The modem converts the digital data to an analog signal and sends it out over the network. The responding modem retrieves the signal, converts it back into digital data and passes the data along to its computer via circuit 3, accompanied by the timing pulse of circuit 17. Likewise, the responding computer follows the same procedure in sending data to the initialising computer.

Step 5 Once both sides have completed their transmissions, both computers deactivate their request-to-send circuits; the modems turn off their carrier signals, their received line signal detectors (there is no longer any signal to detect), and their clear-to-send circuits.

**Null Modem**

Suppose you need to connect two DTEs in the same building, for example, two workstations or a terminal to a workstation. Modems are not needed to connect two compatible digital devices directly; the transmission never needs to cross analog lines, such as telephone lines, and therefore does not need to be modulated. But you do need an interface to handle the exchange (readiness establishment, data transfer, receipt, etc.), just as an EIA-232 DTE-DCE cable does.

The solution, provided by the EIA standard, is called a null modem. A null modem provides the DTE-DTE interface without the DCEs. But why use a null modem? If all you need is the interface, why not just use a standard EIA-232 cable? To understand the problem, examine Figure 6.13. Part a shows a connection using a telephone network. The two DTEs are exchanging information through DCEs. Each DTE sends its data through pin 2 and the DCE receives it on pin 2; and each DTE receives data through pin 3 that has been forwarded by the DCE using its own pin 3. As you can see, the EIA-232 cable connects DTE pin 2 to DCE pin 2 and DCE pin 3 to DTE pin 3. Traffic using pin 2 is always outgoing from the DTEs. Traffic using pin 3 is always incoming to the DTEs. A DCE recognizes the direction of a signal and passes it along to the appropriate circuit.

Without DCEs to switch the signals to or from the appropriate pins, both DTEs are attempting to transmit over the same pin 2 wire—and to receive over the same pin 3 wire. The DTEs are transmitting to each other's transmit pins, not to their receive pins. The receive circuit (3) is void because it has been isolated completely from the transmission. The transmit circuit (2) therefore ends up full of collision noise and signals that can never be received by either DTE. No data can get through from one device to another.
Crossing Connections  For transmission to occur, the wires must be crossed so that pin 2 of the first DTE connects to pin 3 of the second DTE and pin 2 of the second DTE connects to pin 3 of the first. These two pins are the most important. Several other pins, however, have similar problems and also need rewiring.

A null modem is an EIA-232 interface that completes the necessary circuits to fool the DTEs at either end into believing that they have DCEs and a network between them. Because its purpose is to make connections, a null modem can be either a length of cable or a device, or you can make one yourself using a standard EIA-232 cable and a breakout box that allows you to cross-connect wires in any way you desire. Of these options, the cable is the most commonly used and the most convenient.

Other Differences  Whereas an EIA-232 DTE-DCE interface cable has a female connector at the DTE end and a male connector at the DCE end, a null modem has female connectors at both ends to allow it to connect to the EIA-232 DTE ports, which are male

Other Interface Standards
Both data rate and cable length (signal distance capability) are restricted by EIA-232! data rate to 20 Kbps and cable length to 50 feet (15 meters). To meet the needs of user who require more speed and/or distance, the EIA and the ITU-T have introduced additional interface standards: EIA-449, EIA-530, and X.21.
**EIA-449**

The mechanical specifications of EIA-449 define a combination of two connectors; one with 37 pins (DB-37) and one with 9 pins (DB-9), for a combined 46 pins (see Figure 6.15). The functional specifications of the EIA-449 give the DB-37 pins properties similar to those of the DB-25. The major functional difference between the 25- and 37-pin connectors is that all functions relating to the secondary channel have been removed from DB-37. Because the secondary channel is seldom used, EIA-449 separates those functions out and puts them in the second, 9-pin connector (DB-9), this way a second channel is available to systems that need it.

**DB-37 Pin Functions**

To maintain compatibility with EIA-232, EIA-449 defines two categories of pins to be used in exchanging data, control, and timing information.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
<th>Category</th>
<th>Pin</th>
<th>Function</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shield</td>
<td></td>
<td>20</td>
<td>Receive Common</td>
<td>II</td>
</tr>
<tr>
<td>2</td>
<td>Signal rate</td>
<td></td>
<td>21</td>
<td>Unassigned</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>Unassigned</td>
<td></td>
<td>22</td>
<td>Send data</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>Send data</td>
<td>I</td>
<td>23</td>
<td>Send timing</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>Send timing</td>
<td>I</td>
<td>24</td>
<td>Receive data</td>
<td>I</td>
</tr>
<tr>
<td>6</td>
<td>Receive data</td>
<td>I</td>
<td>25</td>
<td>Request to send</td>
<td>I</td>
</tr>
<tr>
<td>7</td>
<td>Request to send</td>
<td>I</td>
<td>26</td>
<td>Receive timing</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>Receive timing</td>
<td>I</td>
<td>27</td>
<td>Clear to send</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>Clear to send</td>
<td>I</td>
<td>28</td>
<td>Terminal in service</td>
<td>II</td>
</tr>
<tr>
<td>10</td>
<td>Local loopback</td>
<td>II</td>
<td>29</td>
<td>Data mode</td>
<td>I</td>
</tr>
<tr>
<td>11</td>
<td>Data mode</td>
<td>I</td>
<td>30</td>
<td>Terminal ready</td>
<td>I</td>
</tr>
<tr>
<td>12</td>
<td>Terminal ready</td>
<td>I</td>
<td>31</td>
<td>Receive ready</td>
<td>I</td>
</tr>
<tr>
<td>13-</td>
<td>Receive ready</td>
<td>I</td>
<td>32</td>
<td>Select standby-</td>
<td>II</td>
</tr>
<tr>
<td>14</td>
<td>Remote loopback</td>
<td>II</td>
<td>33</td>
<td>Signal quality</td>
<td>I</td>
</tr>
<tr>
<td>15</td>
<td>Incoming call</td>
<td></td>
<td>34</td>
<td>New signal</td>
<td>II</td>
</tr>
<tr>
<td>16</td>
<td>Select frequency</td>
<td>II</td>
<td>35</td>
<td>Terminal timing</td>
<td>I</td>
</tr>
<tr>
<td>17</td>
<td>Terminal timing</td>
<td>I</td>
<td>36</td>
<td>Standby indicator</td>
<td>II</td>
</tr>
<tr>
<td>18</td>
<td>Test mode</td>
<td>II</td>
<td>37</td>
<td>Send common</td>
<td>II</td>
</tr>
<tr>
<td>19</td>
<td>Signal ground</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Category I Pins**

