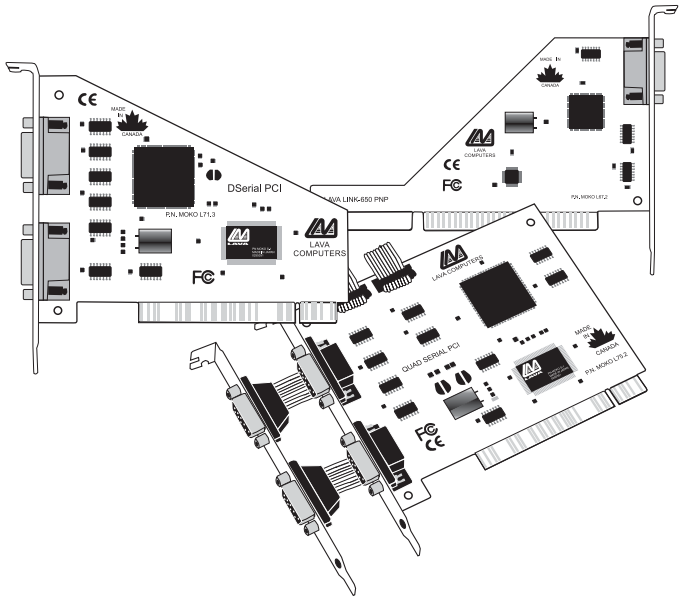
RS-232: Serial Ports



www.lavalink.com

Table of Contents

List of Figures

List of Tables

EIA/TIA-232-F: Serial Ports

Lava Computer MFG Inc.

This white paper defines and discusses serial port technologies in the context of Lava's serial product offerings.

Introduction

Serial communications taken as a whole is a large topic. To keep this white paper shorter than a book, it will focus on asynchronous RS-232 serial port communications, primarily in the PC world, and will emphasize the connectors and UARTs most frequently seen on today's PCs. After defining a serial port and RS-232, this paper will move from the heart of a serial port, the UART (Universal Asynchronous Receiver-Transmitter), outwards through the port's connectors and cables, and finish with operating systems and an overview of Lava's serial products.

What is a serial port?

A serial port is a connector and related electronics that uses one of a number of serial input/output (I/O) standards to interface data communications equipment (DCE) and data terminal equipment (DTE). The serial port connector is usually either a 9-pin (DB-9) or a 25-pin (DB-25) connector, but other types also exist (RJ-11, RJ-12, 8-pin DIN, RJ-45 or DB-37, for example). For examples of DCE and DTE cabling and connectors, see "Connectors and cables" on page 14.

The serial port's signalling and data handling are managed by a Universal Asynchronous Receiver-Transmitter (or UART). A UART—usually designed as a chip—produces an electrical signal that follows an accepted serial I/O standard.

Among PCs, the usual standard for serial I/O is called RS-232, but other similar serial standards, such as RS-422 and RS-485, are also fairly common. Universal Serial Bus

(USB) and IEEE 1394 (FireWire®) are serial communications as well, but are sufficiently different in nature that they will not be discussed here.

What is RS-232?

RS-232 is a species of serial connection described in a specification written by the Electronic Industries Association (EIA) which, in conjunction with the Telecommunications Industry Association, defines the standards for traditional serial data transfer. Formally, the RS-232 standard is called EIA/TIA–232–F, reflecting the initials of the organizations that administer it.

The RS-232 serial port specification defines the types of RS-232 communications equipment as well as the signalling, electrical, and mechanical characteristics of RS-232 serial ports.

DCE AND DTE

The RS-232 specification defines two types of communication equipment: *Data Terminal Equipment*, or DTE; and *Data Circuit-Terminating Equipment*, or DCE. The two differ in pinout assignments—for practical purposes a PC can be considered as a DTE, and a modem as a DCE. An RS-232 link connects a DTE and a DCE, or uses a crossover (sometimes called a “null-modem”) cable to make the connection as if it were between a DTE and a DCE (as when connection two PCs to each other with a serial cable). DTE and DCE configurations for cables and connectors are show in “Connectors and cables” on page 14.

SIGNALLING

The RS-232 standard defines 25 signal lines in its interface, although in practice PCs rarely use more than nine of these lines. Just three of these lines—RD, TD, and GND—are required for bi-directional serial data communication. The rest, including the remainder of the basic nine—DCD, DTR, DSR, RTS, CTS, and RI—are designated for a variety of control purposes. The significance of the main nine serial signals is described in “RS-232 signal descriptions” on page 13.

ELECTRICAL

Voltages. RS-232 signals are indicated by voltage differences with respect to a ground signal, and can vary between +3 to +15 volts and -3 to -15 volts. At the same time, serial receivers must be undamaged by voltages up to ±25 volts.

The control lines use a “positive” logic to indicate their state. That is, a positive voltage on a wire carrying a control signal (DCD, DTR, DSR, RTS, CTS, and RI on a nine-wire serial connection) indicates that the control signal involved can be described as “On,” “Asserted,” or “True.” A negative voltage on a control line indicates the opposite; that the control signal involved can be described as “Off,” “De-asserted,” or “False.”

The data lines are just the opposite. They use a so-called “negative” logic, meaning that a negative voltage on the wire carrying the data signal (RD or TD) is described as “On,” “Asserted,” or “True,” and that a positive voltage on the wire is interpreted as “Off,” “Deasserted,” or “False.”

Timing. RS-232 also defines the timing of electrical signalling. An RS-232 link differentiates between the bits of a serial data stream by collecting information about the voltage of its data lines. In the simplest terms, it does this by monitoring the lines for a start bit (described below in “Stop and start bits” on page 4), and then reading the state of the line at predefined intervals, with each interval representing the next bit in the stream of data. The timing of these intervals is determined by the data rate of the link—this process in effect makes the serial connection follow a clock within each byte, although the timing between one byte of data and the next is not dictated by a clock.

The number of readings taken within a byte is determined by the number of data bits set for the link (described below in “Data bits” on page 5), whether the connection has a parity bit (described below in “Parity bits” on page 4), and the configuration of stop bits (described below in “Stop and start bits” on page 4). Once the stop bit is read, the connection waits for the next start bit to arrive.

MECHANICAL

Each line in an RS-232 interface is assigned a pin number for the various connectors that RS-232 can use. The nine primary lines, and their assignments in DB-9 and DB-25 connectors, are described in “Connectors and cables” on page 14.

Synchronous vs. asynchronous RS-232

RS-232 signals can be synchronous or asynchronous. Synchronous RS-232 signals are synchronized by a clock that dictates the timing of each bit that is sent. The timing provided by the clock is shared by both sides of the serial connection, so each side is aware of the timing of the next byte of data.

