

Hacker Dictionary: RS-485 Will Go The Distance

[Originalartikel](#)

[Backup](#)

RS485 is a communication standard that should be part of the advanced hardware hacker's arsenal; it's not commonly encountered, but powerful exactly when you need it. It's a physical layer interface for wired communications that uses a single differential pair for noise immunity, has good long-distance properties, and allows many connections to a single bus. Because of that, you will encounter it in security systems and even cameras, wired sensor networks, DMX512 lighting and all sorts of industrial electronics. For our hobbyist goals, you can absolutely use RS485 to build your home (or room) automation system, or a relatively large robot without all those worries that wireless brings.

The name might remind you of RS232, and that's because both RS232 and RS485 are standards that come from EIA (Electronics Industries Alliance). It also might remind you of RS422, if you've ever seen this name mentioned online; RS422 and RS485 are closely intertwined, sharing most of the physical layer, and I'll show how exactly they relate.

Physical Layer

At its simplest, a RS485 link is akin to a UART communications channel, but differential and half-duplex, with TX and RX combined into a single communications line. Bytes sent as unclocked data at a pre-arranged baud rate, so your USB-UART adapter or MCU's UART peripheral will work for most RS485 purposes. Just like with UART, there are only a few commonly used baud rates, so it should be easy to guess. The differential signal means that you will need to use special transmitters and receivers for working with RS485 links; they aren't pricey, so it's wise to have a few in stock before the day when you could really use it.

One of the main benefits of RS485 is that you can put multiple devices on a single link. This is a pleasant difference from logic-level UART or RS232, especially for our sensor and other smart device network building needs. The tradeoff is that an RS485 link is half-duplex; that is, two or more devices cannot transmit at the same moment of time.

If you ever really need a full-duplex connection, where devices X and Y might want to transmit without regard for each other, you will want to use two RS485 links; one link for a X transmits, Y receives; direction and other for the Y transmits, X receives; direction, point-to-point style. Congratulations, you've invented RS422, but you've lost the very useful bus nature of RS485.



data-permalink=„https://hackaday.com/2022/04/05/hacker-dictionary-rs-485-will-go-the-distance/haddia_rs485_101_1d/“ data-orig-file=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png“ data-orig-size=„1881,583“ data-comments-opened=„1“ data-image-meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"0"}“ data-image-title=„haddia_rs485_101_1d“ data-image-description=„“ data-image-caption=„“

data-medium-file=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?w=400“

data-large-file=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?w=800“

class=„aligncenter wp-image-529107 size-large“

src=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?w=800“

alt=„Timing diagram for TTL UART data and corresponding RS485 signals“ width=„800“

height=„248“ srcset=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png 1881w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?resize=250,77 250w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?resize=400,124 400w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?resize=800,248 800w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_1d.png?resize=1536,476 1536w“ referrerpolicy=„no-referrer“ /></p><p>Signals are transmitted

differentially. In place of one signal, there are two signals: one is always opposite the other – when you transmit a high logic level, line A goes high and line B goes low, and when you transmit a low logic level, line A goes low and line B goes high. The receiver doesn’t measure the absolute level of the signals relative to ground – instead, it measures the difference between them. We have <a

href=„<https://hackaday.com/2016/03/29/when-difference-matters/>“>talked about differential pairs before, explaining their benefits like high noise reduction. Using differential pairs is what allows for RS485 links hundreds of meters long that you will see people online brag about building!</p><p><img data-attachment-id=„529115“

data-permalink=„https://hackaday.com/2022/04/05/hacker-dictionary-rs-485-will-go-the-distance/hadimg_rs485_101_thumb/“

data-orig-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg“

data-orig-size=„1200,1200“ data-comments-opened=„1“ data-image-meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"1"}“ data-image-title=„hadimg_rs485_101_thumb“ data-image-description=„“

data-image-caption=„“

data-medium-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg?w=400“

data-large-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg?w=625“

class=„alignright size-medium wp-image-529115“

src=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg?w=400“

alt=„“ width=„400“ height=„400“


srcset=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg 1200w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg?resize=250,250 250w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg?resize=400,400 400w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_thumb.jpg?resize=625,625 625w“ referrerpolicy=„no-referrer“ /></p><p>One fun part about RS485 is the A/B naming conventions. In the last paragraph, I’ve used “A” and “B” as “non-inverting” and “inverting” signals respectively – that is, voltage on “A” has the same polarity as the logic level

of the signal transmitted, but voltage on B is the opposite of it.

