July 2002 Vol.4 No.7

LINK

Lava I/O News

Inside this issue:

LAVA

Serial Ports 1: The UART

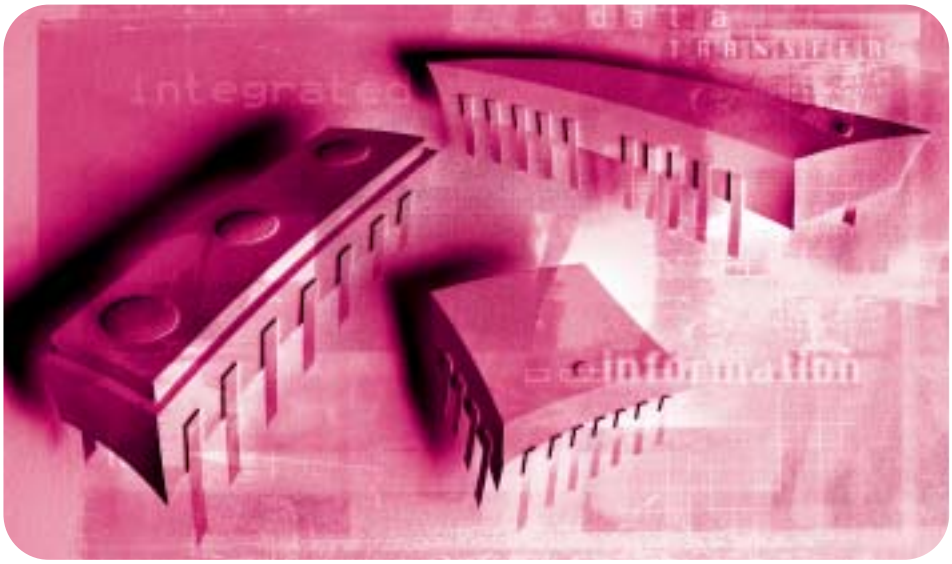
The UART: The heart of serial port communications

We've all got serial ports on our PCs, but because they are a well-established part of our systems, the sophistication of serial ports is easy to ignore. However, understanding how a serial port works makes troubleshooting and optimizing serial connections a whole lot easier.

Behind any serial port communication is a UART, or Universal Asynchronous Receiver-Transmitter. A UART is generally an integrated circuit, or part of an integrated circuit. It contains the firmware 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 UART sends a byte's data one bit at a time. When receiving data from a serial line, the UART converts the serial data back to parallel data for the computer's CPU to use.

Put in the simplest terms, the UART transmits a byte of data by breaking it into its constituent bits, then packaging 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. Next, the UART transmits the byte through the wires of the serial connection. In addition, the UART also manages the way data is handled between the computer's CPU and the UART's transmit and receive buffers. These buffers temporarily store bytes that are in transit, so that the CPU receives fewer interrupt requests from the UART, and so that the UART has fewer input/output overrun errors.

(see diagram, p.2)



What's Goin' on in That Little UART of Yours?

In Windows, the Communications Port Properties dialog box has line setting parameters that configure the serial port's UART, allowing users to set the port's speed, the number of data bits framed, the type of parity the port will use, the number of stop bits, and the type of flow control used. Windows' Advanced Port Settings dialog box permits setting the trigger levels on the UART to ensure optimum performance from the serial port.

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 bits of the byte in the order of least significant bit to most significant bit. Most of the time, 8 bits can be used.

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. Without matching parity settings on both sides of a link, the receiving UART will not make sense of the data.