Asynchronous RS-232 signals are delineated by voltage changes that will identify the start and stop of any byte of data. Within any byte of data, the receiver is actually applying a clock to measure the elements of the data transmission, and will sample the voltage level within the byte the number of times that corresponds to the number of discrete bits of data it expects the byte, along with its framing and possible parity bits, to have.

This paper focuses on aysnchronous RS-232.

UARTs

DEFINITION

The core of any serial port is a UART, or Universal Asynchronous Receiver-Transmitter. A UART is generally an integrated circuit. It contains the software that converts a parallel data stream of 8-bit bytes into a serial format of single bits, or vice versa. When transmitting data over a serial line, the data is transmitted one bit at a time. When receiving data from a serial line, the UART converts serial data back to parallel data for the computer's CPU to use.

To transmit a byte of data, the UART takes it, breaks it into its constituent bits, and packages it so that the byte can be successfully identified and reassembled by the receiving serial port's UART. This process is called "framing" the byte.

The UART then transmits the byte through the wires of the serial connection.

The UART also manages the way data is handled between its transmit and receive buffers and the computer's CPU.

STOP AND START BITS

When framing a byte, the UART adds bits that indicate the start and stop of a byte. Without these start and stop bits, the flow of serial data would be an undifferentiated stream. Some people talk this way, in fact.

The UART adds a start bit and optionally either 1 or 2 stop bits. The voltage of the second stop bit is the same as the voltage of the first, so in practical terms a second stop bit is redundant. Because two stop bits look like one long stop bit, a serial link can work (albeit inefficiently) with one side set to frame bytes with one stop bit and with the other side set to frame bytes with two stop bits. In rare cases, 1½ stop bits can be used; this is in effect simply holding the stop bit's voltage for 1½ the duration of a single stop bit.

PARITY BITS

As it frames a byte for transmission, the UART may also add a bit called a "parity bit" to provide a means of checking that the data received matches the data that was transmitted. RS-232 parity bits can have one of five possible settings: none, even, odd, mark, or space.

Parity "None". If parity is set to "none," no parity bit is added to the data being sent.

Parity "Even". If parity is enabled and set to "even," the UART looks at each data bit of the byte it is preparing to send, and counts the number of data bits with a logic value of 1. If the count is an even number, then a parity bit with a logic value of 1 is created and added to the end of the byte, after the data bits and before the stop bit (or bits). When this byte is received on the other side of the serial connection, the receiving UART counts the number of data bits with a logic value of 1 and determines whether the count is even or odd. It then compares its assessment of even or odd status with the status indicated by the parity bit. If they match, all is fine; if they don't match, an error is reported. This is done by setting bit 2 in the Line Status Register to a logic value 1; see "REGISTER 5: Line Status Register (LSR)" on page 12.

Parity "Odd". If parity is enabled and set to "Odd," the UART creates a parity bit just as it does when creating an "Even" parity bit, except that the logic value "1" is set when the count of bits with a logic value 1 is *odd*. The receiving UART assesses the quality of the received data just as it does with an "Even" parity bit.

Parity "Mark". If parity is enabled and set to "Mark," the transmitting UART will attach a parity bit with a logic value 1 to the byte. The receiving UART will check that this bit is still 1 when it is received.

Parity “Space”. If parity is enabled and set to “Space,” the transmitting UART will attach a parity bit with a logic value 0 to the byte. The receiving UART will check that this bit is still 0 when it is received.

A couple of things can be seen here: first, each side of a serial connection needs to understand the parity bit assumptions of the other side. Without proper parity settings, the data will not make sense when it is received, or it will make an inverted sense: correctly transmitted data will be judged incorrect, and vice versa. Second, mark and space parity are not really worth the bandwidth they employ: they are created without reference to the data itself, and are a poor means of assessing the quality of data.

DATA BITS

RS-232 serial data can be configured to specify the number of data bits in a frame. The number of data bits used can be 4, 5, 6, 7, or 8. When the UART sends the data bits, it sends the data bit of the byte in the order of least significant bit to most significant bit. The UART’s data register is shown in Table 2 on page 9.

BUFFERS AND TRIGGERS

Most serial ports on PCs today have UARTs equipped with buffers, or small portions of memory devoted to storing groups of bytes in transit through the UART. Buffers greatly improve the efficiency of CPU/serial port interactions, and help to reduce transmission errors called “overrun errors” that occur when a new byte arrives at a serial port before the previous byte has left the UART.

Serial port buffers are called “FIFO” buffers because they operate in a first-in, first-out (FIFO) fashion: bytes put into the buffer are released from the buffer before bytes arriving later.

When a UART needs the CPU to process data into or out of a buffer, it generates a hardware interrupt request, or IRQ. Interrupt requests are the most basic level of flow control: they indicate to the CPU of the computer managing the serial port that the serial port needs some sort of attention.

A UART’s receive buffer stores data received from the serial port, and gathers bytes for the CPU to collect. If the application using the UART is aware of the UART's capability, input data overruns can be greatly reduced.

A UART’s transmit buffer stores bytes coming from the CPU. It reduces the frequency of interruptions to the CPU’s operation by minimizing the number of interrupts the UART must request to have its buffer refilled.

UARTs have adjustable settings governing when data in the buffers is released. These settings are the "trigger levels" of the UART. The trigger level for the input buffer indicates the number of bytes of data required to be in that buffer before the CPU is asked to collect them; the trigger level for the transmit buffer indicates the number of bytes of data remaining in that buffer before the CPU is asked to refill it.

WINDOWS PORT CONFIGURATION SCREENS

The Windows dialog box for setting the number of data bits, the parity bit, and the number of stop bits is shown in Figure 6, “Serial port Settings dialog box,” on page 21.

The Windows dialog box for setting buffer levels is shown in Figure 7, “Serial port Advanced Settings dialog box,” on page 21.

UART HISTORY

In the early days of the PC, serial transmissions were handled by the 8250 UART. This early UART had a number of limitations, including having an input register that could hold only one byte at a time. Its successor, the 16450 UART, had the same limitation in its input register. As a result, these UARTs were not usually capable of handling the data from newer modems that had speeds greater than 9600 bits per second. When the data flow was faster than the UART could handle the chance arose of input data overruns: a character of data would still be left in the input buffer when the next byte of data arrived, and so would be lost.

The next advance in UART design was the 16550 UART, and it remains a generally popular UART today. This UART is capable of data speeds that can match the speeds of modems transmitting data across conventional telephone lines. It has two 16-byte FIFO buffers, one for transmitting data, and one for receiving data, with independently adjustable trigger levels.

