About your SINGLE USER LICENSE of TurboCom(TM)

TurboCom is a commercial product developed by John Loram at Bio-Engineering Research Laboratories, Berkeley, CA.

TurboCom was developed on a speculative basis in response to the perceived need for improved Asynchronous communications capabilities under Microsoft Windows 3.

The Single User License, printed on the envelope in which TurboCom is packaged, grants specific rights. Please take a few minutes to read and understand the extend of these grants. We have tried to make the elements of the Single User License as unencumbering as possible while protecting the substantial effort that has been made to provide a useful product.

TurboCom is not a shareware product. Please do not share it with your friends and associates, even on a trial basis. You may of course test TurboCom on any system and as many systems as you wish, so long as the Single User License agreement is not violated.

A reproduction of the TurboCom Single User License is provide at the end of this document.

TurboCom is available under Single User licenses, Site Licenses, and Licenses for bundling with other software and hardware products.

Quantity discounts are available for single user and site licenses. For further information contact John Loram at:

Bio-Engineering Research Laboratories
2831 Seventh St.
Berkeley, California 94710

(415) 540-8080

Basic Installation

Installing TurboCom is a five step process. It may be accomplished while in DOS or Windows.

1) Copy the three files named **TURBOCOM.DRV**, **TURBOVCD.386**, and **TURBOBUF.386** from the TurboCom distribution disk into the **\WINDOWS\SYSTEM** sub-directory. If your system is based on the 80286 processor you need only copy **TURBOCOM.DRV**.

2) Copy the file named **TURBOCOM.SYS** from the TurboCom distribution disk to the directory in which you keep your DOS **.SYS** files. On many systems this would be the root directory of the boot volume (e.g. C:\). **TURBOCOM.SYS** is not required for IBM PS/2 Microchannel models 50 and above.

3) Append the contents of **TURBOCOM.INI** from the TurboCom distribution disk to your Windows **SYSTEM.INI** file, which you will find in your main **\WINDOWS** sub-directory. Make sure that **TURBOCOM.INI** is appended to the **SYSTEM.INI** file, not to one of the other **.INI** files located in this sub-directory.

4) Find and modify the following profile strings in your **SYSTEM.INI** file:

In the [boot] section of **SYSTEM.INI**:

change profile string **comm.drv=comm.drv** to read **comm.drv=TurboCom.drv**

In the [386enh] section of **SYSTEM.INI** make two changes (unnecessary for 286 machines):

change profile string **device=*vcd** to read **device=TurboVcd.386**

change **device=*combuff** to read **device=TurboBuf.386**

5) Add the following line to your **CONFIG.SYS** file:

DEVICE=TURBOCOM.SYS

This line must appear in the **CONFIG.SYS** file before any line which loads a driver involving Com ports such as a serial port mouse driver, or a serial port network driver. It can be safely placed as the first device driver in the **CONFIG.SYS** file. **TURBOCOM.SYS** is not required for IBM PS/2 Microchannel models 50 and above.

Now re-boot DOS and Windows so that the new drivers will be loaded.

How Does TurboCom Work

TurboCom works on three levels to greatly improved Windows and DOS asynchronous communications performance.

Level 1 Improvements - Comm Port hangups and COM3/4 availability:
TurboCom.sys is a DOS device driver that extends the boot-time system initialization functions of the ROM BIOS of your ISA based system. Few ISA ROM BIOSes make provisions to initialize COM3 and COM4 hardware and the associated ROM BIOS data area where comm port base addresses are stored. No ROM BIOS for ISA based system properly initialize the 16550 UART. The result of these limitations is that many systems cold boot with their serial port hardware incompletely initialized, and warm boot with their UARTs in unpredictable, often inaccessible states. TurboCom.sys performs proper hardware initialization of 8250, 16450, and 16550 UARTs, and builds a complete UART data table in the ROM BIOS data area of system RAM. It then reports the results of its efforts and discoveries and exits without taking up system RAM.

