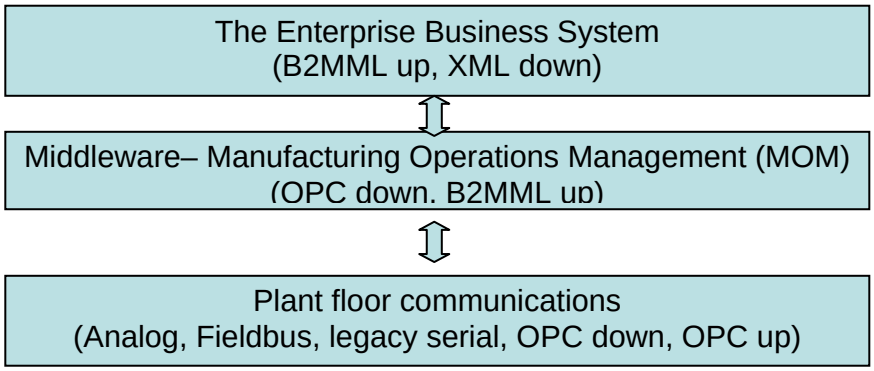
# Ethernet Enabling Serial Devices in High Availability Manufacturing?

*Written by: Paul Wacker, Advantech Corporation, Industrial Automation Group*

Today, manufacturing organizations world wide are working on reducing complexity in the manufacturing process and increasing the transparency of processes throughout the enterprise.

The ability to communicate data from any part of the enterprise to any other part of the enterprise is core to making modern manufacturing work.




The venerable Purdue model of manufacturing has been compressed to the three layers shown here. There is the plant floor, and all the communications that occur there. There is the MOM system, which connects the plant floor to the manufacturing business systems, and to the enterprise business system.

On the plant floor, there are communications tasks between field devices and machines, the controllers on the floor, and the control systems that sit above the controller level, the unit control systems in discrete manufacturing and the DCS (distributed control systems) in process automation. Between the plant floor and the business systems there are communications requirements for instantaneous communications between the MOM systems and the other systems like asset management, production control, and production management.

In some cases, loss of communications between any of these devices and systems could shut down the line, or process train, or even shut down the plant itself. It is absolutely essential that real time manufacturing have non-stop communications that are reliable and robust.

**The Legacy of Serial Devices**

Early in the history of industrial computing, serial interfaces such as RS-232 became common as devices such as PLCs required programming and it became necessary to transmit data to computers running spreadsheets and databases for data collection and analysis. But RS-232 has significant drawbacks as a means of communicating between many devices. RS-232 is limited to a dedicated connection, with one PC or host per serial device, and with a definite physical limitation of 50 feet from device to device. Another standard, RS-422, provided for longer distance communication, up to 3600 feet, but was still extremely limited. The RS-232 standard provided for communication speeds that are quite slow in comparison with modern data transmission speed: 300 to 9600 bits per second, as opposed to, for example, Ethernet communications, which may be up to 10 Gigabits per second. The multi-drop version of the RS-232/422 standard was marginally faster and could have several slave units connected to a single master. Neither RS-232 nor RS-422 is scaleable to the level of multi-device communications over an entire factory floor.

These communication interfaces were “brittle.” That is, custom application program interfaces, or APIs, were required for every device-to-device transmission, and a minor change in an API, or in the data being transmitted, could break the communications method entirely. Even the advent of Microsoft Windows and DLL libraries still required dedicated device drivers to be loaded on the host for each device on the serial loop. The effect of this brittleness on uptime is uniformly negative.

Asynchronous serial communications connections, such as RS-232 and Modbus, are still among the most commonly used industrial device interfaces. Common uses include configuration and setup, operator interface (HMI), production setup (batch download), and even production monitoring, as well as troubleshooting and diagnostics. Serial port output is common on a wide variety of devices, from clean-room particle counters, to vision systems, to marquee displays, to scales, scanners, and of course PLCs and PACs on the plant floor.