Category I includes those pins whose functions are compatible with those of EIA-232 (although most have been renamed). For each Category I pin, EIA-449 defines two pins, one
in the first column and one in the second column. For example, both pins 4 and 22 are called send data. These two pins have the equivalent functionality of pin 2 in EIA-232. Both pins 5 and 23 are called send timing, and both pins 6 and 24 are called receive data. Even more interesting, these pairs of pins are vertically adjacent to one another in the connector, with the pin from the second column occupying the position essentially below its counterpart from the first column. (Number the DB-37 connector based on the numbering of the DB-25 connector to see these relationships.) This structure is what gives **EIA-449** its power. How the pins relate will become clear later in this section, when we discuss the two alternate methods of signalling defined in the electrical specifications.

**Category II Pins**

Category II pins are those that have no equivalent in EIA-232 or have been redefined. The numbers and functions of these new pins are as follows:

- **Local loopback:** Pin 10 is used for local loopback testing.
- **Remote loopback:** Pin 14 is used for remote loopback testing.
- **Select frequency:** Pin 16 is used to choose between two different frequency rates.
- **Test mode:** Pin 18 is used to do testing at different levels.
- **Receive common:** Pin 20 provides a common signal return line for unbalanced circuits from the DCE to the DTE.
- **Terminal in service:** Pin 28 indicates to the DCE whether or not the DTE is operational.
- **Select standby:** Pin 32 allows the DTE to request the use of standby equipment in the event of failure.
- **New signal:** Pin 34 is available for multiple-point applications where a primary DTE controls several secondary DTEs. When activated, pin 34 indicates that one DTE has finished its data exchange and a new one is about to start.
- **Standby indicator:** Pin 36 provides the confirmation signal from the DCE in response to select standby (pin 32).
- **Send common:** Pin 37 provides a common signal return line for unbalanced circuits from the DTE to the DCE.

**DB-9 Pin Functions**

Table 6.2 lists the pin functions of the DB-9 connector. Note that the DB-9 connector here is different from the one discussed in EIA-232.
<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shield</td>
</tr>
<tr>
<td>2</td>
<td>Secondary receive ready</td>
</tr>
<tr>
<td>3</td>
<td>Secondary send data</td>
</tr>
<tr>
<td>4</td>
<td>Secondary receive data</td>
</tr>
<tr>
<td>5</td>
<td>Signal ground</td>
</tr>
<tr>
<td>6</td>
<td>Receive common</td>
</tr>
<tr>
<td>7</td>
<td>Secondary request to send</td>
</tr>
<tr>
<td>8</td>
<td>Secondary clear to send</td>
</tr>
<tr>
<td>9</td>
<td>Send common</td>
</tr>
</tbody>
</table>

**Electrical Specifications: RS-423 and RS-422**

EIA-449 uses two standards to define its electrical specifications: **RS-423** (for unbalanced circuits) and RS-422 (for balanced circuits).

**RS-423: Unbalanced Mode**

**RS-423** is an unbalanced circuit specification, meaning that it defines only one line for propagating a signal. All signals in this standard use a common return (or ground) to complete the circuit. In unbalanced-circuit mode, EIA-449 calls for the use of only the first pin of each pair of Category I pins and all Category II pins.

**Distance** | **Data rate**
---|---
40 ft | 100 kbps
4000 ft | 1 kbps

**RS-423: Unbalanced mode**

**RS-422: Balanced Mode**

**RS-422** is a balanced circuit specification, meaning that it defines two lines for the propagation of each signal. Signals again use a common return (or ground) for the return of the signal. In balanced mode, EIA-449 utilizes all pairs of pins in Category I but does not use
the Category II pins. As you can see from the electrical specifications for this standard, the ratio of data rate to distance is much higher than that of the unbalanced standard or of EIA-232: 10 Mbps for transmissions of 40 feet.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Data rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 ft</td>
<td>10 Mbps</td>
</tr>
<tr>
<td>4000 ft</td>
<td>1 kbps</td>
</tr>
</tbody>
</table>

RS-422: Balanced mode

They do not, however, carry identical signals. The signal on one line is the complement of the signal on the other. When plotted, the complement looks like a mirror image of the original signal (see Figure 6.17). Instead of listening to either actual signal, the receiver detects the differences between the two. This mechanism makes a balanced circuit less susceptible to noise than an unbalanced circuit and improves performance.

As the complementary signals arrive at the receiver, they are put through a subtracter (a differential amplifier). This mechanism subtracts the second signal from the first before interpretation. Because the two signals complement each other, the result of this subtraction is a doubling of the value of the first signal. For example, if at a given moment the first signal has a voltage of 5, the second signal will have a voltage of -5. The result of subtraction, therefore, is 5 - (-5), which equals 10.

If noise is added to the transmission, it impacts both signals in the same way (positive noise affects both signals positively; negative noise affects both negatively). As a result, the noise is eliminated during the subtraction process. For example, say that two volts of noise are introduced at the point where the first signal is at 5 volts and its complement is at -5 volts. The addition distorts the first signal to 7 volts, and the second to -3 volts. 7 - (-3) still equals 10. It is this ability to neutralise the effects of noise that allows the superior data rates of balanced transmission.
**EIA-530**

EIA-449 provides much better functionality than EIA-232. However, it requires a DB-37 connector that the industry has been reluctant to embrace because of the amount of investment already put into the DB-25. To encourage acceptance of the new standard, therefore, the EIA developed a version of EIA-449 that uses DB-25 pins: **EIA-530**.

The pin functions of EIA-530 are essentially those of EIA-449 Category I plus three pins from Category II (the loopback circuits). Of the EIA-232 pins, some have been omitted, including ring indicator, signal quality detector, and data signal rate selector. EIA-530 does not support a secondary circuit.

**X.21**

X.21 is an interface standard designed by the ITU-T to address many of the problems existing in the EIA interfaces and, at the same time, paves the way for all-digital communication.

