Instrumentation II (III/I)

Course Code: EX-602
(Module#3)

Dhawa Sang Dong
(Lecturer)

Kathmandu Engineering College
Kalimati, Kathmandu

May 2024

Chapter#3

Serial Interfacing with Microprocessor Based System

✓ Class Outline

1. Advantages of Serial Data Transfer over Parallel Transfer
2. Synchronous and Asynchronous Data Transfer
3. Errors in Serial Data Transfer
4. Simplex, Half Duplex and Full Duplex Data Transfer
5. Serial Standards RS232, RS423 and RS422
6. Universal Serial Bus (USB)
7. USB Bus, Signal Throughput, Protocols

Advantages of Serial Data Transfer over Parallel Transfer

Serial Data Transfer

- For distant communication or data transfer, parallel data transfer is costly due to many wires, and higher loss probability.
- Therefore, parallel data is converted into serial at transfer.
- Data is transmitted over single line or pair of wires.
- at receiving terminal, serial data is converted back to parallel form and processed further.

Advantages of Serial Data Transfer over Parallel Transfer

Advantages of Serial data Transfer

1. Voltage drop is not a serious problem/issue:
   - ✓ Serial: 1 $\longrightarrow$ -3V to -25V; 0 $\longrightarrow$ +3V to +25V
   - ✓ Parallel: 1 $\longrightarrow$ +5V; 0 $\longrightarrow$ 0V
2. Serial transmission needs less number of wires reducing cost of transmission lines.
3. cross-talk is of small issue in serial transmission compared to parallel transmission.
4. many ICs and peripherals have serial interface.

Advantages of Serial Data Transfer over Parallel Transfer

Advantages of Serial data Transfer...

5. clock skew (degradation transfer speed due to lowest speed line) between different cables is not a issue.
6. serial data transfer can be longer distant transmission compared to parallel transmission.
7. cheaper implementation compared to parallel transmission.

Note*: But one of the major issue of serial data transmission is slow data transmission rate.

Synchronous and Asynchronous Data Transfer

Serial Data Transmission

- Data are sent one bit at a time over the serial channel.
- Receiver has to wait for all data bits to be received if bit-processing is not possible (adding delay time).
- serial transmission can be synchronous or asynchronous.

Serial Synchronous Data Transmission

- data is received or transmitted (continuously and consistently) based on clock signal,
- data bits are sent at each clock pulse at specific data rate.
- interpretation is possible only when start and end of each data block/frame is known to receiver.

Synchronous and Asynchronous Data Transfer

Serial Synchronous Data Transmission ...

- as indication to start of data unit, transmitter must send SYNC character which could be one or more SYNC character.
- sometime, SYNC can be replaced with unique bit pattern depending upon the communication protocol established which is known as flag.
- receiver has to wait for SYNC or flag before interpretation of data received.
- if the transmitter is not ready to send data, the line is held in marking condition (different bit pattern than SYNC or flag).
- there is no gap between characters being transmitted.

Synchronous and Asynchronous Data Transfer

Serial Synchronous Data Transmission ...

- data transmission is faster because of no start and stop bits.
- for higher speed and synchronization, there is complex interfacing logic or circuit.
- data will be interpreted in wrong way when devices are out of synchronous.

Advantages and Disadvantage: | Synchronous Communication

- Advantages: higher the communication, higher the bandwidth and peripherals.
- Disadvantages: possibility of inaccuracy when transmitter and receiver are out of sync, so periodic sync is needed.

Synchronous and Asynchronous Data Transfer

Serial Synchronous Data Transmission ...

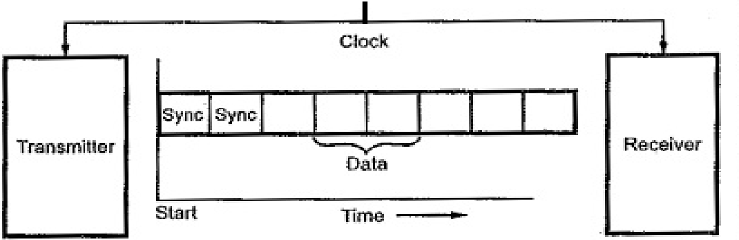


Fig. 1 Synchronous Serial Transmission Format

- data is not sent as individual bytes rather in a large data block or data frame.
- SYNC or flag is used to start or terminate the transmission.

Synchronous and Asynchronous Data Transfer

Serial Asynchronous Data Transmission

- Transmitter and receiver are not synchronized.
- transmitter sends data character by character, i.e one data unit at a time.
- each data unit starts with start bit and end with stop bit.
- it also includes one parity bit to indicate even or odd parity of data – for error detection at receiver side.
- for an ASCII character, data unit contains:
    - ✓ 1 start bit
    - ✓ 7 or 8 bit character
    - ✓ 1 parity bit
    - ✓ 1 or 2 stop bit.

Synchronous and Asynchronous Data Transfer

Serial Asynchronous Data Transmission...

- when there is no data over the line, there is constant high.
- to indicate start of data unit, line goes low for one bit (time), then actual data unit is sent.
- while sending data, least significant bit(LSB) is send first.
- after data bit and parity bit, the signal line goes high to indicate stop.

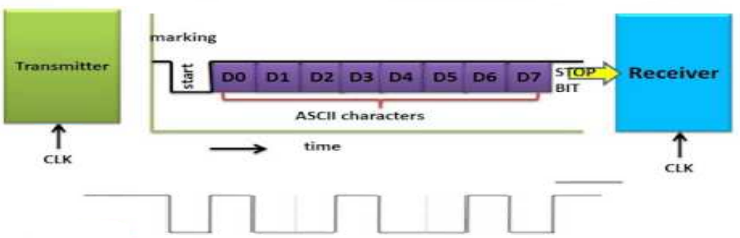


Fig. 2 Asynchronous Serial Transmission Format

Synchronous and Asynchronous Data Transfer

Serial Asynchronous Data Transmission...

- since there is start-bit and stop-bit, there might be gap between two data unit.
- all bits including start-bit, stop-bit, and parity-bit determine the baud rate.
- generally stop-bit and start-bit includes gaps to allow transmitter and receiver synchronize the data transmission.

Note:

✍ Generally, asynchronous communication is preferred for slow speed peripherals to communicate with computer.

✍ It does not need complex and costly hardware as compared to synchronous transmission.

Synchronous and Asynchronous Data Transfer

Serial Data Unit (SDU) and Serialization

- ✓ 1 start bit $\longrightarrow$ always low
- ✓ 7 or 8 bit data unit
- ✓ 1 parity bit
- ✓ 1 or 2 stop bit $\longrightarrow$ always high.

Baud Rate	**Time**
110	9.09 ms
300	3.33 ms
1200	833 μs
2400	417 μs

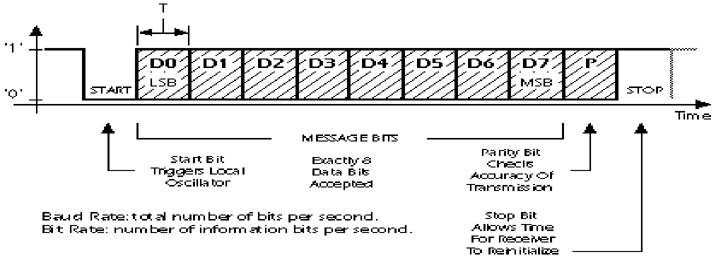


Fig. 3 SDU or Frame Format

Synchronous and Asynchronous Data Transfer

Serial Data Unit (SDU) and Serialization ...

