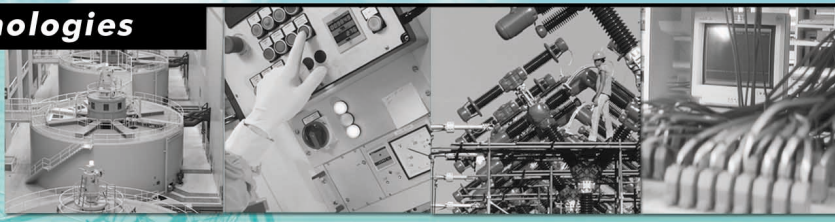
The Serial-to-Ethernet Guidebook

Solutions Utilizing Serial Device Server Technology

Moxa Technologies



PUBLISHED BY

Moxa Technologies Co., Ltd.

Developer of Network and Serial Communications Solutions

For more information, please contact us at:

Address: F4, No.135, Lane 235, Pao-Chiao Rd.,
Shing-Tien City, Taipei, Taiwan, R.O.C.

Tel. : +886-2-8919-1230

Fax. : +886-2-8919-1231

E-mail : info@moxa.com.tw.

http:// www.moxa.com

ISBN: 986-99976-1-9

1st Printing December 2001
2nd Printing February 2002
3rd Printing February 2004

Preface

The serial-to-Ethernet products discussed in this book are referred to as *serial device servers*, and have a relatively short history, having been developed first by Moxa Technologies Co., Inc. in 1994. With the popularity of serial device servers tied closely to the increasing prevalence of Ethernet LANs in both business and industrial settings, serial-to-Ethernet products have begun to take off in recent years, and are certainly destined to become a major player in the communications and industrial automation markets. In fact, although the marriage of *serial* with *Ethernet* was originally one of convenience, it is quickly becoming a matter of necessity.

Our primary aim in writing this book is to provide potential users of serial-to-Ethernet products with the knowledge required to choose the most appropriate product and operation mode for their particular application. Moxa's 15-year history in the development and marketing of many successful communications products, including a complete line of serial device servers, gives us the authority to speak from a knowledgeable and informative viewpoint about serial-to-Ethernet concepts. All of our products undergo a rigorous regimen of research and testing before being put on the market, with the knowledge gained serving as a solid foundation for educating the industry about our technology.

Chapter 1 Introduction

The purpose of this Serial-to-Ethernet Guidebook is to provide Product Managers and Sales Personnel working in the communications industry with a general introduction to the terminology and methodology of the serial-to-Ethernet field.

Chapter 2 Serial-to-Ethernet Applications

Serial-to-Ethernet refers to any product or process used in the marriage of the serial and Ethernet interfaces. In general, this is an important field for both business and industry, since millions of legacy serial devices, most without built-in Ethernet ports, are still in common use today. A *serial device server* allows companies to connect legacy serial devices to an Ethernet LAN/WAN, providing many more options for data acquisition, device management, and industrial control than would otherwise be available.

Chapter 3 Case Studies

In this chapter, we examine five separate examples that illustrate the implementation of serial device server technology. There are, of course, many other situations to which device server technology can be applied. But even though your own specific application may not appear, as you read through this chapter you should be able to see how to adapt the examples to suit your needs.

Chapter 4 Finding the Right Solution

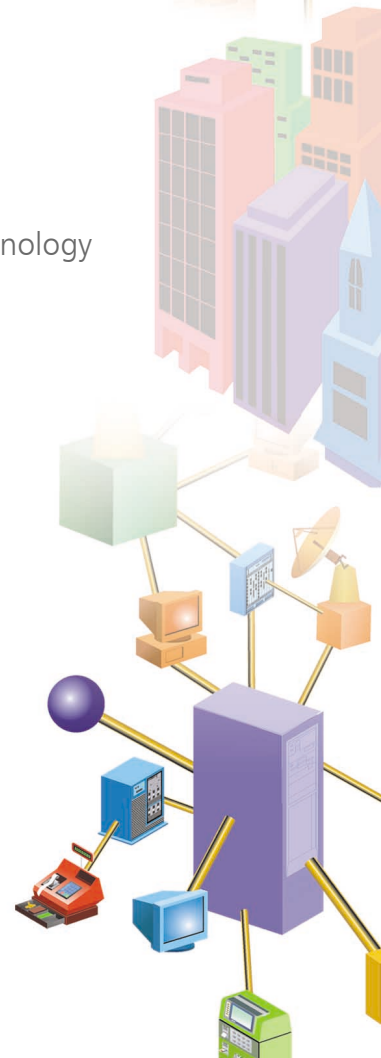
After reading this far, you should have a good idea about the features and functions of serial device servers. The problem now is how to find the serial device server product most appropriate for your application. Keep in mind that a good serial device server supplier should provide you with a product that has most of the previously mentioned features, such as a Windows driver, TCP/UDP Client/Server, Ethernet Modem, and Pair Connection. In addition, take into account agency approvals, such as CE, FCC, UL, CUL, TÜV, since they are a good indicator of product quality.

Table of Contents

Chapter 1.

Introduction

- Device Server Technology
- Impact of Ethernet on Industry
- Benefits of Serial-to-Ethernet Technology



The purpose of this Serial-to-Ethernet Guidebook is to provide Product Managers and Sales Personnel working in the communications industry with a general introduction to the terminology and methodology of the serial-to-Ethernet field. After reading the book, Product Managers should have enough technical expertise to discuss the implementation of serial-to-Ethernet solutions with their hardware and software engineering colleagues, and Sales Personnel should have the knowledge required to suggest appropriate serial-to-Ethernet solutions to a variety of customers. In addition, both serial device manufacturers and serial device users should gain enough perspective on the subject to know that including serial-to-Ethernet products in their 21st Century communications solutions is the way to remain competitive.

Device Server Technology

Several asynchronous serial interfaces, including RS-232, RS-422, RS-485, LonWorks, CANbus, Profibus, and Interbus are currently in use to connect various types of devices, such as sensors, card readers, meters, and PLCs. In this book, however, we concentrate on products and applications that require access to RS-232, RS-422, and RS-485 interface devices via an Ethernet connection, since these three serial interfaces are the most widely used in industry.

The transformation between the serial and Ethernet interfaces takes place at the electronic signal and network protocol levels, such as in the transformation of data from the RS-232 format into a format suitable for a TCP/IP network. To do this requires Device Server technology, in which a *Device Server*, or more specifically a *Serial Device Server*, is a smart, standalone device with a tiny embedded operating system and CPU that is, nevertheless, large enough to contain its own operating system, as well as the necessary software protocols, such as the TCP/IP stack. A Serial Device Server also comes equipped with the required hardware interfaces, such as RS-232, RS-422, and RS-485 ports. The device server can transfer, and even process data between the serial and Ethernet interfaces to carry out predefined tasks.

Impact of Ethernet on Industry

Ethernet has been the backbone of digital communications in the IT industry for some time now. Industry on the other hand, and in particular, the industrial automation field, has been slow to follow suit. But the move to using Ethernet as an overall communications backbone is picking up steam, and many industry insiders believe that within a relatively short period of time, Ethernet will not only be used at the information level, but also at the control and device levels. Due to the fact that many legacy serial devices do not have a built-in capability to connect to an Ethernet, Serial Device Servers are playing a pivotal role in the adoption of Ethernet technology.