Level 2 Improvements - Windows Comm Port bugs and 16550 UART support:
TurboCom.drv and TurboVCD.386 comprise the two basic Windows Comm Drivers. TurboCom.drv is the Windows 3 Comm Driver. It provides the application program interface (API) for Windows applications which use the serial and parallel ports, and the interrupt level routines for Real and Standard mode operation.

Two bugs, one dealing with comm port hangups during high-speed full duplex operations and another causing IRQ conflicts were corrected. The interrupt level code was restructured to make it more efficient standard UARTs, and take better advantage of the high performance hardware features of the 16550 UART.

TurboCom brings new levels of performance to Windows Asynchronous port usage by providing support for the hardware buffers in the 16550 high performance UART. The sixteen byte transmit and receive buffers of the 16550 UART make it highly resistant to the often poor interrupt response time of multi-tasking environments like Windows. In addition, with the 16550 UART's ability to collect multiple characters before issuing an interrupt, the receive and transmit interrupt overhead can be reduced by as much as an order of magnitude.

Level 3 Improvements - DOS Comm support and 16550 access:
TurboBuff.386 provides a RAM based UART buffer for DOS applications running under enhanced mode Windows. TurboBuf collects characters from the UART into a system RAM buffer while a DOS application is switched out. When the application is switch in during its time-slice, TurboBuf provides the collected characters to the DOS application by simulating the UART, interrupts and all.

Performance Tuning TurboCom for Windows and DOS Applications

TurboCom provides a number of performance modifying parameters which can be placed in the TurboCom section of the SYSTEM.INI file. Some of them are quite esoteric and will not need to be change in most systems. Others, will likely be changed by every user.

First, a little about the purpose and structure of your SYSTEM.INI file.
The SYSTEM.INI file is similar in function to the DOS CONFIG.SYS file; it tells Windows which Windows device drivers to load, and it allows those device drivers access to user supplied configuration information.

The SYSTEM.INI file's contents are composed of sections in the format of [SectionName] (note the square brackets) and profile strings in the format of Profile=String (note the equal sign).

Profile=Strings are always associated with the specific [SectionName] which proceed them. Should another [SectionName] inadvertently be placed between a Profile=String and its associated [SectionName], the association is broken and strange unpredictable events will almost certainly occur, often at boot time but sometimes much later.
Make a copy of your SYSTEM.INI file before you change it and keep the copy in a separated directory.

Identifying the Communications Device

COMxDeviceType=n, default n = 16, legal values = 0, 1, 2, or 16

If your system does not exclusively use 16550 style UARTs, you will want to change one or more of the COMxDeviceType profile strings to reflect your actual hardware configuration.

TurboCom does a good job of identifying which kind of communication devices (e.g., UARTs) are available in your system. However, should you wish to turn off the use of the advanced features of whichever device has been detected, you can. TurboCom will override inappropriate choices and tell you about it. To force device identification, change the following lines(s) of text in your SYSTEM.INI file, in the [TurboCom] section.

COMxDeviceType=n

Where x = the Com port number 1, 2, 3 or 4, and n is the identifier for the communications device:

0 = unknown,
1 = 8250 / 8250-B style UARTs,
2 = 8250A / 16450 style UARTs,
16 = 16550 style UARTs**Identifying the Communications Device** (cont.)

For example;

COM1DeviceType=16 tells TurboCom that Com port 1 is a 16550 style UART.

COM3DeviceType=0 tells TurboCom to treat the Com3 device as the simplest of communication devices, which is currently the same as choice #1, the 8250 / 8250-B.

Setting High Baud Rates for Windows Applications

If you wish to use baud rates above the 19,200 limit imposed by the standard Windows Comm driver, the COMxBaudX profile string allows you to do so.

COMxBaudX=n, default n = 1, legal values = 1, 2, or 3:

Most commercial Windows applications only provide for baud rate settings of 19,200 baud and below. This is because the standard Windows Comm driver rejects any attempt to set a higher rate.