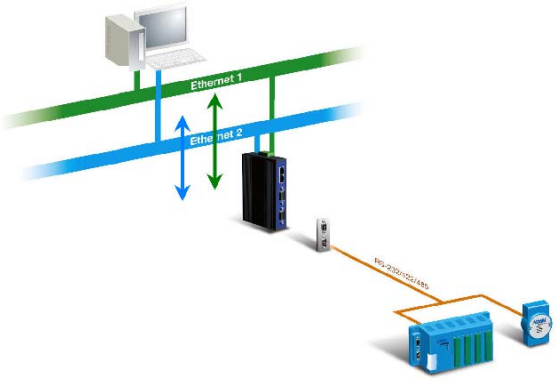
Enter the Ethernet serial device server. This device operates as a portal through which serial data can be acquired, routed through the Ethernet network, and used by both new applications and legacy applications that used to run on a PC that had to be directly connected to the serial device.

**Ethernet and the Challenge of Redundancy**

The ubiquitous use of Ethernet on the plant floor, the laboratory, and in the test lab has made it possible to solve the distance and interface limitations of serial devices. Instead of the unique one cable-one device system required by even standard Modbus systems, Ethernet permits the use of servers and gateways through which the serial data is transported over the Ethernet network to the recipient host.

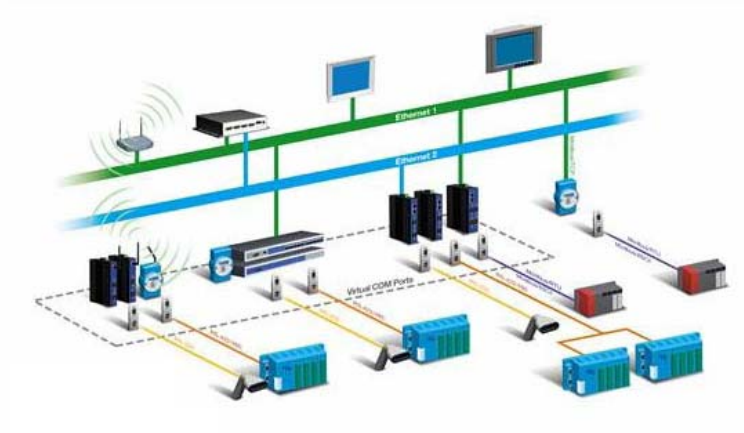
But what happens if the communications links are interrupted? It is simple and relatively inexpensive to install and operate a redundant link. Dual and independent network connections ensure that any single-point failure will not interrupt communication. If one network fails, the other allows communication to continue without fault.

Both the LAN connections and the power connections are duplicated in this model, either as redundant DC power inputs or as primary DC power with battery backup power supply. This provides two ways to make sure that the dual-LAN serial device server has continuous uptime, especially in the event of a power-down due to a machine failure on the plant floor. This means that diagnostic data will get through, even though the machine has failed.

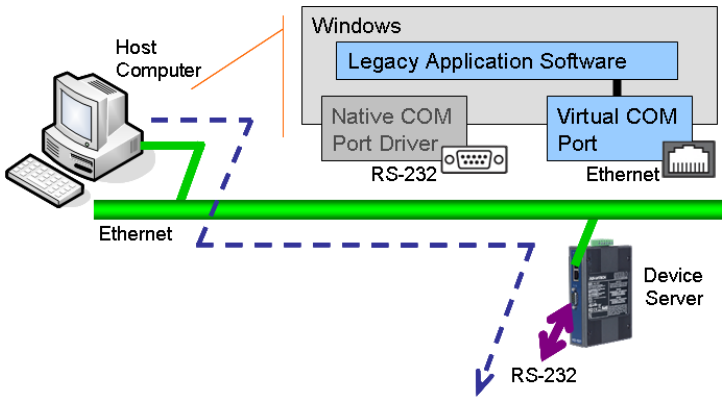


The key to setting up redundant serial communications with the plant network is to first look at the way the network is structured. In the example below, we have a redundant Ethernet LAN, with a number of virtual COM ports providing access to the serial devices on the plant floor: bar code readers, programmable controllers, and other devices. Note that the LAN is capable of being accessed both wired and wirelessly, without making any difference to the redundancy of the network. This allows the easy integration of legacy Windows applications that use serial data inputs into a modern plant floor network.

