



SHOP

LEARN

BLOG

CUSTOM KITS

Serial Terminal Basics

CONTRIBUTORS:  JOEL_E_B,  JIMBLOM, MAETTU_THIS

 FAVORITE 49

Serial Terminal Overview

COM ports. Baud rate. Flow control. Tx. Rx. These are all words that get thrown around a lot when working with electronics, especially microcontrollers. For someone who isn't familiar with these terms and the context in which they are used, they can be confusing at times. This tutorial is here to help you understand what these terms mean and how they form the larger picture that is serial communication over a terminal.

In short, serial terminal programs make working with microcontrollers that much simpler. They allow you to see data sent to and from your microcontroller, and that data can be used for a number of reasons including troubleshooting/debugging, communication testing, calibrating sensors, configuring modules, and data monitoring. Once you have learned the ins and outs of a terminal application, it can be a very powerful tool in your electronics and programming arsenal.

Covered in this Tutorial

There are lots of different terminal programs out there, and they all have their pros and cons. In this tutorial we will discuss what a terminal is, which terminal programs are best suited for certain situations and operating systems, and how to configure and use each program.

Suggested Reading

You should be familiar with these topics before diving into this tutorial. If you need a refresher, feel free to pop on over to these links. We'll be right here waiting.

- Serial Communication
- Analog vs Digital
- Binary
- Hexadecimal
- ASCII
- Installing FTDI Drivers
- RS-232 vs TTL Serial Communication
- Logic Levels
- Connector Basics particularly the USB section

What is a Terminal?

Terminal emulators go by many names, and, due to the varied use of the word terminal, there can often be some confusion about what someone means when they say terminal. Let's clear that up.

Brief History

To understand the use of the word terminal, we must visit the not so distant past. Back when computers were big, bulky, and took up entire rooms, there were only a handful of ways to interface with them. Punch cards and paper tape reels were one such interface, but there was also what was known as a terminal that was used for entering and retrieving data. These terminals came in many form factors, but they soon began to resemble what would become their personal computer descendants. Many consisted of a keyboard and a screen. Terminals that could display text only were referred to as text terminals, and later came graphical terminals. When discussing terminal emulators, it's these terminal of days past that are being referenced.



An OG terminal

Modern Terminals

Today, terminal programs are "emulating" the experience that was working on one of these terminals. They are known as emulators, applications, programs, terms, TTYs, and so on. For the purposes of this tutorial, just the word terminal will be used. Many terminals use to emulate specific types of computer terminals, but today, most terminals are more generic in their interface.

When working on a modern operating system, the word **terminal window** will often be used to describe working within one of these applications. And, often, when reading other tutorials and hookup guides, you will be requested to open a terminal window. Just know that means to open whichever one of these terminals programs strikes your fancy.

It is also worth noting that many terminal programs are capable of much more than just serial communication. Many have network communication capabilities such as telnet and SSH. However, this tutorial will not cover these features.

Terminal vs Command Line

A terminal is not a command prompt, though the two are somewhat similar. In Mac OS, the command prompt is even called Terminal. Hence the confusion when using that word. Regardless, you can perform some of the same tasks in a command prompt that you could also perform within a terminal window, but it doesn't work the other way around; you cannot issue command line statements within a terminal window. We will go over how to create a serial terminal connection within a command line interface later in this tutorial. For now, just know how to distinguish between the two.

Basic Terminology

Here are some terms you should be familiar with when working within a serial terminal window. Many of these terms are covered in a lot more detail in our Serial Communication tutorial. It highly recommended that you read that page as well to get the full picture.

ASCII - Short for the American Standard Code for Information Interchange's character encoding scheme, ASCII encodes special characters from our keyboards and converts them to 7-bit binary integers that can be recognized by a number of programs and devices. ASCII charts are very helpful when working with serial terminals.

Baud Rate - In short, baud rate is how fast your data is being transmitted and received. 9600 is the standard rate, but other speeds are typical amongst certain devices. Just remember that all the links in your chain of communication have to be "speaking" at the same speed, otherwise data will be misinterpreted on one end or the other.

Transmit (TX) - Also known as Data Out or TXO. The TX line on any device is there to transmit data. This should be hooked up to the RX line of the device with which you would like to communicate.

Receive (RX) - Also known as Data In or RXI. The RX line on any device is there to receive data. This should be hooked up to the TX line of the device with which you would like to communicate.

COM Port (Serial Port) - Each device you connect to your computer will be assigned a specific port number. This helps to identify each device connected. Once a device has a port assigned to it, that port will be used every time that device is plugged into the computer.

Your device will show up on your computer as either **COM#** (if you're on a Windows machine) or **/dev/tty.usbserial-#####** (if you're on a Mac/Linux computer), where the #'s are unique numbers or alphabetic characters.

TTY - TTY stands for teletypewriter or teletype. Much like terminal is synonymous with the terminals of old, so too is teletype. These were the electromechanical typewriters used to enter information to the terminal and, thus, to the mainframe. When working with terminals on Mac and Linux, you will often see tty used to represent a communication port rather than 'COM port'.

Data, Stop, and Parity Bits - Each packet of data sent to and from the terminal has a specific format. These formats can vary, and the settings of your terminal can be adjusted accordingly to work with different packet configurations. One of the most common configurations you'll see is 8-N-1, which translates to 8 data bits, no parity bit, and one stop bit.

Flow Control - Flow control is controlling the rate at which data is sent between devices to ensure that the sender is not sending data faster than the receiver can receive the data. In most applications used throughout these tutorials, you will not need to use flow control. The flow control may also be present in the shorthand notation: 8-N-1-None, which stands for no flow control.

Carriage Return & Line Feed - Carriage return and line feed are the ASCII characters sent when you press the enter key on your keyboard. These terms have roots from the days of typewriters. Carriage return meant the carriage holding the paper would return to the starting point of that particular line. Line feed (aka new line) meant the carriage should move to the next line to prevent typing over the previous line.

When typing on a modern keyboard, these terms still apply. Every time you press enter (or return) you are telling your cursor to move down to the next line and move to the beginning of that new line.

Consulting our handy-dandy ASCII table, we can see that the character for line feed is 10 (0x0A in hex) and carriage return is 13 (0x0D in hex). The importance of these two characters cannot be stressed enough. When working in a terminal window you'll often need to be aware of which of these two characters, if not both, are

being used to emulate the enter key. Some devices only need one character or the other to know that a command has been sent. More importantly, when working with microcontrollers, be aware of how you are sending data. If a string of 5 characters needs to be sent to the micro, you may need a string that can actually hold 7 characters on account of the 10 and 13 sent after every command.

Local Echo - Local echo is a setting that can be changed in either the serial terminal or the device to which you are talking, and sometimes both. This setting simply tells the terminal to print everything you type. The benefit from this is being able to see if you are in fact typing the correct commands should you encounter errors. Be aware, though, that sometimes local echo can come back to bite you. Some devices will interpret local echo as double type. For example, if you type `hello` with local echo on, the receiving device might see `hheelllloo`, which is likely not the correct command. Most devices can handle commands with or without local echo. Just be aware that this can be an issue.

Serial Port Profile (SPP) - The Serial Port Profile is a Bluetooth profile that allows for serial communication between a Bluetooth device and a host/slave device. With this profile enabled, you can connect to a Bluetooth module through a serial terminal. This can be used for configuration purposes or for communication purposes. While not exactly pertinent to this tutorial, it's still good to know about this profile if you want to use Bluetooth in a project.

Connecting to Your Device

Now that you know what a terminal is and the lingo that comes with the territory, it's time to hook up a device and communicate with it. This page will show you how to connect a device, how to discover which port it has been assigned, and how to communicate over that port.

What You'll Need

For this example you'll need

- An FTDI Basic - 5V or 3.3V will work fine. You can also use an FTDI Cable if that's all you have.
- A USB Mini-B Cable - (Not necessary if you have an FTDI Cable.)
- A jumper wire - Most FTDI products have female headers, so a male-to-male jumper cable should suffice. Or, you could just use a piece of wire that is stripped on both ends.

Discovering Your Device

Once you have all your supplies ready, attach the FTDI Basic to the USB cable, and attach the cable to your computer. If this is the first time you've plugged in a device of this nature into your computer, you may need to install the drivers. If this is the case, visit our FTDI Driver Installation Guide. If the drivers are all up to date, carry on.

Depending on which operating system you're using, there are a few different ways to discover which port your device has been assigned.

Device Manger (Windows)

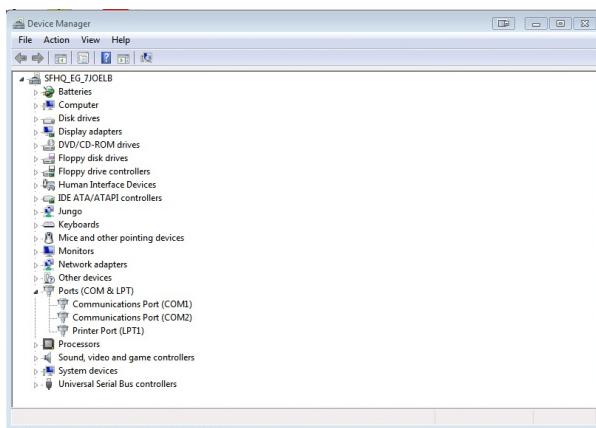
No matter which version of Windows you have, you have a program called **Device Manager**. To open device manager, open the start menu, and type

```
devmgmt.msc
```

into the search bar. Press enter, and it'll open right up. Or, you can right-click on MyComputer, select

Properties, and open the Device Manager from there (Windows 7). If you intend on using your computer to communicate with several serial devices, it may be worth creating a desktop shortcut to Device Manager.