When control signals are encoded using meaningful control characters from a system such as ASCII, they can be transmitted over data lines. However, X.21 elimination most of the control circuits of the EIA standards and instead directs their traffic over the data circuits. Therefore, both the DTE and the DCE must have added circuit logic that enables them to transform the control codes into bit streams that can be sent over the data line. Both also need additional logic to discriminate between Control information and data upon receipt.

X.21 is designed to work with balanced circuits at 64 kbps, a rate that is becoming the industry standard.

**DB-15 connector** (specified by X.21)

Another advantage offered by X.21 is that of timing lines to control byte synchronisation in addition to the bit synchronisation provided by the EIA standards. By adding timing pulse (pins 7 and 14), X.21 improves the overall synchronisation of transmissions.
**DB–15 pins**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shield</td>
<td>9</td>
<td>Transmit data or control</td>
</tr>
<tr>
<td>2</td>
<td>Transmit data or control</td>
<td>10</td>
<td>Control</td>
</tr>
<tr>
<td>3</td>
<td>Control</td>
<td>11</td>
<td>Receive data or control</td>
</tr>
<tr>
<td>4</td>
<td>Receive data or control</td>
<td>12</td>
<td>Indication</td>
</tr>
<tr>
<td>5</td>
<td>Indication</td>
<td>13</td>
<td>Signal element timing</td>
</tr>
<tr>
<td>6</td>
<td>Signal element timing</td>
<td>14</td>
<td>Byte timing</td>
</tr>
<tr>
<td>7</td>
<td>Byte timing</td>
<td>15</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>Signal ground</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Modem Transmission Modes**

Three transmission modes are used by modems: simplex (SDX), half-duplex (HDX), and full-duplex (FDX).

**Simplex (SDX)**

Data are sent or received in one direction only. Simplex modems are used in applications such as weather and news services wires, which send data from a central location to the newsrooms. Of newspapers and radio and TV stations over based telephone circuits. Simplex modems use full bandwidth of the telephone circuit.

**Half–duplex (HDX)**

The communication channel is shared btw. Sending and receiving stations. In this mode communications can take place in either direction, but in only one direction at a time. The time that it takes for transmission directions to change between two stations is referred to as modem turnaround time.

The disadvantage of HDX modem is that each time the direction of communication is reversed, the telephone circuit must be "turned around". Another disadvantage of HDX communication is that the receiving terminal can not provide an immediate feedback in case of errors.

**Full–duplex (FDX)**

Modems capable of operating in FDX can transmit and receive data simultaneously. In two–wire telephone circuits, this is performed by using frequency-division multiplexing (FDM), where it separates the band of channel by two. Low band and high band are located within the
pass band of the telephone lines. One modem transmits on the low band and receives on the high band; the other modem transmits on the high band and receives on the low band, thus allowing the FDX operation.

Another technique is called echo cancelling, in which both modems transmit simultaneously on the same frequency.

The FDX modems have been designated to operate in one of the two modes:
- Originate mode (Terminal originate the cell)
- Answer mode (Terminal answer the cell)

In the originate mode, transmission occurs in the low band frequencies and reception occurs in the high band frequencies. In the answer mode, transmission occurs in the high band frequencies and reception occurs in low band frequencies.

**MODEMS**

A modem stands for modulation/demodulation. A modulator converts a digital signal to an analog signal. A demodulator converts an analog to a digital signal.

The most familiar type of DCE is a modem. Both modulators and demodulators, however, do use the same techniques as digital-to-analog encoders: modulators to further encode a signal, and demodulators to decode it. A modulator treats a digital signal as a series of 1s and 0s, and so can transform it into a completely analog signal by using the digital-to-analog mechanisms of ASK, FSK, PSK, and QAM.

Each DCE must be compatible with both its own DTE and with other DCEs. A modem must use the same type of encoding (such as NRZ-L), the same voltage levels to mean the same things, and the same timing conventions as its DTE. A modem must also be able to communicate with other modems.

**Theoretical Bit Rates for Modems**

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Half-duplex</th>
<th>Full-Duplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASK, FSK, 2-PSK</td>
<td>2400</td>
<td>1200</td>
</tr>
<tr>
<td>4-PSK, 4-QAM</td>
<td>4800</td>
<td>2400</td>
</tr>
<tr>
<td>8-PSK, 8-QAM</td>
<td>7200</td>
<td>3600</td>
</tr>
<tr>
<td>16-QAM</td>
<td>9600</td>
<td>4800</td>
</tr>
<tr>
<td>32-QAM</td>
<td>12000</td>
<td>6000</td>
</tr>
<tr>
<td>64-QAM</td>
<td>14400</td>
<td>7200</td>
</tr>
<tr>
<td>128-QAM</td>
<td>16800</td>
<td>8400</td>
</tr>
<tr>
<td>256-QAM</td>
<td>19200</td>
<td>9600</td>
</tr>
</tbody>
</table>
**Modem standards**
- Bell modems
- ITU-T modems

**Bell Modems**
- (the first commercial modems were produced by the Bell telephone company in the early 1970s).
- **103/113 Series** The Bell 103/113 series modems operate in FDX mode over two-wire switched telephone line. Transmission is asynchronous. Using FSK encoding, session originator frequencies are 1,070 Hz=0 and 1,270 Hz=1. Answerer frequencies are 2,025 Hz=0 and 2,225 Hz=1. Data rate is 300 bps. The 113 series is a variation of the 103 series with additional testing features.

- **202 Series** The bell 202 series modems operate in HDX mode over two-wire switched telephone lines. Transmission is asynchronous, using FSK encoding, because the 202 series is HDX, only one pair of transmission frequency is used: 1,200 Hz=0, and 2,400 Hz=1. 202 series includes a secondary transmission frequency operating in either direction at 387 Hz, using ASK encoding, with a data rate of only 5 bps, used for flow control or error control.

- **212 Series** The Bell 212 series modems have two speeds. The option of a second speed allow for compatibility with a wider number of system. Both speeds operate in full-duplex mode over switched telephone lines. The lower speed 300 bps uses FSK encoding for asynchronous transmission. Just like the 103/113 series. The higher speed, 1,200 bps, can operate in either asynchronous a synchronous mode, and uses 4-PSK encoding. While the 1,200 bps in the same data rate as that achieved by the 202 series, the 212 series achieves that rate in full-duplex rather than half-duplex mode. By moving from FSK to PSK encoding, the designers have dramatically increased the efficiency of transmission. In 202 series, two frequencies are used to send different bits in one direction. In series 212, two frequencies represent two different direction of transmission.

