The following excerpt is from the book:

Serial Port Complete
COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems
Second Edition

Jan Axelson

Copyright 1998, 1999, 2000, and 2007 by Janet L. Axelson

All rights reserved.

Published by Lakeview Research LLC

ISBN 978-1931448-06-2

For more about the book and serial-port design and programming, visit:

www.LVR.com

1

Options and Choices

A serial port is a computer interface that transmits data one bit at a time. In common use, the term “serial port” refers to ports that use a particular asynchronous protocol. These ports include the RS-232 ports on PCs and many serial ports in embedded systems. Most serial ports are bidirectional: they can both send and receive data. Transmitting one bit at a time might seem inefficient but has advantages, including the ability to use inexpensive cables and small connectors.

In PCs, applications access most serial ports as COM ports. Applications that use Microsoft’s .NET Framework class library can use the SerialPort class to access COM ports. Some USB devices function as virtual COM ports, which applications can access in the same way as physical serial ports. Some Ethernet and Wi-Fi devices function as serial servers that enable applications to access serial ports over a network.

Microcontrollers in embedded systems can use serial ports to communicate with other embedded systems and PCs. Language compilers for microcontrollers often provide libraries with functions that simplify serial-port programming.

When to use a Serial Port

Device developers have many options for computer interfaces. Table 1-1 compares popular wired interfaces.

Serial ports are ideal for many communications between embedded systems or between embedded systems and PCs. Serial ports can also be a good choice when you need very long cables or a basic network among PCs, embedded systems, or a combination. Some systems include a serial port that is hidden from users but available to technicians for debugging and diagnostics.

Advantages

These are some advantages of asynchronous serial ports and COM-port programming:

- Serial ports can exchange just about any type of information. Applications suited for serial ports often involve reading sensors, switches, or other inputs or controlling motors, relays, displays, or other outputs.
- The hardware is inexpensive and readily available. PCs that don't have built-in serial ports can use USB/serial converters. Just about every microcontroller family includes variants with built-in serial ports.
- Other than the Start, Stop, and optional parity bits added to each transmitted byte, serial interfaces assume nothing about the content of the data being transmitted. In contrast, USB and Ethernet use sophisticated protocols that define the format of transmitted data. Hardware or firmware must implement these protocols, adding complexity that some applications don't need.
- Cables can be very long. An RS-232 interface can use cables of 130 ft or more. An RS-485 cable can be over 4000 ft. In contrast, the maximum distance between a USB device and its host is 16 ft, or 98 ft with five hubs. Ethernet cables have a maximum length of 328 ft.
- The cables are inexpensive. Many links can use unshielded cables with 3–9 wires.
- For devices that connect to PCs, Windows and other operating systems provide drivers for accessing COM ports. Programming languages provide classes, libraries, or other tools for COM-port communications.

Table 1-1: Comparison of popular computer interfaces. Where a standard doesn't specify a maximum, the table shows a typical maximum.

Interface	Format	Number of Devices (maximum)	Distance (maximum, ft)	Speed (maximum, bps)	Typical Use
RS-232 (TIA-232)	asynchronous serial	2	50-100	20k (faster with some hardware)	Modem, basic communications
RS-485 (TIA-485)	asynchronous serial	32 unit loads (up to 256 devices with some hardware)	4000	10M	Data acquisition and control systems
Ethernet	serial	1024	1600	10G	PC network communications
IEEE-1394b (FireWire 800)	serial	64	300	3.2G	Video, mass storage
IEEE-488 (GPIB)	parallel	15	60	8M	Instrumentation
I^{2}C	synchronous serial	40	18	3.4M	Microcontroller communications
Microwire	synchronous serial	8	10	2M	Microcontroller communications
MIDI	serial current loop	2 (more with flow-through mode)	50	31.5k	Music, show control
Parallel Printer Port	parallel	2 (8 with daisy-chain support)	10–30	8M	Printer
SPI	synchronous serial	8	10	2.1M	Microcontroller communications
USB	asynchronous serial	127	16 (up to 98 ft with 5 hubs)	1.5M, 12M, 480M	PC peripherals

- A USB device accessed as a COM port doesn't have to have an asynchronous serial interface. The device can have a parallel or other interface as needed to suit the application.
- Wireless technologies enable transmitting serial data without cables.