The RS485 standard, as it's written, actually has the opposite meaning for these two: A is inverting and B is non-inverting. However, the RS485 transceiver makers have all (consistently) reversed them relative to the standard, to the point where Wikipedia has [a good few paragraphs](https://en.wikipedia.org/wiki/RS-485#Signals) complaining about it. When it comes to the naming convention being used in this article, I will use the signal names you will actually deal with when it comes to RS485 ICs and devices, as opposed to the ones defined in the standard but not actually respected by devices you will encounter. It might be better if the world standardized on $D+$ and $D-$ as with USB, but that ship has sailed.

Differential Implications

The difference between two signals, rather than their voltage relative to a common ground, is what the controller actually cares about. What happens if they're close to each other; say, there's no transmitter driving the line at the moment? If the difference between A and B is within 200 mV, the signal is considered invalid, and your receivers might get confused, or, at best, reject it. To avoid this, biasing resistors are typically used: a pullup on A and a pulldown on B. The typical biasing resistor values are calculated to create a voltage a bit over 200 mV (to account for noise) on receiver inputs after accounting with the cable impedance and termination resistors; essentially, calculating a three-resistor voltage divider.



data-permalink=„https://hackaday.com/2022/04/05/hacker-dictionary-rs-485-will-go-the-distance/haddia_rs485_101_2b/“

data-orig-file=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png“ data-orig-size=„1284,713“ data-comments-opened=„1“ data-image-meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"0"}"“ data-image-title=„haddia_rs485_101_2b“ data-image-description=„“ data-image-caption=„“

data-medium-file=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png?w=400“

data-large-file=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png?w=800“

class=„wp-image-529108 size-medium alignleft“

src=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png?w=400“

alt=„KiCad diagram of a typical biasing and termination network in a two-device RS485 link“

width=„400“ height=„222“

srcset=„https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png 1284w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png?resize=250,139 250w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png?resize=400,222 400w, https://hackaday.com/wp-content/uploads/2022/03/haddia_rs485_101_2b.png?resize=800,444 800w“

referrerpolicy=„no-referrer“ />

Common practice is to put two resistors in one spot, and have them do all the biasing work. You can absolutely ballpark something like 500 ohms worth of pull, and [this appnote](https://www.ti.com/lit/pdf/slla272) has a straightforward formula on page 4 if you'd like to know more; a simple resistor divider calculation is all it takes. There's more to the bus, however; let's dive into the layout and termination requirements.

In differential links like this, especially as speeds get higher, [impedance](https://hackaday.com/2015/07/29/say-it-with-me-input-impedance/) starts to matter; which, in large part, is a requirement for the cabling you use. For a start, it has to be a twisted pair. The recommended impedance for RS485 links

is 120 ohms and that's what RS485 hardware is typically designed for, but for many hobbyist applications, you can absolutely use one of the pairs in an Ethernet cable (100 ohms) or a decent USB cable (90 ohms) and call it a day.

For impedance matching and reflection dampening reasons, RS485 also requires that you add termination resistors at the ends of the network; they're typically 120 ohms. Typical RS485 modules you can get come with 120R termination resistors, and you likely won't need to adjust them to match your cable's impedance; but you will need to remove the termination resistors that are not on the ends of the daisy chain! Interested in more explanations of why termination matters, with pretty graphs and explanations of all the different types of termination possible? <a

href=„<https://www.ti.com/lit/pdf/snla034>“ target=„_blank“>This appnote will help you, too.

The RS485 bus doesn't lend itself to stubs, and especially not to star topology networks; pulling the same link into multiple different directions is heavily discouraged due to reflections caused by signal coming back from ends of multiple different branches at once, and difficulties properly terminating such a topology. The best network will result from daisy-chaining the devices, as in, having them in a straight line, with RS485 devices connected inline, stubs as short as reasonably possible; under six inches is the usual guideline, but less is better.

