

GPS Signal

Contents

Articles

Serial communication	1
RS-232	3
Serial port	12
Universal asynchronous receiver/transmitter	19
USB	26

References

Article Sources and Contributors	55
Image Sources, Licenses and Contributors	57

Article Licenses

License	58
---------	----

Serial communication

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 (no need for serializer and deserializer, or SerDes) and to outstrip its disadvantages (clock skew, interconnect density). The migration from PCI to PCI Express is an example.

Cables that carry serial data

Main article: data cable

Many serial communication systems were originally designed to transfer data over relatively large distances through some sort of data cable.

The term "serial" most often refers to the RS232 port on the back of the original IBM PC, often called "the" serial port, and "the" serial cable designed to plug into it, and the many devices designed to be compatible with it.

Practically all long-distance communication transmits data one bit at a time, rather than in parallel, because it reduces the cost of the cable. The cables that carry this data (other than "the" serial cable) and the computer ports they plug into are usually referred to with a more specific name, to reduce confusion.

Keyboard and mouse cables and ports are almost invariably serial -- such as PS/2 port and Apple Desktop Bus and USB.

The cables that carry digital video are almost invariably serial -- such as coax cable plugged into a HD-SDI port, a webcam plugged into a USB port or Firewire port, Ethernet cable connecting an IP camera to a Power over Ethernet port, FPD-Link, etc.

Other such cables and ports, transmitting data one bit at a time, include Serial ATA, Serial SCSI, Ethernet cable plugged into Ethernet ports, the Display Data Channel using previously reserved pins of the VGA connector or the DVI port or the HDMI port.

Serial buses

Many communication systems were generally originally designed to connect two integrated circuits on the same printed circuit board, connected by signal traces on that board (rather than external cables).

Integrated circuits are more expensive when they have more pins. To reduce the number of pins in a package, many ICs use a serial bus to transfer data when speed is not important. Some examples of such low-cost serial buses include SPI, I²C, UNI/O, and 1-Wire.

Serial versus parallel

The communication links across which computers—or parts of computers—talk to one another may be either serial or parallel. A parallel link transmits several streams of data simultaneously along multiple channels (e.g., wires, printed circuit tracks, or optical fibres); a serial link transmits a single stream of data.

Although a serial link may seem inferior to a parallel one, since it can transmit less data per clock cycle, it is often the case that serial links can be clocked considerably faster than parallel links in order to achieve a higher data rate.

A number of factors allow serial to be clocked at a higher rate:

- Clock skew between different channels is not an issue (for unclocked asynchronous serial communication links).
- A serial connection requires fewer interconnecting cables (e.g., wires/fibres) and hence occupies less space. The extra space allows for better isolation of the channel from its surroundings.
- Crosstalk is less of an issue, because there are fewer conductors in proximity.

In many cases, serial is a better option because it is cheaper to implement. Many ICs have serial interfaces, as opposed to parallel ones, so that they have fewer pins and are therefore less expensive.

Examples of serial communication architectures

- Morse code telegraphy
 - RS-232 (low-speed, implemented by serial ports)
 - RS-422
 - RS-423
 - RS-485
 - I²C
 - 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
-

External links

- Serial Interface Tutorial for Robotics ^[1] (contains many practical examples)
- Serial interfaces listing (with pinouts) ^[2]
- Wiki: Serial Ports ^[3]
- Visual studio 2008 coding for Serial communication ^[4]
- Introduction to I²C and SPI protocols ^[5]
- Serial communication introduction ^[6]

References

- [1] http://www.societyofrobots.com/microcontroller_uart.shtml
- [2] http://pinouts.ru/pin_SerialPorts.shtml
- [3] <http://c2.com/cgi/wiki?SerialPorts>
- [4] <http://www.thaiio.com/prog-cgi/VBnetSerialPort.htm>
- [5] <http://www.byteparadigm.com/kb/article/AA-00255>
- [6] <https://learn.sparkfun.com/tutorials/serial-communication/all>

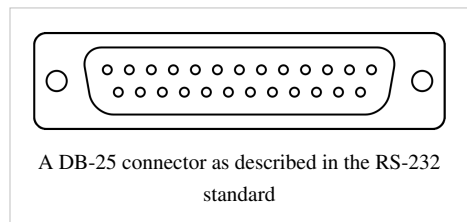
RS-232

This article is about the RS-232 standard. For RS-232 variants, see serial port.

"V.24" redirects here. For other uses, see V24 (disambiguation).

In telecommunications, **RS-232** is a standard for serial communication transmission of data. It formally defines the signals connecting between a *DTE* (*data terminal equipment*) such as a computer terminal, and a *DCE* (*data circuit-terminating equipment*, originally defined as *data communication equipment*), such as a modem. The RS-232 standard is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors. The current version of the standard is *TIA-232-F Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, issued in 1997.

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, RS-232 is hampered by low transmission speed, large voltage swing, and large standard connectors. In modern personal computers, USB has displaced RS-232 from most of its peripheral interface roles. Many computers do not come equipped with RS-232 ports and must use either an external USB-to-RS-232 converter or an internal expansion card with one or more serial ports to connect to RS-232 peripherals. RS-232 devices are still found, especially in industrial machines, networking equipment, and scientific instruments.



Scope of the standard

The Electronic Industries Association (EIA) standard RS-232-C as of 1969 defines:

- Electrical signal characteristics such as voltage levels, signaling rate, timing and slew-rate of signals, voltage withstand level, short-circuit behavior, and maximum load capacitance.
- Interface mechanical characteristics, pluggable connectors and pin identification.
- Functions of each circuit in the interface connector.
- Standard subsets of interface circuits for selected telecom applications.

The standard does not define such elements as the character encoding or the framing of characters, or error detection protocols. The character format and transmission bit rate are set by the serial port hardware which may also contain circuits to convert the internal logic levels to RS-232 compatible signal levels. The standard does not define bit rates for transmission, except that it says it is intended for bit rates lower than 20,000 bits per second.

History

RS-232 was first introduced in 1962 by the *Radio Sector* of the EIA.^[1] The original DTEs were electromechanical teletypewriters, and the original DCEs were (usually) modems. When electronic terminals (smart and dumb) began to be used, they were often designed to be interchangeable with teletypewriters, and so supported RS-232. The C revision of the standard was issued in 1969 in part to accommodate the electrical characteristics of these devices. Wikipedia:Citation needed

Since the requirements of devices such as computers, printers, test instruments, POS terminals and so on were not foreseen by the standard, designers implementing an RS-232 compatible interface on their equipment often interpreted the standard idiosyncratically. The resulting common problems were non-standard pin assignment of circuits on connectors, and incorrect or missing control signals. The lack of adherence to the standards produced a thriving industry of breakout boxes, patch boxes, test equipment, books, and other aids for the connection of disparate equipment. A common deviation from the standard was to drive the signals at a reduced voltage. Some manufacturers therefore built transmitters that supplied +5 V and -5 V and labeled them as "RS-232 compatible". Wikipedia:Citation needed

Later personal computers (and other devices) started to make use of the standard so that they could connect to existing equipment. For many years, an RS-232-compatible port was a standard feature for serial communications, such as modem connections, on many computers. It remained in widespread use into the late 1990s. In personal computer peripherals, it has largely been supplanted by other interface standards, such as USB. RS-232 is still used to connect older designs of peripherals, industrial equipment (such as PLCs), console ports and special purpose equipment.

The standard has been renamed several times during its history as the sponsoring organization changed its name, and has been variously known as EIA RS-232, EIA 232, and most recently as TIA 232. The standard continued to be revised and updated by the Electronic Industries Alliance and since 1988 by the Telecommunications Industry Association (TIA). Revision C was issued in a document dated August 1969. Revision D was issued in 1986. The current revision is *TIA-232-F Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, issued in 1997. Changes since Revision C have been in timing and details intended to improve harmonization with the CCITT standard V.24, but equipment built to the current standard will interoperate with older versions. Wikipedia:Citation needed

Related ITU-T standards include V.24 (circuit identification) and V.28 (signal voltage and timing characteristics). Wikipedia:Citation needed

In revision D of EIA-232, the D-subminiature connector was formally included as part of the standard (it was only referenced in the appendix of RS 232 C). The voltage range was extended to +/- 25 volts, and the circuit capacitance limit was expressly stated as 2500 pF. Revision E of EIA 232 introduced a new, smaller, standard D-shell 26-pin

"Alt A" connector, and made other changes to improve compatibility with CCITT standards V.24, V.28 and ISO 2110.^[2]

Limitations of the standard

Because RS-232 is used beyond the original purpose of interconnecting a terminal with a modem, successor standards have been developed to address the limitations. Issues with the RS-232 standard include:

- The large voltage swings and requirement for positive and negative supplies increases power consumption of the interface and complicates power supply design. The voltage swing requirement also limits the upper speed of a compatible interface.
- Single-ended signaling referred to a common signal ground limits the noise immunity and transmission distance.
- Multi-drop connection among more than two devices is not defined. While multi-drop "work-arounds" have been devised, they have limitations in speed and compatibility.
- Asymmetrical definitions of the two ends of the link make the assignment of the role of a newly developed device problematic; the designer must decide on either a DTE-like or DCE-like interface and which connector pin assignments to use.
- The handshaking and control lines of the interface are intended for the setup and takedown of a dial-up communication circuit; in particular, the use of handshake lines for flow control is not reliably implemented in many devices.
- No method is specified for sending power to a device. While a small amount of current can be extracted from the DTR and RTS lines, this is only suitable for low power devices such as mice.
- The 25-way connector recommended in the standard is large compared to current practice.

Role in modern personal computers

Main article: Serial port

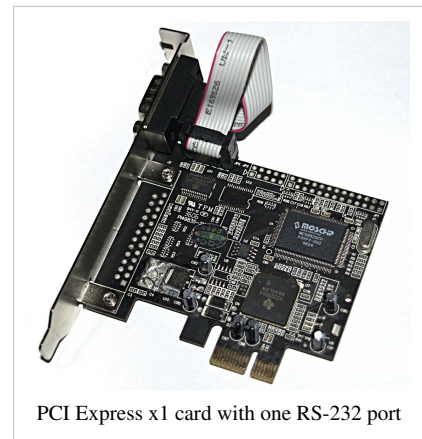
In the book *PC 97 Hardware Design Guide*, Microsoft deprecated support for the RS-232 compatible serial port of the original IBM PC design. Today, RS-232 has mostly been replaced in personal computers by USB for local communications. Compared with RS-232, USB is faster, uses lower voltages, and has connectors that are simpler to connect and use. However, USB is limited by standard to no more than 5 meters of cable, thus favoring RS-232 when longer distances are needed. Both standards have software support in popular operating systems.

USB is designed to make it easy for device drivers to communicate with hardware. USB is more complex than the RS-232 standard because it includes a protocol for transferring data to devices. This requires more software to support the protocol used. There is no direct analog to the terminal programs used to let users communicate directly with serial ports.

Serial ports of personal computers are also sometimes used to directly control various hardware devices, such as relays or lamps, since the control lines of the interface can be easily manipulated by software. Personal computers may use a serial port to interface to devices such as uninterruptible power supplies. In some cases, serial data is not exchanged, but the control lines are used to signal conditions such as loss of power or low battery alarms. A USB interface requires a device that can decode the serial data.

Devices that convert between USB and RS-232 do not work with all software or on all personal computers.

In fields such as laboratory automation or surveying, RS 232 devices may continue to be used. PLCs, VFDs, servo drives, and CNC equipment are programmable via RS-232. Some manufacturers have responded to this demand:



PCI Express x1 card with one RS-232 port

Toshiba re-introduced the DE-9M connector on the Tecra laptop.

Serial ports with RS-232 are also commonly used to communicate to headless systems such as servers, where no monitor or keyboard is installed, during boot when operating system is not running yet and therefore no network connection is possible. An RS-232 serial port can communicate to some embedded systems such as routers as an alternative to network mode of monitoring.

Standard details

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, that 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 standard does not define character framing within the data stream, or character encoding.

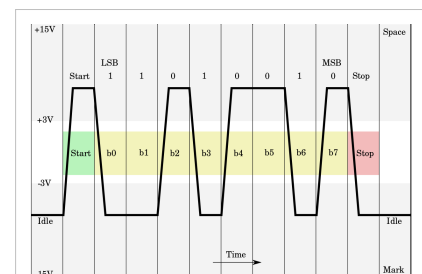
Voltage levels

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 or the range -3 to -15 volts with respect to the ground/common pin; consequently, 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". Control signals have the opposite polarity: the asserted or active state is positive voltage and the deasserted or inactive state is negative voltage. Examples of control lines include request to send (RTS), clear to send (CTS), data terminal ready (DTR), and data set ready (DSR).

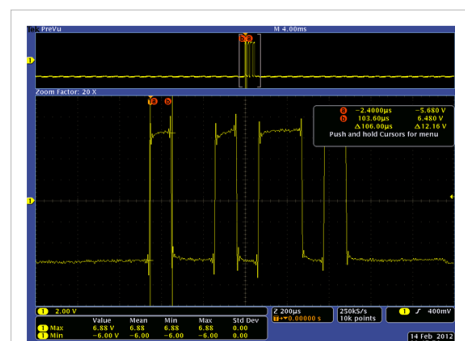
The standard specifies a maximum open-circuit voltage of 25 volts; signal levels of ± 5 V, ± 10 V, ± 12 V, and ± 15 V are all commonly seen depending on the voltages available to the line driver circuit. Some RS-232 driver chips have inbuilt circuitry to produce the required voltages from a 3 or 5 volt supply. RS-232 drivers and receivers must be able to withstand indefinite short circuit to ground or to any voltage level up to ± 25 volts. The slew rate, or how fast the signal changes between levels, is also controlled.

Because the voltage levels are higher than logic levels typically used by integrated circuits, special intervening driver circuits are required to translate logic levels. These also protect the device's internal circuitry from short circuits or transients that may appear on the RS-232 interface, and provide sufficient current to comply with the slew rate requirements for data transmission.

Because both ends of the RS-232 circuit depend on the ground pin being zero volts, problems will occur when connecting machinery and computers where the voltage between the ground pin on one end, and the ground pin on the other is not zero. This may also cause a hazardous ground loop. Use of a common ground limits RS-232 to



Diagrammatic oscilloscope trace of voltage levels for an ASCII "K" character (0x4B) with 1 start bit, 8 data bits, 1 stop bit. This is typical for start-stop communications, but the standard does not dictate a character format or bit order.



RS-232 data line on the terminals of the receiver side (Rx) probed by an oscilloscope (for an ASCII "K" character (0x4B) with 1 start bit, 8 data bits, 1 stop bit and no parity bits).

applications with relatively short cables. If the two devices are far enough apart or on separate power systems, the local ground connections at either end of the cable will have differing voltages; this difference will reduce the noise margin of the signals. Balanced, differential, serial connections such as USB, RS-422 and RS-485 can tolerate larger ground voltage differences because of the differential signaling.

Unused interface signals terminated to ground will have an undefined logic state. Where it is necessary to permanently set a control signal to a defined state, it must be connected to a voltage source that asserts the logic 1 or logic 0 level, for example with a pullup resistor. Some devices provide test voltages on their interface connectors for this purpose.

Connectors

RS-232 devices may be classified as Data Terminal Equipment (DTE) or Data Communication Equipment (DCE); this defines at each device which wires will be sending and receiving each signal. The standard recommended but did not make mandatory the D-subminiature 25-pin connector. According to the standard, male connectors have DTE pin functions, and female connectors have DCE pin functions. Other devices may have any combination of connector gender and pin definitions. Many terminals were manufactured with female connectors but were sold with a cable with male connectors at each end; the terminal with its cable satisfied the recommendations in the standard. The standard specifies 20 different signal connections. Since most devices use only a few signals, smaller connectors can often be used.

Personal computer manufacturers replaced the DB-25M connector by the smaller DE-9M connector. Different pin numbers were used for the signals (for this see serial port). This connector, with varying pinouts, became common for personal computers and related devices.

Presence of a 25-pin D-sub connector does not necessarily indicate an RS-232-C compliant interface. For example, on the original IBM PC, a male D-sub was an RS-232-C DTE port (with a non-standard current loop interface on reserved pins), but the female D-sub connector on the same PC model was used for the parallel Centronics printer port. Some personal computers put non-standard voltages or signals on some pins of their serial ports.

Signals

The following table lists commonly used RS-232 signals and pin assignments. See serial port (pinouts) for non-standard variations including the popular DE-9 connector.

Signal			Origin		DB-25 pin
Name	Typical purpose	Abbreviation	DTE	DCE	
Data Terminal Ready	Indicates presence of DTE to DCE.	DTR	●		20
Data Carrier Detect	DCE is connected to the telephone line.	DCD		●	8
Data Set Ready	DCE is ready to receive commands or data.	DSR		●	6
Ring Indicator	DCE has detected an incoming ring signal on the telephone line.	RI		●	22
Request To Send	DTE requests the DCE prepare to receive data.	RTS	●		4
Clear To Send	Indicates DCE is ready to accept data.	CTS		●	5
Transmitted Data	Carries data from DTE to DCE.	TxD	●		2
Received Data	Carries data from DCE to DTE.	RxD		●	3
Common Ground		GND	common		7
Protective Ground		PG	common		1

The signals are named from the standpoint of the DTE. The ground signal is a common return for the other connections. The DB-25 connector includes a second "protective ground" on pin 1.

Data can be sent over a secondary channel (when implemented by the DTE and DCE devices), which is equivalent to the primary channel. Pin assignments are described in following table:

Signal	Pin
Common Ground	7 (same as primary)
Secondary Transmitted Data (STD)	14
Secondary Received Data (SRD)	16
Secondary Request To Send (SRTS)	19
Secondary Clear To Send (SCTS)	13
Secondary Carrier Detect (SDCD)	12

Ring Indicator' (RI), is a signal sent from the modem to the terminal device. It indicates to the terminal device that the phone line is ringing. In many computer serial ports, a hardware interrupt is generated when the RI signal changes state. Having support for this hardware interrupt means that a program or operating system can be informed of a change in state of the RI pin, without requiring the software to constantly "poll" the state of the pin. RI is a one-way signal from the modem to the terminal (or more generally, the DCE to the DTE) that does not correspond to another signal that carries similar information the opposite way.

On an external modem the status of the Ring Indicator pin is often coupled to the "AA" (auto answer) light, which flashes if the RI signal has detected a ring. The asserted RI signal follows the ringing pattern closely, which can permit software to detect distinctive ring patterns.

The Ring Indicator signal is used by some older uninterruptible power supplies (UPS's) to signal a power failure state to the computer.

Certain personal computers can be configured for wake-on-ring, allowing a computer that is suspended to answer a phone call.

Cables

Main article: Serial cable

The standard does not define a maximum cable length but instead defines the maximum capacitance that a compliant drive circuit must tolerate. A widely used rule of thumb indicates that cables more than 50 feet (15 m) long will have too much capacitance, unless special cables are used. By using low-capacitance cables, full speed communication can be maintained over larger distances up to about 1,000 feet (300 m). For longer distances, other signal standards are better suited to maintain high speed.

Since the standard definitions are not always correctly applied, it is often necessary to consult documentation, test connections with a breakout box, or use trial and error to find a cable that works when interconnecting two devices. Connecting a fully standard-compliant DCE device and DTE device would use a cable that connects identical pin numbers in each connector (a so-called "straight cable"). "Gender changers" are available to solve gender mismatches between cables and connectors. Connecting devices with different types of connectors requires a cable that connects the corresponding pins according to the table above. Cables with 9 pins on one end and 25 on the other are common. Manufacturers of equipment with 8P8C connectors usually provide a cable with either a DB-25 or DE-9 connector (or sometimes interchangeable connectors so they can work with multiple devices). Poor-quality cables can cause false signals by crosstalk between data and control lines (such as Ring Indicator). If a given cable will not allow a data connection, especially if a Gender changer is in use, a Null modem may be necessary.

Conventions

For functional communication through a serial port interface, conventions of bit rate, character framing, communications protocol, character encoding, data compression, and error detection, not defined in RS 232, must be agreed to by both sending and receiving equipment. For example, consider the serial ports of the original IBM PC. This implementation used an 8250 UART using asynchronous start-stop character formatting with 7 or 8 data bits per frame, usually ASCII character coding, and data rates programmable between 75 bits per second and 115,200 bits per second. Data rates above 20,000 bits per second are out of the scope of the standard, although higher data rates are sometimes used by commercially manufactured equipment. Since most RS-232 devices do not have automatic baud rate detection, users must manually set the baud rate (and all other parameters) at both ends of the RS-232 connection.

In the particular case of the 8250 UART used by the IBM PC and others, baud rates were programmable by writing integer values to a divider register and by selecting one of several clock prescalers for the divider. This allowed a PC to be connected to devices using rates other than those standardized for modems. Not all baud rates can be programmed, due to the clock frequency of the 8250 UART in the PC, and the granularity of the baud rate setting. This includes the baud rate of MIDI, 31,250 bits per second, which is not achievable by a standard IBM PC serial port.^[3] MIDI-to-RS-232 interfaces designed for the IBM PC include baud rate translation hardware to adjust the baud rate of the MIDI data to something that the IBM PC can support, for example 19,200 or 38,400 bits per second.

RTS/CTS handshaking

Further information: Flow control (data)

In older versions of the specification, RS-232's use of the RTS and CTS lines is asymmetric: The DTE asserts RTS to indicate a desire to transmit to the DCE, and the DCE asserts CTS in response to grant permission. This allows for half-duplex modems that disable their transmitters when not required, and must transmit a synchronization preamble to the receiver when they are re-enabled. This scheme is also employed on present-day RS-232 to RS-485 converters, where the RS-232's RTS signal is used to ask the converter to take control of the RS-485 bus—a concept that does not otherwise exist in RS-232. There is no way for the DTE to indicate that it is unable to accept data from the DCE.

A non-standard symmetric alternative, commonly called "RTS/CTS handshaking," was developed by various equipment manufacturers. In this scheme, CTS is no longer a response to RTS; instead, CTS indicates permission from the DCE for the DTE to send data to the DCE, and RTS indicates permission from the DTE for the DCE to send data to the DTE. RTS and CTS are controlled by the DTE and DCE respectively, each independent of the other. This was eventually codified in version RS-232-E (actually TIA-232-E by that time) by defining a new signal, "RTR (Ready to Receive)," which is CCITT V.24 circuit 133. TIA-232-E and the corresponding international standards were updated to show that circuit 133, when implemented, shares the same pin as RTS (Request to Send), and that when 133 is in use, RTS is assumed by the DCE to be ON at all times.

Thus, with this alternative usage, one can think of RTS asserted (positive voltage, logic 0) meaning that the DTE is indicating it is "ready to receive" from the DCE, rather than requesting permission from the DCE to send characters to the DCE.

Note that equipment using this protocol must be prepared to buffer some extra data, since a transmission may have begun just before the control line state change.

RTS/CTS handshaking is an example of hardware flow control. However, "hardware flow control" in the description of the options available on an RS-232-equipped device does not always mean RTS/CTS handshaking.

3-wire and 5-wire RS-232

A minimal "3-wire" RS-232 connection consisting only of transmit data, receive data, and ground, is commonly used when the full facilities of RS-232 are not required. Even a two-wire connection (data and ground) can be used if the data flow is one way (for example, a digital postal scale that periodically sends a weight reading, or a GPS receiver that periodically sends position, if no configuration via RS-232 is necessary). When only hardware flow control is required in addition to two-way data, the RTS and CTS lines are added in a 5-wire version.

Seldom used features

The EIA-232 standard specifies connections for several features that are not used in most implementations. Their use requires 25-pin connectors and cables.

