How to Program and Debug RS-485 Networks

Q: Some of the data sheets for RS-485 devices mention a 9-bit UART protocol. What does it do?

A: In a network with many devices you need a way to identify each one uniquely. You don't want valve-actuator data going to a pump-motor controller! Although most of us think of a universal asynchronous receiver transmitter (UART) as an 8-bit device, some UART integrated circuits allow a ninth bit. The state of this "extra" bit (D8) identifies the eight bits of information within the data (D7-D0) as an address or as data. A logic-1 in the D8 causes the receiving UART–also set for 9-bit mode–to save the data in a register the device can check for an address match. If the addresses match, the device will accept following bytes as information to act on. When sending bytes of data, the D8 address bit gets set to a logic-0.

If the addresses do not match, the device ignores all communications until another address comes along on the bus. Then it goes through the address-match steps again.

Q: I understand how the RS-485 bus works, so how do I actually control devices on the network?

A: The RS-485 specification covers only the signaling, timing, and electrical operations. How you use the bus depends on what you want to do, the amount of development time available, and so on.

When you move on to applications you have a choice of many programs and protocols. The Modbus serial-line protocol, for example, operates as a master-slave system that comprises one master and several slaves. The master transmits information to the slaves, which can reply when requested. Slaves do not communicate with each other and don't initiate communications with the master. For more information, see: "MODBUS over Serial Line Specification & Implementation Guide," http://www.modbus.org/docs/Modbus_over_serial_line_V1.pdf. Modbus communications place few restrictions on developers and it's an open standard without royalties.

RS-485 communications even control model railroads. Find details here: http://www.bidib.org/bidibus/bidibus_e.html#T2
and here: http://www.lenzusa.com/techinfo/xpressnetfaq.htm.

Q: If I set up an RS-422 or RS-485 network and it doesn't work, how do I test it?

A: Here are several steps that might help:

1. Use an oscilloscope to determine whether or not a driver can put signals on the bus. A dual-trace scope lets you examine both differential signals referenced to ground. Also look for noise on the bus. Decrease the sweep time to look for 60- or 50-Hz noise caused by ground loops. Often low-frequency gets blurred at high scope

sample rates or sweep speeds.
Also consider using an RS-485 bus analyzer. Several companies sell analyzer hardware and software. A small USB-style logic analyzer such as the Saleae Logic-8 could also help decode RS-485 communications.

2. When no driver connects to the bus, does the Data Out + line have a higher voltage than the Data Out – line? If not, check the pull-up and pull-down resistors on the bus. You want the bus in a known state when no driver transmits. If you test an RS-485 device on its own, enable or connect pull-up resistors so you can look at real logic levels. A three-state bus in its high-impedance mode can be difficult to check.
3. If you see an oscilloscope signal that looks right for the selected data rate, but a device doesn't "see" its address and data, ensure all networked devices have the same communication settings. These include the data-bit rate (baud rate), number of data bits, number of stop bits, parity set for on or off, and the type type of parity (if any). All devices must have the same configuration to exchange information. Devices must have unique addresses.
4. Have you properly chosen 9-bit operations for your network? Do all receivers and transmitters have the 9-bit mode selected? Have you properly set the addresses of bus receivers?
5. Check the termination resistors at the end of your network as described earlier. You should have only two 120-ohm resistors; one at the most-distant driver and the other at the most distant receiver.
6. Does your data rate exceed the limit for your bus length?
7. Look for possible incorrect connections. The A line (Data +) and B line (Data -) must connect properly at each device.
8. If you use a cable with several twisted pairs, do all connections use the same pair for half-duplex communications, or the same two pairs for full-duplex communications.
9. Try a loop-back test. Disconnect one device from the network, preferably a device connected to a PC on which you can run test software. If your RS-485 interface has separate Data Out (driver) and Data In (receiver) connections, connect the Data-Out + to the Data-In + contact, and connect the Data-Out – to the Data-In – contact. Then you can transmit information and examine the data received.
If you have a 2-wire RS-485 half-duplex interface, it might include a "No Echo" circuit that disables the receiver when the driver transmits information. In this case, you need another RS-485 device to use as a receiver.

Use the Sealevel Systems WinSSD® Serial Diagnostic Utility, available as a free download. This software features a "…synchronous/asynchronous diagnostic utility for Windows. For asynchronous serial troubleshooting, WinSSD allows the user to modify the default UART parameters, perform external loopback tests, toggle modem control signals and transmit test pattern messages. Included applications allow terminal mode operations, bit error rate testing, and throughput monitoring. When used with a Sealevel synchronous serial adapter, the user has full control over electrical interface, framing method, RSET and TSET source, transmitter and receiver bit rate, oscillator frequency, CRC, preamble, clock encoding, sync character, and more."