Limits

No single interface is ideal for every purpose. Limits to asynchronous serial interfaces include these:

- The computers at each end must convert between the transmitted serial data and the CPU's parallel data bus. The conversion is usually handled automatically by hardware, however.
- The specified maximum bit rate for RS-232 is 20 kbps. But many interface chips can exceed this rate, and RS-485 supports speeds of up to 10 Mbps. Communications between a PC and a USB Virtual COM ports aren't limited by RS-232's maximum bit rate.
- Windows doesn't promise real-time performance for serial communications. Sending or receiving data may need to wait as the operating system attends to other tasks. But the delays are normally short and are common to other interfaces on Windows systems. Embedded systems typically can control the scheduling of serial communications more precisely.

System Components

Communicating via serial ports requires three things: computers with serial ports, a cable or wireless interface that provides a physical link between the ports, and programming to manage the communications.

The Computers

Just about any computer can use serial-port communications, including inexpensive microcontrollers and PCs that don't have built-in serial ports.

Examples of Serial Ports

Devices with asynchronous serial ports typically contain a hardware component called a Universal Asynchronous Transmitter/Receiver (UART). The UART converts between parallel and serial data and handles other low-level details of serial communications.

These are some examples of ports controlled by UARTs:

- Microcontroller serial ports. Many microcontrollers contain one or more UARTs for serial-port communications. When a hardware UART isn't available, microcontroller firmware can emulate a UART's functions, typically with assistance from an on-chip timer.
- External UART chips that interface to microcontrollers or other CPUs.
- The RS-232 serial ports that were standard on PCs and other devices before USB became common. Each of these ports contains a UART that interfaces to the system's CPU. Any PC with a free expansion slot can add this type of port on an expansion card.
- RS-232 ports on PC Cards (also called PCMCIA cards). Any PC with a free PC-Card slot can use these.
- Serial ports that connect to PCs via USB converter modules.
- Other serial ports used in long-distance and networking applications, often in industrial-control applications. These interfaces include RS-485, RS-422, and RS-423. Expansion cards, PC Cards, and USB converters with these interfaces are available.
- Ports on serial-server modules that connect to Ethernet or Wi-Fi networks.

On PCs, ports that applications can access as COM ports include these:

- RS-232 ports on older motherboards or on expansion cards.
- Ports that connect to a PC via a USB converter that uses a driver that assigns a COM port to the device. Converters are available as modules and as chips for incorporating into circuits. A converter can convert between USB and RS-232, RS-485, TTL serial, or even a parallel interface.
- Internal modems that interface to phone lines.
- Serial ports on network serial-server modules.

For USB virtual COM-port devices, Windows includes a driver for USB's communication devices class (CDC). For improved performance, some converters use vendor-specific drivers in place of the provided Windows drivers.

Computer Types

Computers that communicate via serial ports don't have to be all the same type. Tiny microcontrollers can talk to the latest PCs as long as both ends of the link use compatible interfaces and protocols. The PC examples in this book are for the family of computers that has evolved from the IBM PC, including desktop and notebook PCs. Other computer types also have serial ports that are built in or available via converters or expansion cards.

An embedded system is a computer-controlled device dedicated to performing a single task or a set of related tasks. Embedded systems are typically built into, or embedded in, the devices they control. For example, a modem is an embedded system that handles tasks of data communications over the phone system. Some embedded systems are one-of-a-kind or small-quantity projects. Many involve monitoring or control tasks.

Embedded systems often use microcontrollers, which contain a CPU and I/O hardware such as UARTs. Microcontroller chips can be classified by data-bus width: 8-bit chips have an 8-bit data path and are popular in monitoring and control applications. Chips with 4-, 16-, and 32-, and 64-bit data buses are also available. Different chips have different combinations of features and abilities, including asynchronous and synchronous serial ports, USB controllers, type and amount of memory for storing programs and data, and support for power-saving modes.

The Physical Link

The physical link between computers consists of the wires or other medium that carries information from one computer to another and the connectors and other components that interface the medium to the computers.

RS-232 links can use just about any cable type and require one line per signal plus a common ground line. RS-485 networks typically use twisted-pair cables with a pair for each differential signal. Other options for serial communications include fiber-optic cable, which encodes data as the presence or absence of light, and wireless technologies, which enable sending data as electromagnetic (radio) or infrared signals through the air.

Computers connected by wires must have a common ground reference, typically provided by a ground wire in the cable.

Programming

A computer must perform the following tasks in serial communications:

- Detect and process received data.
- Provide and send data as needed.
- Carry out any other tasks the computer is responsible for.

If the connection is to a serial network, each computer must ignore communications intended for other computers in the network and comply with network protocols for addressing transmitted data to the appropriate computer(s).