Signal rate selection

The DTE or DCE can specify use of a "high" or "low" signaling rate. The rates as well as which device will select the rate must be configured in both the DTE and DCE. The prearranged device selects the high rate by setting pin 23 to ON.

Loopback testing

Many DCE devices have a loopback capability used for testing. When enabled, signals are echoed back to the sender rather than being sent on to the receiver. If supported, the DTE can signal the local DCE (the one it is connected to) to enter loopback mode by setting pin 18 to ON, or the remote DCE (the one the local DCE is connected to) to enter loopback mode by setting pin 21 to ON. The latter tests the communications link as well as both DCE's. When the DCE is in test mode it signals the DTE by setting pin 25 to ON.

A commonly used version of loopback testing does not involve any special capability of either end. A hardware loopback is simply a wire connecting complementary pins together in the same connector (see *loopback*).

Loopback testing is often performed with a specialized DTE called a bit error rate tester (or BERT).

Timing signals

Some synchronous devices provide a clock signal to synchronize data transmission, especially at higher data rates. Two timing signals are provided by the DCE on pins 15 and 17. Pin 15 is the transmitter clock, or send timing (ST); the DTE puts the next bit on the data line (pin 2) when this clock transitions from OFF to ON (so it is stable during the ON to OFF transition when the DCE registers the bit). Pin 17 is the receiver clock, or receive timing (RT); the DTE reads the next bit from the data line (pin 3) when this clock transitions from ON to OFF.

Alternatively, the DTE can provide a clock signal, called transmitter timing (TT), on pin 24 for transmitted data. Data is changed when the clock transitions from OFF to ON and read during the ON to OFF transition. TT can be used to overcome the issue where ST must traverse a cable of unknown length and delay, clock a bit out of the DTE after another unknown delay, and return it to the DCE over the same unknown cable delay. Since the relation between the transmitted bit and TT can be fixed in the DTE design, and since both signals traverse the same cable length, using TT eliminates the issue. TT may be generated by looping ST back with an appropriate phase change to align it with the transmitted data. ST loop back to TT lets the DTE use the DCE as the frequency reference, and correct the clock to data timing.

Synchronous clocking is required for such protocols as SDLC, HDLC, and X.25.

Secondary channel

There is a secondary data channel, identical in capability to the first. Five signals (plus the common ground of the primary channel) comprise the secondary channel: Secondary Transmitted Data (STD), Secondary Received Data (SRD), Secondary Request To Send (SRTS), Secondary Clear To Send (SCTS), and Secondary Carrier Detect (SDCD).

Related standards

Other serial signaling standards may not interoperate with standard-compliant RS-232 ports. For example, using the TTL levels of near +5 and 0 V puts the mark level in the undefined area of the standard. Such levels are sometimes used with NMEA 0183-compliant GPS receivers and depth finders.

A 20 mA current loop uses the absence of 20 mA current for high, and the presence of current in the loop for low; this signaling method is often used for long-distance and optically isolated links. Connection of a current-loop device to a compliant RS-232 port requires a level translator. Current-loop devices can supply voltages in excess of the withstand voltage limits of a compliant device. The original IBM PC serial port card implemented a 20 mA current-loop interface, which was never emulated by other suppliers of plug-compatible equipment.

Other serial interfaces similar to RS-232:

- RS-422 (a high-speed system similar to RS-232 but with differential signaling)
- RS-423 (a high-speed system similar to RS-422 but with unbalanced signaling)
- RS-449 (a functional and mechanical interface that used RS-422 and RS-423 signals - it never caught on like RS-232 and was withdrawn by the EIA)
- RS-485 (a descendant of RS-422 that can be used as a bus in multidrop configurations)
- MIL-STD-188 (a system like RS-232 but with better impedance and rise time control)
- EIA-530 (a high-speed system using RS-422 or RS-423 electrical properties in an EIA-232 pinout configuration, thus combining the best of both; supersedes RS-449)
- EIA/TIA-561 8 Position Non-Synchronous Interface Between Data Terminal Equipment and Data Circuit Terminating Equipment Employing Serial Binary Data Interchange
- EIA/TIA-562 Electrical Characteristics for an Unbalanced Digital Interface (low-voltage version of EIA/TIA-232)
- TIA-574 (standardizes the 9-pin D-subminiature connector pinout for use with EIA-232 electrical signalling, as originated on the IBM PC/AT)

Development tools

When developing or troubleshooting systems using RS-232, close examination of hardware signals can be important to find problems. A serial line analyzer is a device similar to a logic analyzer but specialized for RS-232's voltage levels, connectors, and, where used, clock signals. The serial line analyzer can collect, store, and display the data and control signals, allowing developers to view them in detail. Some simply display the signals as waveforms; more elaborate versions include the ability to decode characters in ASCII or other common codes and to interpret common protocols used over RS-232 such as SDLC, HDLC, DDCMP, and X.25. Serial line analyzers are available as standalone units, as software and interface cables for general-purpose logic analyzers and oscilloscopes, and as programs that run on common personal computers.

References



Wikimedia Commons has media related to **RS-232**.



Wikibooks has a book on the topic of: *Serial Programming:RS-232 Connections*

- [1] Metering Glossary (http://www.landisgyr.eu/en/pub/services_support/metering_glossary.cfm?eventGlossary=glossary.Search&initial=E) Landis + Gyr Tutorial (see *EIA*)
- [2] S. Mackay, E. Wright, D. Reynders, J. Park, *Practical Industrial Data Networks:Design, Installation and Troubleshooting*, Newnes, 2004 ISBN 07506 5807X, pages 41-42
- [3] *InfoWorld* 30 Mar 1992 page 80

Serial port

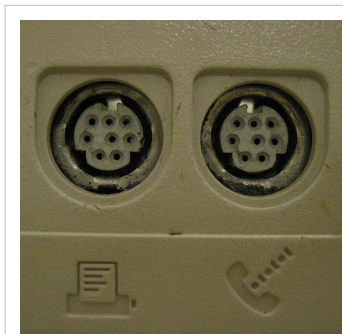
In computing, a **serial port** is a serial communication physical interface through which information transfers in or out one bit at a time (in contrast to a parallel port). Throughout most of the history of personal computers, data was transferred through serial ports to devices such as modems, terminals and various peripherals.



A male DE-9 connector used for a serial port on an IBM PC compatible computer along with the serial port symbol. (Pinout)

While such interfaces as Ethernet, FireWire, and USB all send data as a serial stream, the term "serial port" usually identifies hardware more or less compliant to the RS-232 standard, intended to interface with a modem or with a similar communication device.

Modern computers without serial ports may require serial-to-USB converters to allow compatibility with RS 232 serial devices. Serial ports are still used in applications such as industrial automation systems, scientific instruments, shop till systems and some industrial and consumer products. Server computers may use a serial port as a control console for diagnostics. Network equipment (such as routers and switches) often use serial console for configuration. Serial ports are still used in these areas as they are simple, cheap and their console functions are highly standardized and widespread. A serial port requires very little supporting software from the host system.



Pair of female Mini DIN-8 connectors used for RS-422 serial ports on a Macintosh LC computer

Hardware

Some computers, such as the IBM PC, used an integrated circuit called a UART, that converted characters to (and from) asynchronous serial form, and automatically looked after the timing and framing of data. Very low-cost systems, such as some early home computers, would instead use the CPU to send the data through an output pin, using the bit-banging technique. Before large-scale integration (LSI) UART integrated circuits were common, a minicomputer or microcomputer would have a serial port made of multiple small-scale integrated circuits to implement shift registers, logic gates, counters, and all the other logic for a serial port.

Early home computers often had proprietary serial ports with pinouts and voltage levels incompatible with RS-232. Inter-operation with RS-232 devices may be impossible as the serial port cannot withstand the voltage levels produced and may have other differences that "lock in" the user to products of a particular manufacturer.

Low-cost processors now allow higher-speed, but more complex, serial communication standards such as USB and FireWire to replace RS-232. These make it possible to connect devices that would not have operated feasibly over slower serial connections, such as mass storage, sound, and video devices.

Many personal computer motherboards still have at least one serial port, even if accessible only through a pin header. Small-form-factor systems and laptops may omit RS-232 connector ports to conserve space, but the electronics are still there. RS-232 has been standard for so long that the circuits needed to control a serial port became very cheap and often exist on a single chip, sometimes also with circuitry for a parallel port.

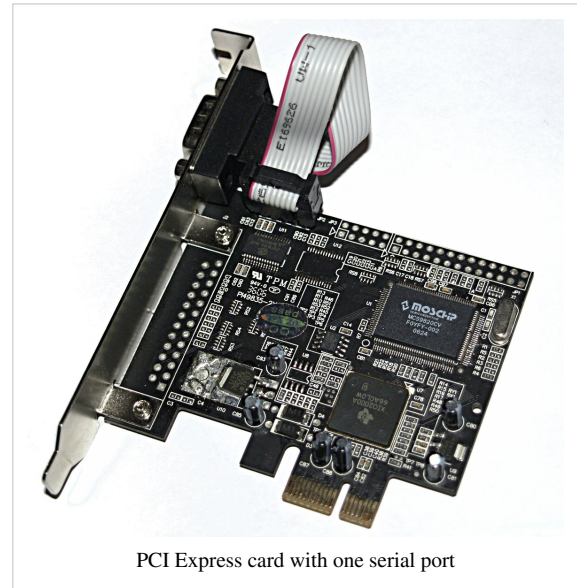
DTE and DCE

The individual signals on a serial port are unidirectional and when connecting two devices the outputs of one device must be connected to the inputs of the other. Devices are divided into two categories "data terminal equipment" (DTE) and "data circuit-terminating equipment" (DCE). A line that is an output on a DTE device is an input on a DCE device and vice-versa so a DCE device can be connected to a DTE device with a straight wired cable. Conventionally, computers and terminals are DTE while modems and peripherals are DCE.

If it is necessary to connect two DTE devices (or two DCE devices but that is more unusual) a cross-over null modem, in the form of either an adapter or a cable, must be used.

Connectors

While the RS-232 standard originally specified a 25-pin D-type connector, many designers of personal computers chose to implement only a subset of the full standard: they traded off compatibility with the standard against the use of less costly and more compact connectors (in particular the DE-9 version used by the original IBM PC-AT). The desire to supply serial interface cards with two ports required that IBM reduce the size of the connector to fit onto a single card back panel. A DE-9 connector also fits onto a card with a second DB-25 connector that was similarly changed from the original Centronics-style connector. Starting around the time of the introduction of the IBM PC-AT, serial ports were commonly built with a 9-pin connector to save cost and space. However, presence of a 9-pin D-subminiature connector is not sufficient to indicate the connection is in fact a serial port, since this connector was also used for video, joysticks, and other purposes.



PCI Express card with one serial port

Some miniaturized electronics, particularly graphing calculators and hand-held amateur and two-way radio equipment, have serial ports using a phone connector, usually the smaller 2.5 or 3.5 mm connectors and use the most basic 3-wire interface.

Many models of Macintosh favored the related RS-422 standard, mostly using German Mini-DIN connectors, except in the earliest models. The Macintosh included a standard set of two ports for connection to a printer and a modem, but some PowerBook laptops had only one combined port to save space.

The standard specifies 20 different signal connections. Since most devices use only a few signals, smaller connectors can often be used. For example, the 9-pin DE-9 connector was used by most IBM-compatible PCs since the IBM PC AT, and has been standardized as TIA-574. More recently, modular connectors have been used. Most common are 8P8C connectors. Standard EIA/TIA 561 specifies a pin assignment, but the "Yost Serial Device Wiring Standard"^[1] invented by Dave Yost (and popularized by the Unix System Administration Handbook) is common on Unix computers and newer devices from Cisco Systems. Many devices don't use either of these standards. 10P10C connectors can be found on some devices as well. Digital Equipment Corporation defined their own DECconnect connection system which was based on the Modified Modular Jack (MMJ) connector. This is a 6-pin modular jack where the key is offset from the center position. As with the Yost standard, DECconnect uses a symmetrical pin layout which enables the direct connection between two DTEs. Another common connector is the DH10 header connector common on motherboards and add-in cards which is usually converted via a cable to the more standard 9-pin DE-9 connector (and frequently mounted on a free slot plate or other part of the housing).

Pinouts

The following table lists commonly used RS-232 signals and pin assignments.

Signal		Origin		DB-25	DE-9 (TIA-574)	MMJ	8P8C ("RJ45")					10P10C ("RJ50")		
Name	Abbreviation	DTE	DCE				EIA/TIA-561	Yost (DTE)	Yost (DCE)	Cyclades ^[2]	Digi (ALTPIN option)	National Instruments ^[3]	Cyclades	Digi
Transmitted Data	TxD	●		2	3	2	6	6	3	3	4	8	4	5
Received Data	RxD		●	3	2	5	5	3	6	6	5	9	7	6
Data Terminal Ready	DTR	●		20	4	1	3	7	2	2	8	7	3	9
Data Carrier Detect	DCD		●	8	1	N/A	2	2	7	7	1	10	8	10
Data Set Ready	DSR		●	6	6	6	1			8	N/A	5	9	2
Ring Indicator	RI		●	22	9	N/A		N/A	N/A	N/A	N/A	2	10	1
Request To Send	RTS	●		4	7	N/A	8	8	1	1	2	4	2	3
Clear To Send	CTS		●	5	8	N/A	7	1	8	5	7	3	6	8
Signal Ground	G	common		7	5	3,4	4	4,5	4,5	4	6	6	5	7
Protective Ground	PG	common		1	N/A	N/A	N/A	N/A	N/A	N/A	3	N/A	1	4

The signals are named from the standpoint of the DTE, for example, an IBM-PC compatible serial port. The ground signal is a common return for the other connections; it appears on two pins in the Yost standard but is the same signal. The DB-25 connector includes a second "protective ground" on pin 1. Connecting this to pin 7 (signal reference ground) is a common practice but not essential.

Note that EIA/TIA 561 combines DSR and RI,^{[4][5]} and the Yost standard combines DSR and DCD.

Hardware abstraction

Operating systems usually use a symbolic name to refer to the serial ports of a computer. Unix-like operating systems usually label the serial port devices `/dev/tty*` (*TTY* is a common trademark-free abbreviation for *teletype*) where `*` represents a string identifying the terminal device; the syntax of that string depends on the operating system and the device. On Linux, 8250/16550 UART hardware serial ports are named `/dev/ttyS*`, USB adapters appear as `/dev/ttyUSB*` and various types of virtual serial ports do not necessarily have names starting with `tty`.

The Microsoft MS-DOS and Windows environments refer to serial ports as COM ports: COM1, COM2,...etc. Ports numbered greater than COM9 should be referred to using the `\\.\COM10` syntax.



A converter from USB to an RS-232 compatible serial port; more than a physical transition, it requires a driver in the host system software and a built-in processor to emulate the functions of the IBM XT compatible serial port hardware.

Common applications for serial ports

The RS-232 standard is used by many specialized and custom-built devices. This list includes some of the more common devices that are connected to the serial port on a PC. Some of these such as modems and serial mice are falling into disuse while others are readily available.

Serial ports are very common on most types of microcontroller, where they can be used to communicate with a PC or other serial devices.

- Dial-up modems
- Configuration and management of networking equipment such as routers, switches, firewalls, load balancers
- GPS receivers (typically NMEA 0183 at 4,800 bit/s)
- Bar code scanners and other point of sale devices
- LED and LCD text displays
- Satellite phones, low-speed satellite modems and other satellite based transceiver devices
- Flat-screen (LCD and Plasma) monitors to control screen functions by external computer, other AV components or remotes
- Test and measuring equipment such as digital multimeters and weighing systems
- Updating Firmware on various consumer devices.
- Some CNC controllers
- Uninterruptible power supply
- Stenography or Stenotype machines.
- Software debuggers that run on a second computer.
- Industrial field buses

- Printers
- Computer terminal, teletype
- Older digital cameras
- Networking (Macintosh AppleTalk using RS-422 at 230.4 kbit/s)
- Serial mouse
- Older GSM mobile phones
- Some Telescopes
- IDE hard drive repair

Since the control signals for a serial port can be easily turned on and off by a switch, some applications used the control lines of a serial port to monitor external devices, without exchanging serial data. A common commercial application of this principle was for some models of uninterruptible power supply which used the control lines to signal "loss of power", "battery low alarm" and other status information. At least some Morse code training software used a code key connected to the serial port, to simulate actual code use. The status bits of the serial port could be sampled very rapidly and at predictable times, making it possible for the software to decipher Morse code.

Settings

Many settings are required for serial connections used for asynchronous start-stop communication, to select speed, number of data bits per character, parity, and number of stop bits per character. In modern serial ports using a UART integrated circuit, all settings are usually software-controlled; hardware from the 1980s and earlier may require setting switches or jumpers on a circuit board. One of the simplifications made in such serial bus standards as Ethernet, FireWire, and USB is that many of those parameters have fixed values so that users can not and need not change the configuration; the speed is either fixed or automatically negotiated. Often if the settings are entered incorrectly the connection will not be dropped; however, any data sent will be received on the other end as nonsense.

Speed

Serial ports use two-level (binary) signaling, so the data rate in bits per second is equal to the symbol rate in bauds. A standard series of rates is based on multiples of the rates for electromechanical teleprinters; some serial ports allow many arbitrary rates to be selected. The port speed and device speed must match. The capability to set a bit rate does not imply that a working connection will result. Not all bit rates are possible with all serial ports. Some special-purpose protocols such as MIDI for musical instrument control, use serial data rates other than the teleprinter series. Some serial port systems can automatically detect the bit rate.

The speed includes bits for framing (stop bits, parity, etc.) and so the effective data rate is lower than the bit transmission rate. For example with 8-N-1 character framing only 80% of the bits are available for data (for every eight bits of data, two more framing bits are sent).

Bit rates commonly supported include 75, 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200 bit/s. Crystal oscillators with a frequency of 1.843200 MHz are sold specifically for this purpose. This is 16 times the fastest bit rate and the serial port circuit can easily divide this down to lower frequencies as required.

Data bits

The number of data bits in each character can be 5 (for Baudot code), 6 (rarely used), 7 (for true ASCII), 8 (for most kinds of data, as this size matches the size of a byte), or 9 (rarely used). 8 data bits are almost universally used in newer applications. 5 or 7 bits generally only make sense with older equipment such as teleprinters.

Most serial communications designs send the data bits within each byte LSB (Least significant bit) first. This standard is also referred to as "little endian." Also possible, but rarely used, is "big endian" or MSB (Most Significant Bit) first serial communications; this was used, for example, by the IBM 2741 printing terminal. (See Bit numbering for more about bit ordering.) The order of bits is not usually configurable within the serial port interface. To communicate with systems that require a different bit ordering than the local default, local software can re-order the bits within each byte just before sending and just after receiving.

Parity

Main article: Parity bit

Parity is a method of detecting errors in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of 1s, then it must have been corrupted. However, an even number of errors can pass the parity check.

Electromechanical teleprinters were arranged to print a special character when received data contained a parity error, to allow detection of messages damaged by line noise. A single parity bit does not allow implementation of error correction on each character, and communication protocols working over serial data links will have higher-level mechanisms to ensure data validity and request retransmission of data that has been incorrectly received.

The parity bit in each character can be set to none (N), odd (O), even (E), mark (M), or space (S). None means that no parity bit is sent at all. Mark parity means that the parity bit is always set to the mark signal condition (logical 1) and likewise space parity always sends the parity bit in the space signal condition. Aside from uncommon applications that use the 9th (parity) bit for some form of addressing or special signalling, mark or space parity is uncommon, as it adds no error detection information. Odd parity is more useful than even, since it ensures that at least one state transition occurs in each character, which makes it more reliable. The most common parity setting, however, is "none", with error detection handled by a communication protocol.

Stop bits

Stop bits sent at the end of every character allow the receiving signal hardware to detect the end of a character and to resynchronise with the character stream. Electronic devices usually use one stop bit. If slow electromechanical teleprinters are used, one-and-one half or two stop bits are required.

Conventional notation

The D/P/S (Data/Parity/Stop) conventional notation specifies the framing of a serial connection. The most common usage on microcomputers is 8/N/1 (8N1). This specifies 8 data bits, no parity, 1 stop bit. In this notation, the parity bit is not included in the data bits. 7/E/1 (7E1) means that an even parity bit is added to the seven data bits for a total of eight bits between the start and stop bits. If a receiver of a 7/E/1 stream is expecting an 8/N/1 stream, half the possible bytes will be interpreted as having the high bit set.

Flow control

Main article: Flow control (data)

A serial port may use signals in the interface to pause and resume the transmission of data. For example, a slow printer might need to handshake with the serial port to indicate that data should be paused while the mechanism advances a line.

Common hardware handshake signals (hardware flow control) use the RS-232 RTS/CTS or DTR/DSR signal circuits. Generally, the RTS and CTS are turned off and on from alternate ends to control data flow, for instance when a buffer is almost full. DTR and DSR are usually on all the time and, per the RS-232 standard and its successors, are used to signal from each end that the other equipment is actually present and powered-up. However, manufacturers have over the years built many devices that implemented non-standard variations on the standard, for example, printers that use DTR as flow control.

Another method of flow control (software flow control) uses special characters such as XON/XOFF to control the flow of data. The XON/XOFF characters are sent by the receiver to the sender to control when the sender will send data, that is, these characters go in the opposite direction to the data being sent. The circuit starts in the "sending allowed" state. When the receiver's buffers approach capacity, the receiver sends the XOFF character to tell the sender to stop sending data. Later, after the receiver has emptied its buffers, it sends an XON character to tell the sender to resume transmission. These are non-printing characters and are interpreted as handshake signals by printers, terminals, and computer systems.

XON/XOFF flow control is an example of in-band signaling, in which control information is sent over the same channel used for the data. XON/XOFF handshaking presents difficulties as XON and XOFF characters might appear in the data being sent and receivers may interpret them as flow control. Such characters sent as part of the data stream must be encoded in an escape sequence to prevent this, and the receiving and sending software must generate and interpret these escape sequences. On the other hand, since no extra signal circuits are required, XON/XOFF flow control can be done on a 3 wire interface.

"Virtual" serial ports

Main article: COM port redirector

A **virtual serial port** is an emulation of the standard serial port. This port is created by software which enable extra serial ports in an operating system without additional hardware installation (such as expansion cards, etc.). It is possible to create a large number of virtual serial ports in a PC. The only limitation is the amount of resources, such as operating memory and computing power, needed to emulate many serial ports at the same time.

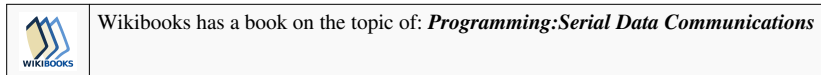
Virtual serial ports emulate all hardware serial port functionality, including Baud rate, Data bits, Parity bits, Stop bits, etc. Additionally they allow controlling the data flow, emulating all signal lines (DTR/DSR/CTS/RTS/DCD/RI) and customizing pinout. Virtual serial ports are common with Bluetooth and are the standard way of receiving data from Bluetooth-equipped GPS modules.

Virtual serial port emulation can be useful in case there is a lack of available physical serial ports or they do not meet the current requirements. For instance, virtual serial ports can share data between several applications from one GPS device connected to a serial port. Another option is to communicate with any other serial devices via internet or LAN as if they are locally connected to computer (Serial over LAN/Serial-over-Ethernet technology). Two computers or applications can communicate through an emulated serial port link. Virtual serial port emulators are available for many operating systems including MacOS, Linux, and various mobile and desktop versions of Microsoft Windows.