- **201 Series** The 201 series modems operate in either half- duplex mode over two-wire switched lines or full- duplex mode over four- wire leased lines. Transmission is synchronous, using 4-PSK encoding, which mean that only one frequency is needed for transmission over each pair of wire. Splitting the two directions of transmission into two
physically separate lines allows each direction to use the entire bandwidth of the line. This means, that with essentially the same technology, the date rate is double to 2,400 bps (or 1,200 baud) in both half – duplex modes (2,400 bps is still half the theoretical max data rate for 4-PSK encoding over two- wire phone lines.

**- 208 series** The 208 series modems operate in full-duplex mode over four-wire teased lines. Transmission is synchronous, using 8-PSK encoding. Like the 201 series, the 208 series modems achieve full-duplex status by doubling the number of wires used and dedicating the equivalent of an entire line of each direction of transmission. The difference here is that the encoding/decoding technology is now able to distinguish btw. Eight different phase shifts. This modem has a baud rate of 1,600. At three bits per baud (8-PSK creates tribit), that rate translates to a bit rate of 4,800 bps.

**- 209 series** The 209 series modems operate in full-duplex mode over four-wire leased lines. Transmission is synchronous, using 16-QAM encoding. These modems achieve full-duplex status by doubling the number of wires so that each direction of transmission has a channel to itself. This series however, allows for use of the entire bandwidth of each channel. Each shift represents a quadbit, with 16-QAM, the data rate 9,600 bps.

**ITU-T Modem Standards**

Two groups of ITU-T modems: those that are essentially equivalent to Bell series modems and those that are not.

**ITU-T/Bell Compatibility**

<table>
<thead>
<tr>
<th>ITU-T</th>
<th>Bell</th>
<th>Baud Rate</th>
<th>Bit Rate</th>
<th>Modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>v.21</td>
<td>103</td>
<td>300</td>
<td>300</td>
<td>FSK</td>
</tr>
<tr>
<td>v.22</td>
<td>212</td>
<td>600</td>
<td>1200</td>
<td>4-PSK</td>
</tr>
<tr>
<td>v.23</td>
<td>202</td>
<td>1200</td>
<td>1200</td>
<td>FSK</td>
</tr>
<tr>
<td>v.26</td>
<td>201</td>
<td>1200</td>
<td>2400</td>
<td>4-PSK</td>
</tr>
<tr>
<td>v.27</td>
<td>208</td>
<td>1600</td>
<td>4800</td>
<td>8-PSK</td>
</tr>
<tr>
<td>v.29</td>
<td>209</td>
<td>2400</td>
<td>9600</td>
<td>16-QAM</td>
</tr>
</tbody>
</table>
ITU-T modems that do not have equivalent in Bell series are given below:

- **V.22 bis** The term bis means the second generation. The V.22 bis is a two-speed modem, i.e. it can operate at either 1,200 or 2,400 bps. The used speed depends on the speed of the DCE at the other end. In 1,200 bps mode, the V.22 bis uses 4-DPSK (dibit) encoding at a transmission rate of 600 baud.

  - 00 → 0° phase change
  - 01 → 90° phase change
  - 10 → 180° phase change
  - 11 → 270° phase change

In 2,400 bps mode, the V.22 bis uses 16-QAM (quad bit)

- **V.32** The V.32 is an enhanced version of the V.29 that uses a combined modulation and encoding technique called trellis-coded modulation. Trellis is essentially QAM plus redundant bit. The data stream is divided into four bit sections. Instead of a quadbit, however, a quintbit (five-bit pattern) is transmitted. The value of the extra bit is calculated from the value of the data bits. V.32 calls for 32-QAM with a baud rate of 2,400. Because only four bits of each quintbit represent data, the resulting speed is \( 4 \times 2,400 = 9,600 \) bps.

V.32 modems and used with two-wire switched Line in what is called pseudo-duplex. Mode pseudo-duplex is based on a technique called echo cancellation.

- **V.32 bit** The V.32 bis modem was the first of the ITU-T standards to supported 14,400 bps transmission. The V.32 bis uses 64-QAM transmission (six bits per baud) at a rate of 2,400 baud (2,400*6 = 14,400 bps).

  An addition enhancement provided by the V.32 bis is the inclusion of an automatic fall-back and fall-forward feature that enables the modem to adjust its speed upward and downward depending on the quality of the line or signal.

- **V.32 turbo** The V.32 turbo is an enhanced version of V.32 bis. It uses 256-QAM to provide a bit rate of 19,200 bps.

- **V.33** The V.33 is also based on V.32 this modem, however, uses trellis-coded modulation based on 128-QAM at 2,400 baud. Each signal change represents a pattern of seven bits: six data bits and one redundant bit. Six bits of the data per change (baud) give it a speed of 6*2,400=14,400 bps.
- **V.34** This modem provides a bit rate of 28,800 bps, earning the nickname V.fast. It achieves this rate by representing 12 bits with each signal change. In addition, the V.34 is designed to provide data compression. With data compression, the V.34 can achieve data rates as fast as two to three times its normal speed.

**Intelligent modems**

Intelligent modems contain software to support a number in addition to modulation and demodulation of functions, such as automatic answering and dialling.

Intelligent modems were first introduced by Hayes Microcomputers products, Inc. more recently; other manufactures have come out with what are referred to as Hayes-compatible modems.