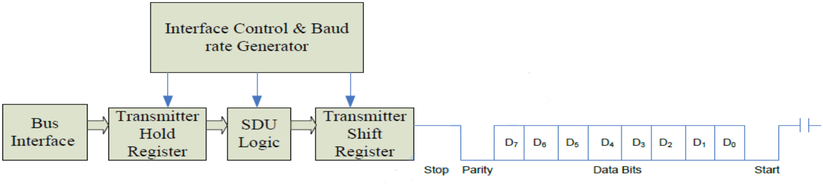


Fig. 4 SDU at transmitting side

- Output Buffer (TX Hold Register) first will be loaded with data from CPU fetched by interface circuit.
- According to SDU format, SDU logic puts the start bit at first, and calculate the parity.
- it appends the parity bit to the MSB data bit then stop bit.
- then data is transferred to transmitter shift register.
- For no data, transmitter shift register possesses a logic high.

Synchronous and Asynchronous Data Transfer

Serial Data Unit (SDU) and Serialization ...

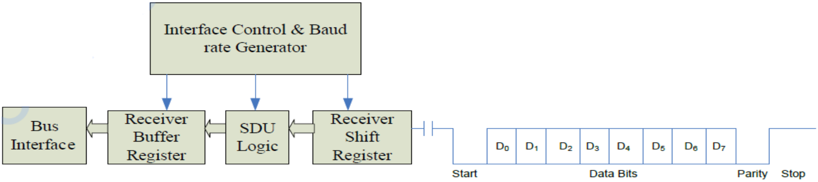


Fig. 5 SDU at receiving side

- Inverse process will go at receiving side
- start bit (logic 0) act as trigger to receive the serial data.
- Firstly, SDU bits are loaded into receiver shift register
- Receiving SDU Logic separates the start, stop, and parity bits
- extracted data bits are then transferred to receiver buffer register from which CPU reads the data byte as received data.

Errors in Serial Data Transfer

- For proper communication, data format and baud rate must be coincide, receiver may interpret data byte differently, otherwise.
- upon receiving SDU, interpretation may involve various error
  1. Framing Error
  2. Break Error
  3. Overrun Error
  4. Parity Error

Framing Error

- non-synchronous start and stop bit may cause unfit frame
- mainly bit loss at the receiving end.

Break Error

- When there is logic low for long time than usual SDU, receiver perceives lost connection.
- only when transmitter send high (signaling no data), receiver reversed the connection loss interpretation.

Errors in Serial Data Transfer

Overrun Error

- when incoming bit rate is greater than data reading rate by receiver, some of the older data might get overwritten by the latter data on receiver buffer.

Parity Error

- when calculated parity is not same as defined by parity bit then error is parity error.
- This parity error is used to detect the error in received data.

7 bit data	**Count of '1'**	**Even-Parity**	**Odd-Parity**
0000000	0	0000000**0**	0000000**1**
1010001	3	1010001**1**	1010001**0**
1101001	4	1101001**0**	1101001**1**

Errors in Serial Data Transfer

Error Check in Data Communication

- noise at transmission line or different clocks between transmitter and receiver cause changes in data bits.
- to check if there is error in data received, additional error checking bits are added at transmitting end – redundant bits.
- if the receiver detect the error, the receiver either request for re-transmission or correct the error bits using proper error correction coding techniques.
- some common error checking practices are:
  1. Parity Check
  2. Checksum
  3. Cyclic Redundancy Check

Errors in Serial Data Transfer

Parity Check: | Error Check in Data Communication

- this is the simplest method of error check by counting 1s in data byte to be transmitted.
- in this method; for instance, if ASCII code is to be transmitted, either D_7 is used for parity information or extra bit is added as parity bit at the beginning.
- parity may be either Even Parity or Odd Parity.

ASCII Code	Count of '1'	Even-Parity	Odd-Parity
A/01000001	2	**0**1000001	**1**1000001
C/01000011	3	**1**1000011	**0**1000011
A/01000001	2 ✓ extra bit	**0**01000001	**1**01000001
C/01000011	3 ✓ extra bit	**1**01000011	**0**01000011

Note: ✍ For even number of bit error (2/4/6..) and error in parity bit itself have no solution – error can't be detected.

Errors in Serial Data Transfer

CheckSum: | Error Check in Data Communication

- it involves addition of all data bytes in block and making 1's complement if carry is used else 2's complement.
- the check sum byte is used at receiving end and if the sum of checksum and addition of data byte received results zero when complemented, data is free of error.
- when there is two bit changed in different data byte but at the same bit position, error could not be detected.

Binary representation of 5 is: 0 1 0 1

1's Complement of 5 is: 1 0 1 0

2's Complement of 5 is: (1's Complement + 1) i.e.

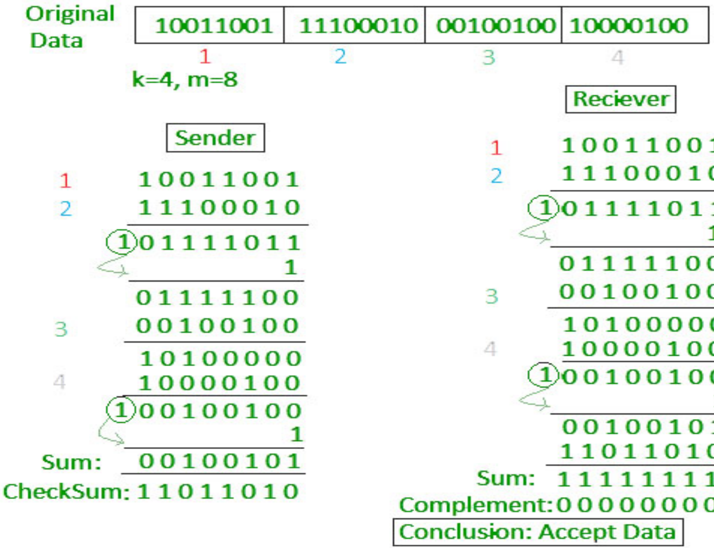
1 0 1 0 (1's Compliment)

+ 1

1 0 1 1 (2's Complement i.e. -5)

Errors in Serial Data Transfer

CheckSum: | Error Check in Data Communication



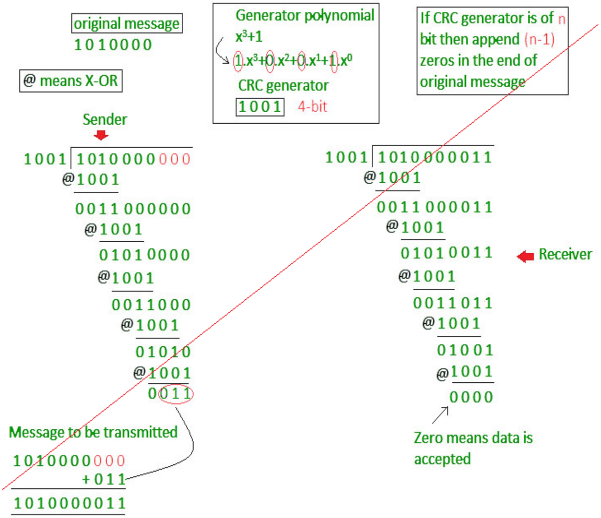
Errors in Serial Data Transfer

Cyclic Redundancy Check (CRC): | Error Check

- in this method, the data stream is represented as polynomials and is divided by fixed polynomials (generator polynomial)
- remainder is append to the data stream and used as information to detect the error.
- at receiver side, data stream is divided by the same generator polynomials.
- if the remainder is zero, data is error free else corrupted data; data can be requested re-transmission.
- while doing division, append n-1 bit for generator of n bit and start generating code-word.

