Serial communication interface.
RS232 pinout.

Contents

What is RS232 Protocol?

The RS232 protocol is a popular serial interface that is used to connect computers to peripheral devices such as modems. We will take a look at the serial port pinouts used to implement RS232 as well as some additional reference information concerning the protocol.
The RS232 protocol transmits data of wires employing signal levels that differ from the standard 5V in order to minimize signal interference. It performs asynchronous transmission at a constant rate that is synchronized with the start pulse signal's level. Distances of up to 20 meters are the limit for reliable data transfer using the RS232 interface.

What does RS232 stand for?

Data transfer standards are developed by the Electronic Industry Association (EIA). The prefix RS denotes a Recommended Standard, and all of the EIA standards begin with those characters. The formal specification of RS232 is that it is an interface which uses serial binary data exchange to communicate between DTE and DCE devices. DTE is the acronym for Data Terminal Equipment and DCE represents Data Communication Equipment. The basic example of these two types of equipment defines a computer as a DTE device with a modem filling the role of DCE.

Serial communication is implemented by the transmission of serial data between the DTE and DCE. For instance, a computer (DTE) might send the binary data "11011101" serially to the modem (DCE) which then replies by sending "11010101" back to the DTE device.

The **RS232 protocol specifies** the operation mode, electrical standards, number of bits, and voltage levels to be used when transferring data between a DTE and DCE.

Serial connectors

Serial communication devices make use of 9 or 25 pin D-type connectors for their cabled connections. They are commonly designated as DB-9 or DB-25 with the number used to differentiate between the pin counts. Various manufacturers' names may replace the DB in the specifications. The plugs contain sockets and pins, with each pin numbered and labeled. A serial pinout diagram is presented below.

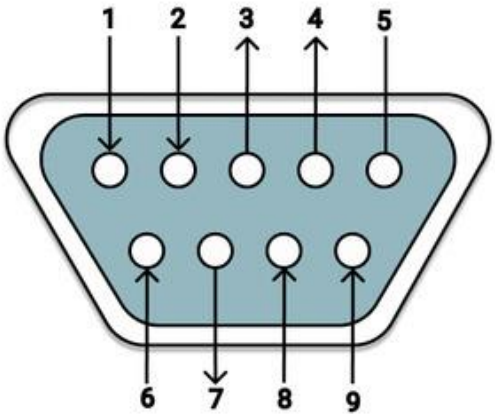
The RS232 protocol uses a 9 pin serial port that can have either male or female connectors. The most recent version of the protocol is known as RS232C.

RS232C retains the features of RS232 but uses 25 pins rather than a 9 pin serial pinout. Whether a DB9 serial pinout or a 25 pin connection is used, only three of the pins are required to connect terminal devices.

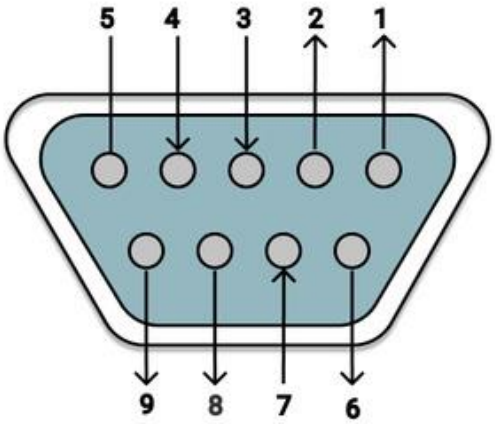
COM Port Pinout and Configuration

RS232 manages communication flowing between the DTE and DCE using serial pinouts of either the DB9 or DB25 variety. These D-sub connectors can terminate with an RS232 female pinout or DB25 or DB9 male connector pins. Each pin in a 9 or 25 serial connector pinout has its own distinct function.

DB-9 Male



DB-9 Female



DB-9 Male

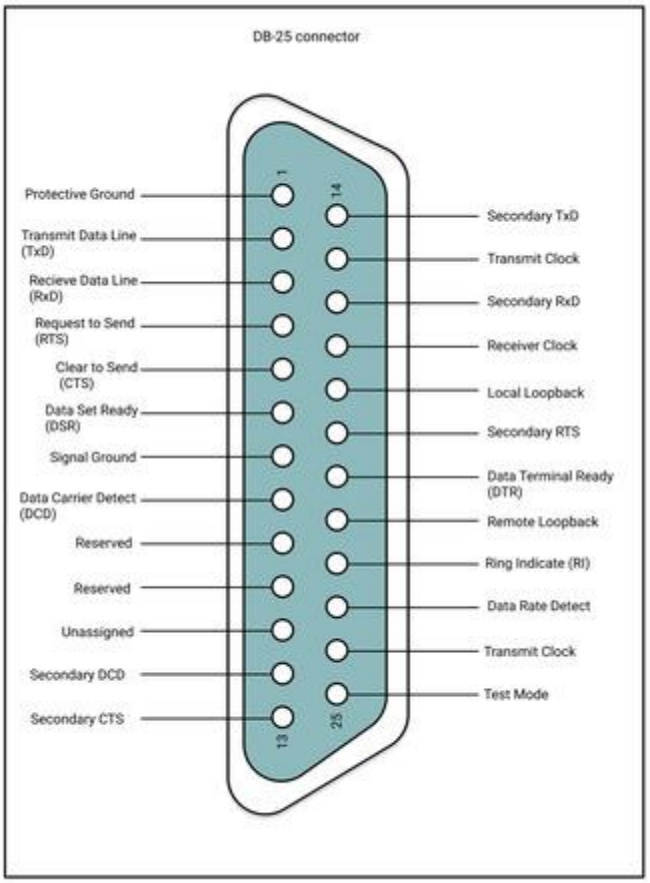
Pin	Signal Direction	Signal Name	Signal Function
1	→	CD	Carrier Detected
2	←	RxD	Receive Data
3	→	TxD	Transmit Data
4	→	DTR	Data Terminal Ready
5	—	GND	Ground
6	←	DSR	Data Set Ready
7	←	RTS	Request To Send
8	→	CTS	Clear To Send
9	→	RI	Ring Indicator

→ Transmitted from DTE Device
← Receive by DTE Device

DB-9 Female

Pin	Signal Direction	Signal Name	Signal Function
1	←	CD	Carrier Detected
2	→	TxD	Transmit Data
3	←	RxD	Receive Data
4	→	DTR	Data Terminal Ready
5	—	GND	Ground
6	←	DSR	Data Set Ready
7	→	CTS	Clear To Send
8	←	RTS	Request To Send
9	←	RI	Ring Indicator

→ Transmitted from DCE Device
← Receive by DCE Device



Functional Description:

In addition to defining electrical characteristics, RS232 specifies the signals used in serial cable pinouts and serial ports. Familiar items such as timing signals and ground are included in these specifications.

Following is a list of the signals used in an RS232 COM port pinout:

Protective Ground -This signal is connected to the chassis ground of the metallic connector.

Common Ground - Zero reference voltage level for all the control signals.

TxD (Transmit Pin) - To transmit data from DTE to DCE.

RxD (Receive Pin) - Sends data from DCE to DTE.

DTR (Data Terminal Ready) - DTE is ready to accept the request.

DCD (Data carrier Detect) - DCE accepts a carrier from a DTE located at a remote location.

DSR (Data Set Ready) - DCE is prepared to send and receive the information.

RI (Ring Indicator) - Detects the incoming ring tone on the telephone line.

RTS (Request to Send) - DTE call for DCE to send the data.

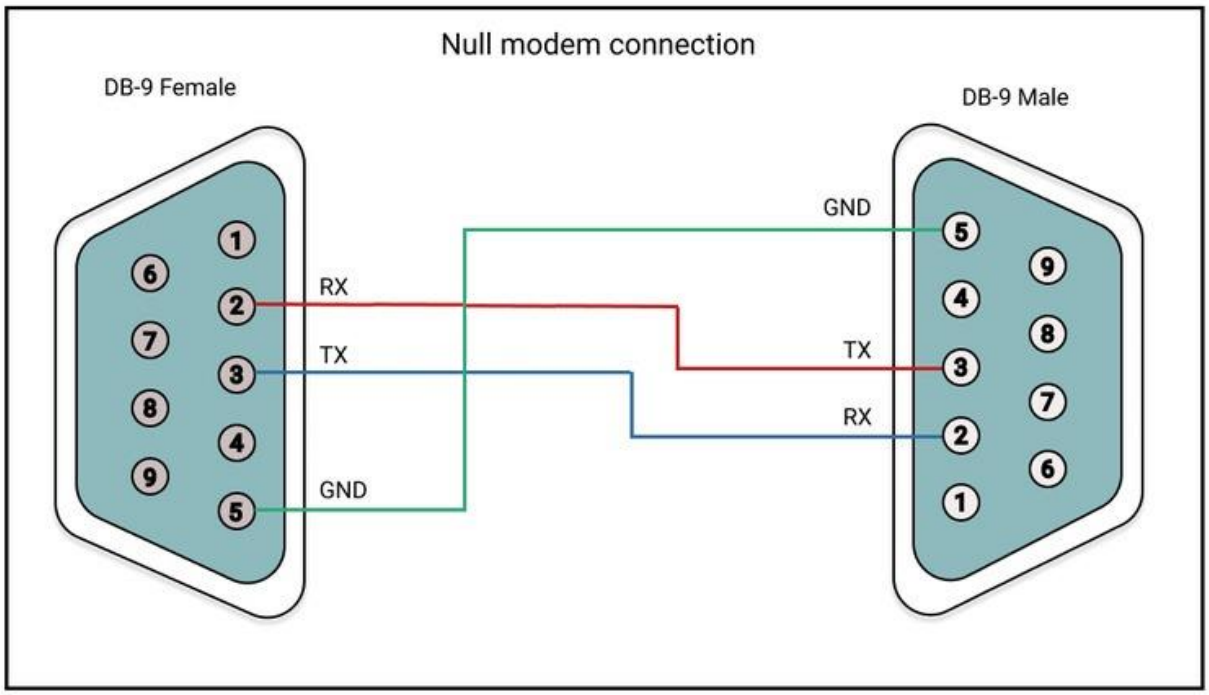
RTR (Ready to Receive) - DTE is geared up to receive data coming from DCE.

CTS (Clear To Send) - DCE is in a ready state to accept data coming from DTE.

These signals are the primary RS232 signals, but the protocol allows for secondary signals as well. They include secondary DTE, RTS, DCD, TxD, and RxD. The secondary signals are used to optionally connect DTE and DCE equipment.

RS-232 cables

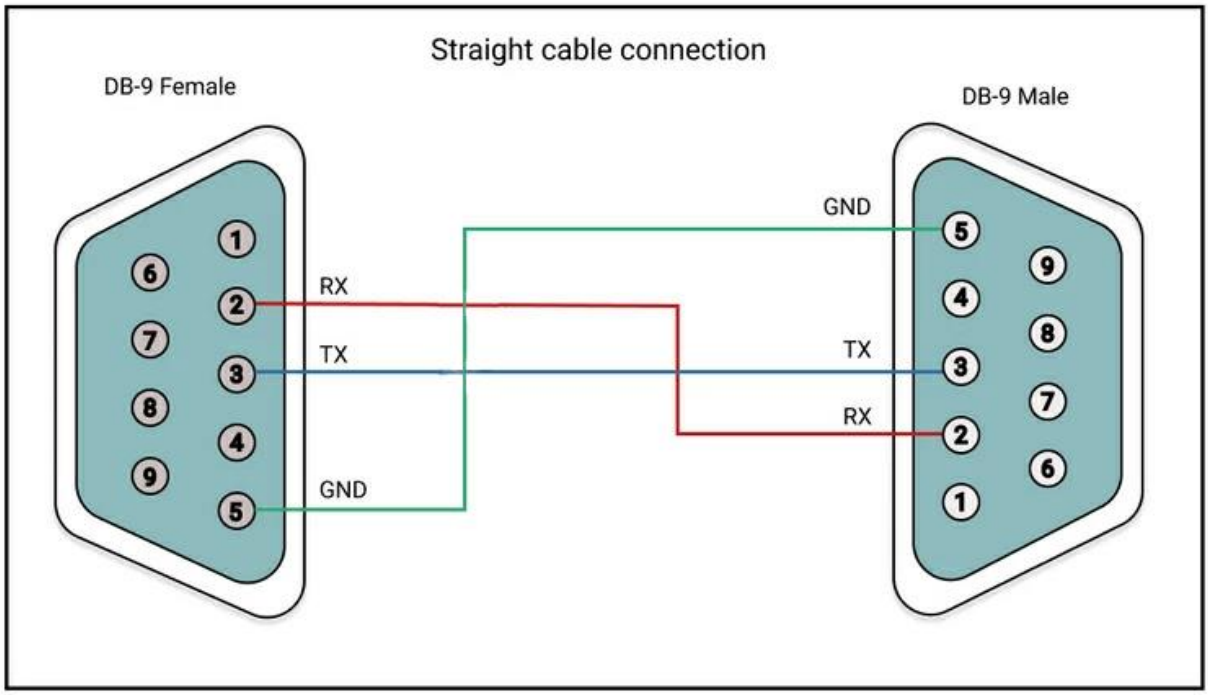
Null modems enable serial communication between DTE and DCE devices. An RS232 null modem pinout links the Tx pin of a male connector with the Rx pin on an RS232 female and the Rx male's pin to the female's Tx pin. Using the RS232 protocol you can connect two computers that do not have modems by using a null modem cable. This highlights one of the original uses of the RS232 protocol, which was developed in order to let teletype machines communicate with each other through their modems.