Ok, Ok, How Do I Start?

<img data-attachment-id=„529113“

data-

permalink=„https://hackaday.com/2022/04/05/hacker-dictionary-rs-485-will-go-the-distance/hadimg_rs485_101_4a/“

data-orig-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg“ data-

orig-size=„4032,3024“ data-comments-opened=„1“ data-image-

meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"1"}“ data-image-title=„hadimg_rs485_101_4a“ data-image-description=„“ data-image-caption=„“

data-

medium-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?w=400“

data-

large-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?w=800“

class=„alignright size-medium wp-image-529113“

src=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?w=400“ alt=„Two "MAX485" modules side by side. The top one is brand new, while the last one has some resistors removed, a piece of twisted pair inside the terminal block, oh, and an IC with a crater“ width=„400“ height=„300“

srcset=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg 4032w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?resize=250, 188 250w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?resize=400, 300 400w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?resize=800, 600 800w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?resize=1536, 1152 1536w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_4a.jpg?resize=2048, 1536 2048w“ referrerpolicy=„no-referrer“ /></p><p>So how do you actually get into RS485, as a hobbyist just learning about the powers it gives you? If you look up “RS485” at your online retailer of choice, you will encounter these cheap elongated blue modules with pin headers and a terminal block – and feel free to just get those. Most of them have clones of the MAX485 IC – one of the many IC designs for talking RS485 and certainly a staple of RS485 hardware, others being generally pin-compatible for the same feature set, like SN75176 and SP485.</p><p>These PCBs come with 120 Ω termination resistors and 20 kΩ biasing

resistors; you will likely want to change these. Remove the termination resistors for any modules not at the ends of the link, and replace biasing resistors with those that get you to a stable bus idle voltage point; a pair of 560 Ω resistors on one of the modules will do well for biasing.

The MAX485 works with a single RS485 link, so it is half-duplex: you have to switch it between receiving and transmitting. On the MAX485, transmitter and receiver enable inputs are two different pins, but basically everybody joins them into one pin. This is possible because the RE (Receiver Enable) signal is inverted, and this lets you switch between transmit and receive with a single GPIO. Essentially, you short pins 2 and 3 of the MAX485 together, either with a jumper wire or with a tactically placed drop of solder, then set them to a low logic level for receiving or high logic level for transmitting with a GPIO. If you're using a USB-UART or UART peripheral, check if you have the RTS signal available to you; it can be directly wired to a RS485 transceiver's \sim RE and DE for seamless RS485 link operation!

You can buy these MAX485 modules and mostly be set, but the ICs on them are clones and might burn out on you every now and then. Getting some legit MAX485 ICs from a reputable store and replacing them with your hot air station of choice is a good idea if you want your setup to be reliable. One of the problems with MAX485 is that it requires 5 V and might not work at 3.3 V, so you might as well get some MAX3485 chips and solve both of these problems in one go. Otherwise, if you got a Raspberry Pi in one hand and a cheap MAX485 module in the other, feel free to use any of the level shifting options, from using one of the usual logic level shifters to simply [pulling the VCC of MAX485 down with a diode](https://hackaday.com/2017/01/20/cheating-at-5v-ws2812-control-to-use-a-3-3v-data-line/).

With all this customization and rework, you're probably thinking you might as well design your own RS485 interface. If you've already committed to a PCB design for the project, you're probably right.

Adding More Devices

The RS485 physical layer lets you put many devices on the same link, like I2C. You cannot do that with three devices that have UART. The requirement, again, is that you only ever have one transmitter active at a time; so, by default, you'd have all transmitters on all devices disabled, and only enabled at a time when one device needs to say something.

The maximum number of devices that you can put on a single bus is described through the concept of unit load, a function of the input impedance of receivers and the biasing. In other words, receivers put a certain amount of load on the transmission line, and when the line is loaded too much, the signal levels get too low. The unit load values of your ICs can be found in their respective datasheets, with typical values resulting in up to 32 receivers on a single bus, and available transceiver ICs with low enough unit load that lets you get that number up to 256.