References

- [1] Yost Serial Device Wiring Standard (<http://yost.com/computers/RJ45-serial>)
- [2] Cyclom-Y Installation Manual, page 38, retrieved on 29 November 2008 (ftp://ftp.cyclades.com/pub/cyclades/cyclom-y/doc/y_30.pdf#38)
- [3] National Instruments Serial Quick Reference Guide, February 2007 (<http://www.ni.com/pdf/manuals/371253c.pdf>)
- [4] Hardware Book RS-232D (<http://www.hardwarebook.info/RS-232D>)
- [5] RS-232D EIA/TIA-561 RJ45 Pinout (<http://www.t0rchthe.net/rj45console/index.html>)

External links



- Serial Port Programming in Linux (<http://trainingkits.gweb.io/serial-linux.html>)
- RS-232 and other serial port pinouts list (<http://pinouts.ru/SerialPorts/>)
- Back of an old desktop computer showing 25-pin male serial port. (http://classconnection.s3.amazonaws.com/263/flashcards/2010263/jpg/img_30181349289098468.jpg)

Universal asynchronous receiver/transmitter

A **universal asynchronous receiver/transmitter**, abbreviated **UART** /ˈjuːɑːrt/, is a piece of computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485. The *universal* designation indicates that the data format and transmission speeds are configurable. The electric signaling levels and methods (such as differential signaling etc.) are handled by a driver circuit external to the UART.

A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are now commonly included in microcontrollers. A dual UART, or **DUART**, combines two UARTs into a single chip. An octal UART or **OCTART** combines eight UARTs into one package, an example being the NXP SCC2698. Many modern ICs now come with a UART that can also communicate synchronously; these devices are called **USARTs** (universal synchronous/asynchronous receiver/transmitter).

Transmitting and receiving serial data

See also: Asynchronous serial communication

The Universal Asynchronous Receiver/Transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion.^[1] At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signalling levels. External signals may be of many different forms. Examples of standards for voltage signaling are RS-232, RS-422 and RS-485 from the EIA. Historically, current (in current loops) was used in telegraph circuits. Some signaling schemes do not use electrical wires. Examples of such are optical fiber, IrDA (infrared), and (wireless) Bluetooth in its Serial Port Profile (SPP). Some signaling schemes use modulation of a carrier signal (with or without wires). Examples are modulation of audio signals with phone line modems, RF modulation with data radios, and the DC-LIN^[2] for power line communication.

Communication may be *simplex* (in one direction only, with no provision for the receiving device to send information back to the transmitting device), *full duplex* (both devices send and receive at the same time) or *half duplex* (devices take turns transmitting and receiving).

Character framing

The right-most (least significant) data bit is always transmitted first. If parity is present, the parity bit comes after the data bits but before the stop bit(s).

Bit number	1	2	3	4	5	6	7	8	9	10	11
	Start bit	5–8 data bits								Stop bit(s)	
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Stop	

The idle, no data state is high-voltage, or powered. This is a historic legacy from telegraphy, in which the line is held high to show that the line and transmitter are not damaged. Each character is sent as a logic low start bit, a configurable number of data bits (usually 8, but users can choose 5 to 8 or 9 bits depending on which UART is in use), an optional parity bit if the number of bits per character chosen is not 9 bits, and one or more logic high stop bits.

The start bit signals the receiver that a new character is coming. The next five to nine bits, depending on the code set employed, represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always in the **mark** (logic high, i.e., '1') condition and called the stop bit(s). They signal the receiver that the character is completed. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

If the line is held in the logic low condition for longer than a character time, this is a break condition that can be detected by the UART.

Receiver

All operations of the UART hardware are controlled by a clock signal which runs at a multiple of the data rate, typically 8 times the bit rate. The receiver tests the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. If the apparent start bit lasts at least one-half of the bit time, it is valid and signals the start of a new character. If not, it is considered a spurious pulse and is ignored. After waiting a further bit time, the state of the line is again sampled and the resulting level clocked into a shift register. After the required number of bit periods for the character length (5 to 8 bits, typically) have elapsed, the contents of the shift register are made available (in parallel fashion) to the receiving system. The UART will set a flag indicating new data is available, and may also generate a processor interrupt to request that the host processor transfers the received data.

Communicating UARTs usually have no shared timing system apart from the communication signal. Typically, UARTs resynchronize their internal clocks on each change of the data line that is not considered a spurious pulse. Obtaining timing information in this manner, they reliably receive when the transmitter is sending at a slightly different speed than it should. Simplistic UARTs do not do this, instead they resynchronize on the falling edge of the start bit only, and then read the center of each expected data bit, and this system works if the broadcast data rate is accurate enough to allow the stop bits to be sampled reliably.

It is a standard feature for a UART to store the most recent character while receiving the next. This "double buffering" gives a receiving computer an entire character transmission time to fetch a received character. Many UARTs have a small first-in, first-out FIFO buffer memory between the receiver shift register and the host system interface. This allows the host processor even more time to handle an interrupt from the UART and prevents loss of received data at high rates.

Transmitter

Transmission operation is simpler since it is under the control of the transmitting system. As soon as data is deposited in the shift register after completion of the previous character, the UART hardware generates a start bit, shifts the required number of data bits out to the line, generates and appends the parity bit (if used), and appends the stop bits. Since transmission of a single character may take a long time relative to CPU speeds, the UART will maintain a flag showing busy status so that the host system does not deposit a new character for transmission until the previous one has been completed; this may also be done with an interrupt. Since full-duplex operation requires characters to be sent and received at the same time, UARTs use two different shift registers for transmitted and received characters.

Application

Transmitting and receiving UARTs must be set for the same bit speed, character length, parity, and stop bits for proper operation. The receiving UART may detect some mismatched settings and set a "framing error" flag bit for the host system; in exceptional cases the receiving UART will produce an erratic stream of mutilated characters and transfer them to the host system.

Typical serial ports used with personal computers connected to modems use eight data bits, no parity, and one stop bit; for this configuration the number of ASCII characters per second equals the bit rate divided by 10.

Some very low-cost home computers or embedded systems dispense with a UART and use the CPU to sample the state of an input port or directly manipulate an output port for data transmission. While very CPU-intensive (since the CPU timing is critical), the UART chip can thus be omitted, saving money and space. The technique is known as bit-banging.

Synchronous transmission

USART chips have both synchronous and asynchronous modes. In *synchronous* transmission, the clock data is recovered separately from the data stream and no start/stop bits are used. This improves the efficiency of transmission on suitable channels since more of the bits sent are usable data and not character framing. An asynchronous transmission sends no characters over the interconnection when the transmitting device has nothing to send; but a synchronous interface must send "pad" characters to maintain synchronization between the receiver and transmitter. The usual filler is the ASCII "SYN" character. This may be done automatically by the transmitting device.

History

Some early telegraph schemes used variable-length pulses (as in Morse code) and rotating clockwork mechanisms^[3] to transmit alphabetic characters. The first UART-like devices (with fixed-length pulses) were rotating mechanical switches (*commutators*). Various character codes using 5, 6, 7, or 8 data bits became common in teleprinters and later as computer peripherals. Gordon Bell designed the UART for the PDP series of computers. The teletypewriter made an excellent general-purpose I/O device for a small computer. To reduce costs, including wiring and back-plane costs, these computers also pioneered flow control using XON and XOFF characters rather than hardware wires.

Western Digital made the first single-chip UART WD1402A around 1971; this was an early example of a medium scale integrated circuit. Another popular chip was a SCN2651 from the Signetics 2650 family.

An example of an early 1980s UART was the National Semiconductor 8250. In the 1990s, newer UARTs were developed with on-chip buffers. This allowed higher transmission speed without data loss and without requiring such frequent attention from the computer. For example, the popular National Semiconductor 16550 has a 16 byte FIFO, and spawned many variants, including the *16C550*, *16C650*, *16C750*, and *16C850*.

Depending on the manufacturer, different terms are used to identify devices that perform the UART functions. Intel called their 8251 device a "Programmable Communication Interface". MOS Technology 6551 was known under the name "Asynchronous Communications Interface Adapter" (ACIA). The term "Serial Communications Interface" (SCI) was first used at Motorola around 1975 to refer to their start-stop asynchronous serial interface device, which others were calling a UART. Zilog manufactured a number of Serial Communication Controllers or SCCs.

Structure

A UART usually contains the following components:

- a clock generator, usually a multiple of the bit rate to allow sampling in the middle of a bit period.
- input and output shift registers
- transmit/receive control
- read/write control logic
- transmit/receive buffers (optional)
- parallel data bus buffer (optional)
- First-in, first-out (FIFO) buffer memory (optional)

Special receiver conditions

Overrun error

An "overrun error" occurs when the receiver cannot process the character that just came in before the next one arrives. Various devices have different amounts of buffer space to hold received characters. The CPU must service the UART in order to remove characters from the input buffer. If the CPU does not service the UART quickly enough and the buffer becomes full, an Overrun Error will occur, and incoming characters will be lost.

Underrun error

An "underrun error" occurs when the UART transmitter has completed sending a character and the transmit buffer is empty. In asynchronous modes this is treated as an indication that no data remains to be transmitted, rather than an error, since additional stop bits can be appended. This error indication is commonly found in USARTs, since an underrun is more serious in synchronous systems.

Framing error

A "framing error" occurs when the designated "start" and "stop" bits are not found. As the "start" bit is used to identify the beginning of an incoming character, it acts as a reference for the remaining bits. If the data line is not in the expected state (hi/lo) when the "stop" bit is expected, a *Framing Error* will occur.

Parity error

A Parity Error occurs when the parity of the number of 1 bits disagrees with that specified by the parity bit. Use of a parity bit is optional, so this error will only occur if parity-checking has been enabled.

Break condition

A "break condition" occurs when the receiver input is at the "space" level for longer than some duration of time, typically, for more than a character time. This is not necessarily an error, but appears to the receiver as a character of all zero bits with a framing error. The term "break" derives from current loop signaling, which was the traditional signaling used for teletypewriters. The "spacing" condition of a current loop line is indicated by no current flowing, and a very long period of no current flowing is often caused by a break or other fault in the line.

Some equipment will deliberately transmit the "space" level for longer than a character as an attention signal. When signaling rates are mismatched, no meaningful characters can be sent, but a long "break" signal can be a useful way to get the attention of a mismatched receiver to do something (such as resetting itself). Unix-like systems can use the long "break" level as a request to change the signaling rate, to support dial-in access at multiple signaling rates.

UART models

Model	Description
EXAR XR21V1410	
Intersil 6402	
CDP 1854 (RCA, now Intersil)	
Zilog Z8440	2000 kbit/s. Async, Bisync, SDLC, HDLC, X.25. CRC. 4-byte RX buffer. 2-byte TX buffer. DMA. ^[4]
8250	Obsolete with 1-byte buffers. These UARTs' maximum standard serial port speed is 9600 bits per second if the operating system has a 1 millisecond interrupt latency.
8251	
Z8530/85C30	
Motorola 6850	
6551	
Rockwell 65C52	
16450	
16550	This UART's FIFO was broken, so it cannot safely run any faster than the 16450 UART. The 16550A and later versions fix this bug.
16550A	This UART has 16-byte FIFO buffers. Its receive interrupt trigger levels can be set to 1, 4, 8, or 14 characters. Its maximum standard serial port speed if the operating system has a 1 millisecond interrupt latency is 115.2 kbit/s. Operating systems with lower interrupt latencies could handle higher baud rates like 230.4 kbit/s or 460.8 kbit/s. This chip can provide signals to facilitate a third party DMA controller perform DMA transfers to and from the UART. This was known as DMA mode because it was meant to be coupled with a DMA controller in this mode to perform the transfers on behalf of the CPU. ^[5] It was introduced by National Semiconductor, which has been sold to Texas Instruments. National Semiconductor claimed that this UART could physically run at up to 1.5 Mbit/s.
16C552	
16650	This UART was introduced by Startech Semiconductor which is now owned by Exar Corporation and is not related to Startech.com. Early versions had a broken FIFO buffer and therefore cannot safely run any faster than the 16450 UART. Versions of this UART that were not broken had 32-character FIFO buffers and could function at standard serial port speeds up to 230.4 kbit/s if the operating system has a 1 millisecond interrupt latency. Current versions of this UART by Exar claim to be able to physically handle up to 1.5 Mbit/s. This UART introduces the Auto-RTS and Auto-CTS features in which the RTS# signal is controlled by the UART to signal the external device to stop transmitting when the UART's buffer is full to or beyond a user-set trigger point and to stop transmitting to the device when the device drives the CTS# signal high (logic 0).
16750	64-byte buffers. This UART can handle a maximum standard serial port speed of 460.8 kbit/s if the maximum interrupt latency is 1 millisecond. This UART was introduced by Texas Instruments. TI claims that early models can run up to 1 Mbit/s physically, and later models can run up to 5 Mbit/s physically.
16850	128-byte buffers. This UART can handle a maximum standard serial port speed of 921.6 kbit/s if the maximum interrupt latency is 1 millisecond. This UART was introduced by Exar Corporation. Exar claims that early models can run up to 1.5 Mbit/s physically, and later models can run up to 6.25 Mbit/s physically.
16C850	

16950	128-byte buffers. This UART can handle a maximum standard serial port speed of 921.6 kbit/s if the maximum interrupt latency is 1 millisecond and if the UART is not connected to a DMA engine or is connected to a DMA engine that is not enabled. This UART supports 9-bit characters in addition to the 5-8 bit characters other UARTs support. This was introduced by Oxford Semiconductor, which is now owned by PLX Technology. Oxford/PLX claims that this UART can run up to 15 Mbit/s physically. PCI Express variants by Oxford/PLX can safely run much faster than other variants because they are integrated with a first party
16C950	bus mastering PCIe DMA engine. This DMA engine is controlled by the UART's DMA mode signals that were defined for the 16550. The DMA engine will prevent buffer overruns by moving data in the receive buffer to the host computer's memory via PCIe, and can speed up transmission by moving data to be sent in the host computer's memory to the transmit buffer if it is not full. Both of these operations do require some setup by the CPU, but are automated by the UART and the DMA engine after setup is complete.
16954	Quad port version of the 16950/16C950. 128-byte buffers per port. This UART can handle a maximum standard serial port speed of 921.6 kbit/s if the maximum interrupt latency is 1 millisecond and if the UART is not connected to a DMA engine or is connected to a DMA engine that is not enabled. This UART supports 9-bit characters in addition to the 5-8 bit characters other UARTs support. This was introduced by Oxford Semiconductor, which is now owned by PLX Technology. Oxford/PLX claims that this UART can run up to 15 Mbit/s physically. PCI Express variants by Oxford/PLX can safely run much faster than other variants because they are integrated with a first party bus mastering PCIe DMA engine. This DMA engine is controlled by the
16C954	UART's DMA mode signals that were defined for the 16550. The DMA engine will prevent buffer overruns by moving data in the receive buffer to the host computer's memory via PCIe, and can speed up transmission by moving data to be sent in the host computer's memory to the transmit buffer if it is not full. Both of these operations do require some setup by the CPU, but are automated by the UART and the DMA engine after setup is complete.
SCC2691	Currently produced by NXP, the 2691 is a single channel UART that also includes a programmable counter/timer. The 2691 has a single byte transmitter holding register and a 4-byte receive FIFO. Maximum standard speed of the 2692 is 115.2 kbit/s. Non-standard speeds are supported.
SCC2692	Currently produced by NXP, these dual UARTs (DUART) are essentially a pair of SCC2691 UARTs in a single package, but with a common counter/timer. Each channel is independently programmable and supports independent transmit and receive data rates. Like the 2691, the 2692 has a single byte transmitter holding register and a 4-byte receive FIFO per channel. Maximum standard speed of both of the 2692's channels is 115.2 kbit/s. The 26C92 is an upwardly compatible version of the dual channel 2692, with 8-byte transmit and receive FIFOs for improved performance during continuous bi-directional asynchronous transmission (CBAT) on both channels at the maximum standard speed of 230.4 kbit/s.
SC26C92	Both the 2692 and 26C92 may also be operated in RS-422 and RS-485 modes, and can also be programmed to support non-standard data rates. The devices are produced in PDIP-40, PLCC-44 and 44 pin QFP packages, and are readily adaptable to both Motorola and Intel buses. They have also been successfully adapted to the 65C02 and 65C816 buses.
SCC2698B	Currently produced by NXP, the 2698 octal UART (OCTART) is essentially four SCC2692 DUARTs in a single package. Specifications are the same as the SCC2692 (not the SCC26C92). The device is produced in PDIP-64 and PLCC-84 packages, and is readily adaptable to both Motorola and Intel buses. The 2698 has also been successfully adapted to the 65C02 and 65C816 buses.
SCC28C94	Currently produced by NXP, the 28C94 quadruple UART (QUART) is functionally similar to a pair of SCC26C92 DUARTs mounted in a common package. Some additional signals are present for interrupt management and the auxiliary input/output pins are arranged differently than those of the 26C92. Otherwise, the programming model for the 28C94 is very similar to that of the 26C92, requiring only minor code changes. The 28C94 supports a maximum standard speed of 230.4 kbit/s, is available in a PLCC-52 package, and is readily adaptable to both Motorola and Intel buses.
SCC28L198	Currently produced by NXP, the 28L198 octal UART (OCTART) is essentially an upscaled enhancement of the SCC28C94 QUART (described above), with eight independent communications channels, as well as an arbitrated interrupt system for efficient processing during periods of intense channel activity. The 28L198 supports a maximum standard speed of 460.8 kbit/s, is available in PLCC-84 and LQFP-100 packages, and is readily adaptable to both Motorola and Intel buses. The 28L198 will operate on 3.3 or 5 volts.
Z85230	Synchronous/Asynchronous modes, 2 ports, DMA. 4-byte buffer to send, 8-byte buffer to receive per channel. SDLC/HDLC modes. 5 Mbit/s in synchronous mode.
Hayes ESP	1 kB buffers, 921.6 kbit/s, 8-ports. ^[6]

UART in modems

Modems for personal computers that plug into a motherboard slot must also include the UART function on the card. The original 8250 UART chip shipped with the IBM personal computer had a one character buffer for the receiver and the transmitter each, which meant that communications software performed poorly at speeds above 9600 bits/second, especially if operating under a multitasking system or if handling interrupts from disk controllers. High-speed modems used UARTs that were compatible with the original chip but which included additional FIFO buffers, giving software additional time to respond to incoming data.

A look at the performance requirements at high bit rates shows why the 16, 32, 64 or 128 byte FIFO is a necessity. The Microsoft specification for a DOS system requires that interrupts not be disabled for more than 1 millisecond at a time. Some hard disk drives and video controllers violate this specification. 9600 bit/s will deliver a character approximately every millisecond, so a 1 byte FIFO should be sufficient at this rate on a DOS system which meets the maximum interrupt disable timing. Rates above this may receive a new character before the old one has been fetched, and thus the old character will be lost. This is referred to as an overrun error and results in one or more lost characters.

A 16 byte FIFO allows up to 16 characters to be received before the computer has to service the interrupt. This increases the maximum bit rate the computer can process reliably from 9600 to 153,000 bit/s if it has a 1 millisecond interrupt dead time. A 32 byte FIFO increases the maximum rate to over 300,000 bit/s. A second benefit to having a FIFO is that the computer only has to service about 8 to 12% as many interrupts, allowing more CPU time for updating the screen, or doing other chores. Thus the computer's responses will improve as well.

References

- [1] Adam Osborne, *An Introduction to Microcomputers Volume 1: Basic Concepts*, Osborne-McGraw Hill Berkeley California USA, 1980 ISBN 0-931988-34-9 pp. 116-126
- [2] <http://www.yamar.com/DC-LIN.html>
- [3] http://www.railroad-signaling.com/tty/m19/M19_8w.jpg
- [4] 090529 zilog.com
- [5] 090529 cs.utk.edu
- [6] bill.herrin.us - Hayes ESP 8-port Enhanced Serial Port Manual (<http://bill.herrin.us/freebies/hayes-esp8/>), 2004-03-02

External links

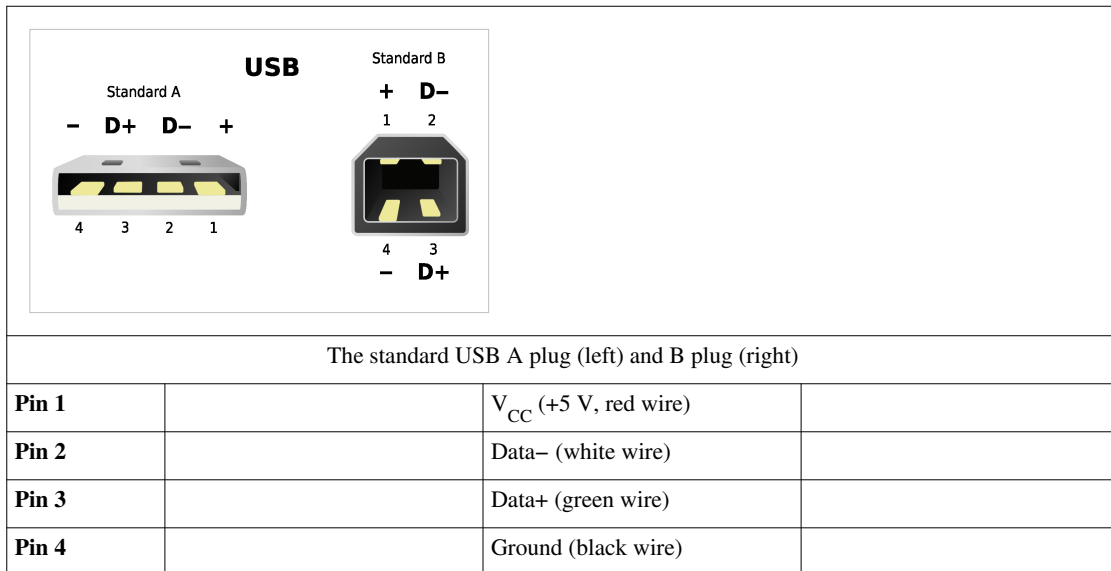
- A tutorial on the RS-232 standard (http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html#anchor1155404), describing the definition of mark, space and their relationship with negative and positive voltages
- FreeBSD Tutorials (http://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/), includes standard signal definitions, history of UART ICs, and pinout for commonly used DB25 connector.
- UART Tutorial for Robotics (http://www.societyofrobots.com/microcontroller_uart.shtml), contains many practical examples.
- UART transceiver over powerline (<http://www.yamar.com/sig60.php/>), allows multiplex UART networking over the powerline.
- CladLabs: UART (<http://cladlab.com/electronics/circuit-design/communication-protocols/uart-protocol>), a tutorial on the UART protocol explaining flow control, transmission speed, radiation hardening, powerline transceivers and reviewing terminal programs.

USB

For other uses, see USB (disambiguation).

Universal Serial Bus (USB)

Certified USB logo	
Type	Bus
Production history	
Designer	Compaq, Digital Equipment Corporation, IBM, Intel, Microsoft, NEC and Nortel
Designed	1996
Manufacturer	Intel, Compaq, Microsoft, NEC, Digital Equipment Corporation, IBM, Nortel
Produced	1997–present
Superseded	Serial port, parallel port, game port, Apple Desktop Bus, PS/2 connector
General specifications	
Length	2–5 m (6 ft 7 in–16 ft 5 in) (by category)
Width	12 mm (A-plug), 8.45 mm (B-plug); 7 mm (Mini / Micro-USB)
Height	4.5 mm (A-plug), 7.78 mm (B-plug, pre-v3.0); 1.5–3 mm (Mini/Micro-USB)
Hot pluggable	Yes
External	Yes
Cable	4 wires plus shield (pre-3.0); 9 wires plus shield (USB 3.0)
Pins	4: 1 supply, 2 data, 1 ground (pre-3.0); 9 (USB 3.0); 11 (powered USB 3.0); 5 (pre-3.0 Micro-USB)
Connector	Unique
Electrical	
Signal	<u>5 volt DC</u>
Max. voltage	5.00±0.25 V (pre-3.0); 5.00+0.25-0.55 V (USB 3.0)
Max. current	0.5–0.9 A (general); 5 A (charging devices)
Data	
Data signal	Packet data, defined by specifications
Width	1 bit
Bitrate	1.5/12/480/5,000/10,000 Mbit/s (depending on mode)
Max. devices	127
Protocol	Serial
Pin out	



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 smartphones, 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.