Benefits of Serial-to-Ethernet Technology

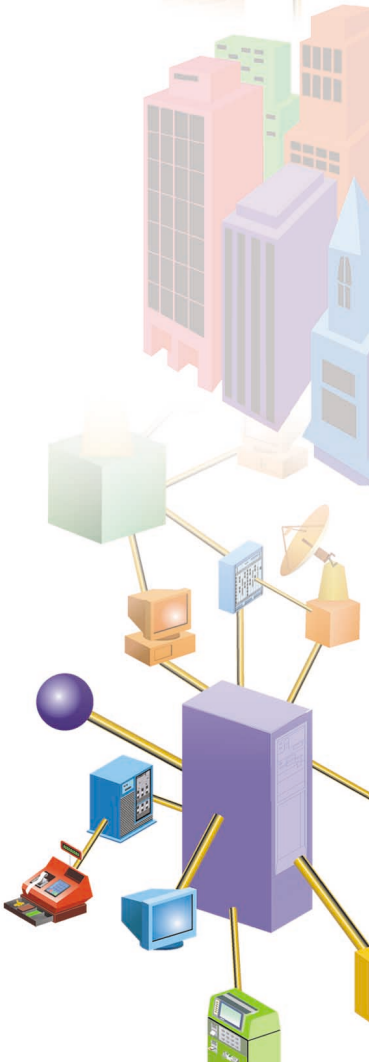
Serial Device Server products have been used in many fields due to their powerful ability to network-enable legacy serial devices. The major benefits are:

- Brings the TCP/IP platform to serial devices, streamlining business management and operation.
- Low cost management of serial devices from the TCP/IP platform. Remote, mobile management greatly reduces the cost of system downtime and human labor costs.

Chapter 2.

Serial-to-Ethernet Applications

- Device Server Operation Modes
- Real COM Mode
- Socket Mode
  - TCP Server
  - TCP Client
  - UDP Server / Client



Serial-to-Ethernet refers to any product or process used in the marriage of the serial and Ethernet interfaces. In general, this is an important field for both business and industry, since millions of legacy serial devices, most without built-in Ethernet ports, are still in common use today. A *serial device server* allows companies to connect legacy serial devices to an Ethernet LAN/WAN, providing many more options for data acquisition, device management, and industrial control than would otherwise be available.

Device Server Operation Modes

In this section, we describe the various device server operation modes. The options include operation modes that use a driver installed on the host computer, operation modes that rely on TCP/IP socket programming concepts, and operation modes that typically involve communication between pairs of appropriately configured device servers.

The operation modes described here are suitable for a wide range of applications, including

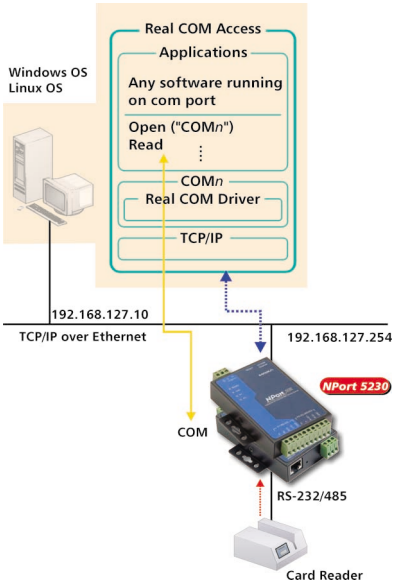
- Data acquisition
- Factory automation
- Security/Attendance/Entrance control
- Medical automation

Real COM Mode

Most serial device server manufacturers provide native Real drivers that work with Windows 95/98/ME and Windows NT/2000 systems, and many also provide fixed tty and real tty drivers for the Linux and Unix operating systems. The driver establishes a *transparent* connection between host and serial device by creating a virtual local COM/tty port on the host computer for each of the serial device server's serial ports. This is the quintessential point, since it means that users do not need to rewrite or purchase new serial communications software for their applications. In fact, the driver for a serial device server is often referred to as *port redirector software* since its primary function is to intercept what the computer thinks are data intended for its serial port, and redirect the data through the computer's Ethernet card. A true Real COM driver not only transfers data between remote serial port and PC, but is also able to manipulate the serial port's line signals, such as the RTS, CTS, DTR, DSR, and DCD pins. In fact, a Real COM driver provides you with the maximum ability to control most of the world's serial devices.

Real COM Mode: Single-Host application

Single-Host application is a natural extension of the host-to-device setup in which a serial device, such as a card reader, is connected directly to a PC's COM port. That is, one host connects to the serial device, and data acquisition and control of the device is limited to this one host. Only one host computer is allowed access to the serial devices connected to the device server's serial ports.



The figure shown here nicely illustrates a typical scenario of a device server configured for Single-Host application. In this case, a Card Reader is attached to a device server with IP address 192.168.127.254. The device server's Ethernet port is attached to a LAN, generally by using a straight-through Ethernet cable to connect to a hub or switch. A host computer with IP address set to 192.168.127.10 is connected to the same LAN.

The schematic shows the general path taken as data are passed from the Card Reader, out onto the Ethernet, through the host computer, and then up to the user's application. What you should keep in mind is that the serial device server automatically prepares the serial data for transport across the Ethernet.

The device server's firmware contains the full TCP/IP protocol stack, allowing the serial data to be appropriately packed into a TCP packet, formed into an Ethernet frame, and then sent to the host's Ethernet card. The host computer passes the packet up through its own TCP/IP protocol stack, with the serial data safely delivered to whichever of the host's applications has been charged with receiving the data.

Real COM Mode: Multi-Host application

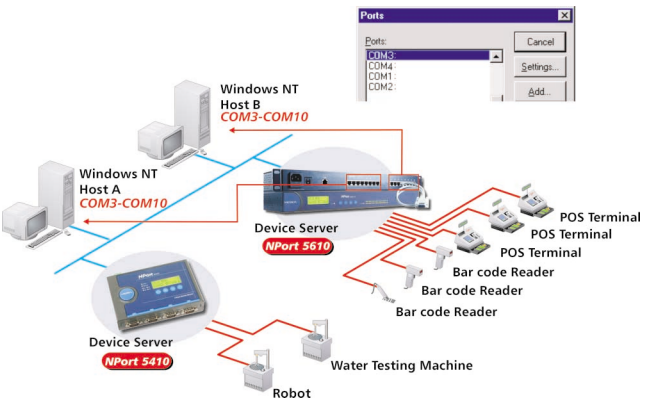
Multi-Host application provides a great deal of versatility in setting up an environment in which one or more computers can access multiple serial devices, and includes the option of providing access over the Internet. To be precise, Multi-Host application can be characterized as follows:

1. More than one host is allowed access to the serial devices attached to the device server's serial ports.
2. The hosts and device server can be located on different LANs. (That is, the device server can be configured so that transmitted data are allowed to pass through one or more routers.)

We can distinguish between two different situations, depending on how the device server's individual ports are configured.

Server Sharing

Server Sharing applies specifically to multi-port device servers, and allows one device server to be configured so that two or more hosts share the same device server, but with each host retaining exclusive control over one or more of the device server's serial ports.

