USART, UART, RS232, USB, SPI, I2C, TTL, etc. what are all of these and how do they relate to each other?

As a novice electronics hobbyist, I have heard these terms and more being thrown around everywhere. At its root, I understand that they are all based on communication between devices, computers, peripherals, etc.

I have a basic understanding of how all of them work, but I get confused when I see so many of them and am having difficulty understanding how they relate to each other. For example, is UART a subset of USART? What is the difference between RS232 and Serial? What are the core differences between all of these communication methods: reliability, cost, application, speed, hardware requirements?

If you can imagine, I have all of these terms written on cards, scattered on the coffee table, and I need someone to just help me organize my understanding. Forgive me if this question is a little vague, but I really feel that is the nature of this question all together.

communication

edited Feb 19 '13 at 13:18

Camil Staps
7,878 9 37 89

asked Aug 13 '12 at 18:14

capcom
2,129 6 28 51

related question: Difference between UART and RS232? – Nick Alexeev ♦ Feb 7 at 20:42

4 Answers

Serial is an umbrella word for all that is "Time Division Multiplexed", to use an expensive term. It means that the data is sent spread over time, most often one single bit after another. All the protocols you're naming are serial protocols.

UART, for Universal Asynchronous Receiver Transmitter, is one of the most used serial protocols. It's almost as old as I am, and very simple. Most controllers have a hardware UART on board. It uses a single data line for transmitting and one for receiving data. Most often 8-bit data is transferred, as follows: 1 start bit(low level), 8 data bits and 1 stop bit(high level). The low level start bit and high level stop bit mean that there's always a high to low transition to start the communication. That's what describes UART. No voltage level, so you can have it at 3.3 V or 5 V, whichever your microcontroller uses. Note that the microcontrollers which want to communicate via UART have to agree on the transmission speed, the bit-rate, as they only have the start bit's falling edge to synchronize. That's called asynchronous communication.

For long distance communication (That doesn't have to be hundreds of meters) the 5 V UART is not very reliable, that's why it's converted to a higher voltage, typically +12 V for a "0" and -12 V for a "1". The data format remains the same. Then you have **RS-232** (which you actually should call EIA-232, but nobody does.)

The timing dependency is one of the big drawbacks of UART, and the solution is **USART**, for Universal Synchronous/Asynchronous Receiver Transmitter. This can do UART, but also a synchronous protocol. In synchronous there's not only data, but also a clock transmitted. With each bit a clock pulse tells the receiver it should latch that bit. Synchronous protocols either need a higher bandwidth, like in the case of Manchester encoding, or an extra wire for the clock, like SPI and I2C.

SPI (Serial Peripheral Interface) is another very simple serial protocol. A master sends a clock signal, and upon each clock pulse it shifts one bit out to the slave, and one bit in, coming from the slave. Signal names are therefore SCK for clock, MOSI for Master Out Slave In, and MISO for Master In Slave Out. By using SS (Slave

Select) signals the master can control more than 1 slave on the bus. There are two ways to connect multiple slave devices to one master, one is mentioned above i.e. using slave select, and other is daisy chaining, it uses less hardware pins(select lines), but software gets complicated.

I2C (Inter-Integrated Circuit, pronounced "I squared C") is also a synchronous protocol, and it's the first we see which has some "intelligence" in it; the other ones dumbly shifted bits in and out, and that was that. I2C uses only 2 wires, one for the clock (SCL) and one for the data (SDA). That means that master and slave send data over the same wire, again controlled by the master who creates the clock signal. I2C doesn't use separate Slave Selects to select a particular device, but has addressing. The first byte sent by the master holds a 7 bit address (so that you can use 127 devices on the bus) and a read/write bit, indicating whether the next byte(s) will also come from the master of should come from the slave. After each byte receiver must send a "0" to acknowledge the reception of the byte, which the master latches with a 9th clock pulse. If the master wants to write a byte the same process repeats: the master puts bit after bit on the bus and each time gives a clock pulse to signal that the data is ready to be read. If the master wants to receive data it only generates the clock pulses. The slave has to take care that the next bit is ready when the clock pulse is given. This protocol is patented by NXP(formerly Phillips), to save licensing cost, Atmel using the word TWI(2-wire interface) which exactly same as I2C, so any AVR device will not have I2C but it will have TWI.

Two or more signals on the same wire may cause conflicts, and you would have a problem if one device sends a "1" while the other sends a "0". Therefore the bus is wired-OR'd: two resistors pull the bus to a high level, and the devices only send low levels. If they want to send a high level they simply release the bus.

TTL (Transistor Transistor Logic) is not a protocol. It's an older technology for digital logic, but the name is often used to refer to the 5 V supply voltage, often incorrectly referring to what should be called UART.