Later UARTs such as the 16650 and 16750 continue this evolution. The 16650 UART has a 32-byte buffer; the 16750 has a 64-byte buffer. Each also has adjustable trigger levels.

How the UART structures serial communication

FIGURE 1. Transmitting serial data

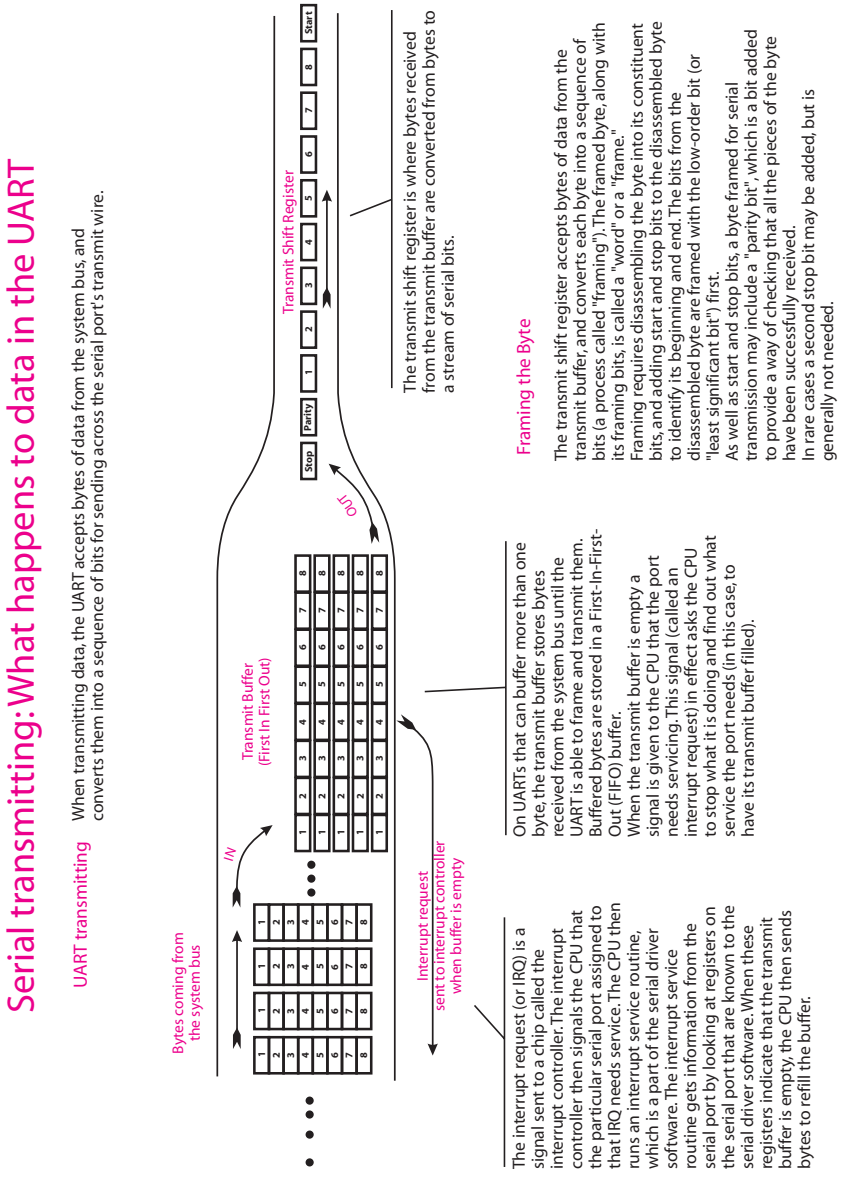
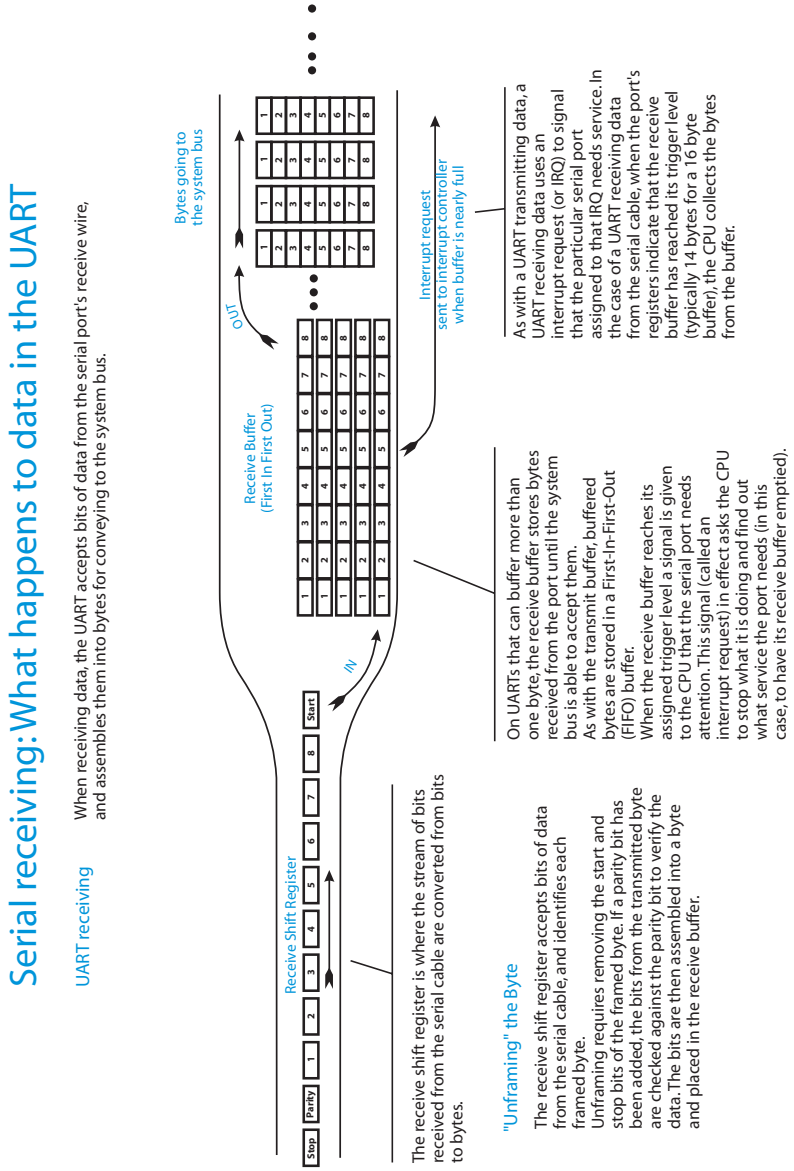


FIGURE 2. Receiving serial data



UART registers

The registers of a UART represent, at any given moment, the state of the UART's operation and of the data it contains. The UART's registers record such things as the state of the UART's transmit and receive buffers, the trigger levels for the buffers, the types of interrupts the UART can produce and their status, the settings currently in use for framing bytes, the possible error conditions the UART may be experiencing, the port's baud rate, and the settings and status of modem control lines.