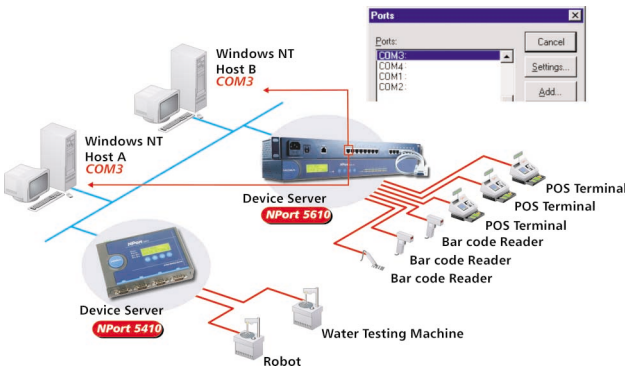


For example, as illustrated in the figure shown here, a 16-port serial device server shares its ports with two different hosts. Host A is given exclusive control of ports 1 to 8, and Host B has exclusive control of ports 9 to 16. This type of setup allows for greater economy, since the user is able to make better use of *all* of the device server's serial ports. You should also not be confused by the fact that the two sets of ports are both labeled COM3 to COM10. Since the COM port names exist on two different hosts, each with its own operating system, it is obvious that no conflicts will arise.

Port Sharing

Asynchronous Port Sharing

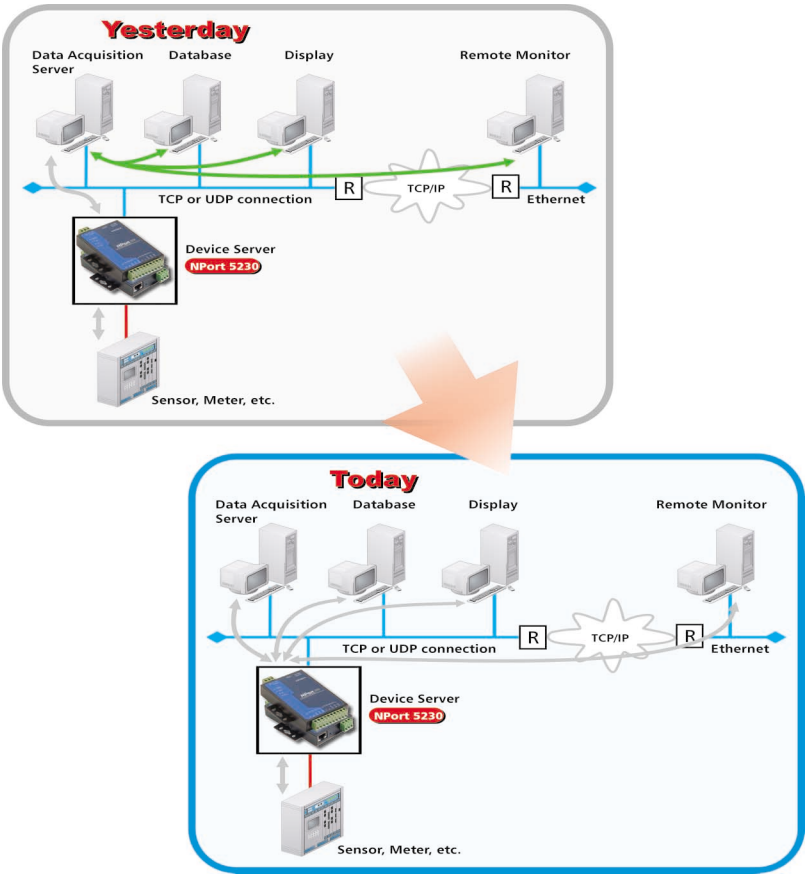
This option allows the device server to be configured so that two or more hosts share access to the same port or ports on one device server. For example, as shown in the illustration, we could configure port 1 on the device server so that Host A and Host B both have access to these two ports.



But why do we call this type of port sharing *asynchronous*? The reason is that only one host can access a port at any given time. For example, if Host B attempts to open a connection with port 1, but Host A has already opened a connection with this port, Host B's request will be denied until Host A releases the COM port. Note that this kind of application is suitable for high-security environments.

synchronous Port Sharing

In some situations, it is required to use more than one host to connect to the same serial device, with the ability to receive data from that device at the same time. Depending on the type of higher-level application software that is used, it is also possible to set up the maximum connections to have more than one connection. This function is useful in some redundant systems.



For example, as illustrated in the figure shown here, a 2-port serial device server shares its ports with 4 different hosts.

Yesterday's solution has a data acquisition server to be responsible for controlling the serial device. Other database servers or remote displays can query the data acquisition server or the database to share the serial data. If the data acquisition server crashes, the whole system fails as well.

Today's solution can use a serial device server to support up to 4 transparent tunnels on the Ethernet to receive the same serial data. All of these 4 hosts can both receive from and send data to the serial device. By using synchronous port sharing function, these 4 hosts can build up a redundant application to switch the controlling ability between different hosts. This function is also helpful for remote displays or monitoring applications.

Socket Mode

The second class of operation mode, referred to as *Socket Mode*, provides a way to directly access serial device servers over TCP/IP networks, but without first installing a driver. To control serial device servers directly over the TCP/IP network layer, knowledge of basic concepts related to TCP/IP networks, such as TCP, UDP, IP, Netmask, and Routing is a must. This is not a major obstacle though, since Internet related products and technologies are quite popular, and most networking engineers are now familiar with TCP/IP concepts.

Sockets are standard APIs (Application Programming Interfaces) used to access network devices over a TCP/IP network. Two Socket API standards are in common use. The original standard was developed for the Unix/ Linux environment, and is generally just referred to as *Sockets*. The other Socket API standard is used with the Windows environment, and is called *WinSock*. Although there are fundamental differences between these two standards, most of the API function calls from the two systems have the same structure, and consequently Socket based network control programs are easily migrated between Unix and Windows environments. In fact, since such programs are portable between almost all system platforms—including Windows, Linux, Sun OS, and even RTOS (Real-Time Operating Systems), such as VxWorks, Windows CE, pSOS—sockets have been widely adopted by most system programmers.

TCP vs. UDP

Before discussing the details of *Socket Mode*, we give a brief description of the two transport protocols, TCP (*Transmission Control Protocol*) and UDP (*User Datagram Protocol*), both of which rely on the IP-layer to send data over a TCP/IP network.

A very poignant way of characterizing TCP is to say that it provides *connection oriented and reliable data transmission.* The *connection oriented* part means that before a host begins to transfer data, it must first establish a connection with the serial device server. This is similar to the everyday task of making a phone call, in which case you must first reach someone on the other end before a conversation can begin.

The second part of the characterization is that TCP provides a mechanism for *reliable data transmission*. This means that each packet is carefully checked before being delivered, and then the device receiving the packet is required to issue an acknowledgement to the sender after receiving a valid packet. This process of issuing and receiving acknowledgements allows

TCP to detect transmission errors, and then request that specific packets be retransmitted as needed. Furthermore, TCP automatically re-assembles packets based on the packets' connect sequence number. Taken together, these characteristics make TCP a naturally reliable data transmission protocol.

In real world applications, TCP is excellent for data-stream transmission, and is a good solution for handling situations for which the quality of the underlying communication medium is not particularly reliable. However, TCP is not an inherently fast responding protocol. For example, you may have to wait a significant amount of time while attempting to establish a connection with another network node. TCP will also attempt to re-established broken connections, but it could take on the order of minutes for TCP to declare that a connection has failed.

The second transport option is UDP, which is ideal for transmitting limited amounts of data but with a fast overall transmission speed. It is for this reason that UDP is often characterized as providing *speedy data transmission*. This comes from the fact that UDP does not require establishing a connection before sending a UDP datagram, and the datagram recipient does not issue an acknowledgement. Furthermore, the overall size of a UDP datagram is typically smaller than the usual TCP packet size, resulting in a much improved response time and data transmission efficiency. The situation is similar to sending someone a letter. All you need to do after addressing the letter is drop it in the nearest mailbox. The most you can hope for is that the letter reaches its destination intact (as it usually does), but the recipient is not required to send an acknowledgement. Although UDP is a naturally speed-oriented protocol, as opposed to reliability-oriented, it is still possible to enhance reliability by building it into your application.

