UNIT-5

SERIAL DATA TRANSFER SCHEMES

SERIAL COMMUNICATION

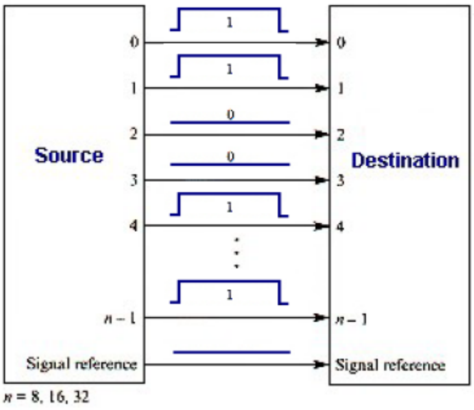
INTRODUCTION

Serial communication is common method of transmitting data between a computer and a peripheral device such as a programmable instrument or even another computer. Serial communication transmits data one bit at a time, sequentially, over a single communication line to a receiver. Serial is also a most popular communication protocol that is used by many devices for instrumentation. This method is used when data transfer rates are very low or the data must be transferred over long distances and also where the cost of cable and synchronization difficulties makes parallel communication impractical. Serial communication is popular because most computers have one or more serial ports, so no extra hardware is needed other than a cable to connect the instrument to the computer or two computers together.

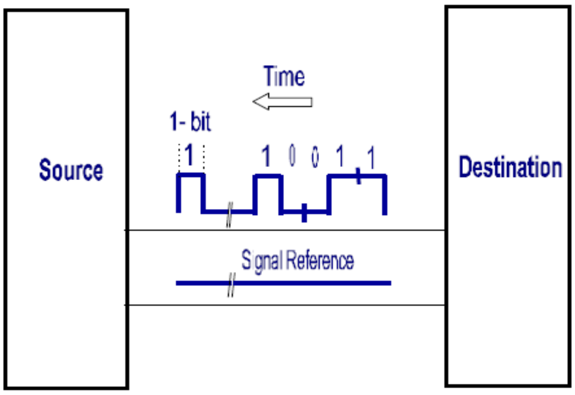
SERIAL AND PARALLEL TRANSMISSION

Let us now try to have a comparative study on parallel and serial communications to understand the differences and advantages & disadvantages of both in detail.

We know that parallel ports are typically used to connect a PC to a printer and are rarely used for other connections. A parallel port sends and receives data eight bits at a time over eight separate wires or lines. This allows data to be transferred very quickly. However, the setup looks more bulky because of the number of individual wires it must contain. But, in the case of a serial communication, as stated earlier, a serial port sends and receives data, one bit at a time over one wire. While it takes eight times as long to transfer each byte of data this way, only a few wires are required. Although this is slower than parallel communication, which allows the transmission of an entire byte at once, it is simpler and can be used over longer distances. So, at first sight it would seem that a serial link must be inferior to a parallel one, because it can transmit less data on each clock tick. However, it is often the case that, in modern technology, serial links can be clocked considerably faster than parallel links, and achieves a higher data rate.



Parallel Transmission



Serial Transmission

Even in shorter distance communications, serial computer buses are becoming more common because of a tipping point where the disadvantages of parallel busses (clock skew, interconnect density) outweigh their advantage of simplicity. The serial port on your PC is a full-duplex device meaning that it can send and receive data at the same time. In order to be able to do this, it uses separate lines for transmitting and receivingdata.

From the above discussion we could understand that serial communications have many advantages over parallel one like:

- Requires fewer interconnecting cables and hence occupies less space.
- "Cross talk" is less of an issue, because there are fewer conductors compared to that of parallel communication cables.
- Many IC s and peripheral devices have serial interfaces.
- Clock skew between different channels is not an issue.
- Cheaper to implement.

Clock skew:
Clock skew is a phenomenon in synchronous circuits in which the clock signal sent from the clock circuit arrives at different components at different times, which can be caused by many things, like:

- Wire-interconnect length
- Temperature variations
- Variation in intermediate devices
- capacitive coupling
- Material imperfections

SERIAL DATA TRANSMISSION MODES

When data is transmitted between two pieces of equipment, three communication modes of operation can be used.

Simplex: In a simple connection, data is transmitted in one direction only. For example, from a computer to printer that cannot send status signals back to the computer.

Half-duplex: In a half-duplex connection, two-way transfer of data is possible, but only in one direction at a time.

Full duplex: In a full-duplex configuration, both ends can send and receive data simultaneously, which technique is common in our PCs.

SERIAL DATA TRANSFER SCHEMS

Like any data transfer methods, Serial Communication also requires coordination between the sender and receiver. For example, when to start the transmission and when to end it, when one particular bit or byte ends and another begins, when the receiver's capacity has been exceeded, and so on. Here comes the need for synchronization between the sender and the receiver. A protocol defines the specific methods of coordinating transmission between a sender and receiver. For example a serial data signal between two PCs must have individual bits and bytes that the receiving PC can distinguish. If it doesn't, then the receiving PC can't tell where one byte ends and the next one begin or where one bit ends and begins. So the signal must be synchronized in such a way that the receiver can distinguish the bits and bytes as the transmitter intends them to be distinguished.

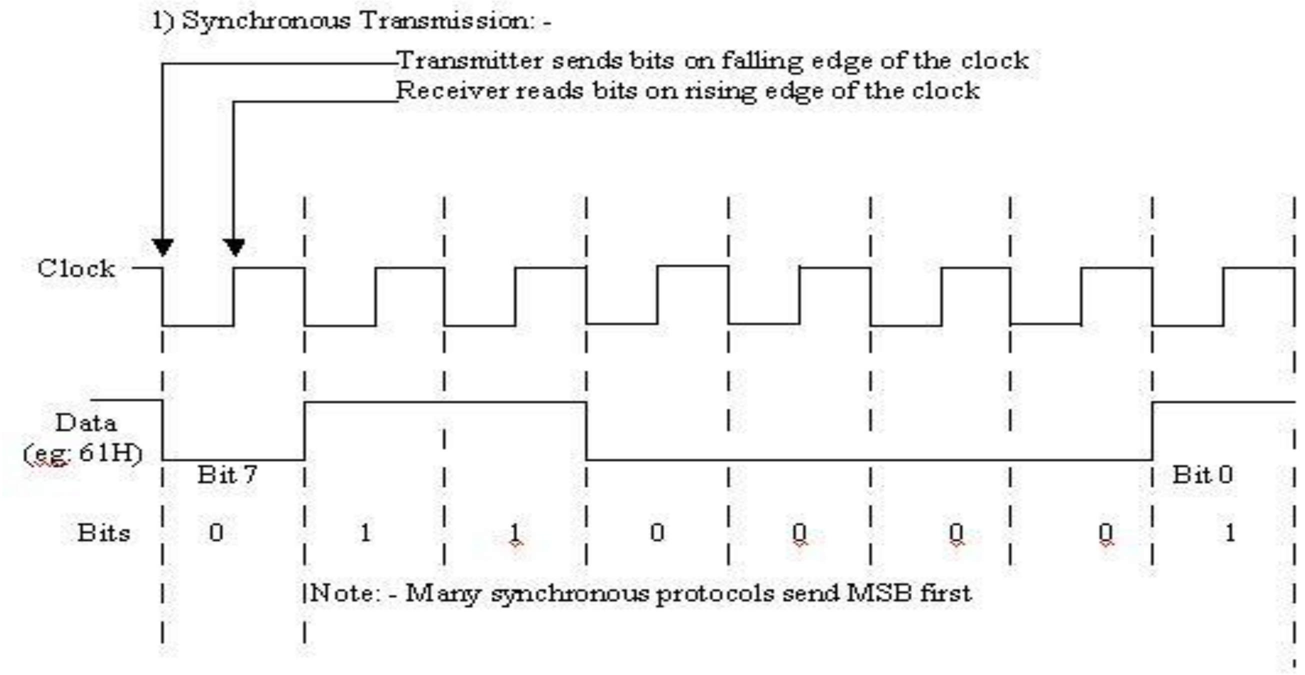
There are two ways to synchronize the two ends of the communication.

