Module 3

Data Link control

Lesson 1

Interfacing to the media and synchronization

Special Instructional Objectives

On completion of this lesson, the students will be able to:

- Explain various modes of communication
- Distinguish between half-duplex and full-duplex modes of communication
- Distinguish between Asynchronous and Synchronous modes of communication
- Specify the RS-232 standard used for DTE-DCE interface
- Explain the function of a null-modem
- Explain the functionality and standards used for MODEMS

3.1.1 Introduction

In the previous module we have discussed various encoding and modulation techniques, which are used for converting data in to signal. To send signal through the transmission media, it is necessary to develop suitable mechanism for interfacing data terminal equipments (DTEs), which are the sources of data, to the data circuit terminating equipments (DCEs), which converts data to signal and interfaces with the transmission media. The way it takes place is shown in Fig. 3.1.2. The link between the two devices is known as *interface*. But, before we discuss about the interface we shall introduce various modes of communication in Sec. 3.1.2. Various aspects of framing and synchronization for bit-oriented framing have been presented in Sec. 3.1.3. Character-oriented framing has discussed in Sec. 3.1.4. Finally, We shall discuss about the interface in detail along with some standard interfaces in Sec. 3.1.5.

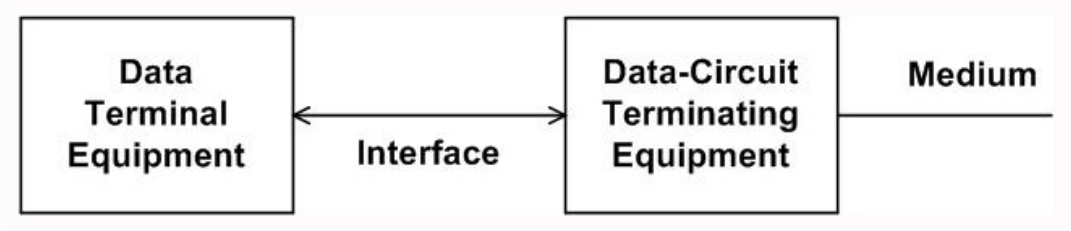


Figure 3.1.2 Interfacing to the medium

3.1.2 Possible Modes of communication

Transmission of digital data through a transmission medium can be performed either in serial or in parallel mode. In the serial mode, one bit is sent per clock tick, whereas in parallel mode multiple bits are sent per clock tick. There are two subclasses of transmission for both the serial and parallel modes, as shown in Fig 3.1.3.

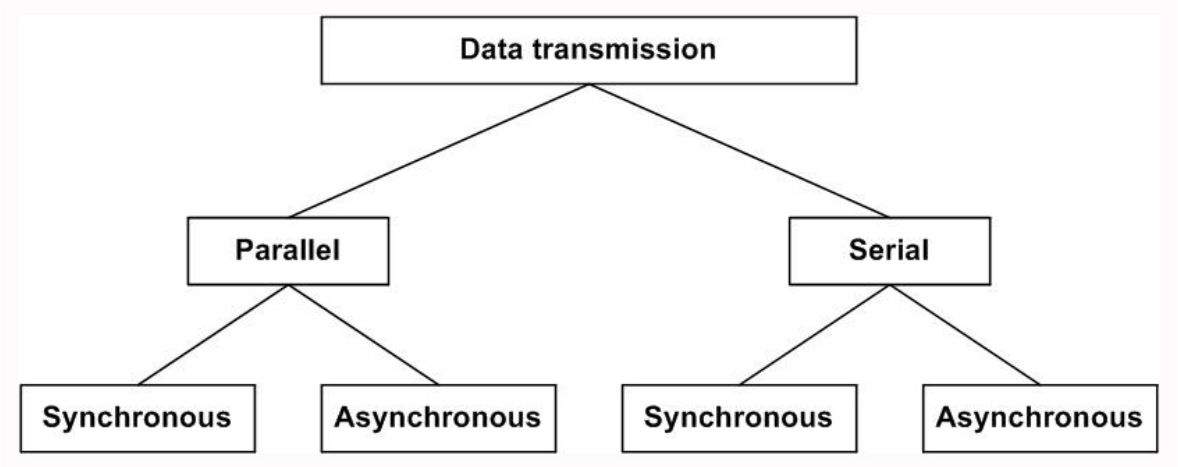


Figure 3.1.3 Different modes of transmission

Parallel Transmission

Parallel transmission involves grouping several bits, say n, together and sending all the n bits at a time. Figure 3.1.4 shows how parallel transmission occurs for n = 8. This can be accomplishes with the help of eight wires bundled together in the form of a cable with a connector at each end. Additional wires, such as request (req) and acknowledgement (ack) are required for asynchronous transmission.

Primary advantage of parallel transmission is higher speed, which is achieved at the expense of higher cost of cabling. As this is expensive for longer distances, parallel transmission is feasible only for short distances.