Another UDP protocol characteristic worth mentioning is that in addition to performing point-to-point data transmission, it can also use broadcasting or multicasting technologies to handle point to multi-point transmissions. In real world applications, UDP is excellent for the type of data packet transmissions used when connecting to card readers, control machines (PLCs, CNCs, etc.), and other similar devices. And when your network environment is running smoothly, you can use UDP to accelerate the overall transmission efficiency.

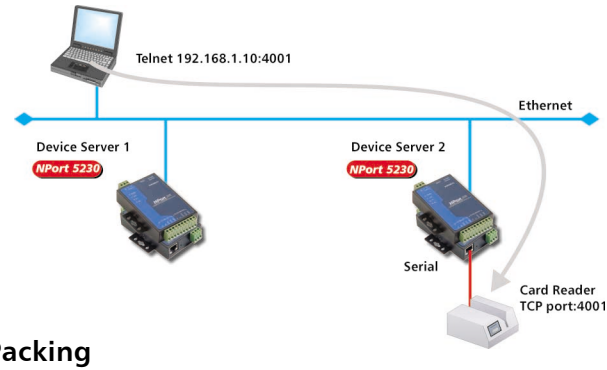
Modern serial device servers provide the capability for communication based on either TCP or UDP. The above introduction should give you the knowledge needed to choose which protocol is best suited for your particular application.

Addressing

You might be interested in how we identify a serial device server, or in particular a serial port belonging to the device server, over a TCP/IP network. TCP and UDP both provide "IP + Port No." addressing to establish contact with specific network devices. With respect to a serial device server, for example, you can use Telnet to access devices attached to its serial ports. The figure shown below illustrates a typical scenario. In this case, 192.168.1.10 is the IP address of the serial device server, and 4001 is the TCP port number that has been associated with one of the device server's ports. Issuing the command

```
telnet 192.168.1.10 : 4001
```

from the host allows the system administrator to easily establish contact with the serial device.



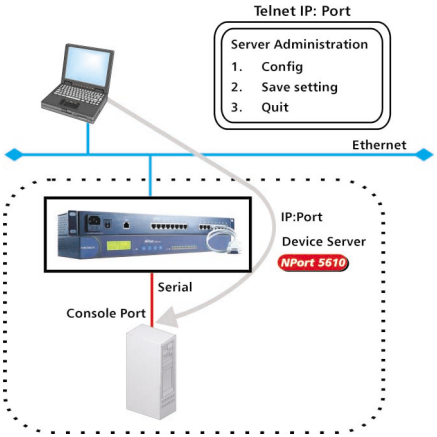
Data Packing

One potential problem that designers of serial device servers must consider is the transmission speed mismatch between the Ethernet and serial sides of the device server. The problem arises when the device server is called upon to transform serial data into a TCP packet or UDP datagram suitable for transmission over the Ethernet, and comes from the fact that the device server's TCP/IP software could easily split a stream of serial data into one or more packets or datagrams. When a packet or datagram containing partial serial data arrives at the application charged with processing the data, the application could very well fail.

The solution to this problem is to pack the stream of serial data into a buffer located inside the device server's memory, and then require data belonging to one serial transmission to be sent together as one packet or datagram. A good data packing function will allow the user to set both a time limit and size of data limit, so that data acquisition is made to match whichever application is being used to process the serial data.

TCP Server

In *TCP Server Mode*, the serial device server acts as a network agent for the serial device. For example, when a serial device (e.g., a file server) which has a serial console port connects to the serial device server under TCP Server mode, the console port becomes a network-accessible point by way of the serial device server. In TCP Server Mode, the serial device server can accept a TCP connection from the control host, and then provide dual direction transmission between control host and serial device.

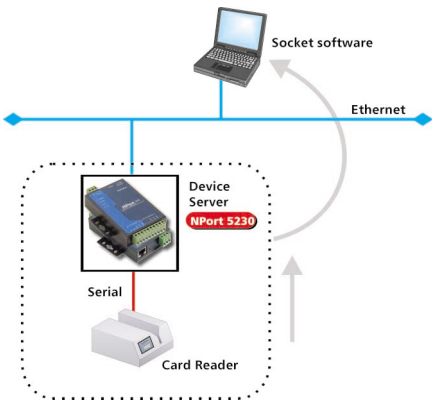


Consider the example illustrated in the above figure, in which a serial device server is connected to the console port of a file server. In this case, the file server could be part of a large *server farm*, or housed in a remote location not easily accessed by the system administrator. By connecting the serial device server's Ethernet port to a LAN, the system administrator gains access to the PC's console management features from a host located on the same LAN, or on the Internet if the LAN is connected to a public network.

TCP Client

The *TCP Client Mode* of operation is designed for serial devices that need to actively establish communication with a server program located on a different network device. In this case, the serial device server running *TCP Client* actively establishes a TCP connection with the server software. After data from the serial device is transferred to the server software, the serial device server automatically disconnects the TCP connection. While the connection is active, the server software is also able to send data to the serial device via the serial device server. You could say that TCP Client achieves the goal of *Connect-on-Demand*, a useful feature for hosts that must deal with so many serial devices that the number exceeds the maximum simultaneous TCP connections allowed.

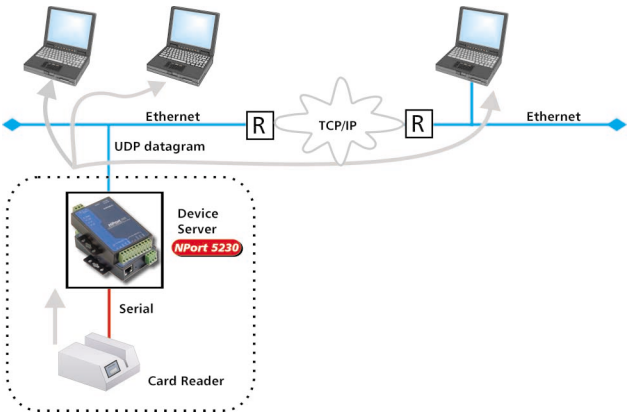
In addition, you should make sure that your serial device is equipped with Data Packing features (discussed earlier) to ensure compatibility with most server software.



As an example, consider the card reader shown in the above figure. Depending on its specific function, a card reader is generally used sporadically over any 24-hour period. Once a card is swiped, the stored information is transformed into the appropriate serial signal, which is then sent to the device server via the device server's serial port. The device server, which would have already been set up to send data to a particular server application on a remote PC or device, forms the serial data into a TCP packet, requests a connection with the remote application, and then sends the packet across the Ethernet to the remote host.

UDP Server/Client

The *UDP Server/Client Mode* of operation is designed for applications that require *speedy data transmission* over UDP protocol layer. By using serial device servers with the UDP Server/Client feature, your serial device can deliver data to multiple destinations at almost the same time. Thanks to the speedy nature of UDP and the Data Packing feature, the serial device server turns traditional serial devices into powerful network-enabled devices. It is suitable for connecting input devices, such as scanners, card readers, fingerprint readers, and optical scanners.