1. Synchronous data transmission
2. Asynchronous data transmission

Synchronous Data Transmission

The synchronous signaling methods use two different signals. A pulse on one signal line indicates when another bit of information is ready on the other signal line.

In synchronous transmission, the stream of data to be transferred is encoded and sent on one line, and a periodic pulse of voltage which is often called the "clock" is put on another line, that tells the receiver about the beginning and the ending of each bit.


1) Synchronous Transmission: -
Transmitter sends bits on falling edge of the clock
Receiver reads bits on rising edge of the clock
Clock
Data
(eg: 61H)
Bit 7
Bit 0
Bits
0
1
1
0
0
0
0
1
Note: - Many synchronous protocols send MSB first

Advantages: The only advantage of synchronous data transfer is the Lower overhead and thus, greater throughput, compared to asynchronous one.

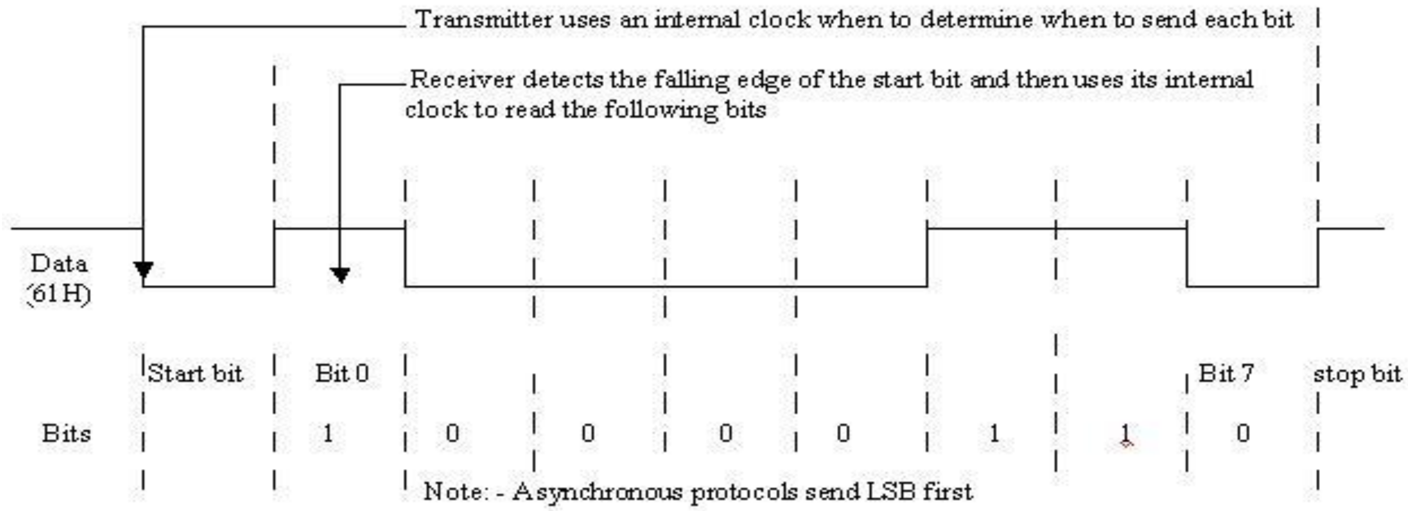
Disadvantages:

- Slightly more complex
- Hardware is more expensive

Asynchronous data transmission

The asynchronous signaling methods use only one signal. The receiver uses transitions on that signal to figure out the transmitter bit rate (known as auto baud) and timing. A pulse from the local clock indicates when another bit is ready. That means synchronous transmissions use an external clock, while asynchronous transmissions are use special signals along the transmission medium. Asynchronous communication is the commonly prevailing communication method in the personal computer industry, due to the reason that it is easier to implement and has the unique advantage that bytes can be sent whenever they are ready, a no need to wait for blocks of data to accumulate.

2) Asynchronous Transmission: -


Transmitter uses an internal clock when to determine when to send each bit
Receiver detects the falling edge of the start bit and then uses its internal clock to read the following bits
Data
(61H)
Start bit
Bit 0
Bit 7
stop bit
Bits
1
0
0
0
0
1
1
0
Note: - Asynchronous protocols send LSB first

Advantages:

- Simple and doesn't require much synchronization on both communication sides.
- The timing is not as critical as for synchronous transmission; therefore hardware can be made cheaper.
- Set-up is very fast, so well suited for applications where messages are generated at irregular intervals, for example data entry from the keyboard.

Disadvantages: One of the main disadvantages of asynchronous technique is the large relative overhead, where a high proportion of the transmitted bits are uniquely for control purposes and thus carry no useful information.

8251A-PROGRAMMABLE COMMUNICATION INTERFACE
(8251A-USART-Universal Synchronous/Asynchronous Receiver/Transmitter)

INTRODUCTION

A USART is also called a programmable communications interface (PCI). When information is to be sent by 8086 over long distances, it is economical to send it on a single line. The 8086 has to convert parallel data to serial data and then output it. Thus lot of microprocessor time is required for such a conversion.

Similarly, if 8086 receives serial data over long distances, the 8086 has to internally convert this into parallel data before processing it. Again, lot of time is required for such a conversion. The 8086 can delegate the job of conversion from serial to parallel and vice versa to the 8251A USART used in the system.

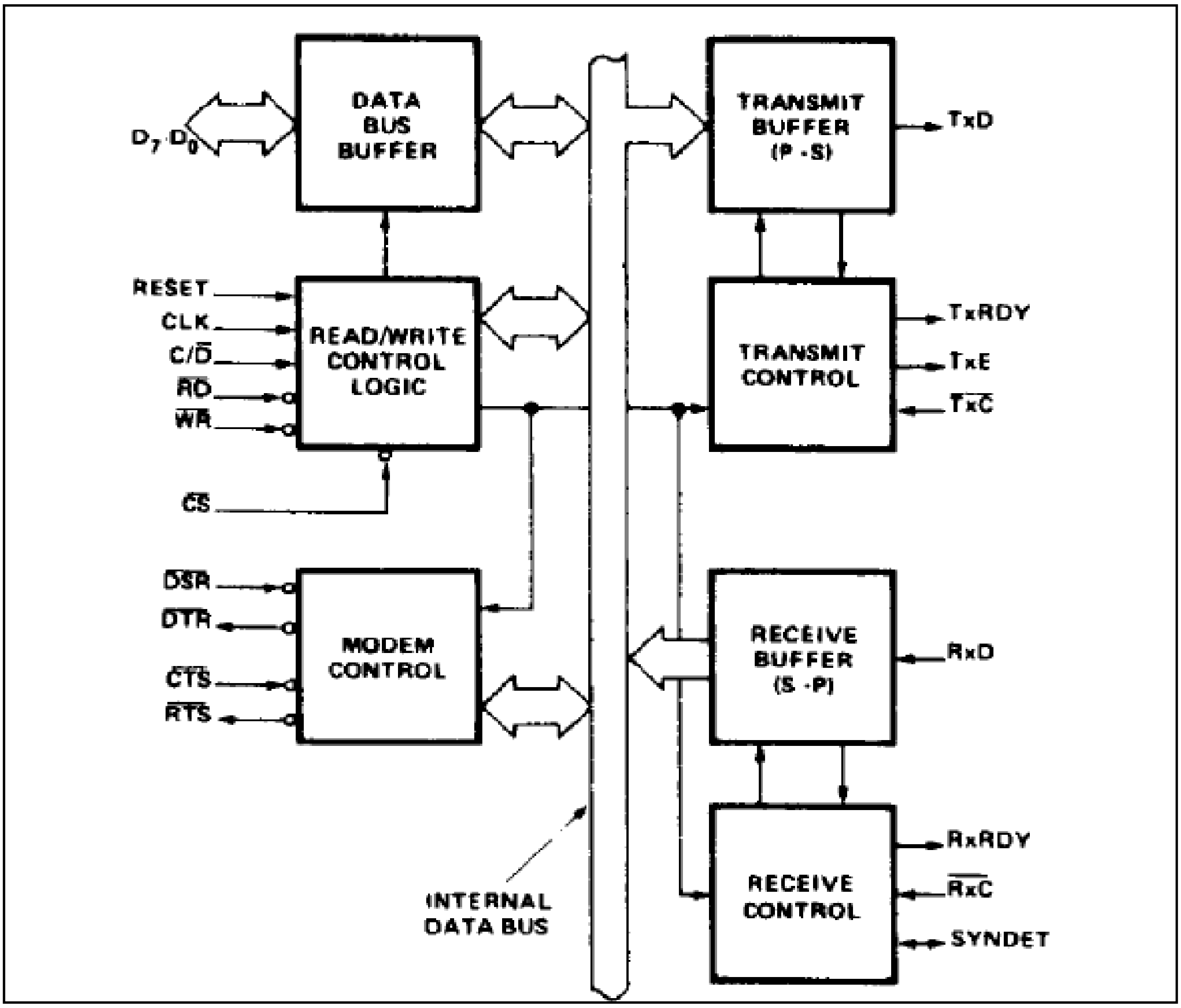
The Intel8251A is the industry standard Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel microprocessor families such as 8080, 85, 86 and 88. The 8251A converts the parallel data received from the processor on the D_{7-0} data pins into serial data, and transmits it on TxD (transmit data) output pin of 8251A. Similarly, it converts the serial data received on RxD (receive data) input into parallel data, and the processor reads it using the data pins D_{7-0}.