Instructions in the Hayes and compatible modems are called AT commands format is:

\[ \text{AT command [parameter] command [parameter]} \ldots \]

Each command starts with the letters AT followed by one or more commands, each of which can take one or more parameters. E.g., to have the modem dial (408) 486- 8902, the command is: TD 4088648902

### Some AT commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Put modem in answer mode</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>Use V.22 bis at 1200 bps</td>
<td>The number to dial 0 or 1</td>
</tr>
<tr>
<td>D</td>
<td>Dial the number</td>
<td>n</td>
</tr>
<tr>
<td>E</td>
<td>Enable/disable echo printing</td>
<td>0 or 1</td>
</tr>
<tr>
<td>H</td>
<td>Put modem on/off hook</td>
<td>0 or 1</td>
</tr>
<tr>
<td>L</td>
<td>Adjust speaker volume</td>
<td>-</td>
</tr>
<tr>
<td>P</td>
<td>Use pulse dialing</td>
<td>-</td>
</tr>
<tr>
<td>T</td>
<td>Use tone dialing</td>
<td>-</td>
</tr>
<tr>
<td>AT</td>
<td>Attenuation</td>
<td>-</td>
</tr>
<tr>
<td>DT</td>
<td>Dial using DTMF tones</td>
<td>-</td>
</tr>
<tr>
<td>DP</td>
<td>Dial using pulse dialing</td>
<td>-</td>
</tr>
<tr>
<td>FO</td>
<td>HDX</td>
<td>-</td>
</tr>
<tr>
<td>F1</td>
<td>FDX</td>
<td>-</td>
</tr>
<tr>
<td>H</td>
<td>Hang up</td>
<td>-</td>
</tr>
<tr>
<td>O</td>
<td>Switch from command to on-line mode</td>
<td>-</td>
</tr>
<tr>
<td>Z</td>
<td>Reset modem</td>
<td>-</td>
</tr>
<tr>
<td>+++</td>
<td>Switch from on-line to command mode</td>
<td>-</td>
</tr>
</tbody>
</table>


**56K Modems**

Traditional modems have a limitation on the data rate (max. 33.6 kbps), as determined by the Shannon formula. However, new modems, with a bit rate of 56,000 bps, called 56K modems, are now in the market. These modems can be used only if one party is using digital signalling (such as through an Internet provider). They are asymmetrical in that the downloading (flow of data from Internet provider to the PC) is a maximum of 56 kbps, while the uploading (flow of data from PC to the Internet provider) can be a maximum of 33.6 kbps.

In traditional modems transmission of data from a computer at site A to another at site B, analog data are converted to digital using PCM at the switching station. Here, the analog signal is quantised to create the digital signal. The quantisation noise resulting from the process limits the data rate to 33.6 kbps in each direction.

In 56K modems if one side is an Internet provider and the signal does not have to pass through a PCM converter, quantisation is eliminated in one direction and the data rate can be increased to 56 kbps.

**Uploading**

Transmission of data from the subscriber to the Internet provider (uploading) follows these steps:

1- Digital data are modulated by the modem at site A.
2- Analog data are sent from the modem to the switching station at site A on the local loop.
3- At the switching station, the data are converted to digital using PCM.
4- Digital data travel through the digital network of the telephone company and are received by the Internet provider computer.

The limiting factor in these steps is step 3. Although there is no improvement in data rate, the user actually does not need high data rate, since in this direction, only small blocks of data (such as an e-mail or a small file) are sent.

**Downloading**

Transmission of data from the Internet provider to the modem at site A (downloading) follows these steps:
1- Digital data are sent by the Internet provider’s computer through the digital telephone network.

2- At the switching station, digital data are converted to analog using inverse PCM.

3- Analog data are sent from the switching station at site A to the modem on the local loop.

4- Analog data are demodulated by the modem at site A.

Here, there is no quantisation of data using PCM, therefore, data can be sent at 56 kbps. This is what a user is looking for, since large files are typically downloaded from the Internet.

The maximum data rate in the uploading direction is still 33.6 kbps, but the data rate in the download direction is 56 kbps.
CHAPTER 6
MULTIPLEXING

Multiprocessing DTEs such as timesharing computers are designed to support literally dozens of remote DTEs (PCs, terminals, and so on). Each remote DTE connection is made through a port, an interface to the central computer. Most operating systems allow the various ports to be configured by the system manager or the operating system to specific terminal types (for example, dial-in, asynchronous, synchronous).

A typical set of DTE connections in a multiprocessing environment might consist of the configuration shown in the following Figure, where three interactive terminals (DTE 1, DTE 2, and DTE 3) are connected by a series of individual asynchronous channels to ports E, L and S, respectively, on a shared central processor (the host DTE).

Remote DTEs connected to dedicated ports

If the individual channels connecting the interactive terminals to the ports on the host are only partly in use, it may be possible to have some or all of the terminals multiplex their communications onto a single channel, as shown in the following Figure.

Remote DTEs sharing a channel by multiplexing
The equipment controlling the multiplexing is known as a multiplexer (or MUX) and performs two operations:
* It takes information from the various DTEs and puts it on the multiplexed channel. This is known as multiplexing (MUX).
* It takes information from the multiplexed channel and supplies it to the intended destination DTE. This is known as demultiplexing (DEMUX).

The operation of the multiplexer should be transparent to the DTEs being multiplexed. In this situation, transparency refers to the effect of the multiplexer on communications. For example, there should be no detectable difference in communication speeds whether or not the multiplexer is present. Similarly, the data sent should not be altered in any way by the multiplexer.

The multiplexed channel is typically full-duplex, thereby allowing some DTEs to transmit information while others receive. However, the connections between a multiplexer and its multiplexed DTEs can be full-duplex, half-duplex, simplex, or all three.

**Multiplexer Internals**
- A multiplexer is another example of the input-processing-output cycle. The multiplexing task consists of
  1. Checking a connection for data.
  2. Formatting the data according to the protocol used by the channel.
  3. Transmitting the data on the channel.
- The demultiplexing algorithm is somewhat similar:
  1. Reading the data from the channel.
  2. Determining the destination of the data.
  3. Forwarding the data to the specific connection.

**Multiplexing Techniques**
The multiplexer is responsible for ensuring that the information supplied by one DTE arrives at the correct destination DTE. It employs one of three techniques: frequency division multiplexing, time division multiplexing, and statistical multiplexing.

**Frequency Division Multiplexing**
Frequency division multiplexing, or FDM, involves dividing the multiplexed channel into a number of unique frequencies, each one assigned to a pair of communicating entities. FDM
can be achieved only if the available band width on the multiplexed channel exceeds the bandwidth needs of all the communicating entities.

Whenever a multiplexer receives data for transmission, the data is transmitted by it on the frequency allocated to the transmitting entity. The receiving multiplexer forwards the information received on a specific frequency to the destination associated with that frequency.

The following example illustrates how a frequency division multiplexer connects DTEs 1, 2, and 3 with ports E, L, and S, respectively, on a central host. The frequency allocation is given in the following Table.