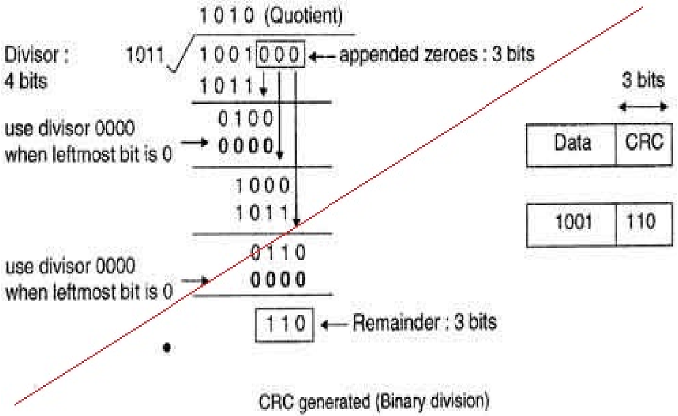
Errors in Serial Data Transfer

Cyclic Redundancy Check (CRC): | Error Check



Errors in Serial Data Transfer

Cyclic Redundancy Check (CRC): | Error Check



CRC generated (Binary division)

Errors in Serial Data Transfer

Cyclic Redundancy Check (CRC): | Error Check

Information: (1,1,0,0) $\longrightarrow$ $i(x) = x^3 + x^2$

Generator polynomial: $g(x) = x^3 + x + 1$

Encoding: $i(x).x^3 = x^6 + x^5$

$$
\begin{array}{r}
x^3 + x^2 + x \\
\hline
x^3 + x + 1 \;\big)\; x^6 + x^5 \\
x^6 + \qquad x^4 + x^3 \\
\hline
x^5 + x^4 + x^3 \\
x^5 + \qquad x^3 + x^2 \\
\hline
x^4 + \qquad x^2 \\
x^4 + \qquad x^2 + x \\
\hline
x
\end{array}
\qquad\qquad
\begin{array}{r}
1110 \\
\hline
1011 \;\big)\; 1100000 \\
1011 \\
\hline
1110 \\
1011 \\
\hline
1010 \\
1011 \\
\hline
\boxed{010}
\end{array}
$$

Transmitted codeword:

$b(x) = i(x).x^3 + r(x) = x^6 + x^5 + x$

$\longrightarrow$ $\underline{b} = (1,1,0,0,0,1,0)$

Errors in Serial Data Transfer

Baud Rate/Bit Rate

- Baud Rate: number of symbol per second or change of the state per second.
- Bit Rate: number of bit per second.

Special Case:

- when the system has only two symbols with representation by '1' or '0' then baud rate is equivalent to bit rate.

Simplex, Half Duplex and Full Duplex Data Transfer

- Depending upon the direction of transmission, data communication can be categorized as:

  ① Simplex, ② Half-Duplex, ③ Full-Duplex

Simplex Mode:

- It is a single direction transmission; data flows towards specified direction only.
- television/radio broadcasting are simplex transmission.
- there is no back communication or data transmission or even acknowledgment to transmitter.

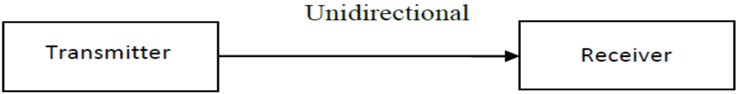


Fig. 6 Simplex Mode Communication

Simplex, Half Duplex and Full Duplex Data Transfer

Hal-Duplex Mode:

- Transmission of data byte in one way only at a time.
- Radio phone by police – only one person at a time
- there is no back communication once one of the party finished its message.

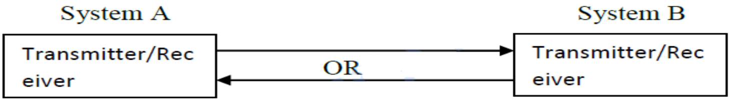


Fig. 7 Half-Duplex Mode Communication

Simplex, Half Duplex and Full Duplex Data Transfer

Full-Duplex Mode:

- Two way communication simultaneously.
- needs higher bandwidth or two way channel for communication.

Fig. 8 Full-Duplex Mode Communication

Serial Standards RS232, RS423 and RS422

Standards in Serial Input/Output

- printer, modems can be connected to computers using serial I/O technique.
- A single manufacturing company is not enough to manufacture all the electronic devices.
- therefore, for the compatibility some kinds of common understanding must be established.
- the understanding is protocol defined by some professional bodies such as IEEE or Electronic Industries Association (EIA) as dejure standard.
- while transmitting information, either current level or voltage level is used as information content.

Serial Standards RS232, RS423 and RS422

Serial RS-232

- it is serial communication interface for distant communication.
- it interfaces ✓Data Communication Equipment (DCE) and ✓Data Terminal Equipment (DTE).
- Serial data exchange takes place between DTE and DCE.
- Computer and other devices generating data for exchange, both receiving and transmitting device, are Data Terminal Equipment
- Modem and other devices which do not generate data rather are used to send serial data are Data Communication Equipment.
- RS-232 interface is standardized by Electronic Industries Associations (EIA) according to handshaking needs.

Serial Standards RS232, RS423 and RS422

Standardized Features: Serial RS-232

- it is either 25-pins or 9-pins interface/port
- It describes the standards such as voltage level, rise and fall time, impedance level, maximum bit rate and capacitance for all signal lines.
  - ✓ DTE $\longrightarrow$ defined to be male connector,
  - ✓ DCE $\longrightarrow$ defined to be female connector.
- it can send 1.492 Kbps (20KBd) for distance of 50ft.
- voltage level and binary logic representation:
  - ✓ logic '1' $\rightarrow$ -3V to -15V.
  - ✓ logic '0' $\rightarrow$ +3V to +15V.

  Note: Normally, $\pm$12V level are used.
- MC1488 line driver converts logic '1' to -9V; logic '0' to +9V.
- MC1489 line receiver converts RS-232 to TTL logic.

Serial Standards RS232, RS423 and RS422

Serial RS-232

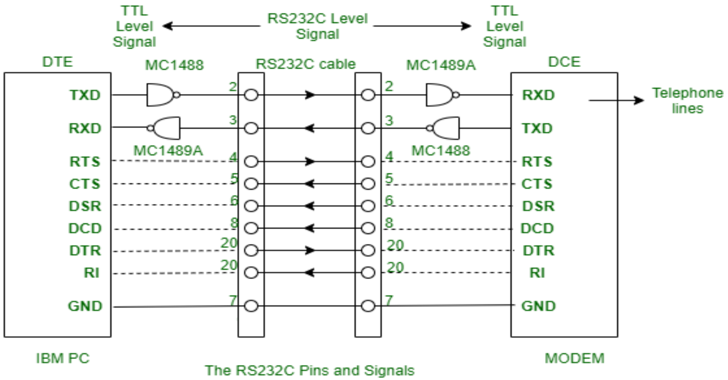
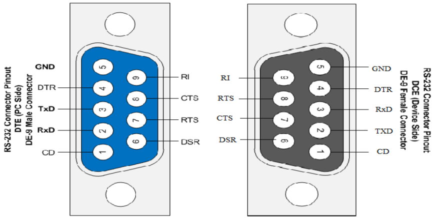
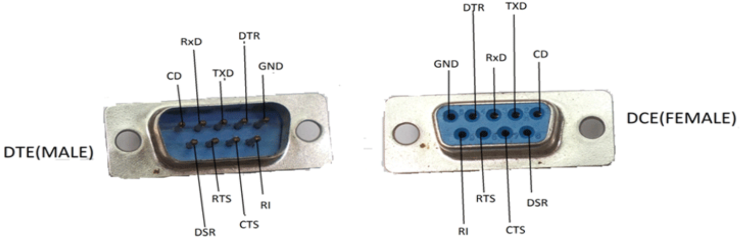