Overview

In general, there are four basic kinds or sizes related to the USB connectors and types of established connection:

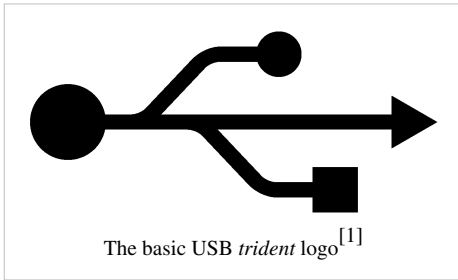
- the older "standard" size, in its USB 1.1/2.0 and USB 3.0 variants (for example, on USB flash drives)
- the "mini" size (primarily for the B connector end, such as on many cameras)
- the "micro" size, in its USB 1.1/2.0 and USB 3.0 variants (for example, on most modern cellphones)
- the versatile "USB On-The-Go" scheme, in both mini and micro sizes.

Unlike other data cables (Ethernet, HDMI etc.), each end of a USB cable uses a different *kind* of connector; an A-type or a B-type. This kind of design was chosen to prevent electrical overloads and damaged equipment, as only the A-type socket provides power. There are cables with A-type connectors on both ends, but they should be used carefully. Therefore in general, each of the different "sizes" requires four different connectors; USB cables have the A-type and B-type connectors, and the corresponding sockets are on the computer or electronic device. In common practice, the A-type connector is usually the full size, and the B-type side can vary as needed.

Counter-intuitively, the "micro" size is the most durable from the point of designed insertion lifetime, as the result of latching mechanism (parts providing gripping force) being moved into plugs on the cable side.

USB connections also come in four data transfer speeds: Low Speed, Full Speed, High Speed and SuperSpeed. High Speed is only supported by specifically designed USB 2.0 High Speed interfaces (that is, USB 2.0 controllers without the High Speed designation do not support it), as well as by USB 3.0 interfaces. SuperSpeed is supported only by USB 3.0 interfaces.

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.



The original USB 1.0 specification, which was introduced in January 1996, defined data transfer rates of 1.5 Mbit/s "Low Speed" and 12 Mbit/s "Full Speed". The first widely used version of USB was 1.1, which was released in September 1998. The 12 Mbit/s data rate was intended for higher-speed devices such as disk drives, and the lower 1.5 Mbit/s rate for low data rate devices such as joysticks.

The USB 2.0 specification was released in April 2000 and was ratified by the USB Implementers Forum (USB-IF) at the end of 2001. Hewlett-Packard, Intel, Lucent Technologies (now Alcatel-Lucent), NEC and Philips jointly led the initiative to develop a higher data transfer rate, with the resulting specification achieving 480 Mbit/s, a 40-times increase over the original USB 1.1 specification.

The USB 3.0 specification was published on 12 November 2008. Its main goals were to increase the data transfer rate (up to 5 Gbit/s), decrease power consumption, increase power output, and be backwards-compatible with USB 2.0.^[2] USB 3.0 includes a new, higher speed bus called SuperSpeed in parallel with the USB 2.0 bus.

For this reason, the new version is also called SuperSpeed. The first USB 3.0 equipped devices were presented in January 2010.

As of 2008^[3], approximately six billion USB ports and interfaces were in the global marketplace, and about two billion were being sold each year.

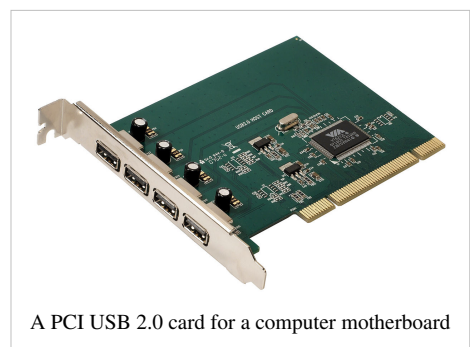


Version history

Prereleases

The USB standard evolved through several versions before its official release in 1996:

- *USB 0.7* – released in November 1994
- *USB 0.8* – released in December 1994
- *USB 0.9* – released in April 1995
- *USB 0.99* – released in August 1995
- *USB 1.0 Release Candidate* – released in November 1995



USB 1.x

Released in January 1996, USB 1.0 specified data rates of *1.5 Mbit/s (Low-Bandwidth)* and *12 Mbit/s (Full-Bandwidth)*. It did not allow for extension cables or pass-through monitors, due to timing and power limitations. Few USB devices made it to the market until USB 1.1 was released in August 1998, fixing problems identified in 1.0, mostly related to using hubs. USB 1.1 was the earliest revision that was widely adopted.

USB 2.0

USB 2.0 was released in April 2000 (now called "*Hi-Speed*"), adding higher maximum signaling rate of *480 Mbit/s* (due to bus access constraints the effective throughput is limited to 35 MB/s or 280 Mbit/s), in addition to the "USB 1.x Full Speed" signaling rate of 12 Mbit/s.

Further modifications to the USB specification have been done via Engineering Change Notices (ECN). The most important of these ECNs are included into the USB 2.0 specification package available from USB.org:



- *Mini-A and Mini-B Connector ECN*: Released in October 2000.
Specifications for Mini-A and B plug and receptacle. Also receptacle that accepts both plugs for On-The-Go. These should not be confused with Micro-B plug and receptacle.
- *Errata as of December 2000*: Released in December 2000
- *Pull-up/Pull-down Resistors ECN*: Released in May 2002
- *Errata as of May 2002*: Released in May 2002
- *Interface Associations ECN*: Released in May 2003.
New standard descriptor was added that allows associating multiple interfaces with a single device function.
- *Rounded Chamfer ECN*: Released in October 2003.
A recommended, compatible change to Mini-B plugs that results in longer lasting connectors.
- *Unicode ECN*: Released in February 2005.
This ECN specifies that strings are encoded using UTF-16LE. USB 2.0 specified Unicode, but did not specify the encoding.
- *Inter-Chip USB Supplement*: Released in March 2006
- *On-The-Go Supplement 1.3*: Released in December 2006.
USB On-The-Go makes it possible for two USB devices to communicate with each other without requiring a separate USB host. In practice, one of the USB devices acts as a host for the other device.
- *Battery Charging Specification 1.1*: Released in March 2007 (Updated 15 Apr 2009).
Adds support for dedicated chargers (power supplies with USB connectors), host chargers (USB hosts that can act as chargers) and the No Dead Battery provision, which allows devices to temporarily draw 100 mA current after they have been attached. If a USB device is connected to dedicated charger, maximum current drawn by the device may be as high as 1.8 A. (Note that this document is not distributed with USB 2.0 specification package only USB 3.0 and USB On-The-Go.)
- *Micro-USB Cables and Connectors Specification 1.01*: Released in April 2007.
- *Link Power Management Addendum ECN*: Released in July 2007.
This adds "sleep", a new power state between enabled and suspended states. Device in this state is not required to reduce its power consumption. However, switching between enabled and sleep states is much faster than switching between enabled and suspended states, which allows devices to sleep while idle.
- *Battery Charging Specification 1.2*: Released in December 2010.
Several changes and increasing limits including allowing 1.5 A on charging ports for unconfigured devices, allowing High Speed communication while having a current up to 1.5 A and allowing a maximum current of 5 A.

USB 3.0

Main article: USB 3.0

USB 3.0 was released in November 2008. The standard defines a new *SuperSpeed* mode with a signaling speed of 5 Gbit/s and, due to encoding overhead, usable data rate of up to 4 Gbit/s (500 MB/s). A USB 3.0 port is usually colored blue, and is backwards compatible with USB 2.0.

The USB 3.0 Promoter Group announced on 17 November 2008 that the specification of version 3.0 had been completed and had made the transition to the USB Implementers Forum (USB-IF), the managing body of USB specifications. This move effectively opened the specification to hardware developers for implementation in products.

The new *SuperSpeed* bus provides a fourth transfer mode at 5.0 Gbit/s (raw data rate), in addition to the modes supported by earlier versions. The payload throughput is 4 Gbit/s (using 8b/10b encoding), and the specification considers it reasonable to achieve around 3.2 Gbit/s (0.4 GB/s or 400 MB/s), which should increase with future hardware advances. Communication is full-duplex in SuperSpeed transfer mode; in the modes supported previously, by 1.x and 2.0, communication is half-duplex, with direction controlled by the host.

As with previous USB versions, USB 3.0 ports come in low-power and high-power variants, providing 150 mA and 900 mA respectively while simultaneously transmitting data at SuperSpeed rates. Additionally, there is a Battery Charging Specification (Version 1.2 – December 2010), which increases the power handling capability to 1.5 A but does *not* allow concurrent data transmission. The Battery Charging Specification requires that the physical ports themselves be capable of handling 5 A of current Wikipedia:Citation needed but the specification limits the maximum current drawn to 1.5 A.

USB 3.1

A January 2013 press release from the USB group revealed plans to update USB 3.0 to 10 Gbit/s, effectively putting it on par with Thunderbolt by mid-2013. The USB 3.1 specification was released on 31 July 2013,^[4] introducing a faster transfer mode called "SuperSpeed USB 10 Gbps"; its logo features a *Superspeed+* (stylized as *SUPER SPEED+*) caption. The USB 3.1 standard increases the signalling rate to 10 Gbit/s, double that of USB 3.0, and reduces line encoding overhead to just 3% by changing the encoding scheme to 128b/132b. Though, some initial tests demonstrated usable transfer speeds of only 7.2 Gbit/s, suggesting a 30% overall overhead.

The USB 3.1 standard is backward compatible with USB 3.0 and USB 2.0. Using three power profiles of those defined in the USB Power Delivery Specification, it lets devices with larger energy demands request higher currents and supply voltages from compliant hosts – up to 2 A at 5 V (for a power consumption of up to 10 W), and optionally up to 5 A at either 12 V (60 W) or 20 V (100 W).

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.^[5] USB devices are linked in series through hubs. One hub—built into the host controller—is the root hub.

A physical USB device may consist of several logical sub-devices that are referred to as *device functions*. A single device may provide several functions, for example, a webcam (video device function) with a built-in microphone



(audio device function). This kind of device is called a *composite device*. An alternative to this is *compound device*, in which the host assigns each logical device a distinctive address and all logical devices connect to a built-in hub that connects to the physical USB cable.

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 could have up to 32 endpoints (16 IN, 16 OUT), though it's rare to have so many. An endpoint is defined and numbered by the device during initialization (the period after physical connection called "enumeration") and so is relatively permanent, whereas a pipe may be opened and closed.

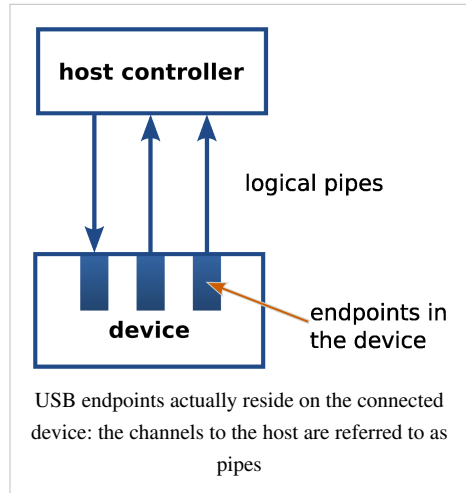
There are two types of pipe: stream and message. A message pipe is bi-directional and is used for *control* transfers. Message pipes are typically used for short, simple commands to the device, and a status response, used, for example, by the bus control pipe number 0. A stream pipe is a uni-directional pipe connected to a uni-directional endpoint that transfers data using an *isochronous*, *interrupt*, or *bulk* transfer:

- *isochronous transfers*: at some guaranteed data rate (often, but not necessarily, as fast as possible) but with possible data loss (e.g., realtime audio or video).
- *interrupt transfers*: devices that need guaranteed quick responses (bounded latency) (e.g., pointing devices and keyboards).
- *bulk transfers*: large sporadic transfers using all remaining available bandwidth, but with no guarantees on bandwidth or latency (e.g., file transfers).

An endpoint of a pipe is addressable with a tuple (*device_address*, *endpoint_number*) as specified in a TOKEN packet that the host sends when it wants to start a data transfer session. If the direction of the data transfer is from the host to the endpoint, an OUT packet (a specialization of a TOKEN packet) having the desired device address and endpoint number is sent by the host. If the direction of the data transfer is from the device to the host, the host sends an IN packet instead. If the destination endpoint is a uni-directional endpoint whose manufacturer's designated direction does not match the TOKEN packet (e.g., the manufacturer's designated direction is IN while the TOKEN packet is an OUT packet), the TOKEN packet is ignored. Otherwise, it is accepted and the data transaction can start. A bi-directional endpoint, on the other hand, accepts both IN and OUT packets.

Endpoints are grouped into *interfaces* and each interface is associated with a single device function. An exception to this is endpoint zero, which is used for device configuration and is not associated with any interface. A single device function composed of independently controlled interfaces is called a *composite device*. A composite device only has a single device address because the host only assigns a device address to a function.

When a USB device is first connected to a USB host, the USB device enumeration process is started. The enumeration starts by sending a reset signal to the USB device. The data rate of the USB device is determined during the reset signaling. After reset, the USB device's information is read by the host and the device is assigned a unique 7-bit address. If the device is supported by the host, the device drivers needed for communicating with the device are loaded and the device is set to a configured state. If the USB host is restarted, the enumeration process is repeated for all connected devices.



The host controller directs traffic flow to devices, so no USB device can transfer any data on the bus without an explicit request from the host controller. In USB 2.0, the host controller polls the bus for traffic, usually in a round-robin fashion. The throughput of each USB port is determined by the slower speed of either the USB port or the USB device connected to the port.

High-speed USB 2.0 hubs contain devices called transaction translators that convert between high-speed USB 2.0 buses and full and low speed buses. When a high-speed USB 2.0 hub is plugged into a high-speed USB host or hub, it operates in high-speed mode. The USB hub then uses either one transaction translator per hub to create a full/low-speed bus routed to all full and low speed devices on the hub, or uses one transaction translator per port to create an isolated full/low-speed bus per port on the hub.

Because there are two separate controllers in each USB 3.0 host, USB 3.0 devices transmit and receive at USB 3.0 data rates regardless of USB 2.0 or earlier devices connected to that host. Operating data rates for earlier devices are set in the legacy manner.

Device classes

The functionality of USB devices is defined by class codes, communicated to the USB host to affect the loading of suitable software driver modules for each connected device. This provides for adaptability and device independence of the host to support new devices from different manufacturers.

Device classes include:

Class	Usage	Description	Examples, or exception
00h	Device	Unspecified ^[6]	Device class is unspecified, interface descriptors are used to determine needed drivers
01h	Interface	Audio	Speaker, microphone, sound card, MIDI
02h	Both	Communications and CDC Control	Modem, Ethernet adapter, Wi-Fi adapter
03h	Interface	Human interface device (HID)	Keyboard, mouse, joystick
05h	Interface	Physical Interface Device (PID)	Force feedback joystick
06h	Interface	Image	Webcam, scanner
07h	Interface	Printer	Laser printer, inkjet printer, CNC machine
08h	Interface	Mass storage (MSC or UMS)	USB flash drive, memory card reader, digital audio player, digital camera, external drive
09h	Device	USB hub	Full bandwidth hub
0Ah	Interface	CDC-Data	Used together with class 02h: communications and CDC control
0Bh	Interface	Smart Card	USB smart card reader
0Dh	Interface	Content security	Fingerprint reader
0Eh	Interface	Video	Webcam
0Fh	Interface	Personal Healthcare	Pulse monitor (watch)
10h	Interface	Audio/Video (AV)	Webcam, TV
DCh	Both	Diagnostic Device	USB compliance testing device
E0h	Interface	Wireless Controller	Bluetooth adapter, Microsoft RNDIS
EFh	Both	Miscellaneous	ActiveSync device
FEh	Interface	Application-specific	IrDA Bridge, Test & Measurement Class (USBTMC), USB DFU (Direct Firmware Update)
FFh	Both	Vendor-specific	Indicates that a device needs vendor-specific drivers

USB mass storage / USB drive

See also: USB mass storage device class, Disk enclosure and External hard disk drive

USB implements connections to storage devices using a set of standards called the USB mass storage device class (MSC or UMS). This was at first intended for traditional magnetic and optical drives and has been extended to support flash drives. It has also been extended to support a wide variety of novel devices as many systems can be controlled with the familiar metaphor of file manipulation within directories. The process of making a novel device look like a familiar device is also known as extension. ^{Wikipedia:Citation needed} The ability to boot a write-locked SD card with a USB adapter is particularly advantageous for maintaining the integrity and non-corruptible, pristine state of the booting medium.

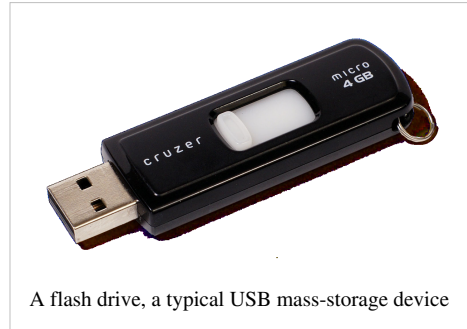
Though most post-Summer 2004 computers can boot from USB mass storage devices, USB is not intended as a primary bus for a computer's internal storage. Buses such as Parallel ATA (PATA or IDE), Serial ATA (SATA), or SCSI fulfill that role in PC class computers. However, USB has one important advantage, in that it is possible to install and remove devices without rebooting the computer (hot-swapping), making it useful for mobile peripherals, including drives of various kinds.

Firstly conceived and still used today for optical storage devices (CD-RW drives, DVD drives, etc.), several manufacturers offer external portable USB hard disk drives, or empty enclosures for disk drives. These offer performance comparable to internal drives, limited by the current number and types of attached USB devices, and by the upper limit of the USB interface (in practice about 30 MB/s for USB 2.0 and potentially 400 MB/s or more^[7] for USB 3.0). These external drives typically include a "translating device" that bridges between a drive's interface to a USB interface port. Functionally, the drive appears to the user much like an internal drive. Other competing standards for external drive connectivity include eSATA, ExpressCard (now at version 2.0), FireWire (IEEE 1394), and most recently Thunderbolt.

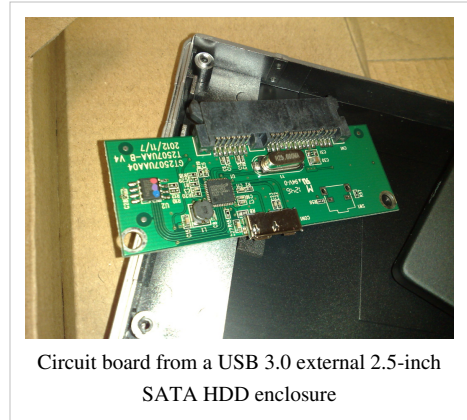
Another use for USB mass storage devices is the portable execution of software applications (such as web browsers and VoIP clients) with no need to install them on the host computer.

Media Transfer Protocol

Media Transfer Protocol (MTP) was designed by Microsoft to give higher-level access to a device's filesystem than USB mass storage, at the level of files rather than disk blocks. It also has optional DRM features. MTP was designed for use with portable media players, but it has since been adopted as the primary storage access protocol of the Android operating system from the version 4.1 Jelly Bean as well as Windows Phone 8 (Windows Phone 7 devices had used the Zune protocol which was an evolution of MTP). The primary reason for this is that MTP does not require exclusive access to the storage device the way UMS does, alleviating potential problems should an Android program request the storage while it is attached to a computer. The main drawback is that MTP is not as well supported outside of Windows operating systems.



A flash drive, a typical USB mass-storage device



Circuit board from a USB 3.0 external 2.5-inch SATA HDD enclosure

Human interface devices

Main article: USB human interface device class

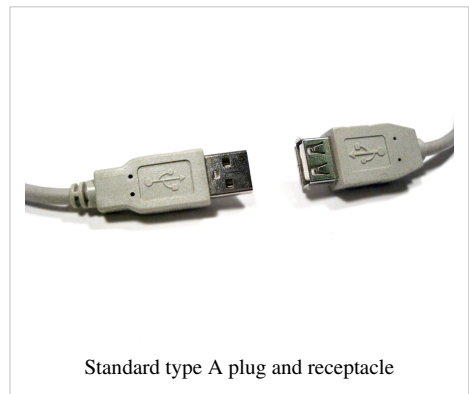
Joysticks, keypads, tablets and other human-interface devices (HIDs) are also progressively migrating from MIDI, and PC game port connectors to USB. [Wikipedia:Citation needed](#)

USB mice and keyboards can usually be used with older computers that have PS/2 connectors with the aid of a small USB-to-PS/2 adapter. For mice and keyboards with dual-protocol support, an adaptor that contains no logic circuitry may be used: the hardware in the USB keyboard or mouse is designed to detect whether it is connected to a USB or PS/2 port, and communicate using the appropriate protocol. Converters also exist that connect PS/2 keyboards and mice (usually one of each) to a USB port. These devices present two HID endpoints to the system and use a microcontroller to perform bidirectional data translation between the two standards.

Connectors and plugs

Connectors properties

The connectors the USB committee specifies support a number of USB's underlying goals, and reflect lessons learned from the many connectors the computer industry has used. The connector mounted on the host or device is called the *receptacle*, and the connector attached to the cable is called the *plug*. The standard purposely defines this to prevent the use of extension cables. [Wikipedia:Citation needed](#) The official USB specification documents also periodically define the term *male* to represent the plug, and *female* to represent the receptacle. [Wikipedia:Citation needed](#)



Standard type A plug and receptacle

Usability and orientation

By design, it is difficult to insert a USB plug into its receptacle incorrectly. The USB specification states that the required USB icon must be embossed on the "topside" of the USB plug, which "...provides easy user recognition and facilitates alignment during the mating process." The specification also shows that the "recommended" "Manufacturer's logo" ("engraved" on the diagram but not specified in the text) is on the opposite side of the USB icon. The specification further states, "The USB Icon is also located adjacent to each receptacle. Receptacles should be oriented to allow the icon on the plug to be visible during the mating process." However, the specification does not consider the height of the device compared to the eye level height of the user, so the side of the cable that is "visible" when mated to a computer on a desk can depend on whether the user is standing or kneeling.



USB extension cord

While it would have been better for usability if the cable could be plugged in with either side up, the original design left this out to make manufacturing as inexpensive as possible. Ajay Bhatt, who was involved in the original USB design team, is working on a new design to make the cable insertable either side up. The new reversible plug is also much smaller than the current USB 3.0 Micro-B connector. It is called *Type-C*, and should be introduced as an

addition to the existing USB 3.1 specification.

Only moderate force is needed to insert or remove a USB cable. USB cables and small USB devices are held in place by the gripping force from the receptacle (without need of the screws, clips, or thumb-turns other connectors have required).

Power-use topology

The standard connectors were deliberately intended to enforce the directed topology of a USB network: type A connectors on host devices that supply power and type B connectors on target devices that draw power. This is intended to prevent users from accidentally connecting two USB power supplies to each other, which could lead to short circuits and dangerously high currents, circuit failures, or even fire. USB does not support cyclic networks and the standard connectors from incompatible USB devices are themselves incompatible.