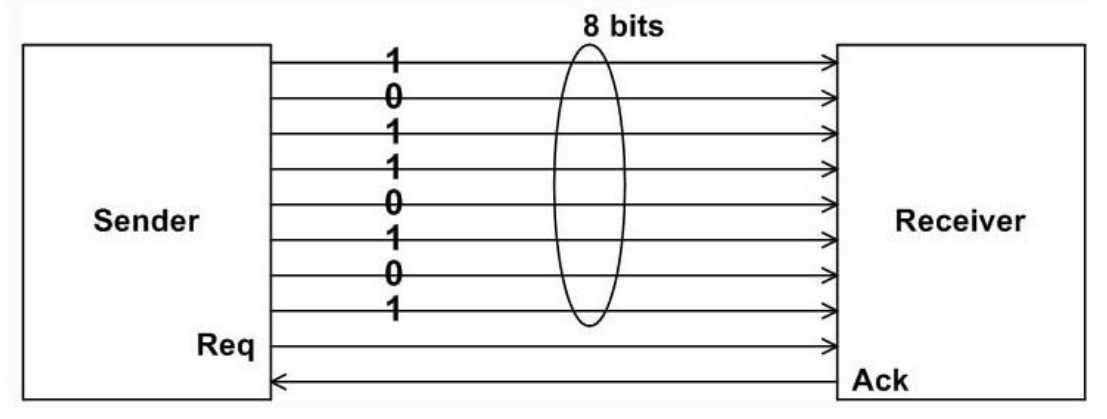


Figure 3.1.4 Parallel mode of communication with n = 8

Serial Transmission

Serial transmission involves sending one data bit at a time. Figure 3.1.5 shows how serial transmission occurs. It uses a pair of wire for communication of data in bit-serial form.

Since communication within devices is parallel, it needs parallel-to-serial and serial-to-parallel conversion at both ends.
Serial mode of communication widely used because of the following advantages:

- Reduced cost of cabling: Lesser number of wires is required as compared to parallel connection
- Reduced cross talk: Lesser number of wires result in reduced cross talk
- Availability of suitable communication media
- Inherent device characteristics: Many devices are inherently serial in nature
- Portable devices like PDAs, etc use serial communication to reduce the size of the connector

However, it is slower than parallel mode of communication.

There are two basic approaches for serial communication to achieve synchronization of data transfer between the source-destination pair. These are referred to as – **asynchronous** and **synchronous**. In the first case, data are transmitted in small sizes, say character by character, to avoid timing problem and make data transfer self-synchronizing, as discussed later. However, it is not very efficient because of large overhead. To overcome this problem, synchronous mode is used. In synchronous mode, a block with large number of bits can be sent at a time. However, this requires tight synchronization between the transmitter and receiver clocks.

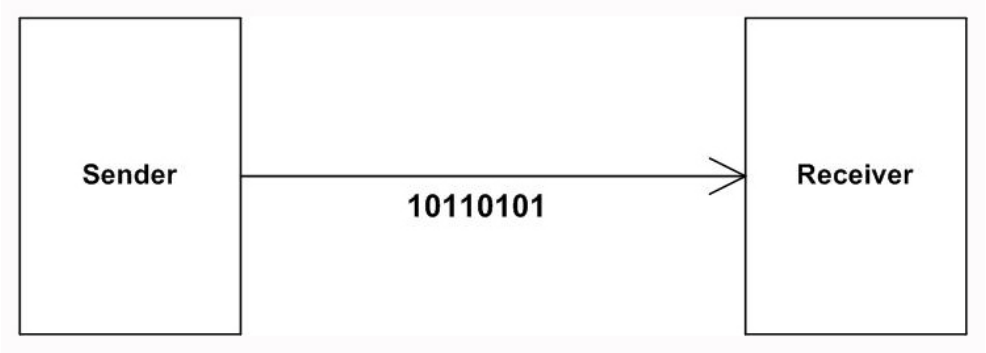


Figure 3.1.5 Serial mode of communication

Direction of data flow:

There are three possible modes in serial communication: simplex, full duplex and half duplex. In **simplex** mode, the communication is unidirectional, such as from a computer to a printer, as shown in Fig. 3.1.6(a). In **full-duplex** mode both the sides can communicate simultaneously, as shown in Fig. 3.1.6 (b). On the other hand, in **half-duplex** mode of communication, each station can both send and receive data, as shown in Fig. 3.1.6 (c). But, when one is sending, the other one can only receive and vice versa.

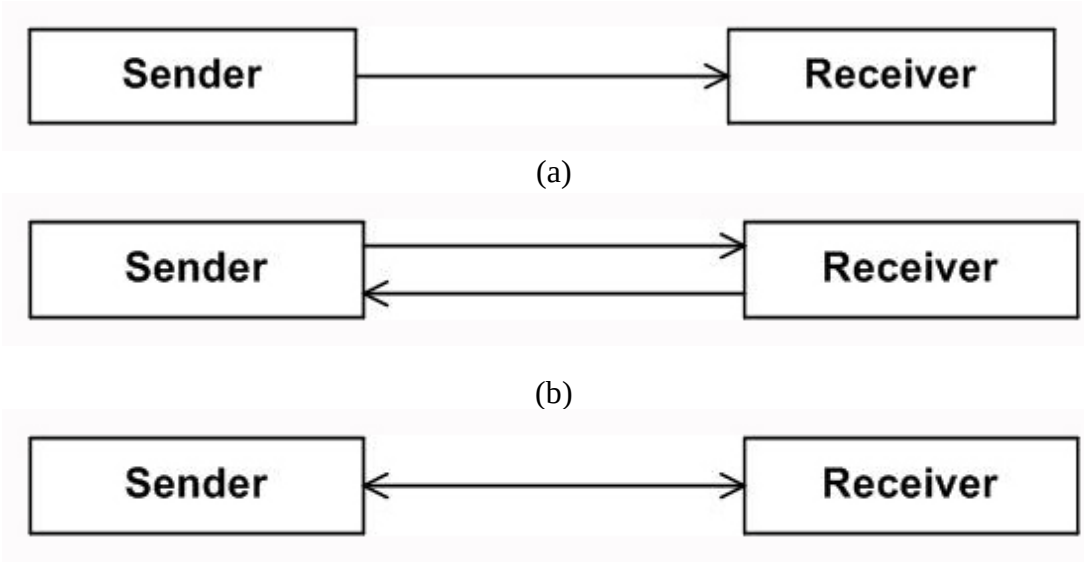


Figure 3.1.6 Direction of data flow

3.1.3 Framing and Synchronization

3.1.3.1Why Framing and Synchronization?