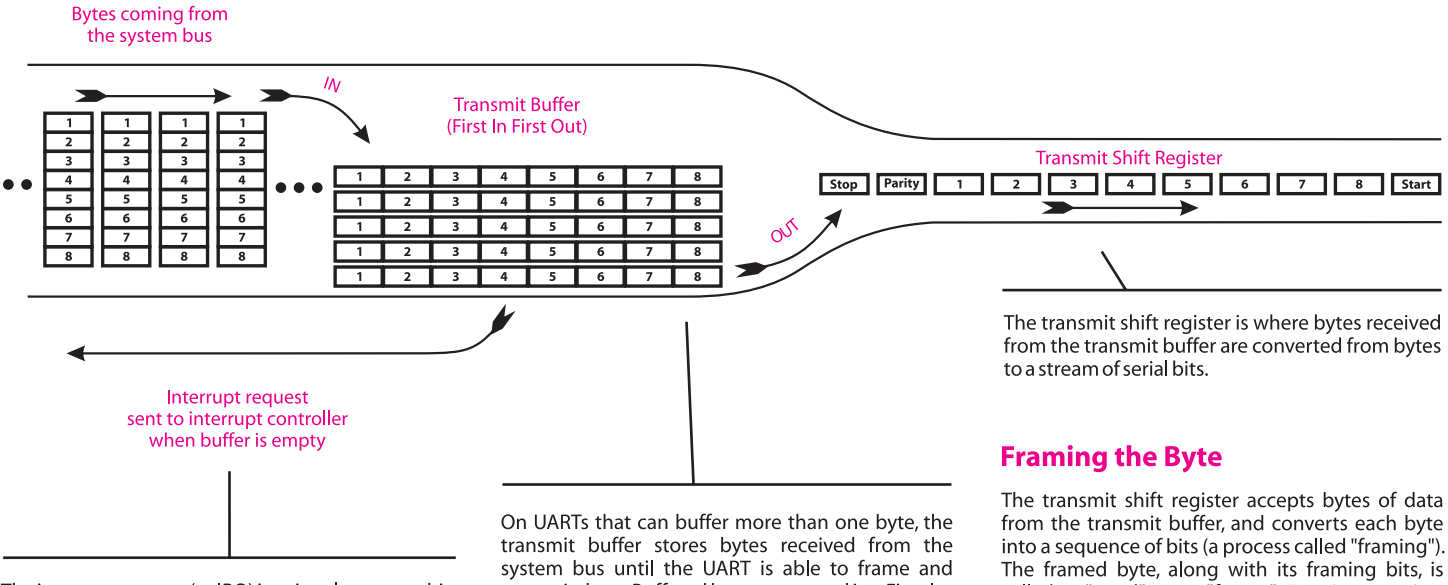
Stop and start bits

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 either 1 or 2 stop bits. In rare cases, 1½ stop bits can be used; this setting in effect simply holds the stop bit's voltage for 1½ times the duration of a single stop bit.

What Happens to Data in the UART

Transmitting

When transmitting data, the UART accepts bytes of data from the system bus, and converts them into a sequence of bits for sending across the serial port's transmit wire.



The transmit shift register is where bytes received from the transmit buffer are converted from bytes to a stream of serial bits.

The interrupt request (or IRQ) is a signal sent to a chip called the interrupt controller. The interrupt controller then signals the CPU that the particular serial port assigned to that IRQ needs service. The CPU then runs an interrupt service routine, which is a part of the serial driver software. The interrupt service routine gets information from the serial port by looking at registers on the serial port that are known to the serial driver software. When these registers indicate that the transmit buffer is empty, the CPU then sends bytes to refill the buffer.

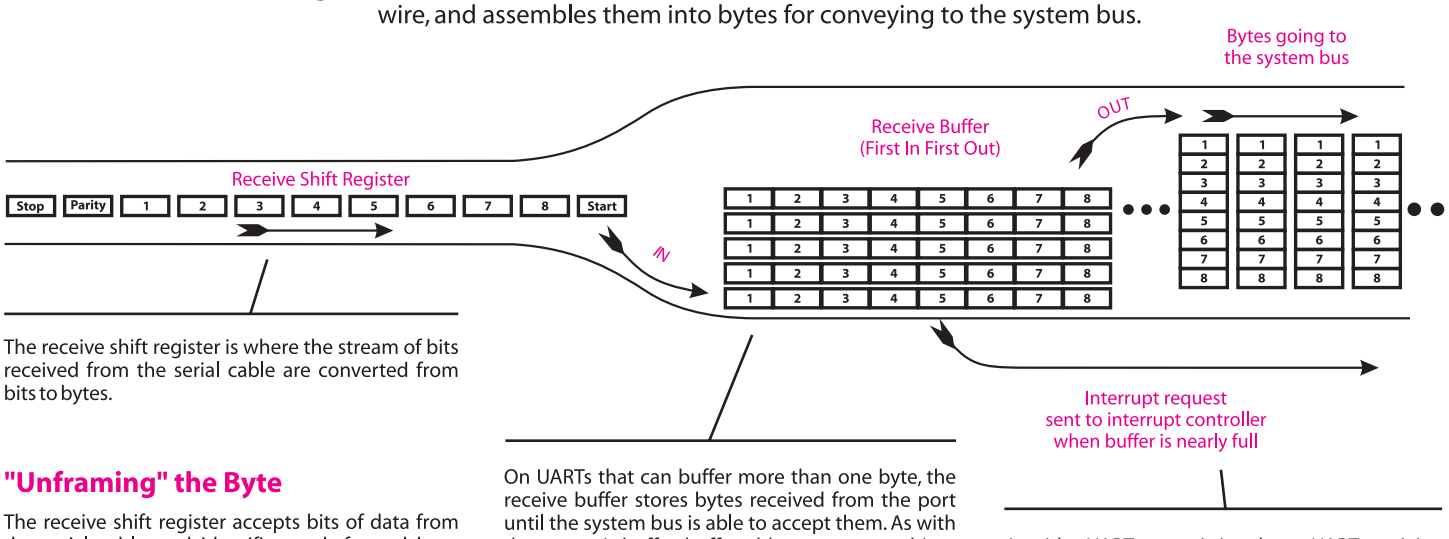
On UARTs that can buffer more than one byte, the transmit buffer stores bytes received from the system bus until the UART is able to frame and transmit them. Buffered bytes are stored in a First-In-First-Out (FIFO) buffer. When the transmit buffer is empty a signal is given to the CPU that the port needs servicing. This signal (called an interrupt request) in effect asks the CPU to stop what it is doing and find out what service the port needs (in this case, to have its transmit buffer filled).

