MP, NCIT

Chapter-5
Data Communication Basics

Com. EIX Batch

- Peripherals connected to a computer need special communication links for interfacing them with the central Processing Unit (CPU).
- The purpose of those communication links are to resolve the differences that exist between the processor and each peripherals.
- The major differences are:-

(I) Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from operation of CPU and memory, which are electronic devices. So, the conversion of signal values is required.

(II) The data transfer rate of peripherals is usually slower than the transfer rate of CPU, and consequently synchronization mechanism may be needed.

(III) The operating modes of peripherals are different from eachother and each must be controlled so as not to disturb the operation of other peripherals connected to CPU.

(IV) Data codes and formats of peripherals differ from that of CPU and memory.

- To resolve these differences, computer system include special hardware components between

the CPU and peripherals, to supervise and synchronize all inputs and outputs transfer. These components are called Interface Units, because they interface between the processor bus and peripheral devices.

- The two major types of I/O Interfaces are:
  (a) Serial Interface, (b) Parallel Interface

(a) Serial Interfacing:

- exchanges data with the peripherals in serial mode
- data are transmitted one bit at a time.
- slow, inexpensive to implement.
- serial interface converts parallel mode bus system to serial mode
- If the bus has 'n' data lines, the serial I/O interface accepts 'n' bits of data simultaneously from the bus. These 'n' bits are sent to the I/O devices, one bit at a time, requiring 'n' time slots for transmission.

  eg. Keyboard Interfacing.

(b) Parallel Interfacing:

- Some I/O devices can handle data at speeds that cannot be supported by serial interfaces. In such case, parallel interface is required.
- In parallel interface, 'n' bits of data are handled simultaneously by the bus and on the links to the device.
- faster interchange of data, in this interface, and is expensive, due to the need of multiple wires.
- Many I/O devices, particularly those requiring high data transfer rate use this arrangement.

  eg. Printer Interface

Communication modes:

- Communication can be broadly defined as unilateral or bilateral transfer of meaningful data between two points, through a medium.
- Communication can be in:-
  (i) Simplex mode,
  (ii) Half Duplex mode, and
  (iii) Full Duplex mode.

(i) Simplex mode:

Transmission of a data over a single channel in one direction only.

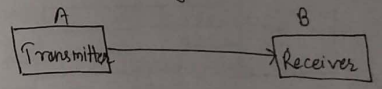


fig (a) Simplex mode.

eg. commercial radio stations (AM or FM radio).

(ii) Half-duplex mode:

- Transmission of data over a single channel in one direction, but only one at a time.

  eg. walki-twalki.

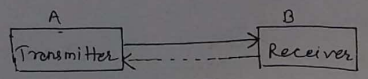


Fig (b) Half-duplex mode

(iii) Full-duplex mode:

- Simultaneous transmission of data in both directions. eg. mobile phone, telephone, etc

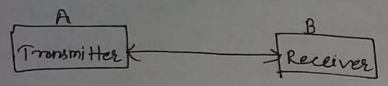


Fig. (c) Full duplex mode

Methods of Communication:

① Parallel Communication:

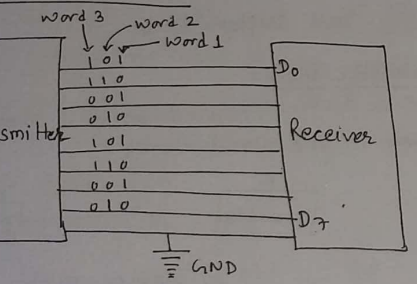


Fig. (a) Parallel data transfer

- A word of 'n' bits is transmitted in parallel.
- Channel comprises of n-lines, last line is called common ground.
- Time required to transfer one word is equal to time taken for a bit to transmit.

- Impractical, for long distance transmission, since costs for large number of lines increases.