Once you've got Device Manager open, expand the Ports tab. Here is where the information we need lives.



In this image, we have just a few COM Ports showing up. The first thing to know is that COM1 is **ALWAYS** reserved for the true Serial Port, not USB. You know those grey, bulky cables, which have a DB9 connection on each end. Yeah, that serial port. Many computers (especially laptops) no longer have serial ports, and they are becoming obsolete in exchange for more USB ports. Nevertheless, the OS still reserves COM1 for that port for people who still have an true serial port on their computer.

Another port that is likely to show up on most computers is LPT1. This is reserved for the parallel port. Parallel ports and cables are becoming even more obsolete than serial cables, but, again, many computers still have these ports (they're often used to connect to printers) and have to accommodate for that in the OS.

With those out of the way, we can focus on the ports that we do need to use. Now with your FTDI plugged in, you should see a new COM port get added to the list.



Typically, your computer will enumerate your devices. For instance, if this is the first serial communication device you've plugged into your computer, it should enumerate as COM2. On my computer this is the not the first device I've plugged in, but rather the eighth, so it has enumerated as COM9 (don't forget about COM1).

What's important to know is that once a device has been associated with your computer and has had a port assigned to it, the computer will remember that device every time it's attached. So, if you have an Arduino board that has been assigned COM4 for example, it is not necessary to open Device Manager and check which COM port it is on every time, because that device will now always be on COM4. This can be good and bad. Most people will never plug more than a couple dozen serial devices into their computers. However, some people *will* plug in lots of devices, and your computer can only assign so many ports (256 if I remember correctly). Thus, it may be necessary to delete some COM ports. We will discuss that in the tips and tricks section.

If you do have multiple devices and are not sure which device is the one you just plugged in, unplug it, watch for whichever COM port disappears, and then plug it back in. The COM port should reappear letting you know that's the device you're looking for.

One last thing to mention is that all serial devices, even if they require different drivers, will show up as COM ports in Windows. For example, an Arduino Uno and the FTDI Basic both have different drivers and are technically two different types of devices. Windows doesn't discriminate. It will treat both devices the same, and all you have to worry about is with which COM port it's associated. Mac OS and Linux treat this slightly differently. Read on to find out.

Command Line (Mac, Linux)

Similar to Windows, Mac OS and Linux assign a specific port to every device attached to the computer. However, unlike Windows, there is no specific program you can open up to view all the devices currently attached. Have no fear. There is still a simple solution to find your device.

The default command line interface for Mac OS X is Terminal. To open it, go to your Utilities folder. There you should see the icon for Terminal. I'm going to assume that if you're using Linux, you already know how to open a command line window.



Once open, you should see the typical terminal screen.



To see a list of all the available Serial ports on both Mac and Linux, type this command:

```
ls /dev/tty.*
```

You should now see a list of all serial ports on your computer.



```
joelbartlett:~ joelbartlett$ ls /dev/tty.*  
/dev/tty.Bluetooth-Modem      /dev/tty.RN52-056E-SPPDev  
/dev/tty.Bluetooth-PDA-Sync   /dev/tty.iMedia-C105-RNI-SPP  
/dev/tty.Dandelion-SPPDev     /dev/tty.usbmodemfd1331  
/dev/tty.RN52-0530-SPPDev    /dev/tty.usbserial-A6007wOD  
joelbartlett:~ joelbartlett$
```

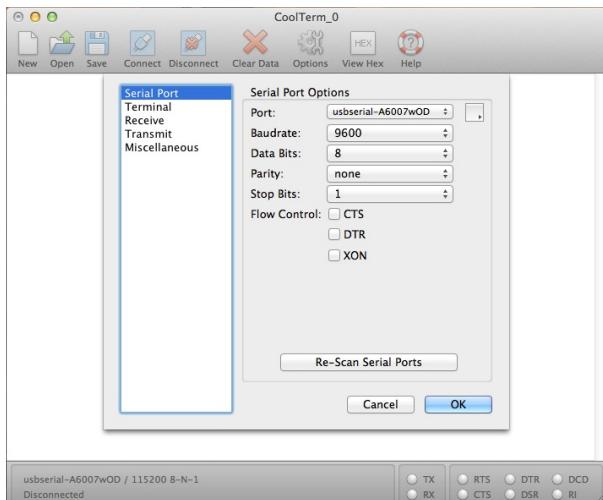
You'll notice a few Bluetooth ports on there. I have several Bluetooth devices paired with my computer, so you may have more or less devices that show up depending on what devices have been paired with your computer. (Notice the SPP portion of these names. That indicates that Bluetooth device can talk to the serial terminal as well.)

The important devices to note are the `tty.usbserial` and the `tty.usbmodem`. For this example I have both an FTDI Basic and an Arduino Uno plugged into my computer. This is just to show you the key difference between the two. As mentioned earlier, some devices are treated differently depending on how they communicate with the computer. The FT232 IC on the FTDI basic is a true serial device, and, thus, it shows up as `usbserial`. The Uno on the other hand, is an HID device and shows up as a `usbmodem` device. The HID (Human Interface Device) profile is used for keyboards, mice, joysticks, etc., and, as an HID device, the computer treats it slightly different despite the fact that it can still send serial data. In either case, these `tty.usb_____` ports are what we're after when connecting to a serial terminal.

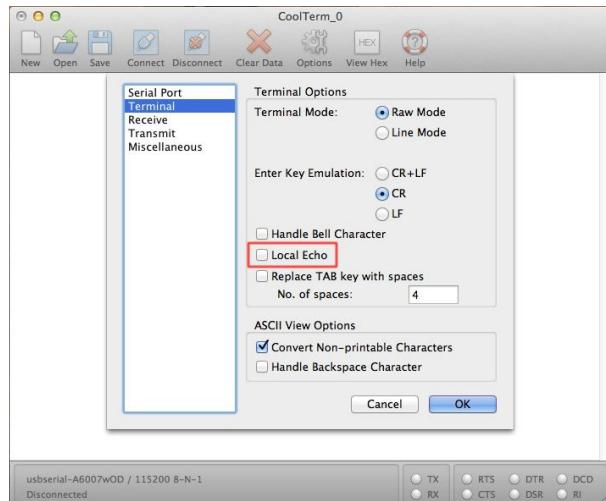
Echo Test

With that out of the way, it's time to actually communicate with the FTDI. The specifics of each terminal program will be discussed in the following sections. This example will be shown in CoolTerm, but be aware that this can be done with any terminal.

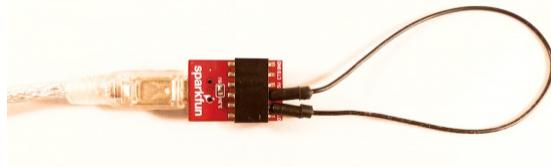
Open up a terminal with the correct settings: 9600, 8-N-1-None.



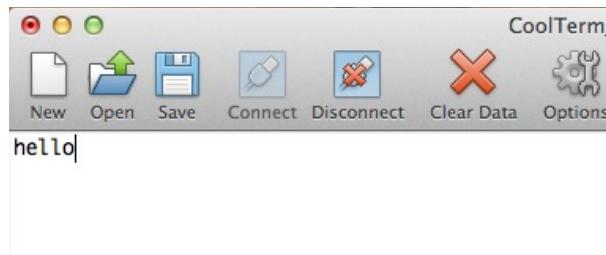
Make sure local echo is turned off for this test.



Take your jumper wire and connect it to the TX and RX lines of the FTDI Basic.



Now type!



Everything you type should be displayed in the terminal window. It's nothing fancy, but you are now communicating with the terminal. Data is being sent from your keyboard, to the computer, through the USB cable to the FTDI, out the FTDI's TX pin, into the RX pin, back through the USB cable, into the computer, and is finally displayed in the terminal window. Don't believe me? Unplug the jumper and type some more. Pending you did turn local echo off, you should not see anything being typed. This is the echo test.

Extra Credit

If you have two FTDI boards or other similar serial devices, try hooking up both of them. Connect the TX line of one to the RX line of the other and vice versa. Then, open two serial terminal windows (yes, you can have multiple terminal windows open at once), each connected to a different device. Make sure they are both set to the same baud rate and settings. Then connect, and start typing. What you type in one terminal should show up in the opposite terminal and vice versa. You've just created a very simplistic chat client!

Now let's explore the different terminal programs.

Arduino Serial Monitor (Windows, Mac, Linux)

The Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. And,

because using a terminal is such a big part of working with Arduinos and other microcontrollers, they decided to included a serial terminal with the software. Within the Arduino environment, this is called the Serial Monitor.

Making a Connection

Serial monitor comes with any and all version of the Arduino IDE. To open it, simply click the Serial Monitor icon.

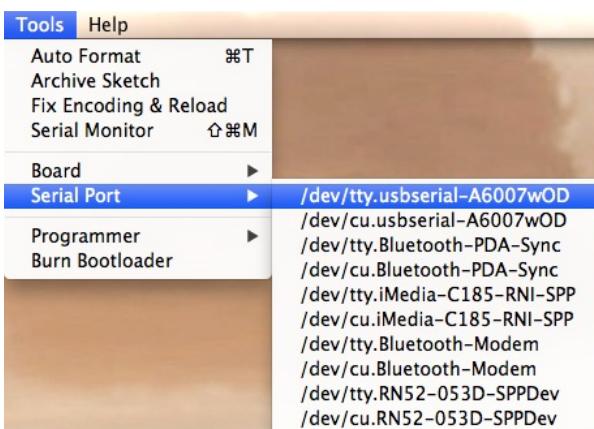


The icon is located to the right of the other icons in Arduino 0023 and below.