Framing the Byte

The transmit shift register accepts bytes of data from the transmit buffer, and converts each byte into a sequence of bits (a process called "framing"). The framed byte, along with its framing bits, is called a "word" or a "frame." Framing requires disassembling the byte into its constituent bits, and adding start and stop bits to the disassembled byte to identify its beginning and end. The bits from the disassembled byte are framed with the low-order bit (or "least significant bit") first. As well as start and stop bits, a byte framed for serial transmission may include a "parity bit", which is a bit added to provide a way of checking that all the pieces of the byte have been successfully received. In rare cases a second stop bit may be added, but is generally not needed.

Receiving

When receiving data, the UART accepts bits of data from the serial port's receive wire, and assembles them into bytes for conveying to the system bus.



The receive shift register is where the stream of bits received from the serial cable are converted from bits to bytes.

"Unframing" the Byte

The receive shift register accepts bits of data from the serial cable, and identifies each framed byte. Unframing requires removing the start and stop bits of the framed byte. If a parity bit has been added, the bits from the transmitted byte are checked against the parity bit to verify the data. The bits are then assembled into a byte and placed in the receive buffer.

On UARTs that can buffer more than one byte, the receive buffer stores bytes received from the port until the system bus is able to accept them. As with the transmit buffer, buffered bytes are stored in a First-In-First-Out (FIFO) buffer. When the receive buffer reaches its assigned trigger level a signal is given to the CPU that the serial port needs attention. This signal (called an interrupt request) in effect asks the CPU to stop what it is doing and find out what service the port needs (in this case, to have its receive buffer emptied).

As with a UART transmitting data, a UART receiving data uses an interrupt request (or IRQ) to signal that the particular serial port assigned to that IRQ needs service. In the case of a UART receiving data from the serial cable, when the port's registers indicate that the receive buffer has reached its trigger level (typically 14 bytes for a 16 byte buffer), the CPU collects the bytes from the buffer.

(continued from page1)

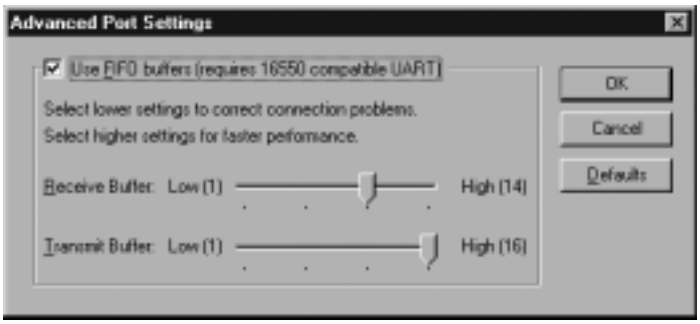
Flow control

The Communications 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 dedicated lines of the serial connection to send flow control signals. Software flow control places control signals on the same lines that the data uses, and they travel along with the data. 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 dedicated control wires.

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.

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.



Set buffer levels in Windows' Advanced Port Settings.

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 buffers, one for transmitting data, and one for receiving data, with independently adjustable levels — called "trigger levels" — for emptying and refilling its buffers.

Later UARTs such as the 16650 and 16750 continue this evolution. The 16650 UART, used on all of Lava's "LavaPort" serial cards, has a 32-byte buffer; the 16750 has a 64-byte buffer. Each also has adjustable trigger levels.