Straight-through cable

The other type of RS-232 Cable is the Straight-through cable. It is a one to one connector, It transmits a pin of one device that is connected to the

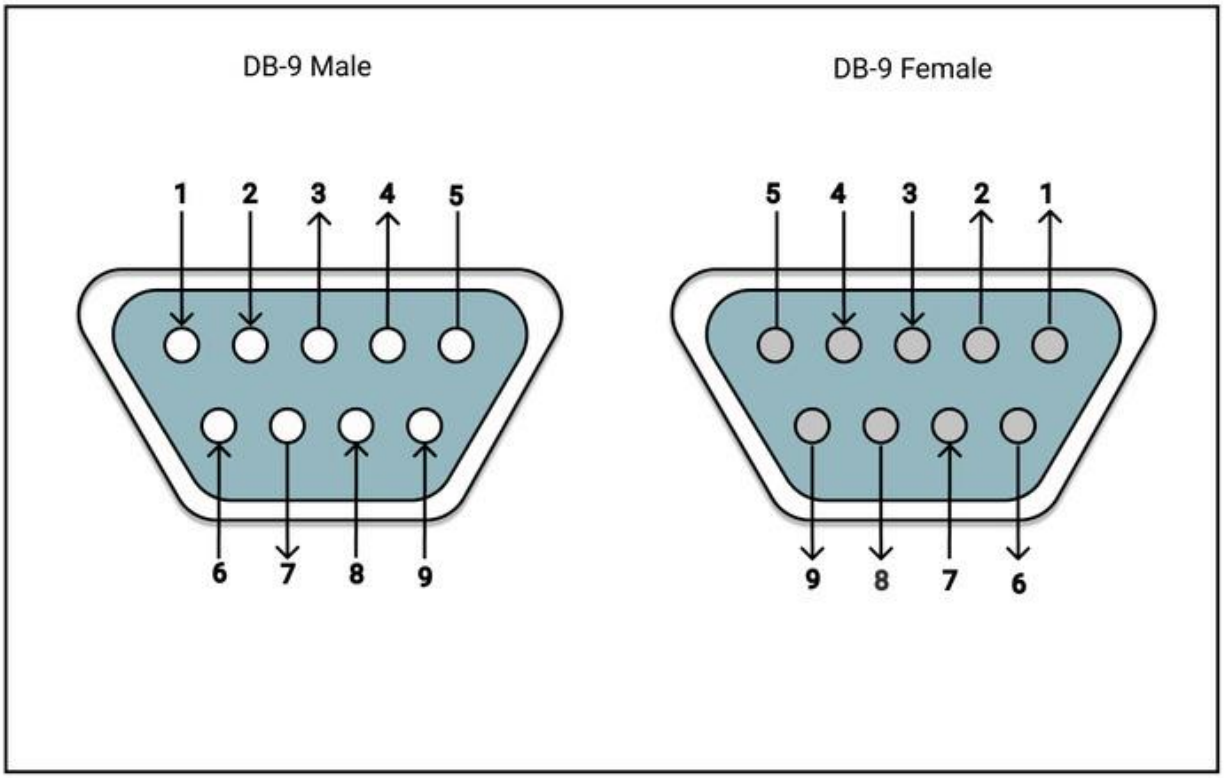
transmit pin of another device and the receiver pin of one device is connected to the receiver pin of another device.



Conclusion:

Modern hardware designs use innovative serial communication protocols like USB, Ethernet, and Wi-Fi.

But still, RS232 has proven to be used. The reason is, RS232 signals spread over longer distances. Moreover, it has better noise immunity. It is proven to be compatible across different manufacturers for interfacing computer and modems.



DB-9 Male

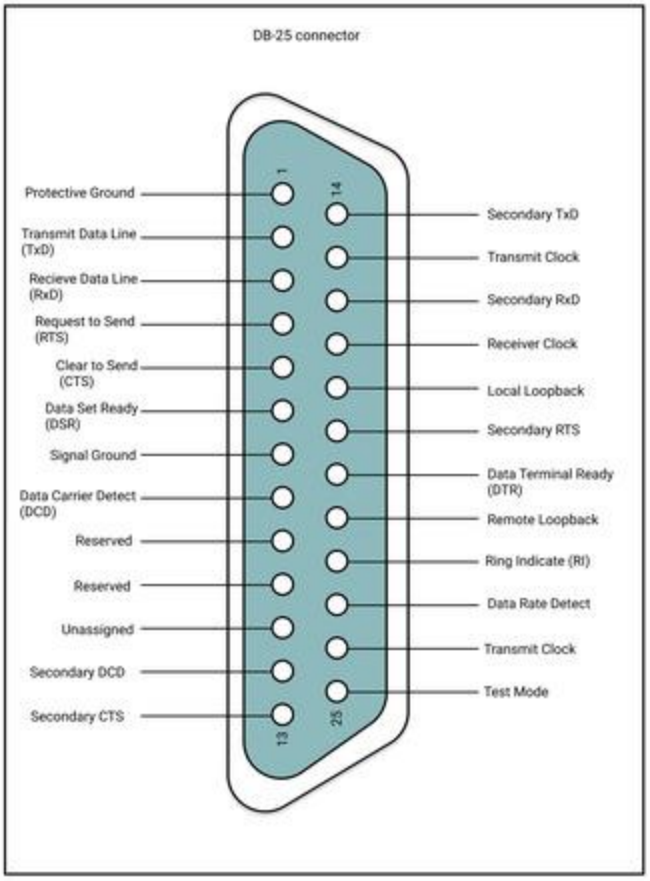
Pin	Signal Direction	Signal Name	Signal Function
1	→	CD	Carrier Detected
2	←	RxD	Receive Data
3	→	TxD	Transmit Data
4	→	DTR	Data Terminal Ready
5	—	GND	Ground
6	←	DSR	Data Set Ready
7	←	RTS	Request To Send
8	→	CTS	Clear To Send
9	→	RI	Ring Indicator

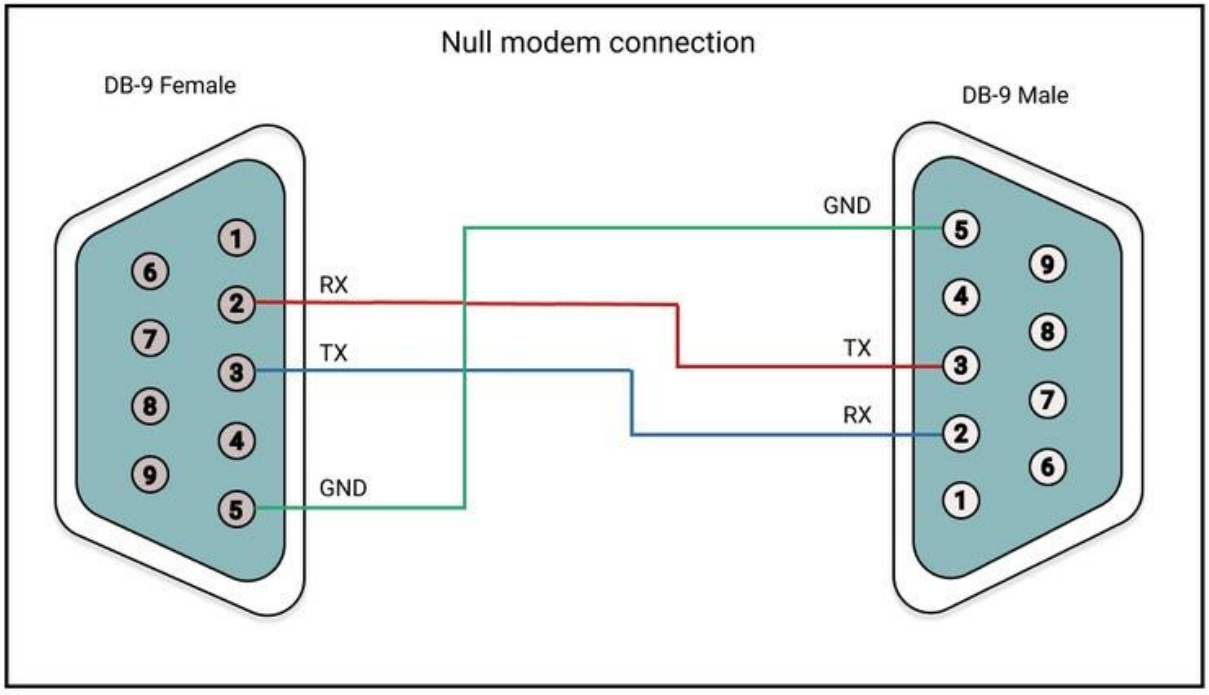
→ Transmitted from DTE Device
← Receive by DTE Device

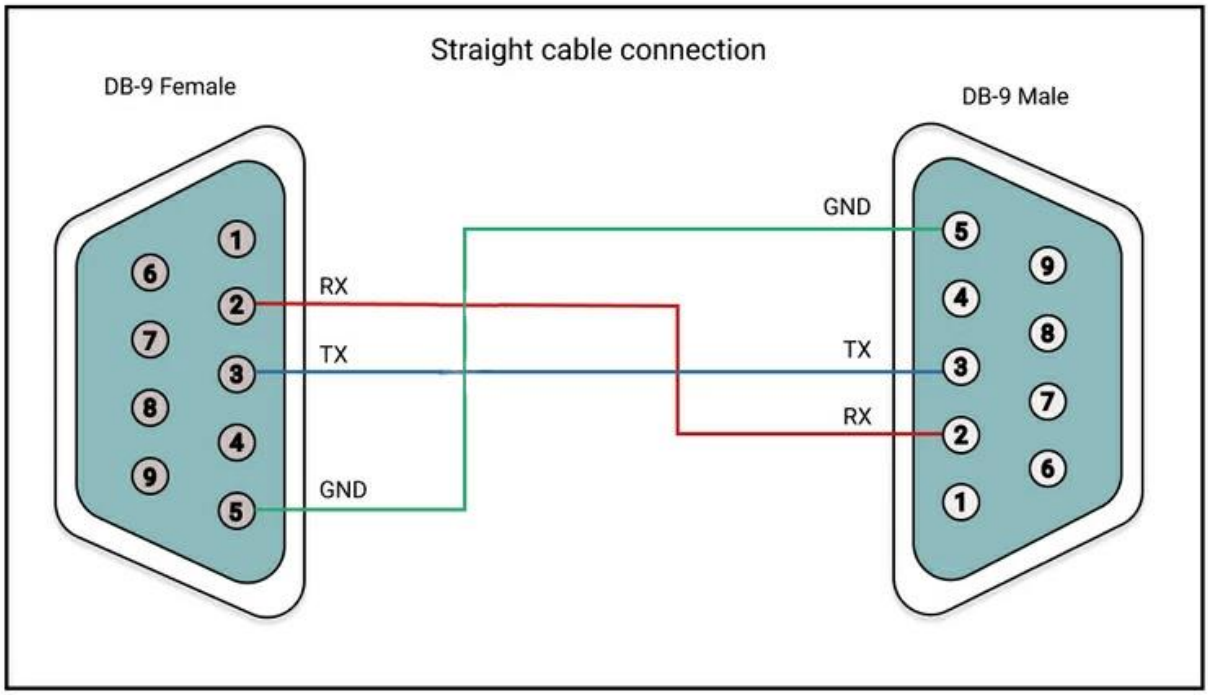
DB-9 Female

Pin	Signal Direction	Signal Name	Signal Function
1	←	CD	Carrier Detected
2	→	TxD	Transmit Data
3	←	RxD	Receive Data
4	→	DTR	Data Terminal Ready
5	—	GND	Ground
6	←	DSR	Data Set Ready
7	→	CTS	Clear To Send
8	←	RTS	Request To Send
9	←	RI	Ring Indicator

→ Transmitted from DCE Device
← Receive by DCE Device


DB-25 connector
Protective Ground
Transmit Data Line (TxD)
Recieve Data Line (RxD)
Request to Send (RTS)
Clear to Send (CTS)
Data Set Ready (DSR)
Signal Ground
Data Carrier Detect (DCD)
Reserved
Reserved
Unassigned
Secondary DCD
Secondary CTS
1
13
14
25
Secondary TxD
Transmit Clock
Secondary RxD
Receiver Clock
Local Loopback
Secondary RTS
Data Terminal Ready (DTR)
Remote Loopback
Ring Indicate (RI)
Data Rate Detect
Transmit Clock
Test Mode