However, some of this directed topology is lost with the advent of multi-purpose USB connections (such as USB On-The-Go in smartphones, and USB-powered Wi-Fi routers), which require A-to-A, B-to-B, and sometimes Y/splitter cables. See the USB On-The-Go connectors section below, for a more detailed summary description.

Durability

The standard connectors were designed to be robust. Because USB is hot-pluggable, the connectors would be used more frequently, and perhaps with less care, than other connectors. Many previous connector designs were fragile, specifying embedded component pins or other delicate parts that were vulnerable to bending or breaking. The electrical contacts in a USB connector are protected by an adjacent plastic tongue, and the entire connecting assembly is usually protected by an enclosing metal sheath.

The connector construction always ensures that the external sheath on the plug makes contact with its counterpart in the receptacle before any of the four connectors within make electrical contact. The external metallic sheath is typically connected to system ground, thus dissipating damaging static charges. This enclosure design also provides a degree of protection from electromagnetic interference to the USB signal while it travels through the mated connector pair (the only location when the otherwise twisted data pair travels in parallel). In addition, because of the required sizes of the power and common connections, they are made after the system ground but before the data connections. This type of staged make-break timing allows for electrically safe hot-swapping.

The newer Micro-USB receptacles are designed for up to 10,000 cycles of insertion and removal between the receptacle and plug, compared to 1,500 for the standard USB and 5,000 for the Mini-USB receptacle. This is accomplished by adding a locking device and by moving the leaf-spring connector from the jack to the plug, so that the most-stressed part is on the cable side of the connection. This change was made so that the connector on the less expensive cable would bear the most wear instead of the more expensive micro-USB device.

Compatibility

The USB standard specifies relatively loose tolerances for compliant USB connectors to minimize physical incompatibilities in connectors from different vendors. To address a weakness present in some other connector standards, the USB specification also defines limits to the size of a connecting device in the area around its plug. This was done to prevent a device from blocking adjacent ports due to the size of the cable strain relief mechanism (usually molding integral with the cable outer insulation) at the connector. Compliant devices must either fit within the size restrictions or support a compliant extension cable that does.

In general, cables have only plugs, and hosts and devices have only receptacles. Hosts almost universally have type-A receptacles, and devices one or another type-B variety. Type-A plugs mate only with type-A receptacles, and type-B with type-B; they are deliberately physically incompatible. However, an extension to USB standard specification called USB On-The-Go allows a single port to act as either a host or a device—chosen by which end of the cable plugs into the receptacle on the unit. Even after the cable is hooked up and the units are communicating, the

two units may "swap" ends under program control. This capability is meant for units such as PDAs in which the USB link might connect to a PC's host port as a device in one instance, yet connect as a host itself to a keyboard and mouse device in another instance.

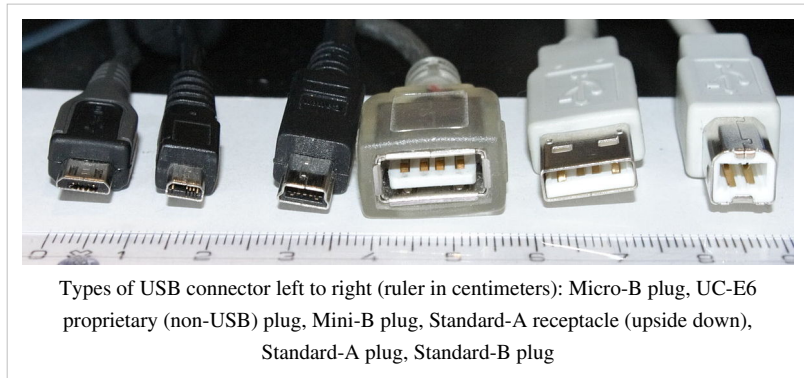
USB 3.0 connectors

Main article: USB 3.0 backward compatibility

Type A plugs and receptacles from both USB 3.0 and USB 2.0 are designed to interoperate. Type B plugs and receptacles in USB 3.0 are somewhat larger than those in USB 2.0; thus, USB 2.0 Type B plugs can fit into USB 3.0 Type B receptacles, while the opposite is not possible.

Connector types

There are several types of USB connector, including some that have been added while the specification progressed. The original USB specification detailed Standard-A and Standard-B plugs and receptacles; the B connector was necessary so that cabling could be plug ended at both ends and still prevent users from connecting one computer receptacle to another. The first engineering change notice to the USB 2.0 specification added Mini-B plugs and receptacles.



The first engineering change notice to the USB 2.0 specification added Mini-B plugs and receptacles.

The data connectors in the Standard-A plug are actually recessed in the plug as compared to the outside power connectors. This permits the power to connect first, which prevents data errors by allowing the device to power up first and then transfer the data. Some devices operate in different modes depending on whether the data connection is made. This difference in connection can be exploited by inserting the connector only partially. For example, some battery-powered MP3 players switch into file transfer mode and cannot play MP3 files while a USB plug is fully inserted, but can be operated in MP3 playback mode using USB power by inserting the plug only part way so that the power slots make contact while the data slots do not. This enables those devices to be operated in MP3 playback mode while getting power from the cable. Wikipedia:No original research

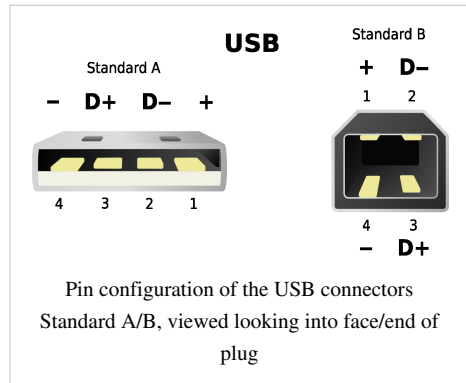
To reliably enable a charge-only feature, modern USB accessory peripherals now include charging cables that provide power connections to the host port but no data connections, and both home and vehicle charging docks are available that supply power from a converter device and do not include a host device and data pins, allowing any capable USB device to charge or operate from a standard USB cable.

Standard connectors

The USB 2.0 Standard-A type of USB plug is a flattened rectangle that inserts into a "downstream-port" receptacle on the USB host, or a hub, and carries both power and data. This plug is frequently seen on cables that are permanently attached to a device, such as one connecting a keyboard or mouse to the computer via USB connection.

USB connections eventually wear out as the connection loosens through repeated plugging and unplugging. The lifetime of a USB-A male connector is approximately 1,500 connect/disconnect cycles.

A Standard-B plug—which has a square shape with beveled exterior corners—typically plugs into an "upstream receptacle" on a device that uses a removable cable, e.g., a printer. On some devices, the Type B receptacle has no data connections, being used solely for accepting power from the upstream device. This two-connector-type scheme (A/B) prevents a user from accidentally creating an electrical loop.



Mini and Micro connectors

Various connectors have been used for smaller devices such as digital cameras, smartphones, and tablet computers. These include the now-deprecated (i.e. de-certified but standardized) Mini-A and Mini-AB connectors (Mini-B connectors are still supported but not OTG (On The Go, i.e. mobile) compliant). The Mini-B USB connector was standard for transferring data to and from the early data phones and PDAs, such as Blackberry models.

The Mini-A and Mini-B plugs are approximately 3 by 7 mm. The micro-USB plugs have a similar width and approximately half the thickness, enabling their integration into thinner portable devices. The micro-A connector is 6.85 by 1.8 mm with a maximum overmold size of 11.7 by 8.5 mm. The micro-B connector is 6.85 by 1.8 mm with a maximum overmold size of 10.6 by 8.5 mm.



The Micro-USB connector was announced by the USB-IF on 4 January 2007. The Micro B USB connector has a maximum current rating of either 1 A per pin, or 1.8 A for pins 1 and 5, and 0.5 A for pins 2, 3, and 4.^[8] The Mini-A connector and the Mini-AB receptacle connector were deprecated on 23 May 2007. While many currently available devices and cables still use Mini plugs, the newer Micro connectors are being widely adopted and as of December 2010, they are the most widely used. Wikipedia:Citation needed The thinner micro connectors are intended to replace the Mini plugs in new devices including smartphones, personal digital assistants, and cameras.^[9] The Micro plug design is rated for at least 10,000 connect–disconnect cycles—



significantly more than the Mini plug design. It is also designed to reduce the mechanical wear on the device; instead the easier-to-replace cable is designed to bear the mechanical wear of connection and disconnection. The *Universal Serial Bus Micro-USB Cables and Connectors Specification* details the mechanical characteristics of

Micro-A plugs, Micro-AB receptacles (which accept both Micro-A and Micro-B plugs), and Micro-B plugs and receptacles, along with a Standard-A receptacle to Micro-A plug adapter.

The cellular phone carrier group, Open Mobile Terminal Platform (OMTP) in 2007 endorsed Micro-USB as the standard connector for data and power on mobile devices. In addition, on 22 October 2009 the International Telecommunication Union (ITU) has also announced that it had embraced micro-USB as the *Universal Charging Solution* its "energy-efficient one-charger-fits-all new mobile phone solution", and added: "Based on the Micro-USB interface, UCS chargers also include a 4-star or higher efficiency rating— up to three times more energy-efficient than an unrated charger."



USB Mini A (left) and USB Mini B (right) plugs

The European Standardisation Bodies CEN, CENELEC and ETSI (independent of the OMTP/GSMA proposal) defined a common External Power Supply (EPS) for use with smartphones sold in the EU based on micro-USB. 14 of the world's largest mobile phone manufacturers signed the EU's common EPS Memorandum of Understanding (MoU). Apple Inc., one of the original MoU signers, makes micro-USB adapters available – as permitted in the Common EPS MoU – for its iPhones equipped with Apple's proprietary 30 pin dock connector or (later) "Lightning" connector.

USB On-The-Go connectors

Main article: USB On-The-Go

All current USB On-The-Go (OTG) devices are required to have one, and only one, USB connector – a Micro-AB receptacle. Non-OTG compliant devices are not allowed to use the micro-AB receptacle, due to power supply shorting hazards on the V_{BUS} line. The micro-AB receptacle is capable of accepting both Micro-A and Micro-B plugs, attached to any of the legal cables and adapters as defined in revision 1.01 of the Micro-USB specification. Prior to the development of Micro-USB, USB On-The-Go devices were required to use Mini-AB receptacles to perform the equivalent job.






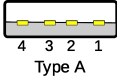


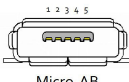
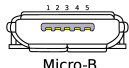
The OTG device with the A-plug inserted is called the A-device and is responsible for powering the USB interface when required and by default assumes the role of host. The OTG device with the B-plug inserted is called the B-device and by default assumes the role of peripheral. An OTG device with no plug inserted defaults to acting as a B-device. If an application on the B-device requires the role of host, then the Host Negotiation Protocol (HNP) is used to temporarily transfer the host role to the B-device.

OTG devices attached either to a peripheral-only B-device or a standard/embedded host have their role fixed by the cable, since in these scenarios it is only possible to attach the cable one way. [Wikipedia:Citation needed](#)

Host and Device interface receptacles






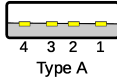




Connectors (receptacles and plugs) mating matrix is displayed below, with the following notes:

- All pin numbers, as depicted below, are individually numbered for each connector. In other words, the same pin numbers can have differently assigned functions (V_{CC} , D+, D−, GND or ID) between different connectors. For example, pin numbered below as "4" on the Mini-B receptacle is used for On-The-Go host/client identification, while pin numbered below as "4" on the Type-A plug is connected to GND.
- Type-A and Type-B receptacles are depicted below as viewed from their back sides (opposite from their mating surfaces), causing the pin numbers to be switched around (mirrored) when compared to the usual "top view" numbering scheme. Here, "top view" refers to viewing the connector from its mating surface.

Receptacle (images not to scale)	Plug (images not to scale)					
	 Type A	 Type B	Mini-A	 Mini-B	 Micro-A	 Micro-B
 Type A	Yes	No	No	No	No	No
 Type B	No	Yes	No	No	No	No
Mini-A	No	No	Deprecated	No	No	No
Mini-AB	No	No	Deprecated	Deprecated	No	No
 Mini-B	No	No	No	Yes	No	No
 Micro-AB	No	No	No	No	Yes	Yes
 Micro-B	No	No	No	No	No	Yes

Cable plugs (USB 1.x/2.0)

USB cables exist with various combinations of plugs on each end of the cable, as displayed below. Notes listed in the section above apply here as well.

Plug (images not to scale)	Plug (images not to scale)					
	 Type A	 Type B	Mini-A	 Mini-B	 Micro-A	 Micro-B
 Type A	Non-standard	Yes	Non-standard	Yes	Non-standard	Yes
 Type B	Yes	No	Deprecated	No	Non-standard	No
Mini-A	Non-standard	Deprecated	No	Deprecated	No	Non-standard
 Mini-B	Yes	No	Deprecated	No	Non-standard	No
 Micro-A	Non-standard	Non-standard	No	Non-standard	No	Yes
 Micro-B	Yes	No	Non-standard	No	Yes	No

Non-standard

Existing for specific proprietary purposes, and in most cases not inter-operable with USB-IF compliant equipment. However, there do exist compliant A-to-A cables with a circuit in the middle that behaves as a pair of devices, such as the Easy Transfer Cable.

In addition to the above cable assemblies comprising two plugs, an "adapter" cable with a Micro-A plug and a Standard-A receptacle is compliant with USB specifications. Other combinations of connectors are not compliant.

Deprecated

Some older devices and cables with Mini-A connectors have been certified by USB-IF. The Mini-A connector is obsolete: no new Mini-A connectors and neither Mini-A nor Mini-AB receptacles will be certified.

Cable plugs (USB 3.0)

See also: USB 3.0 Connectors

USB 3.0 introduced a new Micro-B cable plug, see photo on the right. It consists of a standard USB 1.x/2.0 Micro-B cable plug, with additional 5-pin plug "stacked" on side of it. That way, USB 3.0 Micro-A host connector preserved its backwards compatibility with the USB 1.x/2.0 Micro-B cable plugs.



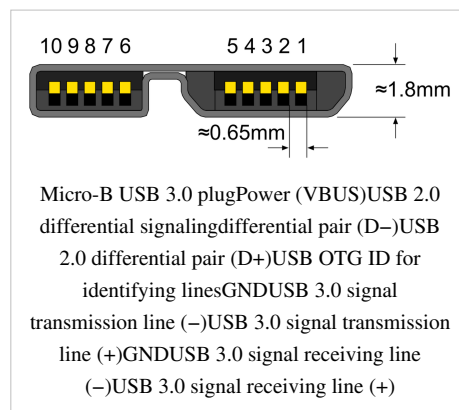
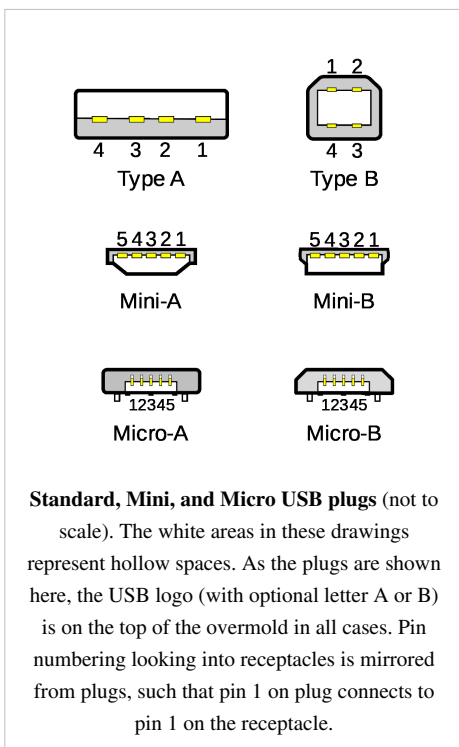
USB 3.0 Micro-B plug

Pinouts

See also: USB 3.0 Pinouts

USB is a serial bus, using four shielded wires for the USB 2.0 variant: two for power (V_{BUS} and GND), and two for differential data signals (labelled as D+ and D- in pinouts). Non-Return-to-Zero Inverted (NRZI) encoding scheme is used for transferring data, with a sync field to synchronise the host and receiver clocks. D+ and D- signals are transmitted on a twisted pair, providing half-duplex data transfers for USB 2.0.

There are additional wires present in the USB 3.0 variant, consisting of two twisted pairs and providing various improvements, including full-duplex data transfers at increased speed.



USB 1.x/2.0 standard pinout

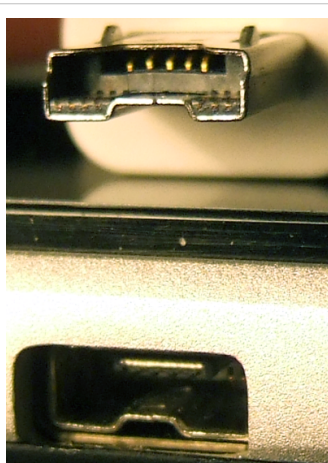
Pin	Name	Cable color	Description
1	V _{BUS}	Red (or Orange)	+5 V
2	D-	White (or Gold)	Data -
3	D+	Green	Data +
4	GND		Ground

USB 1.x/2.0 Mini/Micro pinout

Pin	Name	Cable color	Description
1	V _{BUS}	Red	+5 V
2	D-	White	Data -
3	D+	Green	Data +
4	ID	N/A	Permits distinction of a host connection from device connection: <ul style="list-style-type: none"> • host: connected to the signal ground • device: not connected
5	GND		Signal ground

Proprietary connectors and formats

Manufacturers of personal electronic devices might not include a USB standard connector on their product for technical or marketing reasons. Some manufacturers provide proprietary cables that permit their devices to physically connect to a USB standard port. Full functionality of proprietary ports and cables with USB standard ports is not assured; for example, some devices only use the USB connection for battery charging and do not implement any data transfer functions.



HTC ExtMicro USB port and connector



Nokia Pop-Port connector

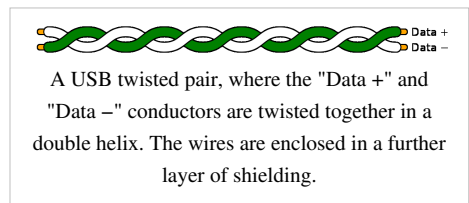
Colors

Color	Description
Black or white	USB 1.x or USB 2.0
Blue	USB 3.0
Yellow or red (ports only)	High current and/or sleep-and-charge

USB ports and connectors are often color-coded to distinguish their different functions and USB versions. These colors are not part of the USB specification and can vary between manufacturers;Wikipedia:Citation needed for example, USB 3.0 specification mandates appropriate color-coding while it only recommends blue inserts for Standard-A USB 3.0 connectors and plugs.

Cabling

The data cables for USB 1.x and USB 2.x use a twisted pair to reduce noise and crosstalk. USB 3.0 cables contain twice as many wires as USB 2.x to support SuperSpeed data transmission, and are thus larger in diameter.



The USB 1.1 Standard specifies that a standard cable can have a maximum length of 5 meters with devices operating at Full Speed (12 Mbit/s), and a maximum length of 3 meters with devices operating at Low Speed (1.5 Mbit/s).

USB 2.0 provides for a maximum cable length of 5 meters for devices running at Hi Speed (480 Mbit/s). The primary reason for this limit is the maximum allowed round-trip delay of about 1.5 μs. If USB host commands are unanswered by the USB device within the allowed time, the host considers the command lost. When adding USB device response time, delays from the maximum number of hubs added to the delays from connecting cables, the maximum acceptable delay per cable amounts to 26 ns. The USB 2.0 specification requires that cable delay be less than 5.2 ns per meter (192 000 km/s, which is close to the maximum achievable transmission speed for standard copper wire).

The USB 3.0 standard does not directly specify a maximum cable length, requiring only that all cables meet an electrical specification: for copper cabling with AWG 26 wires the maximum practical length is 3 meters (9.8 ft).

Power

USB Power standards

Specification	Current	Voltage	Power
USB 1.0	150 mA	5 V	0.75 W
USB 2.0	500 mA ^[a]	5 V	2.5 W
USB 3.0	900 mA ^[b]	5 V	4.5 W
USB 3.1	2 A	5 V	10 W
	5 A	12 V	60 W
	5 A	20 V	100 W
USB Battery Charging	0.5–1.5 A	5 V	2.5–7.5 W

USB Power Delivery	2 A	5 V	10 W
	3 A	12 V	36 W
	3 A	20 V	60 W
	5 A	20 V	100 W
a. Up to five unit loads; in USB 2.0, unit load is 100 mA. b. Up to six unit loads; in USB 3.0, unit load is 150 mA.			

The USB 1.x and 2.0 specifications provide a 5 V supply on a single wire to power connected USB devices. The specification provides for no more than 5.25 V and no less than 4.75 V ($5\text{ V} \pm 5\%$) between the positive and negative bus power lines (V_{BUS} voltage). For USB 3.0, the voltage supplied by low-powered hub ports is 4.45–5.25 V.

A unit load is defined as 100 mA in USB 2.0, and 150 mA in USB 3.0. A device may draw a maximum of five unit loads (500 mA) from a port in USB 2.0, or six unit loads (900 mA) in USB 3.0. There are two types of device: low-power and high-power. A low-power device (such as a USB HID) draws at most one-unit load, with minimum operating voltage of 4.4 V in USB 2.0, and 4 V in USB 3.0. A high-power device draws, at most, the maximum number of unit loads the standard permits. Every device functions initially as low-power (including high-power functions during their low-power enumeration phases), but may request high-power, and get it if available on the providing bus.



Y-shaped USB 3.0 cable; with such a cable, a device can draw power from two USB ports simultaneously

Some devices, such as high-speed external disk drives, require more than 500 mA of current and therefore may have power issues if powered from just one USB 2.0 port: erratic function, failure to function, or overloading/damaging the port. Such devices may come with an external power source or a Y-shaped cable that has two USB connectors (one for power and data, the other for power only) to plug into a computer. With such a cable, a device can draw power from two USB ports simultaneously. However, USB compliance specification states that "use of a 'Y' cable (a cable with two A-plugs) is prohibited on any USB peripheral", meaning that "if a USB peripheral requires more power than allowed by the USB specification to which it is designed, then it must be self-powered."

A bus-powered hub initializes itself at one-unit load and transitions to maximum unit loads after it completes hub configuration. Any device connected to the hub draws one-unit load regardless of the current draw of devices connected to other ports of the hub (i.e., one device connected on a four-port hub draws only one-unit load despite the fact that more unit loads are being supplied to the hub).Wikipedia:Citing sources#What information to include

A self-powered hub supplies maximum supported unit loads to any device connected to it. In addition, the V_{BUS} presents one-unit load upstream for communication if parts of the Hub are powered down.Wikipedia:Please clarifyWikipedia:Citing sources#What information to include

Charging ports

The *USB Battery Charging Specification Revision 1.1* (released in 2007) defines a new type of USB port, called the *charging port*. Contrary to the *standard downstream port*, for which current draw by a connected portable device can exceed 100 mA only after digital negotiation with the host or hub, a charging port can supply currents between 500 mA and 1.5 A without the digital negotiation. A charging port supplies up to 500 mA at 5 V, up to the rated current at 3.6 V or more, and drop its output voltage if the portable device attempts to draw more than the rated current. The charger port may shut down if the load is too high.

Two types of charging port exist: the *charging downstream port* (CDP), supporting data transfers as well, and the *dedicated charging port* (DCP), without data support. A portable device can recognize the type of USB port; on a dedicated charging port, the D+ and D- pins are shorted with a resistance not exceeding 200 ohms, while charging downstream ports provide additional detection logic so their presence can be determined by attached devices.