② Serial Communication:

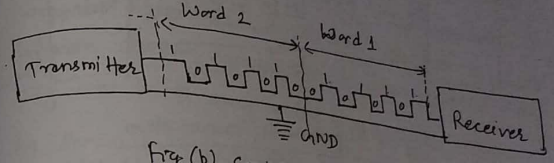


Fig (b) Serial data transfer.

- In serial data transfer, each bit of the word is sent in succession, one at a time over a single pair of wires.
- A parallel to serial converter is used to convert the incoming parallel data to serial form and then the data is sent out with the Least significant Bit, D_0 first and Most significant Bit, D_7 at last.
- The time taken to transmit a word in serial data transmission will be 'n' times more than the time taken in parallel data transmission.

Types of Serial Data Transfer:

① Asynchronous Serial Data Transfer,

② Synchronous Serial Data Transfer

① Asynchronous Serial Data Transfer:

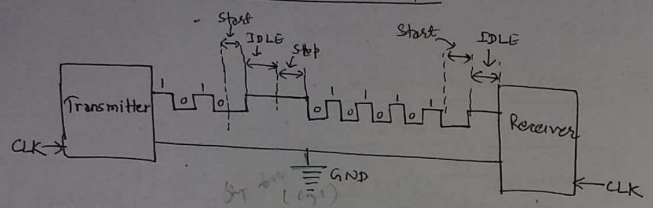


Fig (a) Asynchronous Communication

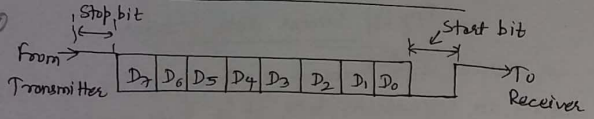
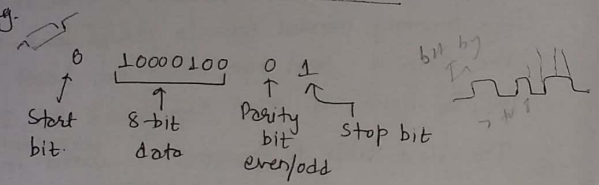


Fig (b) Transmission format for Asynchronous Transfer

eg.



→ In this type of transmission, the receiving device doesnot need to be synchronized with the transmitting device.

→ The transmitting device can send one or more data units when it is ready to send.

→ Each data units must be formatted. In other words, each data unit must contain a 'start bit' and 'stop bit (or bits)' indicating the beginning and the end of each data unit.

→ When no data are sent over the line, it is maintained at an idle value, a logic 1.

→ start bit is a logic 0 and the stop bit is a logic 1.

→ Both the transmitter and receiver are given a separate clocking signals (CLK). It is not essential that they both be of exactly the same frequency.

→ As shown in fig (a), when the character is to be transmitted, the transmitter first sends out a low bit. This transition is perceived by the receiver and it gets ready to receive the data.

Then transmitter sends out the word bit by bit, one after the other, synchronous with its own clock.

The receiver assembles the data one by one synchronous with its own clock. Before the transmission starts, the receiver must know the transmission 'baud rate' for proper assembling at its end.

After all the bits of a character are sent, the transmitter sends out a stop bit, which is idle value (logic 1), to indicate the end of transmission.

→ Data may consists of 5 to 8 bits.

→ Asynchronous serial data communication is generally used for low speed data transmission (speed less than Kilobit per second).

② Synchronous Serial Data Transfer:

- Synchronous communication is used for transferring large amount of data at a stretch without frequent start or stops.