The descriptions of UART registers below apply to 16550 and 16650 UARTs. Specifics on the registers of other UARTs, such as the 16450 and 16750, are available in manufacturers' data sheets. The names and abbreviations of registers may vary from one UART manufacturer to another, but the basic functioning is the same. Serial ports with UARTs from different manufacturers might otherwise not work together. For detailed descriptions of UARTs, see the manufacturers' data sheets.

TABLE 1. Summary of UART registers

Register	Description
Register 0 R/W	Receiver (read) / Transmitter (write)
Register 1 R/W	Interrupt Enable Register (IER)
Register 2 R	Interrupt Identification Register (IIR)
Register 2 W	FIFO Control Register (FCR: 16550/16650)
Register 2 R/W	Enhanced Features Register (EFR: 16650 only)
Register 3 R/W	Line Control Register (LCR)
Register 4 R/W	Modem Control Register (MCR)
Register 5 R/W	Line Status Register (LSR)
Register 6 R/W	Modem Status Register (MSR)
Register 7 R/W	Scratch register. Not used.

REGISTER 0: DATA REGISTER

Reading from the data register fetches the next input byte, once it is ready. Writing to the data register transmits the byte written to it over the serial line.

TABLE 2. Data register

Bit	Description
bit 7	Data bit 7
bit 6	Data bit 6
bit 5	Data bit 5
bit 4	Data bit 4
bit 3	Data bit 3
bit 2	Data bit 2
bit 1	Data bit 1
bit 0	Data bit 0

Note: Data bit 0 is the least significant bit; that is, it is the first bit serially transmitted or received.

REGISTER 1: INTERRUPT ENABLE REGISTER (IER)

The Interrupt Enable register enables each of four types of interrupts when the appropriate bit is set to a one.

TABLE 3. Interrupt Enable Register (IER)

Bit	Description
bit 3	Enable interrupt on RS-232 input.
bit 2	Enable interrupt on receiver error or break.
bit 1	Enable interrupt on transmitter buffer empty (TBE).
bit 0	Enable interrupt on received data (RxRDY).

REGISTER 2: INTERRUPT IDENTIFICATION REGISTER (IIR)

Reading the Interrupt Identification (read only) register once an interrupt has occurred identifies the interrupt as follows:

TABLE 4. Interrupt Identification Register (IIR)—read only

Bit 2	Bit 1	Bit 0	Priority	Interrupt
0	0	1	none	none
1	1	0	0 (high)	Serialization or break.
1	0	0	1	Received data.
0	1	0	2	Transmit buffer empty.
0	0	0	3 (low)	RS-232 input.

REGISTER 2: INTERRUPT IDENTIFICATION REGISTER (IIR)

In the 16550 and 16650, Register 2 (write only) is also the FIFO control register. Writing bits 6 & 7 will set the RX FIFO trigger level (number of bytes received before an interrupt is generated).

TABLE 5. Interrupt Identification Register (IIR)—write only

Bit 7	Bit 6	16550 Trigger	16650 Trigger
0	0	1 byte	8 bytes
0	1	4 bytes	16 bytes
1	0	8 bytes	24 bytes
1	1	14 bytes	28 bytes

The TX FIFO level can also be set in the 16650 by setting bits 4 & 5. See the 16650 data sheet for more details.

REGISTER 2: ENHANCED FEATURE REGISTER (EFR) [16650 ONLY]

The EFR can only be accessed after writing a BF to the Line Control Register (LCR), after which the advanced features on the 16650 are enabled by setting bit 4 of the EFR. For more details, see the 16650 data sheet.

REGISTER 3: LINE CONTROL REGISTER (LCR)

RS-232 line parameters are selected by writing to this register.

TABLE 6. Line Control Register (LCR)

Bit	Description
bit 7	Divisor Latch Access Bit (DLAB) = 0 When the Divisor Latch Access Bit (DLAB) is 1, registers 0 and 1 become the LS and MS bytes of the Baud Rate Divisor registers. The baud rate is computed as (115200 / BaudRate-Divisor) for a given crystal. Common baud rates are matched to a selection of divisors in the table below.
bit 6	BREAK on (1), off (0).
bit 5	Stick parity (1)
bit 4	Even parity select (1)
bit 3	Parity enable (1)
bit 2	One stop bit (0), two stop bits (1).
bits 1-0	Data bits = 5 (00), 6 (01), 7 (10), 8 (11).

In effect:
None (000)
Odd (001)
Even (011)
Mark (101)
Space (111)

TABLE 7. Parity definitions

Parity	Description
Odd	The parity bit is 1 if the sum of the data bits is odd.
Even	The parity bit is 1 if the sum of the data bits is even.
None	There is no parity bit.
Mark	The parity bit is always set to 0.
Space	The parity bit is always set to 1.

TABLE 8. Common baud rates

Baud	Divisor
300	0180
1200	0060
2400	0030
4800	0018
9600	000C
19200	0006
38400	0003
57600	0002
115200	0001

NOTES:

1. Must write BF hex to LCR before EFR [16650 ONLY] can be accessed (see 16650 data sheet).

REGISTER 4: MODEM CONTROL REGISTER (MCR)

RTS, DTR, loopback testing, and General Purpose Outputs #1 and #2 are controlled by the Modem Control register as follows:

TABLE 9. Modem Control Register (MCR)

Bit	Description
bit 7	Clock select. X1 (if 0), X4 (if 1). [16750 ONLY]
bit 6	IR enable [16650 ONLY]
bit 5	Interrupt type select [16650 ONLY]
bit 4	Enable local loopback.
bit 3	Enable GP02. Necessary for UART interrupts.
bit 2	Enable GP01.
bit 1	Set / clear RTS.
bit 0	Set / clear DTR.