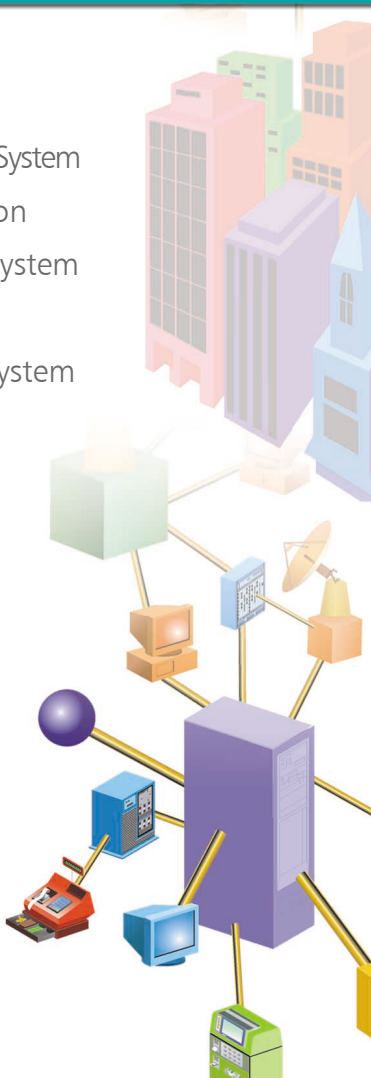


As an example, consider the situation illustrated in the above figure. In this case, the application dictates that card reader data should be sent to more than one host, all of which are connected to the Internet. The way to enforce such a requirement is to configure the device server connected to the card reader for UDP Server/Client mode, and then set up the system so that data are sent to each of the hosts.

Chapter 3.

Case Studies

- Campus Attendance/Entrance Control System
- Medical Equipment Data Acquisition
- Remote Environment Monitoring System
- Rack Server Remote Management
- Power Distribution Management System

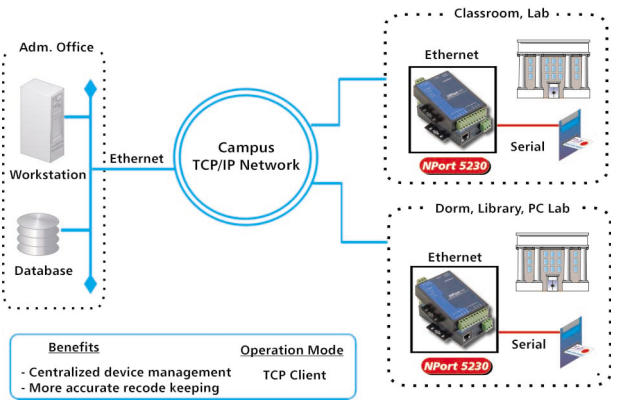


In this chapter, we examine five separate examples that illustrate the implementation of serial device server technology. There are, of course, many other situations to which device server technology can be applied. But even though your own specific application may not appear, as you read through this chapter you should be able to see how to adapt the examples to suit your needs.

Campus Attendance/Entrance Control System

Colleges and universities are prime examples of the type of environment well suited for the installation of serial device servers. This is due to the fact that a school's existing Ethernet network can be used for campus-wide device connectivity. In fact, most campuses consist of an even mixture of dormitories, laboratories, and shared office space, all of which must provide students and faculty with network access. Most areas of campus also require some type of access control to restricted areas, and must provide round-the-clock security. Other applications include setting up centralized control of computers located in computer rooms across campus, providing network connectivity to various types of lab equipment, and giving maintenance personnel remote management capability of the school's heating and air conditioning systems.

The benefits of using device server technology in a campus environment include centralized management and control of campus serial devices, more accurate record keeping, and improved safety for faculty and students.

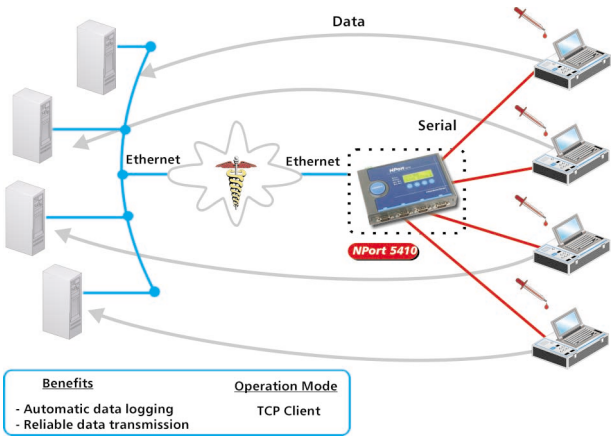


As a concrete example, let's consider the task of maintaining accurate attendance and facility-use records. In this scenario, each classroom, laboratory, dorm entrance, and library is outfitted with a card reader that connects to the campus network through a serial device server. This particular application would be configured as the TCP Client Mode of operation, allowing the card readers to actively connect to networked hosts. Magnetic stripped student ID cards have been in use for many years, so that students are already geared up to participate. A student's entrance into a classroom or facility is recorded

automatically when their ID card is inserted into the card reader, giving instructors and administrative personnel valuable information concerning classroom attendance, as well as the rate at which students use various on-campus facilities.

Medical Equipment Data Acquisition

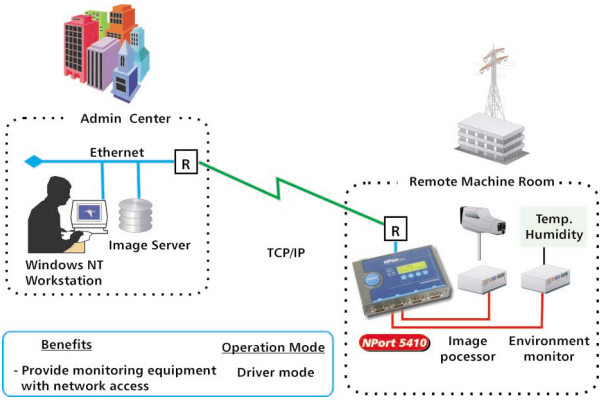
Today's hospitals, medical screening clinics, and pharmaceutical labs are full of electronic monitoring equipment, most of which can be connected to and then monitored via the COM ports of a PC. A prime example is the maternity ward that monitors the contractions of an expectant mother and the heartbeat of the fetus. This involves two monitors which when connected to the same PC allows the heartbeats to be viewed in real-time at a nursing station, giving nurses ample time to respond when it comes time to deliver the baby. The problem is how to handle several patients at the same time. One possibility, of course, is to use a different computer for each patient, but this is not a very efficient use of resources. A more reasonable and cost-effective solution is to make use of the hospital's built-in Ethernet LAN and connect each monitor to a serial device server, allowing data from all patients to be conveniently processed by the same computer program.



Another example is the modern medical screening facility that uses limited staff, but offers personalized examinations to large numbers of people. In this case, each staff member, whether a physician, nurse, or medical technician, operates one station, and each station contains either a PC or some type of testing equipment to record data. Testing equipment, such as the self-contained blood analysis machines shown in the above figure, is connected to a serial device server, which is then plugged into the facility's Ethernet. Results of a patient's blood analysis is sent over an Ethernet LAN to a central computer where the data is recorded to await expert diagnosis by a qualified physician.

Remote Environment Monitoring System

An important problem of recent origin involves the remote monitoring of arrays of computers and other electronic equipment sensitive to environmental extremes. This includes enterprises with a computer room consisting of small numbers of servers and routers, as well as huge operations, such as Remote Data Centers and Server Farms that manage hundreds or thousands of computers. Whatever the situation, this type of sensitive electronic equipment must be placed in a location with temperature and humidity kept within strict bounds. The problem faced by administrators is how to remotely monitor such environments so that faulty air conditioning equipment can be repaired as needed, and security can be notified when intruders enter restricted areas.