FEATURES

- Compatible with extended range of Intel microprocessors.
- It provides both synchronous and asynchronous data transmission.
- Synchronous 5-8 bit characters.
- Asynchronous 5-8 bit characters.
- It has full duplex, double buffered transmitter and receiver.
- Detects the errors-parity, overrun and framing errors.
- All inputs and outputs are TTL compatible.
- Available in 28-pin DIP package.

ARCHITECTURE

The 8251A is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion. The internal block diagram of 8251A is shown in fig below.



Data Bus Buffer:_ This bidirectional, 8-bit buffer used to interface the 8251A to the system data bus and also used to read or write status, command word or data from or to the 8251A.

Read/Write control logic: The Read/Write Control logic interfaces the 8251A with microprocessor, determines the functions of the 8251A according to the control word written into its control register and monitors the data flow. This section has three registers and they are control register, status register and data buffer.

- The active low signals $\overline{RD}$, $\overline{WR}$, $\overline{CS}$ and $C/\overline{D}$ are used for read/write operations with these three registers.
- When$C/\overline{D}$) is high, the control register is selected for writing control word or reading status word. When $C/\overline{D}$ is low, the data buffer is selected for read/write operation.
- When the reset is high, it forces 8251A into the idle mode.
- The clock input is necessary for 8251A for communication with microprocessor and this clock does not control either the serial transmission or the reception rate.

Transmitter section: The transmitter section accepts parallel data from microprocessor and converts them into serial data. The transmitter section is double buffered, i.e., it has a buffer register to hold an 8-bit parallel data and another register called output register to convert the parallel data into serial bits. When output register is empty, the data is transferred from buffer to output register. Now the processor can again load another data in buffer register.

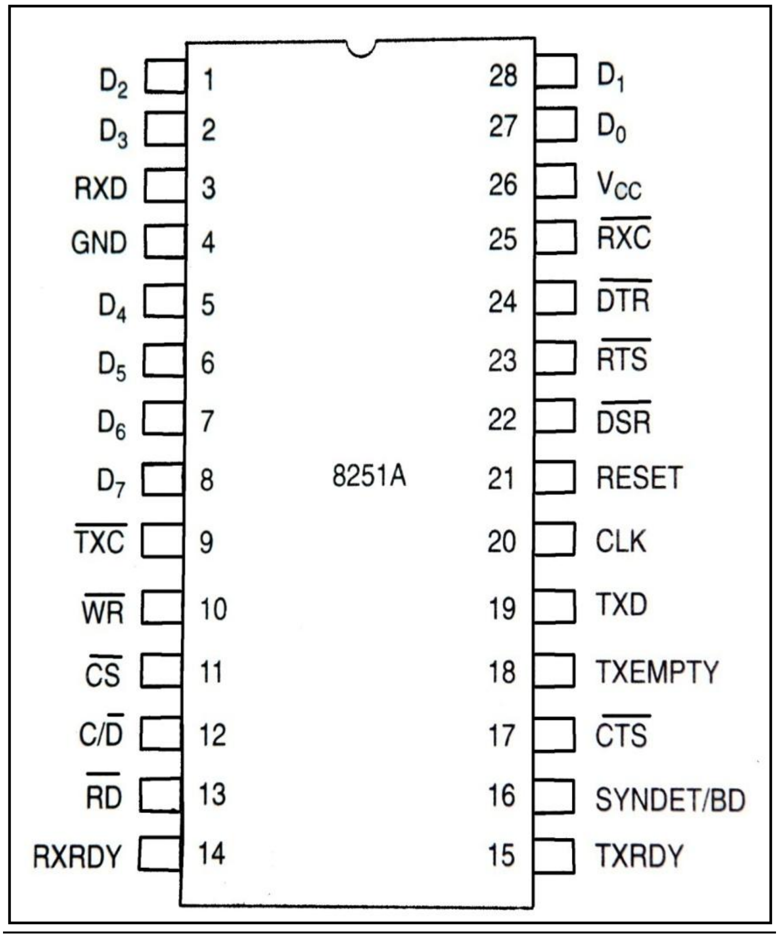
- If buffer register is empty, then TxRDY is goes to high.
- If output register is empty then TxEMPTY goes to high.
- The clock signal $\overline{TxC}$ controls the rate at which the bits are transmitted by the USART.
- The clock frequency can be 1,16 or 64 times the baud rate.

Receiver Section: The receiver section accepts serial data and convert them into parallel data. The receiver section is double buffered, i.e., it has an input register to receive serial data and convert to parallel, and a buffer register to hold the parallel data. When the RxD line goes low, the control logic assumes it as a START bit, waits for half a bit time and samples the line again. If the line is still low, then the input register accepts the following bits, forms a character and loads it into the buffer register. The microprocessor reads the parallel data from the buffer register.

- When the input register loads a parallel data to buffer register, the RxRDY line goes high.
- The clock signal $\overline{RxC}$ controls the rate at which bits are received by the USART.
- During asynchronous mode, the signal SYNDET/BRKDET will indicate the break in the data transmission. During synchronous mode, the signal SYNDET/BRKDET will indicate the reception of synchronous character.

MODEM Control: The MODEM control unit allows to interface a MODEM to 8251A and to establish data communication through MODEM over telephone lines. This unit takes care of handshake signals for MODEM interface.

PIN DIAGRAM



Vcc: +5V supply

GND: Ground

D7-D0: This is bidirectional data bus which receives control words and transmits data from the microprocessor and sends status words and received data to microprocessor.

$\overline{\boldsymbol{CS}}$**:** When signal goes low, the 8251A is selected by the MPU for communication.

$\overline{\boldsymbol{RD}}$: When signal goes low, the MPU either reads a status from the status register or accepts data from data buffer.

$\overline{\boldsymbol{WR}}$: When signal is low, the MPU either writes in the control register or sends output to the data buffer.

C/$\overline{\boldsymbol{D}}$: This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the microprocessor. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.

RESET: A high on this signal reset 8252A & forces it into the idle mode.

CLK: Clock input, usually connected to the system clock for communication with the microprocessor. $\overline{\boldsymbol{TxC}}$: This is a clock input signal which determines the transfer speed of transmitted data. In "synchronous mode," the baud rate will be the same as the frequency of TXC. In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC.

$\overline{\boldsymbol{RxC}}$: This is a clock input signal which determines the transfer speed of received data. In "synchronous mode," the baud rate is the same as the frequency of RXC. In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.

TxD- Transmitted Data Output: Output signal to transmit the data to peripherals

TxRDY: This is an output terminal which indicates that the 8251is ready to accept a transmitted data character.

TxEMPTY: This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character.

RxD- Receive Data Input: Bits are received serially on this line.

RxRDY: This is a terminal which indicates that the 8251 contains a character that is ready to READ.

SYNNDET/BD: This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset. In "external synchronous mode, "this is an input terminal. A "High" on this input forces the 8251 to start receiving data characters. In "asynchronous mode," this is an output terminal which generates "high level"output upon the detection of a "break" character if receiver data contains a "low-level" space between the stop bits of two continuous characters.