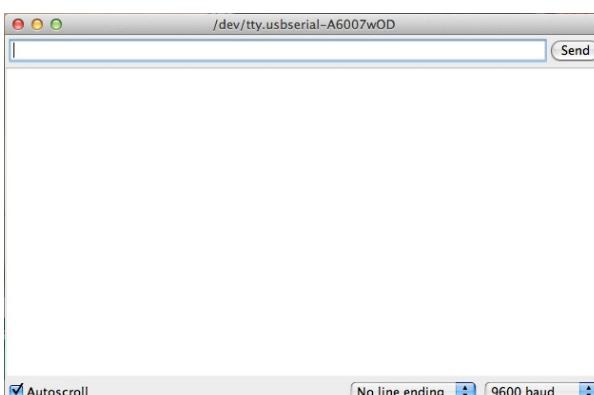


The icon is located to the far right in Arduino 1.0 and beyond.

Selecting which port to open in the Serial Monitor is the same as selecting a port for uploading Arduino code. Go to *Tools* -> *Serial Port*, and select the correct port.

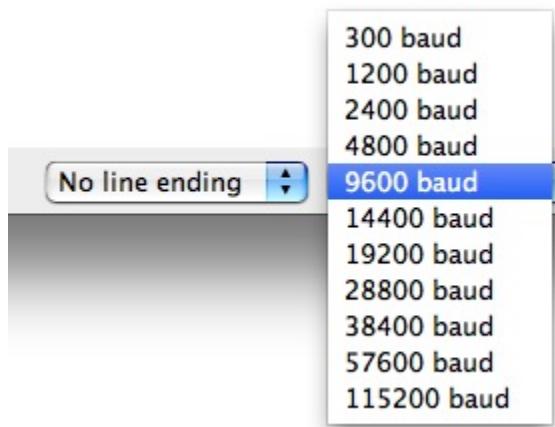


Once open, you should see something like this:

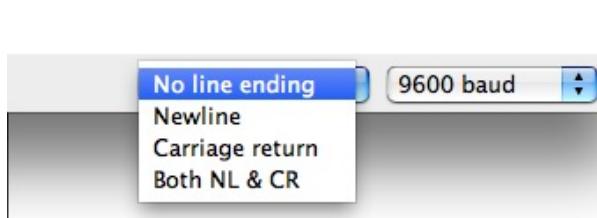


Settings

The Serial Monitor has limited settings, but enough to handle most of your serial communication needs. The first setting you can alter is the baud rate. Click on the baud rate drop-down menu to select the the correct baud rate.



You can also change the enter key emulation to carriage return, line feed, both, or neither.



Last, you can the the terminal to autoscroll or not by checking the box in the bottom left corner.

Pros

- The Serial Monitor is a great quick and easy way to establish a serial connection with your Arduino. If you're already working in the Arduino IDE, there's really no need to open up a separate terminal to display data.

Cons

- The lack of settings leaves much to be desired in the Serial Monitor, and, for advanced serial communications, it may not do the trick.

Hyperterminal (Windows)

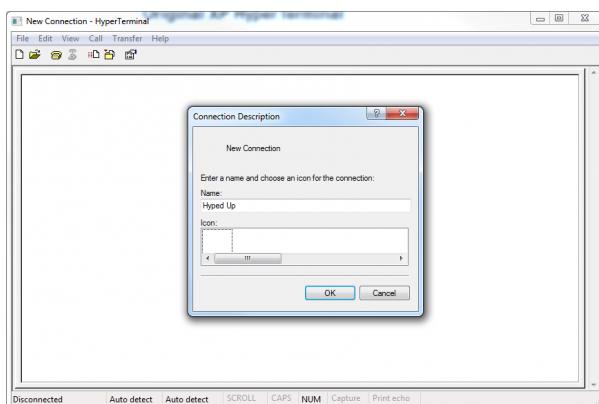
HyperTerminal is the defacto terminal program for any Windows OS up to XP -- Windows Vista, 7, and 8 don't include it. If you're on Windows Vista, 7, or 8, and really just have to have HyperTerminal, a little scouring of the Internet should turn up some workarounds. Better alternatives are more easily available however- we'll get to those shortly.

If you're on a pre-Vista machine, and only have HyperTerminal to work with, here are some tips and tricks for using it:

Initiating a Connection

When initially opening up HyperTerminal, it will present you with a "Connection Description" dialog. Enter any

name you please, and, if you really want to get fancy, select your favorite icon. Then hit "OK". (If this window didn't pop up go to **File > New Connection** to open it.)

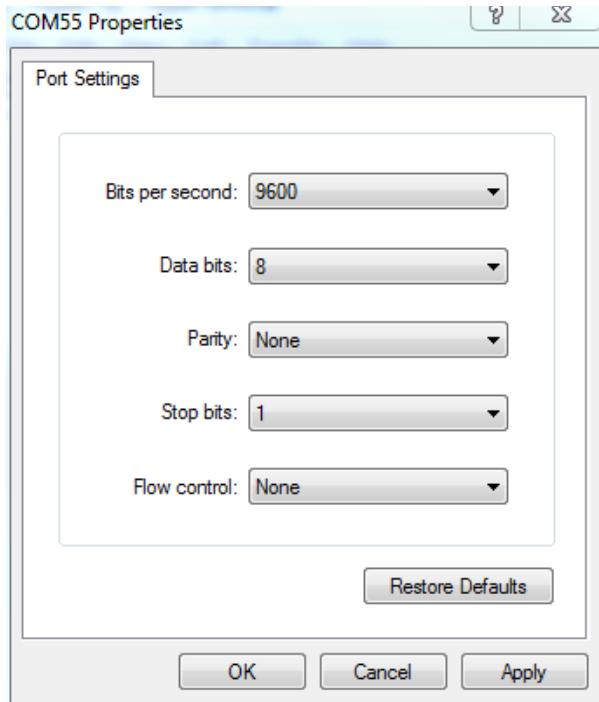


None of the settings in this first window have any effect on the serial communication.

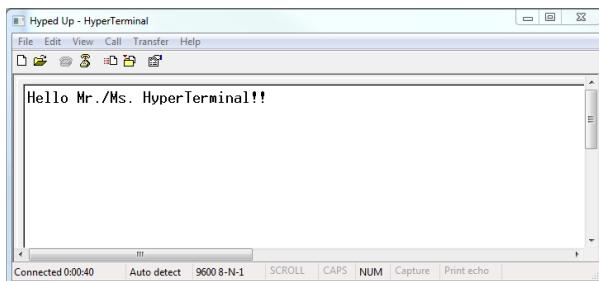
On the next window, ignore the first three text boxes -- we're not working with a dial-up modem here. **Do select your COM port** next to the "Connect using" box. Then hit "OK".



The settings on the next box should look pretty familiar. Make sure the "Bits per second" dropdown is set to the correct baud rate. And verify that all of the other settings are correct. Hit "OK" once everything looks correct there.



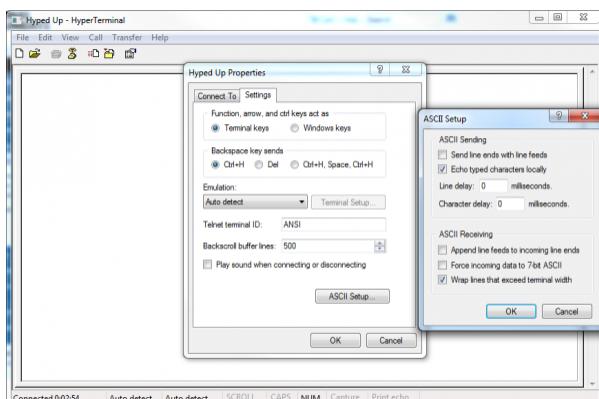
It doesn't look like much, but you now have an open terminal! Type in the blank white area to send data, and anything that is received by the terminal will show up there as well.



Adjusting Settings

There are some limited adjustments we can make to the HyperTerminal UI. To find them, go to **File > Properties**. Under the "**Settings**" tab you'll see most of the options.

If you want to see what you're typing in the terminal, you can turn on **local echo**. To flip this switch, hit the "**ASCII Setup**" button, then **check "Echo typed characters locally"**.



The other settings are very specific to formatting how characters are sent or received. For most cases they should be left be.

Those who have used HyperTerminal have either come to accept it for what it is, or sought out some other -- any other(!) -- terminal program. It's not great for serial communication, but it does work. Let's explore some of the better alternatives!

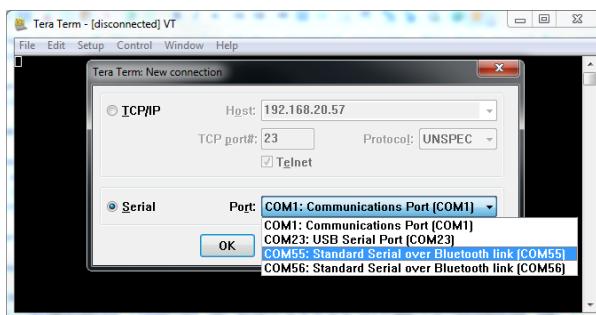
Tera Term (Windows)

Tera Term is one of the more popular Windows terminal programs. It's been around for years, it's open source, and it's simple to use. For Windows users, it's one of the best options out there.