Fig. 9 Connection of DTE and DCE through RS-232 Interface

Serial Standards RS232, RS423 and RS422

Serial RS-232





Connection of DTE and DCE through RS-232 Interface

Serial Standards RS232, RS423 and RS422

Serial RS-232 Signals and used in handshaking

Signals	**DB-9P**	**DB-25P**	**Signal Flow**	**Description**
TxD	3	2	DTE to DCE	Transmitted Data
RxD	2	3	DCE to DTE	Received Data
$\overline{RTS}$	7	4	DTE to DCE	Request to Send
$\overline{CTS}$	8	5	DCE to DTE	Clear to Send
$\overline{DSR}$	6	6	DCE to DTE	Data Set Ready
GND	5	7	Common Ref	Signal Ground
$\overline{DCD}$	1	8	DCE to DTE	Data Carrier Detect
$\overline{DTR}$	4	20	DTE to DCE	Data Terminal Ready
RI	9	22	DCE to DTE	Ring Indicator
DSRD	-	23	DCE to DTE	Data Signal Rate Detector

Serial RS-232 Signals and used in handshaking

Data Terminal Ready (DTR)

- when terminal power is turned on and once it runs self checks, it send $\overline{DTR}$ signal to DCE to tell it is ready.

Data Set Ready (DSR)

- when DCE is powered on and ready to transmit or receive data, it asserts $\overline{DSR}$ signal to terminal.

Request to Send (RTS)

- when DTE is ready to send a character it will assert $\overline{RTS}$ signal to the modem (DCE).

Serial RS-232 Signals and used in handshaking

Data Carrier Detect (DCD)

- Modem (DCE) will assert $\overline{DCD}$ signal to the terminal to indicate that it has established connection with computer.

Clear to Send (CTS)

- when modem/DCE is fully ready to exchange data, it asserts $\overline{CTS}$ to terminals.

Data Signal Rate Detect (DSRD)

- it i used for switching different baud rate.

Serial RS-232 Signals and used in handshaking

Ring Indicator (RI)

- Deactivating DTR or DSR breaks the connection but RI works independent of DTR.
- when it changes its state, hardware interrupt is generated; modem may activate RI signal even if DTR is not active.

Transmitted Data (TxD)

- the DTE sends serial data characters to the modem.

Received Data (RxD)

- modem or DCE will receive data through this line.

Serial RS-232 Signals and used in handshaking

Digital Transmission using Modem and Standard Phone Line

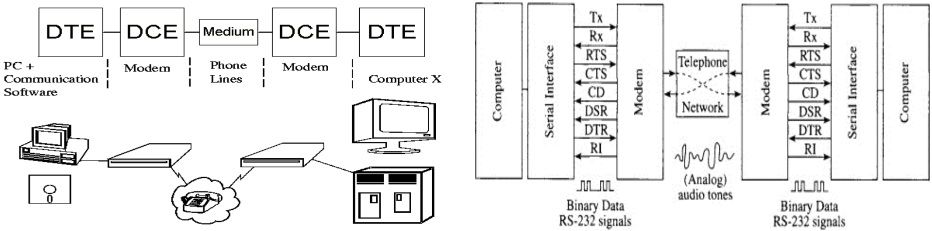


Fig. 10 Digital Data Transmission using MODEM and Telephone Line

- standard telephone system can be used for serial data over long distance.
- high speed digital data is modulated to voice signal band being telephone lines are of voice-band (300Hz to 4300Hz).
- the modulation is carried at the modem.

Serial RS-232 Signals and used in handshaking

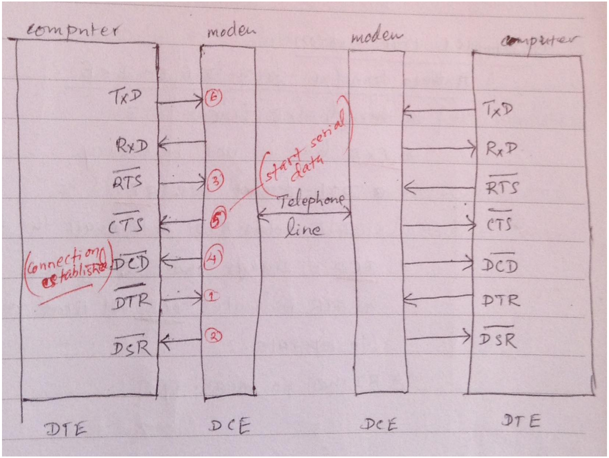
Digital Transmission using Modem and Standard Phone Line

- ✓ ① DTE asserts $\overline{DTR}$ to tell the modem/DCE that it is ready.
- ✓ ② DCE asserts $\overline{DSR}$ to the terminal and dials up.
- ✓ ③ DTE asserts $\overline{RTS}$ to DCE signifying ready to send.
- ✓ ④ DCE then asserts $\overline{DCD}$ to indicate connection established.
- ✓ ⑤ Once $\overline{CTS}$ from DCE detected, DTE starts serial data transfer.
- ✓ ⑥ when data transfer completed, DTE asserts $\overline{RTS}$ high and DCE de-asserts $\overline{CTS}$ to stop transmission.

✎ Communication between two computers through RS-232 port without modem can be done in null modem configuration.

Serial RS-232 Signals and used in handshaking

Digital Transmission using Modem and Standard Phone Line



Digital Data Transfer using Modem and standard phone line

Serial RS-232 Signals and used in handshaking

Simplex Transmission: | Serial RS-232

- Data transfer from DTE to DCE
  - ✓ Data transfer takes through TxD line, RxD is unused at this time.
  - ✓ DCE does not use RTS that is DTE holds RTS active always.
  - ✓ DCD is always inactive by DCE as DTR indicates DCE a time ready to operate or not.
  - ✓ Ring Indicator (RI) has no meaning.
- Data transfer from DCE to DTE
  - ✓ data transfer takes through RxT line, TxD is unused here.
  - ✓ DCE does not use RTS or CTS signal and active all time.
  - ✓ DCE puts DCD active as it may detect carrier signal from external device.
  - ✓ active DTR indicates DTE ready to operate.
  - ✓ Ring Indicator (RI) signifies the call to DTE from external device via DCE.

Serial RS-232 Signals and used in handshaking

Half-Duplex Transmission: | Serial RS-232

- Either DTE or DCE can operate as receiver or transmitter at a time (strictly in ordered manner).
- Only one data line is available depending upon data direction
  - ✓ TxD $\rightarrow$ DTE to DCE;
  - ✓ RxD $\rightarrow$ DCE to DTE
- RTS and CTS are used for handshaking
- Data transfer from DTE to DCE
  - ✓ DTE activates RTS signal and wait for acknowledgment signal (CTS Signal)
  - ✓ now the data transfer takes place.

Serial RS-232 Signals and used in handshaking

Full-Duplex Transmission: | Serial RS-232

- RTS and CTS signals have no meaning being always active for bidirectional transfer.
- DSR signal is enabled in most modem (in some modem, DSR may be active when preparation for destination call is completed).
- when external device has call to DTE that is when carrier is detected, DSR is active by DCE.
- RI indicates the connection request from external device to DTE via DCE (Modem).