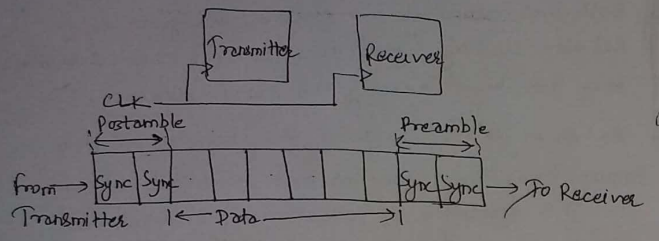


Fig. Transmission format for Synchronous transfer

- Receiver (Rx) and Transmitter (Tx) are synchronized by a master clock ie both depend upon same clock signal.
- A block of characters is transmitted along with synchronization information.
- Usually one or two synchronizing characters (SYNC) are used to indicate the start of each synchronous data stream.
- The transmitter sends the data character by character, bit by bit. After sending all the characters, the transmitter sends another pattern of bit, to indicate the end of transmission.
- Synchronous Serial Data Transfer is generally used for high speed transmission greater than 20K per second.

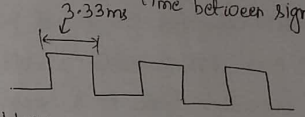
Baud Rate:

- The term 'baud rate' is used to indicate the rate at which serial data is being transferred.
- The rate at which the bits are transmitted (bits per second) is called a baud, technically, however, it is defined as the number of signal changes/second.

ie.

$$\text{Baud rate} = \frac{1}{\text{time between signal transition}}$$

eg.



Here,

$$\text{Baud rate} = \frac{1}{3.33\text{ms}} = 300.3 \text{ Bd (Bits/second)}$$

RS232C Serial Data Standard: ----- (X) Imp

- Defined by EIA (Electronic Industries Association).
- This standard describes the function of 25 signal and handshake pins for serial-data transfer. It also describes the voltage levels, impedance levels, rise and fall times, maximum bit rate and maximum capacitance for these signal lines.
- Also, it is an interface convention developed to standardize the interfaces between Data Terminal Equipment (DTE) [eg terminals or computers that are sending or receiving data] and Data Communication Equipment (DCE) [eg modems and other devices used to send serial data], employing serial binary data exchange.

→ RS232C standard is used for serial communication.

→ RS232C works in a negative logic.

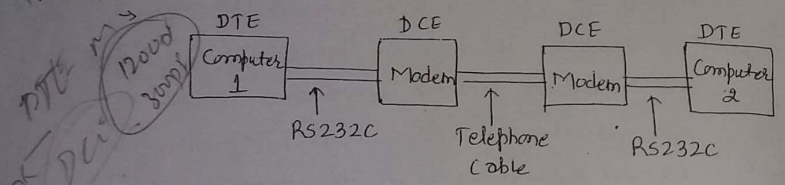
→ Voltage levels for RS232C:

Logic High - Mark → (-3V to -25V, no load)
→ (-3V to -15V, under load)

Logic Low - Space → (+3V to +15V, under load)
→ (+3V to +25V, no load)

→ RS232C standard specifies 25 signal pins and it also specifies that DTE connector should be a male and the DCE connector should be a female.

→ A specific connector is not given, but the most commonly used connectors are the DB-25P male (with 25 pins) and a DE-9P male (with 9 pins) where 25 pins are not needed.



→ Transmission line normally use a twisted pair of shielded wire, with line capacitance of no more than 1200pF and no less than 300pF.

→ The standard specifies the line length to 50ft only at maximum data rate of 20 Kbaud.

-3V to -25V no
-3V to -15V

Pin configuration of RS232C (DE-9P)

Pin	Description
1	Data Carrier Detect ($\overline{CDCD}$)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data Terminal Ready (DTR) → [computer to modem]
5	Signal Ground (GND)
6	Data Set Ready ($\overline{DSR}$) → [modem to computer as it is ready.]
7	Request to Send ($\overline{RTS}$) → [computer to modem]
8	Clear to Send ($\overline{CTS}$)
9	Ring Indicator (RI)

For DB-25P ie. 25 pins, the important pins are as follows:-

Pin No.	Signals	Functions
2	Transmit Data, TxD	o/p, transmit data from DTE to DCE
3	Receive Data, RxD	i/p, DTE receives data from DCE.
4	Request to Send, RTS	general purpose output from DTE.
5	Clear to Send, CTS	Input to DTE, used as handshake
6	Data set Ready, DSR	I/p to DTE, indicates that DCE is ready.
7	Signal Ground, GND	Common reference betn DTE and DCE
8	Data Carrier Detect, DCD	Used by DTE to disable data reception.
20	Data Terminal Ready, DTR	o/p, means DTE is ready.

