Microsoft®

Windows 3.1 Serial Communications Questions & Answers

April, 1992

How does Windows 3.1 improve serial communications over Windows 3.0?

Most of the serial communications problems experienced by users of Windows 3.0 are addressed in Windows 3.1. Here is a summary of the changes:

1) *COM port addresses and IRQs are configurable in the Control Panel:* Windows 3.0 used pre-set address values for serial ports. Windows 3.1 now allows any port address & IRQ. This solves problems using ports COM3 and COM4, for which Windows 3.0 used non-standard default values.

2) *Faster baud rates:* Windows applications can now set speeds higher than 19.2 Kbaud, up to 57.6 Kbaud. Contact your Windows software manufacturer for an update to support this new feature.

3) *Reduced system overhead:* Windows 3.1 is fine-tuned for better performance overall, including faster throughput and fewer errors during serial communications. A new driver interface allows Windows to pass data in blocks to the driver, rather than a character at a time. See the Windows 3.1 Device Development Kit for details.

4) *16550 UART Support for Windows applications:* This is described in detail below.

5) *EnableCommNotification API:* Windows applications can register to be notified by a window message when serial port events occur, rather than performing constant polling, which reduces overall system performance.

6) *Serial ports on the same IRQ can be used "in rotation":* Although on most PCs, serial ports that use the same IRQ cannot be used simultaneously, Windows 3.0 sometimes prevented them from being used in rotation, one after another. This is fixed in 3.1.

7) *Full duplex improvements:* Some users experienced lockups during full-duplex transfers at fast baud rates under Windows 3.0. These are corrected in Windows 3.1.

How many serial ports does Windows support?

The Windows 3.1 communications driver provides access to standard MS-DOS serial ports COM1, COM2, COM3, and COM4. Third party serial adapters with more than four ports can be used under Windows with special drivers available from their manufacturer. Windows allows a maximum of nine serial ports (COM1 - COM9).

What base addresses and IRQs does Windows use for serial ports?

Windows must know the *base I/O addresses* and *interrupt request lines* (IRQs) of the serial ports on your PC. It determines these as follows:

BIOS Data Area: Windows looks here (40:0) first for port base addresses. Most PCs specify address values here for built-in COM ports, but not for add-in adapters. If Windows finds an address here, it will use this over any other defaults or settings.

Serial ports commonly use the following base I/O address and IRQ values:

Port	*Address*	*IRQ*
*COM1 **	03F8	4
COM2	02F8	3
COM3	03E8	4
COM3 (PS/2)	3220	3
COM4	02E8	3

* If the BIOS data area specifies COM1 address as 03F8, then Windows uses IRQ3 instead.

Control Panel Ports Advanced Dialog: You can tell Windows the addresses and IRQs of ports which are not specified in the BIOS data area using the Control Panel Ports.Advanced dialog. This sets the COMxBase and COMxIRQ values in the [386Enh] section of SYSTEM.INI. IRQ values 8 - 15 are allowed. Setting the port address in this dialog will not override a BIOS data area value. For more information on COMxBase and other SYSTEM.INI switches, see the file SYSINI.WRI in your Windows directory.

Windows 3.0 would not recognize my COM3 or COM4 ports. I changed my COM3Base and COM4Base settings in SYSTEM.INI, but this didn't help. Is this fixed in Windows 3.1?

Yes! The problem in Windows 3.0 was that the comm driver (comm.drv) and virtual comm device (vcd) weren't using the same values. Now, both modules use the values you specify. But you should no longer modify SYSTEM.INI manually. Instead, use the Control Panel Ports.Advanced dialog. These settings apply to both standard and enhanced modes.

When can I use the same IRQ for different serial ports?

In general, two devices should not be used simultaneously with the same IRQ. For example, if your COM1 and COM3 ports both use IRQ4 (the common value), then you will have problems if you have a serial mouse on COM1 and a modem on COM3.

Some PCs and serial port adapters support "IRQ sharing", the ability to have multiple ports using the same IRQ. This is common on MicroChannel (MCA) bus architecture PCs, such as IBM PS/2s, and EISA bus PCs. Windows 3.1 fully supports IRQ sharing serial adapters. To use them properly, be sure that you have set the IRQ values correctly using the Control Panel Ports.Advanced dialog. Also, if you do not have an MCA or EISA PC, you must set "COMIrqSharing=TRUE" in the [386Enh] section of your SYSTEM.INI file. For more information on COMIrqSharing and other SYSTEM.INI switches, see the file SYSINI.WRI in your Windows directory.