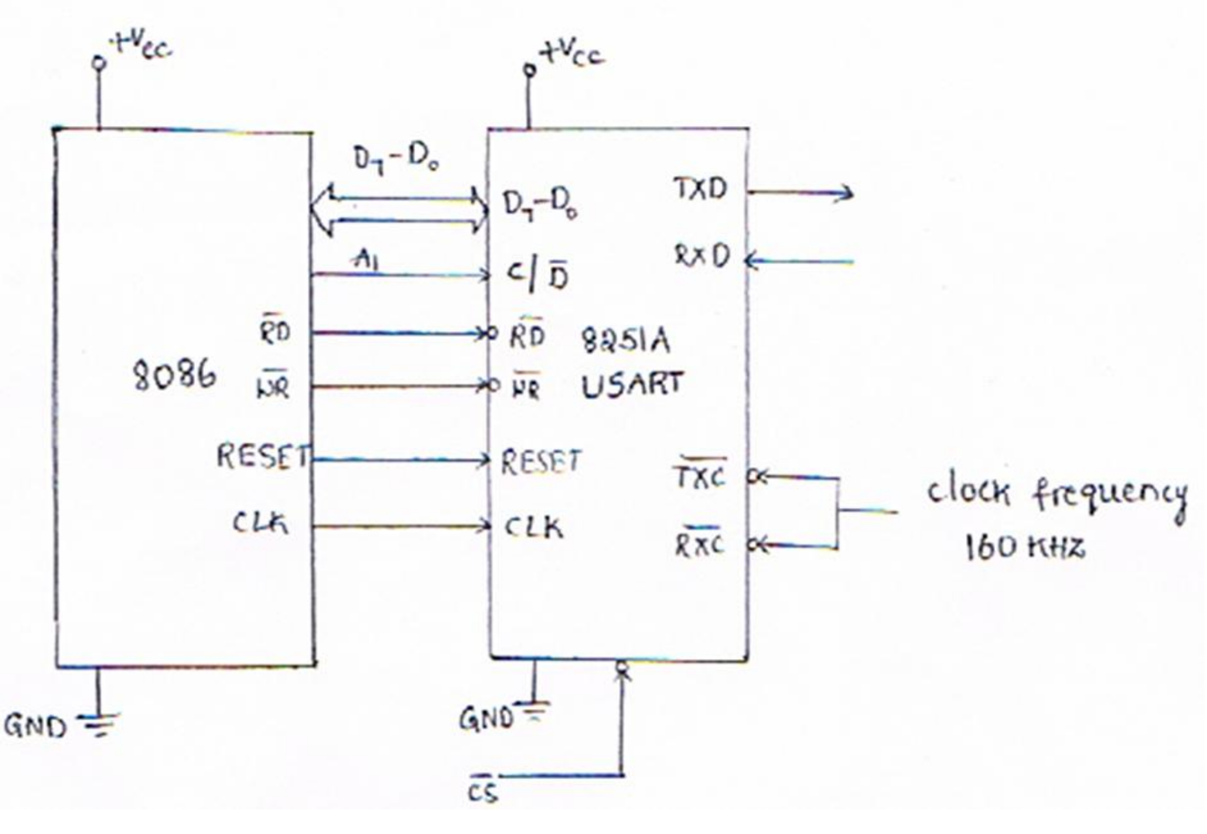
$\overline{DSR}$ - Data Set Ready: Checks if the Data Set is ready when communicating with a modem.

$\overline{DTR}$ - Data Terminal Ready: Indicates that the device is ready to accept data when the 8251 is communicating with a modem.

$\overline{CTS}$ - Clear to Send: If it’s low, the 8251A is enabled to transmit the serial data provided the enable bit in the command byte is set to ‘1’.

$\overline{RTS}$ - Request to Send Data: Low signal indicates the modem that the receiver is ready to receive a data byte from the modem.

8251A USART INTERFACING WITH 8086



PROGRAMMING THE 8251A

Prior to starting a data transmission or reception, the 8251A must be loaded with a set of control words generated by the microprocessor. These control signals define the complete functional definition of the 8251A and must immediately follow a reset operation (internal or external). The control words are split into two formats.

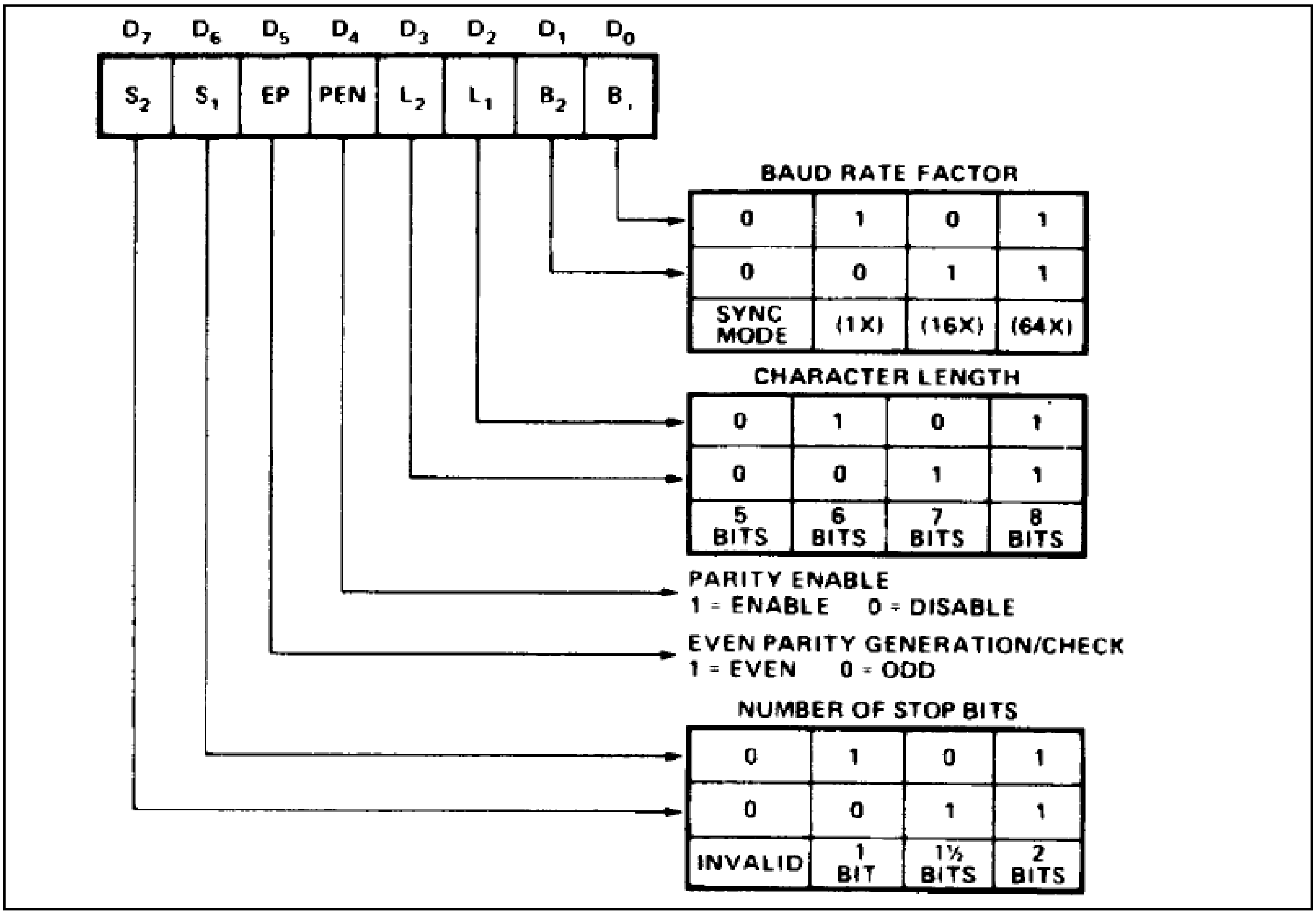
1. Mode instruction
2. Command instruction

Mode instruction: Mode instruction is used for setting the function of the 8251A. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length
- Parity bit
- Baud rate factor (asynchronous mode)
- Internal/external synchronization (synchronous mode)
- Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction format is shown in Figures below. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.


