USART vs UART: Know the difference

Jacob Beningo -September 21, 2015

7 Comments

Have you ever used the term UART only to be corrected by another engineer that it isn't a UART but USART? In certain circumstances the interchangeability of these terms may be appropriate but in many cases it is in error. Let's examine what a USART and a UART are, and discuss the major differences.

Most embedded engineers are familiar with a UART: a Universal Asynchronous Receiver/Transmitter. It is a microcontroller peripheral that converts incoming and outgoing bytes of data into a serial bit stream. A start bit initiates the serial bit stream and a stop bit (or two) completes the data word. A UART also has the option of adding a parity bit to the stream to assist in detecting if a bit error occurs during transmission. Figure 1 shows a standard example of what an engineer would expect to see from data transmitted through a UART.

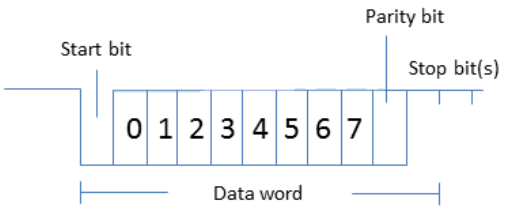

Figure 1 – UART Serial Data

A USART -- a Universal Synchronous/Asynchronous Receiver/Transmitter -- is a microcontroller peripheral that converts incoming and outgoing bytes of data into a serial bit stream. Hmm. The definition of a USART is identical to that of a UART, but with "synchronous" added to the term. Surely there are some more meaningful differences? Otherwise, a USART would just be known as a UART.

Well, there are differences – important ones. The first difference between a USART and a UART is the way in which the serial data may be clocked. A UART generates its data clock internally to the microcontroller and synchronizes that clock with the data stream by using the start bit transition. There is no incoming clock signal that is associated with the data, so in order to properly receive the data stream the receiver needs to know ahead of time what the baud rate should be.

A USART, on the other hand, can be set up to run in synchronous mode. In this mode the sending peripheral will generate a clock that the receiving peripheral can recover from the data stream without knowing the baud rate ahead of time. Alternatively, the link will use a completely separate line to carry the clock signal. The use of the external clock allows the data rate of the USART to be much higher than that of a standard UART, reaching up to rates of 4 Mbps.

The second major difference between a USART and a UART is the number of protocols the peripheral can support. A UART is simple and only offers a few options from its base format, such as number of stop bits and even or odd parity. A USART is more complex and can generate data in a form corresponding to many different standard protocols such as IrDA, LIN, Smart Card, Driver Enable for **RS-485** interfaces, and Modbus, to name a few. A USART also has the same asynchronous capabilities as a UART, that is, a USART can generate the same type of serial data as seen in Figure 1.

USART and UART peripherals have definitely different capabilities and can be useful in different situations, so a developer may find both peripherals onboard a standard microcontroller. For example, take a microcontroller that is targeting low-power design such as the STM32 family. The STM32 parts have both a USART and a UART peripheral on-chip. The USART is meant to do all of the "heavy lifting" serial communication during periods of “high” energy consumption. When the microcontroller is asleep and in a low power mode, though, the UART peripheral can handle low speed communications while offering a reduced energy footprint.

Are USARTs and UARTs the same? Technically the answer is no. A USART generally has more capabilities that a standard UART and the ability to generate clocked data allows the USART to operate at baud rates well beyond a UART's capabilities. A USART does encompass the capabilities of a UART, though, and in many applications, despite having the power of a USART, developers use them as simple UARTs, ignoring or avoiding the synchronous clock generation capability of these powerful peripherals. No wonder so many people use the terms as though they were synonyms.

Also see:

- **Isolated USB-to-UART converter builds in 20 minutes for $20**
- **5 Tips for driver design**
- **Introduction to direct memory access**

*Jacob Beningo is a Certified Software Development Professional (CSDP) whose expertise is in embedded software. He works with companies to decrease costs and time to market while maintaining a quality and robust product. Feel free to contact him at jacob@beningo.com, at his website www.beningo.com, and sign-up for his monthly Embedded Bytes Newsletter **here**.*

7 Comments

Write a Comment

Submit Comment

To comment
please Log In

Brett_cgb

I have problems with a few of the protocols attributed to Synchronous (USART) operations - at their heart, LIN, IrDA, RS-485/422 are still asynchronous protocols supported by a UART with some extra operational and hardware support. These include determining baud rate (LIN), shortening transmitted pulses (IrDA), and differential drive (RS-485/422). None of these depend on transmitting or encoding a clock into the data.