Null modem connection
DB-9 Female
DB-9 Male
RX
TX
GND
GND
TX
RX

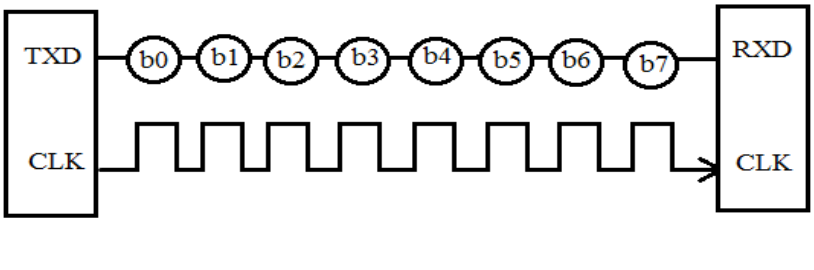

Straight cable connection
DB-9 Female
DB-9 Male
RX
TX
GND
GND
TX
RX

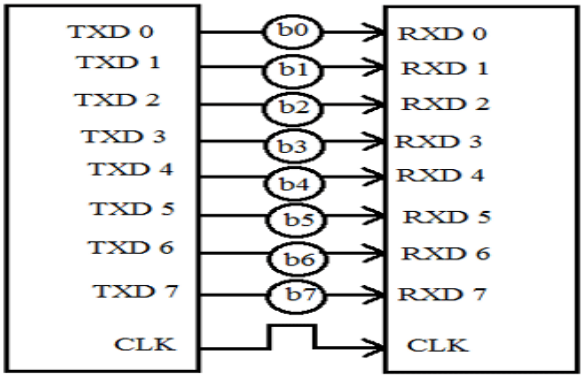
RS232 Serial Communication Protocol: Basics, Working & Specifications

One of the oldest, yet popular communication protocol that is used in industries and commercial products is the RS232 Communication Protocol. The term RS232 stands for "Recommended Standard 232" and it is a type of serial communication used for transmission of data normally in medium distances. It was introduced back in the 1960s and has found its way into many applications like computer printers, factory automation devices etc. Today there are many modern communication protocols like the RS485, SPI, I2C, CAN etc..

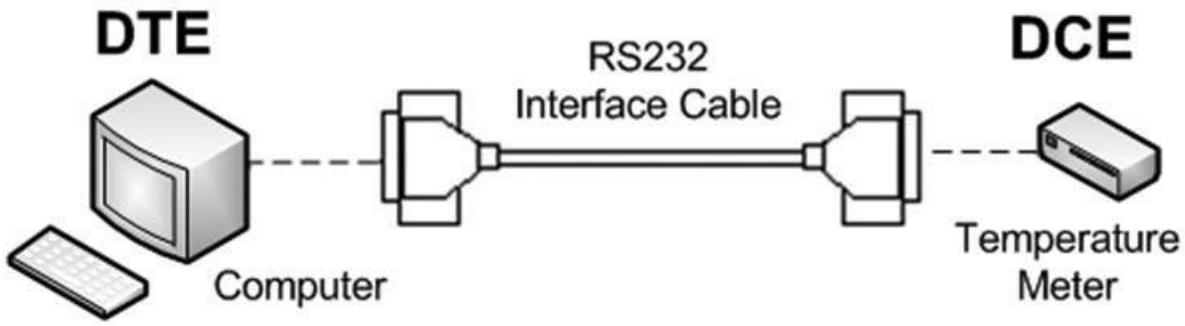
What is a serial communication?

In telecommunication, the process of sending data sequentially over a computer bus is called as serial communication, which means the data will be transmitted bit by bit. While in parallel communication the data is transmitted in a byte (8 bit) or character on several data lines or buses at a time. Serial communication is slower than parallel communication but used for long data transmission due to lower cost and practical reasons.





Modes of Data Transfer in Serial Communication:



Asynchronous Data Transfer – The mode in which the bits of data are not synchronized by a clock pulse. Clock pulse is a signal used for synchronization of operation in an electronic system.

Synchronous Data Transfer – The mode in which the bits of data are synchronized by a clock pulse.

Characteristics of Serial Communication:

Baud rate is used to measure the speed of transmission. It is described as the number of bits passing in one second. For example, if the baud rate is 200 then 200 bits per Sec passed. In telephone lines, the baud rates will be 14400, 28800 and 33600.

Stop Bits are used for a single packet to stop the transmission which is denoted as "T". Some typical values are 1, 1.5 & 2 bits.

Parity Bit is the simplest form of checking the errors. There are of four kinds, i.e., even odd, marked and spaced. For example, If 011 is a number the parity bit=0, i.e., even parity and the parity=1, i.e., odd parity.

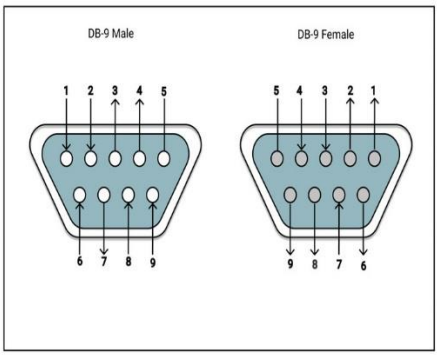
What is RS232?

RS232C "Recommended Standard 232C" *is the recent version of Standard 25 pin whereas,* ***RS232D*** *which is of 22 pins. In new PC's male D-type which is of 9 pins.*

RS232 is a standard protocol used for serial communication, it is used for connecting computer and its peripheral devices to allow serial data exchange between them.

As it obtains the voltage for the path used for the data exchange between the devices. It is used in serial communication up to 50 feet with the rate of 1.492kbps.

As EIA defines, the RS232 is used for connecting **Data Transmission Equipment (DTE)** and **Data Communication Equipment (DCE)**.



DB-9 Male

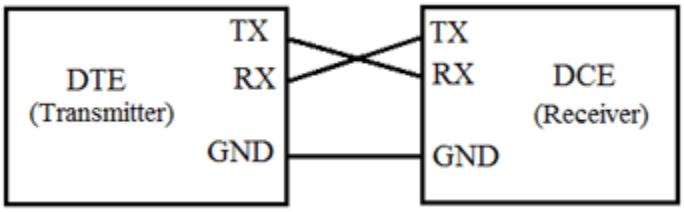
Pin	Signal Direction	Signal Name	Signal Function
1	→	CD	Carrier Detected
2	←	RxD	Receive Data
3	→	TxD	Transmit Data
4	→	DTR	Data Terminal Ready
5	—	GND	Ground
6	←	DSR	Data Set Ready
7	←	RTS	Request To Send
8	→	CTS	Clear To Send
9	→	RI	Ring Indicator

→ Transmitted from DTE Device
← Receive by DTE Device

DB-9 Female

Pin	Signal Direction	Signal Name	Signal Function
1	←	CD	Carrier Detected
2	→	TxD	Transmit Data
3	←	RxD	Receive Data
4	→	DTR	Data Terminal Ready
5	—	GND	Ground
6	←	DSR	Data Set Ready
7	→	CTS	Clear To Send
8	←	RTS	Request To Send
9	←	RI	Ring Indicator

→ Transmitted from DCE Device
← Receive by DCE Device



Universal Asynchronous Data Receiver &Transmitter (UART) used in connection with RS232 for transferring data between printer and computer. The microcontrollers are not able to handle such kind of voltage levels, connectors are connected between RS232 signals. These connectors are known as the **DB-9 Connector** as a serial port and they are of two type's **Male connector (DTE) & Female connector (DCE).**

Electrical Specifications

Let us discuss the electrical specifications of RS232 given below:

- **Voltage Levels:** RS232 also used as ground & 5V level. Binary 0 works with voltages up to +5V to +15Vdc. It is called as 'ON' or spacing (high voltage level) whereas Binary 1 works with voltages up to -5V to -15Vdc. It is called as 'OFF' or marking (low voltage level).
- **Received signal voltage level:** Binary 0 works on the received signal voltages up to +3V to +13 Vdc & Binary 1 works with voltages up to -3V to -13 Vdc.
- **Line Impedances:** The impedance of wires is up to 3 ohms to 7 ohms & the maximum cable length are 15 meters, but new maximum length in terms of capacitance per unit length.
- **Operation Voltage:** The operation voltage will be 250v AC max.
- **Current Rating:** The current rating will be 3 Amps max.
- **Dielectric withstanding voltage:** 1000 VAC min.
- **Slew Rate:** The rate of change of signal levels is termed as Slew Rate. With its slew rate is up to 30 V/microsecond and the maximum bitrate will be 20 kbps.

Note : The ratings and specification changes with the change in equipment model.

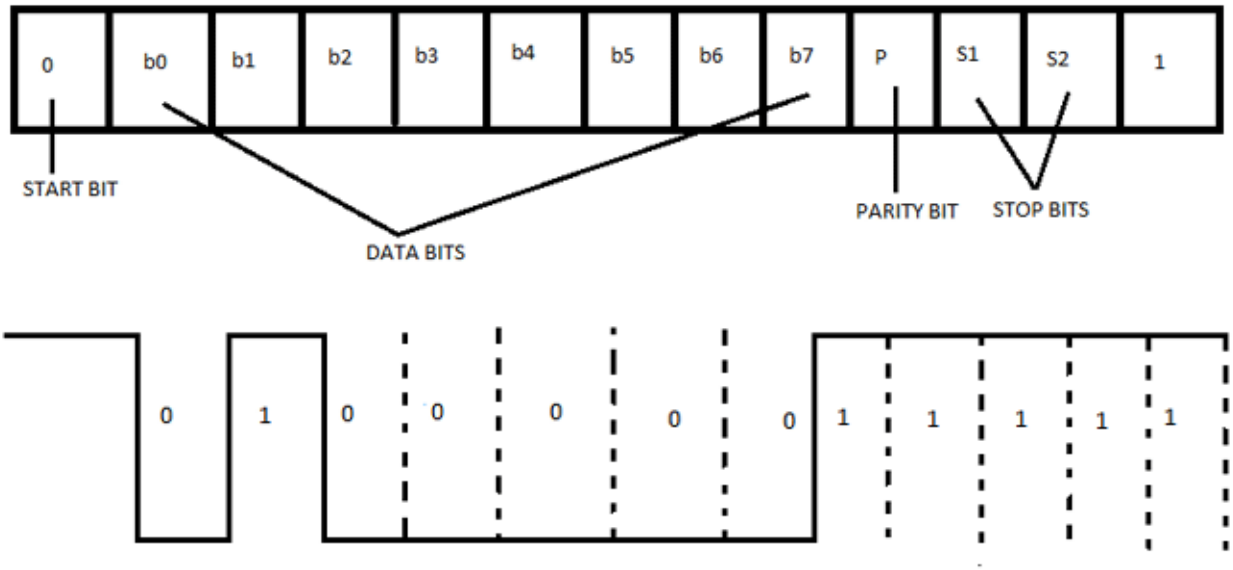
How RS232 Works?

RS232 works on the two-way communication that exchanges data to one another. There are two devices connected to each other, **(DTE) Data Transmission Equipment & (DCE) Data Communication Equipment** which has the pins like **TXD, RXD, and RTS & CTS**.

Now, from **DTE** source, the **RTS** generates the *request to send* the data.
Then from the other side **DCE**, the **CTS,** clears the path for receiving the data.
After clearing a path, it will give a signal to **RTS** of the **DTE** source to send the signal.
Then the bits are transmitted from **DTE** to **DCE**.
Now again from **DCE** source, the request can be generated
by **RTS** and **CTS** of **DTE** sources clears the path for receiving the data and gives a signal to send the data.
This is the whole process through which data transmission takes place.

RS-232 Function	Purpose
Transmit Date (TD)	Carries data from DTE to DCE
Receive Data (RD)	Carries data from DCE to DTE
Request to Send (RTS)	DTE requests DCE to prepare to receive data
Clear to Send (CTS)	Indicates DCE is ready to accept data
DCE Ready (DSR)	DCE is ready to receive commands or data
Received Line Signal Detector (DCD)	DCE is connected to the line
DTE Ready (DTR)	Indicates presence of DTE to DCE
Ring Indicator (RI)	DCE has detected and incoming ring signal on the line

For example: The signals set to logic 1, i.e., -12V. The data transmission starts from next bit and to inform this, DTE sends start bit to DCE. The start bit is always '0', i.e., +12 V & next 5 to 9 characters is data bits. If we use parity bit, then 8 bits data can be transmitted whereas if parity doesn't use, then 9 bits are being transmitted. The stop bits are sent by the transmitter whose values are 1, 1.5 or 2 bits after the data transmission.