D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
S_2 S_1 EP PEN L_2 L_1 B_2 B_1

BAUD RATE FACTOR

0	1	0	1
0	0	1	1
SYNC MODE	(1X)	(16X)	(64X)

CHARACTER LENGTH

0	1	0	1
0	0	1	1
5 BITS	6 BITS	7 BITS	8 BITS

PARITY ENABLE
1 = ENABLE 0 = DISABLE

EVEN PARITY GENERATION/CHECK
1 = EVEN 0 = ODD

NUMBER OF STOP BITS

0	1	0	1
0	0	1	1
INVALID	1 BIT	1½ BITS	2 BITS

Fig. Mode instruction format, Asynchronous mode

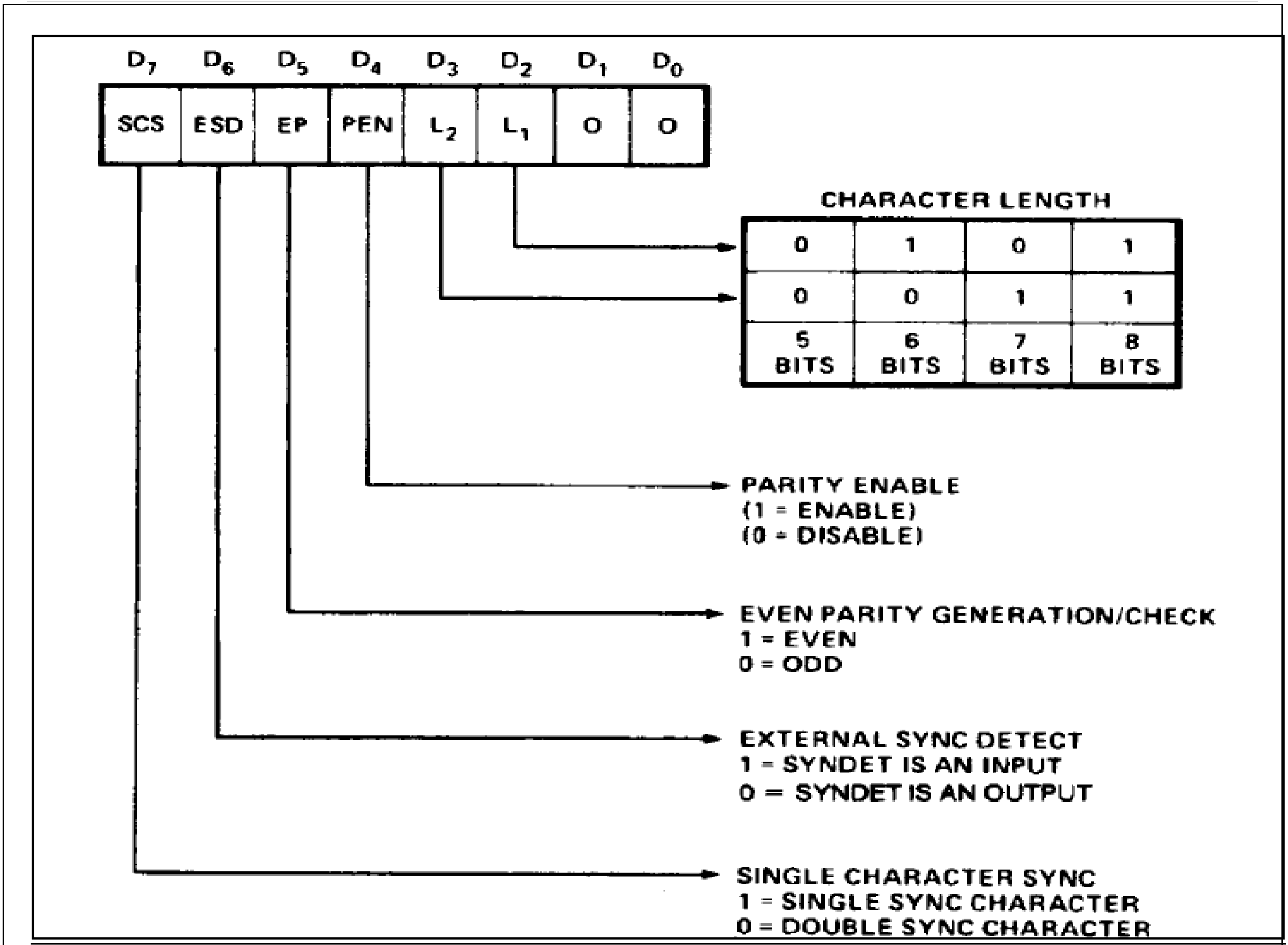


Fig. Mode instruction format, Synchronous mode

Command Instruction: Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters.

Items to be set by command are as follows:

- Transmit Enable/Disable
- Receive Enable/Disable
- DTR, RTS Output of data.
- Resetting of error flag.
- Sending to break characters
- Internal resetting
- Hunt mode (synchronous mode)

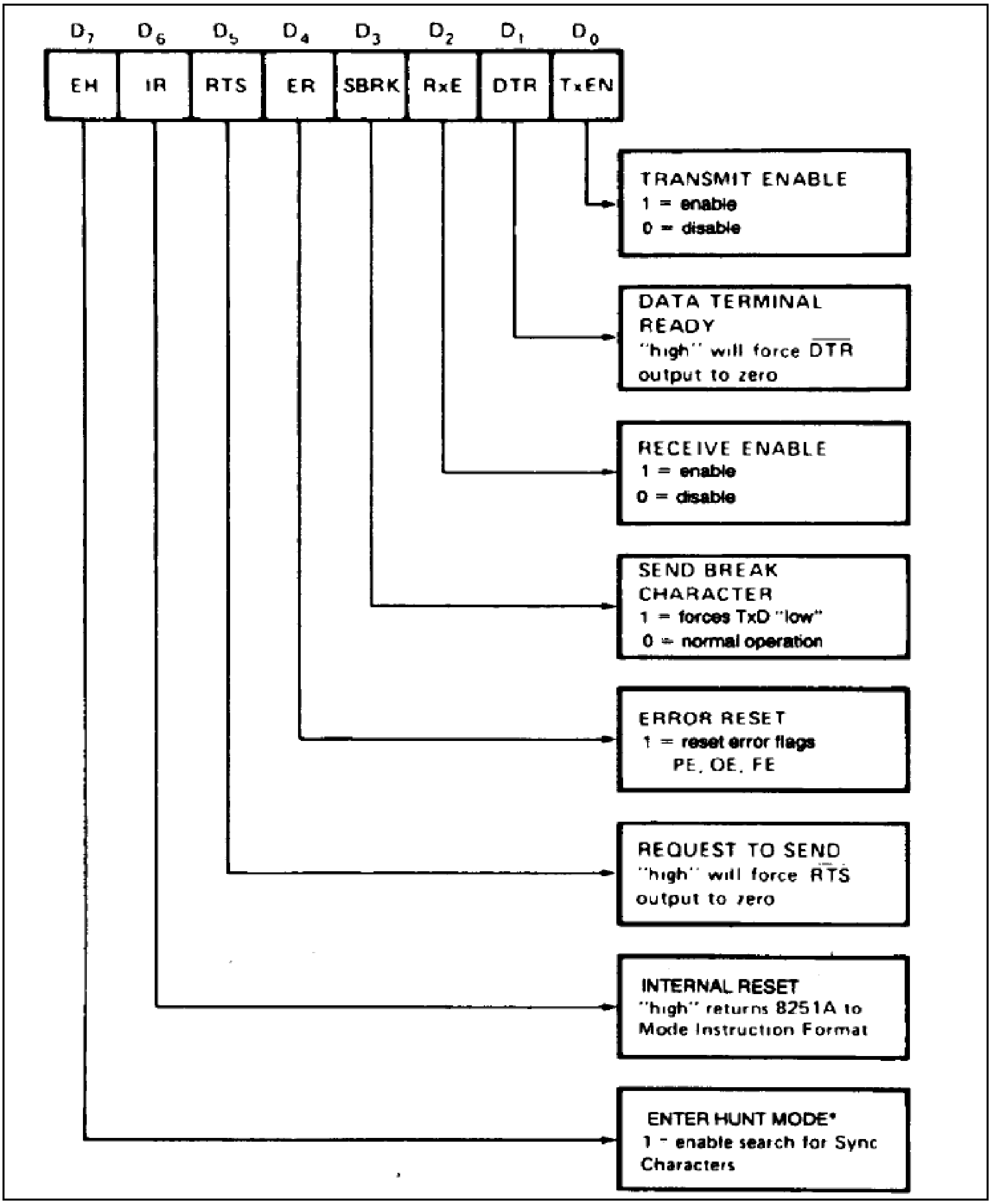


Fig. Command instruction format

Status Word: It is possible to see the internal status of the 8251 by reading a status word. The format of status word is shown below.