With charging downstream ports, current passing through the thin ground wire may interfere with high-speed data signals; therefore, current draw may not exceed 900 mA during high-speed data transfer. A dedicated charge port may have a rated current between 500 and 1,500 mA. For all charging ports, there is maximum current of 5 A, as long as the connector can handle the current (standard USB 2.0 A-connectors are rated at 1.5 A).

Before the battery charging specification was defined, there was no standardized way for the portable device to inquire how much current was available. For example, Apple's iPod and iPhone chargers indicate the available current by voltages on the D- and D+ lines. When $D+ = D- = 2.0$ V, the device may pull up to 500 mA. When $D+ = 2.0$ V and $D- = 2.8$ V, the device may pull up to 1 A of current. When $D+ = 2.8$ V and $D- = 2.0$ V, the device may pull up to 2 A of current.

Dedicated charging ports can be found on USB power adapters that convert utility power or another power source (e.g., a car's electrical system) to run attached devices and battery packs. On a host (such as a laptop computer) with both standard and charging USB ports, the charging ports should be labeled as such.

To support simultaneous charge and data communication, even if the communication port does not support charging a demanding device, so-called *accessory charging adapters* (ACA) are introduced. By using an accessory charging adapter, a device providing a single USB port can be attached to both a charger, and another USB device at the same time.

The *USB Battery Charging Specification Revision 1.2* (released in 2010) makes clear that there are safety limits to the rated current at 5 A coming from USB 2.0. On the other hand, several changes are made and limits are increasing including allowing 1.5 A on charging downstream ports for unconfigured devices, allowing high speed communication while having a current up to 1.5 A, and allowing a maximum current of 5 A. Also, revision 1.2 removes support for USB ports type detection via resistive detection mechanisms.



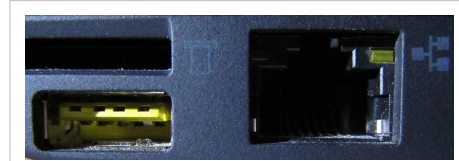
Small gadget providing voltage and current readouts for devices charged over USB.

Sleep-and-charge ports

Sleep-and-charge USB ports can be used to charge electronic devices even when the computer is switched off. Normally, when a computer is powered off, the USB ports are powered down. This prevents phones and other devices from being able to charge unless the computer is powered on. Sleep-and-charge USB ports remain powered even when the computer is off. On laptops, charging devices from the USB port when it is not being powered from AC drains the laptop battery faster; most laptops have a facility to stop charging if their own battery charge level gets too low. Desktop machines need to remain plugged into AC power for Sleep-and-charge to work.

These ports are found colored differently (mostly red or yellow). On Dell laptops, the port is marked with the standard USB symbol with an added lightning bolt icon on the right side. Dell calls this feature "PowerShare."

On Acer Inc. and Packard Bell laptops, sleep-and-charge USB ports are marked with a non-standard symbol (the letters "USB" over a drawing of a battery); the feature is simply called "Power-off USB".



A yellow USB port denoting sleep-and-charge

Mobile device charger standards

In China

As of 14 June 2007[3], all new mobile phones applying for a license in China are required to use a USB port as a power port for battery charging.^[10] This was the first standard to use the convention of shorting D+ and D-.

OMTP/GSMA Universal Charging Solution

In September 2007, the Open Mobile Terminal Platform group (a forum of mobile network operators and manufacturers such as Nokia, Samsung, Motorola, Sony Ericsson and LG) announced that its members had agreed on micro-USB as the future common connector for mobile devices.

The GSM Association (GSMA) followed suit on 17 February 2009, and on 22 April 2009, this was further endorsed by the CTIA – The Wireless Association, with the International Telecommunication Union (ITU) announcing on 22 October 2009 that it had also embraced the Universal Charging Solution as its "energy-efficient one-charger-fits-all new mobile phone solution", and added: "Based on the Micro-USB interface, UCS chargers will also include a 4-star or higher efficiency rating—up to three times more energy-efficient than an unrated charger."

EU Smartphone Power Supply Standard

Main article: Common External Power Supply

In June 2009, many of the world's largest mobile phone manufacturers signed an EC-sponsored Memorandum of Understanding (MoU), agreeing to make most data-enabled mobile phones marketed in the European Union compatible with a common External Power Supply (EPS). The EU's common EPS specification (EN 62684:2010) references the USB Battery Charging standard and is similar to the GSMA/OMTP and Chinese charging solutions. In January 2011, the International Electrotechnical Commission (IEC) released its version of the (EU's) common EPS standard as IEC 62684:2011.



The Micro-USB interface is commonly found on chargers for mobile phones

Non-standard devices

Some USB devices require more power than is permitted by the specifications for a single port. This is common for external hard and optical disc drives, and generally for devices with motors or lamps. Such devices can use an external power supply, which is allowed by the standard, or use a dual-input USB cable, one input of which is used for power and data transfer, the other solely for power, which makes the device a non-standard USB device. Some USB ports and external hubs can, in practice, supply more power to USB devices than required by the specification but a standard-compliant device may not depend on this.

In addition to limiting the total average power used by the device, the USB specification limits the inrush current (i.e., that used to charge decoupling and filter capacitors) when the device is first connected. Otherwise, connecting a device could cause problems with the host's internal power. USB devices are also required to automatically enter ultra low-power suspend mode when the USB host is suspended. Nevertheless, many USB host interfaces do not cut off the power supply to USB devices when they are suspended. Wikipedia:Citation needed

Some non-standard USB devices use the 5 V power supply without participating in a proper USB network, which negotiates power draw with the host interface. These are usually called *USB decorations*. Wikipedia:Citation needed Examples include USB-powered keyboard lights, fans, mug coolers and heaters, battery chargers, miniature vacuum cleaners, and even miniature lava lamps. In most cases, these items contain no digital circuitry, and thus are not standard compliant USB devices. This may cause problems with some computers, such as drawing too much current and damaging circuitry. Prior to the Battery Charging Specification, the USB specification required that devices connect in a low-power mode (100 mA maximum) and communicate their current requirements to the host, which then permits the device to switch into high-power mode.

Some devices, when plugged into charging ports, draw even more power (10 watts or 2.1 Amps) than the Battery Charging Specification allows. The iPad and MiFi 2200 are two such devices. Barnes & Noble NOOK Color devices also require a special charger that runs at 1.9 Amps.

USB Power Delivery

In July 2012 the USB Promoters Group announced the finalization of the USB Power Delivery ("PD") specification, an extension that specifies using certified "PD aware" USB cables with standard USB type A/B connectors to deliver up to 100 W of power at 20 V. For PD-aware cables with USB-micro B/AB connectors the maximum power supported is up to 60 W at 20 V, 36 W at 12 V and 10 W at 5 V. In all cases, either host-to-device or device-to-host configurations are supported.

The intent is to permit uniformly charging laptops, tablets, USB-powered disks and similarly higher power consumer electronics, as a natural extension of existing European and Chinese mobile telephone charging standards. This may also affect the way electric power used for small devices is transmitted and used in both residential and public buildings.



USB-powered mini fans



USB vacuum cleaner novelty device

PoweredUSB

Main article: PoweredUSB

PoweredUSB is a proprietary extension that adds four additional pins supplying up to 6 A at either 5 V, 12 V, or 24 V. It is commonly used in point of sale systems to power peripherals such as barcode readers, credit card terminals, and printers.

Signaling

USB allows the following signaling rates (the terms *speed* and *bandwidth* are used interchangeably, while "high-" is alternatively written as "hi-"):

- A *low-speed* (USB 1.0) rate of 1.5 Mbit/s is defined by USB 1.0. It is very similar to full-bandwidth operation except each bit takes 8 times as long to transmit. It is intended primarily to save cost in low-bandwidth human interface devices (HID) such as keyboards, mice, and joysticks.
- The *full-speed* (USB 1.1) rate of 12 Mbit/s is the basic USB data rate defined by USB 1.0. All USB hubs can operate at this speed.
- A *high-speed* (USB 2.0) rate of 480 Mbit/s was introduced in 2001. All hi-speed devices are capable of falling back to full-bandwidth operation if necessary; i.e., they are backward compatible with USB 1.1. Connectors are identical for USB 2.0 and USB 1.x.
- A *SuperSpeed* (USB 3.0) rate of 5.0 Gbit/s. The written USB 3.0 specification was released by Intel and its partners in August 2008. The first USB 3.0 controller chips were sampled by NEC in May 2009, and the first products using the USB 3.0 specification arrived in January 2010. USB 3.0 connectors are generally backwards compatible, but include new wiring and full duplex operation.

USB signals are transmitted on a twisted-pair data cable with $90\Omega \pm 15\%$ characteristic impedance, labeled D+ and D-. Prior to USB 3.0, these collectively use half-duplex differential signaling to reduce the effects of electromagnetic noise on longer lines. Transmitted signal levels are 0.0 to 0.3 volts for low and 2.8 to 3.6 volts for high in full-bandwidth and low-bandwidth modes, and -10 to 10 mV for low and 360 to 440 mV for high in hi-bandwidth mode. In FS mode, the cable wires are not terminated, but the HS mode has termination of $45\ \Omega$ to ground, or $90\ \Omega$ differential to match the data cable impedance, reducing interference due to signal reflections. USB 3.0 introduces two additional pairs of shielded twisted wire and new, mostly interoperable contacts in USB 3.0 cables, for them. They permit the higher data rate, and full duplex operation.

A USB connection is always between a host or hub at the "A" connector end, and a device or hub's "upstream" port at the other end. Originally, this was a "B" connector, preventing erroneous loop connections, but additional upstream connectors were specified, and some cable vendors designed and sold cables that permitted erroneous connections (and potential damage to circuitry). USB interconnections are not as fool-proof or as simple as originally intended.

The host includes $15\ \text{k}\Omega$ pull-down resistors on each data line. When no device is connected, this pulls both data lines low into the so-called "single-ended zero" state (SE0 in the USB documentation), and indicates a reset or disconnected connection.

A USB device pulls one of the data lines high with a $1.5\ \text{k}\Omega$ resistor. This overpowers one of the pull-down resistors in the host and leaves the data lines in an idle state called "J". For USB 1.x, the choice of data line indicates of what signal rates the device is capable; full-bandwidth devices pull D+ high, while low-bandwidth devices pull D- high. The "k" state is just the opposite polarity to the "j" state.

Transmission rates

The theoretical maximum data rate in USB 2.0 is 480 Mbit/s (60 MB/s) per controller and is shared amongst all attached devices. Some chipset manufacturers overcome this bottleneck by providing multiple USB 2.0 controllers within the southbridge.

Typical hi-speed USB hard drives can be written to at rates around 25–30 MB/s, and read from at rates of 30–42 MB/s, according to routine testing done by CNet. This is 70% of the total bandwidth available. Mask Tests, also known as Eye Diagram Tests, are used to determine the quality of a signal in the time domain. They are defined in the referenced document as part of the electrical test description for the high-speed (HS) mode at 480 Mbit/s.

According to a USB-IF chairman, "at least 10 to 15 percent of the stated peak 60 MB/s (480 Mbit/s) of Hi-Speed USB goes to overhead—the communication protocol between the card and the peripheral. Overhead is a component of all connectivity standards". Tables illustrating the transfer limits are shown in Chapter 5 of the USB spec.

For isochronous devices like audio streams, the bandwidth is constant, and reserved exclusively for a given device. The bus bandwidth therefore only has an effect on the number of channels that can be sent at a time, not the "speed" or latency of the transmission.

Latency

For USB1 low-speed (1.5 Mbit/s) and full-speed (12 Mbit/s) devices the shortest time for a transaction in one direction is 1 ms. USB2 high-speed (480 Mbit/s) uses transactions within each micro frame (125 μs) where using 1-byte interrupt packet results in a minimal response time of 940 ns. 4-byte interrupt packet results in 984 ns.

Communication

During USB communication data is transmitted as packets. Initially, all packets are sent from the host, via the root hub and possibly more hubs, to devices. Some of those packets direct a device to send some packets in reply.

After the sync field, all packets are made of 8-bit bytes, transmitted least-significant bit first. The first byte is a packet identifier (PID) byte. The PID is actually 4 bits; the byte consists of the 4-bit PID followed by its bitwise complement. This redundancy helps detect errors. (Note also that a PID byte contains at most four consecutive 1 bits, and thus never needs bit-stuffing, even when combined with the final 1 bit in the sync byte. However, trailing 1 bits in the PID may require bit-stuffing within the first few bits of the payload.)

USB PID bytes

Type	PID value (msb-first)	Transmitted byte (lsb-first)	Name	Description
<i>Reserved</i>	0000	0000 1111		
Token	1000	0001 1110	SPLIT	High-bandwidth (USB 2.0) split transaction
	0100	0010 1101	PING	Check if endpoint can accept data (USB 2.0)
Special	1100	0011 1100	PRE	Low-bandwidth USB preamble
Handshake			ERR	Split transaction error (USB 2.0)
	0010	0100 1011	ACK	Data packet accepted
	1010	0101 1010	NAK	Data packet not accepted; please retransmit
	0110	0110 1001	NYET	Data not ready yet (USB 2.0)
	1110	0111 1000	STALL	Transfer impossible; do error recovery

Token	0001	1000 0111	OUT	Address for host-to-device transfer
	1001	1001 0110	IN	Address for device-to-host transfer
	0101	1010 0101	SOF	Start of frame marker (sent each ms)
	1101	1011 0100	SETUP	Address for host-to-device control transfer
Data	0011	1100 0011	DATA0	Even-numbered data packet
	1011	1101 0010	DATA1	Odd-numbered data packet
	0111	1110 0001	DATA2	Data packet for high-bandwidth isochronous transfer (USB 2.0)
	1111	1111 0000	MDATA	Data packet for high-bandwidth isochronous transfer (USB 2.0)

Packets come in three basic types, each with a different format and CRC (cyclic redundancy check):

Handshake packets

Handshake packets consist of a PID byte, and are generally sent in response to data packets. The three basic types are *ACK*, indicating that data was successfully received, *NAK*, indicating that the data cannot be received and should be retried, and *STALL*, indicating that the device has an error condition and cannot transfer data until some corrective action (such as device initialization) occurs.

USB 2.0 added two additional handshake packets: *NYET* and *ERR*. *NYET* indicates that a split transaction is not yet complete. A *NYET* packet also tells the host that the receiver has accepted a data packet, but cannot accept any more due to full buffers. The host then sends *PING* packets and continues with data packets once the device *ACK*'s the *PING*. The *ERR* handshake indicates that a split transaction failed.

The only handshake packet the USB host may generate is *ACK*. If it is not ready to receive data, it should not instruct a device to send.

Token packets

Token packets consist of a PID byte followed by two payload bytes: 11 bits of address and a 5-bit CRC. Tokens are only sent by the host, never a device.

IN and *OUT* tokens contain a 7-bit device number and 4-bit function number (for multifunction devices) and command the device to transmit *DATAx* packets, or receive the following *DATAx* packets, respectively.

An *IN* token expects a response from a device. The response may be a *NAK* or *STALL* response, or a *DATAx* frame. In the latter case, the host issues an *ACK* handshake if appropriate.

An *OUT* token is followed immediately by a *DATAx* frame. The device responds with *ACK*, *NAK*, *NYET*, or *STALL*, as appropriate.

SETUP operates much like an *OUT* token, but is used for initial device setup. It is followed by an 8-byte *DATA0* frame with a standardized format.

Every millisecond (12000 full-bandwidth bit times), the USB host transmits a special *SOF* (start of frame) token, containing an 11-bit incrementing frame number in place of a device address. This is used to synchronize isochronous and interrupt data transfers. High-bandwidth USB 2.0 devices receive 7 additional *SOF* tokens per frame, each introducing a 125 μ s "microframe" (60000 high-bandwidth bit times each).

USB 2.0 added a *PING* token, which asks a device if it is ready to receive an *OUT/DATA* packet pair. The device responds with *ACK*, *NAK*, or *STALL*, as appropriate. This avoids the need to send the *DATA* packet if the device knows that it will just respond with *NAK*.

USB 2.0 also added a larger 3-byte *SPLIT* token with a 7-bit hub number, 12 bits of control flags, and a 5-bit CRC. This is used to perform split transactions. Rather than tie up the high-bandwidth USB bus sending data to a slower USB device, the nearest high-bandwidth capable hub receives a *SPLIT* token followed by one or two USB packets at

high bandwidth, performs the data transfer at full or low bandwidth, and provides the response at high bandwidth when prompted by a second SPLIT token.

Data packets

A data packet consists of the PID followed by 0–1,023 bits of data payload (up to 1,024 in high bandwidth, at most 8 at low bandwidth), and a 16-bit CRC.

There are two basic forms of data packet, *DATA0* and *DATA1*. A data packet must always be preceded by an address token, and is usually followed by a handshake token from the receiver back to the transmitter. The two packet types provide the 1-bit sequence number required by Stop-and-wait ARQ. If a USB host does not receive a response (such as an ACK) for data it has transmitted, it does not know if the data was received or not; the data might have been lost in transit, or it might have been received but the handshake response was lost.

To solve this problem, the device keeps track of the type of DATAx packet it last accepted. If it receives another DATAx packet of the same type, it is acknowledged but ignored as a duplicate. Only a DATAx packet of the opposite type is actually received.

If the data is corrupted while transmitted or received, the CRC check fails. When this happens, the receiver does not generate an ACK, which makes the sender resend the packet.

When a device is reset with a SETUP packet, it expects an 8-byte DATA0 packet next.

USB 2.0 added *DATA2* and *MDATA* packet types as well. They are used only by high-bandwidth devices doing high-bandwidth isochronous transfers that must transfer more than 1024 bits per 125 μs microframe (8,192 kB/s).

PRE packet

Low-bandwidth devices are supported with a special PID value, *PRE*. This marks the beginning of a low-bandwidth packet, and is used by hubs that normally do not send full-bandwidth packets to low-bandwidth devices. Since all PID bytes include four 0 bits, they leave the bus in the full-bandwidth K state, which is the same as the low-bandwidth J state. It is followed by a brief pause, during which hubs enable their low-bandwidth outputs, already idling in the J state. Then a low-bandwidth packet follows, beginning with a sync sequence and PID byte, and ending with a brief period of SE0. Full-bandwidth devices other than hubs can simply ignore the PRE packet and its low-bandwidth contents, until the final SE0 indicates that a new packet follows.

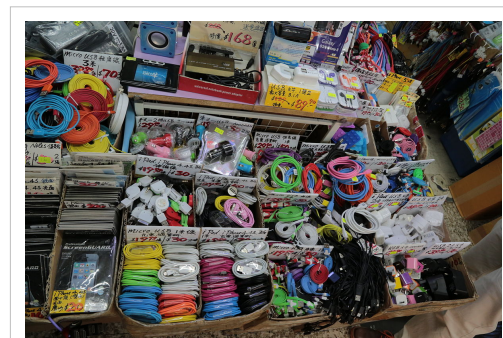
Comparisons with other connection methods

FireWire

At first, USB was considered a complement to IEEE 1394 (FireWire) technology, which was designed as a high-bandwidth serial bus that efficiently interconnects peripherals such as disk drives, audio interfaces, and video equipment. In the initial design, USB operated at a far lower data rate and used less sophisticated hardware. It was suitable for small peripherals such as keyboards and pointing devices.

The most significant technical differences between FireWire and USB include:

- USB networks use a tiered-star topology, while IEEE 1394 networks use a tree topology.
- USB 1.0, 1.1 and 2.0 use a "speak-when-spoken-to" protocol; peripherals cannot communicate with the host unless the host specifically requests communication. USB 3.0 allows for device-initiated communications towards



A variety of USB cables for sale in Hong Kong

the host. A FireWire device can communicate with any other node at any time, subject to network conditions.

- A USB network relies on a single host at the top of the tree to control the network. In a FireWire network, any capable node can control the network.
- USB runs with a 5 V power line, while FireWire in current implementations supplies 12 V and theoretically can supply up to 30 V.
- Standard USB hub ports can provide from the typical 500 mA/2.5 W of current, only 100 mA from non-hub ports. USB 3.0 and USB On-The-Go supply 1.8 A/9.0 W (for dedicated battery charging, 1.5 A/7.5 W Full bandwidth or 900 mA/4.5 W High Bandwidth), while FireWire can in theory supply up to 60 watts of power, although 10 to 20 watts is more typical.

These and other differences reflect the differing design goals of the two buses: USB was designed for simplicity and low cost, while FireWire was designed for high performance, particularly in time-sensitive applications such as audio and video. Although similar in theoretical maximum transfer rate, FireWire 400 is faster than USB 2.0 Hi-Bandwidth in real-use, especially in high-bandwidth use such as external hard-drives. The newer FireWire 800 standard is twice as fast as FireWire 400 and faster than USB 2.0 Hi-Bandwidth both theoretically and practically. The chipset and drivers used to implement USB and FireWire have a crucial impact on how much of the bandwidth prescribed by the specification is achieved in the real world, along with compatibility with peripherals.

Ethernet

The IEEE 802.3af Power over Ethernet (PoE) standard specifies a more elaborate power negotiation scheme than powered USB. It operates at 48 V DC and can supply more power (up to 12.95 W, PoE+ 25.5 W) over a cable up to 100 meters compared to USB 2.0, which provides 2.5 W with a maximum cable length of 5 meters. This has made PoE popular for VoIP telephones, security cameras, wireless access points and other networked devices within buildings. However, USB is cheaper than PoE provided that the distance is short, and power demand is low.

Ethernet standards require electrical isolation between the networked device (computer, phone, etc.) and the network cable up to 1500 V AC or 2250 V DC for 60 seconds. USB has no such requirement as it was designed for peripherals closely associated with a host computer, and in fact it connects the peripheral and host grounds. This gives Ethernet a significant safety advantage over USB with peripherals such as cable and DSL modems connected to external wiring that can assume hazardous voltages under certain fault conditions.

MIDI

Digital musical instruments are another example where USB is competitive in low-cost devices. However Power over Ethernet and the MIDI plug standard have an advantage in high-end devices that may have long cables. USB can cause ground loop problems between equipment, because it connects ground references on both transceivers. By contrast, the MIDI plug standard and Ethernet have built-in isolation to 500V or more.

eSATA/eSATAp

The eSATA connector is a more robust SATA connector, intended for connection to external hard drives and SSDs. eSATA's transfer rate (up to 6 Gbit/s) is similar to that of USB 3.0 (up to 5 Gbit/s on current devices; 10 Gbit/s speeds via USB 3.1, announced on July 31, 2013). A device connected by eSATA appears as an ordinary SATA device, giving both full performance and full compatibility associated with internal drives.

eSATA does not supply power to external devices. This is an increasing disadvantage compared to USB. Even though USB 3.0's 4.5 W is sometimes insufficient to power external hard drives, technology is advancing and external drives gradually need less power, diminishing the eSATA advantage. eSATAp (power over eSATA; aka ESATA/USB) is a connector introduced in 2009 that supplies power to attached devices using a new, backwards-compatible, connector. On a notebook eSATAp usually supplies only 5 V to power a 2.5-inch HDD/SSD; on a desktop workstation it can additionally supply 12 V to power larger devices including 3.5-inch HDD/SSD and

5.25-inch optical drives.

eSATAp support can be added to a desktop machine in the form of a bracket connecting to motherboard SATA, power, and USB resources.

eSATA, like USB, supports hot plugging, although this might be limited by OS drivers and device firmware.

Thunderbolt

Thunderbolt combines PCI Express and Mini DisplayPort into a new serial data interface. Current Thunderbolt implementations have two channels, each with a transfer speed of 10 Gbit/s, resulting in an aggregate unidirectional bandwidth of 20 Gbit/s.