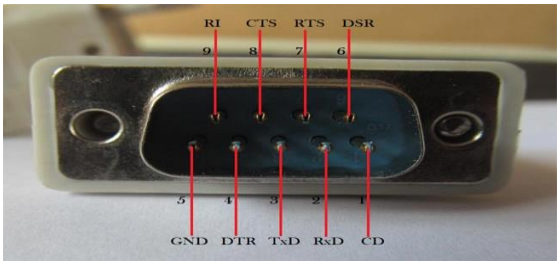


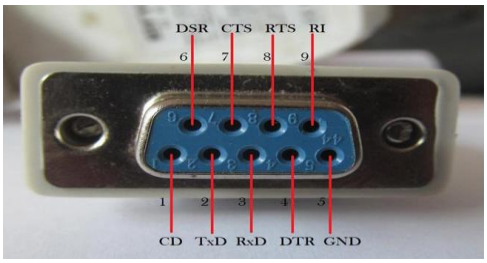
Mechanical Specification

For mechanical specifications, we have to study about two types of connectors that is **DB-25** and **DB-9**. In DB-25, there are 25 pins available which are used for many of the applications, but some of the applications didn't use the whole 25 pins. So, the 9 pin connector is made for the convenience of the devices and equipments.

Now, here we are discussing the **DB-9** pin connector which is used for connection between microcontrollers and connector. These are of two types: **Male Connector (DTE)** & **Female Connector (DCE)**. There are 5 pins on the top row and 4 pins in the bottom row. It is often called **DE-9 or D-type connector.**

Pin Structure of DB-9 Connector:





Pin Description DB-9 Connector:

PIN No.	Pin Name	Pin Description
1	CD (Carrier Detect)	Incoming signal from DCE
2	RD (Receive Data)	Receives incoming data from DTE
3	TD (Transmit Data)	Send outgoing data to DCE
4	DTR (Data Terminal Ready)	Outgoing handshaking signal
5	GND (Signal ground)	Common reference voltage
6	DSR (Data Set Ready)	Incoming handshaking signal
7	RTS (Request to Send)	Outgoing signal for controlling flow
8	CTS (Clear to Send)	Incoming signal for controlling flow
9	RI (Ring Indicator)	Incoming signal from DCE

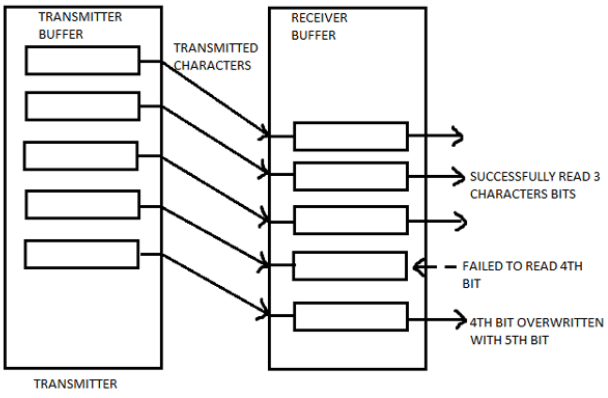
What is Handshaking?

How can a transmitter, transmits and the receiver receives data successfully. So, the Handshaking defines, for this reason.

Handshaking is the process which is used to transfer the signal from DTE to DCE to make the connection before the actual transfer of data. The messaging between transmitter & receiver can be done by handshaking.

There are **3 types of handshaking processes** named as:-

No Handshaking:
If there is no handshaking, then DCE reads the already received data while DTE transmits the next data. All the received data stored in a memory location known as receiver's buffer. This buffer can only store one bit so receiver must read the memory buffer before the next bit arrives. If the receiver is not able to read the stored bit in the buffer and next bit arrives then the stored bit will be lost.
As shown in below diagram, a receiver was unable to read the 4^{th} bit till the 5^{th} bit arrival and this result overriding of 4^{th} bit by 5^{th} bit and 4^{th} bit is lost.



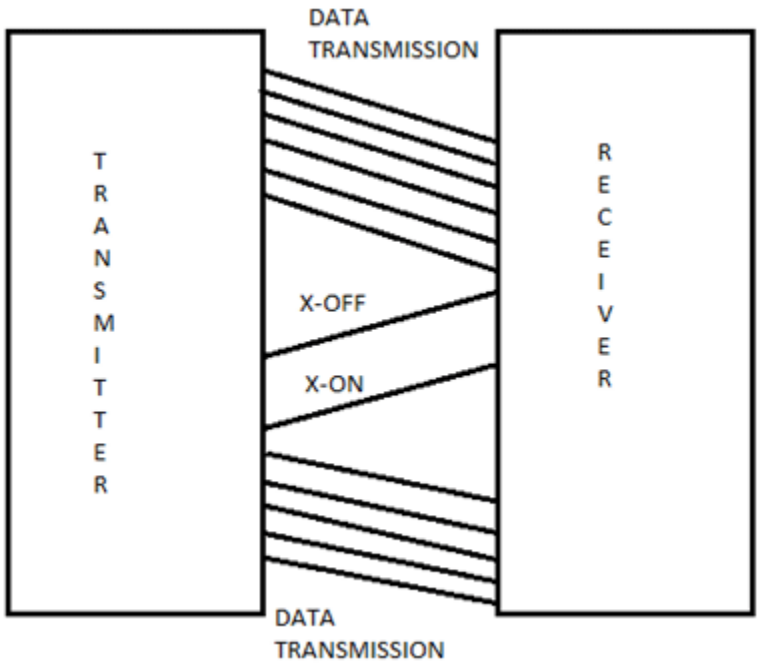
Hardware Handshaking:

- It uses specific serial ports, i.e., RTS & CTS to control data flow.
- In this process, transmitter asks the receiver that it is ready to receive data then receiver checks the buffer that it is empty, if it is empty then it will give signal to the transmitter that I am ready to receive data.
- The receiver gives the signal to transmitter not to send any data while already received data cannot be read.

- Its working process is same as above described in handshaking.

Software Handshaking:

- In this process, there are two forms, i.e., X-ON & X-OFF. Here, ‘X’ is the transmitter.
- X-ON is the part in which it resumes the data transmission.
- X-OFF is the part in which it pauses the data transmission.
- It is used to control the data flow and prevent loss during transmission.



Applications of RS232 Communication

- RS232 serial communication is used in old generation PCs for connecting the peripheral devices like mouse, printers, modem etc.
- Nowadays, RS232 is replaced by advanced USB.
- It is also used in PLC machines, CNC machines, and servo controllers because it is far cheaper.
- It is still used by some microcontroller boards, receipt printers, point of sale system (PoS), etc.

I2C Communication Protocol

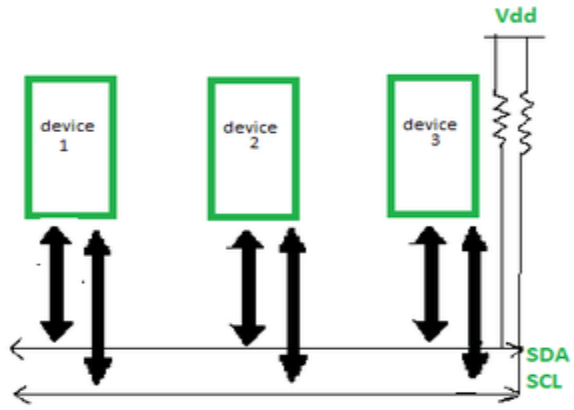
I2C stands for **Inter-Integrated Circuit.** It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface (TWI).

Working of I2C Communication Protocol :
It uses only 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high.
Serial Data (SDA) – Transfer of data takes place through this pin.
Serial Clock (SCL) – It carries the clock signal.

I2C operates in 2 modes –

- Master mode
- Slave mode

Each data bit transferred on SDA line is synchronized by a high to the low pulse of each clock on the SCL line.

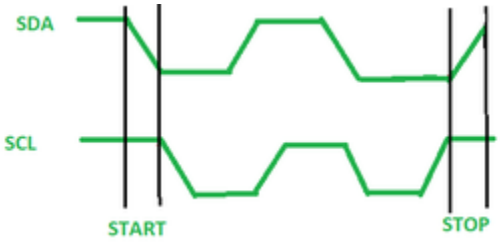


According to I2C protocols, the data line can not change when the clock line is high, it can change only when the clock line is low. The 2 lines are open drain, hence a pull-up resistor is required so that the lines are high since the devices on the I2C bus are active low. The data is transmitted in the form of packets which comprises 9 bits. The sequence of these bits are –

1. **Start Condition** – 1 bit
2. **Slave Address** – 8 bit
3. **Acknowledge** – 1 bit

Start and Stop Conditions :

START and STOP can be generated by keeping the SCL line high and changing the level of SDA. To generate START condition the SDA is changed from high to low while keeping the SCL high. To generate STOP condition SDA goes from low to high while keeping the SCL high, as shown in the figure below.



Start and Stop Condition

Repeated Start Condition :

Between each start and stop condition pair, the bus is considered as busy and no master can take control of the bus. If the master tries to initiate a new transfer and does not want to release the bus before starting the new transfer, it issues a new START condition. It is called a REPEATED START condition.

Read/Write Bit :

A high Read/Write bit indicates that the master is sending the data to the slave, whereas a low Read/Write bit indicates that the master is receiving data from the slave.

ACK/NACK Bit :

After every data frame, follows an ACK/NACK bit. If the data frame is received successfully then ACK bit is sent to the sender by the receiver.

Addressing :

The address frame is the first frame after the start bit. The address of the slave with which the master wants to communicate is sent by the master to every slave connected with it. The slave then compares its own address with this address and sends ACK.

I2C Packet Format :

In the I2C communication protocol, the data is transmitted in the form of packets. These packets are 9 bits long, out of which the first 8 bits are put in SDA line and the 9th bit is reserved for ACK/NACK i.e. Acknowledge or Not Acknowledge by the receiver.

START condition plus **address packet** plus one more **data packet** plus **STOP condition** collectively form a complete **Data transfer.**

Features of I2C Communication Protocol :

- **Half-duplex Communication Protocol –**
  Bi-directional communication is possible but not simultaneously.
- **Synchronous Communication** – The data is transferred in the form of frames or blocks. Can be configured in a multi-master configuration.
- **Clock Stretching –**
  The clock is stretched when the slave device is not ready to accept more data by holding the SCL line low, hence disabling the master to raise the clock line. Master will not be able to raise the clock line because the wires are AND wired and wait until the slave releases the SCL line to show it is ready to transfer next bit.
- **Arbitration** –
  I2C protocol supports multi-master bus system but more than one bus can not be used simultaneously. The SDA and SCL are monitored by the masters. If the SDA is found high when it was supposed to be low it will be inferred that another master is active and hence it stops the transfer of data.
- **Serial transmission –**
  I2C uses serial transmission for transmission of data.
- Used for low-speed communication.

Advantages :

- Can be configured in multi-master mode.
- Complexity is reduced because it uses only 2 bi-directional lines (unlike SPI Communication).
- Cost-efficient.
- It uses ACK/NACK feature due to which it has improved error handling capabilities.

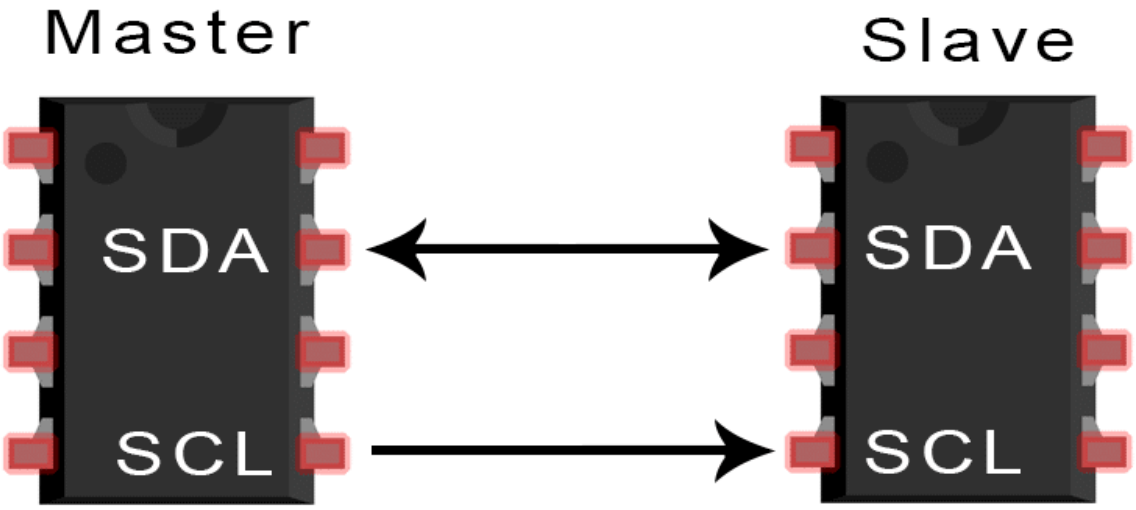
Limitations :

- Slower speed.
- Half-duplex communication is used in the I2C communication protocol.

