

[sparkfun.com](https://www.sparkfun.com)

# RS-232 vs. TTL Serial Communication

*Skill Level: Beginner*



## [RS-232 vs. TTL Serial Communication](#)

by [Jimb0](#) | November 23, 2010 | [10 comments](#)

---

One of the tools we use most when debugging our projects is [serial](#) input/output. Serial is very easy to implement, and it allows you to send/receive any data you need from your microcontroller to a computer's serial port so it can be viewed using a [terminal emulator](#). These two devices are compatible from a software perspective, however you can't just hook a microcontroller up to a computer because the hardware interfaces are not compatible.

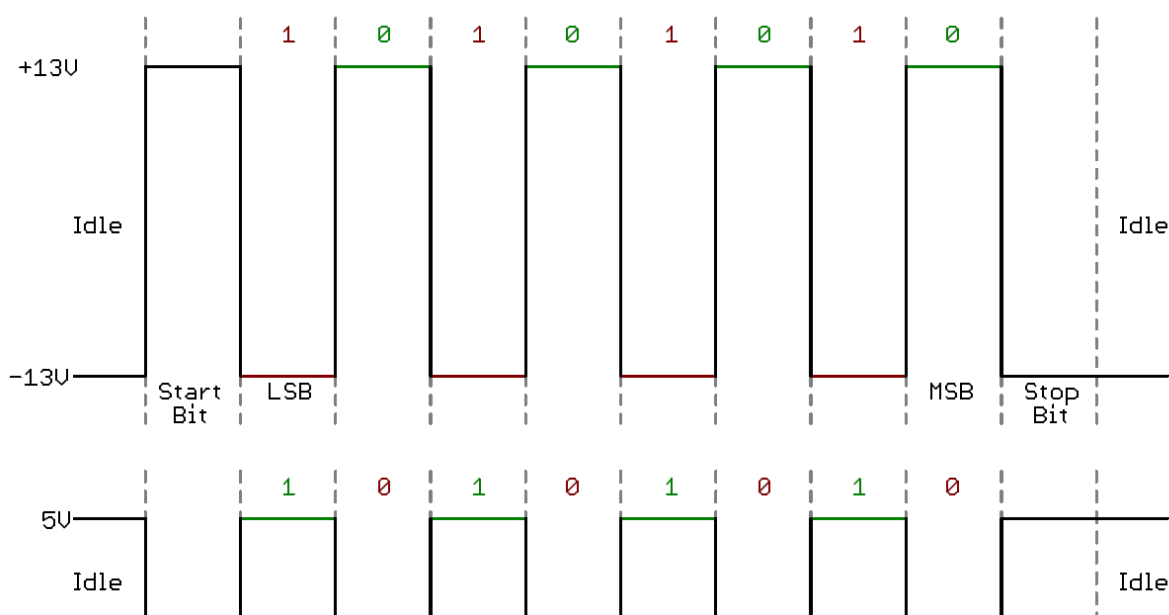
Most microcontrollers these days have built in UARTs (universally asynchronous receiver/transmitter) that can be used to receive and transmit data serially. UARTs transmit one bit at a time at a specified data rate (i.e. 9600bps, 115200bps, etc.). This method of serial communication is sometimes referred to as **TTL serial** (transistor-transistor logic). Serial communication at a [TTL](#) level will always remain between the limits of **0V and Vcc**, which is often 5V

or 3.3V. A logic high ('1') is represented by  $V_{cc}$ , while a logic low ('0') is 0V.

The serial port on your computer (if it's lucky enough to have one, they're quickly becoming a relic) complies with the **RS-232** ([Recommended Standard 232](#)) telecommunications standard.

RS-232 signals are similar to your microcontroller's serial signals in that they transmit one bit at a time, at a specific [baud](#) rate, with or without [parity](#) and/or stop bits. The two differ solely at a hardware level. By the RS-232 standard a logic high ('1') is represented by a negative voltage – anywhere from -3 to -25V – while a logic low ('0') transmits a positive voltage that can be anywhere from +3 to +25V. On most PCs these signals swing from **-13V to +13V**.

The more extreme voltages of an RS-232 signal help to make it less susceptible to noise, interference, and degradation. This means that an RS-232 signal can generally travel longer physical distances than their TTL counterparts, while still providing a reliable data transmission.