Serial RS-232 Signals and used in handshaking

RS-232 in Null Modem Connection | Serial RS-232

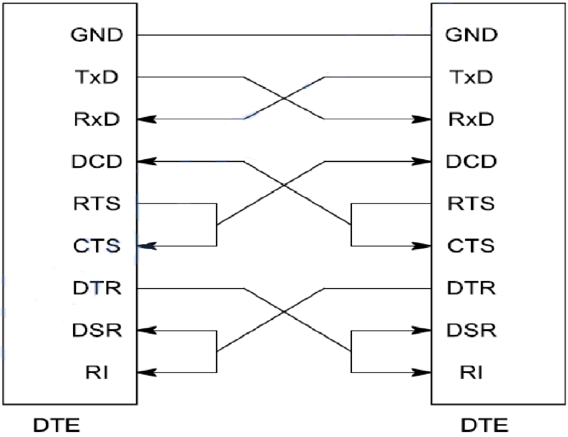
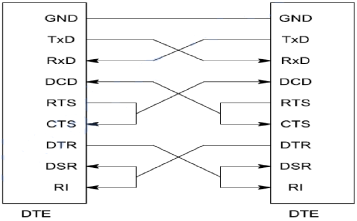


Fig. 11 Null modem Connection for RS-232 Terminals

Serial RS-232 Signals and used in handshaking

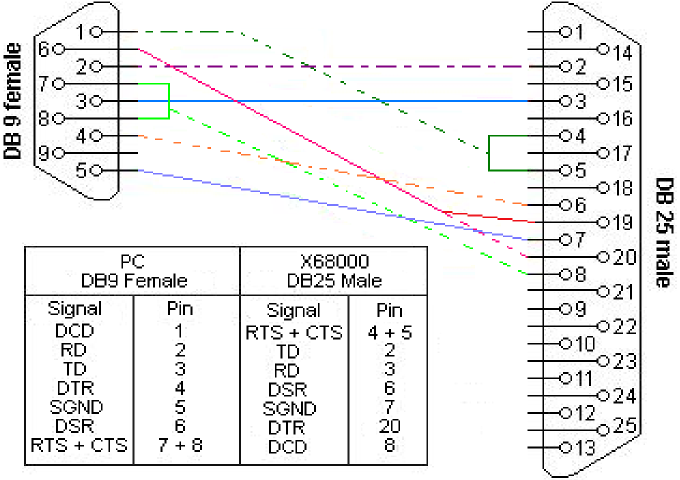
RS-232 in Null Modem Connection | Serial RS-232



Null modem Connection for RS-232 Terminals

- Since two DTEs are connecting directly, similar pins meet the same, that is: TxD $\rightarrow$ TxD; RxD $\rightarrow$ RxD so on.
- So, pins are to be crossed; for instance, RxD and TxD of both DTE is to be crossed.
- RTS can be used to activate ✓ CTS of same DTE and activation of ✓ DCD signal to the next DTE.
- Activation of DTR of one can be used to activate ✓ DSR and ✓ RI of other DTE as shown in Fig. 11

Serial RS-232 Signals and used in handshaking



PC DB9 Female		X68000 DB25 Male	
Signal	Pin	Signal	Pin
DCD	1	RTS + CTS	4 + 5
RD	2	TD	2
TD	3	RD	3
DTR	4	DSR	6
SGND	5	SGND	7
DSR	6	DTR	20
RTS + CTS	7 + 8	DCD	8

Fig. 12 Null modem Connection to printer using RS-232 Terminals

Serial RS-232 Signals and used in handshaking

RS-232 in Null Modem Connecting Printer | Serial RS-232

- Printer is not DCE but DTE, so TxD of PC will be connected to RxD of printer.
- RTS and CTS are interconnected to each other enabling immediate transfer; DCD and RI at PC has no meaning.
- Similarly, to send data to printing module, RTS and CTS of printer is interconnected. and DTR to DSR and DCD of the printer (Pins 8, 20, and 6)
- in serial interface, overrun error may occur but can be resolved using parallel interface.
- in parallel interface, busy signal can be used to indicate printer can not accept data temporarily.
- Pin-19 of printer works as buffer full signal, so for input buffer full, pin-19 is disabled to stop transfer temporarily.

Serial RS-232 Signals and used in handshaking

RS-423A Interface:

- ✓ RS-232 $\rightarrow$ efficient only for 50ft (20kbd)
- ✓ drastic reduction for longer transmission line due to open signal line (no return path) with common signal ground.
- ✓ the next solution is RS-423A.

Feature of RS-423A Interface:

- ✓ low impedance/coaxial cable(50Ω) signal ended signal line reducing signal reflection. (What is differential signaling?)
- ✓ logic: $0 \longrightarrow +4$V to $+6$V
- ✓ logic: $1 \longrightarrow$ -4V to -6V
- ✓ max data rate of 100 kbd (100x1000 baud) over 40ft and 1kbd over 4000ft.

Serial RS-232 Signals and used in handshaking

RS-423A Interface

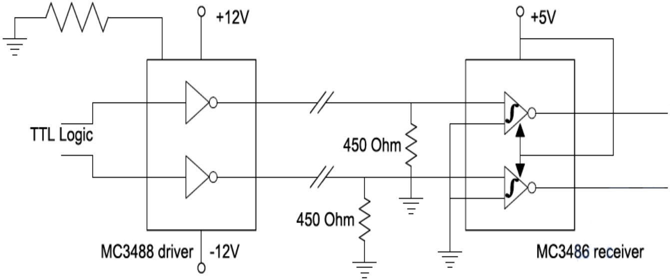


Fig. 13 MC3488 driver and MC3486 receiver used for RS-423A interface

Note: Resistor at receiving terminal (450 Ω) is used for impedance matching(?) reduces the signal reflection.

Serial RS-232 Signals and used in handshaking

RS-422A Interface:

- ✓ differential signal over twisted cable with differential amplifier.
- ✓ any electrical noise induced one signal line will be equally induced to other signal line.
- ✓ Receiver MC3486 responds to only voltage difference between signal lines.

Feature of RS-422A Interface:

- ✓ logic high $\rightarrow$ line ‘b’ more positive than ‘a’.
- ✓ logic low $\rightarrow$ reverse the above (‘a’ more positive than to ‘b’)
- ✓ voltage difference greater than 0.4V but less than 12V
- ✓ Tx rate is 10Mbd for 40ft and 100kbd for 4000ft.
- ✓ Tx line is fully terminated resulting no reflection, so high speed.

Serial RS-232 Signals and used in handshaking

RS-422A Interface

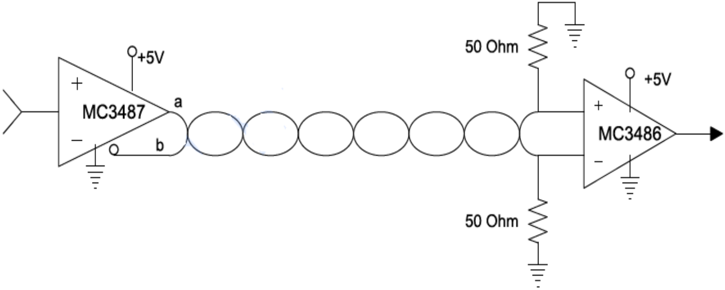


Fig. 14 MC3488 driver and MC3486 receiver used for RS-422A interface

Serial Standards RS232, RS423 and RS422

Comparison of Serial Input/Output standards:

Specification	RS-232C	RS-423A	RS422A
Speed	20 kbd	✓100 kbd at 40ft ✓1kbd at 4000ft	✓10 Mbd at 40ft ✓100kbd at 4000ft
Distance	50ft	4000ft	4000ft
logic 0	+3V to +25V	+4V to +6V	'b' line > 'a' line
logic 1	-3V to -25V	-4V to -6V	'a' line > 'b' line
Receiver input Voltage	±15V	±12V	±7V
Mode of Operation	single ended input output	differential input single output	differential input output
noise immunity	2.0V	3.4V	1.8V
Input Impedance	3 - 7 KΩ	> 4KΩ (good?)	>4KΩ
Short Circuit Current	500mA	150 mA	150mA

Universal Serial Bus (USB)

- connecting peripherals with different serial and parallel ports has problem of no auto configuration and no hot-plug ability.
- USB provides hot-plug ability and auto configuration with expandable, fast, bi-directional and low cost serial interface.
- Single connector type (interfacing logic), USB, provides wide ranges peripherals such as keyboards, mice, printers, cameras etc without direct usage of system resources.
- it is industrial standard developed in mid 1990s which defines cable standards with protocols used for connection, communication and power supply between computer and devices connected.
- USB really replaced many earlier interfaces such as serial and parallel interfaces along with separate power supply.