Normally, units of data transfer are larger than a single analog or digital encoding symbol. It is necessary to recover clock information for both the signal (so we can recover the right number of symbols and recover each symbol as accurately as possible), and obtain synchronization for larger units of data (such as data words and frames). It is necessary to recover the data in words or blocks because this is the only way the receiver process will be able to interpret the data received; for a given bit stream. Depending on the byte boundaries, there will be seven or eight ways to interpret the bit stream as ASCII characters, and these are likely to be very different. So, it is necessary to add other bits to the block that convey control information used in the data link control procedures. The data along with preamble, postamble, and control information forms a **frame**. This framing is necessary for the purpose of synchronization and other data control functions.

3.1.3.2 Synchronization

Data sent by a sender in bit-serial form through a medium must be correctly interpreted at the receiving end. This requires that the beginning, the end and logic level and duration of each bit as sent at the transmitting end must be recognized at the receiving end. There are three synchronization levels: *Bit, Character and Frame.* Moreover, to achieve synchronization, two approaches known as *asynchronous* and *synchronous* transmissions are used.

Frame synchronization is the process by which incoming frame alignment signals (i.e., distinctive bit sequences) are identified, i.e. distinguished from data bits, permitting the data bits within the frame to be extracted for decoding or retransmission. The usual

practice is to insert, in a dedicated time slot within the frame, a non-information bit that is used for the actual synchronization of the incoming data with the receiver.

In order to receive bits in the first place, the receiver must be able to determine how fast bits are being sent and when it has received a signal symbol. Further, the receiver needs to be able to determine what the relationship of the bits in the received stream have to one another, that is, what the logical units of transfer are, and where each received bit fits into the logical units. We call these logical units *frames*. This means that in addition to bit (or transmission symbol) synchronization, the receiver needs word and frame synchronization.

3.1.3.3 Synchronous communication (bit-oriented)

Timing is recovered from the signal itself (by the carrier if the signal is analog, or by regular transitions in the data signal or by a separate clock line if the signal is digital). Scrambling is often used to ensure frequent transitions needed. The data transmitted may be of any bit length, but is often constrained by the frame transfer protocol (data link or MAC protocol).

Bit-oriented framing only assumes that bit synchronization has been achieved by the underlying hardware, and the incoming bit stream is scanned at all possible bit positions for special patterns generated by the sender. The sender uses a special pattern (a flag pattern) to delimit frames (one flag at each end), and has to provide for data transparency by use of bit stuffing (see below). A commonly used flag pattern is HDLC's 01111110 flag as shown in Fig. 3.1.7. The bit sequence 01111110 is used for both preamble and postamble for the purpose of synchronization. A frame format for bit-oriented synchronous frame is shown in Fig. 3.1.8. Apart from the flag bits there are control fields. This field contains the commands, responses and sequences numbers used to maintain the data flow accountability of the link, defines the functions of the frame and initiates the logic to control the movement of traffic between sending and receiving stations.

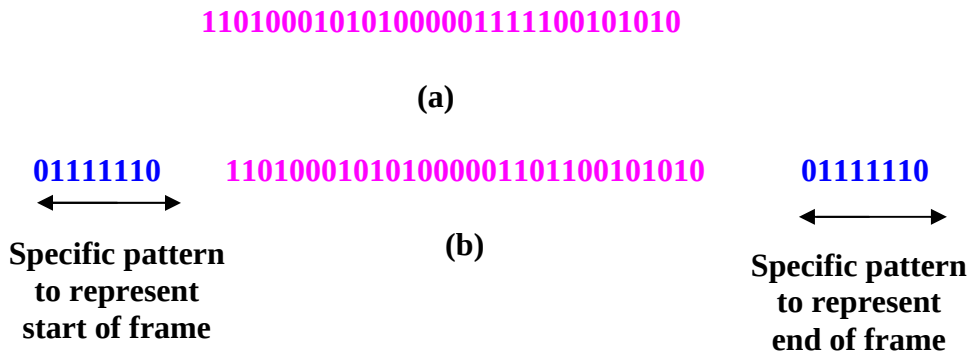


Figure 3.1.7 Bit oriented framing (a) Data to be sent to the peer, (b) Data after being character stuffed.

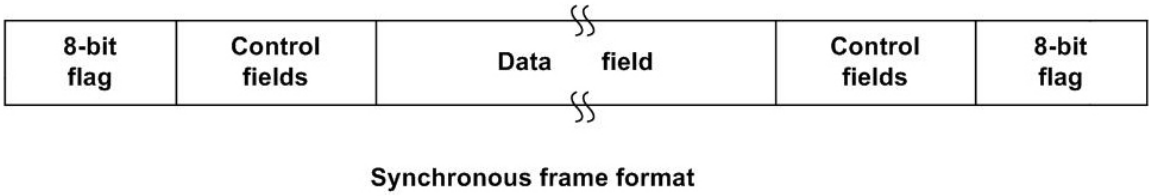


Figure 3.1.8 Frame format for synchronous communication

Summary of the approach:

- Initially 1 or 2 synchronization characters are sent
- Data characters are then continuously sent without any extra bits
- At the end, some error detection data is sent

Advantages:

- Much less overhead
- No overhead is incurred except for synchronization characters

Disadvantages:

- No tolerance in clock frequency is allowed
- The clock frequency should be same at both the sending and receiving ends

Bit stuffing: If the flag pattern appears anywhere in the header or data of a frame, then the receiver may prematurely detect the start or end of the received frame. To overcome this problem, the sender makes sure that the frame body it sends has no flags in it at any position (note that since there is no character synchronization, the flag pattern can start at any bit location within the stream). It does this by *bit stuffing*, inserting an extra bit in any pattern that is beginning to look like a flag. In HDLC, whenever 5 consecutive 1's are encountered in the data, a 0 is inserted after the 5th 1, regardless of the next bit in the data as shown in Fig. 3.1.9. On the receiving end, the bit stream is piped through a shift register as the receiver looks for the flag pattern. If 5 consecutive 1's followed by a 0 is seen, then the 0 is dropped before sending the data on (the receiver destuffs the stream). If 6 1's and a 0 are seen, it is a flag and either the current frame are ended or a new frame is started, depending on the current state of the receiver. If more than 6 consecutive 1's are seen, then the receiver has detected an invalid pattern, and usually the current frame, if any, is discarded.

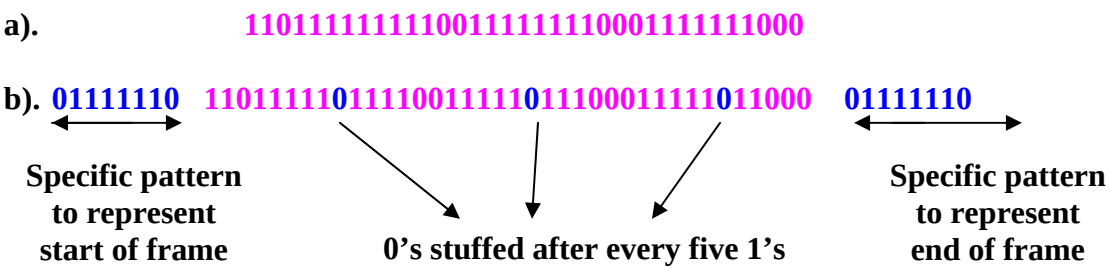


Figure 3.1.9 Bit oriented (a) Data to be sent to the peer, (b) Data after being bit stuffed.

With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern. Thus, if receiver loses track of where it is, all it has to do is to scan the input for flag sequence, since they can only occur at frame boundaries and never within data. In addition to receiving the data in logical units called frames, the receiver should have some way of determining if the data has been corrupted or not. If it has been corrupted, it is desirable not only to realize that, but also to make an attempt to obtain the correct data. This process is called error detection and error correction, which will be discussed in the next lesson.

3.1.3.4 Asynchronous communication (word-oriented)

In asynchronous communication, small, fixed-length words (usually 5 to 9 bits long) are transferred without any clock line or clock is recovered from the signal itself. Each word has a start bit (usually as a 0) before the first data bit of the word and a stop bit (usually as a 1) after the last data bit of the word, as shown in Fig. 3.1.10. The receiver's local clock is started when the receiver detects the 1-0 transition of the start bit, and the line is sampled in the middle of the fixed bit intervals (a bit interval is the inverse of the data rate). The sender outputs the bit at the agreed-upon rate, holding the line in the appropriate state for one bit interval for each bit, but using its own local clock to determine the length of these bit intervals. The receiver's clock and the sender's clock may not run at the same speed, so that there is a relative clock drift (this may be caused by variations in the crystals used, temperature, voltage, etc.). If the receiver's clock drifts too much relative to the sender's clock, then the bits may be sampled while the line is in transition from one state to another, causing the receiver to misinterpret the received data. There can be variable amount of gap between two frames as shown in Fig. 3.1.11.