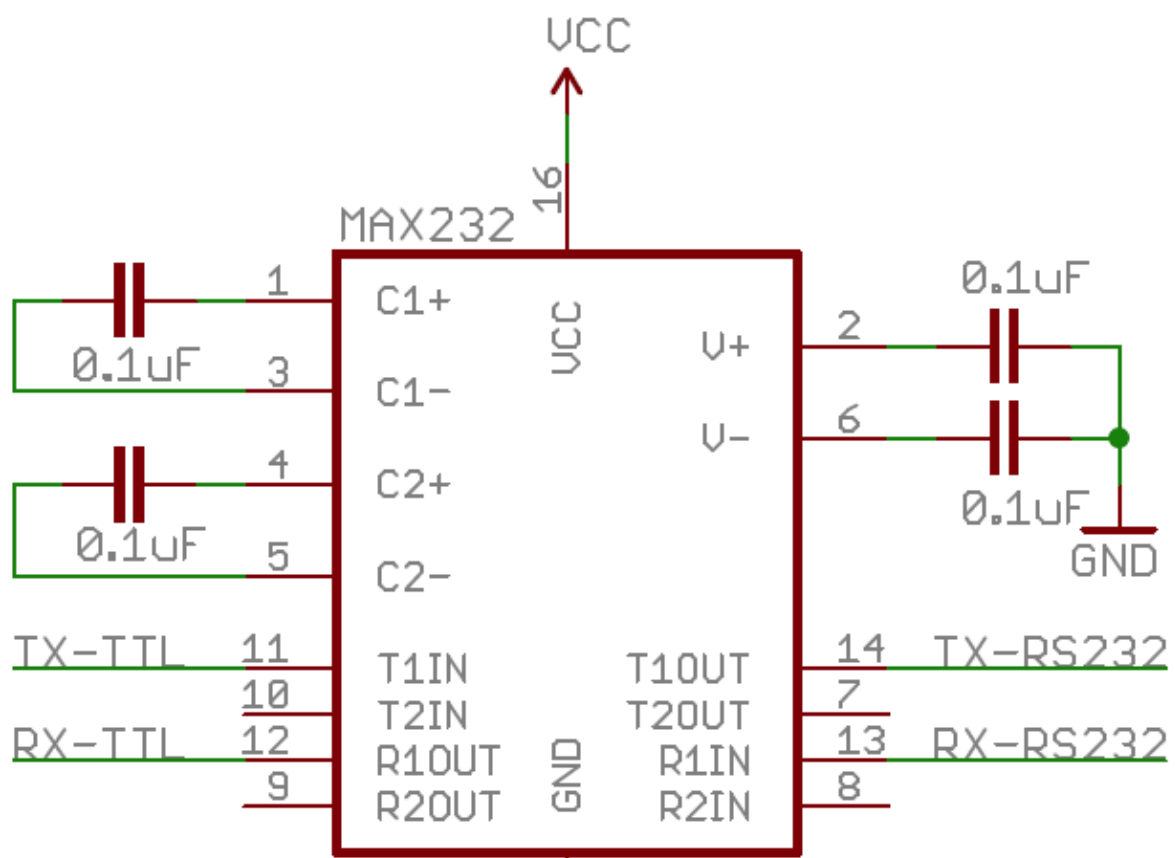




This timing diagram shows both a TTL (bottom) and RS-232 signal sending 0b01010101