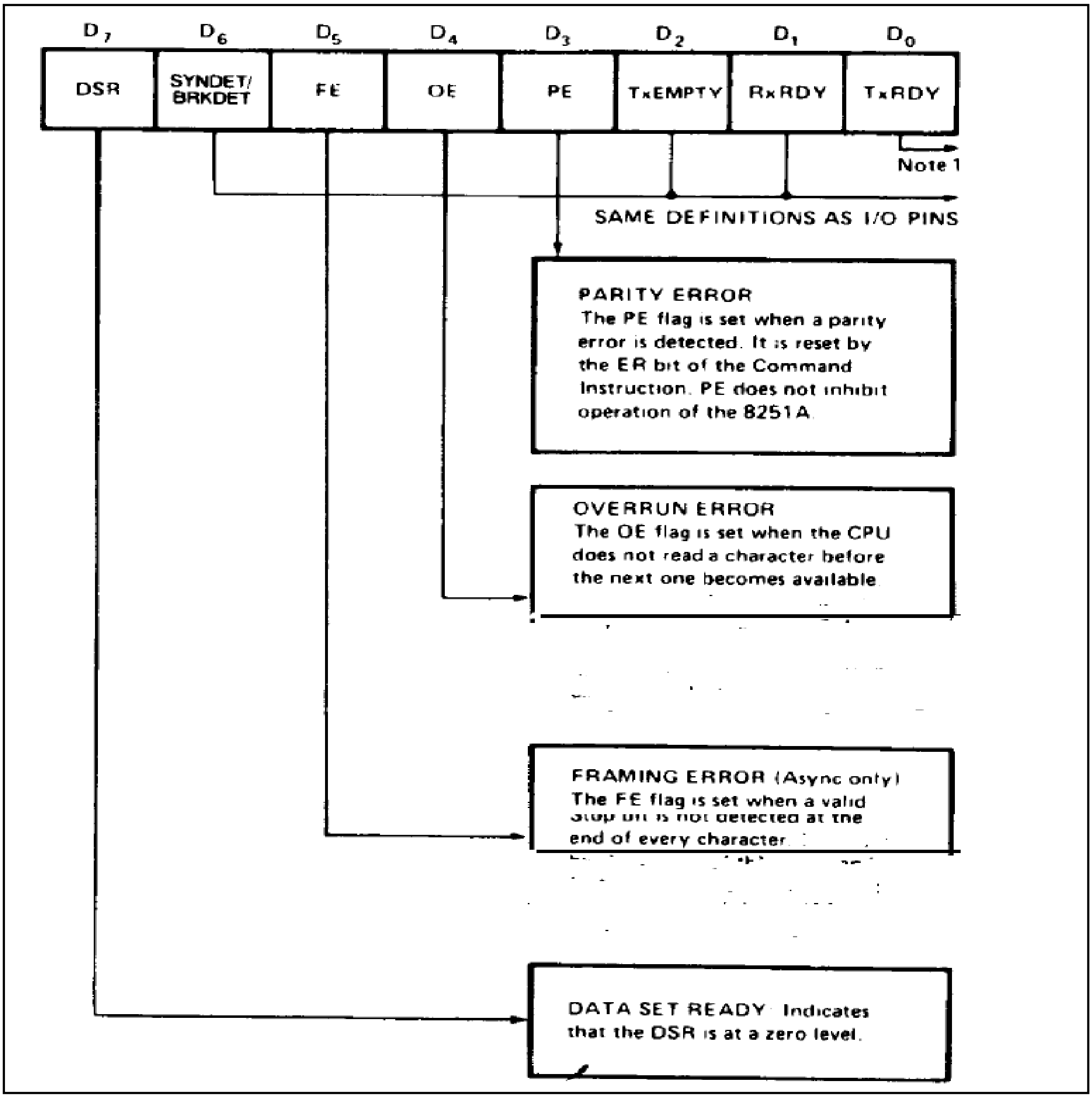

D7
D6
D5
D4
D3
D2
D1
D0
DSR
SYNDET/ BRKDET
FE
OE
PE
TxEMPTY
RxRDY
TxRDY
Note 1
SAME DEFINITIONS AS I/O PINS
PARITY ERROR
The PE flag is set when a parity error is detected. It is reset by the ER bit of the Command Instruction. PE does not inhibit operation of the 8251A.
OVERRUN ERROR
The OE flag is set when the CPU does not read a character before the next one becomes available.
FRAMING ERROR (Async only)
The FE flag is set when a valid stop bit is not detected at the end of every character.
DATA SET READY Indicates that the DSR is at a zero level.

Fig. Status word

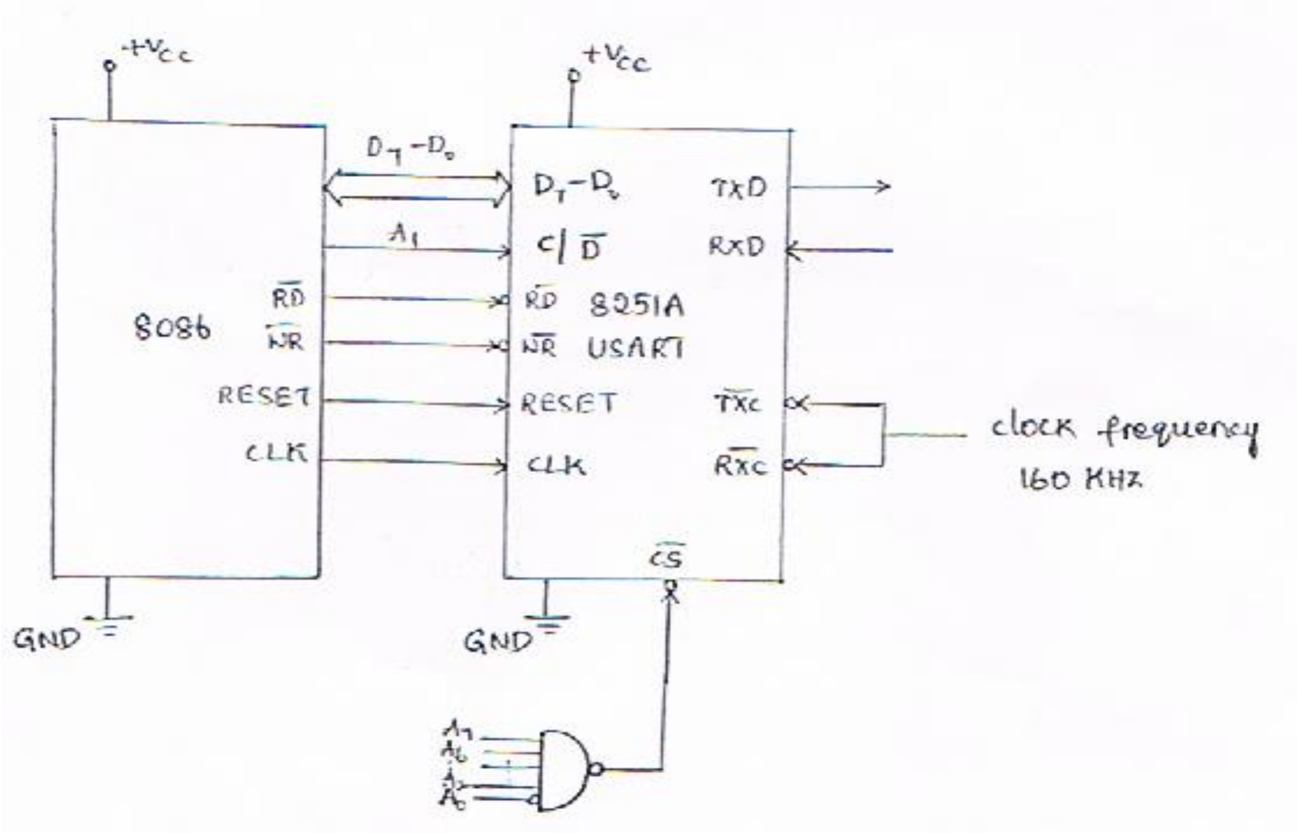
Programming Example:

Design the hardware interface circuit for interfacing 8251 with 8086. Set the 8251A in asynchronous mode as a transmitter and receiver with even parity enabled, 2 stop bits, 8-bit character length, frequency 160 KHz and baud rate 10K.

a). Write an ALP to transmit 100 bytes of data string starting at location 2000:5000H.

b). Write an ALP to receive 100 bytes of data string and store it at 3000:4000H

DESIGN



PORT ADDRESSES

A7	A6	A5	A4	A3	A2	A1=C/$\overline{D}$	A0	Address
1	1	1	1	1	1	1	0	FEH→Control word register
1	1	1	1	1	1	0	0	FCH→ Data

CONTROL WORD

Mode control word

1	1	1	1	1	1	1	0

Mode control word=FEH

Command word

0	0	0	1	0	0	0	1

Command word= 11H

PROGRAM

a). To transmit 100 bytes

```
        ASSUME CS: CODE
CODE    SEGMENT
        ORG 3000H
        MOV AX, 2000H
        MOV DS, AX
        MOV SI, 5000H
        MOV CL, 64H
        MOV AL, FEH
        OUT FEH, AL
        MOV AL, 11H
        OUT FEH, AL
WAIT:   IN AL, FEH
        AND AL, 01H
        JZ WAIT
        MOV AL, [SI]
        OUT FCH, AL
        INC SI
        DEC CL
        JNZ WAIT
        INT 03H
CODE    ENDS
        END
```

b) To receive 100 bytes of data

```
        ASSUME CS: CODE
CODE    SEGMENT
        MOV AX, 3000H
        MOV DS, AX
        MOV SI, 4000H
        MOV CL, 64H
        MOV AL, 7EH
        OUT FEH, AL
        MOV AL, 14H
        OUT FEH, AL
WAIT:   IN AL, FEH
        AND AL, 38H
        JZ READY
        MOV AL, 14H
        OUT FEH, AL
READY:IN AL, FEH
        AND AL, 02H
        JZ READY
        IN AL, FCH
        MOV [SI], AL
        INC SI
        DEC CL
        JNZ WAIT
        INT 03H
CODE    ENDS
        END
```

RECOMMENDED STANDARD -232C (RS-232C)

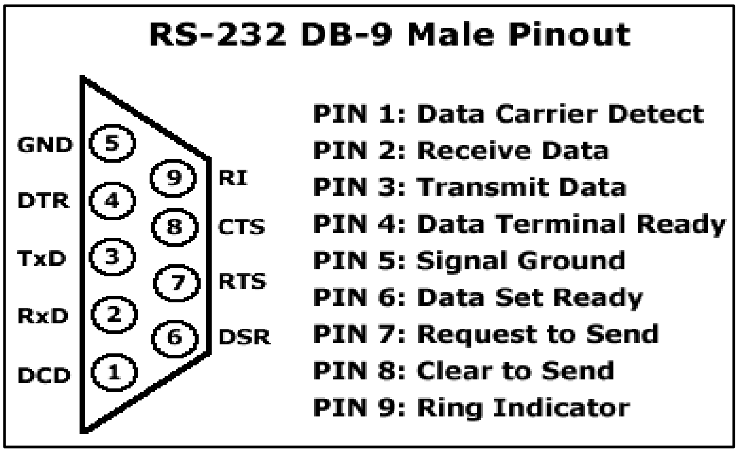
RS-232 was first introduced in 1962 by the *Radio Sector* of the Electronic Industries Association (EIA). RS-232 (Recommended standard-232) is a standard interface approved by the Electronic Industries Association (EIA) for connecting serial devices. In other words, RS-232 is a long-established standard that describes the physical interface and protocol for relatively low-speed serial data communication between computers and related devices. An industry trade group, the Electronic Industries Association (EIA), defined it originally for teletypewriter devices. In 1987, the EIA released a new version of the standard and changed the name to EIA-232-D. Many people, however, still refer to the standard as RS-232C, or just RS-232. RS-232 is the interface that your computer uses to talk to and exchange data with your modem and other serial devices. The serial ports on most computers use a subset of the RS-232C standard.

RS-232C is defined as the “Interface between data terminal equipment and data communications equipment using serial binary data exchange.” This definition defines data terminal equipment (DTE) as the computer, while data communications equipment (DCE) is the modem. A modem cable has pin-to-pin connections, and is designed to connect a DTE device to a DCE device. In addition to communications between computer equipment over telephone lines, RS-232C is now widely used for direct connections between data acquisition devices and computer systems. As in the definition of RS-232, the computer is data transmission equipment (DTE). RS-232C cables are commonly available with 4, 9 or 25-pin wiring. The 25-pin cable connects every pin; the 9-pin cables do not include many of the uncommonly used connections; 4-pin cables provide the bare minimum connections, and have jumpers to provide “handshaking” for those devices that require it.

An RS-232 serial port was once a standard feature of a personal computer, used for connections to modems, printers, mice, data storage, uninterruptible power supplies, and other peripheral devices. However, the low transmission speed, large voltage swing, and large standard connectors motivated development of the Universal Serial Bus, which has displaced RS-232 from most of its peripheral interface roles.

In RS-232, user data is sent as a time-series of bits. Both synchronous and asynchronous transmissions are supported by the standard. In addition to the data circuits, the standard defines a number of control circuits used to manage the connection between the DTE and DCE. Each data or control circuit only operates in one direction, which is, signaling from a DTE to the attached DCE or the reverse. Since transmit data and receive data are separate circuits, the interface can operate in a full duplex manner, supporting concurrent data flow in both directions.

The RS-232 standard defines the voltage levels that correspond to logical one and logical zero levels for the data transmission and the control signal lines. Valid signals are either in the range of +3 to +15 volts for logic 0 or the range -3 to -15 volts for logic 1, the range between -3 to +3 volts is not a valid RS-232 level. For data transmission lines (TxD, RxD and their secondary channel equivalents) logic one is defined as a negative voltage, the signal condition is called "mark." Logic zero is positive and the signal condition is termed "space." The 9-pin RS-232C standard is shown in figure below.



TTL TO RS-232C AND RS-232C TO TTL CONVERSION

The RS-232C standard does not define elements as the character encoding or the framing of characters, or error detection protocols. Details of character format and transmission bit rate are controlled by the serial port hardware, often a single integrated circuit called a USART that converts data from parallel to asynchronous start-stop serial form. Details of voltage levels, slew rate, and short-circuit behavior are typically controlled by a line driver (MC 1488) that converts from the USART's logic levels (TTL levels) to RS-232 compatible signal levels, and a receiver (MC 1489) that converts RS-232 compatible signal levels to the USART's logic levels (TTL levels). The figure shows the conversion of TTL to RS-232C and Rs-232C to TTL levels.