Profile

Since 1989, Universal MicroElectronics (UME) has built its reputation as an East Coast leader in CPU, memory, and peripheral distribution. With an extensive selection of brand name components, UME is dedicated to providing its customers with genuine, high quality products at the industry's best prices. Their strong purchasing power enables them to offer not only a comprehensive inventory and aggressive pricing, but also RapidServ delivery. With a host of options including same-day shipping and local delivery/pick up services, you can count on UME to get an order to you when you need it. UME is committed to delivering the quality products and services resellers need to build a successful business.

UME carries memory, processors, hard drives, motherboards, video cards, wireless networking, network cards, cd-rom/dvd/cd rewriters, modems, sound cards, monitors, and of course Lava boards. Have a technical concern with one of UME's products? UME offers dedicated technical support to assist with all inquiries.

UME offers:

- A Comprehensive Product line
- Same Day Shipping
- Convenient Payment Terms
- Consistent Availability
- Competitive Prices
- First Class Service
- Lava Products

Universal MicroElectronics Inc.
400-E Apgar Drive
Somerset, NJ
USA 08873
TEL: 732.469.0033
FAX: 732.469.0038
www.umemem.com

PRODUCT SUMMARY

Serial Boards

PCI	SSerial-PCI	Single 9-pin serial, 16550 UART
	SSerial-PCI/LP	Single 25-pin serial, 16550 UART, low profile
	DSerial-PCI	Dual 9-pin serial, 16550 UARTs
	DSerial-PCI/LP	Dual 9-pin serial, 16550 UARTs, low profile
	Quattro-PCI	Four-port 9-pin serial, 16550 UARTs
	Octopus-550	Eight-port 9-pin serial, 16550 UARTs
	LavaPort-650	Single 9-pin serial, 16650 UART
	LavaPort-PCI	Dual 9-pin serial, 16650 UARTs
	LavaPort-Quad	Four-port 9-pin serial, 16650 UARTs
ISA	SSerial-550	Single 25-pin serial, Com 1-4, 16550 UART, IRQ 3/4/5/7
	DSerial-550	Dual 9-pin serial, Com 1-4, 16550 UARTs, IRQ 2/3/4/5/7/10/11/12/15
	RS422-550	Dual 9-pin serial, 16550 UARTs, RS422 pinout
	LavaPort-ISA	Single 9-pin serial, Com 1-4 16650 UART, IRQ 2/3/4/5/10/11/12/15
	LavaPort-PnP	Single 9-pin serial, 16650 UART, plug and play

Parallel Boards

PCI	Parallel-PCI	Single EPP parallel
	Parallel-PCI/LP	Single EPP parallel, low profile
	Dual Parallel-PCI	Dual EPP parallel
ISA	Parallel Bi-directional	Single bi-directional parallel port, LPT 1/2/3, IRQ 5/7
	Parallel-ECP/EPP	Single ECP/EPP parallel, LPT 1-6, IRQ 2/3/4/5/7/10/11/12

Combo Boards

PCI	SP-PCI	Single 9-pin serial, 16550 UART + single bi-directional parallel
	2SP-PCI	Dual serial (9 & 25-pin), 16550 UARTs + single EPP parallel
	LavaPort-Plus	Dual serial (9 & 25 pin), 16650 UARTs + single EPP parallel
ISA	2SP-550	Dual 9-pin serial, Com 1-4, 16550 UARTs + single bi-dir. parallel, LPT 1-2

USB 2.0 & 1.1 Devices

USB 2.0 Host Adapter	Dual USB 2.0 ports, 480 Mbps, fits in PCI slot
Kazan	Hard drive enclosure with USB 2.0-to-IDE interface
USB 1.1 Host Adapter	Dual USB 1.1 ports, 12 Mbps, fits in PCI slot
SPH-USB 1.1 Hub	Three powered USB ports, parallel port, serial port, connects to USB

IEEE 1394 (FireWire®) Devices

IEEE 1394 FireHost	Dual IEEE 1394 ports, 400 Mbps, fits in PCI slot
FireDrive®	Hard drive enclosure with FireWire®-to-IDE interface
IEEE 1394/IDE Controller	FireWire®-to-IDE hard drive interface

Specialty Boards

PCI	8255-PIO	8255 PIO interface card

2 Vulcan Street
Toronto, ON
Canada
M9W 1L2

TEL: 416.674.5942
FAX: 416.674.8262
www.lavalink.com

LIFETIME WARRANTY