Universal Serial Bus (USB)

Features of USB:

1. Single Connector Type:

   Almost all legacy connectors are replaced with well defined standardized USB connectors bringing different devices with single type connector.

   This simplifies the design for different connecting devices.

2. Hot-Swappable:

   USB has hot-plug ability that means simply can be plugged or unplugged while running computer without reboot.

3. Plug and Play:

   Operating System software automatically identifies, configures and loads the appropriate device driver when USB device is plugged in.

Universal Serial Bus (USB)

Features of USB:

4. High Performance:

   USB offers low speed of 1.5 Mbps (USB 1.0), full speed 12Mbps(USB 1.0), and

   High speed 480Mbps (USB-2.0) data transfer rate;

   while USB-3.0 offer throughput of 5.0 Gbps.

5. Expandability:

   up to 127 different peripherals may be connected through a single bus at a time, theoretically.

6. Power Supply from the bus:

   USB distributes the power to all connected devices (low power devices) eliminating external power source.

Universal Serial Bus (USB)

Features of USB:

7. Easy to use for end users:

   Single interface standard simplifies to figure out the connection sockets for users;
   The operating system automatically recognizes the device attached via USB interface loading appropriate driver.

8. Low cost implementation:

   Most of the complexity of the USB protocol is handled by host making design simple and low cost.

9. Robustness:

   Error handling/fault recovery mechanism is built into the protocol;
   dynamic insertion and removal of devices is identified in user perceived real time;supports fault device identification.

Universal Serial Bus (USB)

Features of USB:

10. Wide range of workloads and applications:

    suitable for device with ranges of few kbps to several Mbps.

    supports isochronous as well as synchronous transfer type over the same lines (what is isochronous transfer?)

    concurrent operations of many devices;

    multi-connection possibility with multiple message streaming between host and devices.

    lower protocol overhead.

11. Isochronous bandwidth: guaranteed bandwidth and low latency suitable for audio band;

    isochronous workload may use entire bus bandwidth.

Universal Serial Bus (USB)

USB 1.0 | USB Standards

- USB 1.0 was released in Jan 15, 1996.
- supports data rate of 1.5Mbps (low-bandwidth/low-speed) and 12Mbps (Full-bandwidth/full-speed).
- USB 1.1 was released in Sept 23, 1998 (backward – USB 1.0).
- improvement specification allows wider usage.
- problem in USB 1.0 related to extension was fixed.

USB 2.0 | USB Standards

- USB 2.0 specification was released in April 2000, ratified by USB implementation Forum (USB-IF) at the end of 2001.
- Major improvement $\rightarrow$ supports speed up to 480Mbps.
- USB 2.0 supports high-speed (480Mbps), full-speed (12Mbps), and low-speed (1.5Mbps) with one host per bus at a time.

Universal Serial Bus (USB)

USB 3.0 | USB Standards

- USB 3.0 specification was released in Nov 12, 2008.
- brought significant performance enhancement with backward compatibility (to USB 2.0 host/device).
- supporting speed some 5Gbps, now the super-speed USB could be the next evolution.
- major improvements targeted was to enhance data speed, decreased power consumption, and backward compatibility.
- First USB 3.0 equipped device was introduced in Jan 2010.
- File size of 25GB could be transferred within 70 second.
- backward compatibility, that is, USB 2.0 device will work with USB 3.0 host; and USB 3.0 device work with USB 2.0 host.
- no device polling, lower active and idle power requirements.

Universal Serial Bus (USB)

USB Interconnect

- connection model is as shown in Fig. 15
- multiple USB devices are attached using special USB class known as hub.
- connection point provided by hub is known as port.
- host with embedded hub are called root hub.
- USB devices are connected to host logically as if it is directly connected to host.
- thus the topology is tiered as star topology.

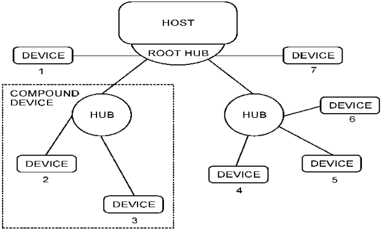


Fig. 15 USB physical Topology

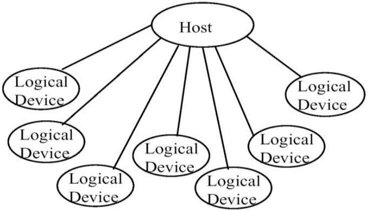


Fig. 16 USB physical Topology

USB Bus, Signal Throughput, Protocols

- ✓ ground should be connected to ground of host.
- ✓ D+ and D- are differential data pairs with 15KΩ pull down resister; used for initial plug-in detection too.
- ✓ Vcc for power supply if any from Host.

Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

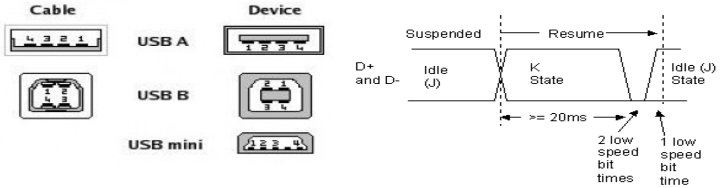


Fig. 17 USB Electrical Signalings

USB Bus, Signal Throughput, Protocols

Bus State	**Signal Levels**
Differential '1'	D+ high, D- low
Differential '0'	D+ low, D- high
Signal Ended Zero (SEO)	D+ and D- low
Single Ended One (SE1)	D+ and D- high
Data J State: Low-speed Full-speed	 Differential '0' /(D+ low, D- high) Differential '1' /(D+ high, D- low)
Data K State: Low-speed Full-speed	 Differential '1' /(D+ high, D- low) Differential '0' /(D+ low, D- high)
Idle State: Low-speed Full-speed	 Differential '0' /(D+ low, D- high) Differential '1' /(D+ high, D- low)
Resume State	Data K State
Start of Packet (SOP)	Data lines switch from idle to K state
End of Packet (EOP)	SEO for 2 bit times followed by J state for 1 bit time
disconnect	SEO for $\geq 2\mu s$
Connect	idle for $2.5\mu s$
Reset	SEO for $\geq 2.5\mu s$

USB Bus, Signal Throughput, Protocols

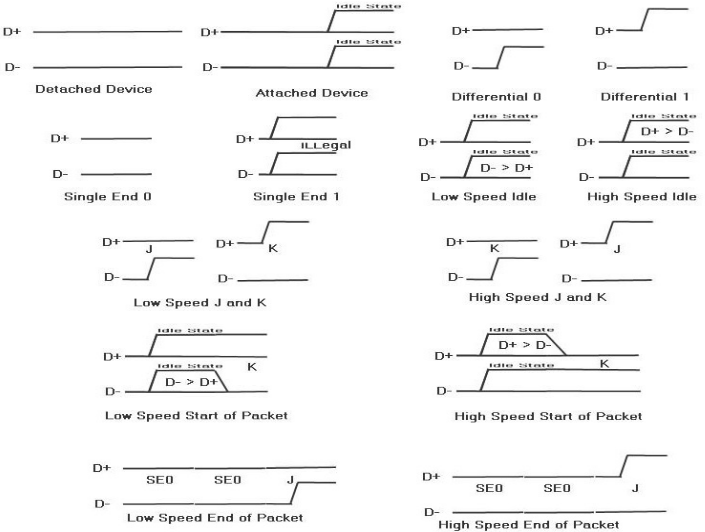


Fig. 18 USB BUS States

USB Bus, Signal Throughput, Protocols

J, K and SEO state:

- ✓ J State has same polarity as idle state (line with pull-up resister is high, and other line low)

  it is driven to that state by either host or USB device.
- ✓ K State is just the opposite polarity to the J State.
- ✓ Single Ended Zero (SEO) both lines are being pulled low.