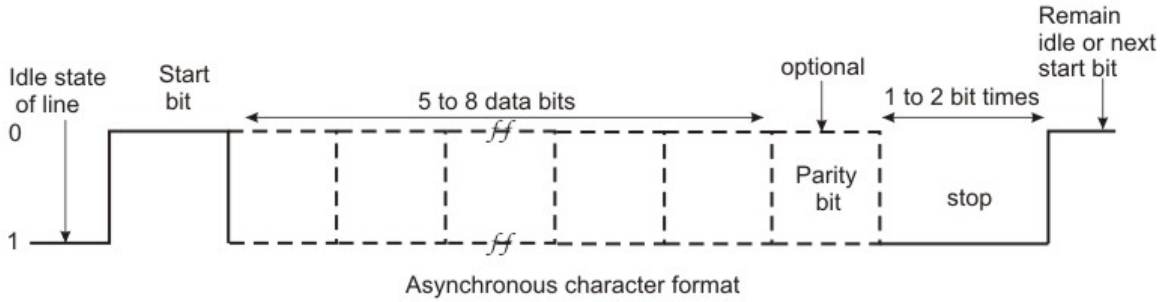


Figure 3.1.10 Character or word oriented format for asynchronous mode

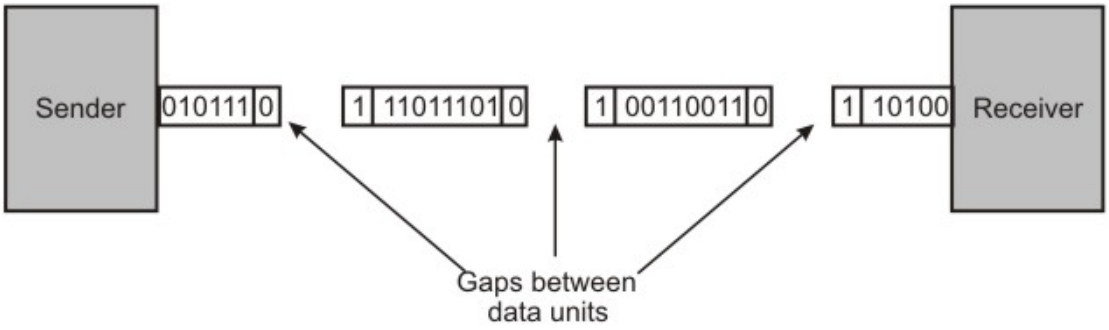


Figure 3.1.11 Data units sent with variable gap sent in asynchronous mode

Advantages of asynchronous character oriented mode of communication are summarized below:

- Simple to implement
- Self synchronization; Clock signal need not be sent
- Tolerance in clock frequency is possible
- The bits are sensed in the middle hence ± ½ bit tolerance is provided

This mode of data communication, however, suffers from high overhead incurred in data transmission. Data must be sent in multiples of the data length of the word, and the two or more bits of synchronization overhead compared to the relatively short data length causes the effective data rate to be rather low. For example, 11 bits are required to transmit 8 bits of data. In other words, baud rate (number of signal elements) is higher than data rate.

3.1.4 Character Oriented Framing

The first framing method uses a field in the header to specify the number of characters in the frame. When the data link-layer sees the character count, it knows how many characters follow, and hence where the end of the frame is. This technique is shown in Fig. 3.1.12 for frames of size 6, 4, and 8 characters, respectively. The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the character count of 4 in the second frame becomes 5, as shown in Fig. 3.1.12(b), the destination will get out of synchronization and will be unable to locate the start of next frame. Even if the checksum is incorrect so the destination knows that the frame is bad, it still had no way of telling where the next frame starts. Sending a frame back to the source and asking for retransmission does not help either, since the destination doesn't know how many characters to skip over to the start of retransmission. For this reason the character count method is rarely used.

Character-oriented framing assumes that character synchronization has already been achieved by the hardware. The sender uses special characters to indicate the start and end of frames, and may also use them to indicate header boundaries and to assist the

receiver gain character synchronization. Frames must be of an integral character length. Data transparency must be preserved by use of character as shown in Fig. 3.1.13.

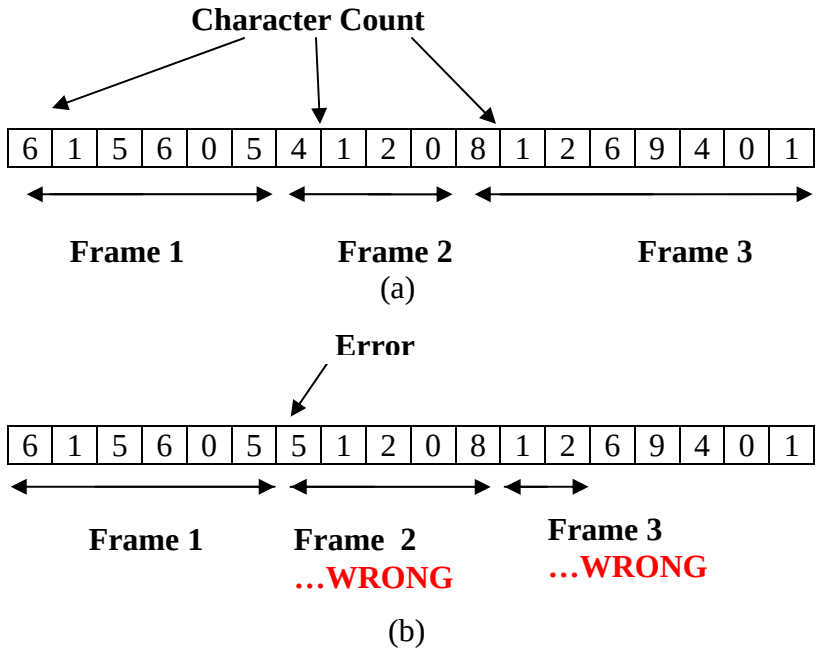


Figure 3.1.12 A Character Stream (a) Without error and (b) with error

Most commonly, a DLE (data link escape) character is used to signal that the next character is a control character, with DLE SOH (start of header) used to indicate the start of the frame (it starts with a header), DLE STX (start of text) used to indicate the end of the header and start of the data portion, and DLE ETX (end of text) used to indicate the end of the frame.

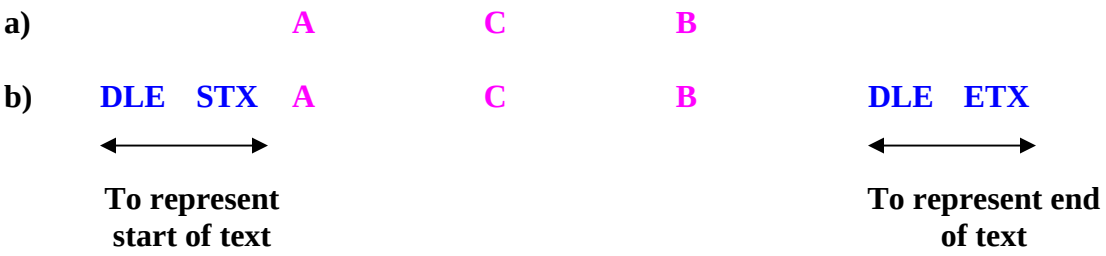


Figure 3.1.13 Character Oriented (a) Data to be send to the peer, (b) Data after being character stuffed

A serious problem occurs with this method when binary data, such as object program are being transmitted. It may easily happen when the characters for **DLE STX or DLE ETX** occur in the data, which will interfere with the framing. One way to overcome this problem is to use character stuffing discussed below.

