

Subject: **Considering USB to Serial port adapters**To: **OriLink® users**From: **Mikael Theorin/MITH****PM**

RS232 to USB converters

- Recommended is always mother board or I/O-card true com port.
- Introduction to USB to RS232 conversion
- Differences at the application level
- Hardware specific problems
- USB to RS232 converter selection criteria

USB to RS232 conversion

One of the most common questions asked is how to get a RS232 port on a PC that only have USB ports.

Although RS232 and USB (universal serial bus) are both serial communication standards aimed to connect peripheral devices to computers, they are totally different in design. A simple cable is not enough to connect RS232 devices to a computer with only USB ports. There are however converter modules and cables that can be successfully used to connect RS232 devices to computers via an USB port. These adapters and cables contain electronics, and the success rate depends on the capabilities of this electronics and the device driver software that is shipped with the converter to communicate with these electronics over the USB bus. Before buying your USB to RS232 converter, it is advised that you read this document first.

Differences from the application point of view

RS232 is a definition for serial communication on a 1:1 base. RS232 defines the interface layer, but not the application layer. To use RS232 in a specific situation, application specific software must be written on devices on both ends of the connecting RS232 cable. The developer is free to define the protocol used to communicate. RS232 ports can be either accessed directly by an application, or via a device driver in the operating system.

USB on the other hand is a bus system which allows more than one peripheral to be connected to a host computer via one USB port. Hubs can be used in the USB chain to extend the cable length and allow for even more devices to connect to the same USB port. The standard not only describes the physical properties of the interface, but also the protocols to be used. Because of the complex USB protocol requirements, communication with USB ports on a computer is always performed via a device driver.

It is easy to see where the problems arise. Developers have lots of freedom where it comes to defining RS232 communications and ports are often directly, or almost directly accessed in the application program. Settings like baud rate, data bits, hardware software flow control can often be changed within the application. The USB interface does not give this flexibility. When however an RS232 port is used via an USB to RS232 converter, this flexibility should be present in some way. Therefore to use an RS232 port via an USB port, a second device driver is necessary which emulates a RS232 UART, but communicates via USB.

Many applications expect a certain timing with RS232 communications. With ports directly fitted in a computer this is most of the time no problem. The application communicates directly, or via a thin device driver layer with the UART, and everything happens within a well defined time frame. The USB bus is

however shared by several devices. Communication congestion may be the result of this, and the timeframe in which specific RS232 actions are performed might not be so well defined as in the direct port approach. Also, the double device driver layer with an RS232 driver working on top of the complex USB driver might add extra overhead to the communications, resulting in delays.

Hardware specific problems

RS232 ports which are physically mounted in a computer are often powered by three power sources: +5 Volt for the UART logic, and -12 Volt and +12 Volt for the output drivers. USB however only provides a +5 Volt power source. Some USB to RS232 converters use integrated DC/DC converters to create the appropriate voltage levels for the RS232 signals, but in very cheap implementations, the +5 Volt voltage is directly used to drive the output. This may sound strange, but many RS232 ports recognize a voltage above 2 Volt as a space signal, where a voltage of 0 Volt or less is recognized as a mark signal. This is not according to the original standard, because in the original RS232 standard, all voltages between -3 Volt and +3 Volt result in an undefined signal state. The well known Maxim MAX232 series of RS232 driver chips have this non-standard behavior for example. Although the outputs of these drivers swings between -10 Volt and +10 Volt, the inputs recognize all signals swinging below 0 Volt and above 2 Volt as valid signals.

This non-standard behavior of RS232 inputs makes it even more difficult to select the right RS232 to USB converter. If you connect and test an RS232 to USB converter over a serial line with another device, it might work with some devices, but not with others. This can particularly become a problem with industrial applications. Low-cost computers are often equipped with cheap RS232 drivers and when you test the RS232 to USB converter with such a computer, it might work. But the same converter may fail if you try it in an industrial environment. The chances that RS232 ports from low-cost computers accept signals in the 0..5 Volt range are higher than with industrial equipment which is often specifically designed to be immune for noise.

Another hardware specific problem arises from handshaking to prevent buffer overflows at the receiver's side. RS232 applications can use two types of handshaking, either with control commands in the data stream, called software flow control, or with physical lines, called hardware flow control. Not all USB to RS232 converters provide these hardware flow control lines. It is not always easily identified if an application needs them. Some applications do not use hardware flow control at all, and those cheap USB to RS232 converters will work without problems. Other applications use hardware flow control, but infrequently. Only with large data bursts, or in situations where the CPU is busy performing other tasks, hardware flow control might kick in to prevent data loss. In those situations, communications may seem error free, but with sometimes bytes lost, or unspecified errors in the communications.

USB to RS232 converter selection criteria

Resuming, when choosing the right USB to RS232 converter, look at the following potential problems:

- A mother board or I/O-card true basic com port is always recommended to get best and secure functionality especially for business critical systems.
- Does your application have very tight timing requirements? In that case it might be better to use an internal RS232 port, instead of an USB to RS232 converter. The extra layer at the device driver level and bus congestion might make the communications less reliable.
- What are the RS232 output voltages of the converter? Do they meet the requirements for the equipment you want to connect?
- What are the handshaking requirements for your application? If hardware flow control is required, make sure that these inputs and outputs on the converter are present.