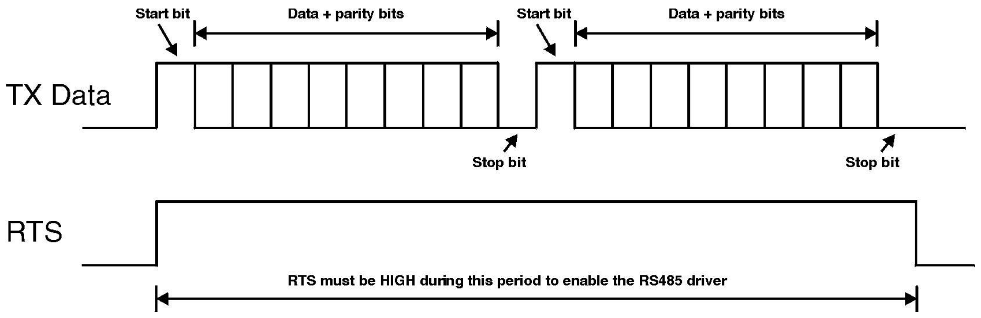
What is RTS Control?

RTS Control is relevant only if you are converting to 2-wire RS485 (where the converter is a Master or a Slave) or to 4-wire RS485 (where the converter is a Slave). It is not required for RS422 which is a point to point system only and on which the driver is permanently enabled.

RTS Control is a method with which the RS232 device (typically a PC) tells an RS232-RS485 converter when it should enable its RS485 driver, i.e. when it should be transmitting. There is no technical reason why the converter cannot determine this by itself but it increases the cost of the converter. It also makes it sensitive to the baud rate and character length (the number of bits) and these therefore need to be configured somehow on the converter.

On a converter which is an interface converter only and does not monitor the data, e.g. the K2, K3, K485-ISOL, KD485-STD, an external signal is required. When providing RTS Control, the RS232 device raises its RTS output immediately before it starts to communicate, and drops it after the last stop bit of the message has been transmitted. The converter uses this signal to control its RS485 driver. The advantages of using RTS Control is that the converter is simpler and therefore cheaper, and it does not care about the baud rate (within its limits) or the number of bits, parity, etc.

The following diagram illustrates a message comprising of two characters and the RTS Control signal which would be required to successfully transmit this message. Both characters are shown as 8-bit data (or 7 bits with parity).



A more sophisticated converter, e.g. the K2-ADE, K3-ADE or KD485-ADE, does not need RTS Control because it generates it internally by monitoring the data with a microprocessor. But you have to configure the baud rate etc on the converter.

The RTS Control function has to be written into the application program and is not an operating system function which you can configure in e.g. the Windows Control Panel. Many RS485-oriented application programs have it, particularly those written for industrial applications. Some do not. The only way to establish if a particular application program provides RTS Control is to ask the programmer who wrote it, or the vendor. If this is not possible, and no reliable information is available, you should assume that RTS Control is not available and choose an "ADE" converter.

Do not confuse RTS Control with the more common operating mode of the RTS signal which is hardware flow control and which is unsuitable for controlling an RS232-RS485 converter.

If you are a software developer: It is a lot easier to get the RTS turn-off timing exact under MS-DOS than under Windows. Under MS-DOS, simply wait for both the TX-buffer-empty and all-sent UART flags to go true and then drop RTS. Under Windows, you can use various timing methods (none of which will be precise) or configure the converter to have its receiver always enabled (so you receive your own transmit data) and when you have received the last character of your transmission, drop RTS. The required RTS turn-off accuracy depends on how fast the slave device responds; if it starts transmitting its response within 1 bit of the end of your transmission then it may be impossible to do this under Windows and an ADE converter will be required. If however it does not start its response for e.g. 10ms then (at 9600 baud) a simple timer should be sufficient. The Windows NT (and higher) comms API offers a "RTS control" function but this is reliable only to within 10ms or so. KK Systems user-programmable products (KD485-PROG and PPC) contains special functions to assist with precise driver turnoff.