  J and K states are used for full speed and low speed links and are opposite polarity.

Single Ended One (SE1):

- it is illegal condition where both lines are high which is malfunctioning condition or link.

USB Bus, Signal Throughput, Protocols

Reset:

- when host wants to start communication, it will start applying reset condition which sets the USB device to its default unconfigured state.
- it involves pulling down both the lines to low level Single Ended Zero(SEO) at least for 10ms.
- the device may recognize the reset condition after 2.5μs
- Reset means switch the device to known state.

EOP Signal:

- End of Packet (EOP) is an SEO state for 2 bit time, followed by a J state for 1 bit time.

USB Bus, Signal Throughput, Protocols

Idle/Suspend State:

- USB suspend means power down when device is not used.
- Suspending USB is achieved by not sending anything to the device for 3ms.
- generally, start of packet (SOP) for full speed, or keep alive for low speed signal will be sent by host every 1ms keeping the device awake.
- at suspend, USB draw no more than 0.5mA from Vbus.
- suspended device must recognize the reset signal and resume.

USB Bus, Signal Throughput, Protocols

Resume:

- to resume suspend/idle state, host reverses the polarity of data line (K state) at least for 20ms.
- signaling is completed with low speed end of packet signal.
- for remote resume, with its wake up feature, device must have been for idle state at least for 5ms and apply wakeup K condition for 1 to 15 ms;
- host then takeover the resume signal within 1ms.

Keep Alive Signal:

- this is low speed End of the Packet (EOP).
- it is sent at least once every millisecond on a low speed link to keep the device awake.

USB Bus, Signal Throughput, Protocols

Throughput: | USB

- throughput is the actual output.
- USB throughput is determined with followings:
  - ✓ device ability to sink or source the data.
  - ✓ bandwidth consumption by other devices in the bus.
  - ✓ efficiency of host's USB ports.
  - ✓ type of data to transfer.

USB Bus, Signal Throughput, Protocols

Speed: | USB

- Speed is indicated by pulling either D+ or D- line to high 3.3V
- Full speed device will use a pull up resister attached to D+ to specify full speed device.
- Low speed device will use a pull up resister to D- indicating low speed device.
- these pull up resistors are used by host or hub to detect other devices connected.
- without pull up resistor, USB assumes no device connected to the bus.
- High-speed devices start as full-speed devices and then transition to high-speed operation through a speed negotiation process called "chirping."
- Initially, they use a pull-up on the D+ line like a full-speed device.

USB Bus, Signal Throughput, Protocols

Speed: | USB

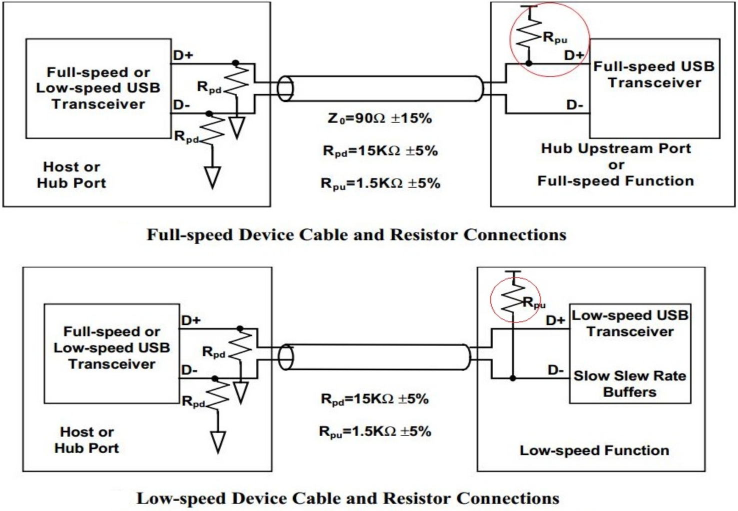


Fig. 19 USB device with pull resistors

USB Bus, Signal Throughput, Protocols

Protocol: | USB

- USB is made up of several layers of protocol stacks
- in most cases, USB controller ICs will take care of lower layers
- Each USB transaction consists:

  (a) Token Packet (b) Optional Data Packet (c) Status Packet.

- USB is host centric bus, host initiates all transactions.
- The First Packet – token packet – is generated by host to describe:
  - ✓ What is to follow, whether data transaction will be read/write;
  - ✓ What the device address and designated point is.
- Next Packet is generally data packet carrying payload and is followed by handshaking packet;
- handshaking/status packet reports if the data or token was received successfully or if the endpoint is stalled or not available to accept data.

USB Bus, Signal Throughput, Protocols

Common USB Packet Fields: | USB Protocol

- Data over the USB bus is transmitted LSB first.
- following fields are presented in USB packet:

1. **Sync:** | USB Packet Field
   - all packets must start with a sync field
   - it is 8bit long at low and full speed; and 32bit for high speed.
   - it is used to synchronize the clock receiver with transmitter.
   - the last two bit indicates where to start PID field.
2. **PID:** | USB Packet Field
   - PID stands for Packet ID – it is used to identify the type of packet that is being sent.
   - there are 4bit PID, but to ensure successful receive, 4bit more complemented making 8bit PID.

PID_0	PID_1	PID_2	PID_3	$nPID_0$	$nPID_1$	$nPID_2$	$nPID_3$

USB Bus, Signal Throughput, Protocols

Common USB Packet Fields

3. ADDR: | USB Packet Field
   - the address field specifies device to which packet is designated.
   - the address bit has 7 bit with possible 127 USB devices.
   - Fist address 0 is invalid which is not assigned to any device.
4. ENDP: | USB Packet Field
   - it is made up of 4bits allowing 16 possible endpoints.
   - low speed device, however, can only have two additional endpoints on the top of the default pipe.
5. CRC: | USB Packet Field
   - token packet has 5 bit CRC while 16bits for data packet.
6. EOP: | USB Packet Field
   - End of Packet (EOP) is signaled by Single Ended Zero(SEO) for approximately 2 bit time followed by a J for 1bit time.

USB Bus, Signal Throughput, Protocols

USB Packet Types

- USB packets are categorized into four different types:

Token Packet	indicates type of transactions to follow
Data Packet	Contains payload/actual data
Handshaking Packet	data acknowledgment or error reporting
Start of frame	indicates start of a new frame

Token Packets:

- Token packet can be further categorized as:

✓ In - informs USB devices; host wants to read information.

✓ Out - informs USB devices; host wants to send information.

✓ Setup - used to begin control transfer.

- Token Packet field format:

Sync	PID	ADDR	ENDP	CRC5	EOP

USB Bus, Signal Throughput, Protocols

USB Packet Types...

Data Packet

- There are two types of data packets Data0 and Data1, each capable of transmitting upto 1024 bytes of data.

Sync	PID	Data	CRC16	EOP

✓ max data payload for low-speed devices $\rightarrow$ 8 bytes

✓ max data payload for full-speed devices $\rightarrow$ 1023 bytes

✓ max data payload for high-speed devices $\rightarrow$ 1024 bytes

✓ data must be sent in multiples of bytes.

USB Bus, Signal Throughput, Protocols

USB Packet Types...

Status/Handshake Packets:

There are three types of handshake packets ACK and NAK, and STALL with simply of the PID (complemented 4bits).