Comparison between I2C and SPI Communication Protocols

Features	I2C Communication Protocol	SPI Communication Protocol
Number of wires	2 (SDA and SCL)	4 (MOSI, MISO, SCK, and SS)
Communication type	Half-duplex	Full-duplex
Maximum number of devices	Limited by addressing scheme	Limited by number of chip select (SS) lines
Data transfer speed	Slower	Faster
Error handling	Improved due to ACK/NACK feature	Not as robust
Cost	Cost-efficient due to fewer wires	More expensive due to additional wires
Complexity	Simpler due to fewer wires	More complex due to additional wires
Multi-master configuration	Yes	Yes
Synchronous communication	Yes	Yes
Clock stretching	Yes	No
Arbitration	Yes	No

BASICS OF THE I2C COMMUNICATION PROTOCOL

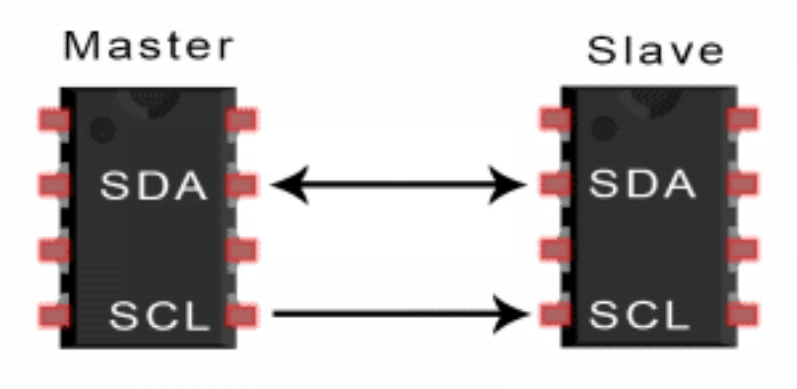


You'll probably find yourself using I2C if you ever build projects that use OLED displays, barometric pressure sensors, or gyroscope/accelerometer modules.

INTRODUCTION TO I2C COMMUNICATION

I2C combines the best features of SPI and UARTs. With I2C, you can connect multiple slaves to a single master (like SPI) and you can have multiple masters controlling single, or multiple slaves. This is really useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.

I2C only uses two wires to transmit data between devices:



SDA (Serial Data) – The line for the master and slave to send and receive data.

SCL (Serial Clock) – The line that carries the clock signal.

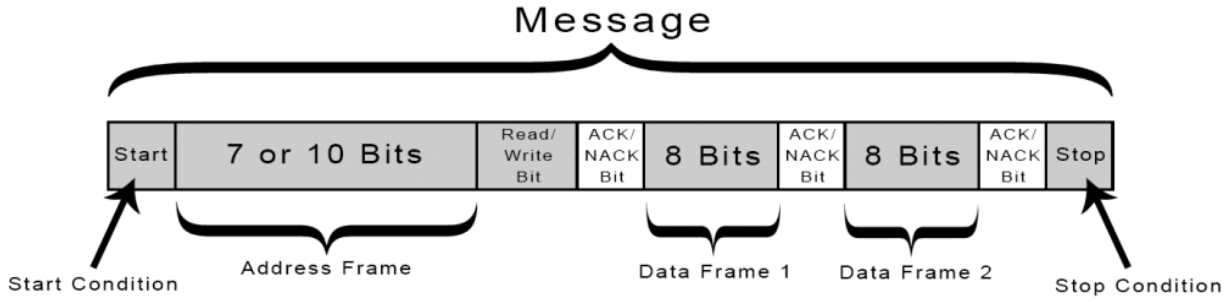
I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).

Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

HOW I2C WORKS

With I2C, data is transferred in *messages*. Messages are broken up into *frames* of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame:



Start Condition: The SDA line switches from a high voltage level to a low voltage level *before* the SCL line switches from high to low.
Stop Condition: The SDA line switches from a low voltage level to a high voltage level *after* the SCL line switches from low to high.
Address Frame: A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

ADDRESSING

I2C doesn't have slave select lines like SPI, so it needs another way to let the slave know that data is being sent to it, and not another slave. It does this by *addressing*. The address frame is always the first frame after the start bit in a new message.

The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage ACK bit back to the master. If the address doesn't match, the slave does nothing and the SDA line remains high.

READ/WRITE BIT

The address frame includes a single bit at the end that informs the slave whether the master wants to write data to it or receive data from it. If the master wants to send data to the slave, the read/write bit is a low voltage level. If the master is requesting data from the slave, the bit is a high voltage level.

THE DATA FRAME

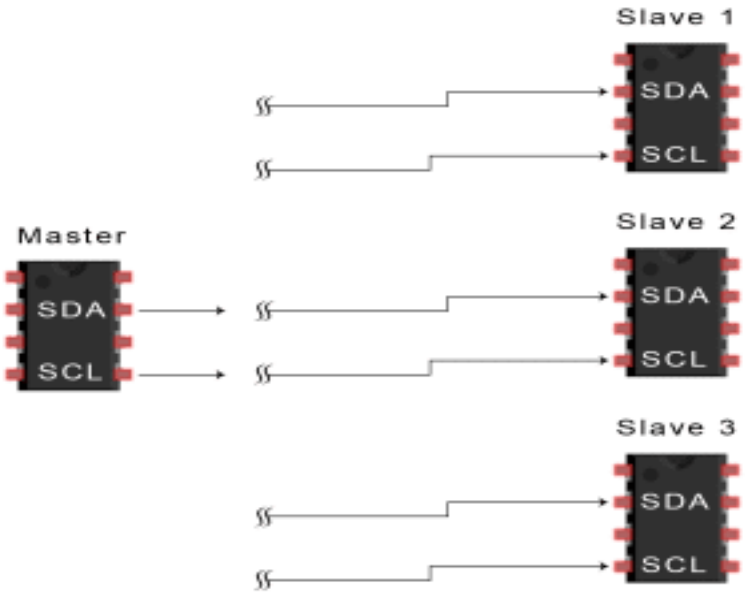
After the master detects the ACK bit from the slave, the first data frame is ready to be sent. The data frame is always 8 bits long, and sent with the most significant bit first. Each data frame is immediately followed by an ACK/NACK bit to verify that the frame has been received successfully.

The ACK bit must be received by either the master or the slave (depending on who is sending the data) before the next data frame can be sent.

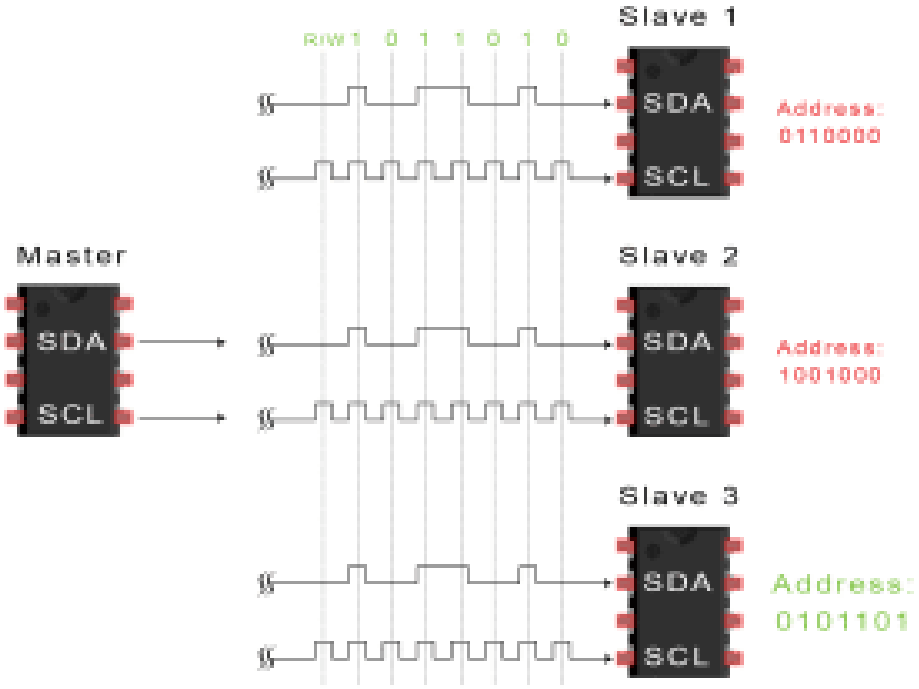
After all of the data frames have been sent, the master can send a stop condition to the slave to halt the transmission. The stop condition is a voltage transition from low to high on the SDA line after a low to high transition on the SCL line, with the SCL line remaining high.

STEPS OF I2C DATA TRANSMISSION

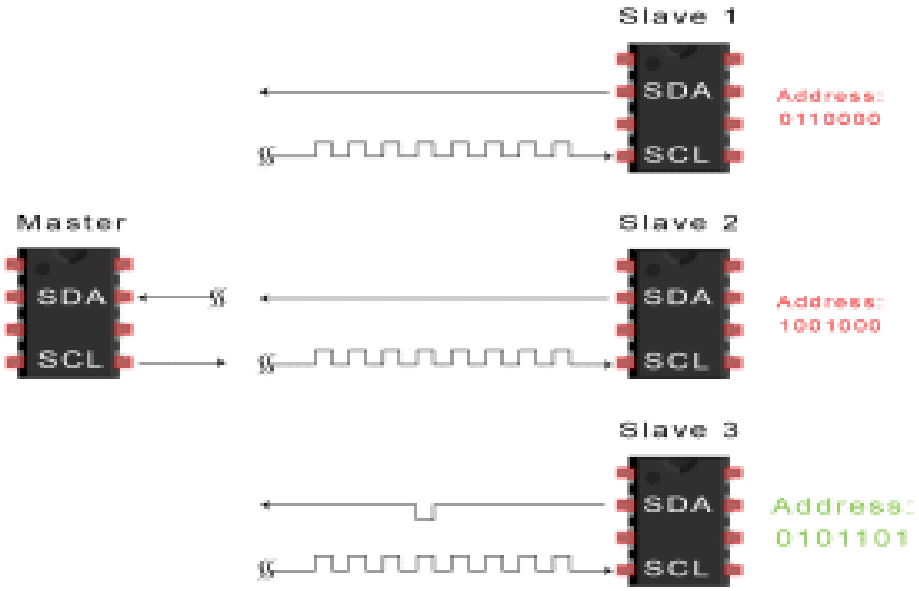
1. The master sends the start condition to every connected slave by switching the SDA line from a high voltage level to a low voltage level *before* switching the SCL line from high to low:



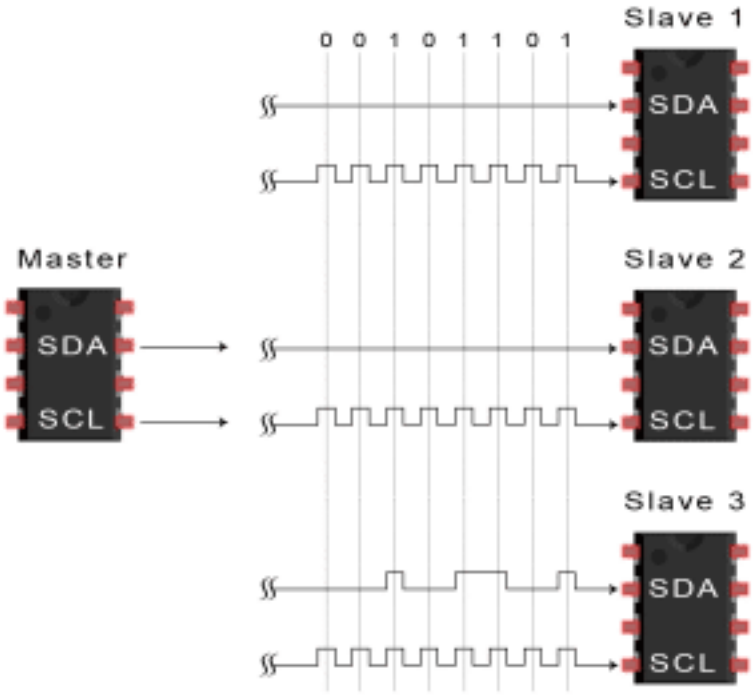
2. The master sends each slave the 7 or 10 bit address of the slave it wants to communicate with, along with the read/write bit:



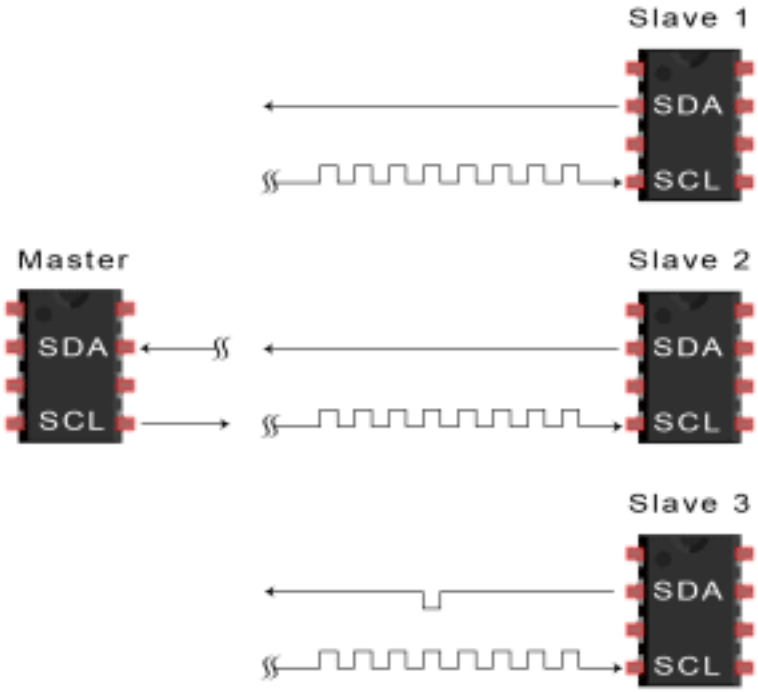
3. Each slave compares the address sent from the master to its own address. If the address matches, the slave returns an ACK bit by pulling the SDA line low for one bit. If the address from the master does not match the slave's own address, the slave leaves the SDA line high.