→ The main problem with RS232C is that it can only transfer data reliably about 50ft (16.4m) only at a maximum rate of 20kbaud. If longer lines are used, the ~~transmitter~~ transmission rate is drastically reduced.

→ For high speed transmission, the standards RS422A and RS423A are used.

RS232C Interface with DTE and DCE:

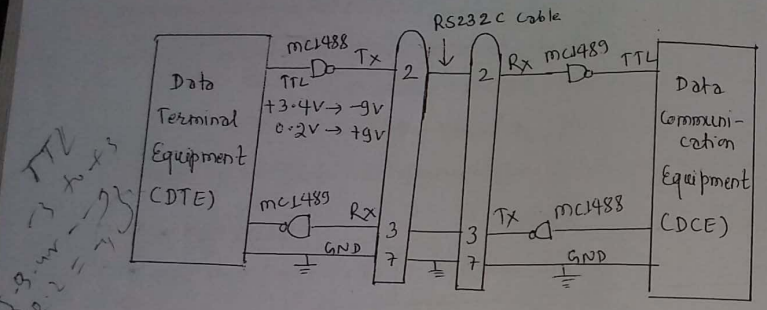


Fig RS232C Interfacing with minimum lines.

→ The signalling in RS-232C is not compatible with TTL logic level.

For TTL	For RS232C
0V to 0.2V → logic 0	+3V to +15V → logic 0
3.4V to 5V → logic 1	-3V to -15V → logic 1.

→ The voltage translators called line drivers and line receivers are required to interface TTL logic with RS232C signal.

→ The line driver, MC1488 converts logic 1 to approximately -9V and logic 0 into +9V.

→ Before, it is received by the DCE, it is again converted by the line receiver MC1489 into TTL compatible logic

Why there is signal conversion?

→ It is because:

- RS232C standard came into existence before TTL level.
- High level of noise margin at -3V to +3V.

So,

low range { 3.4V → -9V, 0.2V → +9V } high range

→ The minimum interface between computer and a ~~computer~~ peripheral requires three line pins 2, 3 and 7.

DTE → transmits on pin 2 and receives on pin 3.

DCE → transmits on pin 3 and receives on pin 2.

Null-modem

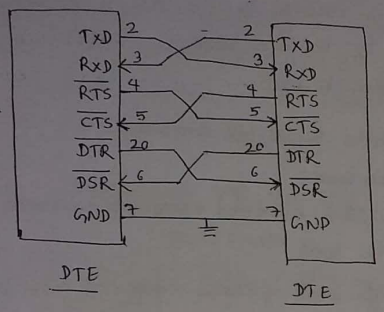


Fig. Null modem for connecting two RS-232C data terminal type devices

→ Here, the TxD of the terminal sends data to RxD input of the computer and vice-versa

→ No modems used, so cheap connection.

Comparison of Serial I/O Standards:

Specifications	RS-232C	RS-422A	RS-423A
Speed	20 Kbaud	10 Mbaud at 40ft 100 Kbaud at 4000ft	100 Kbaud at 30ft 1 Kbaud at 4000ft
Distance	50ft	4000ft	4000ft.
Logic 0	> +3V to +25V	B > A*	+4 to +6V
Logic 1	< -3V to -25V	B < A	-4 to -6V
Receiver Input Voltage	±15V	±7V	±12V

* B and A are differential input to op amp.

Binary Synchronous Communication Protocol - BISYNC

→ BISYNC is referred to as a byte-controlled protocol (BCP), because specified ASCII or EBCDIC characters (bytes) are used to indicate the start of the message and to handshake between the transmitter and receiver.

