The upper memory area (the area between 640K and 1024K) is normally reserved for use by your system's hardware. On most PCs, there are "holes" in this area--upper memory addresses that are not associated with any physical RAM or ROM chips. QEMM maps memory from outside the first megabyte of RAM into the vacant areas. In addition, QEMM uses advanced techniques to reclaim some parts of upper memory that were previously reserved for use by hardware. High RAM is QEMM's name for the memory mapped into the upper memory addresses. Once memory is mapped into upper memory addresses, QEMM can use that High RAM to load TSRs, device drivers, and parts of DOS. By loading these items into upper memory instead of conventional memory, more conventional memory is available for your other programs.

The XBDA (Extended BIOS Data Area) is a RAM region on IBM PS/2s and some PC clones that contains hardware information beyond that contained in the BIOS data area. The XBDA is normally located at the top of conventional memory and can be an obstacle to effective memory management.

Paradox (Borland)
Clipper (Computer Associates)
DESQview (Quarterdeck)
Folio Views (Folio Corp.)
Generic CAD (AutoDesk)
Lotus 1-2-3 2.x (Lotus)
Lotus Agenda (Lotus)
FoxPro (Microsoft)
Quattro Pro (Borland)
Wildcat BBS (Mustang)
Turbo C (Borland)
DESQview/X (Quarterdeck)
Intellicom (Liberation Enterprises)
Geoworks (Geoworks)
Lotus Magellan (Lotus)
Lotus Symphony (Lotus)
PC-Write (Quicksoft)
Q & A (Symantec)
WordPerfect 5.x and 6.x (WordPerfect Corp./Novell)
Harvard Graphics (SPC)
dBASE (Borland)

Allocation Method

This drop-down list box lets you decide how QEMMs **FreeMeg** feature safeguards the first megabyte of memory while other Windows programs are loading.

If you choose **Original**, which is QEMMs default, then FreeMeg allocates nearly all of the first megabyte while Windows programs are loading, to prevent some programs from monopolizing precious first-megabyle memory that other Windows programs may need to load. When using the Original method, FreeMeg first allocates all chunks of first-megabyte memory larger than 32K; then all chunks bigger than 16K; then all chunks bigger than 8K; and so on, until all chunks of first-megabyte memory larger than 512 bytes (or whatever number you set the FreeMeg Block Size to) are taken.

If you choose **Worst-Case**, then FreeMeg allocates all first megabyte memory while Windows programs are loading, then keeps only every other 512-byte block (or whatever number you set the FreeMeg block size to) and frees the rest of the 512-byte blocks. This method makes it impossible for any program to monopolize a large region of first-megabyte memory, but still leaves behind a great many small chunks of memory, in case Windows needs lots of small bits of first-megabyte memory for the data blocks it uses to keep track of programs. However, this method may cause your Windows programs to load more slowly. You probably wont need to use the Worst-Case option, but you can try it if a particular program cannot load with FreeMeg enabled.

If you choose **None**, QEMM disables the FreeMeg feature. This makes it possible for a Windows program to monopolize first-megabyte memory and prevent other Windows programs from loading, no matter how much memory remains on the system.

Overview of QEMM Setup

QEMM Setup makes it easy to enable or disable QEMM's optional features, as well as add or delete QEMM's fine-tuning and troubleshooting parameters. QEMM Setup also provides you with hints on using QEMM and lets you view the QEMM READ ME file for late-breaking information and technotes covering a variety of technical issues. QEMM Setup can also assist you in troubleshooting any problems that might occur.
QEMM Setup is organized into five tabbed pages:

- Features
- Compatibility
- QDPMI
- DOS-Up
- Windows

Each page includes a **Reset** button which discards any changes you have made on that page since you last saved your QSetup settings. The **Reset All** button at the bottom of the screen discards changes made to all of the pages. The Windows page also includes a **Defaults** button which restores the settings on this page to their default values. After making changes to any of the pages, you must select **Save** to store your changes.