About each of these you can write a book, and it looks I'm well on my way. This is just a very brief overview, let us know if some things need clarification.

edited Jun 30 '14 at 12:39

Myanju
158 7

answered Aug 13 '12 at 18:51

stevenvh
117k 11 369 578

+1 only on TTL, I'm guessing that the OP means Serial TTL which in my experience describes the UART signals before the RS232 transceiver/bus driver. – kenny Aug 13 '12 at 19:29

2 @Kenny - but there ain't such a thing as "serial TTL". That's UART. What if the voltage is 3.3 V? The TTL only refers to the 5 V. – stevenvh Aug 13 '12 at 20:01

that's what I'm talking about, UART's I/O are often called in my experience TTL Serial. – kenny Aug 13 '12 at 20:46

Amazing response! I will read it a few more times, and let you know if I have questions. Thanks, Steven. – capcom Aug 16 '12 at 2:58

5 A better bet would be to fish for Uart age then:-) – Vaibhav Garg Aug 22 '12 at 2:37

This is very near the territory of something where you would be better reading articles than asking questions for custom responses, but to address one major point which manufacturer's sometimes blur:

There are two basic types of serial interfaces: synchronous and asynchronous.

Synchronous interfaces have the data transmitted with its timing relative to an explicit clock, which is also provided. The classic example of this is SPI, but there are also special forms such as I2S for audio converters, JTAG, FPGA configuration interfaces, etc. Many parallel communications channels are just this idea extended to moving more bits at once. Often but not always these send the most significant bit first.

Asynchronous interfaces have the timing encoded in the data stream itself. For "serial ports" and related standards such as RS232, the timing of the word is relative to the start bit, and the receiver merely samples the line at the right intervals thereafter. Other interfaces can be a bit more complicated and require fancier clock recovery using phase locked loops and algorithms. A UART is a "Universal Asynchronous Receiver Transmitter" - really the name for a functional block often used to implement a "serial port" with some flexiblity to word length, rate, and start/end conditions. Things such as RS232, RS422, etc are standards for

off-board electrical signalling of the data you would get from these - voltage, single ended or differential, if a 1 is high or low, etc. Traditionally UARTs send the least significant bit first.

The "USART" can be a source of legitimate confusion, as it is a sort of hybrid device, a "Unversal Synchronous/Asynchronous Receiver Transmitter" Essentially, this is and is most commonly used as a UART, but it can also be configured to generate (or consider) a separate clock synchronized to the data, and may be able to reverse the bit order. It is usually configurable to interoperate with SPI, but it may not be able to remove the time dedicated to start/stop bits so may not be able to operate with something like I2S which can expect to have data flowing continuously without gaps between words.

answered Aug 13 '12 at 18:52

Chris Stratton
13.4k 1 17 46

RS-232 is a very simple serial protocol that was originally used for modems and teletypes. It is what is commonly called a serial port (or a COM port in MS-Windows). On the line it nominally uses ±12V levels, but they may vary widely as the detection is specified at ±3V. There is always a line driver (nowadays usually from the MAX232 family) that converts these levels to and from the internal digital signal levels of a computer or microcontroller.

TTL mean Transistor-Transistor-Logic and has its level for logical zero near 0V and for logical one near 5V. Often any 5V logic is called TTL, although most circuits nowadays are built as CMOS. Today there are also many circuits that work at 3.3V, which is no longer TTL.

With respect to the internal levels the levels on the RS-232 line are inverted, +12V corresponds to logical low and -12V corresponds to logical high, which can be confusing.

For describing the data format one usually shows the logical signal. When the line is idle it is high. A transmission starts with a low start bit, the data bits, an optional parity bit and one to two stop bits (logical 1). This is called asynchronous transmission, because the start bit synchronizes the data for each byte separately.

A UART (Universal Asynchronous Receiver Transmitter) is a device in a computer or microcontroller that does this kind of asynchronous communication.

A USART (Universal Asynchronous Synchronous Receiver Transmitter) is a device that can in addition do some kind(s) of synchronous transmission, hence the additional S. Which kind varies, you need to look it up in the data sheet.

SPI, I²C, and USB are different (and in the case of USB very long) stories.

answered Aug 14 '12 at 6:19

starblue
5,277 2 13 24

I agree with what have been mentioned about SPI and CAN protocols. To improve better performance, CAN protocol has been designed. In this Arbitration concept is used in which two devices are ready to communicate, then depending on their priority the transmission or reception takes place. CAN is widely used in many industries.

answered Feb 19 '13 at 10:38

user19166
21 1

---

protected by Community ♦ Nov 18 '14 at 7:30

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?