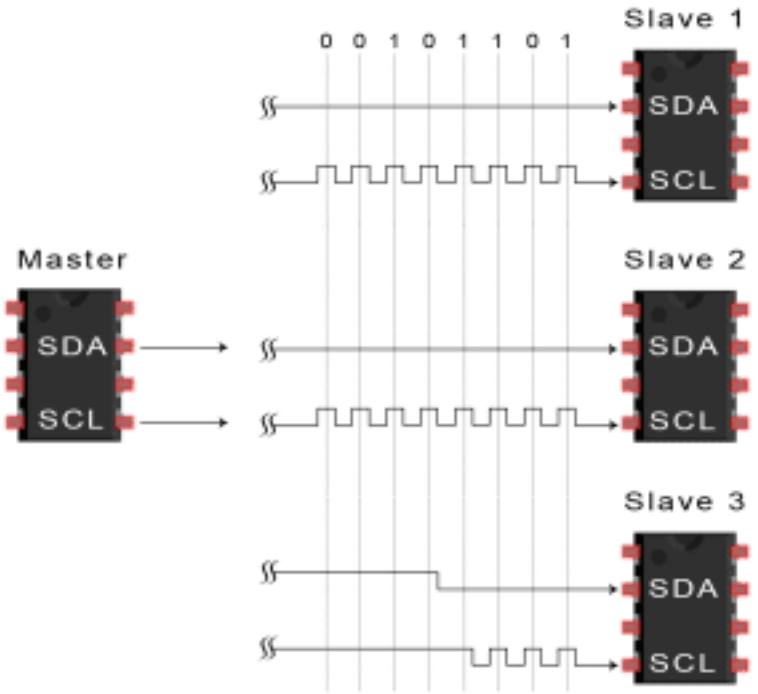
4. The master sends or receives the data frame:



5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame:

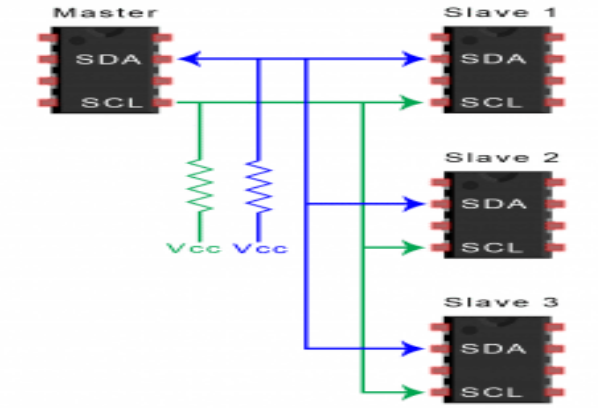


6. To stop the data transmission, the master sends a stop condition to the slave by switching SCL high before switching SDA high:



SINGLE MASTER WITH MULTIPLE SLAVES

Because I2C uses addressing, multiple slaves can be controlled from a single master. With a 7 bit address, 128 (2^7) unique address are available. Using 10 bit addresses is uncommon, but provides 1,024 (2^{10}) unique addresses. To connect multiple slaves to a single master, wire them like this, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc:



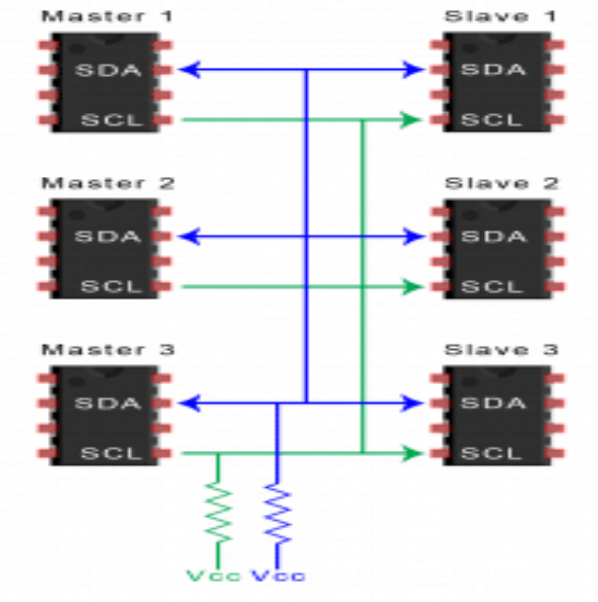
MULTIPLE MASTERS WITH MULTIPLE SLAVES

Multiple masters can be connected to a single slave or multiple slaves.

The problem with multiple masters in the same system comes when two masters try to send or receive data at the same time over the SDA line.

To solve this problem, each master needs to detect if the SDA line is low or high before transmitting a message.

If the SDA line is low, this means that another master has control of the bus, and the master should wait to send the message. If the SDA line is high, then it's safe to transmit the message. To connect multiple masters to multiple slaves, use the following diagram, with 4.7K Ohm pull-up resistors connecting the SDA and SCL lines to Vcc:



ADVANTAGES AND DISADVANTAGES OF I2C

There is a lot to I2C that might make it sound complicated compared to other protocols, but there are some good reasons why you may or may not want to use I2C to connect to a particular device:

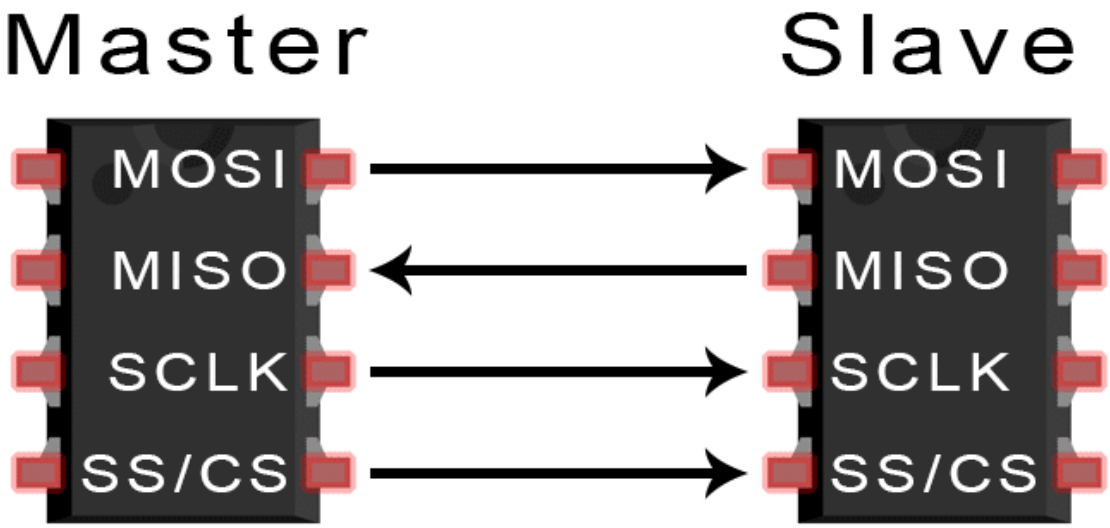
ADVANTAGES

- Only uses two wires
- Supports multiple masters and multiple slaves
- ACK/NACK bit gives confirmation that each frame is transferred successfully
- Hardware is less complicated than with UARTs
- Well known and widely used protocol

DISADVANTAGES

- Slower data transfer rate than SPI
- The size of the data frame is limited to 8 bits
- More complicated hardware needed to implement than SPI

BASICS OF THE SPI COMMUNICATION PROTOCOL



When you connect a microcontroller to a sensor, display, or other module, do you ever think about how the two devices talk to each other? What exactly are they saying? How are they able to understand each other?

Communication between electronic devices is like communication between humans. Both sides need to speak the same language. In electronics, these languages are called *communication protocols*. Luckily for us, there are only a few communication protocols we need to know when building most DIY electronics projects.

First, we'll begin with some basic concepts about electronic communication, then explain in detail how SPI works.

SPI, I2C, and UART are quite a bit slower than protocols like USB, ethernet, Bluetooth, and WiFi, but they're a lot more simple and use less hardware and system resources. SPI, I2C, and UART are ideal for communication between microcontrollers and between microcontrollers and sensors where large amounts of high speed data don't need to be transferred.

SERIAL VS. PARALLEL COMMUNICATION

Electronic devices talk to each other by sending *bits* of data through wires physically connected between devices.

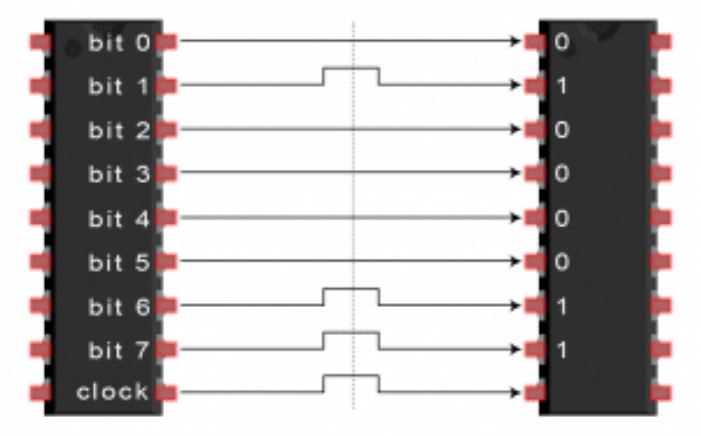
A bit is like a letter in a word, except instead of the 26 letters (in the English alphabet), a bit is binary and can only be a 1 or 0.

Bits are transferred from one device to another by quick changes in voltage.

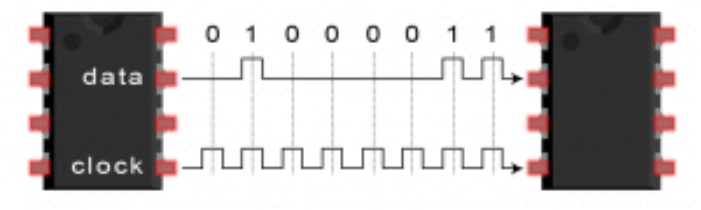
In a system operating at 5 V, a 0 bit is communicated as a short pulse of 0 V, and a 1 bit is communicated by a short pulse of 5 V.

The bits of data can be transmitted either in parallel or serial form.

In parallel communication, the bits of data are sent all at the same time, each through a separate wire. The following diagram shows the parallel transmission of the letter “C” in binary (01000011):



In serial communication, the bits are sent one by one through a single wire. The following diagram shows the serial transmission of the letter “C” in binary (01000011):

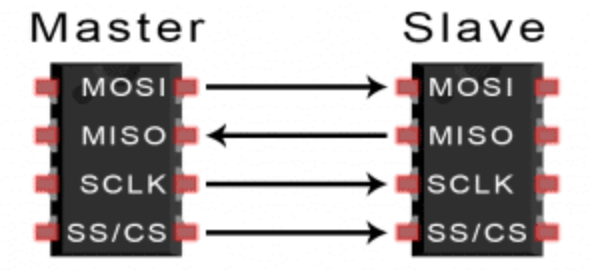


INTRODUCTION TO SPI COMMUNICATION

SPI is a common communication protocol used by many different devices. For example, SD card reader modules, RFID card reader modules, and 2.4 GHz wireless transmitter/receivers all use SPI to communicate with microcontrollers.

One unique benefit of SPI is the fact that data can be transferred without interruption. Any number of bits can be sent or received in a continuous stream. With I2C and UART, data is sent in packets, limited to a specific number of bits. Start and stop conditions define the beginning and end of each packet, so the data is interrupted during transmission.

Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instruction from the master. The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave (more on this below).



MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.
MISO (Master Input/Slave Output) – Line for the slave to send data to the master.
SCLK (Clock) – Line for the clock signal.
SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

Wires Used	4
Maximum Speed	Up to 10 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	1
Max # of Slaves	Theoretically unlimited*

*In practice, the number of slaves is limited by the load capacitance of the system, which reduces the ability of the master to accurately switch between voltage levels.

HOW SPI WORKS

THE CLOCK

The clock signal synchronizes the output of data bits from the master to the sampling of bits by the slave. One bit of data is transferred in each clock cycle, so the speed of data transfer is determined by the frequency of the clock signal. SPI communication is always initiated by the master since the master configures and generates the clock signal.

Any communication protocol where devices share a clock signal is known as *synchronous.* SPI is a synchronous communication protocol. There are also *asynchronous* methods that don't use a clock signal. For example, in UART communication, both sides are set to a pre-configured baud rate that dictates the speed and timing of data transmission.