I'm having problems using COM4. What's wrong?

Use the Control Panel Ports.Advanced dialog to make sure that Windows knows the correct port address value and IRQ settings for your COM4 port. The address for COM4 is not usually specified in the BIOS data area of most PCs, since it is not a standard port.

Many serial adapters use a COM4 default base address of 02E8. Unfortunately, this address value conflicts with some peripherals, including 8514/A video adapters and certain network adapters. To workaround this problem, reconfigure one of your peripheral adapters to use a different address. If you change your COM4 address, be sure to use the Control Panel

Ports.Advanced dialog to reset the address value for COM4.

What is "device contention"?

As a multitasking operating system, Windows must handle simultaneous requests for devices, such as comm ports, from different applications. "Device contention" simply refers to the rules Windows uses to determine how applications receive control of a device. The 386 Enhanced dialog of the Control Panel allows you to specify how COM ports are to be allocated to applications.

"Always Warn" means that Windows will present a dialog asking whether an MS-DOS application should receive control of a COM port that it detects may be currently in use by another application.

"Never Warn" means that Windows will always provide COM port access to an MS-DOS application, even if it detects that this port may currently be in use by another application.

"Idle (in sec.)" means that Windows will present a dialog asking whether an MS-DOS application should receive control of a COM port that it detects was in use as recently as the number of seconds specified. Otherwise, Windows will provide COM port access without a warning.

Under Windows 3.0, I could not use different serial ports on the same IRQ "in rotation". Is this fixed in Windows 3.1?

Yes! You can use serial ports that use the same IRQ (e.g. COM1/3 on IRQ4 and COM2/4 on IRQ3) one after the other, but not at the same time unless your serial adapter supports IRQ sharing. However, some MS-DOS comm applications may leave a serial port in a state that prevents Windows from using its IRQ for other ports. You can reset the "stuck" comm IRQ by opening and closing the original comm port with a Windows comm application, such as Terminal.

What is the maximum possible baud rate for Windows applications?

Windows 3.1 allows applications to select baud rates as high as 57.6 Kbaud. However, how well you can transmit and receive data at high speeds depends on many variables, discussed below. With Windows 3.1 in a reasonably busy system, a 19.2 Kbaud XModem transfer can run well on a 20Mhz 386.

Factors affecting top baud rate in Windows applications include:

1) *CPU Speed:* Faster CPUs will allow faster data throughput with fewer errors. This is because they allow the comm driver to service character interrupts faster (preventing UART overflow), and allow applications to handle transfer protocols better.

   If you are experiencing slow serial performance or are losing characters on a 286 based PC, try the following to correct the problem:

   * Reduce your number of MS-DOS drivers and terminate-and-stay-resident programs.
   * Do not load MS-DOS 5.0 into upper memory.
   * Add the following line to the SYSTEM.INI file: **FasterModeSwitch=1**. For more

information on FasterModeSwitch and other SYSTEM.INI file switches, see the file SYSINI.WRI in your Windows directory.

2) *Transfer Protocols:* Data transmission protocols (e.g. XMODEM, YMODEM, ZMODEM, etc.) vary greatly in robustness in a multitasking environment. Some are designed to be very fast, with little error checking, while others trade throughput for error correction. Usually, XModem and ZModem work well with Windows, even at high speeds.

3) *Serial Port Hardware:* Newer serial ports use the 16550 UART, which contains a buffer to reduce overhead and prevent overflow. See additional discussion on this below.

4) *System Overhead:* The more applications, TSRs, and device drivers that are running, the busier the operating system is. This reduces overall execution time for everybody in the system. If you have many programs and devices active while trying to transmit data at high speeds, you are likely to encounter reduced throughput and transmission errors.

5) *Application Quality:* Like most other software, not all serial communications applications are created equal! These vary in quality and performance, so comparison shopping is valuable before purchasing. In general, choose a program that is well-tested and supported by an experienced vendor.

What is the maximum possible baud rate for MS-DOS applications running in Windows?

MS-DOS serial communications applications are designed to run in a non-multitasking environment, and make assumptions about the operating system and devices that are not always true in a multitasking operating system. These applications may allow you to select speeds as high as 115 Kbaud and beyond. While these speeds may work very well in MS-DOS alone, you should expect reduced data throughput when running these applications in a multitasking operating system such as Windows or OS/2.