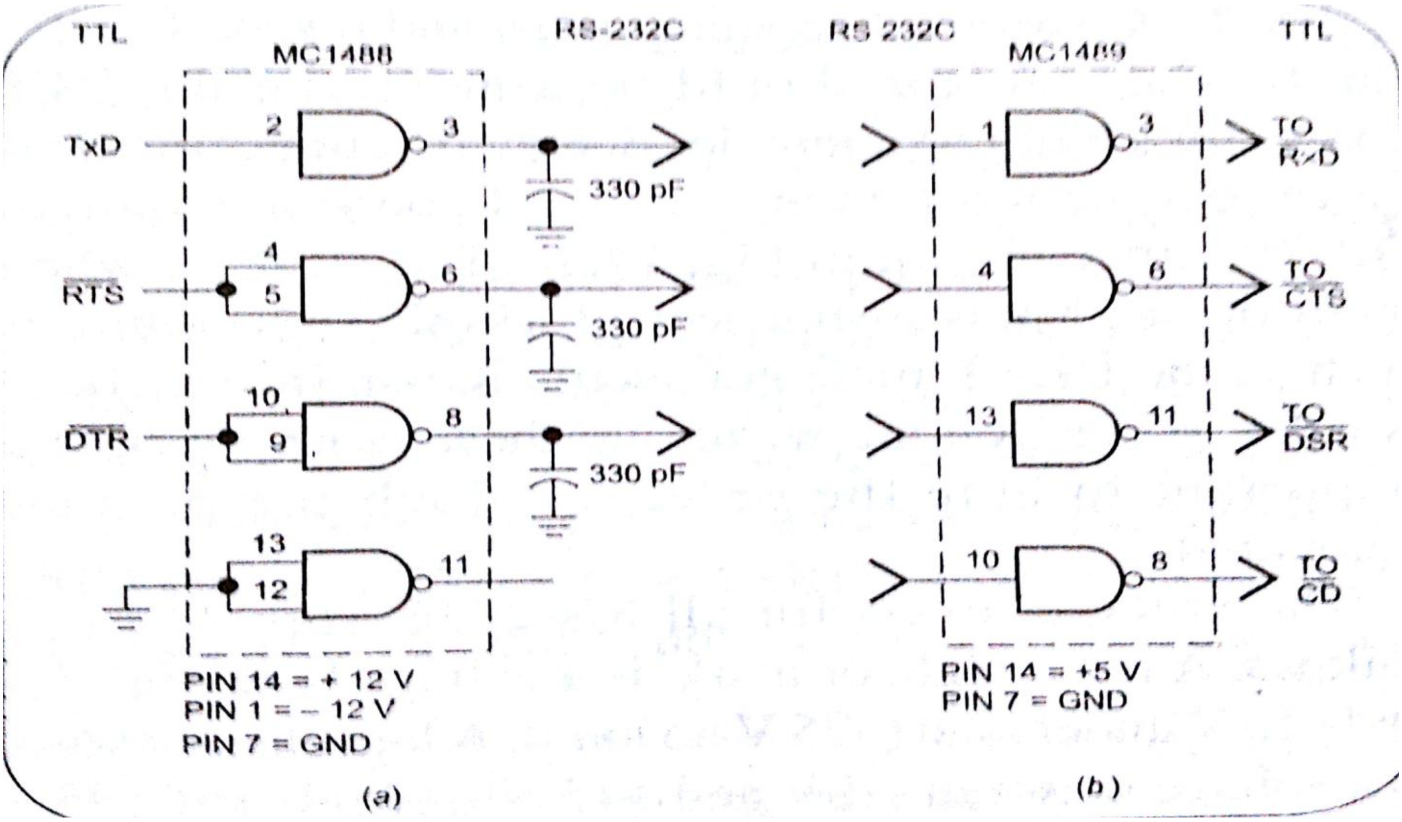


Fig. (a). TTL to Rs-232C and Fig. (b). RS-232C to TTL Conversion

INTRODUCTION TO HIGH SPEED SERIAL COMMUNICATION STANDARDS

In telecommunication and computer science, serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels. Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical. Serial computer buses are becoming more common even at shorter distances, as improved signal integrity and transmission speeds in newer serial technologies have begun to outweigh the parallel bus's advantage of simplicity.

Examples of serial communication standards are:

- Morse code telegraphy
- RS-232 (low-speed, implemented by serial ports)

- RS-422
- RS-423
- RS-485
- I^{2}C
- SPI
- ARINC 818 Avionics Digital Video Bus
- Atari SIO (Joe Decuir credits his work on Atari SIO as the basis of USB)
- **Universal Serial Bus (moderate-speed, for connecting peripherals to computers)**
- FireWire
- Ethernet
- Fibre Channel (high-speed, for connecting computers to mass storage devices)
- InfiniBand (very high speed, broadly comparable in scope to PCI)
- MIDI control of electronic musical instruments
- DMX512 control of theatrical lighting
- SDI-12 industrial sensor protocol
- CoaXPress industrial camera protocol over Coax
- Serial Attached SCSI
- Serial ATA
- SpaceWire Spacecraft communication network
- HyperTransport
- PCI Express
- SONET and SDH (high speed telecommunication over optical fibers)
- T-1, E-1 and variants (high speed telecommunication over copper pairs)
- MIL-STD-1553A/B

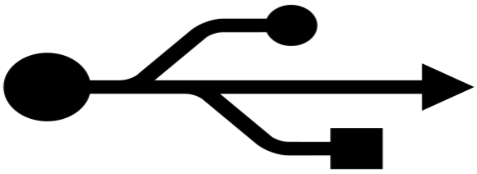
UNIVERSAL SERIAL BUS

INTRODUCTION

Universal Serial Bus (USB) is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication and power supply between computers and electronic devices.

USB was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has become commonplace on other devices, such as smart phones, PDAs and video game consoles. USB has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

As of 2008, approximately 6 billion USB ports and interfaces were in the global marketplace, and about 2 billion were being sold each year. The icon and cable of USB is shown in fig below.



HISTORY

A group of seven companies began the development of USB in 1994: Compaq, DEC, IBM, Intel, Microsoft, NEC and Nortel. The goal was to make it fundamentally easier to connect external devices to PCs by replacing the multitude of connectors at the back of PCs, addressing the usability issues of existing interfaces, and simplifying software configuration of all devices connected to USB, as well as permitting greater data rates for external devices. A team including Ajay Bhatt worked on the standard at Intel. The first integrated circuits supporting USB were produced by Intel in 1995.

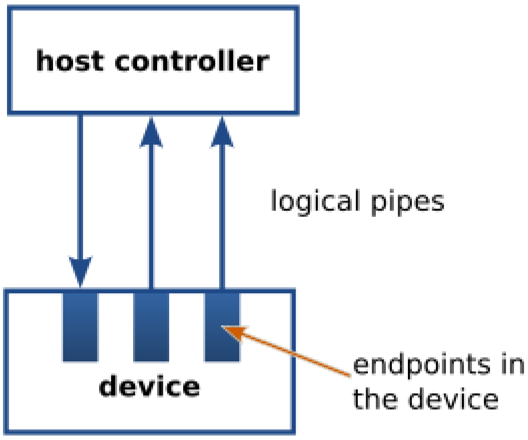
VERSIONS

USB 1 (Full Speed): Released in January 1996, USB 1 specified data rates of 1.5 Mbit/s (Low-Bandwidth) and 12 Mbit/s (Full-Bandwidth).

USB 2.0 (High Speed): USB 2.0: Released in April 2000. Added higher maximum signaling rate of 480 Mbit/s (effective throughput up to 35 MB/s or 280 Mbit/s) (now called "Hi-Speed").

USB 3.0 (Super Speed): USB 3.0 was released in November 2008. The standard claims a theoretical "maximum" transmission speed of up to 5 Gbit/s (625 MB/s). USB 3.0 reduces the time required for data transmission, reduces power consumption, and is backward compatible with USB 2.0.

SYSTEM DESIGN



The design architecture of USB is asymmetrical in its topology, consisting of a host, a multitude of downstream USB ports, and multiple peripheral devices connected in a tiered-star topology. Additional USB hubs may be included in the tiers, allowing branching into a tree structure with up to five tier levels. A USB host may implement multiple host controllers and each host controller may provide one or more USB ports. Up to 127 devices, including hub devices if present may be connected to a single host controller. USB devices are linked in series through hubs. One hub—built into the host controller—is the root hub. USB device communication is based on *pipes* (logical channels). A pipe is a connection from the host controller to a logical entity, found on a device, and named an *endpoint*. Because pipes correspond 1-to-1 to endpoints, the terms are sometimes used interchangeably. A USB device can have up to 32 endpoints, though USB devices seldom have this many endpoints. An endpoint is built into the USB device by the manufacturer and therefore exists permanently, while a pipe may be opened and closed.