REGISTER 5: LINE STATUS REGISTER (LSR)

Reading the Line Status register provides status information as follows (1 for TRUE, 0 for FALSE):

TABLE 10. Line Status Register (LSR)

Bit	Description
bit 7	FIFO data error [16650].
bit 6	Transmitter empty.
bit 5	Transmitter buffer empty.
bit 4	BREAK detect.
bit 3	Framing error. Set to logic 1 when the received character does not have the correct stop bit(s).
bit 2	Parity error. Set to logic 1 when the parity calculation for the receive character does not match the parity setting in the Line Control Register.
bit 1	Overrun error. Set to logic 1 to indicate the character in the receive shift register has been overwritten before being transferred to the receive FIFO buffer. The character in the receive shift register is not transferred to the receive FIFO buffer, so the data in the receive FIFO buffer is not corrupted.
bit 0	Data ready. Set to logic 1 when a complete incoming character has been placed in the receive shift register or receive FIFO buffer. Reset to logic 0 when data in receive shift register or receive FIFO buffer has been read.

REGISTER 6: MODEM STATUS REGISTER (MSR)

Reading the Modem Status register provides the following status information (1 for TRUE, 0 for FALSE):

TABLE 11. Modem Status Register (MSR)

Bit	Description
bit 7	DCD status.
bit 6	RI status.
bit 5	DSR status.
bit 4	CTS status.
bit 3	Delta DCD status.
bit 2	Delta RI status.
bit 1	Delta DSR status.
bit 0	Delta CTS status.

The delta bits (bits 0 through 3) are set whenever one of the status bits (bits 4 through 7) changes since the last time that the Modem Status register was read. Reading the Modem Status register clears the delta bits.

REGISTER 7: SCRATCH REGISTER

Register 7 has no associated function and is available for customized programming uses.

RS-232 signal descriptions

The following signal descriptions apply to asynchronous RS-232.

DTR

Data Terminal Ready. Used by a piece of Data Terminal Equipment to signal that it is available for communication.

DSR

Data Set Ready. The companion signal to DTR, it is used by a piece of Data Circuit-Terminating Equipment to signal that it is available for communication.

CTS

Clear to Send. Used by a piece of Data Circuit-Terminating Equipment to signal it is available to send data, and also used in response to an RTS request for data.

RTS

Request to Send. Used by a piece of Data Terminal Equipment to indicate that it has data to send.

DCD

Data Carrier Detect. Used by a piece of Data Circuit-Terminating Equipment to indicate to the Data Terminal Equipment that it has received a carrier signal from the modem and that real data is being transmitted. Sometimes abbreviated as CD.

RI — **Ring Indicator.** Used by a Data Circuit-Terminating Equipment modem to tell the Data Terminal Equipment that the phone is ringing and that data will be forthcoming.

TD — **Transmit Data.** This wire is used for sending data. Sometimes abbreviated as TXD. This wire will also carry flow control information if software flow control is enabled.

RD — **Receive Data.** This wire is used for receiving data. Sometimes abbreviated as RXD. This wire will also carry flow control information if software flow control is enabled.

GND — **Ground.** This pin is the same for Data Circuit-Terminating Equipment and Data Terminal Equipment, and it provides the return path for both data and handshake signals.

Connectors and cables

CABLE LENGTHS

The original specification for RS-232 cabling recommended a maximum cable length of 50 feet, and this is still a good conservative rule of thumb. To accurately judge maximum cable lengths, it is necessary to consider the specifics of the cable and the data rate that the serial connection will be using.

The specification provides a basis for calculating cable length based on the capacitance at the receiving end of the link. This value is a function of the nature of the receiver and cabling. Various manufacturers will publish capacitance values for their products to assist in determining an acceptable cable length in a given application.

NUMBER OF WIRES

The number of wires in a serial connection can vary. A three-wire serial connection, while capable of supporting a rudimentary serial connection, will not handle hardware handshaking, as it will have only transmit, receive, and ground wires. A 9-pin D-sub cable will accommodate most serial communications between PCs and peripherals. A full 25-wire cable with DB-25 connectors will be capable of carrying any RS-232 signal.

FIGURE 3. DB-9 pin locations

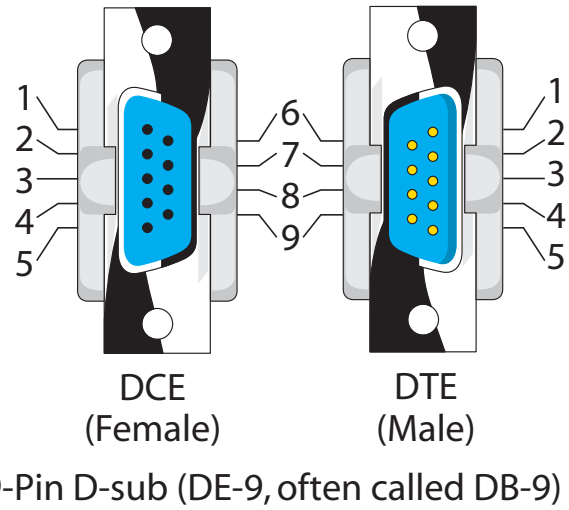


FIGURE 4. DB-25 pin locations

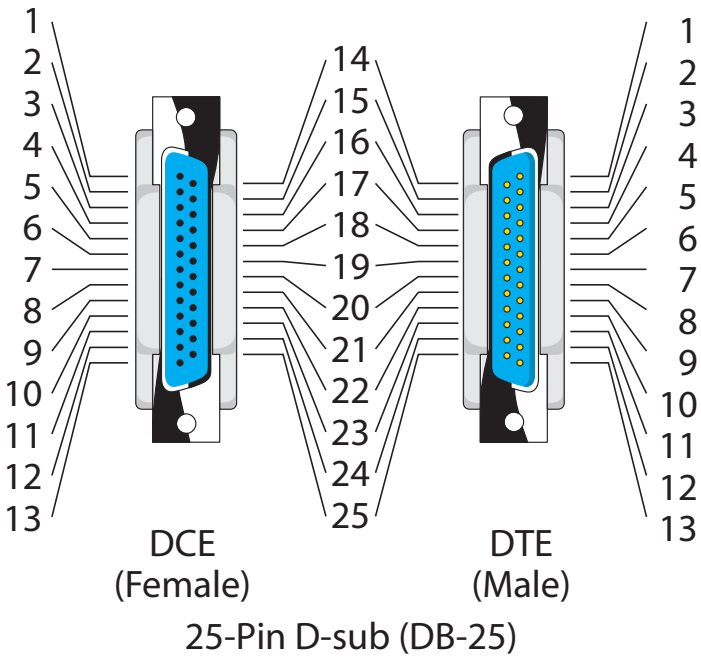


TABLE 12. DTE-to-DCE DB-9 connection (Straight cable)