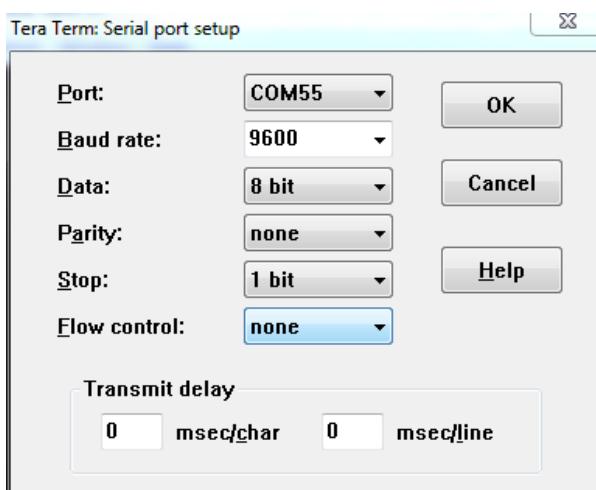
You can download a copy from here. Once you have Tera Term installed, open up it up, and let's poke around.

Making a Connection

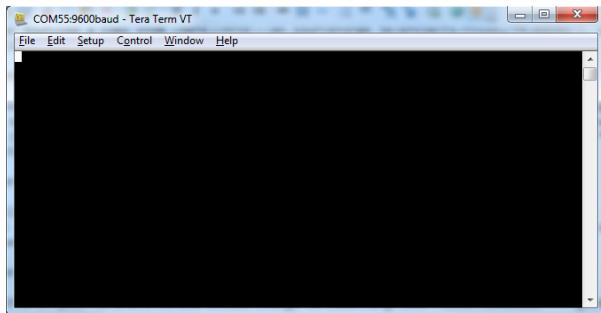
You should initially be presented with a "TeraTerm: New connection" pop-up within the program. Here, you can select which serial port you'd like to open up. Select the "Serial" radio button. Then select your port from the drop-down menu. (If this window doesn't open when you start TeraTerm, you can get here by going to ****File > New connection...***.)



That'll open up the port. TeraTerm defaults to setting the baud rate at 9600 bps (8-N-1). If you need to adjust the serial settings, go up to **Setup > Serial Port**. You'll see a window pop up with a lot of familiar looking serial port settings. Adjust what you need to and hit "OK".



The title of your TeraTerm window should change to something like "COM##:9600baud" -- good sign.

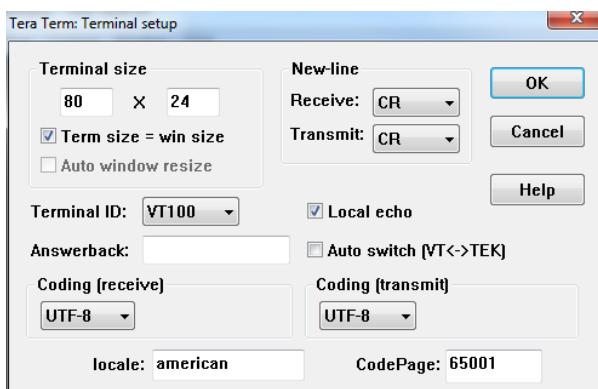


That's about all there is to it. The blank window with the blinking cursor is where data is both sent (by typing it in) and received.

TeraTerm Tips and Tricks

Local Echo

It can be weird to type stuff in the window and not see it show up in the terminal. It's undoubtedly still flowing through the serial terminal to your device, but it can be difficult to type when you don't have any visual feedback for exactly *what* you're typing. You can turn on local echo by going to the **Setup** menu and selecting **Terminal**.



Check the Local echo box if you'd like to turn the feature on.

There are other settings to be made in this window as well. You can adjust the size of the terminal (the values are in terms of characters per row/column), or adjust how new-lines are displayed (either a carriage return, line feed, or both).

Clear Buffer and Clear Screen

If you want to clear your terminal screen you can use either the "Clear buffer" or "Clear screen" commands. Both are located under the **Edit** menu.

Clear screen will do just that, blank out the terminal screen, but any data received will still be preserved in the buffer. Scroll up in the window to have another look at it. Clear buffer deletes the entire buffer of received data -- no more data to scroll up to.

Shortcut Keys

Menus are a pain! If you want to get really fast with TeraTerm, remember some of these shortcuts:

- ALT+N: Connects to a new serial port.
- ALT+I: Disconnects from the current port.
- ALT+V: Pastes text from clipboard to the serial port (not CTRL+V).
- ALT+C: Copy selected text into clipboard (not CTRL+C).
- CTRL+TAB: Switch between two TeraTerm windows.

Real-Term (Windows)

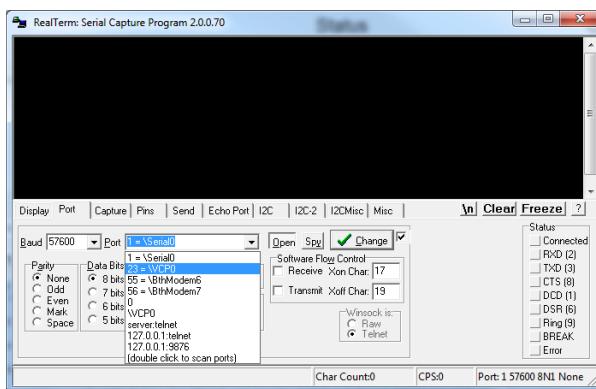
TeraTerm is awesome for simple ASCII-only serial terminal stuff, but what if you need to send a string of binary values ranging from 0-255? For that, we like to use RealTerm. RealTerm is designed specifically for sending binary and other difficult-to-type streams of data.

RealTerm is available to download on their SourceForge page.

Setting Up the Serial Port

When you open up RealTerm, you'll be presented with a blank window like below. The top half is where you'll type data to send, and it'll also display data received. The bottom half is split into a number of tabs where we adjust all of the settings.

Let's get connected! To begin, navigate to the "**Port**" tab. On the "Port" dropdown here, select the number of your COM port. Then, make sure the baud rate and other settings are correct. You can select the baud rate from the dropdown, or type it in manually.



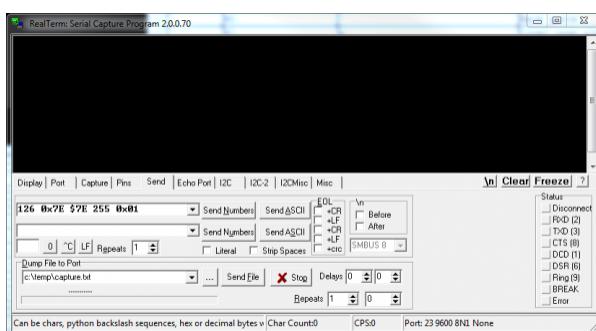
With all of those settings adjusted, you'll have to **click "Open"** twice to close and re-open the port (clicking "Change" doesn't work until after you've established a connection on a COM port).

That's all there is to that! Type stuff in the black ether above to send data, and anything received by the terminal will pop up there too.

Sending Sequences of Values

The ability to send long sequences of binary, hexadecimal, or decimal values is what really sets RealTerm apart from the other terminal programs we've discussed.

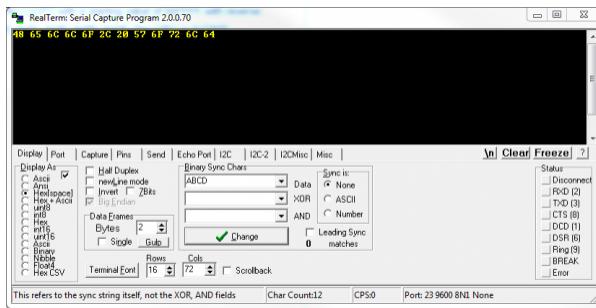
To access this function, head over to the "Send" tab. Then click into either of the two text boxes next to "Send Numbers". This is where you enter your number sequence, each value separated by a space. The numbers can be a decimal value from 0 to 255, or a hexadecimal value, which are prefixed with either a "0x" or a '\$'. Once you have your string typed out, hit "Send Numbers" and away they go!



Why would you need this you ask? Well, let's say you had a Serial Seven Segment Display hooked up to an FTDI Basic, which is connected to your computer. This is a pretty cool setup -- you can control a 7-segment display by just typing in your terminal. But what if you wanted to dim the display? You'd need to send two sequential bytes of value 123 and 0. How would you do that with the handful of keys on a keyboard? Consulting an ASCII table to match binary values to characters, you'd have to press DEL for 127 and CTRL+SHIFT+2 (^@) for 0...or just use the "Send" tab in RealTerm!

Adjusting the Display

Just as you can use RealTerm to send literal binary values, you can also use it to display them. On the "Display" tab, under the "Display As" section are a wide array of terminal display choices. You can have data coming in displayed as standard ASCII characters, or you can have them show up as hex values, or any number of other display types.



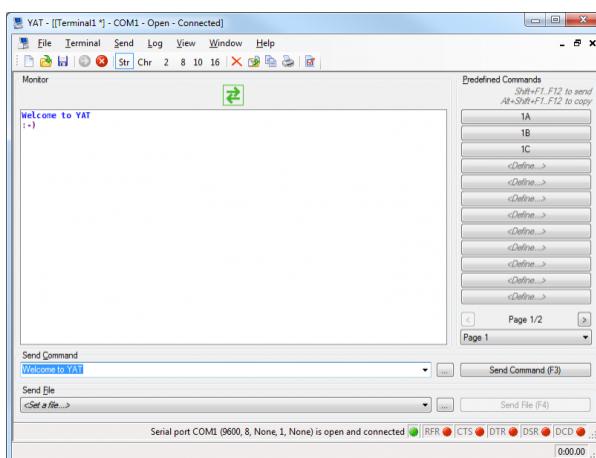
Incoming bytes are displayed as hexadecimal values. Can you decode the secret message?!

RealTerm is preferred for more advanced terminal usage. We'll use it when we need to send specific bytes, but for more basic terminal applications, TeraTerm is our go-to emulator.