QEMM Setup gives you help every step of the way. When you select an option from a menu, you will see an explanation of what the option does. If you still have questions, press **F1** or select the **Help** button for assistance.

Block Size

The Block Size field indicates the largest size block that QEMMs **FreeMeg** feature will leave available when it safeguards first-megabyte memory while Windows programs are loading. By default, this field is set to 512 bytes. A smaller Block Size value will safeguard even tiny chunks of first-megabyte memory, but may slightly slow down the loading of programs; a higher value may speed up program loading slightly, but leaves more first-megabyte memory at risk. You can try increasing the Block Size value if a particular program cannot load with FreeMeg enabled.

Booting Your System Without QEMM

To reboot your PC without QEMM's memory management follow these steps:

- Reset your system by pressing the **Ctrl**, **Alt**, and **Del** keys simultaneously, by pressing the reset button, or, if necessary, by turning the machine on and off.
- When you hear a beep, hold down the **Alt** key until the boot sequence stops.
- If you are using QEMM's **DOS-Up** feature, you will see a message asking if you want to unload the **DOSDATA** device driver. Press Esc to unload DOSDATA, then immediately press and hold down Alt again until you see the following message:

QEMM: Press Esc to unload QEMM or any other key to continue with QEMM.

- Press the **Esc** key.

QEMM will not load, so programs will not load into High RAM; however, your system will be usable.

Return to Hints Main Menu.

Bus-mastering Devices and QEMM

Quarterdeck Technical Note #121

Note: All references to 386 computers or to the 80386 processor, unless otherwise stated, refer to 386 and higher processors.

This note is divided into two parts. The first section on troubleshooting is for those users who believe that they are having problems with QEMM and a SCSI hard drive. Section two provides information on bus-mastering issues.

SECTION ONE: Troubleshooting

Refer to the troubleshooting section of the QEMM manual, or in the technical note QEMM GENERAL TROUBLESHOOTING (**TROUBLE.TEC**) for instructions on how to boot your machine without QEMM.

1) If your machine locks immediately after posting the banner for QEMM386, check to see if the DB=2 parameter is on the QEMM386.SYS line in CONFIG.SYS. If this parameter is not present, add it. If your machine now works, you're done, and you may read the information section below. If your machine still fails, continue with Step 2.

2) Add the parameter VDS:N to the QEMM line, and reboot. If this solves your problem, proceed to Step 3.

3) Remove VDS:N parmeter (if one is present) from the QEMM line, and immediately before the QEMM386.SYS place the line

DEVICE=C:\QEMM\FIXINT13.SYS /STACKSIZE=384

FIXINT13.SYS is in the QEMM directory in QEMM 7.5 and later. It is also available on the Quarterdeck BBS at UK 01245-496943 or Ireland 353 1 2844381. If this solves your problem, you're done. If this does not solve your problem (but the VDS parameter did), replace it. In either case, you may now choose to read the information section below.

SECTION TWO: Information on Bus-mastering

Q: What is a bus-mastering device?

Bus-mastering devices are peripherals, typically hard drives, that do their own direct memory addressing (DMA) without going through the machine's Central Processing Unit (CPU) or its DMA controller. The most common bus-mastering devices are SCSI hard disk controllers, but other types of devices can be bus-mastering as well. Bus-mastering ESDI disk controllers and video cards do exist, and an increasing number of bus-mastering network cards are available as well. While bus-mastering devices are high-performance devices and quite often found on 386 and higher systems, they are, unfortunately, by design incompatible with one of the most common operating modes of the 80386 processor--the Virtual 86 mode.