DB-9 DTE Device (Computer)		DB-9 DCE Device (Modem)
Pin # / RS-232 Signal Name	**Signal Direction**	**Pin # / RS-232 Signal Name**
#1 Data Carrier Detect (DCD)	←	#1 Data Carrier Detect (DCD)
#2 Receive Data (RD)	←	#2 Receive Data (RD)
#3 Transmit Data (TD)	→	#3 Transmit Data (TD)
#4 DTE Ready/Data Terminal Ready (DTR)	→	#4 DTE Ready/Data Terminal Ready (DTR)
#5 Signal Ground/Common (GND)	—	#5 Signal Ground/Common (GND)
#6 DCE Ready/Data Set Ready (DSR)	←	#6 DCE Ready/Data Set Ready (DSR)
#7 Request to Send (RTS)	→	#7 Request to Send (RTS)
#8 Clear to Send (CTS)	←	#8 Clear to Send (CTS)
#9 Ring Indicator (RI)	←	#9 Ring Indicator (RI)
Soldered to DB-9 metal—shield	—	Soldered to DB-9 metal—shield

TABLE 13. DCE-to-DCE DB-9 connection (Crossover cable)

DB-9 DCE Device (Modem)		DB-9 DCE Device (Modem)
Pin # / RS-232 Signal Name	**Signal Direction**	**Pin # / RS-232 Signal Name**
#1 Data Carrier Detect (DCD)		#1 Data Carrier Detect (DCD)
#2 Receive Data (RD)	→	#2 Receive Data (RD)
#3 Transmit Data (TD)	←	#3 Transmit Data (TD)
#4 DTE Ready/Data Terminal Ready (DTR)	→	#4 DTE Ready/Data Terminal Ready (DTR)
#5 Signal Ground/Common (GND)	—	#5 Signal Ground/Common (GND)
#6 DCE Ready/Data Set Ready (DSR)	→	#6 DCE Ready/Data Set Ready (DSR)
#7 Request to Send (RTS)	←	#7 Request to Send (RTS)
#8 Clear to Send (CTS)	→	#8 Clear to Send (CTS)
#9 Ring Indicator (RI)		#9 Ring Indicator (RI)
Soldered to DB-9 metal—shield	—	Soldered to DB-9 metal—shield

TABLE 14. DTE-to-DCE DB-9/DB-25 connection (Straight cable)

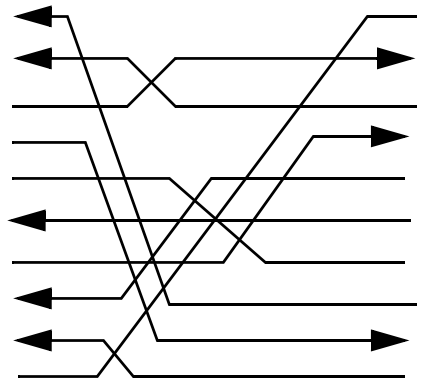
DB-9 DTE Device (Computer)		DB-25 DCE Device (Modem)
Pin # / RS-232 Signal Name	**Signal Direction**	**Pin # / RS-232 Signal Name**
#1 Data Carrier Detect (DCD)		#1 Shield to Frame Ground
#2 Receive Data (RD)		#2 Transmit Data (TD)
#3 Transmit Data (TD)		#3 Receive Data (RD)
#4 DTE Ready/Data Terminal Ready (DTR)		#4 Request to Send (RTS)
#5 Signal Ground/Common (GND)		#5 Clear to Send (CTS)
#6 DCE Ready/Data Set Ready (DSR)		#6 DCE Ready/Data Set Ready (DSR)
#7 Request to Send (RTS)		#7 Signal Ground/Common (GND)
#8 Clear to Send (CTS)		#8 Data Carrier Detect (DCD)
#9 Ring Indicator (RI)		#20 DTE Ready/Data Terminal Ready (DTR)
Soldered to DB-9 metal—shield		#22 Ring Indicator (RI)

TABLE 15. DCE-to-DCE DB-9/DB-25 connection (Crossover cable)

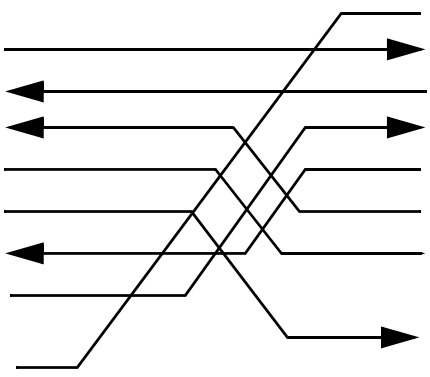
DB-9 DCE Device (Modem)		DB-25 DCE Device (Modem)
Pin # / RS-232 Signal Name	**Signal Direction**	**Pin # / RS-232 Signal Name**
#1 Data Carrier Detect (DCD)		#1 Shield to Frame Ground
#2 Receive Data (RD)		#2 Transmit Data (TD)
#3 Transmit Data (TD)		#3 Receive Data (RD)
#4 DTE Ready/Data Terminal Ready (DTR)		#4 Request to Send (RTS)
#5 Signal Ground/Common (GND)		#5 Clear to Send (CTS)
#6 DCE Ready/Data Set Ready (DSR)		#6 DCE Ready/Data Set Ready (DSR)
#7 Request to Send (RTS)		#7 Signal Ground/Common (GND)
#8 Clear to Send (CTS)		#8 Data Carrier Detect (DCD)
#9 Ring Indicator (RI)		#20 DTE Ready/Data Terminal Ready (DTR)
Soldered to DB-9 metal—shield		#22 Ring Indicator (RI)

TABLE 16. DTE-to-DCE DB-25 connection (Straight cable)

DB-25 DTE Device (Computer)		DB-25 DCE Device (Modem)
Pin # / RS-232 Signal Name	**Signal Direction**	**Pin # / RS-232 Signal Name**
#1 Shield to Frame Ground		#1 Shield to Frame Ground
#2 Transmit Data (TD)		#2 Transmit Data (TD)
#3 Receive Data (RD)		#3 Receive Data (RD)
#4 Request to Send (RTS)		#4 Request to Send (RTS)
#5 Clear to Send (CTS)		#5 Clear to Send (CTS)
#6 DCE Ready/Data Set Ready (DSR)		#6 DCE Ready/Data Set Ready (DSR)
#7 Signal Ground/Common (GND)		#7 Signal Ground/Common (GND)
#8 Data Carrier Detect (DCD)		#8 Data Carrier Detect (DCD)
#20 DTE Ready/Data Terminal Ready (DTR)		#20 DTE Ready/Data Terminal Ready (DTR)
#22 Ring Indicator (RI)		#22 Ring Indicator (RI)

TABLE 17. DCE-to-DCE DB-25 connection (Crossover cable)