3.1.4.1 Character stuffing

When a DLE character occurs in the header or the data portion of a frame, the sender must somehow let the receiver know that it is not intended to signal a control character. The sender does this by inserting an extra DLE character after the one occurring inside the frame, so that when the receiver encounters two DLEs in a row, it immediately deletes one and interpret the other as header or data. This is shown in Fig. 3.1.14. Note that since the receiver has character synchronization, it will not mistake a DLE pattern that crosses a byte boundary as a DLE signal.

a)	**DLE**	**STX**	**A**	**DLE**	**B**
	DLE	**ETX**			
b)	**DLE**	**STX**	**A**	**DLE**	**B**
	DLE	**ETX**			
c)	**DLE**	**STX**	**A**	**DLE**	**B**
	DLE	**ETX**			

Figure 3.1.14 Character Stuffing (a). Data send by network layer, (b) Data after being character stuffed by the data link layer. (c) Data passed to the network layer on the receiver side.

The main disadvantage of this method is that it is closely tied to 8-bit characters in general and the ASCII character code in particular. As networks grow, this disadvantage of embedding the character code in framing mechanism becomes more and more obvious, so a new technique had to be developed to allow arbitrary sized character. Bit-oriented frame synchronization and bit stuffing is used that allow data frames to contain an arbitrary number of bits and allow character code with arbitrary number of bits per character.

3.1.4.2 Data Rate Measures

- The raw data rate (the number of bits that the transmitter can per second without formatting) is only the starting point. There may be overhead for synchronization, for framing, for error checking, for headers and trailers, for retransmissions, etc.
- *Utilization* may mean more than one thing. When dealing with network monitoring and management, it refers to the fraction of the resource actually used (for useful data and for overhead, retransmissions, etc.). In this context, utilization refers to the fraction of the channel that is available for actual data transmission to the next higher layer. It is the ratio of data bits per protocol data unit (PDU) to the total size of the PDU, including synchronization, headers, etc. In other words, it is the ratio of the time spent actually sending useful data to the time it takes to transfer that data and its attendant overhead.

The *effective data rate* at a layer is the net data rate available to the next higher layer. Generally this is the utilization times the raw data rate.

3.1.5 DTE-DCE Interface

As two persons intending to communicate must speak in the same language, for successful communication between two computer systems or between a computer and a peripheral, a natural understanding between the two is essential. In case of two persons a common language known to both of them is used. In case of two computers or a computer and an appliance, this understanding can be ensured with the help of a *standard*, which should be followed by both the parties. Standards are usually recommended by some International bodies, such as, Electronics Industries Association (EIA), The Institution of Electrical and Electronic Engineers (IEEE), etc. The EIA and ITU-T have been involved in developing standards for the DTE-DCE interface known as EIA-232, EIA-442, etc and ITU-T standards are known as V series or X series. The standards should normally define the following four important attributes:

***Mechanical*:** The mechanical attribute concerns the actual physical connection between the two sides. Usually various signal lines are bundled into a cable with a terminator plug, male or female at each end. Each of the systems, between which communication is to be established, provide a plug of opposite gender for connecting the terminator plugs of the cable, thus establishing the physical connection. The mechanical part specifies cables and connectors to be used to link two systems

Electrical: The Electrical attribute relates to the voltage levels and timing of voltage changes. They in turn determine the data rates and distances that can be used for communication. So the electrical part of the standard specifies voltages, Impedances and timing requirements to be satisfied for reliable communication

Functional: Functional attribute pertains to the function to be performed, by associating meaning to the various signal lines. Functions can be typically classified into the broad categories of data control, timing and ground. This component of standard specifies the signal pin assignments and signal definition of each of the pins used for interfacing the devices

Procedural: The procedural attribute specifies the protocol for communication, i.e. the sequence of events that should be followed during data transfer, using the functional characteristic of the interface.

A variety of standards exist, some of the most popular interfaces are presented in this section