Program code carries out these tasks, often with help from hardware.

Languages

The programming for a serial interface can use any language, and the language doesn't have to be the same on every computer. The only requirement is that all of the computers must agree on a format. Microcontroller programs might access UART registers directly or use library functions or other higher-level methods to set communications parameters and exchange data. PC applications typically use higher-level functions to access ports.

Protocols

A protocol is a set of rules that defines how computers manage communications. Serial communications must implement a low-level communication protocol and may also implement a higher-level message protocol.

Communication Protocols

A communication protocol defines how the bits travel, including when a computer can transmit, the bit rate, and in what order the bits transmit. The UART typically handles the details of sending individual bits and storing received bits on the serial port.

Two computers that want to exchange data must agree on whether both ends can transmit at once or whether the computers need to take turns. Most wired links between two computers are full duplex: both computers can transmit at the same time. Many wireless links are half duplex: the computers must take turns. A simplex link is one way only. A network with three or more computers sharing a data path must use a protocol that defines when a computer can transmit.

A communication protocol can include the use of status and control lines. These lines can indicate when a transmitter has data to send or when a receiver is able to accept new data. The process of exchanging this information is called flow control. Hardware flow control uses dedicated lines for the signals. Devices can also use software flow control to provide the same information by sending defined codes, typically in the same path used for data.

Additional status and control lines can provide other information such as the presence of a carrier frequency or a ring signal on a phone line. In serial networks where only one transmitter can be enabled at a time, a transmit-enable line at each computer can enable and disable the transmitters as needed.

Message Protocols

Serial communications often exchange messages that consist of blocks of data with defined formats. A message protocol can specify what type of data a message contains and how information is structured within the message.

The computers in a network need a way to detect which computer is the intended receiver of transmitted data. Networks typically assign an address to each computer and include the receiver's address in each message. For example, a very basic message might consist of two bytes: one byte to identify the receiver and one byte containing data.

To enable a receiving computer to detect the start and end of a message, a message can include codes to indicate these events or a header that stores the message length. A message can also include one or more bytes that the receiving computer uses in error checking.

Applications

One way to think about serial applications is by the primary direction of data flow. In a link between two computers, one computer might gather data from or send commands to the other computer. Or two computers may each be responsible for various monitoring and control functions, sharing information with each other.

In some systems all computers send and receive more or less equally. In others, most of the data flows to or from a central computer. For example, most of the activity in a network might involve one computer that collects data from computers in remote locations.

Example Systems

An everyday example of a system that collects data is a weather-watching network. A desktop PC might serve as a primary computer that controls the activities of one or more secondary computers, which can be embedded systems or PCs. The primary computer sends commands to the secondary computers to tell them how often to collect data, what data to send, and when to send. The data collected might include temperature, air pressure, rainfall, and so on. At intervals, each secondary computer sends its collected data to the primary computer, which stores the data and makes it available for viewing and processing. This basic setup is adaptable to many other types of data-gathering systems.

Other systems are mainly concerned with controlling external devices, rather than gathering data from them. A store-window display might include a set of small robots, each with switches and signals that control motors, lights, and other mechanical or electrical devices. Each robot is an embedded system, and a primary computer controls the show by sending commands to the robots. The robots can also return information about their current states, but the main job of this type of system is to control devices, rather than to collect information from them.

An example of a system involved with both monitoring and controlling is a home-control system that monitors temperature, humidity, motion, switch states, and other conditions throughout a house. Other circuits control the home's heating, cooling, lighting, audio and video systems, and alarms. When the data (or a lack of data) indicates a problem, the system generates an alarm.

Managing Communications

In each of the examples above, one computer typically acts as a primary computer that controls a series of secondary computers. A secondary computer transmits only after the primary computer contacts it and gives it permission.

Some networks have no primary computer. Instead, each computer has equal status with the others, and each can request actions from the others. For example, each computer might transmit in a defined sequence. Or on receiving a message, a computer might have permission to select any other computer to transmit to.

Special-purpose Modules

Many common peripheral functions are available as modules with serial interfaces. These modules make it easy to add a function to a design. For example, LCD modules with serial interfaces are available from Scott Edwards Electronics (*www.seetron.com*). The USBwiz from GHI Electronics (*www.ghielectronics.com*) contains a USB host controller and makes it possible to access USB devices via an asynchronous serial port. Motor controllers with serial interfaces are also available from a variety of sources.

These are just a few examples. This book will guide you in choosing components and writing programs for whatever serial-port application you have in mind.