The Secrets of Flow Control in Serial Communication

Casper Yang, Senior Product Manager

support@moxa.com

Although RS-232/422/485 serial communication is no longer considered to be "high speed," flow control is still an important function for many applications. Most people know what flow control is, but do not know how it works or how it influences the behavior of a communications system. In this paper, we describe flow control in detail, and give you the knowhow needed to fix certain types of communication problems.

High/Low Water Level

Whenever a buffer is used to store data as it is being transmitted, some type of flow control is required to prevent data overrun. Flow control is used in UART FIFOs, drivers, and OS kernels. To implement flow control, the engineer needs to configure both the receive buffer high water level (RBH) and receive buffer low water level (RBL). RBH prevents the receive buffer from overrunning by telling the transmitting side to stop sending data before the Rx buffer is full. For example, if the receive buffer can store 4 KB of data, then you might want to set the RBH value to 3 KB since data will continue to be sent until the transmitting side receives the stop command. That is, incoming data will continue to arrive for a little while even after you say "stop." RBL, on the other hand, defines when data transmission should be resumed. When in doubt, just set RBL to zero, since doing so will ensure that the buffer is completely empty before transmission resumes. However, setting RBL to zero is not very efficient since it could introduce an undesirably large time lag.

 Released on Sep 30, 2009

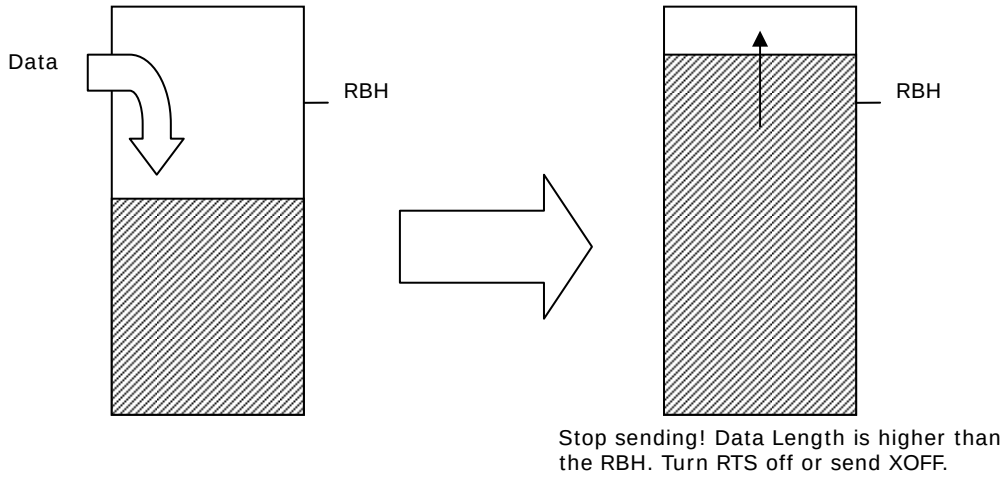

Fig. 1: How the Receive Buffer's RBH Value Controls Data Transmission

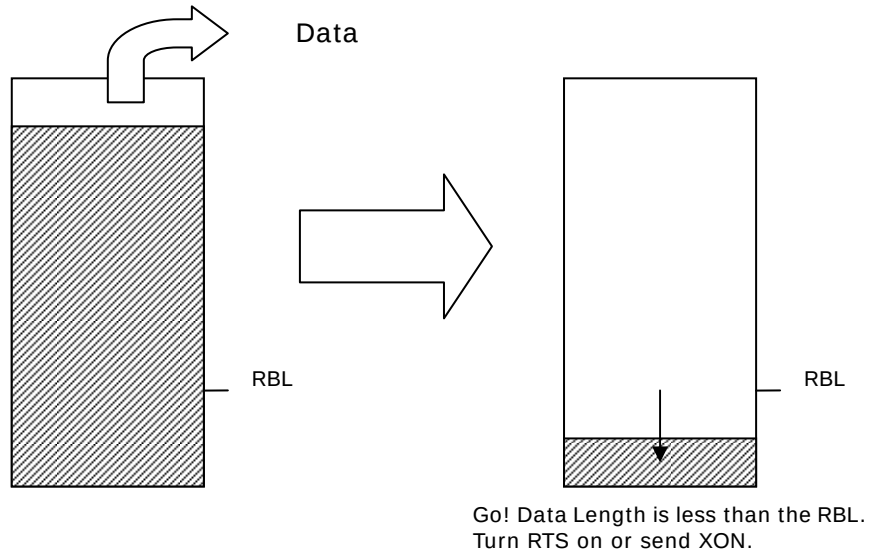

Fig. 2: How the Receive Buffer's RBL Value Controls Data Transmission

RTS/CTS vs. XON/XOFF Flow Control

In serial communication, there are two types of flow control—hardware flow control and software flow control. Hardware flow control uses the RTS/CTS (or DTR/DSR) signals to communicate. When connecting a serial device to a computer, connect the CTS pins to the RTS pins (for DTR/DSR flow control, DSR connects to DTR). The transmitting side checks the state of its CTS pin before transmitting, and then

starts transmitting if it sees “CTS On,” and does not transmit (or stops transmitting) if it sees “CTS Off.” On the other hand, the receiving side uses RTS to say “go” or “stop.” Turning RTS from off to on indicates that the amount of data in the Rx buffer is lower than the RBL, and turning it from on to off means the length is higher than the RBH.