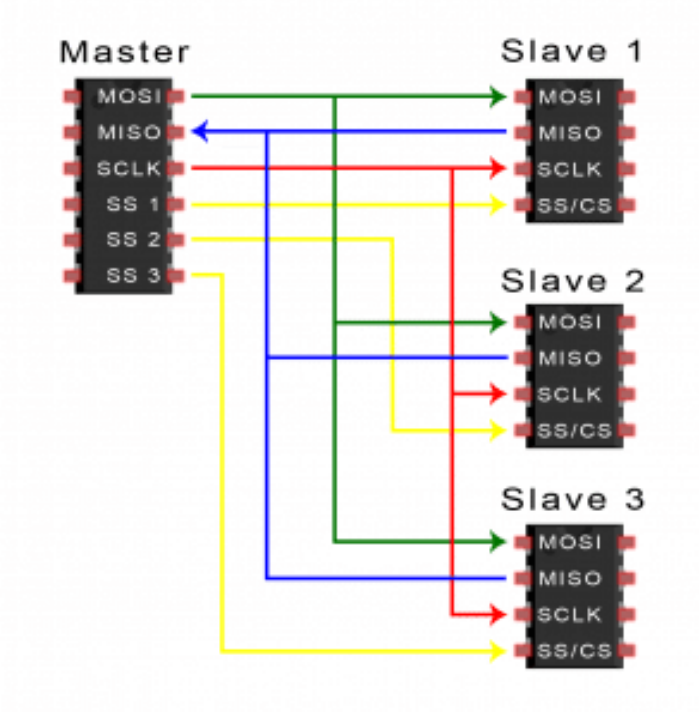
The clock signal in SPI can be modified using the properties of *clock polarity* and *clock phase*. These two properties work together to define when the bits are output and when they are sampled. Clock polarity can be set by the master to allow for bits to be output and sampled on either the rising or falling edge of the clock cycle. Clock phase can be set for output and sampling to occur on either the first edge or second edge of the clock cycle, regardless of whether it is rising or falling.

SLAVE SELECT

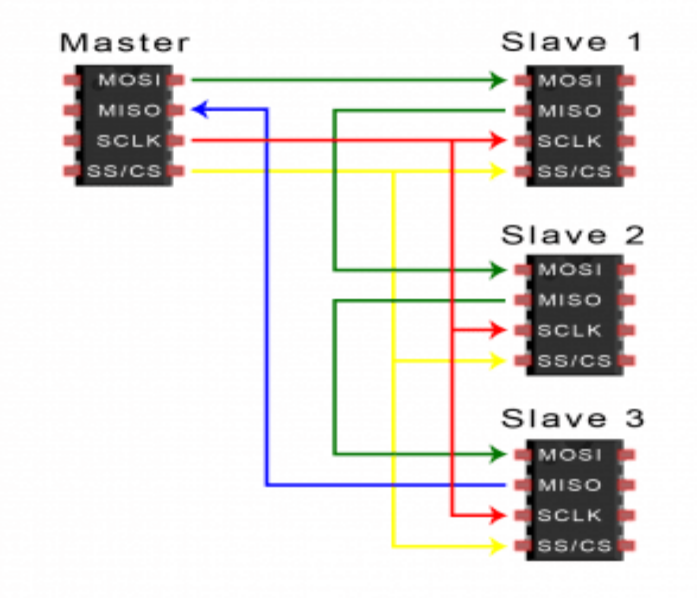
The master can choose which slave it wants to talk to by setting the slave's CS/SS line to a low voltage level. In the idle, non-transmitting state, the slave select line is kept at a high voltage level. Multiple CS/SS pins may be available on the master, which allows for multiple slaves to be wired in parallel. If only one CS/SS pin is present, multiple slaves can be wired to the master by daisy-chaining.

MULTIPLE SLAVES

SPI can be set up to operate with a single master and a single slave, and it can be set up with multiple slaves controlled by a single master. There are two ways to connect multiple slaves to the master. If the master has multiple slave select pins, the slaves can be wired in parallel like this:



If only one slave select pin is available, the slaves can be daisy-chained like this:



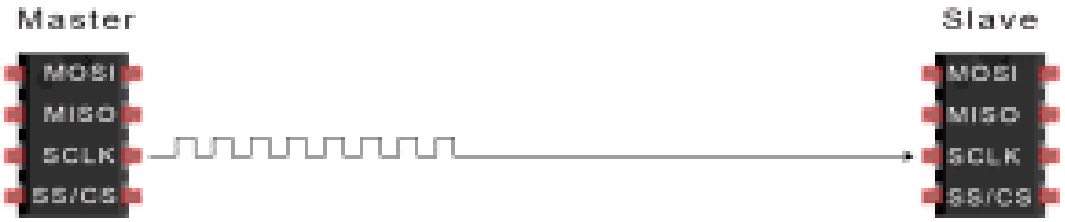
MOSI AND MISO

The master sends data to the slave bit by bit, in serial through the MOSI line. The slave receives the data sent from the master at the MOSI pin. Data sent from the master to the slave is usually sent with the most significant bit first.

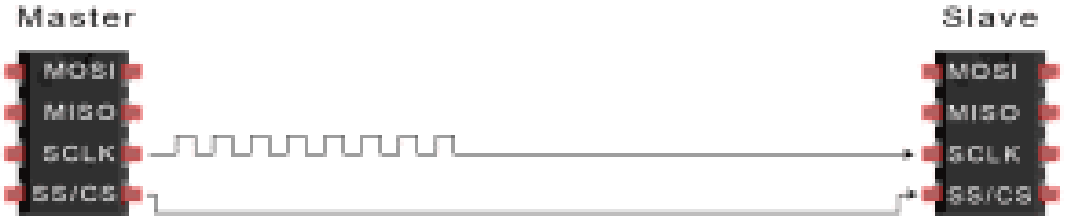
The slave can also send data back to the master through the MISO line in serial. The data sent from the slave back to the master is usually sent with the least significant bit first.

STEPS OF SPI DATA TRANSMISSION

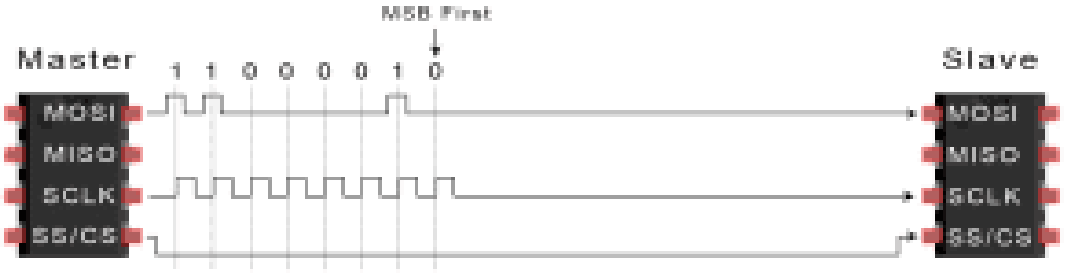
1. The master outputs the clock signal:



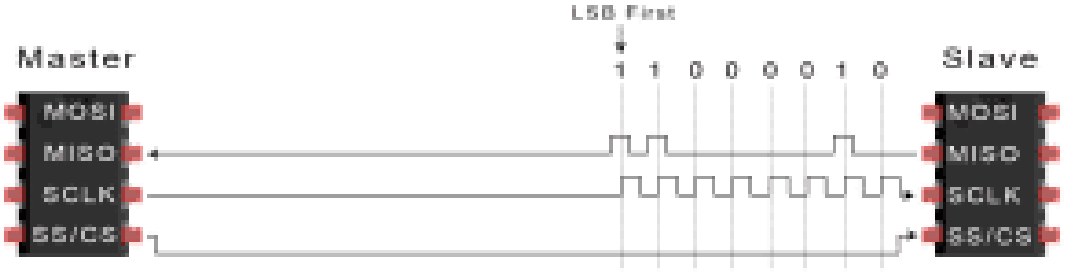
2. The master switches the SS/CS pin to a low voltage state, which activates the slave:



3. The master sends the data one bit at a time to the slave along the MOSI line. The slave reads the bits as they are received:



4. If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads the bits as they are received:



ADVANTAGES AND DISADVANTAGES OF SPI

There are some advantages and disadvantages to using SPI, and if given the choice between different communication protocols, you should know when to use SPI according to the requirements of your project:

ADVANTAGES

- No start and stop bits, so the data can be streamed continuously without interruption
- No complicated slave addressing system like I2C
- Higher data transfer rate than I2C (almost twice as fast)
- Separate MISO and MOSI lines, so data can be sent and received at the same time

DISADVANTAGES

- Uses four wires (I2C and UARTs use two)
- No acknowledgement that the data has been successfully received (I2C has this)
- No form of error checking like the parity bit in UART
- Only allows for a single master

Universal serial bus (USB) is defined as a standard that mentions the specifications used by cables, ports, and protocols that enable simple and universally accepted connectivity between a host and peripheral device. This article explains what USB is, the types of USB, and the importance of the technology.

What Is USB (Universal Serial Bus)?

USB, or universal serial bus, is a mechanism used to connect peripheral devices to computers. Before the advent of USB technology, a PC typically included one or two serial connections, a parallel port, keyboard and mouse connectors, and in some instances, a joystick port.

The USB standard was established in the mid-1990s by a number of American companies, notably IBM, Intel, and Microsoft Corporation, as a more straightforward way to connect computer peripherals.

The port provided a standard method for connecting various devices and offered considerable speed advantages over other alternatives.

Initially, USB technology acceptance was sluggish:

- Computer manufacturers were extremely slow to include the ports on their systems before USB-capable peripherals were available.
- Peripheral device manufacturers were similarly unwilling to promote USB products before USB ports were ubiquitous on newer computers.
- When the tech was initially launched, operating systems had relatively modest supporting capabilities.

The 1998 introduction of the first model of Apple Inc.'s iMac was a major breakthrough. By creating a popular and well-received machine with only USB connections, Apple forced other manufacturers to embrace the standard. Since then, the majority of peripheral devices, including printers, scanners, and keyboards, have all used USB. Even the creation of new technologies (like portable memory sticks popularly called USB storage, which eventually replaced floppy discs) was spurred by the acceptance of this standard.

USB design is now standardized by the USB Implementers Forum (USB-IF), which is composed of organizations that support and promote USB. Not only does the USBIF endorse the USB, but it also maintains the standard and enforces the compliance program.

How does USB (universal serial bus) work?

A computer's 'bus' is a network of cables transporting data between internal components or a computer and its external devices. It is a metaphor for an electronic busbar, which distributes electricity across big, energy-hungry locations such as factories or data centers.

Before the introduction of USB, any peripheral device was connected to a computer through a port of its own construction. Over the years, as the number of peripheral devices rose, a new standardized method of exchanging data from the primary host and a variety of devices was required. This eventually led to the invention of USB.

When a peripheral device is connected to a host computer by USB, the host machine will automatically determine the kind of device and install a driver that lets it function.
USB transmits data between two devices in smaller, bite-sized quantities known as “packets. Each packet transmits a predetermined amount of bytes (a digital information unit). This might include details such as the source and destination of the material, and any anomalies that may have been discovered.
The working of a USB can be broken down as follows:

- **Data transfer in bulk**: This type of transmission is employed by printers or digital scanners for huge quantities of data. It is generally a low-priority transmission and is not time-sensitive. The operation may be delayed if the host machine has many USB devices attached.
- **Small packets for critical device connections**: This feature is used by peripheral systems like keyboards and mice to transmit tiny quantities of data. These transmissions are frequently utilized for occasional but significant requirements. The remote device generates requests and waits for the host to enquire about the precise data it would need. The requests will be reattempted if the first request for transaction fails. Here, the USB would also inform you of any modifications in the device’s status.
- **Control packet transfers for USB management**: This data transmission is used to set up and manage USB devices. The host submits a request to the device, followed by data transmission. Control transfer is also used for status checks. At any one moment, only one control request can be processed.
- **Real-time, uninterrupted data transfer in the isochronous mode**: Isochronous transfer is used for audio, video, and other real-time data. During the transmission, errors may occur, but the transmission will not be paused to resubmit the packets. However, such transfers often include instances in which the fidelity of the data is not crucial, such as the transmission of audio parts in a Voice over Internet Protocol (VoIP) call that the listener may not detect. It is better to exclude these components than to resend data, which might result in audio glitches.

As the number of devices increases, the host maintains a record of the total amount of bandwidth requested by all isochronous and interrupt devices. Together, they may take up to 90% of the available bandwidth, which is 480 megabits per second for standard USB or 4.8 gigabits per second with USB 3.0.

The host prevents access to any further isochronous or interrupt devices once 90% has been consumed. Control packets and bulk transfer packets use any remaining bandwidth (which is always 10% at a minimum). Beyond this threshold, you cannot have multiple USB connections transferring data simultaneously.

How do USB cables work?