Interoperability

Main article: USB adapter

Various protocol converters that convert USB data signals to and from other communications standards.

Related standards

The USB Implementers Forum is working on a wireless networking standard based on the USB protocol. Wireless USB is intended as a cable-replacement technology, and uses ultra-wideband wireless technology for data rates of up to 480 Mbit/s.

USB 2.0 High Speed Inter Chip (HSIC) is a chip-to-chip variant of USB 2.0 that eliminates the conventional analog transceivers found in normal USB. It was adopted as a standard by the USB Implementers Forum in 2007. The HSIC physical layer uses about 50% less power and 75% less board area compared to traditional USB 2.0. HSIC uses two signals at 1.2 V and has a throughput of 480 Mbit/s using 240 MHz DDRWikipedia:Citation needed signaling. Maximum PCB trace length for HSIC is 10 cm. It does not have low enough latency to support RAM memory sharing between two chips.

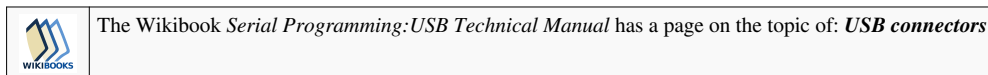
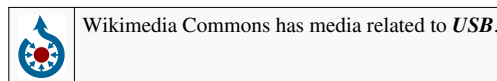
References

- [1] http://www.usb.org/developers/docs/icon_design.pdf
- [2] 08-Sep-2012
- [3] <http://en.wikipedia.org/w/index.php?title=USB&action=edit>
- [4] http://www.usb.org/press/USB-IF_Press_Releases/SuperSpeedUSB_10Gbps_Available_20130731.pdf
- [5] 08-Sep-2012
- [6] Use class information in the interface descriptors. This base class is defined to use in device descriptors to indicate that class information should be determined from the Interface Descriptors in the device.
- [7] *Universal Serial Bus 3.0 Specification*,4.4.11 "Efficiency"
- [8] Micro USB connector spec (http://www.hirose.co.jp/cataloge_hp/e24200011.pdf)
- [9] Micro USB pinout and list of compatible smartphones and other devices (http://pinoutsguide.com/CellularPhones-A-N/smartphone_microusb_connector_pinout.shtml)
- [10] The Chinese FCC's technical standard:
- [11] 100717 usb3.com
- [12] 100717 usb3.com

Further reading

- Axelson, Jan (September 1, 2006). *USB Mass Storage: Designing and Programming Devices and Embedded Hosts* (<http://www.lvr.com/usbms.htm>) (1st ed.). Lakeview Research. ISBN 978-1-931-44804-8. 287 pp.
- ——— (December 1, 2007). *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems* (<http://www.lvr.com/spc.htm>) (2nd ed.). Lakeview Research. ISBN 978-1-931-44806-2. 380 pp.
- ——— (2009). *USB Complete: The Developer's Guide* (<http://www.lvr.com/usbc.htm>) (4th ed.). Lakeview Research. ISBN 978-1-931-44808-6. 506 pp.
- Hyde, John (February 2001). *USB Design by Example: A Practical Guide to Building I/O Devices* (<http://www.intel.com/intelpress/usb/>) (2nd ed.). Intel Press. ISBN 978-0-970-28465-5. 510 pp.
- *Debugging USB 2.0 for Compliance: It's Not Just a Digital World* (<http://cp.literature.agilent.com/litweb/pdf/5988-4794EN.pdf>) (PDF). Technologies Application Note (1382–3). Agilent..

External links



- "USB Implementers Forum" (<http://www.usb.org/>).
- *Universal Host Controller Interface (UHCI)* (<http://stuff.mit.edu/afs/sipb/contrib/doc/specs/protocol/usb/UHCI11D.PDF>) (PDF). Intel.
- *USB 3.0 Standard-A, Standard-B, Powered-B connectors* (http://pinoutsguide.com/Slots/usb_3_0_connector_pinout.shtml). Pinouts guide.
- *Characterization and compliance test* (<http://www.agilent.com/find/USB>). Agilent.
- Muller, Henk. "How To Create And Program USB Devices," (<http://electronicdesign.com/article/embedded/create-program-usb-devices-74233>) *Electronic Design*, July 2012.

Article Sources and Contributors

Serial communication *Source:* <https://en.wikipedia.org/w/index.php?oldid=616067564> *Contributors:* Adoniscik, Al Lemos, Andrew sh, AndyHe829, BenFrantzDale, Bert490, Biot, Biscuitin, Bsroiaadn, Bushing, Churnett, ChrisGualtieri, Cp82, DARTH SIDIOUS 2, Dapevape, DavidCary, Dgtsyb, Dpotter, DreamOfMirrors, Engineerjgc, Epcigenius, Everage, Evic, Flipperritu, Gifflite, Gurch, Harryzilber, Heron, Hitherebrian, IW.HG, IanOsgood, Imroy, J.delanoy, Jordsan, Kbrose, Kevin, Kokkkikumar, Kvng, Longhair, Mackey23, Mandingoeseq, Mange01, Meegs, Michael Hardy, Mild Bill Hiccup, Mmpozulp, Nasa-verve, Nayuki, Nelson50, Ninly, Northamerical000, Offett, Pascal666, Pearle, Pmronchi, Quondum, RJFJR, Raman fastlite, Rick Sidwell, Sam Pointon, Smeirow, Shaddack, Stealth17, Suruena, Tanner Swett, Thaa500, The Anome, Thejoshwolfe, TimothyJKeller, TutterMouse, Username20090319, Vanished user ty12kl89jq10, Wikiwooro0, Wordsworthh, Wtshymanski, Xgoat, Xmaillard, Yefi, Zojoji, Zro, كاشف عميل, 106 anonymous edits

RS-232 *Source:* <https://en.wikipedia.org/w/index.php?oldid=618886055> *Contributors:* 209.75.42.xxx, 64.180.177.xxx, AJim, Abisys, Acather96, Adoniscik, Aldie, Alexander.stohr, Alistair1978, Allen Moore, An-chan, Andrew Hampe, Andy Dingley, Anomalocaric, Antiqueight, Arch dude, Aronzak, Atlant, Ato basehore, Austinnmurphy, AxelBoldt, Bajsejohannes, Baylink, Bert490, Bevo, Bigdumbdinosaur, Biscuitin, Bobblewik, Bollinger, Bon21, Bongwarrior, Bovineone, Breakpoint, Brouhaha, Bryan Derksen, Buddhikaeport, CWenger, Caerwine, CanOfWorms, Captain Segfault, Captainspizzo, Charivari, Chrike, Chris the speller, Christopher Parham, ClementSeveillac, Closeapple, Colonies Chris, CombatWombat42, Conversion rich, Crispmuncher, Crissov, CuriousEric, DJeffo, DanMS, Danhash, Davandron, Dave Yost, Dcxf, Dekisugi, Delirium, Demiurge1000, Deor, Dhanashreevaidya, DieSwartzPunkt, Dlhroher2003, DmitriX, Dmlmax, Dmsar, Druiloor, Ed Brey, Ed g2s, Efalk, Egil, Electron18, Electron9, Engineerism, Evic, Exosot, Extransit, Fahidka, Femto, Ferkelparade, Fingew, Fixedd, Flydpnkrtn, Frap, FrigidNinja, Funandtrvl, Gang65, Gazpacho, Gcbrizan, Gifflite, Gimmetrow, Glenn, Gogo Dodo, Haji akhundov, Hatched3, Hawklord, Hcberkowitz, Heron, Hit.kansagra, Hopp, Hpa, I B Wright, ISC PB, IanOfNorwich, IanOsgood, IlyaHaykinson, Imroy, Inter, Inwind, Isheden, Itai, J Clear, JLD, JaGa, Jahoe, Jason One, Jatkins, Jemaco, Jegor57, Jeh, Jengelh, Jeremy Visser, Jesse Viviano, Jheald, Jonbowen234, Jongs, Jonverve, Jroddi, Justfred, Keilana, Ketiltrout, Koavf, Komalnirala, Krótki, Ksero, Kthbn, Kvng, KyferEz, L Kensington, Lambiam, Liftarn, Lightmouse, Limited Atonement, Linkmurer, Lio, Luna Santin, Lunkwill, MER-C, MKA667, Maarten1980, Madlobster, Makecat, Malcolm Farmer, Mange01, Manifestation, Markthemac, Mastergreg82, Matthiaspaul, Maury Markowitz, Mcculley, Mezzaluna, Mikeblas, Miketwo, Mild Bill Hiccup, Mipadi, Miqounger03, Mmpozulp, Mmxx, Mobius, Mortense, NHRHS2010, Nasa-verve, Neil.steiner, Nekkensj, Ngorham, Nicolaasuni, Nishitapira, Nixdorf, Nmnogueira, Oddbodz, Opardiad, Ortolan88, Overdoer949, Paddu, PaterMcFly, Paul Foxworthy, Petr Kopač, Petri Krohn, Pjb304, Plugwash, Pnm, Pointillist, Poppadepression, Poppafue, Pplshero54, Pthumes, Quibik, RTC, RandomAct, Rchandra, Reddi, Requestion, Reswobslc, Rgraham nz, Rich Frispmuncher, Richpike, Rick Sidwell, Rjwilmsi, Robert the Devil, Robinhw, Ronaldsmith, Rowine719, RoySmith, Rror, S Roper, SEREGA784, SGBailey, Sam Hovecar, Smeirow, SchmuckyTheCat, Scott McNay, Sdimteam, Seaphoto, Shaddack, Shadowjams, Sigmundg, Signal7, SixSix, Snafflekid, Someguy1221, SpaceFlight89, SparhawkWiki, Spitfire, Spoxox, SreekumarC, Sriharsh1234, SummitWulf, Sven Manguard, Sweetey Rose, The Yowler, Theo177, Theresa knott, Thryduulf, Thumperward, Thunderbolt, Tim Starling, Tkbwik, Tokachu, TomViza, Tongtang, Tony esopi patra, Toolnut, Topory, Tothwolf, Towel401, Tsaavik, Uncle Milty, Underdog, Useight, Utcursch, Uzume, Vertago1, Vihljun, Vrenator, Wa2ise, Wa3ftr, Wbm1058, Webclient101, Weevil, WeißNix, West London Dweller, WillFarrell, Wipe, Wireless router, Wmahan, Wonderfl, Wosch21149, Wtmitchell, Wtshymanski, Xenonice, Yonkie, Yurivict, Zoicon5, 634 anonymous edits

Serial port *Source:* <https://en.wikipedia.org/w/index.php?oldid=618146808> *Contributors:* 16@r, l1exec1,3r, APT, Aaa3-other, Abracus, Adam850, Adrzp, Akhlian, Alexander.stohr, Andrew kir, Andrew sh, Andy Dingley, Assjack, Axql, Bellerophon, Benefors, Bert490, Binksternet, Bk0, Blueturtle79, Bobblewik, Brolin Empey, Brouhaha, Cab6403, CanisRufus, Capricorn42, CesarB, Chimin 07, ChrisGualtieri, Clarince63, Clawed, Closedmouth, Coffee, ComCat, Crazycomputers, CyberSkull, Cédric, DMacks, DSLeB, Decemberster, Dkgdkg, Druiloor, Dsimic, DuLithgow, Dulciana, Edward, Egil, EITyrant, Electron9, EpiVictor, Eric B, and Rakim, Everything, Fabulatech, Feraudyh, Gauravswangan, Gavron, Ghetoblaster, Giftlite, Giobe2000, Glider87, Guy Harris, Guzero, Haikupoet, HairY Dude, Haymaker, Hdt83, HenkeB, Henriquevicente, Hetar, Hpa, Hu12, IRP, Ian1469, Imroy, Insidiae, Interior, J.delanoy, Jaanus.kalde, JayC, JayCarlson, Jeh, Jengelh, Jesse Viviano, Jim.henderson, John of Reading, Johnlogic, Kadin2048, Kajaco2, Kewp, Khalid hassani, Kilowatradio, Kokkkikumar, Koman90, KurtRaschke, Kvng, Lee Daniel Crocker, Lightmouse, Luigi.a.cruz, Mac128, Mahjongg, MatthewWilcox, Maxsonbd, Mayor, Mfwithten, Modster, Moreati, Mortense, Mtsjso, MrFish, MrOllie, My Wikipedia, Myanvan, Mywifeandkids, Nikhilgupta2020, Ninly, Orangebodhi, P-Worm, Pedant17, Pekkaphilajasari, Peter Karlsen, Photographerguy, Pklala, Plugwash, Polluks, Puckly, RW Dutton, Radiant chains, RationalBlasphemist, Ray Van De Walker, Reddi, Rei-artur, Rhe br, Rhobite, Ridge Runner, Rmashhadi, Rmhermen, Robinhw, SGBailey, Sandeepgupta26, Saruwine, Scientus, Scott Stirling, Sergiej87, Sergioledesma, Shaddack, Shadowjams, SixSix, Slicing, Snowmanradio, Spike, Spinlock55, Sv1xv, Tarquin, Tempodivalse, Thatio, TheJosh, Thejoshwolfe, Theo177, Thing, Thumperward, Tongtang, Torinir, Tothwolf, Towel401, Trylks, Ttwarding, Uzume, V8Cougar, Vanangamudiyana, Vina, VipX1, Volker, Walkingstick3, Wavelength, Wbm1058, Wderousse, West London Dweller, Whitejay251, Wiki alf, Wiki fanatic, Wikijimmy, Wonderfl, Wtshymanski, Xgoat, Yamla, Yyy, ZZYXx, Zeptooid, Zzuuz, Ibrahimbarbaros, 328 anonymous edits

Universal asynchronous receiver/transmitter *Source:* <https://en.wikipedia.org/w/index.php?oldid=618012933> *Contributors:* Abdull, Agunther, Akadruid, Al Lemos, Alai, Albedo, Alucv, Altermike, Alvin-cs, Amikake3, Anbu121, Andre Engels, Atlant, BD2412, Bhkjersten, Bktero, Blisco, Bloodshedder, Bnossum, Bobblewik, Brenont, CanOfWorms, CanadianLinuxUser, CanisRufus, Carlos Rosa PT, Cdpatel, Chester Markel, Daven200520, Defective, Deineka, Dewritech, Dcentra, Discospinster, Dmaizer, Dmsar, Dthomsen8, Dulciana, ENeville, Electron9, Elomis, Exar Corporation, Excirial, Eyreland, Favonian, Fordsfords, Frap, Frietjes, G1VaE, Gbmthunter, Graham87, Gsl, Guy Harris, Hede2000, Hephaestos, Heron, Hydroxides, JWBE, Javawizard, Jeh, Jesse Viviano, Jfromcanada, Jim1138, Johnlogic, JonHarder, Jonverve, Julien, Kakuoi, Karnesky, Kinema, Kingpin13, Knownot, Koavf, Kuba425, Kvng, Kwamikagami, Kwiki, Leif, Lightmouse, MER-C, MFNickster, Maddog Battie, Magioladitis, Majorkell, Mang0es, Mange01, MarioBlunk, MartinLing, MaterialsScientist, Matt B., MegaHL90, Mk*, Mrt3366, Mumiemonstret, Muon, Nbarth, Oli Filth, Opelio, Ossworks, Paul Richter, PierreAbbat, Polluks, PooH20240, RT Jones, Ray Van De Walker, Reswobslc, Rhinestone K, Robigus, Roofus, Smeirow, Sevsevic, Simonjohndoherty, Srikantshamaga, Ssinghwiki, The Thing That Should Not Be, TheGiantHogwee, Thinking of England, Thomas Willerich, ThreeE, Tiltal, Timharwoodx, Timwi, Toddintr, Ulkl, Uzume, Vskgopu, Wbm1058, West London Dweller, Why Not A Duck, Wikipelli, Wtshymanski, Ysangkok, Zenibus, Zoeyrv, 324 anonymous edits

USB *Source:* <https://en.wikipedia.org/w/index.php?oldid=618943911> *Contributors:* 0612, 132qwerty, 1wolflake, 220 of Borg, 223fms, 4176shelton, 5 albert square, 786b6364, A Train, A0183305, A5b, AJ6J, AMK1211, ANTMAN, AaronLLF, Aaronbrick, Aawaisen, Abce2, Abune, Access Denied, Accounting4Taste, Accurizer, Acela Express, Achraf52, AdRock, Adambro, Adpds4cat, AdeMiami, Adot, AdunaiLayman, Advantecheautomation, Aemaques, Aeons, Agateller, AggroBoy, Agilent, Showard, Agitate, Ahivarn, Ahoerstemeier, Ahuman, Ahunt, Aido2002, Aidor, Airplaneman, Airstplit, Ajfwee, Ajitesh Madai, Ajpvalente, Akasnaskeys, Akkazemi, Alansohn, Alastairand, Aldie, Aleksandrit, Alex murray, Alexander.stohr, Alexi123, Algae, Algocu, Alistair1978, AlistairMcMillan, Allesbehalve, Alpha 4615, Alphathon, Altermike, Aluxosm, Alvin-cs, Alyssa3467, Amdma2003, AmiAyalon1969, Amr.rs, Amux, AnOddName, Andrew sh, Andrew19881123, AndrewLeeson, Andrewatton, Andrewman327, Andrewpkm, Andy Marchbanks, Andyzyweb, Angela, Anibus, Anonymi, Am7a, Antandros, Antialias, Apoc2400, AppOnKey, Arabfaculty, Archer3, Arichnad, Arielco, Aries21erika, Arkrishna, Arlie davis, Arm, Armando, ArnoldReinhold, Aronzak, Arthur Rubin, ArwinJ, Asafot, Asbjornu, Asetwoffity, Ashenai, Ashleypurdy, Asigler, Asim mahakul, Asim18, Asparagus, Atalsandip, Atamido, Atannen, Atenor1932, Atheeb Israr, Atlant, Atomic Taco, Attilios, Audrius u, Austinv311, Avalyn, Avenue X at Cicero, AxelBoldt, Axql, Azhyd, Azuosadiava, B. van der Wee, BDD, BZRAFink, Baffle gab1978, Baoap, Baqeri, Baricom, Bashirama, Bcrsciah198987, BDon003, Bdesham, Bdiijkstra, Beao, Becksguy, Beland, Beleg Täl, Belvdeu, BenFrantzDale, Bender2k14, Benhoyt, Benignabla, Benny45boy, BesigedB, Bestalex, Bgkwtynqghzior, Big gun, Bill Evans at Mariposa, Billywanta, Bitchbastardd, BjKa, Black Walnut, BlackWolf, Blacklife85, Blethering Scot, Blound, Blu3tooth, Blugill, Bo98, Bob Stein - VisiBone, Bobbb53, Bobblewik, Bobo192, Bobrayner, Boijunk, Boing! said Zebede, Bokwai914, Bongwarrior, BorgQueen, Bratch, Brenben92, Brewthaitruse, BrianRecchia, Brianpeiris, Brianski, BrickMcLargeHuge, Brighterorange, Brim, Brsvegas, BrockF85, Brouhaha, Brownsteve, Bruns, Bryan Derksen, Brycen, Bssasindhar, Bstard12, Btornado, Bubba73, Bubeck, Bullzeye, Bungle, Burnte, Burren-p, BurritoBazooka, Buxtehude, Bytecrafter, Bytencoder, Bz2, C0nanPayne, C933103, CALR, CAkIRA, CCFreak2K, CINCABF, CTF831, CWenger, Cab6403, CableCat, Cactus.man, Cadre, Caiafia, Calabraxthis, Callanec, Caltech, Cambrant, CambridgeWayWeather, Can't sleep, clown will eat me, CannonR, Capbat, Capricorn42, CaptainClawz, CaptainVideoJW, Carmichael, Casper2k3, Caspertheghost, Carloslover66667, Caudle, Ceradio, CecilWard, Ceros, CesarB, Cgumas, Charles Gaudette, Chealer, Chiefcoolbreeze, Chimin 07, Chipp C, Chirry2011, Chovain, Chowbok, Chridd, Chris Chittleborough, Chris G, Chris53516, ChrisFAF, Christopher Parham, Chronulator, Chrumps, Chrylis, Chuunen Baka, Ciaranjns, Cj tzm, Clam0p, Classical geographer, Cleared as filed, ClickRick, Closedmouth, Closeplan, Cmcqueen1975, Cmgross, Cntras, Coasterlover1994, Cobaltbluetony, Codemans, Colenso, Colfer2, Colinkgk, Colonies Chris, Comestyles, Comhhppy, CommonsDelinker, Compellingelegance, Compfreak7, Compilation finished successfully, Concentrate2, Coneslayer, Conrad.Irwin, Convolser script, CoolFox, Coolstr24, Cootiequits, Coreywalters06, Coroboy, Corrosive.element, Corvus cornix, Cory.stewart, CountFlux, Courcelles, Crazy Fox, Crispmuncher, Crissov, Cristiandeidaho, Crywalt, Ctachme, Curb Chain, Cwalger, Cwolfsheep, Cybercobra, Cyferz, D zone, D0762, DKqwerty, DMahalko, DSRH, Da Vynci, Daft Crefstman, DaisyChainer, DaleDe, Dalziel 86, Damian Yerrick, Dan100, Danbee, Daniel Pritchard, Dank1993, Dansk14, DarkFalls, Darkdawn75, Darknight512, Darkride, DarthShrine, Dave laird, Dave6, Daveoh, David Biddulph, DavidCary, DavidFarmbrough, DavidMarsh, Davidbspalding, Davidcx, Davidfstr, Davidgordon101, Dcook32p, De728631, Dead3y3, DeadEyeArrow, Deh, Dekisugi, Deletros, Den fjättrade ankan, DenisBlanchette, Depakote, Derek Ross, DerekMorr, Derepi, Desmoh, Devin6687, Dewritech, Diamondland, Diblidabiduu, Didero's dreams, DigitalMediaSage, Dirtyfrank10, Dispenser, Dittavea, Djbyrne17, Djcapelis, Djmckeel1, Dlother, Dltwaddell, DmitriX, Dmlmax, Dnas, Doc Magnus, DocRuby, DocWatson42, Dold2000, Don-vip, DopefishJustin, Dorados, Dori, Dougher, DragonBazooka, Drdmestod, Dredwd90, Dry Erase Markers, Ds13, Dsgreat3, Dsimic, Duckbill, Dvelez1985, Dwight666, Dysprosia, E.boyer7, E090, EagerToddler39, EagleOne, Ebe123, Ed Brey, Ed Poor, Ed g2s, EdLegend, Edgar181, Editore99, Edokter, Ee79, Efitu, Egil, Egmontaz, Ehn, Eighthale, Electron9, ElementFire, EliasOenal, Elpincha, Elsendero, Elvey, Emerson7, Energyequation, Engineerism, Enochlau, Enquire, Enviroboy, Eppri23, EpiVictor, Epsnon291, Equeundil, Eric Wester, Espelp, Espertus, Eurleif, Eus Kevin, Evamy, Evan-Amos, Evert Mow, Everything, Evic, Excirial, Eye of God, Eyreland, Ezhuttukari, F l a n k e r, FT2, Fabulatech, Facts707, Fahidka, Failofbeaner, False vacuum, Falsifian, Favonian, Feedmecereal, FelisLeo, Ffgamera, Fgnievinski, Fibonacci, Fieldday-sunday, Filceolaire, Filelakeshoe, Filzstift, Finnegar, Fir0002, Fireaxe888, Fishnet3722z, Flaz, Flightsoffancy, Floorwalker, Flyer22, FlyingToaster, Focusonpc, Foxj, Fraggel81, Fragelet, Frammy7, Frap, Frederico1234, Fredrik, Frietjes, Fromageequation, Fudoreaper, Fumo78877, Func, Funlunde3, Fustigate314159, Fuzzie, Fvw, G2M, GDonato, GKFX, Gabbe, Gaia Octavia Agrippa, Gail, Gaius Cornelius, Gaius619, Gardar Rurak, Gauravswangan, Gbandst, Geekstuff, Geniac, Georgy90, Gh5046, Ghartwig, Ghermann3, Gidonb, Gifflite, Gilliam, Ginsengbom, Ginsuloff, GioCM, Giraffadate, Givemeomot, Glenn, Gogo Dodo, GoingBatty, Golbez, GokKanga, Good Intentions, Good Olfactory, Goodone121, Gordonblue, Gosale, Gpearson2, Graglin, Grand Am, GrandMoffVixen, Grapetonix, Grauw, GregorB, Gregsea, Grendelkhan, Groink, Groogle, Gryffon5147, Gschizas, Gssa, Guaka, Gutza, Guyjohnton, Gyll, Götz, H2g2bob, H3llbringer, Ha runner, Hankwang, Hans Dunkelberg, Hansschulze, Hargrimm, Harvester, Haseo9999, Hayabusa future, Hcaaman, Hcs, Hdante, Hduckland, Heaveen, Hellgi, Heron, Hgrosner, Hideyuki, Hlandro77, Hobofixer, Hoemaco, Hohohob, Hohum, Homerjaj, Honeycake, Hoof Hearted, Hooperbloom, Hopp, HorsePunchKid, Howicus,