Since TurboCom performs well at 57,600 baud in fast machines, you can multiply the nominal baud rate by a factor of two (2) or three (3) on a per channel basis.

To multiply the baud rate set by your Windows application, change the following line(s) of text in your SYSTEM.INI file, in the [TurboCom] section.

COMxBaudX=n

Where x = the Com port number 1, 2, 3, or 4, and n = the baud rate multiplier 1, 2, or 3.

For example;

COM2BaudX=3 will multiply an application set baud rate of 19,200 baud to an actual 57,600 baud for Com2.

COM4BaudX=2 will multiply an application set baud rate of 19,200 baud to an actual 38,400 baud for Com4.

TurboCom will reject attempts to set rates above 19,200 baud on ports that do not have high performance hardware (e.g., 16550 style UARTs).

The TurboCom API allows a Windows application to directly set baud rates up to 57,600 baud if the appropriate UART is available.

Setting the Receiver FIFO Interrupt Trigger Level for Windows and DOS Applications

Note: This is really high tech stuff, and is here mostly for the un-reconstructed hacker. Unless you really enjoy tinkering, there is probably no reason to change the Receive FIFO Interrupt Trigger Level.

COMxRcvTrigLvl=n, default n = 8, legal values 1, 4, 8, or 14

The 16550 UART has two 16 byte fifos (First In First Out buffers). One is for incoming (received) characters, and the other for outgoing (Transmitted) characters. These two fifos are the 16550's distinguishing features, and are key to the performance enhancements that TurboCom provides.

Associated with the Receive Fifo, is the Receive Fifo Interrupt Trigger Level. This programmable value determines to what level the receive fifo must be filled by incoming data, before the UART will issue a service requesting Data Available interrupt. You may set this variable to one of its four legal values 1, 4, 8, or 14. If your Windows applications lose incoming characters when your system is very active, you might try reducing this variable from the default of 8 to 4 or even 1.

If you are serious about dinking with this variable you should probably become thoroughly familiar with the technical issues concerning the 16550 UART (get a data sheet) and the interrupt response time of you system.

To set this variable, change the following line(s) of text in your SYSTEM.INI file in the [TurboCom] section.

```
COMxRcvTrigLvl=n
```

Where x= the Com port number 1, 2, 3, or 4, and n= the trigger level 1, 4, 8, or 14

For example;

COM3RcvTrigLvl=14 means that the UART will not issue a Data Available interrupt until it has collect 14 bytes of data. It can receive 2 more bytes before it absolutely must be serviced, or a data overrun will occur.

If you're wondering what happens if just 13 characters come in, the answer is that the UART has a timer. That timer will cause an interrupt if data sits in the UART for four character times without another character being received or the fifo being serviced.

Setting the Transmit Fifo Filling Level for Windows and DOS applications

Note: This is another item for the hacker. Unless you really enjoy tinkering, there is probably no reason to change the Transmit Fifo Filling Level.

COMxTxDepth=n, Default n = 8, legal values 1, 2, 3......, 16

Each time the Transmit Fifo becomes empty, the UART issues a service requesting Transmit Fifo Empty interrupt. While an empty Transmit Fifo could be filled with as many as 16 characters, there may be times when this is not desirable. Certain flow control issues might make it desirable to stop transmitting characters before the 16 byte fifo is empty, and there is no mechanism in the 16550 UART to accomplish this. The only option is not to fill the Transmit Fifo so full.

To set the number of characters placed in the Transmit Fifo on each Transmit Fifo Empty interrupt, place the following line(s) of text in your SYSTEM.INI file, in the [TurboCom] section.

```
COMxTxDepth=n
```

Where x = the Com port number 1,2,3 or 4, and n = the fill level 1,2,3....., 16

For example:

COM3TxDepth=12 means that each time the Com3 Transmit Fifo Empty interrupt is serviced, the Transmit Fifo will be filled with 12 characters from the Transmit Queue, if that many are available.