## Solutions

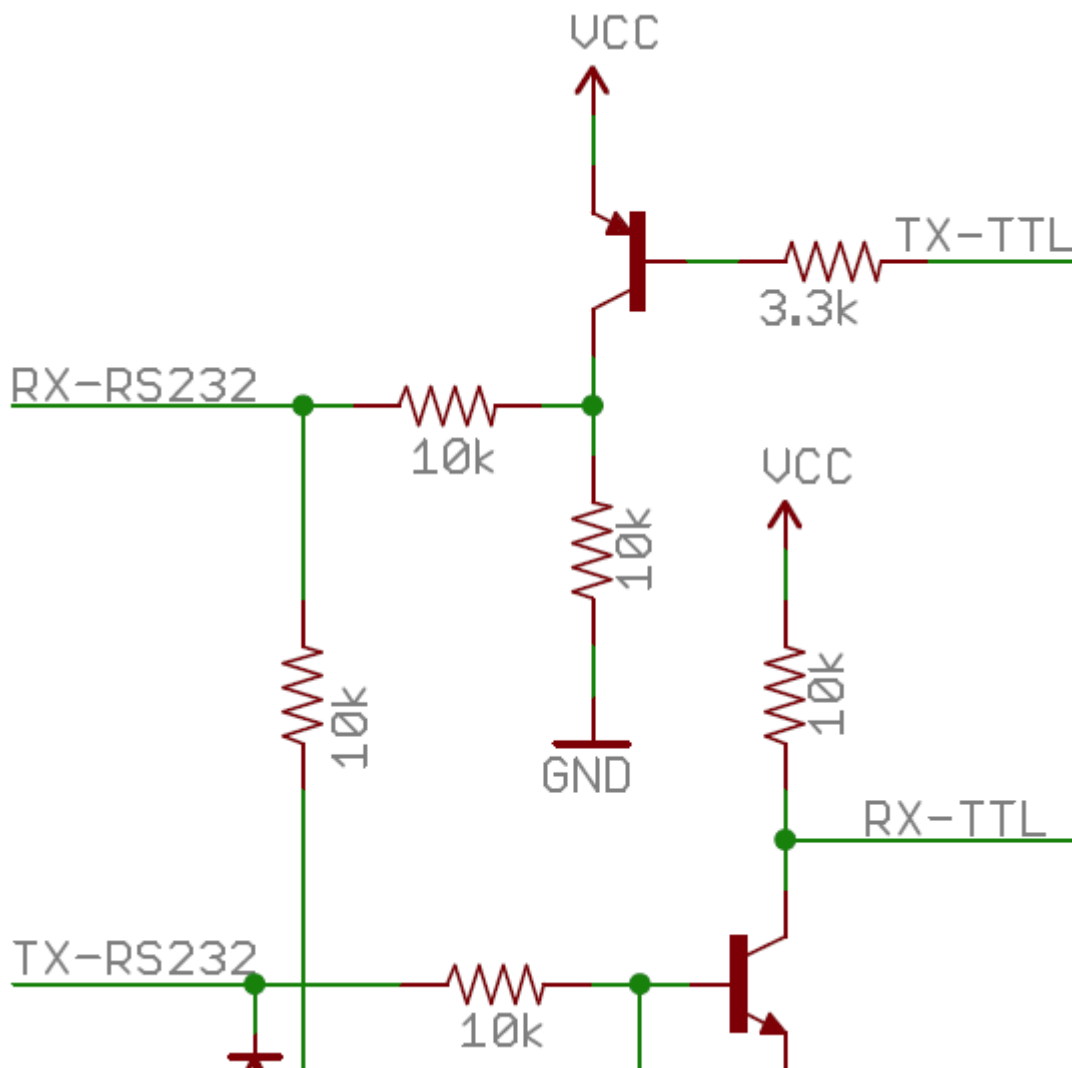
So, you may see where the problem lies in interfacing these two signals. To connect these two ports you not only have to **invert** the signals, but you also have to deal with regulating the potentially harmful RS-232 voltages to something that won't destroy a microcontroller's serial pins. There are a handful of solutions to this problem of voltage converting and inverting. The most common, and easiest solution is just plugging a [MAX-232](#) in between the two devices:

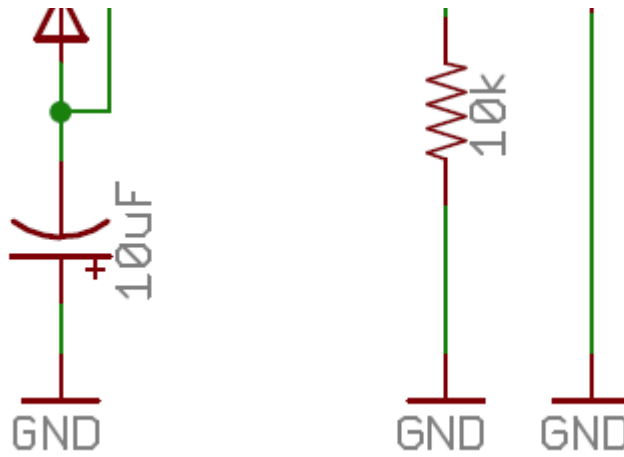




There are many generic derivatives of the MAX-232. [Maxim IC](#) just happened to be the first to market with this neat device (decades ago!) so out of habit, we call all ICs that do similar jobs 'MAX-232s'.

Less expensive solutions, like our [RS-232 Shifter](#), use transistors or inverters to flip the signals, and [charge pumps](#) to get the voltages high enough to be RS-232 compliant:





## Sample Question

With the above information at hand, here's a quick quiz to test your shiny new knowledge:

1. What are the two main differences between RS-232 and TTL signals?
2. True or false: Data is sent from a PC's RS-232 port at 9600 bits-per-second (bps), it's converted to TTL by a MAX232 before reaching a microcontroller. The voltages of the signals look different on each end, but the speed (bps) does not change.

## Answers:

Spoiler Alert! Highlight from here...

1. The '1's and '0's are inverted from each other. The minimum and maximum voltages of RS-232 signals is +/-13V, and only 0 to 3.3V/5V for TTL signals.
2. True. The data rate will always remain the same, even if the

voltages of the RS-232 and TTL signals are different.

...to here to reveal the answers. Because, you know, they're so super-secret.