YAT - Yet Another Terminal (Windows)

YAT is a user-friendly and feature-rich serial terminal. It features text as well as binary communication, predefined commands, a multiple-document user interface and lots of extras.

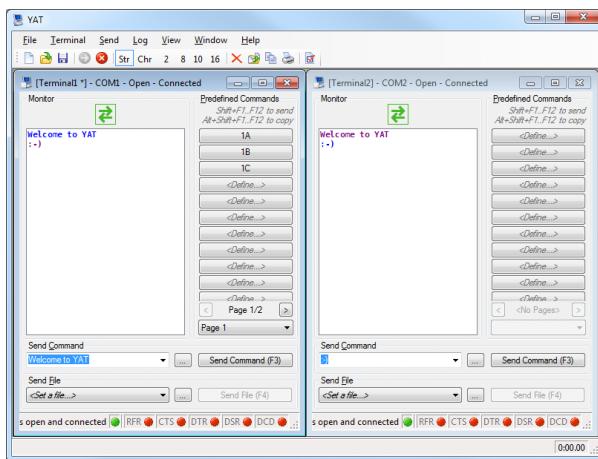
YAT is available to download at SourceForge.



Appearance

YAT features a multiple-document user interface (MDI) that consists of a single workspace with one or more

terminals.



Workspace

Each terminal can be configured according to the device it shall be communicating with. These extra features make a terminal especially easy to use:

- Text command console
- File command list
- Unlimited number of predefined commands
- Drop-down of recent commands

Each terminal has its own monitor to display outgoing and incoming data. The view can be configured as desired:

- Time stamp
- Line number
- End-of-line sequence
- Line length
- Line and bytes transmission rate
- Chronometer

Most of these features can be enabled and configured, or hidden for a cleaner and simpler user interface.



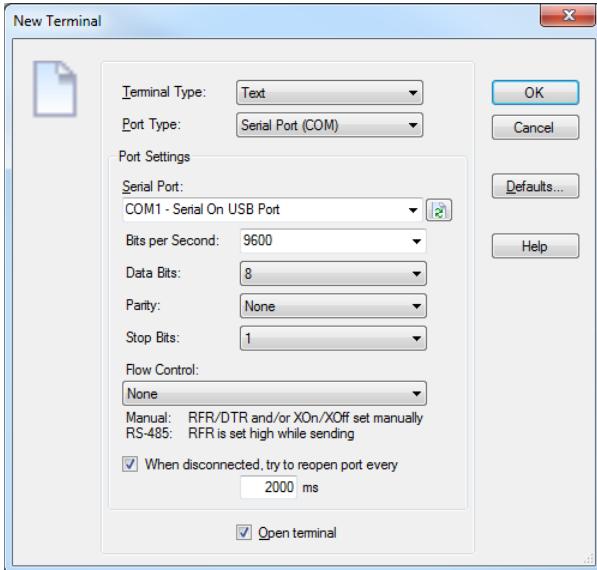
Detailed Monitor



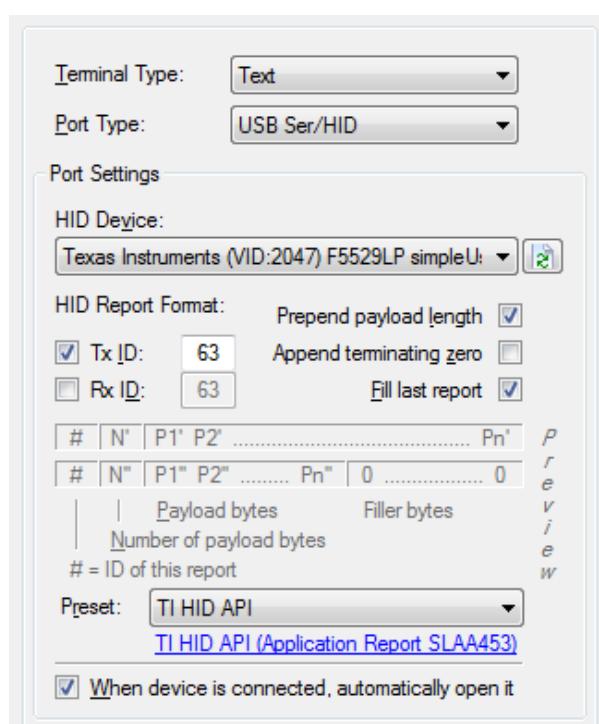
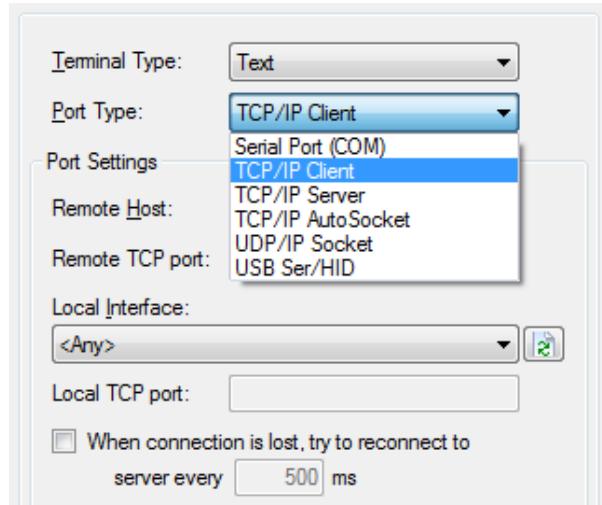
Monitor Status

Terminal Settings

- Text or binary communication
 - Communication port type:
 - Serial Port (COM)
 - TCP/IP Client, Server or AutoSocket
 - UDP/IP Socket
 - USB serial HID
 - Specific settings depending on port type

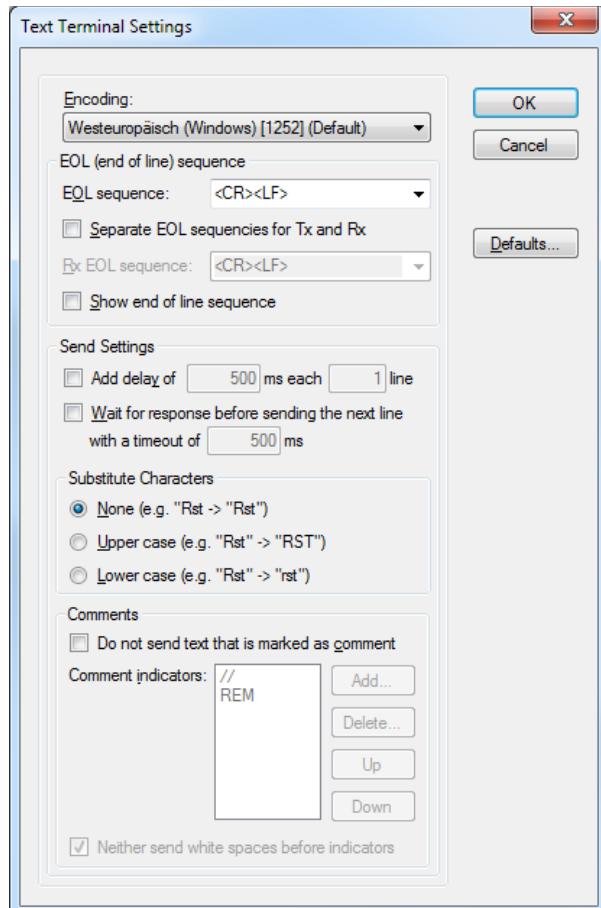


Serial COM Port Settings



Text Terminal Settings

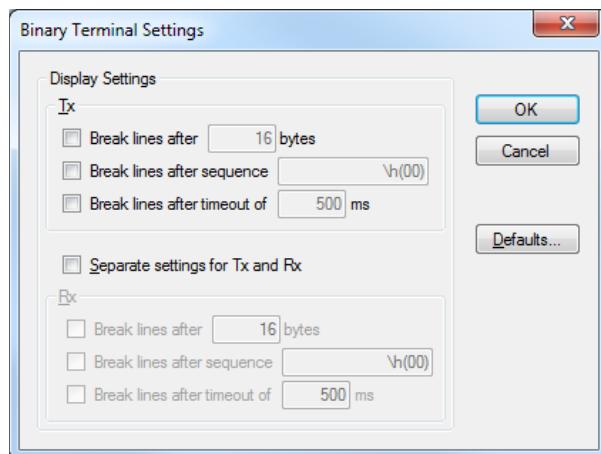
- Full support of any known ASCII and Unicode encoding
- End-of-line configuration
 - Predefined and free-text sequences
 - Possibility to define separate EOL for Tx and Rx
- Send and receive timing options
- Character substitution
- Comment exclusion



Text Terminal Settings

Binary Terminal Settings

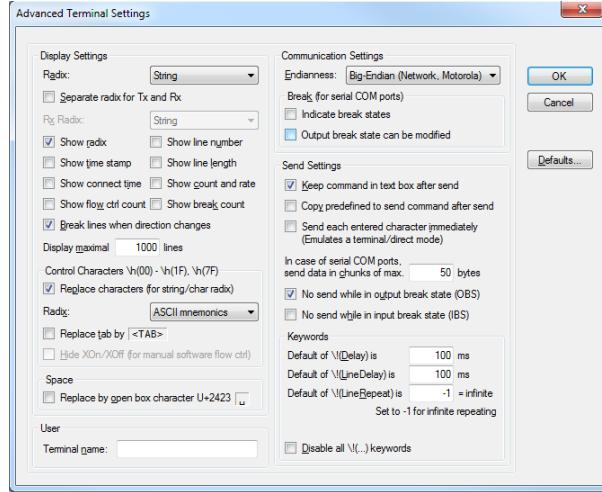
- Configuration of protocol and line representation
- Possibility to define separate settings for Tx and Rx