For your own RS485 communications on a multi-drop network, you'd want some kind of transmission protocol that has coexistence and addressing in mind. There's one popular and powerful option you'll want to know about, and that is [Modbus RTU](https://en.wikipedia.org/wiki/Modbus#Modbus_RTU_frame_format_(primarily_used_on_asynchronous_serial_data_lines_like_RS-485/EIA-485))! It is a data protocol you can use on top of RS485, suited for links with a single controller and multiple peripherals. It's a commercially successful yet open protocol; which is to say, you can speak Modbus in between your own devices, but also to a large range of devices like heavy-duty machinery or [energy meters](https://hackaday.com/2016/05/25/meter-all-the-phases-three-phase-energy-meter-with-openwrt/). There's plenty of decent libraries for Modbus communications for environments like [Python](https://pymodbus.readthedocs.io/en/3.0.0/readme.html) and [Arduino](https://github.com/smarmengol/Modbus-Master-Slave-for-Arduino) to do most of the job for you.

You might notice that the single controller, multiple peripherals principle is similar to I2C, and that's not the only way where Modbus and I2C are alike. For a start, Modbus peripherals, too, have addresses. The way that data is accessed in Modbus is also similar to the register-based system that

the more advanced I2C devices have: each Modbus device has registers you can read from or write into. Writable registers tend to be referred to as coils, read-only registers tend to be referred to as inputs, and there's generally agreed-upon register address ranges telling you at a glance which are which. When building <a

href=„<https://hackaday.com/2016/12/31/mod-your-camera-with-modbus/>“>makeshift Modbus

devices, however, you are free to break these conventions if it suits you.</p><h1>Keeping

Things Grounded</h1><p><img data-attachment-id=„529120“

data-

permalink=„https://hackaday.com/2022/04/05/hacker-dictionary-rs-485-will-go-the-distance/hadqib_rs485_101_6/“

data-orig-file=„https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png“ data-

orig-size=„600,480“ data-comments-opened=„1“ data-image-

meta=„{"aperture":"0","credit":"","camera":

"","caption":"","created_timestamp":"0","

copyright":"","focal_length":"0","iso":"0

","shutter_speed":"0","title":"","orientation&

quot;:"0"}“ data-image-title=„hadqib_rs485_101_6“ data-image-description=„“ data-

image-caption=„“

data-

medium-file=„https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png?w=400“

data-large-file=„https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png?w=600“

class=„alignright size-medium wp-image-529120“

src=„https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png?w=400“

alt=„Quote saying "[...] lack of ground can result in high common-mode offset voltage on one of the ends. "“ width=„400“ height=„320“

srcset=„https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png 600w,

https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png?resize=250,200 250w,

https://hackaday.com/wp-content/uploads/2022/03/hadqib_rs485_101_6.png?resize=400,320 400w“

referrerpolicy=„no-referrer“ />“Do you need a ground connection for a RS485 link?” is a hotly debated question, just like “do I need to tie my USB port shield to ground”.

You will see a lot of people say <a

href=„<https://forum.arduino.cc/t/rs485-is-shared-ground-required/151514>“ target=„_blank“>that a

ground reference isn't technically needed – after all, it's a differential

signal, and at most, shielding could be called for. This will generally work, even! However, lack of

ground <a

href=„<https://store.chipkin.com/articles/rs485-rs485-cables-why-you-need-3-wires-for-2-two-wire-rs485>

5“ target=„_blank“>can result in high common-mode (relative to ground) offset voltage on one

of the ends, and while RS485 transceiver ICs can handle quite a bit of that, it can cause issues

depending on the kind of devices in your network.</p><p>If you're running a link between

devices powered by un-earthed wall-wart PSUs, pulling a common voltage reference along is likely

wise. The same goes for a network of battery-powered devices, since voltage induced on one end of

the network will happily go onto transceiver inputs of other devices that, in turn, might be capacitively

coupled to somewhere else. Not to mention that noise finds a way.</p><p>If

you are using earthed PSUs on both sides, however, running a separate ground line <a