→ Incidently, even in a full-duplex system, BISYNC protocol allows data transfer only in one direction at a time.

→ The general message format for BISYNC is as given below in fig (a):-

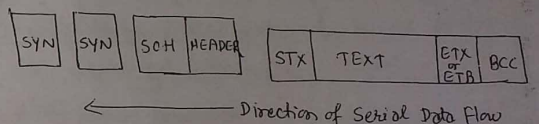


Fig (a) General message format for binary synchronous communication (BISYNC)

→ First of all, before transmission, the transmitter must receive a message from receiver that it is ready to receive a transmitted data.

→ If no message is being sent, the line is an 'idle" condition with a continuous high on the line.

→ To indicate the start of the message, the transmitting system sends two or more previously agreed upon sync characters (eg 16H)

→ The receiver uses these SYN (sync)-characters to synchronize its clock with that of transmitter.

→ A 'header' may then be sent if desired. The header contents are usually defined for a specific system and may include information about the type, priority, and destination of the message that follows.

→ The start of the header is indicated with a special character called Start of Header (SOH), which is represented in ASCII by '01H'.

→ After the header, if present, the beginning of the text portion of the message is indicated by another special character called start-of-text (STX), which in ASCII is represented by 02H.

→ To indicate the end of the text portion of the message, an end-of-text (ETX) character or an end-of-block (ETB) character is sent.

→ The text portion may contain 128 or 256 characters.

→ Immediately following the ETX, character 1 or 2 block check characters (BCC) are sent.

→ For systems using ASCII, the BCC is a single byte which represents complex parity information computed for the text of the message.

→ For systems using EBCDIC, a 16-bit 'cyclic redundancy check' is performed on the text part of the message and the 16-bit result sent as 2 BCCs.

→ The purpose of BCCs is that the receiving system can recompute the value for them from the received data and compare the results with the BCCs sent from the transmitter. If BCCs are not equal, the receiver may ask the transmitter to send the message again.

Handshake for data transfer in BISYNC:

→ To illustrate, the data transfer handshaking between the transmitter and receiver, we assume that we have a "smart" terminal connected to a computer with a half-duplex connection, also assume computer in receive mode.

→ Now, when the terminal determines that it has a block of data to send to computer, it first sends a message with text containing only the single character ENQ (ASCII 05H), which stands for enquiry.

→ The terminal then switches to receive mode to await the reply from computer.

→ The computer reads ENQ message, and if it is not ready to receive data, it sends back a text message containing the single character for negative acknowledge, NAK (ASCII 15H). If it is ready, it sends a message containing affirmative acknowledge (ASCII 06H) for ACK.

→ In either case, the computer then switches to receive mode to await the next message from the terminal.
→ If the terminal received NAK, it may give up or wait a while and try again.
→ If the terminal received ACK, it will send a message containing a block of text and ending with BCC.
→ After sending message, the terminal ~~sends~~ switches to receive mode and awaits a reply from computer as to whether the message was received correctly.
→ The computer meanwhile computes the BCC for the received block of data and compares it with BCC sent with the message. If two BCCs are not equal, computer sends NAK message to terminal. Then, terminal sends the message text again to the computer
→ If two BCCs are equal, then computer sends ACK message to terminal, which tells it to send next message or block of text.
→ For successive blocks, ~~ACK0&ACK1~~, ACK0 and ACK1 are sent alternatively.
→ A variation of BISYNC commonly used to transfer binary files in PC environment and between Unix systems and PCs is called XMODEM.
→ One major problem with BISYNC protocol is that the transmitter must stop after each block of data is transferred and wait for an ACK or NAK signal from receiver.
→ Due to wait and line turnaround times, the actual data transfer rate may be only half of the theoritical rate predicted by physical rate of data link.
→ The HDLC (High Level Data Link Control) greatly reduces this problem.