Binary Terminal Settings

Advanced Settings

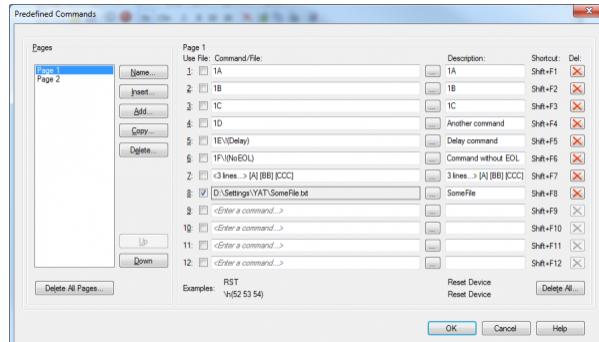
- Various display options
- Various advanced communication options
- Specialized communication options for serial ports (COM)



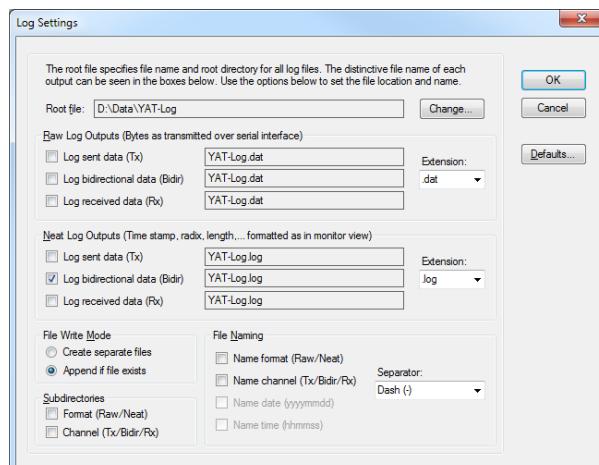
Advanced Settings

Extras

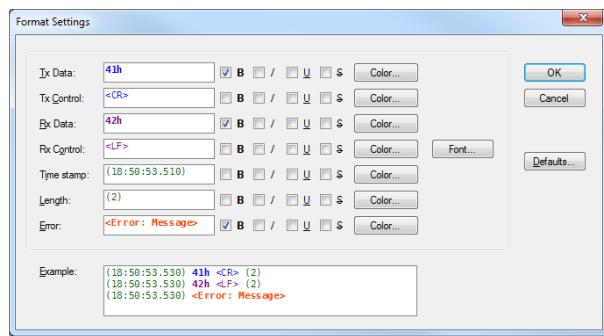
- Escapes for bin/oct/dec/hex like `\h(4F 4B)`
- Escapes for ASCII controls like `<CR><LF>` as well as C-style `\r\n`
- Special commands such as `\!(Delay)`, `\!(LineDelay)` and `\!(LineRepeat)`
- Versatile monitoring and logging of sent and received data
- Formatting options for excellent readability
- Powerful keyboard operation including shortcuts for the most important features
- Versatile shell/PowerShell command line
- x86 (32-bit) and x64 (64-bit) distribution



Predefined Commands



Log Settings



Monitor Format

Change Management and Support

YAT is fully hosted on SourceForge. Feature Requests and Bug Reports can be entered into the according tracker. Both trackers can be filtered and sorted, either using the predefined searches or the list view. Support is provided by a few simple helps integrated into the application, some screenshots on the SourceForge page, and the project's email if none of the above can help.

Development

YAT is implemented in C#.NET using Windows.Forms. The source code is implemented in a very modular way. Utilities and I/O sub-systems can also be used independent on YAT, e.g. for any other .NET based application that needs serial communication, command line handling or just a couple of convenient utilities. Testing is done using an NUnit based test suite. Project documentation is done in OpenOffice. For more details and contributions to YAT, refer to *Help > About*.

CoolTerm (Windows, Mac, Linux)

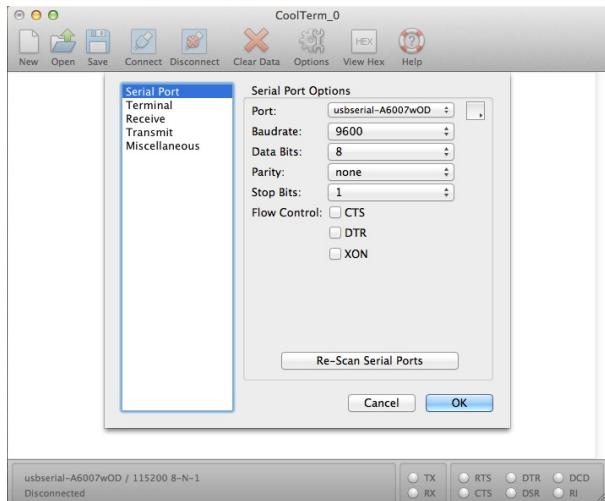
CoolTerm is useful no matter which operating system you're using. However, it is especially useful in Mac OS where there aren't as many terminal options as there are in Windows.

You can download the latest version of CoolTerm [here](#).

Making a Connection

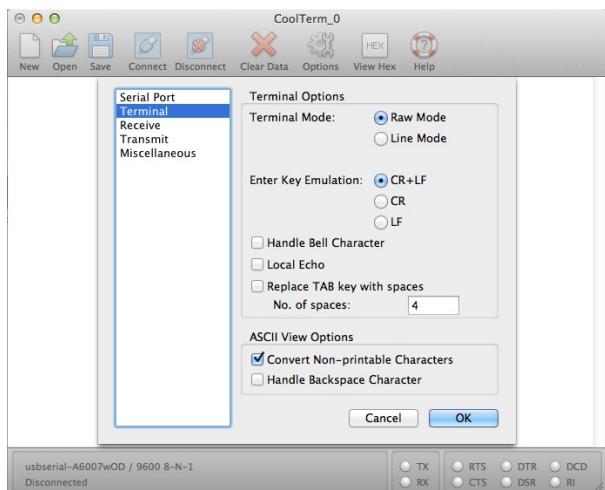
Download and open a CoolTerm window.

To change the settings, click the Options icon with the little gear and wrench. You'll be presented with this menu:



Here, you can select your port, baud rate, bit options, and flow control.

Now click on the Terminal tab on the left.

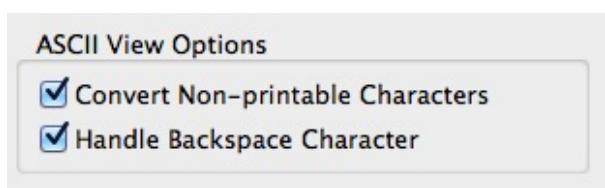


Here, you can change the enter key emulation (carriage return/line feed), turn local echo off or on, and you can switch between line mode and raw mode. Line mode doesn't send data until enter has been pressed. Raw mode sends characters directly to the screen.

Once all your setting are correct, the Connect and Disconnect buttons will open and close the connection. The settings and status of your connection will be displayed in the bottom left corner.

If you need to clear the data in the terminal screen, click the Clear Data icon with the large red X on it.

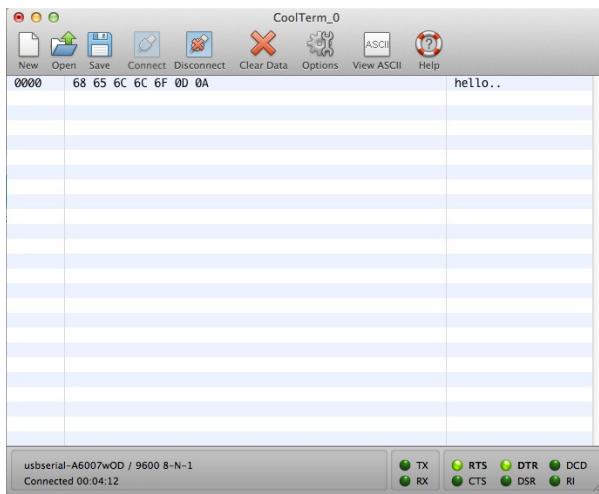
If you're getting annoyed with not being able to use the backspace, turn on 'Handle Backspace Character' under the Terminal tab under Options.



Extended Features

One awesome feature of CoolTerm is Hex View. If you want to see the actual hex values of the data you are sending rather than the ASCII values, Hex View is a tremendous help. Click the View Hex icon. The terminal's

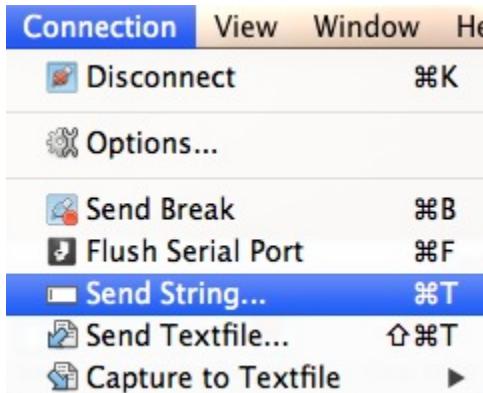
appearance will change slightly. Now whatever you type will show up as hex and ASCII. The first column is just keeping track of line numbers. The second column is the hex values, and the last column is the actual ASCII characters you type.



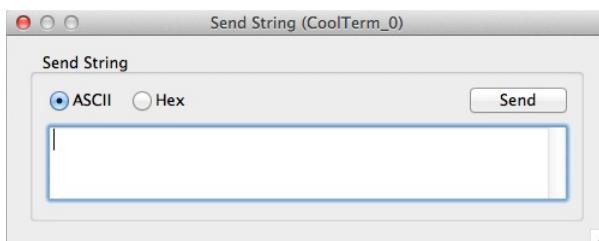
Here I've typed hello and \. Notice the 0D and 0A that appear for carriage return and line feed.