Hughcharlesparker, HumanJHawkins, Hundehalter, Husky, Huttarl, Hydrargyrum, Hyins, I am a true NOS person, I dream of horses, IAMBATMANDEALWITHIT, INVERTED, IRedRat, ISquishy, Ian Pitchford, Ian01, IanGM, Iandiver, Icairns, IceSlider, IcedNut, Icydid, Idabag, IliA Kr., Iloveandersoncooper, Immunize, Imotor, Impasse, Imperator3733, Imroy, Inclusivejunction, Infindebula, Intelkati, Intelligentsium, InternetMeme, Intersofia, Intrgr, IntrigueBlue, InverseHypercube, Iredfj, Iridescent, Irishguy, Isaac, Ita140188, ItsZippy, Iuhjhk87y678, Ixf64, J.delaney, J.smith, JCLAWSON, JDJ, JHP, JLaTondre, JLuna103, JR98664, JTN, JaGa, Jabberwoch, Jacob Poon, Jake Wartenberg, James Kemp, Janto, Jarfil, Jargoness, Jason One, Jasper Deng, Jaufrec, Jay-the-mad'n, JayC, Jaywalk3r, Jcarroll, Jdhood, Jdwinx, Je007, Jean-Baptiste Catté, Jeffq, Jeffro77, Jeffthejiff, Jeltz, JeremyA, Jerriskirkwood, Jerryobject, Jesant13, Jesse Viviano, Jesslovesmeganandpie, Jesster79, Jeremy, Jezmck, Jguard18, Jhsounds, Jidanni, Jim A H, Jim-Bob Harris, Jim.henderson, Jim138, Jimgawn, Jimmi Hugh, Jimp, Jimthing, Jink, Jionpedia, Jjioooooeeeee123, Jk4201, Jmayorga5, Jmcdon10, Jmrowland, Jmundo, Jnavas, Jncraton, Jnsears, JoanneB, Joaopaulo1511, Joe Schmedley, JoeOnSunset, JoeFish13, Joeinwap, Joel D. Reid, Johammer, John Fader, John Reaves, JohnCD, JohnSawyer, Johnhlogic, Johnhteslade, Jon125, JonHarder, JonSangster, Jonathan Grynspan, Jonathunder, Jonel, Jonverve, Jordan Brown, JosephCampisi, Josh Parris, Joshua Scott, Jossi, Jpgordon, Jruderman, Jrvz, JtMinahan, Julesd, JulienR, JustinRossi, Juux, Jwidjaja, Jwinius, Jwoodger, K8 fan, KJRehberg, KSHiger, KUSam, KVeil, Kajaco2, Kalan, Kaldosh, Karanjag, Karn, Katanada, Katieh5584, Katyare, Kazvorpal, Kbh3rd, Kbolino, Kbrose, Keilana, Kelleheretic, KelleyCook, Kemiv, Kendal Ozzel, Kendo70133, Kenny sh, Kenyon, KerryVeenstra, Kghose, Khalid hassani, Khazar2, Kia.Rahimi, Kilo-Lima, King of Arabia, King of Hearts, Kiteinthewind, Kitsunegami, Kittens-Pedro, Kjkolb, Kjohna, Kniesten, KnowledgeOfSelf, Knuckleskin, Kocio, Kooo, Koopa turtle, Korey.conway, Kostmo, Kozuch, Krallja, Krash, Krishvanth, Kristof vt, Ktr101, Kudret abi, Kuru, Kvg, L Kensington, LN2, La Pianista, Lab16, Lada103, Laefer, Lambtron, Landroo, Lankiveil, Lantrix, LarryLACa, Lawpjc, Leandrod, Leandrogfcdutra, Letdorf, Lhopitalified, Lightmouse, Lilash12132, Linas, Ling Kah Jai, Link83, Lipatden, Lithos12, Lithpiperpilot, Little Mountain 5, LittleWink, Littlefda, Livebird, LivingShadow, Lockoom, LodeRunner, Logan, Lonaowna, Lopifalko, LorenzoB, Lorondos, Lotje, LouScheffer, Lovely Chris, Lowellian, Lucasreddinger, Lugevas, Luigi.a.cruz, Luna Santin, Lupo, M1s1ontomars2k4, MCG, MER-C, MISTYFAN4EVER8887, MNAdam, MSR93, MaGioZal, Maaf, Mabdul, Mac, Mad with power, Madison Alex, Manassehkatz, Mange01, Manishearth, Manuactive, Manuelt15, Marc Kupper, Marcfi, Margin1522, Mark Yen, Markhoney, Marksuart44, Marstronix, Martarius, Maschneider, Materials scientist, Mathonius, Matt Crypto, Matta33178, Matthiaspaul, Mauls, Maury Markowitz, MaxHund, Maximus Rex, Maxis ftw, Maxvip, Maxwellversion2, Maxfi, Mayhemm, MayorOTuesday, Mazin07, McGeddon, McSly, Mearling, Memaven2, Mcoorazao, Meaningful Username, MelbourneStar, Mendaliv, Mensch0815, Mentifisto, Merlinsorca, Mewtu, Mfwhiten, Mgdunn, Michael Hardy, Microfilm, MidMadWiki, Midlandstoday, MightyWarrior, Mike.lifeeguard, Mike1024, Mikeblas, Mikhail Ryazanov, Mikus, Mild Bill Hiccup, MilfordBoy991, Minna Sora no Shita, Miqounger03, Mirabilis, Mirddes, Misocroft, MithrandirAgain, Mj fan1995, Mjpieters, Mkdw, Mkouklis, Mlewis000, Mnw2000, Mobius, ModskRule, Modster, Mogism, Mojo-chan, MoolmaAFox, Morcheeba, Moreati, Mortense, Moxfyre, Mpa, Mr Minchin, Mr.Z-man, MrBurns, MrDolomite, MrFish, MrOllie, MrSomeone, Mrappleton, Mrschimpf, Mrtangent, Mschindwein, Mtpaley, Mtp, Mtu, Muad, Mugunth Kumar, Muhandes, Mulad, Mun206, Munge, Mushroom, Music Sorter, Mvjs, Mwarren us, Mwilso24, Mxjose, My man Friday, Mysekurity, Mysidia, N2e, N5ilm, NHRHS2010, NZR, Nageh, Nahaj, Nanshu, Naohiro19, Narge, Nasukaren, NawlinWiki, Nchalada, Neil916, Neilm, NellieBly, Nerfer, Netkinetic, Niceguyedc, Niceter, Nick, NickVeys, Nigelj, Nikkibella21, Nikpapag, Nil Einne, Nishantjr, Nithin.A.P, Nixdorf, Njol, Nk, Nlaporte, Nnetala, Noahspurrier, Nopetro, Norm, Novakreo, Nrbelex, Nsaa, Nubi78, Nullaman, Nurg, Nw15062, O.Kosowski, O18, Oabj34Q, Odam, Odatus, Oehoeboeroe, Ohnoitsjamie, OIEnglish, Old Moonraker, Olekrst, OllieFury, OllyH, Omegared23, Omegatron, Omicronperseid8, Oneiros, Oni Oookami Alfador, OpenToppedBus, Optakeover, Osarius, Osdnok, Osram, Ost316, Ottawa4ever, Outlyer, OverlordQ, Owengibbins, Oxymoron83, PRRfan, Pabouk, Padillah, Palosirkka, Paradoctor, Paranoid, Parmastew, Patriarch, Paul A, PaulColby, PaulMcCulloh, Pavel.nps, Pbacina, Pchov, Pcusser42, Peak, Pedant17, Pedro, PedroDaGr8, Pembers, Perardi, Perfectblue97, Perryizr8, Persian Poet Gal, Peruvianllama, PeterGrecian, Petiatil, Petteria Aimonen, Peyre, Pgan002, Pgiiii, Phantasee, Phantomsteve, Phatom87, Phgao, PhilKnight, Philip Trueman, Phip, Phobie, Phoenix314, Photographerguy, PiMaster3, PiaH, Pie4all88, Pietrow, Pinkadelica, Pinkicious, Pip2andahalf, Pivotto, Pixelpapst, PizzaMan, Pjrm, PlatinumX, PlayStation 69, Plugwash, PluniAlmoni, Pmc, Pmj, Pmsyzz, Pne, Pointillist, Pol098, Polluks, Poppafuze, Prabash.A, Prari, Praveentech, Preslethe, Problemew, ProhibitOnions, Psmith781, Psz, Puchiko, Pugetbill, Pugglewuggle, Pushpinder86, Qasimnb, Qk, Quadell, Quantumsilverfish, Quentin Jones, Quinxorin, Quizzicus, Quota, Qxz, R1SC, R'n'B, R2D2 C3PO R2D2, RA0808, RAMChYLD, RJHall, RadioactiveKiller1, Radiojon, Raggzouken, Rajeshkamboj, Ramu50, RandomDSdevel, Randomperson666111, Raptor007, RasputinAXP, Ravenperch, Ravensfan5252, Ray921, Raymond Hill, Raysonho, Rbellika, Rbrittner, Rchandra, Rcingham, Rdnetto, Reach Out to the Truth, Realist2, Rearden9, Rebroad, Red, RedWolf, Redsully, Reeceyyyy15, RegentsPark, Rehevkor, Reisio, Remember the dot, Remove indian propaganda, Res2216firestar, Retodon8, Revera, Revrant, RexNL, Rexrodo, Rfc1394, Rfi, Rhindle, Rhobite, Rich Farmbrough, Richardcavell, Richardpitt, Ridge Runner, Rikonate, Ringbang, RingtailedFox, Rinix, Rip969, RitKill, Rjwilmsi, Rmhand, Rmsuperstar99, Rob Cranfill, Robert Loring, RobertG, Rocastelo, RockMaestro, Rodneyorpheus, Rohasnagpal, Rollins83, Romanm, Ronark, Ronhones, RoninRVP, Ross5647, Rossheth, Rpkrawczyk, Rrpr, Rufflos, RufusThorne, Rupert Clayton, RussNelson, Russella, Rwestafer, Rwww, Ryan8374, Ryper, S. Neuman, S.K., SLi, SMC, Sadalmelik, Sadharan, SafariSunD, Saimhe, Sajajaja, Salamurai, Salavat, Saltine, Salvio giuliano, Sam Hocevar, Samarqandi, Samuel Grant, Sanbeg, Sango123, Sanspeur, Sasawat, Sasuke Sarutobi, Saulo Paiva, Savant13, Sawyeriii, Smeirow, SchmuckyTheCat, SchreyP, Scollk, Scope creep, Scott, Scuac, Sdsds, Seanmcd27, Seidenstud, Semicolons, SentientSeven, Served333, Sfoehner, Shafkatsharif, Shamilton, ShaunRoselt, Shayno, Sherool, Shirimasen, Shjacks45, Shniken1, Shop Bucuresti, SidP, Sietsse Snel, Sigma 7, Silica-gel, Siliconov, Sillydragon, SimonEast, SimonP, Simulcra, Sintau.tayua, SivaKumar, SixSix, Skintigh, Skylinerspeeder, Sleepy Sentry, Sleske, Sliocki, Smbp, Smjg, SmolderinCorpse, Smyth, Snafflekid, Sncube, Snori, SnowRaptor, Sofia Lucifairy, Soliloquial, Some jerk on the Internet, Someguy1221, Someone not using his real name, Sonicsuns, Sonjaaa, Soren121, South Bay, SpadesSlick, Spambo, Spartan117458, Spe88, Spik3balloon, Spike, Sporg, Srijan89, Srleffler, Stacd, Stan Shebs, StarkRG, Ste.Ri, Ste.virus, SteinbDJ, Stephan Leeds, Stephen Gilbert, Stephen Morley, Stephenchou0722, Stevfranks, Steven Zhang, Steveprutz, Stimson, StoneGiant, Storm Rider, Strait, Strom, Strz4life, Stuart P. Bentley, StuffOfInterest, Sublastic, Sully76, Sumonbd, SuperBeav, Superway25, Suvituuli, Suwa, Sven Godin, Svick, Sweet ann, T Arndt 40, T4bits, THEN WHO WAS PHONE?, Tacvek, Tagishsimon, Tarquin, Taw, Taxman, Tbird20d, Tbolioi, Technopat, Techtonic, Tecknode, TedE, Tedder, TehMulti, Tgeaim, Thaiio, Thals1992, The Pondermatic, The Rambling Man, The Thing That Should Not Be, The Wild Falcon, The guy who kleanz, The.Computer1, TheDoober, TheFearow, TheJosh, Theboss48506, Thebrains29, Thechuck, Thegreatestmover3, Theodore Kloba, Thinkspank101, Thomas Blomberg, Thomas d Stewart, Thorpe, ThreeBlindMice, Threyon, Thue, Thumperward, Thunderboltz, Tiefighter, Tiffaf, Tim Forcer, Timaru, Timbrowne, Timeshift9, Timhoppen, Timm123, Timwi, Tiscando, Titoxd, Tjpeople, Tkgd2007, Tkteun, TobyDZ, Tobz1000, Todd Vierling, Tollsjo, Tom.freeman, Tom812, Tomashcu60, Tomaxer, Tomg1234, Tomlee1968, Tommy2010, Tony1, Tooki, Torresol, Torturetyler, Toyotatundra, Tprbadbury, Trackunf, Transfinite, Traut, Traxs7, Treygdor, TrickyNik, Truerock2, Tsedit, Tsk, Ttwaring, Turkeyphant, TutterMouse, Tweek17, Tweisbach, Twintop, Twocs, Tymoathy, USB lover 101, UdovdM, Ugnich Anton, Uisgebaugh, UncleBubba, Undeference, UnicornTapestry, Unynn, Unyoyega, Urhixidur, Ush3o, Useight, User5910, Utcursch, Utility Monster, Uwe Hermann, V Brian Zurita, VMS Mosaic, Vahid alpha, Vanished user aoiowaiuyr894isdik43, Varnish, VasilievVV, Vaughan Pratt, Veejayyc, Vegaswikian, Versus22, Vertium, Vespriano, Vhann, Victorigrigas, Vid, Viktor, Viljo Viitanen, Vippylaman, VirtualAssist, Vlooo, Voeren, Voetsjoeba, Voidxor, Vrenator, Vuongfat, Wabernat, WackyBoots, Waiwai933, Walter Görlitz, Waltervulej, Wavelength, Wbm1058, Wctaiwan, Webe, Werdemier, Wesrick, Whitis, Whompage, WhosAsking, Whowhodilly, WikHead, Wiki alf, Wiki7777777ikiw, Wikid77, Wikievil666, Wikifan21century, Wikiwinganer, Wikrockciana, Wiknerd, Wimt, Windsok, Wizpig64, Wizzy, Wjwalrus, Wmahan, Wolbo, Wolfgang Kufner, Wonko, Woohookitty, WorldGentoo, Worm That Turned, Wtf305, Wtmitchell, Wtshymanski, Ww, WynnSmith, Wyveryx, Wywin, X1, X1cygnus, Xezbeth, Xmm0, Xolom, Xorx, Xyzzavatar, Y2kboy23, Yanayz, Yeastygoodness, Yellowdesk, Yetasoli, Yintan, Yitzhak, Yosh3000, Yowanvista, Yuckhil, Yusiang1998, Z hosen, Zac67, ZakuSage, Zalgo, Zaran, Zebe, Zenlax, Zenotek, Zephyric, Zephyris, Zidane2k1, Zippanava, Zirconsco, Zlhappyone, Zntrip, Дарко Максимовић, ဝဉ်း ဘဝဉ်း, 言语, 3074 anonymous edits

Image Sources, Licenses and Contributors

File:DB25 Diagram.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:DB25_Diagram.svg *License:* Public Domain *Contributors:* Mobius, Shooke, WikipediaMaster

Image:RS232 PCI-E.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:RS232_PCI-E.jpg *License:* Public Domain *Contributors:* Original version: Towel401. Later version: Kbh3rd.

Image:Rs232 oscilloscope trace.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rs232_oscilloscope_trace.svg *License:* Creative Commons Sharealike 1.0 *Contributors:* Rs232_oscilloscope_trace.jpg; Ktnbn derivative work: Samuel Tardieu (talk)

File:RS232-UART Oscilloscope Screenshot.png *Source:* https://en.wikipedia.org/w/index.php?title=File:RS232-UART_Oscilloscope_Screenshot.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Haji akhundov

Image:Commons-logo.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Commons-logo.svg> *License:* logo *Contributors:* Anomie

Image:Wikibooks-logo-en-noslogan.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Wikibooks-logo-en-noslogan.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Bastique, User:Ramac et al.

Image:Serial port.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Serial_port.jpg *License:* Public Domain *Contributors:* Duncan Lithgow

File:Mac lc printer modem ports.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Mac_lc_printer_modem_ports.jpg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Caroline Ford

Image:FTDI USB SERIAL.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:FTDI_USB_SERIAL.jpg *License:* Public domain *Contributors:* Kilowattradio

File:Certified USB.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Certified_USB.svg *License:* Public Domain *Contributors:* Adrianwo, Magog the Ogre, McZusatz, Tacsipacsi, Tbhotch

File:USB.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:USB.svg> *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Simon Eugster – Simon / ?! 19:02, 7 January 2008 (UTC)

File:USB Icon.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Icon.svg *License:* Public Domain *Contributors:* Mobius at en.wikipedia

File:Usb head Cable.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Usb_head_Cable.jpg *License:* Creative Commons Attribution 3.0 *Contributors:* Vahid alpha

File:USB-Connector-Standard.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:USB-Connector-Standard.jpg> *License:* Creative Commons Zero *Contributors:* Evan-Amos

File:Computer-USB2-card.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Computer-USB2-card.jpg> *License:* Public Domain *Contributors:* Evan-Amos

File:Certified Hi-Speed USB.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Certified_Hi-Speed_USB.svg *License:* Public Domain *Contributors:* Certified Hi-Speed USB

File:SuperSpeed USB.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:SuperSpeed_USB.svg *License:* Public Domain *Contributors:* Gerd Fahrenhorst, HQX320, Magog the Ogre

File:USB pipes and endpoints (en).svg *Source:* [https://en.wikipedia.org/w/index.php?title=File:USB_pipes_and_endpoints_\(en\).svg](https://en.wikipedia.org/w/index.php?title=File:USB_pipes_and_endpoints_(en).svg) *License:* unknown *Contributors:* User:Bdesham

File:USB 2 and 3.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_2_and_3.jpg *License:* GNU Free Documentation License *Contributors:* Bubba73

File:SanDisk Cruzer Micro.png *Source:* https://en.wikipedia.org/w/index.php?title=File:SanDisk_Cruzer_Micro.png *License:* Public Domain *Contributors:* Evan-Amos Beao

File:Circuit board from a USB 3.0 external 2.5-inch HDD enclosure.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Circuit_board_from_a_USB_3.0_external_2.5-inch_HDD_enclosure.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Dsimic

File:Male and Female USB Connectors.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Male_and_Female_USB_Connectors.jpg *License:* GNU Free Documentation License *Contributors:* Original uploader was Zephyris at en.wikipedia Later version(s) were uploaded by Osama bin dipesh at en.wikipedia.

File:Usb extension cable.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Usb_extension_cable.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Original uploader was J.smith at en.wikipedia

File:Usb connectors.JPG *Source:* https://en.wikipedia.org/w/index.php?title=File:Usb_connectors.JPG *License:* Public Domain *Contributors:* Viljo Viitanen

File:A Micro-A USB port.jpeg *Source:* https://en.wikipedia.org/w/index.php?title=File:A_Micro-A_USB_port.jpeg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:heaven

File:MicroB USB Plug.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:MicroB_USB_Plug.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:masamic

File:Mini usb AB.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Mini_usb_AB.jpg *License:* Creative Commons Attribution 2.5 *Contributors:* User Mgdunn on en.wikipedia

File:USB Std A.png *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Std_A.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Std B.png *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Std_B.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Mini B.png *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Mini_B.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Micro A.png *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Micro_A.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Micro B.png *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Micro_B.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Mini-B receptacle.png *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Mini-B_receptacle.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Micro-AB receptacle.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Micro-AB_receptacle.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Music Sorter

File:USB Micro B receptacle.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Micro_B_receptacle.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:BokWai914, User:Music Sorter

File:Connector USB 3 IMGP6033 wp.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Connector_USB_3_IMGP6033_wp.jpg *License:* GNU Free Documentation License *Contributors:* smial (talk)

File:Types-usb th1.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Types-usb_th1.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Frozthbyte, Jdthood, Torsch, 2 anonymous edits

File:USB 3.0 Micro B plug.PNG *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_3.0_Micro_B_plug.PNG *License:* Creative Commons Attribution 3.0 *Contributors:* Tosaka

Image:Htc extmicrousb port and mhl-hdmi plug.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Htc_extmicrousb_port_and_mhl-hdmi_plug.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Davidbspalding

Image:Popport.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Popport.jpg> *License:* Public Domain *Contributors:* EiZei

File:USB Twisted Pair.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_Twisted_Pair.svg *License:* Public Domain *Contributors:* User:WolfWings and User:Inductiveload

File:Y-shaped USB 3.0 cable.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Y-shaped_USB_3.0_cable.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Dsimic

File:USB voltage and current meter.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_voltage_and_current_meter.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Dsimic

File:Thinkpad X220 Yellow USB.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Thinkpad_X220_Yellow_USB.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Xyzzavatar

File:Micro USB phone charger.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Micro_USB_phone_charger.jpg *License:* Public Domain *Contributors:* Reinraum

File:USB fans 1.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_fans_1.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:BOY

File:USBVacuumCleaner.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:USBVacuumCleaner.jpg> *License:* Public Domain *Contributors:* Raysonho@Grid Engine

File:USB signal example.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:USB_signal_example.svg *License:* Creative Commons Zero *Contributors:* User:Petteri Aimonen

File:Cables in Hong Kong.JPG *Source:* https://en.wikipedia.org/w/index.php?title=File:Cables_in_Hong_Kong.JPG *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Victorgrigas

License

Creative Commons Attribution-Share Alike 3.0
[//creativecommons.org/licenses/by-sa/3.0/](https://creativecommons.org/licenses/by-sa/3.0/)