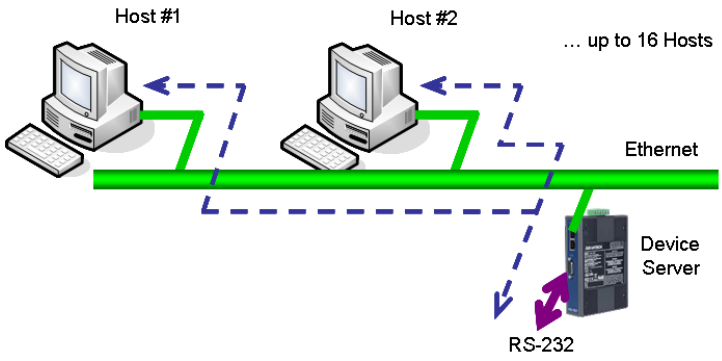


The COM port redirection software is installed on the computer somewhere on the LAN. The remote devices are addressed as virtual COM (VCOM) ports. The entire system behaves, as far as the serial devices and host computer are concerned, as if there is a hard-wired direct connection between the serial device and the COM port on the computer. This is made possible by serial tunneling. Serial data is encapsulated in IP packets and transported over the Ethernet network, just as if it were standard TCP/IP data. Operation is generally transparent to both applications software and connected devices requiring few if any changes, and transfer is bidirectional – data can be both sent and received. One of the reasons for the continued success of the Modbus protocol is the very early port of the protocol for use in a TCP based Ethernet network. Tunneling permits flexible configurations like serial device to serial device, PC to serial device, and serial device to PC.




Most importantly, the data from the serial devices on the network is redundantly transmitted to the computer that is running the legacy application software.

In addition to VCOM operation, a modern serial device server can also be operated in UDP and TCP modes. UDP or User Datagram Protocol is designed to be a lightweight messaging protocol for resource constrained or time sensitive applications. It doesn't guarantee message order or delivery. What UDP does, however, is to break the "one device, one cable" paradigm. The serial data server can be configured in UDP Client Mode, allowing one way communication from up to 16 hosts, or in UDP



Server Mode, allowing bidirectional communication from 8 hosts. So each device connects to the device server, but from there to the rest of the network, the communications are over the Ethernet LAN, redundantly connected. These configurations are often used to implement serial device-based data collection systems, such as bar code readers or distributed processing systems using PACs with serial data connections.

A modern serial device server should also be capable of operating in TCP Server Mode. This enables the use of serial devices, including legacy serial devices with OPC data interchange servers. Modern OPC versions, including the newest, OPC-UA, support "serial encapsulation" of data and OPC servers can be configured to use device servers as OPC clients. Device servers can also be configured to operate over 802.11b wireless Ethernet LAN connections, so the limitations of wiring may be done away with completely.

**The Very Model of a Modern Serial Device Server**

So, what sort of specifications should a modern Ethernet serial device server have? The device server should have the ability to host multiple serial ports. Typical configurations include 1, 2 and 4 port versions, with DB9 serial port connections for easy field connectibility.

The device server should be designed for field mounting, in a DIN-rail configuration, with as little space required as possible. This allows mounting in already cramped cabinets and on existing field DIN-rails.

The device server should have fully redundant power supplies, and be capable of running on just one power input, with a failure trip output on input power fault.


The device server should have dual LAN ports with redundant COMport redirector. Each LAN port should have an independent Media Access Controller (MAC) and a separate IP address, and be capable of operating on the same or different subnets. It should be configured for at least four multiple simultaneous host connections running up to 16 hosts concurrently. And the device server needs to be fast: under 10 ms, permitting applications with ultra fast data, such as motion control.

Redundant Ethernet serial device servers ensure the reliability and robustness that modern factory communications demand. They lengthen the ability of legacy serial devices and legacy software to continue to be used, and they make it possible for serial data communications to continue to operate on the factory floor well into the future.

###