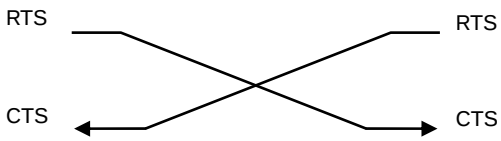

Fig. 3: RTS/CTS Flow Control Cable Wiring

Software flow control uses the Tx and Rx signals to send XOFF and XON control characters. The receiving side sends an XOFF character over its Tx line (which is equivalent to turning RTS from on to off) to tell the transmitting side to stop transmitting, and sends an XON character over its Tx line (which is equivalent to turning RTS from off to on) to tell the transmitting side to resume transmitting. Most serial devices use 0x11 for XON and 0x13 for XOFF by default, although in most cases the values can be set to other values. If your data includes either of the XON/XOFF characters, which could be true for binary execution files, you will need to use other characters instead.

Serial communication requires connecting at least the Tx, Rx, and GND pins, but when using hardware flow control, you also need to connect the RTS and CTS pins. Although this will increase the cost of the cable, hardware flow control is more reliable. If cost is an issue you can use software flow control, provided your data does not conflict with the XON/XOFF characters. You should also keep in mind that the serial driver needs to check each byte of data one by one for the XON/XOFF characters. Because of the problems associated with software flow control, many advanced UARTs support on-chip software flow control.

RS-422/485 Considerations

RS-422/485 uses differential signal paths, which increases the maximum transmission distance, but also requires more cabling than RS-232. For this reason, most RS-422/485 connections only provide Tx and Rx signals and do not provide

RTS/CTS or DTR/DSR signals. In this case, using XON/XOFF flow control is a necessity. Note, however, that 2-wire RS-485 data transmission is a bit different since it uses half-duplex transmission. Since data can only be transmitted in one direction at a time, flow control is not needed. If you are using 4-wire RS-422/485 communication and you want to use RTS/CTS flow control, make sure that both the host and serial device support this function. In fact, most serial solutions (including PCI boards and devices) do not support the RTS/CTS pins when using 4-wire RS-422/485.

XON/XOFF Needs More Latency

Before discussing on-chip flow control, let's look at how a driver implements the flow control function. Hardware flow control uses the CTS and RTS signals. The driver checks the signal on its CTS pin before transmitting, and set its RTS signal to ON when it's ready to receive. Adjusting the RTS signal and reading the CTS state is very easy and fast, and if cost is not an issue, then using hardware flow control will ensure greater reliability.

For software flow control, the driver needs to check each incoming byte of data to see if an XON or XOFF character has been received. Doing this wastes time and is not reliable since before the XOFF character is received the driver may still need to read out data from the UART. This latency will cause the driver to send more data and may cause a data overrun on the receiving side.

Fig. 4: Including XOFF in the data stream could result in too much of a delay.

On-chip Flow Control Lets You Stop Transmitting in Time

Many advanced UARTs have incorporated on-chip flow control to circumvent the complexities inherent in flow control schemes. "On-chip" means that the UART processes the RTS/CTS signals and XON/XOFF characters automatically. To prevent overrun in the UART FIFO, the UART uses an internal RBH and RBL. When the RBH is reached it drops the RTS or sends an XOFF character immediately. It also stops transmitting immediately when the CTS is low or an XOFF character is received. For slower devices this function can be very useful. This is because slower devices

generally have a smaller buffer, and consequently it is essential to stop transmitting in time when an XOFF character is received.

Even if a UART supports on-chip flow control, using a poorly designed driver could cause data loss. This is because even though the UART might have enough space for additional data, the driver may not. Knowing how to prevent driver buffer overrun and designing the driver to work well with UART on-chip flow control is very important, and depends to a large extent on the experience the vendor has with developing drivers.

To enable flow control, you need to check both software and hardware settings. For software flow control, Windows and UNIX/Linux provide standard interfaces. For more information, please check the Win32/UNIX/Linux programming guide. For hardware flow control, make sure the RTS/CTS or DTR/DSR pins are connected correctly. If you don't need to use flow control, make sure the setting is off. If flow control is enabled without an RTS/CTS cable connection, data cannot be sent out because the CTS state is low.

Solution for UART without On-chip Flow Control

On-chip flow control is extremely important for applications that cannot tolerate data loss or buffer overruns. If you have any doubts about your serial device, check with your vendor to see if the device supports on-chip hardware or on-chip software flow control. If not, disabling the device's FIFO could be more reliable. This is because the driver will only send one byte for each interrupt issued, and give the receiving side more time to process the data. For XON/XOFF flow control, disabling the FIFO will result in better latency, and provide enough time to process the XOFF character and prevent overruns.

Conclusion

Many advanced UARTs support on-chip flow control with high/low water settings in the transmit/receive FIFO. To prevent overrun and get good throughput, choose a UART with on-chip flow control and set the high/low water level properly. In most cases, you can set the high water level to 3/4 the buffer length and set the low water level to 1/4 the buffer length. The next time you experience an overrun or data loss, check the flow control settings first.

	Question	Answer
1	What should I do if I have enabled flow control but still get an overrun or experience data loss?	● Check both sides of the connection and enable flow control properly. ● Check to make sure that RTS connects to CTS (and vice versa) if your are using RTS/CTS flow control. ● If you are using XON/XOFF flow control without on-chip support, try disabling the FIFO. ● Try using UARTs with on-chip flow control on both sides of the connection, and make sure the driver you're using supports this function. ● If you are using UARTs that support on-chip flow control on both sides of the connection, then the problem is probably caused by the driver.
2	Does my UART support on-chip flow control?	Most motherboards provide 1 or 2 UARTs, but they are 16550A and do not support this function. Some advanced UARTs, such as Moxa's MU860 or 16950, support both RTS/CTS and XON/XOFF flow control, but most UARTs, such as the 16550C, only support RTS/CTS. Read your product's datasheet or check with your vendor for details.