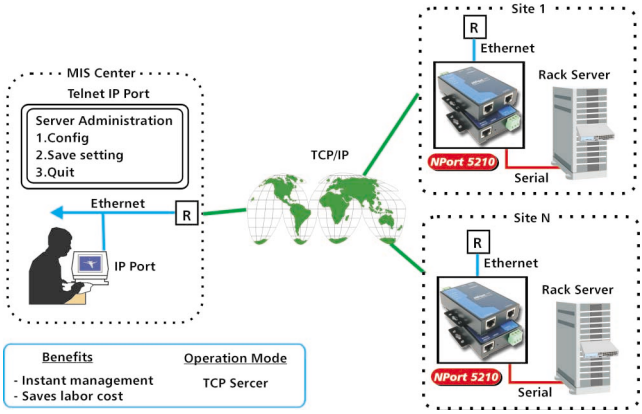
As a concrete example, consider the Remote Machine Room illustrated in the above figure. A typical environment monitoring system for this type of operation consists of security cameras, and sensors that check temperature and humidity. A serial device server can be used to provide immediate Internet access for these devices, giving the administrator a straightforward and economical solution for setting up an environment monitoring system. In this case, the serial device server can be configured in Driver Mode to provide LAN-to-LAN connectivity, making it possible to view images from the security cameras and check data from the sensors, all in real-time, and from any computer connected to the Internet.

We should also point out that a serial device server configured in Driver Mode provides the administrator with a very straightforward way of setting up and managing this type of monitoring system. In this case, the serial devices consist of cameras and sensors, each of which should come with a manufacturer supplied driver that allows the user to access and control the device via a

computer's serial port. What Driver Mode does is allow the administrator to connect serial device to serial device server, connect serial device server to Ethernet, and then access and control the serial device from any computer connected to the network. No change is required to the serial device, or the manner in which it is accessed from the host computer.

Rack Server Remote Management

The globalization of the retail industry brings with it the challenge of devising a reasonable global management strategy that is both practical and scalable. In this case, "practical" means that the solution does not require the implementation of new and expensive software packages, and "scalable" means that as new stores are added to the chain, the only noticeable change is the addition of related data to the management system. You can imagine the following situation. A retail store begins with one location in the United States, expands to another site in the same state, finds it has a great organization, and then continues to expand into other states. As it expands, the business develops a centralized management system that uses modems and regular telephone lines, or for some connections, leased lines. However, expanding into other continents involves much higher telephone and leased line costs.

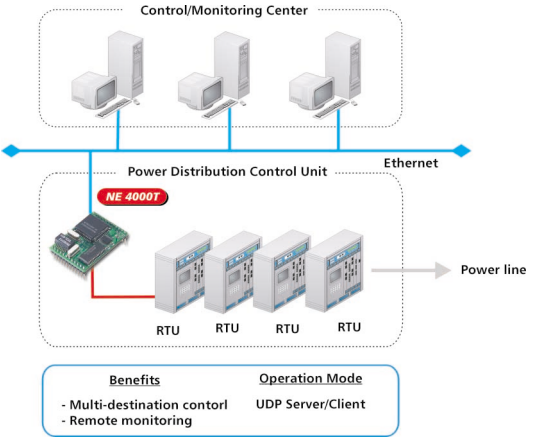


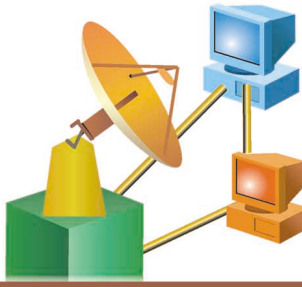
The intelligent way of handling this problem is to make use of the Internet, since it reaches onto every continent of the world and associated access charges are "local," in the sense that each Internet access point requires a minimal fee, but long distance telephone and leased line charges can be avoided. As illustrated in the above figure, serial device servers can also play a central role in this global management solution. Each location makes use of one or more rack servers which are managed via a serial console port. By

connecting the rack server's serial port to a serial device server, which is then connected to the Internet, a manager stationed in the United States has immediate access to any of the chain's rack servers, regardless of location. This greatly reduces overall management costs and keeps losses due to downtime at a minimum.

Power Distribution Management System

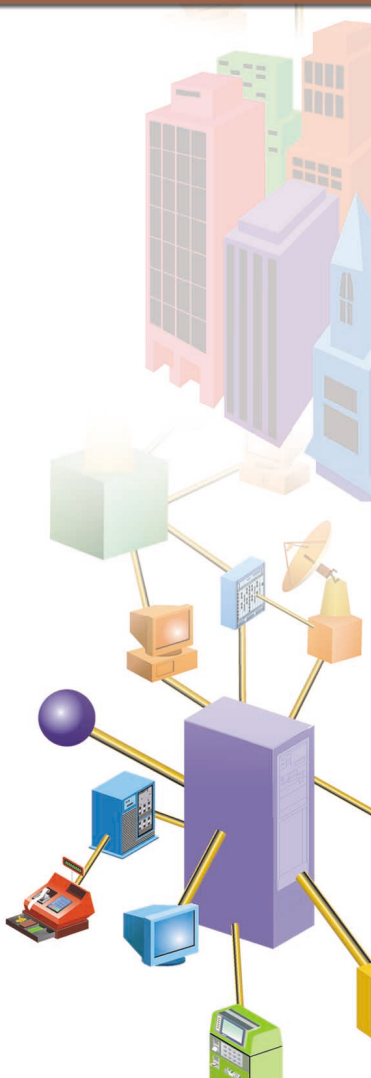
An important issue in the modern power and energy industry is the monitoring and control of power distribution grids, resulting in maximum power utilization. To ensure that power is transferred to the proper areas, an RTU (Remote Terminal Unit) is one of the most important pieces of equipment used to monitor and control power distribution. In the past, communication with RTUs was handled using RS-485 multi-drop networks, but the current trend is towards adopting TCP/IP Ethernet networks. This type of RTU system allows engineers to carry out two important tasks. One task is the remote detection of possible power drains. Considering that power companies are generally responsible for the management of hundreds if not thousands of miles of power lines, it is easy to see that the ability to quickly detect power drains can effectively balance power generation and usage, and thus help conserve energy. Another important task handled by the RTU network management system is the maintenance of redundant operational sites containing system management hardware and software. This can be achieved using a serial device server with the UDP Server/Client function, and thanks to the ability for multi-destination data transmission, this type of setup naturally forms an RTU redundant control system.





Chapter 4.

Finding the Right Solution



After reading this far, you should have a good idea about the features and functions of serial device servers. The problem now is how to find the serial device server product most appropriate for your application. Keep in mind that a good serial device server supplier should provide you with a product that has most of the previously mentioned features, such as a Windows driver, and TCP/UDP Client/Server. In addition, take into account agency approvals, such as CE, FCC, UL, CUL, TUV, since they are a good indicator of product quality. Excellent service is also an important requirement. People using serial device servers often need assistance with network planning, installation, and programming issues, all services which should be offered by good suppliers.

Reference from Moxa

Moxa is currently one of the top 3 serial device server supplier in the industry worldwide. We hope that the following function / product comparison table provided for your reference will help you in choosing the right products. The table is based on the serial device server functions introduced earlier. You may also visit our website at www.moxa.com for more product information.