The USB port is a standard interface for connecting cables to PCs and consumer electronics devices. Users can connect a specially designed wire called the USB cable to this port. One end of the cable connects to the host and the other to the peripheral, and depending on the type of USB, the two ends may or may not be symmetrical.

USB cables may transmit both power and information. To accomplish this, any USB cable has two types of wires. One set transports current, while the other transmits data signals.

There are four metal strips inside the conventional USB 2.0 connection. The two outermost strips are the power supply's positive and negative terminals. The two center strips are designated for data transmission. With the latest USB 3.0 connection, the inclusion of additional data-carrying strips increases the data transmission rate. Four added signaling lines enable USB 3.0's superior speed.

Key features of USB

Universal serial bus connections (enabled by USB ports and USB cables) support the following:

1. **Hot swapping**

Hot swapping is one of the major characteristics of the USB. This functionality enables the removal or replacement of a device without requiring a system restart or interruption.

The PC has to be rebooted when installing or uninstalling a device from an older port. Electrostatic discharge (ESD), an unintended electrical current capable of inflicting extensive damage to fragile electronic devices, was initially averted by rebooting. With USB, this is not necessary. Hot swapping is fault-ruggedized, meaning it can continue functioning despite hardware failure.

2. **Direct current transfer**

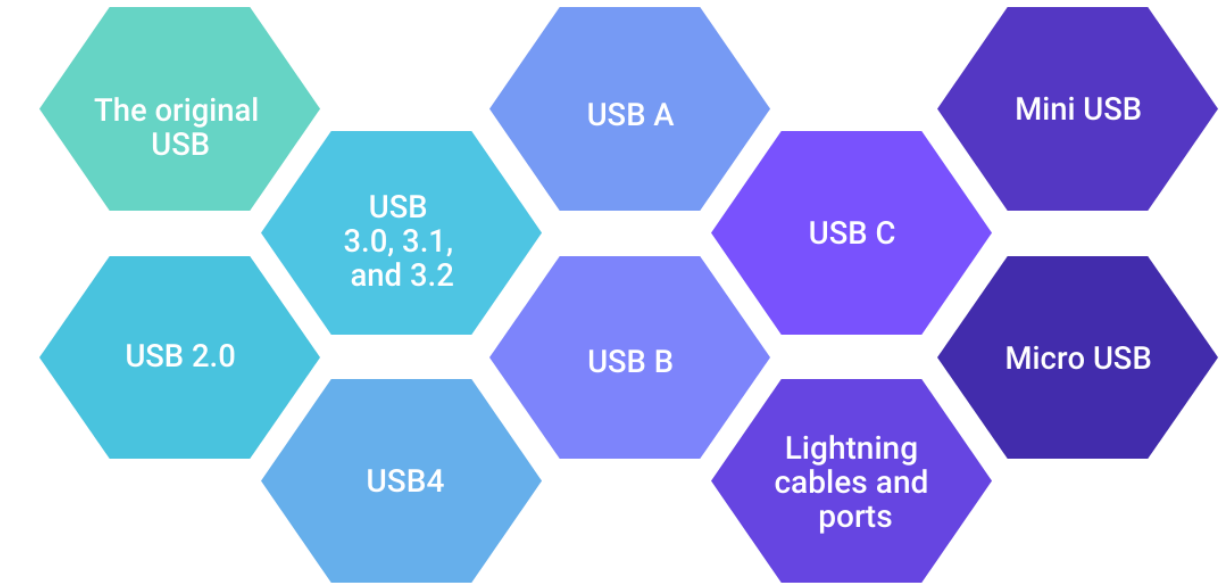
Another aspect of USB is the usage of direct current (DC). Several devices are connected to a DC current through a USB power connection but do not communicate data. Notable examples include USB speakers, small refrigerators, keyboard lamps, and even USB-based device chargers.

3. **The use of multiple contact points**

All USB connections feature at least four contacts used for power, ground, and two data wires (D + and D -). USB 3.0 connectors and above have five contacts. The USB connection is intended to transmit 5V at a maximum current of 500mA. The USB connection may be inserted in only one direction. It is feasible to force an incorrect connection. However, this could result in device damage.

4. **Shielding and protection**

The USB connection is insulated, providing a metal casing that is not part of the circuit. This characteristic is crucial for maintaining the signal's integrity in an electrically "busy" environment. All USB cables are wrapped in plastic at the connection end to avoid damage to the cable and electrical connection.



10 Types of USB

USB can be classified based on four generations of development or the type of cable connections they use. Let us first look at the four versions of USB that have developed over the years.

1. The original USB

Before USBs, systems had serial and parallel ports, as previously explained. Many computer manufacturers, like Intel, Microsoft, and Apple, worked together to develop a universal device that connected multiple external sources to a PC without requiring a reboot. This was the first generation, as available in the 90s.

2. USB 2.0

In 2000, USB 2.0 achieved widespread dominance. The transfer speed was the most significant difference between version 2.0 and version 1.1. The USB 2.0 delivered data at a rate of 480 megabits per second, which is forty times quicker than the USB 1.1. In addition to its significantly enhanced speeds, USB 2.0 was configurable with any USB 1.1 port and vice versa.

3. USB 3.0, 3.1, and 3.2

USB 3.0 was the first device of its kind to transport high-definition video efficiently. It was also known as SuperSpeed USB and had transfer speeds of 5 gigabits per second or approximately 5,120 megabits per second. However, similar to USB 1.0, USB 3.0 could not catch on, and its successors found much more acceptance.

Macbooks and Chromebooks were among the first laptops to adopt the 2014-introduced USB 3.1. The maximum transmission rate of USB 3.1 was 10 gigabits per second. Exactly three years after this, USB 3.2 was launched with transmission rates of 20 gigabits per second, two times faster than USB 3.1. Only these USBs are compatible with Type C connections. These ports are full-duplex, meaning that data may be sent in both directions since USB is symmetrical.

4. USB4

USB4 (also known as USB 4.0) is a 2019 standard issued by the USB-IF in version 1.0. The USB4 protocol is built on the Thunderbolt 3 standard, which Intel Corporation submitted to the USB-IF. Its design may dynamically utilize a single high-speed connection with numerous end-device types, executing each transfer according to its data or application type.

The first products compatible with USB4 were Intel's Tiger Lake processors, with more devices appearing around the end of 2020. This type of USB connection is yet to go mainstream.

Another way to understand the different types of USB is by classifying them based on the nature of the cable connection and physical design. Here are the key types of USB to note, continuing the previous list:

5. USB A

USB type-A connections, often known as type-A connectors, are among the most prevalent USB connectors. Consequently, they are also known as standard A connections. If laptops or desktop computers include a USB port (where USB devices may be plugged in), the port will likely be type A. Note that most recent Apple laptops (Macs) lack USB type-A connections.

6. USB B

Type B is square-shaped and smaller than type A. This is less prevalent than type-A; however, it may be found on computer components such as scanners, printers, external disk drives, etc. This type is mainly suitable for bulk data transfer connections. USB type-B is rarely found on laptops but may be prevalent in older desktop workstations, servers, and mainframes.

7. USB C

USB type-C is probably the most ubiquitous USB connection currently available. It fits readily into the tiniest peripherals we use today, such as cellphones, Bluetooth speakers, etc., due to its compact size. One of the numerous benefits of type-C over other current variations is that it supports "reverse plug orientation," meaning that its plug may be inserted without regard to its orientation.

This means that you may charge your phone through your laptop or vice versa using type-C's bi-directional power supply. The European Parliament Internal Market Committee (IMCO) has

also confirmedOpens a new window that all devices launched in the EU will be mandatorily required to have USB-C charging capabilities.

8. Lightning cables and ports (also built on USB technology)

Modern Apple devices, such as the iPhone and iPad, often include a separate USB connector: the Lightning cord. On one end of the cable is a narrow, rectangular connection, and on the other is a Type C connector. Similar to USB C, it is reversible or symmetrical, meaning it may be inserted either way. However, due to its design, the lightning connection cannot be utilized in any product or gadget other than Apple products.

9. Mini USB

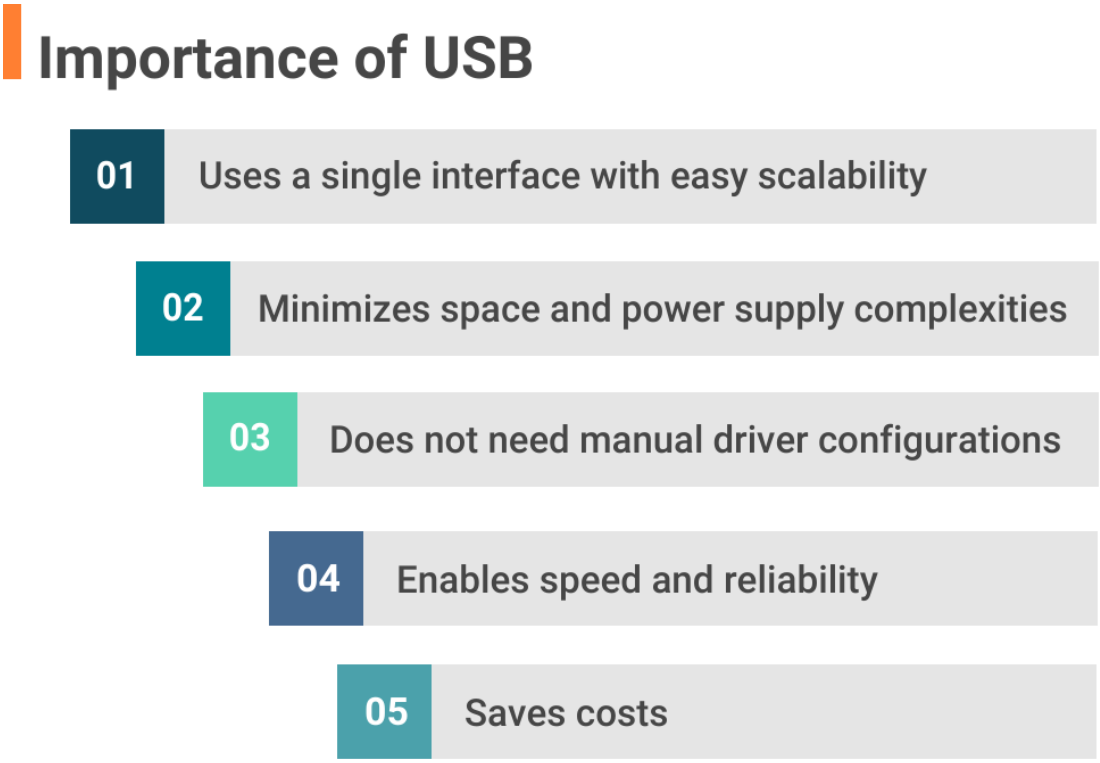
This type of USB optimizes USB A or B connections for slightly older, portable devices. These are the miniature versions of the Type A and Type B USB connections. USB Mini is often found in portable cameras, gaming controllers, and older cell phones.

10. Micro USB

Again, the micro variant optimizes USB A or B for mobile devices – although for relatively newer ones. With the introduction of USB Type C, micro-USBs are now being phased out of newer high-end smartphone models. However, micro USB is still commonly used in inexpensive cell phones and other electronic devices, such as headphones, around the globe.
USB Micro B SuperSpeed is an additional variation of the micro USB standard. As the name indicates, this allows quicker data transmission than standard micro USB B ports. Therefore, these connections are often seen on external disk drives, where massive data transmission occurs frequently.

Importance of USB Technology

For various reasons, USB is a foundational technology in the modern digital era for consumers and businesses alike.



Importance of USB

1. Uses a single interface with easy scalability

USB's adaptability eliminates the need for separate connection types and hardware specifications for each accessory. Additionally, the majority of personal computers feature three to four USB ports. If more USB ports are needed, USB hubs can be employed to add on external ports.

2. Minimizes space and power supply complexities

In comparison to older connections, USB sockets are smaller in size. Additionally, the USB interface was created right from the outset to function as a DC power source. Through its USB connector, any host device may provide the peripheral with 5V DC — between 500 mA (USB 1.0 and 2.0) and 900 mA (USB 3.0).

3. Does not need manual driver configurations

The host device's operating system must only install the USB device driver once. After that, when the peripheral device is connected, the driver is immediately launched to set up the peripheral device. Typically, the device driver for any USB peripheral is loaded automatically the first time the peripheral is connected to the host.

4. Enables speed and reliability

USB offers multiple speed settings, making it more productivity-friendly and efficient. It provides speeds between 1.5Mbit/s to 5Gbit/s. 2013 saw the release of USB 3.1, which boosted the speed to 10Gbit/s. In addition, the USB protocol may detect data transmission errors and tell the transmitter to resend the data.

5. Saves costs

Due to the scalability of the manufacturing process and USB's adaptability and popularity, it is now affordable to produce USB-supported products. Consequently, the components, connections, and cables are widely accessible and inexpensive. During "suspend mode," the peripheral uses less than 500 microamperes for USB 2.0 and less than 2.5 milliamperes for USB 3.0, drastically reducing expenditures.