CHANGING/DELETING COM PORTS (WINDOWS)

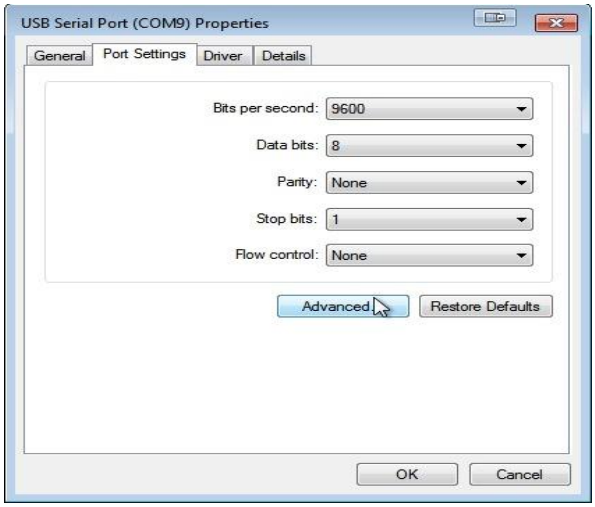
There may come a time when you need a device to be on a specific COM port. An example of this is, in older versions of TeraTerm, you could only connect to COM ports 16 and below. Thus, if your device was on COM 17, you'd have to change it to connect to it. This problem has been addressed in newer versions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

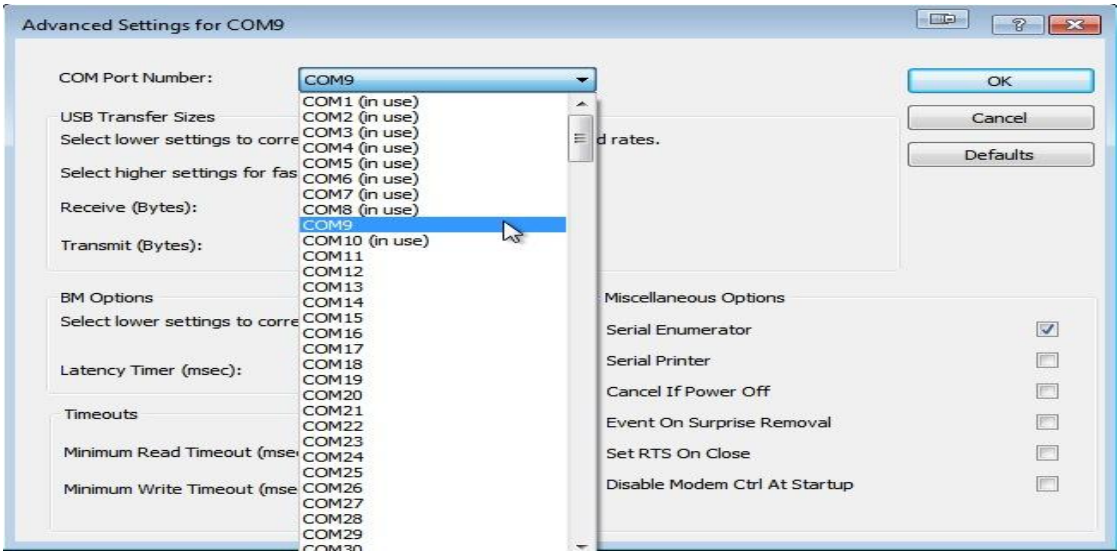
Open Device Manger, and expand the ports tab.

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.

Source: https://learn.sparkfun.com/tutorials/terminal-basics/tips-and-tricks