Product Selection Guide

Items \ Models	NPort DE-311
No. of serial ports	1
Serial interface	3-in-1 RS-232 / RS-422 / RS-485
Connector	DB9
Network interface	10/100M
IP Configuration	Auto IP configuration by DHCP, BootP & Manual
Serial console	Yes
Telnet console	Yes
LCD panel & buttons	N/A
Operation modes	Real COM, TCP Server, TCP Client, UDP,
COM/TTY drivers	Windows 95/98/ME/2000/XP/2003 Real COM driver
Software utility	NPort Management Suite for Winodws
Security	Password for console (MD5 encrpted)
Protocols	ICMP, IP, TCP, UDP, DCHP, BootP, Telnet
Housing	DIN-Rail (35 mm) mount, wall mount, table top
Power input	12 - 30 VDC

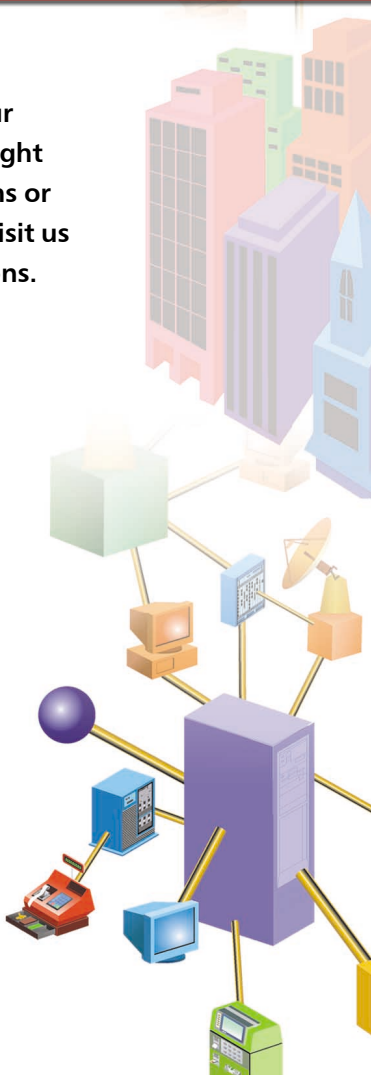
Models / Items	NPort 5000 Series		
	NPort 5210	NPort 5230	NPort 5232 NPort 5232I
No. of serial ports	2	2	2
Serial interface	RS-232	RS-232 x 1 RS-422/485 x 1	RS-422/485
Connector	RJ45 (8-pin)	TB	TB
Network interface	10/100M		
IP Configuration	Auto IP configuration by DHCP, BootP & Manual IP configuration		
Serial console	Serial port1		N/A
Telnet console	Yes		
LCD panel & buttons	N/A		
Operation modes	TCP Server, TCP Client, UDP, Real COM * IP Serial Lib can only be used in TCP Server Mode		
COM/TTY drivers	Windows 95/98/ME/2000/XP/2003 Real COM driver Linux Real TTY driver		
Software utility	NPort Administrator for Windows		
Security	Password for console (MD5 encrpted)		
Protocols	ICMP, IP, TCP, UDP, DCHP, BootP, Telnet, DNS, SNMP, HTTP, SMTP, SNTP		
Housing	DIN-Rail (35 mm) mount, wall mount, table top		
Power input	12 - 30 VDC		

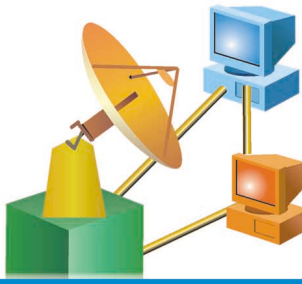
Product Selection Guide

Items \ Models	NPort 5000 Series			
	NPort 5410	NPort 5430 NPort 5430I	NPort 5610-8 NPort 5610-16	NPort 5630-8 NPort 5630-16
No. of serial ports	4	4	8/16	8/16
Serial interface	RS-232	RS-422/485	RS-232	RS-422/485
Connector	DB9M	TB	RJ45 (8-pin)	
Network interface	10/100M			
IP Configuration	Auto IP configuration by DHCP, BootP & Manual IP configuration			
Serial console	Serial port 1	N/A	Serial port 1	N/A
Telnet console	Yes			
LCD panel & buttons	Yes		Yes	
Operation modes	TCP Server, TCP Client, UDP, Real COM * IP Serial Lib can only be used in TCP Server Mode			
COM/TTY drivers	Windows 95/98/ME/2000/XP/2003 Real COM driver Linux Real TTY driver			
Software utility	NPort Administrator for Windows			
Security	Password for console (MD5 encrpted)			
Protocols	ICMP, IP, TCP, UDP, DCHP, BootP, Telnet, DNS, SNMP, HTTP, SMTP, SNTP			
Housing	DIN-Rail (35 mm) mount, wall mount, table top		19" Rack mountable	
Power input	12 - 48 VDC		100-240 VAC or 12-48 VDC	

Questions

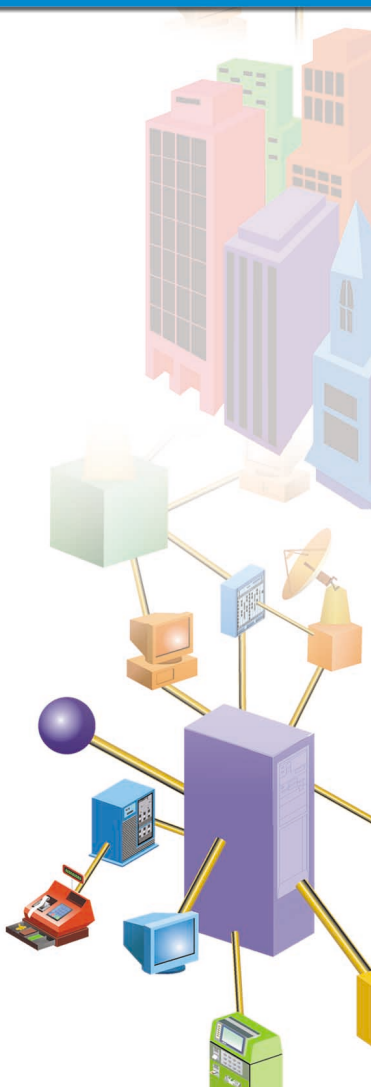
We would be more than happy to hear your opinions and answer any questions you might have about this book. Please send questions or comments to Support@moxa.com.tw, or visit us at www.moxa.com to leave your suggestions.





Glossary of Terms

In this section, we give brief definitions of several important networking terms and acronyms.



-A-

API

In this book, an *API* (Application Programming Interface) is a series of functions that programmers use to communicate with network *protocol suites*. The functions include establishing a connection, reading or writing data, and disconnection.

AT commands

AT commands are basic text-based commands, issued from a computer program, that are used to control the setup and operation of a modem.

-C-

client program

A *client program* is a type of computer program that actively requests service from a *server program*, which often resides on another computer.

CNC

A *CNC* (Computer Numerical Controller) is a control device used as part of a closed loop control system, such as a milling machine. The CNC accepts sensor inputs from other components (motors, position detectors, distance meters, timers, etc.), and then runs a computer program that produces appropriate controller outputs to motors, drillers, etc.

COM port

A *COM port* is a serial communications port (RS-232 interface) on a Windows-based PC. Most PCs come with two built-in COM ports, although the number of ports can be extended into the hundreds by installing either multiport serial boards, or network-based *serial device servers*.

cross-over cable

A *cross-over cable* is used to connect two ports of like nature. A serial cross-over cable is used to connect a DCE device to a DCE device (or a DTE device to a DTE device). An Ethernet cross-over cable can be used to directly connect NIC cards installed in two different computers without using an Ethernet *hub*.

-D-

data-stream transmission

Data-stream transmission refers to a series of data transmitted continuously without interruption.