Setting the COMM Buffer Size for DOS Applications

If your DOS applications running under enhanced mode Windows are experiencing character overruns (lost received characters) this section is for you.

COMxBuffer=n, Default n = 128, legal values 0, 1, 2......, 10000

In traditional (UNIX, VMS, etc.) multi-tasking environments, interrupt level services are provided by the operating system. All applications are required to use these services to access the hardware. Enhanced mode Windows is unique in that each Virtual Machine (VM) provides its own hardware drivers. The Serial Port Drivers, (Comm Drivers) for the Windows VM (Comm.drv, or TurboCom.drv in our case) are provided with the Windows package. The Serial Port Drivers for DOS applications running under Windows are provided either by DOS or by the DOS application itself. The question then arises, what shall we do with a character received by a UART which is "owned" by a VM (DOS application) that has been switched out. TurboBuf provides an answer, by providing a buffer in system RAM for each comm port. TurboBuf collects characters from the UARTs, places them in buffers, and then when the VM (DOS application) which owns a comm port is active, TurboBuf delivers the characters by simulating the UART, interrupts and all. In this way even 16550 unaware DOS applications benefit from its high performance features.

To set the number of bytes buffered for DOS applications, place the following line(s) of text in your SYSTEM.INI file, in the [TurboCom] section.

```
COMxBuffer=n
```

Where x = the Com port number 1,2,3, or 4, and n = the size of the buffer in bytes, 0, 1, 2..... 10000

For example:

COM2Buffer=512 means that TurboBuf will provide a 512 byte buffer for the COM2 serial port. This buffer would be large enough to hold about one half second's worth of characters at 9600 baud. If you assume that there are three VMs active, Windows and two DOS applications, and that the minimum time slice is set to the default value of 20 milliseconds, then 512 bytes would be more than sufficient.

Setting the COMM Boost Time for Windows and DOS applications

If your Windows or DOS applications are experiencing sluggish comm performance, character overruns (lost received characters), or loss of keyboard characters under enhanced mode Windows, this section is for you.

COMBoostTime=n, Default n = 2, legal value 0, 1, 2......, ?

This setting gives a VM addition time each time it receives an interrupt from an comm port. Since TurboCom and a 16550 UART result in substantial reductions in interrupt overhead it may not be necessary to ever increase this value. If you give a VM too much additional time using this method, it can quickly hog the whole system. There is only one COMBoostTime profile string for all comm ports.

To set the number of milliseconds additional time upon each owned comm interrupt, place the following line of text in your SYSTEM.INI file, in the [TurboCom] section.

```
COMBoostTime=n
```

Where n equals the number of additional milliseconds operating time granted to a VM upon receipt of each comm interrupt.

For example:

COMBoostTime=5 will give the VM an additional 5 milliseconds of operating time each time it receives an interrupt from a comm port which it owns.

Hardware: I/O Addresses, port availability, Interrupts, and Com Port Lockups

If you have experienced difficulties with Windows' Message Boxes that proclaim "The COMx port is currently assigned to a DOS application. Do you want to reassign the port to Windows?", and then when you answer in the affirmative nothing happens. Or, Com ports that work one minute but not the next, this section is for you. It will give you some understanding of the roots of your travail, and an explanation of how TurboCom will help.

I/O addresses and Port Availability:

A Windows Comm driver looks in a dedicated area of RAM to find the addresses of your system's (up to) four Com ports. This RAM area is initialized by the system ROM BIOS at boot time. Only a few very recent ROM BIOSes are fully aware of Com3 and Com4 and correctly initialize this data area. If the data area is not properly initialized, and the address the Windows Comm driver finds is ZERO, Windows will make assumptions that depend:

upon whether Windows is operating in Real/Standard vs. Enhanced mode,
upon whether it is looking for COM1/COM2 or COM3/COM4,
upon whether the bus structure is the ISA vs. EISA/MCA,
upon the Model and Submodel data in the ROM BIOS
and finally, upon quantity and the base addresses of the Com ports in you system.