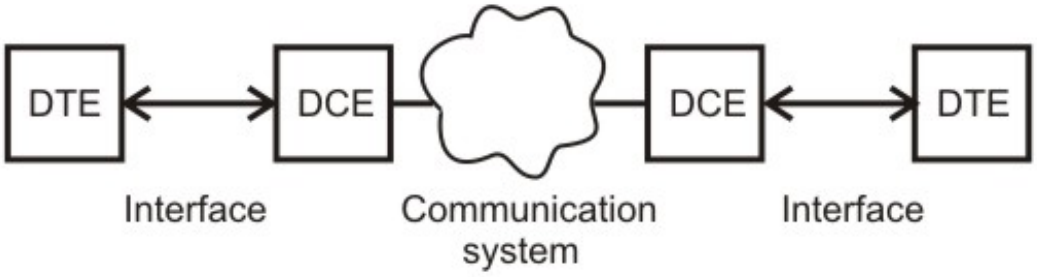


Figure 3.1.15 The DTE-DCE interface

3.1.5.1 The RS-232 C

Most digital data processing devices such as computers and terminals are incapable of transmitting the digital data, which usually is in NRZ-L form, through physical transmission media over long distances. The data processing devices, commonly referred to as *Data Terminal Equipment (DTE)*, utilizes the mediation of another equipment called *Data Circuit communication Equipment (DCE)* to interface with the physical transmission media. An example of a DCE is a MODEM. On the one side, the DCE is responsible for transmitting and receiving bit-serial data in a suitable form for efficient communication through some transmission media such as telephone line. On the other side, the DCE interacts with the DTE by exchanging both data and control information. This is done over a set of wires referred to as interchange circuits. For successful operation of this scheme a high degree of cooperation is required on data processing equipment manufacturers and users, nature of interface between the DTE and DCE. The Electronic Industries Association (EIA) developed the standard RS-232C as an interface between the DTE and DCE as shown in Fig. 3.1.15. Although developed in 1960, it is still widely used for serial binary data interchange. It specifies all the four attributes mentioned above.

Mechanical: A 25-pin connector (DB-25) or 9-pin connector (DB-9) is commonly used for establishing mechanical connection. In most of the applications, however, fewer number of control lines than specified in the standard are used, as not all the systems require their use. The interface established connection between two types of systems, Data terminal Equipment (DTE) and Data communication Equipment (DCE). The equipment that generates, processes and displays the data is called DTE. Computers and monitors are considered as DTEs. A MODEM, which converts digital data into analog form by modulation and also demodulates analog signal to generate digital data, are considered as data communication equipments (DCEs). Modems are used to establish connection through (Transmission media) analog communication channel, such as a telephone line as shown in Fig. 3.1.15.

Electrical: The electrical characteristics specify the signaling between DTE and DCE. It uses single-ended, bipolar voltage and unterminated circuit. The single-ended form uses a single conductor to send and another conductor to receive a signal with the voltage reference to a common ground. The bipolar voltage levels are +3 to + 25V for logic 0 and

–3 to –25V for logic 1. No termination with a resistor either at input or at output is necessary. The most striking feature is that, the voltage levels are not TTL compatible. This necessitates separate voltage supplies and extra hardware for level conversion from TTL-to-RS 232C and vice versa.

The single-ended unterminated configuration is susceptible to all forms of electromagnetic interference. Noise and cross-talk susceptibility are proportional to the cable length and bandwidth. As a result, the RS-232 C is suitable for serial binary data interchange over a short distance (up to 57 ft) and at low rates (up to 20K baud).

Functional: The functional specification of most of the important lines is given in Table 3.1.1. There are two data lines, one for each direction, facilitating full-duplex operation. There are several control and ground lines. The pin number with respect to the connector, abbreviated name and function description of the important lines are given in the table. These nine lines are commonly used.

TABLE 3.1.1 Important RS-232C Pins

Pin No	Function	Short Name
1	Protective ground	
2	Transmit data to DCE	TxD
3	Receive data from DCE	RxD
4	Request to send to DCE	RTS
5	Clear to send from DCE	CTS
6	Data set ready from DCE	DSR
7	Signal ground	
8	Data carrier detect from DCE	DCD
20	Data terminal ready to DCE	DTR

Procedural: The procedural specification gives the protocol, is the sequence of events to be followed to accomplish data communication.

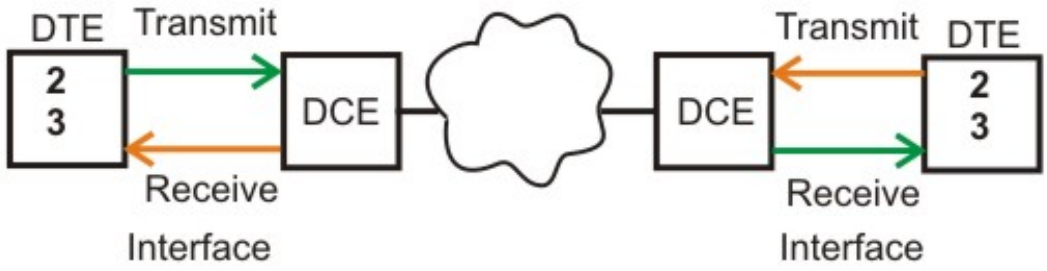
(i) When a DTE is powered on, after self-test it asserts the Data terminal ready (DTR) signal (pin) to indicate that it is ready to take part in communication. Similarly, when the DCE is powered on and gone through its own self-test, it asserts the Data set Ready (DSR) signal (pin 6) to indicate that it is ready to take part in the communication. When the MODEM detects a carrier on the telephone line, it asserts Data carrier detect (DCD) signal (pin 8).

(ii) When the DTE is ready to send data, it asserts request to send (RTS) signal (pin 4). DCE in turn responds with clear to send (CTS) signal (pin 5), when it is ready to receive data and MODEM start sending carrier over the medium indicating that data transmission is eminent. The CTS signal enables the DTE to start transmission of a data frame.