To get back to ASCII mode, click the View ASCII icon.

You can also use the Send String option to send entire strings of text. In the connection menu, select Send String.



You should now have a dialog box with which to send your string in hex or ASCII mode.



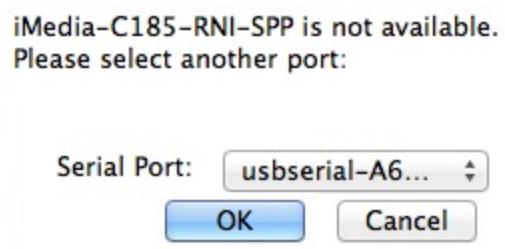
ZTerm (Mac)

You can download the latest version of ZTerm here

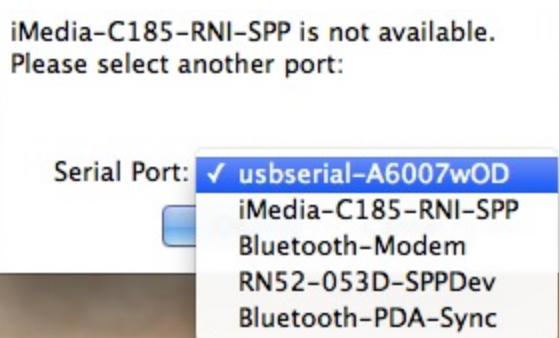
ZTerm is another terminal option for Mac users. Compared to CoolTerm, it seems a lot less user friendly, however, once you find your way around, it's just as useful.

Making a Connection

When you first open ZTerm, you be greeted with this prompt:



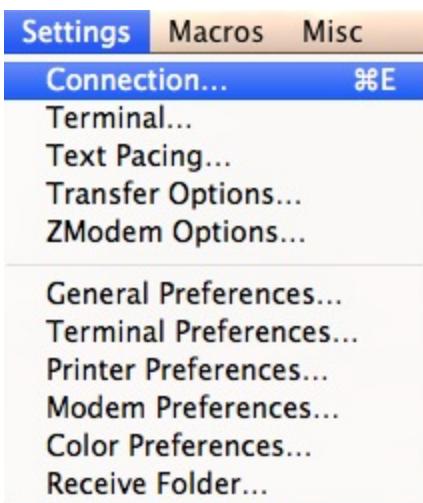
Choose the correct port, and click OK.



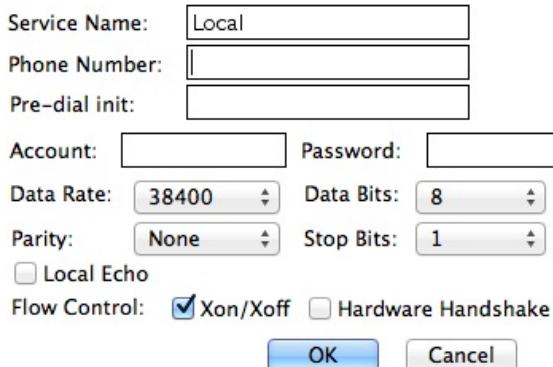
You should now have a blank terminal window.

***Note:** Once you've made a connection, ZTerm will open the most recent connection every time you run it. This can be annoying if you have multiple connections available. To get around this auto connect, hold down the **SHIFT** key as you start ZTerm. This will bypass the auto connect and ask you to which port you'd like to connect.

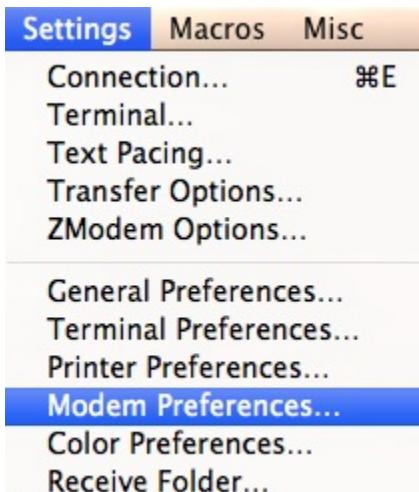
Once you're connected, you can change the terminal settings by going to *Settings -> Connection*.



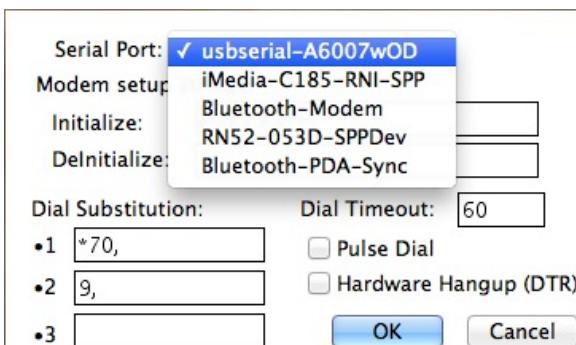
Here you can change the baud rate (data rate); parity, data, and stop bits; flow control; and turn local echo on or off.



If you need to change your port after establishing a connection, go to *Settings -> Modem Preferences*.



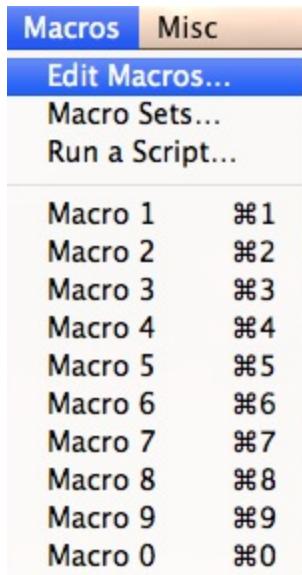
Choose the correct port under the Serial Port dropdown menu.



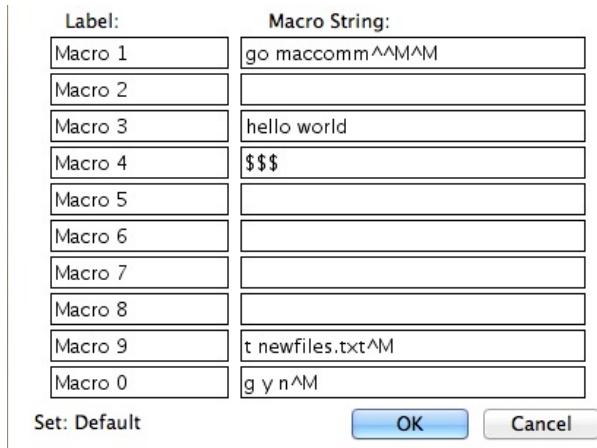
Other Features

ZTerm has lots of other uses for network communication, but that is beyond the scope of this tutorial.

One nice feature that can be used is the macros. Go to *Macros -> Edit Macros*.



Here you can create macros that send whatever strings/commands you'd like. Have a command that you're typing constantly? Make a macro for it!



Command Line (Windows, Mac, Linux)

As mentioned earlier, you *can* use command line interfaces to create serial connections. The major limiting factor is the lack of connection options. Most of the programs we've discussed so far have a slew of options that you can tweak for your specific connection, whereas the command line method is more of a quick and dirty way of connecting to your device in a pinch. Here's how to accomplish this on the three major operating systems.

Terminal and Screen (Mac, Linux)

Mac

Open Terminal. See the Connecting to Your Device section for directions.

Now type `ls /dev/tty.*` to see all available ports.

You can now use the `screen` command to establish a simple serial connection.

Type `screen <port_name> <baud_rate>` to create a connection.

```
| joelmac:~ joel.bartletts$ ls /dev/tty.*  
| /dev/tty.Bluetooth-Modem          /dev/tty.iMedia-C185-RNI-SPP  
| /dev/tty.Bluetooth-PDA-Sync       /dev/tty.usbserial-A6007w0D  
| /dev/tty.RNS2-B53D-SPPDev  
joelmac:~ joel.bartletts$ screen /dev/tty.usbserial-A6007w0D 9600
```

The terminal will go blank with just a cursor. You are now connected to that port!

To disconnect, type `control-a` followed by `control-\`. The screen will then ask if you are sure you want to disconnect.

There are other options you can control from screen, however it is recommended that you only use this method if you are comfortable with the command line. Type `man screen` for a full list of options and commands.

Linux

The `screen` command can also be used in Linux. There are only a few variations from the Mac instructions.

If you do not have `screen` installed, get it with `sudo apt-get install screen`.

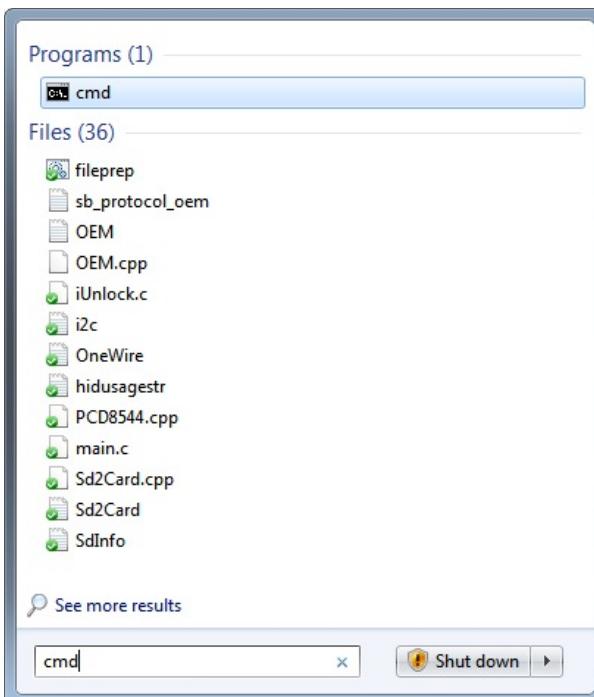
Making a connection is the same as Mac.