The permutations and combinations have created a great deal of misunderstanding and confusion.

All of this is made more complex by a Com port addressing inconsistency between the Windows Comm driver COMM.DRV and the enhanced mode Virtual Machine Comm driver *VCD.

However, all the addressing and availability problems are resolved if the dedicated area of RAM has the correct information about Com port addresses, and the addressing inconsistency between the Windows Comm driver and the Virtual Machine Comm driver is corrected.

TURBOCOM.SYS is provided to properly initialize the data area and TURBOVCD.386 corrects the addressing inconsistency. TURBOCOM.SYS is not required for IBM PS/2 Microchannel models 50 and above.

TURBOCOM.SYS, in part, acts as an extension of the ROM BIOS. When loaded by the CONFIG.SYS file at DOS boot time, TURBOCOM.SYS searches for four Com ports, places their addresses in the dedicated RAM area, and reports its findings.

TURBOCOM.SYS also permits the user to optionally enter load line arguments (switches) which extend the search for Com ports to non-standard I/O addresses, and force a specific physical-to-logical mapping of Com ports that are found. Up to four switches may be entered, one for each logical Com port. For example, if your ISA system has four Com ports at the quasi-standard addresses of 3f8h, 2f8h, 3e8h, and 2e8h, the TURBOCOM.SYS load line in your CONFIG.SYS file might be:

```
DEVICE=TURBOCOM.SYS /3f8 /2f8 /3e8 /2e8
```

This would make the following assignments:

Logical port		Physical port
Com1:	=	3f8h
Com2:	=	2f8h
Com3:	=	3e8h
Com4:	=	2e8h

You could reverse the Logical to Physical connection with the following load line:

```
DEVICE=TURBOCOM.SYS  /2e8 /3e8 /2f8 /3f8
```

Which would make the following assignments:

Logical port		Physical port
Com1:	=	2e8h
Com2:	=	3e8h
Com3:	=	2f8h
Com4:	=	3f8h

If you choose to use a non-standard arrangement of Com port addresses, be aware that many DOS based communications programs make their own Logical to Physical associations, and pay no attentions to those provided by the ROM BIOS. Conflicts of Com port associations between Windows and DOS based applications lead to great confusion and grief!

TURBOCOM.SYS is 16550 UART aware, and will properly initialize these high performance devices.

Interrupt sharing and Com Port Lockups:
While simultaneous hardware interrupt sharing is not possible in the ISA bus environment without special serial port hardware, it should be possible to share hardware interrupts on a sequential basis. However, because of a conceptual error in the Windows Enhanced mode Virtual Comm driver, interrupt sharing of any kind with Windows in an ISA bus environment will lead to Com port lockup. Once a Com port has been acquired by a process (i.e., a Windows or DOS application) the hardware interrupt associated with that Com port will become permanently unavailable to the other serial port sharing the interrupt. If the other serial port, which shares the interrupt, is subsequently activated it will also claim the hardware interrupt. At this point neither port will operate, yet neither port will completely relinquish the interrupt. The only solution is a power off re-boot of your system. Even a warm boot is not enough because of limitations in most ROM BIOS.

TURBOVCD.386 corrects this problem and allows serial ports to be passed about with abandon, provide the one-at-a-time on an IRQ restriction is observed.

GENERAL

7. For purposes of licensing to any agency of the United States government, the software and documentation are provided with RESTRICTED RIGHTS. 8. This agreement is governed by the laws of the State of California and is enforceable by Bio-Engineering Research Laboratories and/or its resellers and distributors. The prevailing party in any action brought in connection with an alleged infringement of proprietary rights in the enclosed software will be entitled to recover its costs and expenses, including attorneys fees.

Bio-Engineering Research Laboratories - 2831 Seventh St., Berkeley, Califorina - (415) 540-8080