Protocols that use USARTs include SPI, I2C, and Manchester encoding. In these, the clock IS transmitted separately or encoded into the data.

When discussing UART vs USART, the first question is "How is the clock transmitted?" If the clock is not transmitted, then use a UART, otherwise, use a USART.

Sept 28, 2015 2:13 PM EDT

1 | 0 Reply

mjaa54

Baud rate corresponds to the transmitted signal i.e. cycles per sec. Bit rate is governed by baud rate and the bit encoding. Depending on the encoding used e.g. FSK, QAM, PAM etc more than 1 bit bit can encoded and transmitted in a single cycle of the transmission signal. Novel combinations of amplitude, frequency and phase encoding of the transmission signal can result (as Mr Knudsen correctly stated) in a bit speed increase over the baud rate by a factor of 2, 4 or 8. Baud rate and bit rate are never the same thing but use similar numbers to quantify each. Mercifully most readers are too young to have been involved in early data comms when the baud rate and bit rate were identical in number (but never identical in function). Hint: If you do not need to understand the details of the transmission signal encoding then talk only of bit speed (bps). Enjoy!

Sept 25, 2015 7:28 AM EDT

0 | 0

Reply

ShaneBK

It is a bit like the difference between BAUD and bps. Unless you have coded by bit twiddling, or scoped it, you may not know the difference. One teacher (Tallaght Institute of Technology) that I know, taught that with RS232, a high voltage represented a 1, a low, a 0. He also completely ignored start, stop, parity, number of databits, inter-frame spaces, and handshaking.

Sept 24, 2015 1:06 PM EDT

0 | 0

Reply

RAMA2

Asynchronous data transmission uses line transitions for start , data,parity and stop.
Synchronous transmission starts with sending a ascii character for synchronisation -"SYN" control character - which is recognised as the start of a stream and the end of the stream is recognised by the SYN character sent before the next byte stream starts.
Asynchronous transmission is chaaracter by character transmission and so wastes bandwidth because each character is sensed by the start and stop bits.....whereas in synchronous transmission 3 bytes of SYN will be the start and then a long stream can be sent .
Clock recovery as pointed in para 5 is used in modem transmissions where QPSK,QAM,ADPCM techniques are used.

Sept 23, 2015 3:02 PM EDT

0 | 0

Reply

Bornholm

Very informative article.
Just to avoid confusion: The MODBUS protocol is an application layer protocol (pure software), independent of the physical media. It can run on a synchronous line, using HDLC for example, and it can also run on a asynchronous line.

Cheers,
Claus Knudsen

PS. Another area of mix-up is bit/s versus Baud! For example 19200b/s can easily run through a pair of old fashioned modems and an analog telephone line using 2400 Baud between the modems.
Baud is events per second and one event can hold information of many bits. In this example one event holds information of 8 bits. :-)

Sept 23, 2015 1:50 PM EDT

0 | 0 Reply

DavidS775

I believe your definition of Baud is mistaken. Baud is symbols per second. Symbols include start bit, stop bit, and parity bits. See **https://en.wikipedia.org/wiki/Baud.**

Sept 23, 2015 6:14 PM EDT

0 | 0 Reply

EB2EB2

From the Wikipedia definition of Symbol rate: "A theoretical definition of a symbol is a waveform, a state or a significant condition of the communication channel that persists for a fixed period of time."
Symbols are independent of start, stop, and parity, which which provide synchronization control.

Jun 29, 2016 1:28 PM EDT

0 | 0 Reply

MOST RECENT

Prototype to production: Connecting to the cloud

7 tips for optimizing embedded software

5 embedded system characteristics to keep an eye on

Prototype to production: Inputs, outputs and analog measurements

Prototype to production: An industrial HVAC monitor using shields

Prototype to production: Hello World

Answering the build-or-buy firmware conundrum

5 firmware criteria in MCU selection

Prototype to production: Using industrial sensors

Understand firmware's total cost

Most Popular | Most Commented

That 60W-equivalent LED: What you don't know, and what no one will tell you…

The Picard Loop

6 famous people you may not know are engineers

10 tips for a successful engineering resume

Electronic products from hell

Why does this year have an extra second?

Da Vinci unsuccessfully tests a flying machine, January 3, 1496

The 5 greatest engineers of all time

Talking about Design Ideas

ICs of the 1930s & 1940s