DB-25 DCE Device (Modem)		DB-25 DCE Device (Modem)
Pin # / RS-232 Signal Name	**Signal Direction**	**Pin # / RS-232 Signal Name**
#1 Shield to Frame Ground		#1 Shield to Frame Ground
#2 Transmit Data (TD)		#2 Transmit Data (TD)
#3 Receive Data (RD)		#3 Receive Data (RD)
#4 Request to Send (RTS)		#4 Request to Send (RTS)
#5 Clear to Send (CTS)		#5 Clear to Send (CTS)
#6 DCE Ready/Data Set Ready (DSR)		#6 DCE Ready/Data Set Ready (DSR)
#7 Signal Ground/Common (GND)		#7 Signal Ground/Common (GND)
#8 Data Carrier Detect (DCD)		#8 Data Carrier Detect (DCD)
#20 DTE Ready/Data Terminal Ready (DTR)		#20 DTE Ready/Data Terminal Ready (DTR)
#22 Ring Indicator (RI)		#22 Ring Indicator (RI)

Serial port resource requirements

Serial ports use a couple of system resources from the computers in which they are installed. First, each serial port needs a reserved range of addresses. As well, serial ports need an assigned interrupt request level, or IRQ.

Resource setting is automatic on systems that support Microsoft's Plug and Play standard. This applies to Lava's PCI serial port cards. ISA cards usually have their resource settings configured by jumper switches on the card. An exception is Lava's LavaPort-PnP, a single-port 16650 UART Plug and Play ISA card.

The resources used by a serial port cannot conflict with the other components in the system, including other serial ports.

ADDRESSES

The first category of resource used by serial ports is addresses. When speaking of serial ports in a system, they are conventionally referred to as *COM* ports. This naming stands for "communications ports." When a system has multiple ports they are referred to in numerical order of address as COM1, COM2, and so on.

COM Port addressing. The standard PC serial ports are placed on conventional addresses and interrupts. See Table 18 on page 19.

TABLE 18. Conventional serial port resource allocations

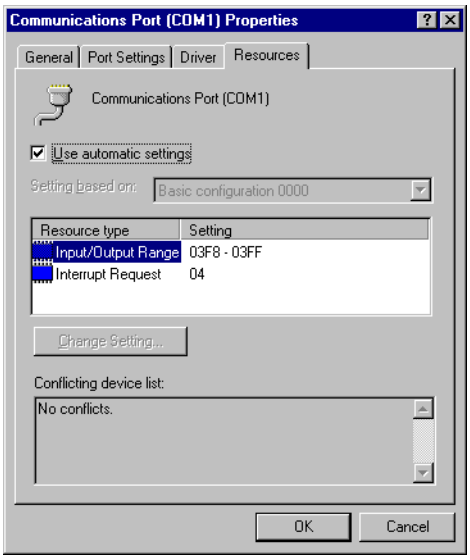
Port	Address	IRQ
COM1	3F8h	4
COM2	2F8h	3
COM3	3E8h	4 or 11
COM4	2E8h	3 or 10

INTERRUPT REQUEST LEVELS

To indicate that a serial port needs some form of service, an interrupt request is sent to the system's interrupt controller, which prioritizes interrupt requests before sending an interrupt to the CPU. This signal is sent on an interrupt line that has an assigned number. When these IRQ assignments conflict with other devices such as sound cards, problems might result, and IRQs may need to be reassigned.

Both the I/O address and the IRQ for a serial port can be seen and changed through Windows' Communication Port Properties window, under the Resources tab.

FIGURE 5. Serial port Resources dialog box



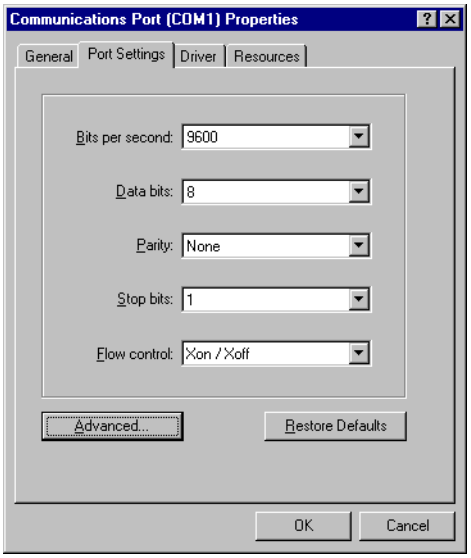
Serial port line settings

In addition to a serial port's system resources, a number of additional settings control how the serial port's UART configures and processes the data it handles. These are the called the "port settings" and include the speed at which the port communicates with another serial port (a function of register settings in the UART), the method of framing of the bytes of data it transmits and receives (also a function of register settings in the UART), and the method of controlling the flow of data into and out of the port. Flow control *inside* the UART is handled by the UART's buffer trigger levels. Both framing and flow control inside the UART have been discussed earlier in this white paper in "UARTs" on page 3.

FLOW CONTROL: CST/RTS AND XON/XOFF

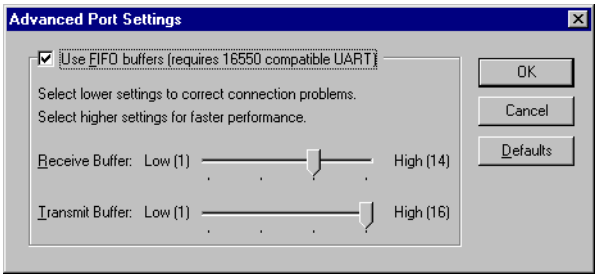
The serial port Settings dialog box has a setting for flow control that allows choosing between hardware or software flow control (called CTS/RTS and Xon/Xoff respectively). Both methods can control the flow of data between the serial port and a device such as a modem. Hardware flow control uses the dedicated CTS and RTS lines of the serial connection to send flow control signals. Software flow control places control signals on the data lines (RD and TD), along with the data being transmitted. In most cases, hardware flow control is preferable because it is more immediately responsive than software flow control. Software flow control can be used when the serial cable does not have CTS and RTS wires.

FIGURE 6. Serial port Settings dialog box



The advanced settings permit configuring the trigger levels for the port's transmit and receive buffers.

FIGURE 7. Serial port Advanced Settings dialog box



How fast are serial ports?

SERIAL UART SPEEDS

The speed of any serial port is largely determined by the UART it uses. The most typical types of UARTs—the 16550 and 16650 UARTs—have theoretical maximum speeds of 115.2 kbps and 460.8 kbps respectively.