href=„<https://hackaday.com/2017/03/09/wtf-are-ground-loops/>“>might create a ground loop. If

the “ground” potential at two different locations is wildly different, you may also end

up pulling considerable, undesirable current through the common path. Remember that the two

transceivers need a common middle voltage reference, but it need not be

“ground”.</p><p>If grounding had straightforward answers, we hackers

wouldn't have so many conflicting opinions about it. In case your ground issues get really

unmanageable, you can get <a

href=„<https://www.analog.com/en/technical-articles/isolated-rs485-transceiver-breaks-ground-loops.html>“ target=„_blank“>isolated RS485 transceivers, or, on a budget, use a few 6N137 high-speed optocouplers, maybe even one of those isolated DC-DCs. Typically, there’s <a

href=„<https://www.ti.com/lit/pdf/snla049>“ target=„_blank“>solutions available for any RS485 problems you encounter, and people have made RS485 links work in pretty grim

circumstances.</p><h1>When It Goes Wrong – Or Right</h1><p>If you experience a noticeable amount of data loss or corruption, it’s worth poking your transmission line with a scope or a multimeter. Is the bus idle state within voltage margins, at different points in the network? Do you have ringing on the signal, and how “bad” is it relative to the norm and the signal levels you’d expect? Does ringing change depending on how far the device is in the chain? Echo your Modbus (or other protocol) packets into a debug console – do they arrive intact?</p><p><img data-attachment-id=„529122“

data-

permalink=„https://hackaday.com/2022/04/05/hacker-dictionary-rs-485-will-go-the-distance/hadimg_rs485_101_7/“

data-orig-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg“ data-orig-size=„950,800“ data-comments-opened=„1“ data-image-

meta=„{"aperture":"0","credit":"","camera":"","caption":"","created_timestamp":"0","copyright":"","focal_length":"0","iso":"0","shutter_speed":"0","title":"","orientation":"1"}“ data-image-title=„hadimg_rs485_101_7“ data-image-description=„“ data-image-caption=„“

data-

medium-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg?w=400“

data-large-file=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg?w=742“ class=„alignright size-medium wp-image-529122“

src=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg?w=400“

alt=„Picture with two boards that use RS485 transmitters for sending addressable LED signals over long wires“ width=„400“ height=„337“

srcset=„https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg 950w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg?resize=250,211 250w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg?resize=400,337 400w, https://hackaday.com/wp-content/uploads/2022/03/hadimg_rs485_101_7.jpg?resize=742,625 742w“

referrerpolicy=„no-referrer“ />With this knowledge of RS485, you won’t just be able to gain control of more and more powerful devices out there, you will also figure out some fun things that will help you in other areas. For instance, you don’t have to put UART-like communications through a RS485 transceiver, you can simply use it for transmitting the state of a GPIO in a noisy environment. Or you can use an RS485 transmitter to <a

href=„<https://www.teknynja.com/2014/02/driving-ws2812-neopixels-rgb-leds-over.html>“ target=„_blank“>dramatically extend range and stability of WS2812 LED protocol

communications, especially when you have high-voltage lines running close to your LED strips.</p><p>There’s an idea floating around that you could use the transmitter section of an RS485 driver IC to drive small vibromotors as an improvised cheap H-bridge – and get yourself some fancy haptics-capable hardware as a result. Small-footprint H-bridges, especially for low-power electronics, can get expensive, chip shortages don’t help either – yet RS485 transceivers are cheap and still abundant. Who wants to bet that this will

work?</p><h1>Terminating These Thoughts</h1><p>RS485 is an underappreciated topic, and while wireless links are enticing, there’s plenty of space for good old rugged differential communications. Whether you’d like a protocol you can understand, an interference-resilient

communications channel, or an inherently secure physical layer for your network, RS485 is here for you. With one more tool in your arsenal, may you design wired long-distance links like never before.</p> </html>

From:
<https://schnipsl.qgelm.de/> - **Qgelm**

Permanent link:
https://schnipsl.qgelm.de/doku.php?id=wallabag:wb2hacker-dictionary_-rs-485-will-go-the-distance

Last update: **2025/06/27 11:17**