The guidelines listed above for Windows serial communications applications also apply to MS-DOS based applications. However, the following guidelines are important for MS-DOS applications running within Windows enhanced mode:

1) *Full-screen vs. Windowed vs. Minimized:* MS-DOS applications run best in full-screen mode. They receive their foreground priority and avoid the overhead of display virtualization. Although MS-DOS applications running in a foreground window receive their foreground execution priority, the overhead of updating the window can seriously reduce application performance. If you wish to run an MS-DOS comm application in the background during a data transfer, run it minimized as an icon rather than in a window.

2) *Foreground & Background Execution Priority:* Windows allocates execution time for MS-DOS applications according to foreground and background priorities that you supply in the PIF file. If your application is not getting the timeslices it needs to process incoming data, you will encounter errors during data transfer. Increase your MS-DOS application's background execution priority in the PIF file to prevent errors when receiving data in the background. And of course, make sure that you specify "Background" execution in the PIF file!

3) *"Lock Application Memory":* In order to allow more MS-DOS applications to run simultaneously, Windows 3.1 enhanced mode swaps application memory to disk. However, this can slow the execution of the MS-DOS application, and significantly reduce comm

performance. Disable this swapping of application memory in your MS-DOS comm application's PIF file by checking the "Lock Application Memory" checkbox in the Advanced Options dialog of the PIF Editor. Note that this PIF setting only applies if you have a permanent swap file and 32-bit disk access is enabled.

Does Windows 3.1 support the 16550 UART buffer?

Many serial port adapters and add-in modems are now using the 16550 Universal Asynchronous Receiver Transmitter (UART), which contains a 16 byte character buffer to reduce interrupt overhead and errors during high-speed serial transmissions. The Windows 3.1 comm driver enables this buffer for Windows serial communications applications in order to reduce interrupt overhead and improve serial throughput at high-speeds. Windows applications do not need to call a special API to receive this support, it is automatically enabled. You can find out if your serial ports use the 16550 UART by running the Microsoft Diagnostics (MSD.EXE) outside of Windows 3.1.

Windows 3.1 does not enable the 16550 buffer for MS-DOS comm applications. When running in Windows enhanced mode, 16550-aware MS-DOS comm applications may fail to detect and enable the UART buffer. This is because Windows is doing its own buffering of the comm port for the application. To allow 16550-aware MS-DOS comm applications to enable the FIFO, you must disable Windows' buffering of a comm port by adding "COMxBuffer=0" in the [386Enh] section of SYSTEM.INI, where x= the ID of the comm port (for example, "COM1Buffer=0"). This may improve serial comm performance under certain conditions.

Some older 16550 UART versions do not properly support the buffer and may cause problems. Windows can detect many of these and will not enable the buffer. However, if you encounter problems, you may need to manually disable the 16550 support by adding "COMxFIFO=FALSE" to the [386Enh] section of SYSTEM.INI. For more information on COMxFIFO and other SYSTEM.INI switches, see the file SYSINI.WRI in your Windows directory.

My Windows communications application replaced the Windows 3.0 driver. Should I continue to use this driver in Windows 3.1?

You are likely to find that the Windows 3.1 comm driver provides better support than some replacement comm drivers developed for Windows 3.0. However, if your replacement comm driver provides special device support or features specific to the application, you should continue to use this with Windows 3.1. If you are not using a version 3.1 comm driver, then you must add the following line to the [386Enh] section of SYSTEM.INI: "COMdrv30=TRUE". For more information on COMdrv30 and other SYSTEM.INI switches, see the file SYSINI.WRI in your Windows directory.

I use TurboCom with Windows 3.0. Should I continue to use this with Windows 3.1?

Windows 3.1 corrects most of the problems that TurboCom was developed to alleviate under Windows 3.0, and provides additional communications features. Version 3.1 also provides Windows application support for the 16550 buffer, which TurboCom added under Windows 3.0.

Unlike TurboCom, Windows 3.1 does not enable the 16550 buffer for MS-DOS communications applications. However, if you follow the guidelines above for fast baud rates, these applications may not require TurboCom under Windows 3.1. Also, if you use TurboCom to run your Windows applications at baud rates higher than they provide options to select, you may want to continue using TurboCom until your application is updated to support higher baud rates under Windows 3.1.

In general, if the Windows 3.1 communications driver meets your needs, then you should use it instead of any third party drivers in order to acheive full Windows 3.1 compatibility.