<table>
<thead>
<tr>
<th>DTE-Port Pair</th>
<th>Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and E</td>
<td>10,000-14,000</td>
</tr>
<tr>
<td>2 and L</td>
<td>5,000-9,000</td>
</tr>
<tr>
<td>3 and S</td>
<td>0-4,000</td>
</tr>
</tbody>
</table>

The 1000-Hz separation between the channels is known as the **guard band** and is used to ensure that one set of signals does not interfere with another. Diagrammatically, the connections and their frequencies are shown in the following Figure.

![Frequency division multiplexing](image)

The advantage of FDM is that each DTE is assigned a unique frequency that can be treated as an unshared channel. However, FDM is not widely used in data communications because of the costs of hardware that can transmit and receive signals on a variety of frequencies. An everyday example of FDM is cable television, in which many signals are "stacked up" and transmitted simultaneously over the cable. The user selects a viewing channel by tuning to that channel's frequency.
**Time Division Multiplexing**

Time division multiplexing, or TDM, requires the multiplexer to timeshare the channel between the various DTEs involved in the communication. That is, at a specific moment the remote multiplexer will send a byte from, say, DTE 1; at the next instance a byte from DTE 2 will be sent, and so on until all DTEs have been polled; the cycle is then repeated.

In the following Figure, DTEs 1, 2, and 3 are in communication with ports E, L, and S, respectively. The multiplexed channel is full-duplex and shared among all the communicating DTEs (the identifier in each box in the multiplexed channel between the two multiplexers should be read as "data from" the specific port or DTE; that is, it contains data rather than the identifier).

![Diagram of Time Division Multiplexing](image)

**Time division multiplexing**

In the Figure above, the topmost channel contains data from DTEs 1, 2, and 3. The rightmost multiplexer is about to receive a byte from DTE 3, and this is followed by a byte from DTE 2. On the lower channel, data is sent from ports E, L, and S; the leftmost multiplexer is about to receive a byte from port S.

To prevent information arriving at the wrong DTE, both multiplexers must be synchronised. That is, the bytes must be sent in an agreed-upon order and each byte that is received must be for the specific DTE (or port) for which it was intended. Synchronisation can be achieved in a number of ways. A common approach is to use a special bit pattern to indicate the start of a new cycle, so that if \( n \) DTEs are being multiplexed, the \((n + 1)\)th byte to be transmitted is a special, synchronising bit pattern.

A time division multiplexer polls each DTE to determine if there is information to be sent. However, a problem arises if one of the DTEs has nothing to send: what does the multiplexer transmit? Something *must* be sent because not sending a byte means that the time allotted to one DTE may be used by another, potentially resulting in the information arriving at the
wrong destination. For example, if DTE 2 has nothing to send to port L, data from DTE 1 might be sent instead.

This problem can be overcome in several ways, including:
- Having a reserved bit pattern (for example, the *NUL* character) that is sent whenever a DTE (or port) has nothing available for transmission.
- Transmitting *nine* bits between the multiplexers, eight for data and the ninth to signal whether the byte contains data or is empty.

**Statistical Multiplexing**

Statistical multiplexing attempts to overcome the problem of idling DTEs by sending information from a DTE only when it is available. For example, if three DTEs are sharing a channel and only one is active, then as much of the channel as possible should be given over to the active DTE. (In the time division multiplexing example in the previous section, two of the three time slots would be empty because of the inactive DTEs.)

Since the multiplexers must be able to determine the intended destination of each byte received, a unique identifier either of the source (i.e., the sending port or DTE) or of the destination (i.e., the receiving port or DTE) of the byte must be included with each byte. This has two implications:
- The bandwidth is reduced because the identifier is sent with each byte. If the identifier is a byte (a reasonable choice, since the multiplexed channel is probably byte-oriented), the bandwidth is halved.
- The multiplexer must know the destination of the byte. Each byte must be transmitted with an identifier that can indicate either the byte's source or its intended destination. If the identifier is the source's address, the remote multiplexer must map that address into a destination connection.
Similarly, if the identifier is a destination address, the local multiplexer must map the local device's identifier into the destination address. Either of these approaches can be through the use of mapping tables.

In addition to the reasons already noted, statistical multiplexing differs from time division multiplexing in that if none of the DTEs are active, the multiplexed channel will be idle.

The above Figure shows how a pair of statistical multiplexers can function. As bytes are sent, they are prefixed with an identifier that allows the receiving multiplexer to determine their destination. In this example, each data byte is sent as two bytes: the address of the destination (displayed in bold) and the data.

In the Figure, the topmost channel (flowing from left to right) contains two bytes destined for port E (K followed by O) and one byte for port S (a P). The lower channel has two bytes for DTE 2 (G followed by O); additionally, there is a period during which nothing is being transmitted on the channel.

If all devices connected to a multiplexer transmit simultaneously for a sustained period, there may not be sufficient bandwidth to handle all of the traffic. For these situations, the multiplexers may simply discard the additional input or buffers may be used to hold the data until transmission can take place.

**Port Selectors**

In most multiprocessing systems, there are more users than there are possible connections to the central computer; in some cases, there may be more remote DTEs than there are ports to the computer. From the system manager's point of view, this is quite reasonable since few users want to be on the computer 24 hours a day (there are exceptions), and the computer manufacturer may place a limit on the maximum number of ports that the computer can support.

In situations where there are more remote DTEs than ports on the central host, a device known as a port selector (or front end, switch, or terminal concentrator) is employed to manage the connections from the remote DTEs to the host's ports. The port selector supports two sets of connections: those to the remote DTEs and those to the host's ports. The port selector is typically connected to all possible remote DTEs and to all ports on the central host.

When a user on a remote DTE wants to initiate a communication with the central host, some form of signal is sent by the user to the port selector— for example, a series of characters (typically one or more carriage returns), a break indication, or the detection of a carrier signal generated when the DTE is powered on. If a free port on the central host is available, the port
selector makes a logical connection between it and the remote DTE. Thereafter, all communications between the remote DTE and the host's port are handled by the port selector mapping the data from the DTE to the port or vice versa. In the following Figure, a central host has two ports (A and B) and there are four possible remote terminals (DTE 1 through DTE 4); the port selector as mapped DTE 1 to port A and DTE 3 to port B.