- ✓ ACK → acknowledge the packet successfully received.
- ✓ NAK → reports the devices temporarily cannot send or receive data; also used during interrupt transactions to inform the host there is no data to send.
- ✓ STALL → the device finds its in a state requiring intervention from host.

Sync	PID	EOP

USB Bus, Signal Throughput, Protocols

USB Packet Types...

Start of Frame Packets:

- It consists of 11bits frame number – sent by host for every 1ms $\pm$ 500ns for full speed bus or every 125 ± 0.0625 μs on a high speed bus.

Sync	PID	Frame Number	CRC5	EOP

USB Bus, Signal Throughput, Protocols

Transfer Model

Endpoints:

- Endpoints can be described as sources or sinks of data.
- Being host centric bus, endpoint occurs at the end of communication channel at the USB function.
- at the software layer, device driver may send a packet to device EP1, for instance.
- send data end up at EP1 out buffer.
- the USB function writes data to EPI IN buffer which sits in the buffer until the host sends a IN packet to that endpoint requesting the data.
- endpoints can be seen as interface between hardware of function device and firmware running on the function device.

USB Bus, Signal Throughput, Protocols

Transfer Model

Pipes: | Transfer Model

- while the device sends and receives data on a series of endpoints, the client software transfers data through pipe.
- Pipe is logical connection between host and endpoints.
- pipe sets parameters associated with them such as how much bandwidth is allocated to it, what transfer type (control, bulk, isochronous or Interrupt), direction of flow, maximum packet/buffer sizes.
- default pipe direction is bi-directional made up of endpoint zero in and endpoint zero out with control transfer.

USB Bus, Signal Throughput, Protocols

Stream Pipe | Transfer Model

- it has no defined USB format, ie you can send any type of data down a stream pipe and can retrieve the data out the other end.
- Stream pipe will support bulk, isochronous and interrupt transfer types.
- Stream pipe can be controlled by host or device.

Message Pipe | Transfer Model

- it has defined USB format; they are host controlled initiated by a request sent from host.
- data is transferred in the desired direction – dictated by the request from the host.
- message pipe allows data to flow in both direction but will only support control transfer.

USB Bus, Signal Throughput, Protocols

Control Transfer: | Transfer Model/Data Flow Types

- typically used for short, simple commands to the devices, and a status response.

Bulk Data Transfer:| Transfer Model/Data Flow Types

- large sporadic transfers using all remaining available bandwidth(but no guarantees on bandwidth and latency)
- device like printer uses bulk data transfer data flow.

USB Bus, Signal Throughput, Protocols

Interrupt Data Transfer: | Transfer Model/Data Flow Types

- devices that needs guaranteed quick responses(bounded latency) – sending very little data.
- mouse, keyboard would choose interrupt mode data transfer.

Isochronous Data Transfer: | Transfer Model/Data Flow Types

- to some guaranteed speed(not necessarily as fast as possible) with possible data loss uses this mode of transfer.
- device like speakers in real-time without error correction.

USB Bus, Signal Throughput, Protocols
Nodes, hosts

USB bus is tiered star topology with single host serving upto 127 (theoretically) USB devices.

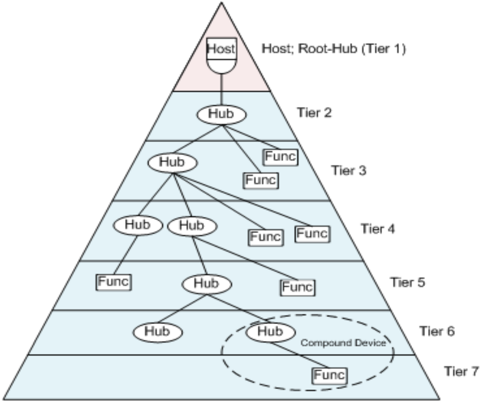


Fig. 20 USB Network Protocol Architecture

USB Bus, Signal Throughput, Protocols
Nodes, hosts

- A device can be plugged into a hub, and the hub can be further extended limited to some fixed number of tiers(six tiers).
- all devices can have upstream connecting host, and the host can have downstream to the connecting devices.
- length of the cable is limited to 5 meters stated in the USB specifications regarding cable delay, power drops.
- so near devices with USB and distant applications can be connected using ethernet.

USB Bus, Signal Throughput, Protocols
Nodes, hosts

Hub

- hub has two major roles: power management and signal distribution
- Hubs provides links to the devices; potentially unlimited USB ports to PC.
- USB should deal with Powered hubs and un-powered hubs.

Powered Hub

- needed when connecting low-power devices such as mouse, digital camera.
- these devices derive their power source from USB bus.
- too many low-power devices connecting through USB cause PC difficult to handle it.

USB Bus, Signal Throughput, Protocols
Nodes, hosts

Un-Powered Hub

- used when high-power devices such as printer, scanners are USB connected.
- these high power devices have their own power supply requiring no power from USB bus.
- safe to use low-power devices provided larger number of USB devices are not connected at the same time.

USB Bus, Signal Throughput, Protocols

USB On The Go (OTG)

✍ Please explore yourself

USB Bus, Signal Throughput, Protocols

Interface Chip: USB Device and USB host

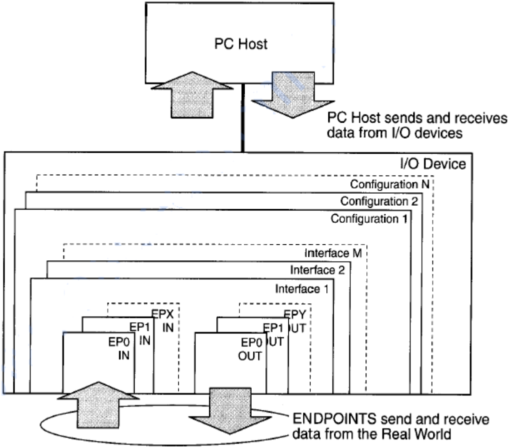


Fig. 21 Logical view of devices host interface

USB Bus, Signal Throughput, Protocols

Interface Chip: USB Device and USB host

- Endpoint is where the data enters or leaves the USB system.
- an IN endpoint is data creator while out is data consumers.
- the connections of endpoint is called interfaces, and directly related to the real-world connections.
- An operating system (OS) will have driver for each interface.
- some devices may have multiple interfaces such as telephone with keypad and audio interfaces (printer: print, scan and fax).
- OS correspondingly manage two separate device drivers.
- A collection of interfaces is configuration, and only one configuration can be active at a time.
- A configuration defines attributes and features of a model.

USB Bus, Signal Throughput, Protocols

Interface Chip: USB Device and USB host

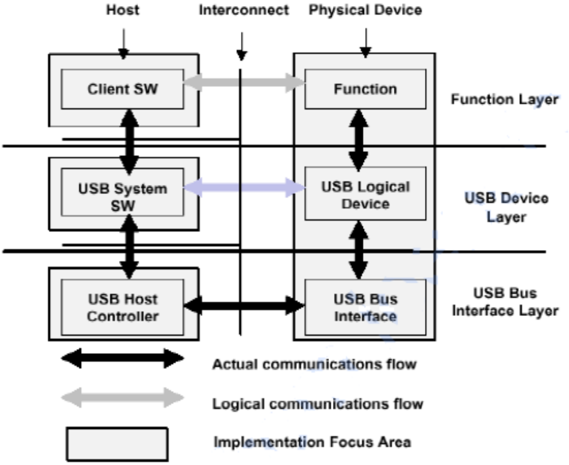


Fig. 22 Interface between device and host

As you go
Assignment

Assignment Module#3 is available at MS - Team.

Deadline for submission: 13th June 2024 (*Before 3:00 PM*)