These advanced UARTs offer substantial speed advantages over earlier designs. However, for a serial connection to take advantage of an advanced UART, several conditions are necessary:

1. There must be driver software present that is designed to handle the UART.

2. The serial port's buffers and trigger levels must be properly configured.
3. Wherever possible, hardware flow control should be used in preference to software flow control.
4. Last but not least, the peripheral involved must be fast enough to benefit from that UART.

If these conditions are not met, a serial connection will operate below the theoretical speed of its UART.

OTHER SPEED FACTORS

Many other factors can influence port speed. In other words, significant speed differences can result in actual use. On the hardware side these factors include the speed and architecture of the computer's CPU, and peripheral need for high-speed serial ports. On the software side, the type of code and operating system will affect port speed. For example, other things being equal, DOS will have faster serial port transfers than Windows.

Operating system support

Lava's serial port boards are tested and compatible with DOS, Windows 3.1/3.11, Windows 95/98/98SE/NT4/2000/Me/XP, and Linux kernel 2.4+. They have also been implemented successfully on a number of UNIX platforms.

The following tables summarize the configuration of serial ports in the Windows environment. As can be seen, setting the various configuration parameters is handled differently from one version of Windows to another.

TABLE 19. Changing serial port settings: PCI cards

TO CHANGE	Operating System							
	DOS	**95**	**98**	**98SE**	**NT4**	**Me**	**2000**	**XP**
address	address change not possible	Device Manager/ Safe Mode Device Manager	Device Manager/ Safe Mode Device Manager	Device Manager/ Safe Mode Device Manager	Port Properties in Control Panel	Device Manager/ Safe Mode Device Manager	address change not possible	address change not possible

Note: Changes to COM assignments made with the Redirect Utility may require a corresponding change in address.

TABLE 19. Changing serial port settings: PCI cards

TO CHANGE	Operating System							
	DOS	**95**	**98**	**98SE**	**NT4**	**Me**	**2000**	**XP**
IRQ	IRQ change not possible	Device Manager/ Safe Mode Device Manager	Device Manager/ Safe Mode Device Manager	Device Manager/ Safe Mode Device Manager	Port Properties in Control Panel	Device Manager/ Safe Mode Device Manager	IRQ change not possible	IRQ change not possible
COM	COM change not possible	Registry/ Lava Redirect Utility	Registry/ Lava Redirect Utility	Registry/ Lava Redirect Utility	Port Properties in Control Panel	Registry/ Lava Redirect Utility	Device Manager/ Safe Mode Device Manager	Device Manager/ Safe Mode Device Manager

Note: Changes to COM assignments made with the Redirect Utility may require a corresponding change in address.

TABLE 20. Changing serial port settings: ISA cards

TO CHANGE	Operating System							
	DOS	**95**	**98**	**98SE**	**NT4**	**ME**	**2000**	**XP**
address	Jumper	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]
IRQ	Jumper	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]	Jumper + Device Manager[1]
COM	Jumper	Jumper + Registry[2]	Jumper + Registry[2]	Jumper + Registry[2]	Jumper	Jumper + Registry[2]	Jumper + Registry/[2] Device Manager	Jumper + Registry/[2] Device Manager

[1] Changes made to jumper settings need to be updated in the Device Manager. Changes made in Device Manager need to matched to jumpers by physically setting the jumpers.
[2] Jumper settings and registry settings must match. See www.lavalink.com for details.

Lava: the source for serial ports

Lava has a range of products with serial ports: thirteen serial-only cards, four serial/parallel combo cards, and the Lava SPH-USB 1.1 hub.

TABLE 21. Lava serial port products

Product		Ports; Interface
PCI	SSerial-PCI	Single 9-pin port, 16550 UART
PCI	SSerial-PCI/LP	Single 9-pin port, 16550 UART, low profile
PCI	DSerial-PCI	Dual 9-pin ports, 16550 UARTs
PCI	DSerial-PCI/LP	Dual 9-pin ports, 16550 UARTs, low profile
PCI	Quattro-PCI	Four 9-pin ports, 16550 UARTs
PCI	Octopus-550	Eight 9-pin ports, 16550 UARTs
PCI	LavaPort-650	Single 9-pin serial, 16650 UART
PCI	LavaPort-PCI	Dual 9-pin ports, 16650 UARTs
PCI	LavaPort-Quad	Four 9-pin ports, 16650 UARTs
ISA	SSerial-550	Single 25-pin port, 16550 UART, COM 1-4, IRQ 3/4/5/7
ISA	DSerial-550	Dual 9-pin ports, 16550 UARTs, COM 1-4, IRQ 2/3/4/5/7/10/11/12/15
ISA	LavaPort-ISA	Single 9-pin port, 16650 UART, COM 1-4, IRQ 2/3/4/5/10/11/12/15
ISA	LavaPort-PnP	Single 9-pin port, 16650 UART, Plug and Play
PCI	SP-PCI	Single 9-pin serial port, 16550 UART + single bi-directional parallel port
PCI	2SP-PCI	Dual serial ports (9- and 25-pin), 16550 UARTs + single EPP parallel port
PCI	LavaPort-Plus	Dual serial ports (9- and 25-pin), 16650 UARTs + single EPP parallel port
ISA	2SP-ISA	Dual serial ports (9- and 25-pin), 16550 UARTs, COM 1-4, IRQ 2/3/4/5/10/11/12/15 + single bi-directional parallel port, LPT 1/2, IRQ 5/7
USB	SPH-USB 1.1 Hub	3 USB 1.1 downstream ports + single 9-pin serial port, 16550 UART + single bi-directional parallel port

In some cases, the speed limitations of a serial port will limit the potential of a peripheral. The table below shows some situations where this may be the case.

TABLE 22. Product-to-peripheral matchup

Legend ✪ Optimal ● Possible limitation or incompatibility	SSerial-PCI	SSerial-PCI/LP	DSerial-PCI	DSerial-PCI/LP	Quattro-PCI	Octopus-550	LavaPort-650	LavaPort-PCI	LavaPort-Quad	SSerial-550	DSerial-550	LavaPort-ISA	LavaPort-PnP	SP-PCI	2SP-PCI	LavaPort-Plus	2SP-ISA	SPH-USB 1.1 Hub
Serial printer	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪
Serial label printer	✪	✪	✪	●	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪
Serial digital camera	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪
Serial modem	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪
Serial ISDN modem	●	●	●	●	●	●	✪	✪	✪	●	●	✪	✪	●	●	✪	●	●
Serial handheld/PDA	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪
Serial mini keypad	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪
Serial Point-of-Sale devices	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪	✪

Lava Computer MFG Inc., 2 Vulcan Street, Toronto, Ontario, CANADA M9W 1L2
Tel: + 416-674-5942 • Fax: +416-674-8262 • www.lavalink.com