**Port Selector**

When the communication finishes, the logical connection is broken and another DTE can use the port. Over a period of time, a DTE might be involved in any number of communications with the central host; however, each communication may use a different port. If all ports are in use, the port selector ignores the incoming requests (possibly issuing a diagnostic message to the user at the remote DTE).

Port selectors can offer multiplexing capabilities as well. That is, one or more remote DTEs might share a single channel from a remote site; the port selector will demultiplex the channel and make logical connections to free ports using the same techniques as for the directly connected remote DTEs.
CHAPTER 7
ERROR DETECTION & CORRECTION

Error Control Coding System
The major design criterion for all communication systems is to achieve error free transmission. However, errors occur. The methods of detecting and sometimes correcting errors are necessary.
Error-control coding involves systematic addition of extra digits to the message.

Two Basic Techniques:
1- Automatic Repeat Request (ARQ):
Request the transmission of the data block received in error. When a data block is received without error, a positive acknowledgement (ACK) or logic 1 is sent back to the transmitter via the reverse channel. If the receiver detects an error, it returns a negative acknowledgement (NAK) signal to the transmitter. In this case the transmitter retransmits the same message and then waits for ACK or NAK response before undertaking further transmission.

Forward Error Correction (FEC)
Coding for detection without correction is simpler than error correction coding. FEC is used in simplex communication or applications where it is impractical or impossible to request
retransmission of the corrupted message block. An example might be the telemetry signals transmitted to an Earth station from a satellite on a deep space mission.

![Diagram of retransmission process]

**Parity Check Codes**

A single bit called the parity bit is added to a group of bits representing a letter, a number or a symbol. ASCII characters on a keyboard for example, are typically encoded into seven bits with eighth bit acting as parity.

Parity is computed by the transmitting device based on the number of 1s set in the character.
- Odd parity bit: total number of 1s in the character plus the parity bit equal to an odd value.
- Even parity bit: total number of 1s in the character plus the parity bit equal to an even value.

**Ex**

<table>
<thead>
<tr>
<th>Data character</th>
<th>Even parity</th>
<th>Sent data unit + parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101000</td>
<td>1</td>
<td>1101000</td>
</tr>
<tr>
<td>1110111</td>
<td>0</td>
<td>1110111</td>
</tr>
<tr>
<td>0011010</td>
<td>1</td>
<td>0011010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data character</th>
<th>Odd parity</th>
<th>Sent data unit + parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010111</td>
<td>1</td>
<td>0010111</td>
</tr>
<tr>
<td>1010110</td>
<td>1</td>
<td>1010110</td>
</tr>
<tr>
<td>1010001</td>
<td>0</td>
<td>1010001</td>
</tr>
</tbody>
</table>

**Redundancy**

Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.

![Detection methods diagram]
**Vertical Redundancy Check (VRC)**

The most and the least expensive mechanism for error detection is the VRC, often called a parity check. In this technique a redundant bit, called a parity bit, is added to every data unit so that the total number of 1s becomes even.

Parity may also be odd, i.e. the total number of 1s is odd.

**Ex**

Suppose the sender wants to send the word ‘world’. In ASCII, the five characters are coded as:

```
  1110111  1101111  1110010  1101100  1100100
  w     o     r     l     d
```

adding the VRC

```
  11101110  11011110  11100100  11011000  11001001
```

If the receiver counts the 1s in each character and comes up with even number, the data will be accepted.

Now suppose that the word ‘world’ is received by the receiver but corrupted during transmission:

```
  1111110  11011110  11101000  11011000  11001001
```

The receiver counts the 1s in each character and comes up with even and odd numbers. The receiver knows that the data are corrupted, discards them, and asks for retransmission.

**Longitudinal Redundancy Check (LRC)**

In LRC, a block of bits is divided into rows and a redundant row of bits is added to the whole block. (The LRC is the even parity of each column separately).
The receiver will check the LRC if the same result is obtained, the data will be accepted, but if corrupted the receiver will ask for retransmission.

**Cyclic Redundancy Check (CRC)**

LRC and VRC may both provide sufficient error detection, especially if the channel is known to be a reliable and reasonably error-free. However, sometimes a near-to-perfect error detection may be needed. An example of this might be an electronic transfer of funds, where an undetected error may cause serious consequences to the bank or its customers if the amount of fund received does not match the sent one. The technique used in many applications requiring better error detection is cyclic redundancy check, or CRC which is based on binary division. In CRC instead of adding bits together to achieve a desired parity, a sequence of redundant bits, called the CRC or the CRC remainder, is appended to the end of a data unit so that the resulting data unit becomes divisible by a second, predetermined binary number. At its destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is accepted. A remainder indicates that the data unit has been damaged and therefore must be rejected.

Basically, the CRC algorithm treats the message as a single bit stream in which each bit is taken as a coefficient of a polynomial. In general, a message k bits long has k terms and is a polynomial of order k-1. A message with terms $m^{k-1} + m^{k-2} + \cdots + m^2 + m^1 + m^0$ can be written as the following polynomial:

$$M(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \cdots + m_2x^2 + m_1x^1 + m_0$$
For example, the message 101101001 can be written as the polynomial:

$$1x^8 + 0x^7 + 1x^6 + 1x^5 + 0x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0$$

This will simplify to

$$x^8 + x^6 + x^5 + x^3 + x^0$$

The generator polynomial, $G(x)$, is a polynomial of degree $g$, which must be less than the degree of the message polynomial, $M(x)$. $G(x)$ is always odd (i.e., the lowest-order term has a value of 1) and must have a value greater than one. The transmission algorithm is as follows:

1- The message $M(x)$ is multiplied by $x^g$ (i.e., the message is shifted left by $g$ bit position; these bit positions are cleared).
2- The result of the multiplication is divided by $G(x)$, given a quotient, $Q(x)$, and a remainder, $R(x)$:

$$\frac{x^g \times M(x)}{G(x)} = Q(x) \oplus \frac{R(x)}{G(x)}$$

$R(x)$ is always less than $G(x)$, since the maximum number of bits in the remainder is $g$.

3- $R(x)$ is added to the shifted message (i.e., the lower $g$ bits), producing the frame to be transmitted $T(x)$.

$$T(x) = x^g \times M(x) \oplus R(x)$$

The receiving entity receives the frame $T(x)$ divide it by $G(x)$. The remainder should be 0.

