USART vs UART: Know the difference

Jacob Beningo - September 21, 2015

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](image)

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.