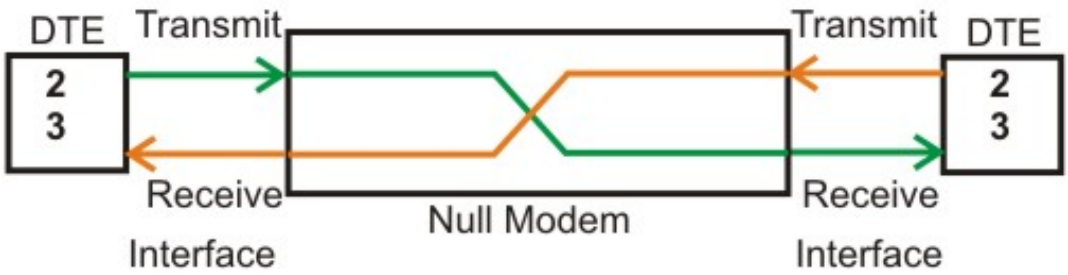
The procedural specification deals with the legal sequence of events on the action-reaction pair of signal lines. For example, the RTS-CTS control lines form an action-reaction pair. Before sending a data, the DTR-DSR pair should be active and then the DTE asserts the RTS signal. In response to this the modern should generate the CTS signal when ready; thereby indicating that data may be transmitted over the TXD. In this manner the action-reaction pairs of lines allows handshaking needed for asynchronous mode of date communication. It also leads to *flow-control*, the rate at which the two systems can communicate with each other.

3.1.5.2 Null Modem

In many situations, the distance between two DTEs may be so close that use of modems (DCE), as shown in Fig. 3.1.16(a), is unnecessary. In such a case the RS-232 C interface may still be used, but with out the DCEs. A scheme known as null modem is used, in which interconnection is done in such a why that both the DTEs are made to feel as if they have been connected through modems. Essentially, null modem is a cable with two connectors at both ends for interfacing with the DTEs. The reason for this behavior is apparent from the swapping interconnection shown in Fig. 3.1.16(b).



(a)



(b)

Figure 3.1.16 Null modem

3.1.5.3 MODEMS

The DCE that is used to interface with the physical transmission media is known as MODEM, derived from MOdulator + DEModulator. The *modulator* converts digital data

into an analog signal using ASK, FSK, PSK or QAM modulation techniques discussed in the previous lesson. A *demodulator* converts an analog signal back into a digital data. Important Parameters of the modems are the *transmission rate* and Bandwidth (Baud rate). The output of a modem has to match the bandwidth of the bandwidth of the medium, the telephone line as shown in Fig. 3.1.17.

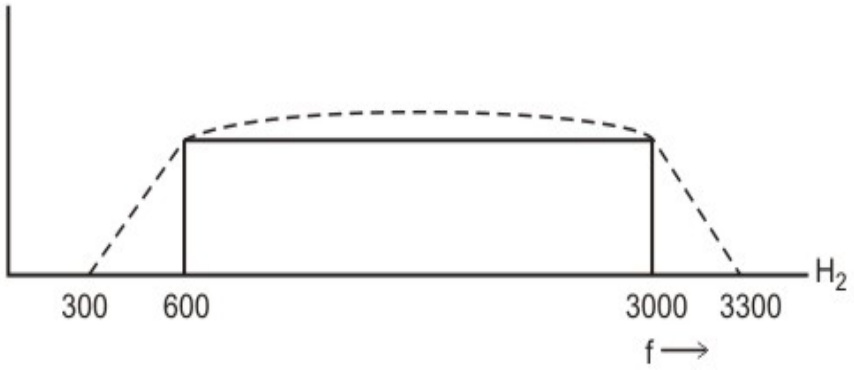


Figure 3.1.17 Bandwidth of the telephone line

Fill In The Blanks:

1. Frame synchronization is the process by which incoming frame _____ are identified.
2. We need Framing and synchronization to recover _____ information.
3. In addition to bit (or transmission symbol) synchronization, the receiver needs _____ and _____ synchronization.
4. The number of bits that the transmitter can per second without formatting is known as raw _____.
5. The ***effective data rate*** at a layer is the net data rate available to the next _____ layer.
6. Character Count uses a field in the header to specify the number of _____ in the frame.

Short Question Answer

1. **Distinguish between data rate and baud rate?**

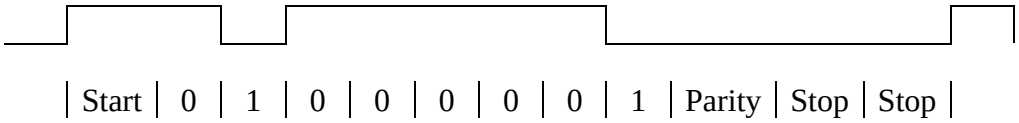
Ans: Data rate is the rate, in bits per second (bps), at which the data can be communicated.
Baud rate is the rate of transmitting the signal elements including the bits required for synchronization.

2. **In what way synchronous and asynchronous serial modes of data transfer differ?**

Ans: In asynchronous serial modes of data transfer, the data is transmitted in small sizes (or frames) along with start bit, parity bit and stop bit for synchronization. This mode of transmission is self-synchronizing with moderate timing requirement. The major drawback is high overhead.
In synchronous mode, the clock frequency of the transmitter and the receiver should be same and hence complex hardware is required to maintain this rigorous timing. But this approach is efficient with very less overhead.

3. **The ASCII character `A' (41H) is sent using RS-232C interface in asynchronous mode. Draw the time domain graph assuming baud rate of 110 bits per second.**

Ans A (41H): 01000001
Frame size = 1+8+1+2 = 12



4. **What is bit-stuffing? Why is it used?**

Ans: Bit-stuffing: In case of synchronous transmission, flag bits (8-bit length sequence 01111110) is attached at the beginning and end of each frame. These flag bits are used for synchronization. It may so happen that the flag bit sequence may appear somewhere inside the frame and this will cause a problem for synchronization. To avoid this problem a process called *bit-stuffing* is used, in which extra bit is stuffed if flag bit like sequence appears inside the frame. For example, in HDLC the transmitter introduces a 0 after each occurrence of five 1's in the data sequence by. At the receiving end these extra 0's are removed.