**Ex**

Consider the transmission of the message 101101001 using the generator polynomial 101001 ($x^5 + x^3 + 1$). Following the algorithm described above, detect the error if any.

1- $G(x)$ has $g = 5$, meaning that $M(x)$ must be shifted to the left by 5:

$$2^5 \times 101101001 = 10110100100000$$

2- The shifted message is then divided by the generator polynomial:

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The quotient $Q(x)$ is discarded.
3- $R(x), 11010$, is added to the shifted message, producing $T(x)$, consisting of $M(x)$ and $R(x)$. $T(x)$ is therefore 1011010011010.

The receiving device accepts the transmitted frame and performs the division using the same generator polynomial:

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
\oplus & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 \\
\oplus & 1 & 0 & 1 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 \\
\oplus & 1 & 0 & 1 & 0 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

(Quotient) (Remainder)

The result of the division is 0, indicating that no errors were detected in the frame. However, if one or more bits become inverted, the division should result in a non-zero remainder. For example, if the frame received was 1010110011010 (rather than 1011010011010), the division would proceed as follows:

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\oplus & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 \\
\oplus & 1 & 0 & 1 & 0 & 0 & 1 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 0 \\
\oplus & 1 & 0 & 1 & 0 & 0 & 1 \\
\hline
1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]

Checksum

The error detection method used by the higher-layer protocols is called checksum. In the sender, the checksum generator subdivides the data unit into equal segments of n bits (usually 16). These segments are added together using one’s complement arithmetic in such a way that the total is also n bits long. The total sum is then complemented and added to the end of the original data unit as redundancy bits, called the checksum field. The extended data unit is transmitted across the network. So if the sum of the data segment is $T$, the checksum is $-T$.

Ex

Suppose that the following block of 16 bits is to be sent using a checksum of 8 bits.

\[
\begin{array}{cccc}
\rightarrow & 10101001 & 00111001 \\
\end{array}
\]
The pattern sent is

\[
\begin{array}{c}
10101001 \\
00111001 \\
\hline
\text{Sum} \quad 11100010 \\
\text{Checksum} \quad 00011101
\end{array}
\]

If there is no error, the receiver adds the three segments together, it will get all 1s, which after complementing are all 0s and shows that there is no error.

\[
10101001 \\
00111001 \\
00011101 \\
\hline
\text{Sum} \quad 11111111 \\
\text{Complement} \quad 00000000
\]

means the pattern is O.K.

**Ex**

Suppose there is a burst error (burst error: two or more bits are corrupted)

\[
\begin{array}{c}
10101111 \\
1111001 \\
00011101 \\
\hline
\text{Result} \quad 111000101 \\
\text{Carry} \quad 1 \\
\hline
11000110 \\
\text{Complement} \quad 00111001
\end{array}
\]

means the pattern is corrupted.

The receiver entity can reject the frame, since the remainder after the division is non-zero.

**Error Correction** Error correction can be handled in two ways: In one, the receiver can have the sender retransmit the entire data. In the other, a receiver can use an error-correction code, which automatically corrects certain errors.
Hamming Code

Hamming codes employ the use of redundant bits that are inserted into the message stream for error correction. The position of these bits are established and known by the transmitter. If the receiver detects an error in the message block, the Hamming bits are used to identify the position of the error. This position, known as the syndrome, is the underlying principle of Hamming code.

If \( m \) is the number of data bits, then the number of redundancy bits \( r \) can be calculated using the following relationship:

\[
2^r \geq m + r + 1
\]

\( m + r \) is the total number of bits in the transmitted bit stream.

For example, if the number bits in the original message is 7 (as in a 7-bit ASCII code), the smallest value that can satisfy the above equation is 4:

\[
2^4 \geq 7 + 4 + 1
\]

The redundancy bits are placed in positions of the bit sequence that are the power of 2.

In Hamming code, each \( r \) bit is the VRC bit for one combination of the data bits:

- Position of \( r_1 \): bits 1, 3, 5, 7, 9, 11, 13
- Position of \( r_2 \): bits 2, 3, 6, 7, 10, 11, 14
- Position of \( r_4 \): bits 4, 5, 6, 7, 12, 13, 14
- Position of \( r_8 \): bits 8, 9, 10, 11, 12, 13

e.g. 7-bits ASCII code:

\[
\begin{array}{cccccccccccc}
11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\
\hline
d & d & d & r & d & d & d & r & d & r & r \\
\end{array}
\]

Ex

Given the ASCII character 1001101, let us construct the Hamming codes:

\[
\begin{array}{cccccccc}
d_6 & d_5 & d_4 & d_3 & d_2 & d_1 & d_0 \\
\hline
1 & 0 & 0 & r_8 & 1 & 1 & 0 & r_4 & 1 & r_2 & r_1 \\
11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

\[
r_1 = d_6 \oplus d_4 \oplus d_3 \oplus d_1 \oplus d_0 \\
= 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1
\]
\[ r_2 = d_6 \oplus d_5 \oplus d_3 \oplus d_2 \oplus d_0 \]
\[ = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0 \]
\[ r_4 = d_3 \oplus d_2 \oplus d_1 \]
\[ = 1 \oplus 1 \oplus 0 = 0 \]
\[ r_8 = d_6 \oplus d_5 \oplus d_4 \]
\[ = 1 \oplus 0 \oplus 0 \]
\[ = 1 \]
\[
\begin{array}{cccccccccc}
d_6 & d_5 & d_4 & r_8 & d_3 & d_2 & d_1 & r_4 & d_0 & r_2 & r_1 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

**Code: 10011100101**

- Now assume that the receiver received the following message
  10010100101 with bit 7 corrupted

The receiver takes the transmission and recalculates four new redundant (VRCs) using the same sets of bits used by the sender plus the relevant parity (r) bit for each set. Once the bit is identified, the receiver reverses its value and corrects the error.

\[
\begin{aligned}
&1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
& r_1 = 1 \\
&1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
& r_2 = 1 \\
&1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
& r_4 = 1 \\
&1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
& r_8 = 0 \\
&0 1 1 1 \\
&7 \rightarrow \text{bit 7 is corrupted} \\
\end{aligned}
\]

The receiver changes the bit in position 7 from 0 to 1.