-E-

Ethernet

Ethernet is a local-area network architecture developed by Xerox, DEC, and Intel in 1976. It operates using a shared bus or star topology, and supports data transfer rates of 10 Mbps, 100 Mbps, and even 1000 Mbps formats.

Ethernet frame

An *Ethernet frame* is a packet of data bits sent from one device to another over an Ethernet network.

-F-

fixed tty driver

A *fixed tty driver* is a driver program used by Unix/Linux systems to control external devices connected to the serial device server's serial port(s). Fixed tty drivers are characterized by the fact that a user's software is able to transmit data via a pseudo tty port based on pre-defined communication parameters, making them suitable for pure data transmission. However, the user's software cannot be used to change a port's communication parameters. The limitation is that the user's software is not able to control DTR, RTS, and DCD signals during operation. Another kind of tty driver, a *real tty driver* supports full serial port control functions.

-H-

host

A *host* is a computer, such as a PC or Linux server, that is connected to a network. Each host is assigned its own unique IP address.

HTML

HTML (HyperText Markup Language) is the computer language used to produce text files that contain typesetting commands which allow the document to be viewed with a web browser (such as Netscape® or Internet Explorer®). HTML defines the structure of tags and attributes used to create Web text documents.

HTTP

HTTP (HyperText Transfer Protocol) defines how messages are formatted and transmitted, and what actions the Web server should take.

-H-

hub

A hub is an Ethernet cable concentrator that allows data signal exchange between different Ethernet wires.

- I -

IP address

An *IP address* is a 32-bit identification number assigned to networking devices connected to a TCP/IP network such as the Internet. IP addresses are written, for example, in the form 192.168.206.10 (see also *public IP address* and *private IP address*).

-L-

LAN

A *LAN* (Local Area Network) is a computer network characterized by the fact that a message sent from one LAN computer to another does not pass through a router. Note that most LANs are located within a relatively small geographical area, such as the a building or campus.

-M-

machine room

A *machine room* houses computers and network equipment, and is usually climate controlled to protect expensive electronic equipment from extreme temperatures and dusty conditions.

multidrop network

A *multidrop network*, which is a type of computer network that allows several devices to communicate over the same pair of wires, is usually associated with the *RS-485* serial interface. In a multidrop network, one node can broadcast data to all other nodes at the same time.

-N-

netmask

A *netmask* is a 32-bit number that is used to determine the network scope of a computer's IP address. The most commonly used netmasks are 255.0.0.0 for Class A networks, 255.255.0.0 for Class B networks, and 255.255.255.0 for Class C networks. For example, a computer with IP address 192.168.254.15 and netmask 255.255.255.0 belongs to a Class C network, in which all computers on the network have IP addresses of the form 192.168.254.*xxx*.

-P-

PLC

A *PLC* (Programmable Logic Controller) is a device used in Industrial Automation as a key component of automatic control systems. A PLC accepts multiple sensor inputs, uses an internal (user-programmable) microcomputer program to process the control signal, and then sends out appropriate controller signals to activate machines, such as motors.

protocol suite

A *protocol suite* is a collection of protocols used for device-to-device communication over a network. The *TCP/IP* protocol suite, for example, is composed of protocols such as TCP, IP, UDP, ICMP, ARP, etc.

-R-

raw data format

Raw data format refers to information that has not been translated or organized in terms of software.

Real COM driver

A *Real COM driver* is a driver program under Windows that completely emulates the behavior of a local COM port. The driver is used for ports on a *serial device server* that connects to the *host* via a TCP/IP network. The serial port on the remote serial device server will exhibit the same behavior as a local COM port.

real tty driver

A *real tty driver* is a driver program used by Unix/Linux systems to control external devices connected to the serial device server's serial port(s). *Real* tty drivers are characterized by the fact that a user's software is able to transmit data by standard tty interfaces. It allows you to perform full functions of the tty interface, including data transmission, and line signal control (such as RTS, CTS, DTR, DSR, DCD, and Break signal). Another, simpler form of tty driver is the *fixed tty driver*, used for pure data transmission.

router

A *router* is a network device that is used to connect two or more LANs or WANs via a leased line, ADSL, or other long distance communications interface. Routers are able to determine data packet destinations.

-R-

RS-232

RS-232 is a standard interface for connecting serial devices. Many modems, display screens, and printers are designed to operate via an RS-232 port.

RS-422

The *RS-422* standard is based on the *RS-232* standard, but is designed to support higher data rates and longer data transmission distances (up to 1.2 km). RS-422 also has greater immunity to electrical interference.

RS-485

RS-485 has many of the same characteristics as RS-422, especially in terms of data transmission distance and immunity to electrical interference. In addition, it supports 2-wire multi-drop operation and allows 32 RS-485 nodes over a 2-wire bus. These characteristics have resulted in its wide use in industrial control networks.

-S-

serial communications

Serial communications refers to the transmission of data bit-by-bit.

serial device server

A *serial device server* is a standalone device that has at least one Ethernet port and one or more serial ports. Serial device servers are equipped with an embedded network operating system and allow computers to access serial devices over a network.

serial tunnel

A *serial tunnel* is used to encapsulate serial data in a TCP/IP packet or datagram, and then send it over a network. Serial tunnels allow a host computer to access data from virtually any type of serial device, over a network connection.

server farm

A *server farm* is an enterprise that houses hundreds or thousands of server computers which are leased to other companies. Businesses that run server farms often provide a full range of support, from simply leasing space for a computer to offering professional web site design and maintenance services.

server program

A *server program* is a computer program that waits passively for requests for service from client programs.

-S-

socket programming

Socket programming refers to scripts that execute functions which read and write data to and from a socket.

straight-through cable

A *straight-through cable* is used to connect two ports of dislike nature. A serial straight-through cable is used to connect a DCE device to a DTE device. An Ethernet straight-through cable is used to connect a computer's NIC to a hub or switch.

switch

In terms of network communication, a *switch* is an intelligent hub that automatically identifies the direction of traffic, and isolates the traffic on different collision domains. Switches can ensure maximum bandwidth usage for each Ethernet connection.

-T-

TCP/IP

The *TCP/IP* protocol suite refers to the family of network protocols used by most Ethernet networks, and by the Internet, to connect hosts. TCP/IP, in which TCP stands for Transmission Control Protocol and IP stands for Internet Protocol, is a standard for transmitting data over networks.

TCP header

A *TCP header* consists of identification and control information that is affixed to the beginning of a TCP packet as the packet passes through the TCP/IP stack.

TCP packet

A *TCP packet* consists of data and headers sent by TCP over a TCP/IP network (compare with *UDP datagram*).

TCP socket

A *TCP socket* is a program that can send and receive TCP/IP messages by opening a socket, and then reading and writing data to and from the socket.

-T-

Telnet

Telnet is a widely used protocol that establishes a network connection with a networked device. Many standard software utilities are based on the Telnet protocol, such as Telnet (system utility name) under Windows and Linux. Many network devices, such as serial device servers and routers have a built-in Telnet console that allows users to configure the device by the Telnet utility.

-U-

UDP datagram

A *UDP datagram* consists of data sent by UDP over a TCP/IP network (compare with *TCP packet*).

-W-

WAN

A *WAN* (Wide Area Network) consists of two or more LANs connected by switches and/or routers.

WinSock

WinSock is the standard Windows API utility that uses *TCP/IP* protocol to connect other network devices.

www.moxa.com

MOXA

Moxa Technologies Co., Ltd.

F4, No. 135, Lane 235, Pao-Chiao Rd.
Shing-Tien City, Taipei, Taiwan, R.O.C
Tel:+886-2-8919-1230 Fax:+886-2-8919-1231