To disconnect, type `control-a` then `shift-k`.

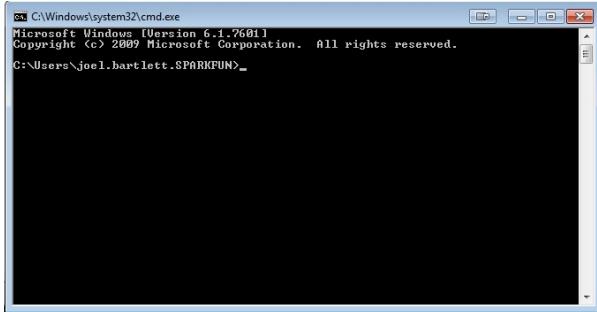
That's all there is to it.

MS-DOS Prompt (Windows)

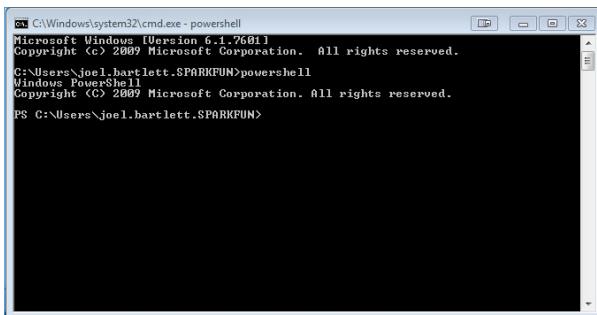
The fastest way to get to the command line in Windows is to click on the start menu, type `cmd` into the search field, and press Enter.



This will open up a blank MS-DOS command line prompt.



To be able to issue Serial commands, you must first enter PowerShell. Type `powershell` to get into PowerShell command mode.



To see a list of all the available COM ports, type

```
[System.IO.Ports.SerialPort]::getportnames()
```

You should now see something like this..



Now create an instance of the port you want with this command

```
$port= new-Object System.IO.Ports.SerialPort COM#,Baudrate,None,8,one
```

With that, you can now connect to and send data to or from that COM port.

```
$port.open()
$port.WriteLine("some string")
$port.ReadLine()
$port.Close()
```

Again, this method of serial communication is only recommended for advanced command line users.

Tips and Tricks

Changing/Deleting COM Ports (Windows)

There may come a time when you need a device to be on a specific COM port. An example of this is, in older versions of TeraTerm, you could only connect to COM ports 16 and below. Thus, if your device was on COM 17, you'd have to change it to connect to it. This problem has been addressed in newer versions of TeraTerm, but there are many other programs out there that only allow a certain number of COM ports.

To get around this, we'll have to dive into Device Manager.

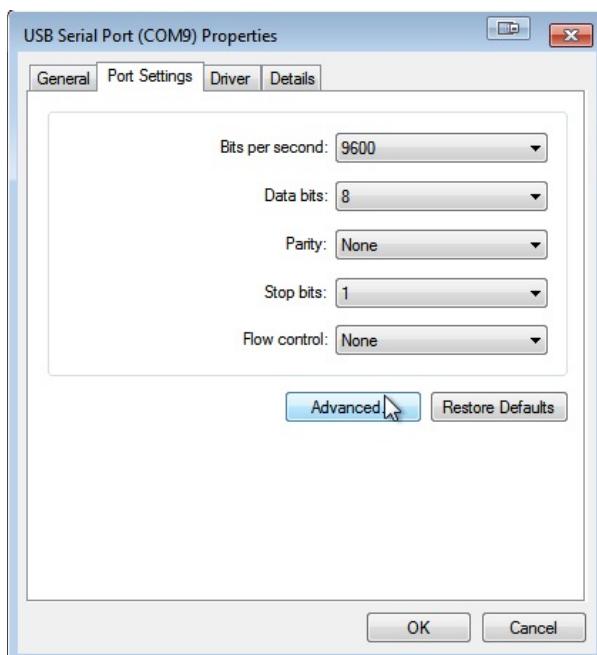
Open Device Manager, and expand the ports tab.



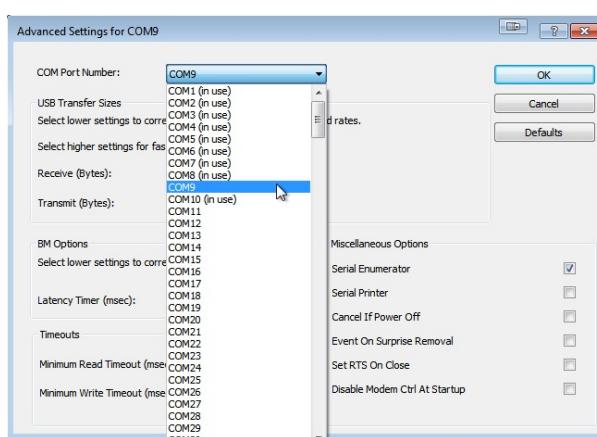
Now right-click on the port you want to alter. Select *Properties*.



In Properties, go to *Port Settings*, and select *Advanced*.



Here, you'll see a drop down menu with all the available COM ports in it. Some of them will have (*in use*) next to them. These are the ports that have been assigned to a serial device.



Notice that COM 9 doesn't have an (*in use*) next to it because that is the port we are currently working with.

If we wanted to change COM 9 to COM 3, we simply select COM 3 in this menu, and click OK. The *(in use)* next to COM 3 should go away. Whatever was connected to COM 9 is now associated with COM 3, and whatever was associated with COM 3 has now been overwritten.

If you need to clear out some old COM ports, you can follow the steps above but for numerous COM ports.

WARNING: Do not select COM 1 when cleaning up old ports. This trick is only for if you really need it and shouldn't be performed very often, for sanity's sake.

TTY vs CU (Mac, Linux)

In Unix and Linux environments, each serial communication port has two parts to it, a `tty.*` and a `cu.*`. When you look at your ports in say the Arduino IDE, you'll see both for one port.



The difference between the two is that a TTY device is used to call into a device/system, and the CU device (call-up) is used to call out of a device/system. Thus, this allows for two-way communication at the same time (full-duplex). This is more important to know if you are doing network communications through a terminal or other program, but it is still a question that comes up frequently. Just know that, for the purposes of this tutorial, always use the `tty` option for serial communication.

Cannot Connect to That Port!

You can only have one connection to a particular port open at any given time (but you can have multiple terminal windows connected to different ports open at the same time). Thus, if you have an Arduino Serial Monitor window open and try to connect to that same port on a different terminal program, it will yell at you and say it could not establish a connection with that port or some such jazz. If you are ever having trouble connecting to a port, make sure it's not open somewhere else.

If you don't have another connection open and still can't connect, make sure all your settings (baud rate, etc.) are correct.

Connected, But Can't See Any Data

If you are connected to the correct port but don't see any data, there are two possible culprits. First check your baud rate. I know I sound like a broken record, but baud rate is the most important setting to match up. Check that baud!

The other culprit could be that the TX and RX lines are reversed. Make sure you have TX->RX and RX->TX.

Programming Arduino and Serial Communication

The Arduino has one dedicated UART, which is just the fancy name for the serial TX and RX lines. It is over these two lines that the Arduino gets programmed. Thus, when working with the Arduino (or other microcontrollers) it's best to avoid using these lines to communicate with other serial devices, especially if you are developing your code and need to upload frequently.

What happens is, if you have another device hooked up to the UART, the data from your computer might not get interpreted correctly by the Arduino leading to code not working the way it's supposed to or not getting uploaded at all.

The same rule applies to serial terminals. If you have a terminal open on the same port that you are trying to

program, it won't work. Arduino will throw some errors about not being able to communicate with that port. If this happens, close your connection, and try again.

One simple way around this is to use the Software Serial Library built into Arduino to create a separate UART for outside serial communication. That way, your Arduino can communicate on one port while still leaving the default UART open for programming.

Resources and Going Further

That was a lot of information! At the very least, you should walk away from this knowing what a terminal window is, how to use it, which terminal program is best suited for you and your operating system, and how to navigate that program's interface. Again, terminal programs are a very powerful tool when working with serial devices and microcontrollers. Now go collect some data!

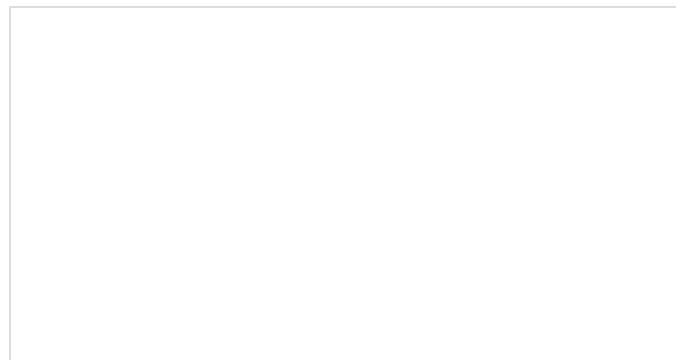
If you'd like to know more about different types of communication, visit these tutorials:

- Serial Peripheral Interface
- I²C

To see some products that require the use of a serial terminal, check out these hook-up guides:

- OpenLog Hookup Guide
- Bluetooth Mate and BlueSMiRF Hookup
- RN-52 Audio Bluetooth Hookup

Or check out this blog related post:



Where does 9600 bps come from?

DECEMBER 5, 2016

Your favorite terminal didn't make the list? Tell us which terminal emulator is your favorite and why in the discussion section.

Interested in learning more foundational topics?

See our **Engineering Essentials** page for a full list of cornerstone topics surrounding electrical engineering.

TAKE ME THERE!