Specifically, the problem is that the device puts data into absolute memory addresses and assumes that the contents of those memory addresses will always remain constant. However, on a 386 processor in Virtual 86 mode, this is often an incorrect assumption. When a 386 memory manager such as QEMM, or a 386 operating environment such as DESQview 386 or Microsoft Windows Enhanced Mode is used, it typically associates physical memory with linear or "logical" addresses. QEMM does this, for instance, to make High RAM appear at addresses between 640K and 1MB. When a bus-mastering device tries to

access data in memory, it presumes that physical and logical addresses are the same. In Virtual 86 mode, a given memory address can, at any moment, contain code or data from various regions of physical memory.

If you are using a bus-mastering device on a 386 that is in Virtual 86 mode and memory paging is occurring (when QEMM is providing High RAM; when QEMM is providing expanded memory; when DESQview 386 or Microsoft Windows is switching from one virtual machine to another), your machine will probably hang when you use the bus-mastering device, unless certain precautions are taken.

Quarterdeck first became aware of the problem from customers who had bus-mastering SCSI hard disk controllers. Users reported that they could boot their machines and start up DESQview. As long as they ran only one application, their system ran fine. As soon as they opened a second application, the system would hang. The problem was also seen by users who were not using DESQview, but who were using the LOADHI feature of QEMM. In both cases, the hang would occur because the disk controller assumed that memory really existed at the address that it was accessing. In theory, this could have caused data corruption, but in reality it never did. The memory corruption was typically so extensive that the systems simply hung as soon as a change in the memory map occurred. Other 386 memory managers exhibited the same symptoms, as did Windows version 3.x when run in Enhanced Mode. QEMM solved the problem in its own code, but this solution was not adopted by Windows when it entered Enhanced Mode (see the reference to SMARTDRV in item 4 below).

Q: What is the best approach to running bus-mastering devices?

There are several possible solutions:

1) THE BEST SOLUTION: Contact the maker of your bus-mastering device and see whether the manufacturer supports the VDS (Virtual DMA Services) specification. VDS is now an industry-wide specification supported by IBM, Microsoft and Quarterdeck, as well as many other hardware and software suppliers. VDS, either provided in the device's ROM or as a device driver, allows a bus-mastering device to find the real physical address of its data when the processor is in Virtual 86 mode. QEMM versions 5.00 and later support the VDS specification. A VDS driver provides the best solution to this problem in terms of reliability, speed and memory efficiency. A VDS driver may be loaded into High RAM if it appears in the CONFIG.SYS file after the QEMM386.SYS line, but you may need to manually add a DB=2 parameter to the QEMM386.SYS device line to accomplish this if you are not using QEMM 7.5 or later; see section 6 below.

2) Make sure you're using QEMM version 7.5 or later. QEMM version 7.5 automatically created a buffer when it detects an addressing problem with a bus-mastering hard drive controller, and this buffer's support continues into Microsoft Windows.

3) Similarly, the drivers of many bus-mastering hard disks have proprietary (that is, non-VDS) buffering options. The best course in this case is to check the documentation for your disk controller to see if the driver has a parameter to set up buffering for disk operations. Some drivers will also document parameters that are specific to 386 operations. For example, the early Adaptec drivers SCSIHA.SYS and AHA1540.SYS included both 386 and disk buffering options invoked by the parameters "/v386" and "/b:64." "/v386" stands for virtual 386; "/b:64" allocates a 64k buffer, for DMA.

Unlike the drivers in (1) above, these drivers do not provide VDS services. If you are using a driver such as this, make sure that it is not loaded high. The purpose of such a driver is to provide buffering into physical addresses that are the same as logical addresses; if the program is loaded high, its buffer will be in logical addresses that are not the same as their physical addresses. Please read the section below titled "Making Sure Your Device Driver Loads Low".

4) Check the documentation for your bus-mastering device and see if it can be configured to use the BIOS or any one of the standard DMA channels. QEMM can correct the problem if the BIOS or standard DMA channels are used.

5) As mentioned above, bus-mastering hard drives can also cause problems for Microsoft Windows 3. Microsoft's solution is in its SmartDrive disk cache. SMARTDRV (and other disk caches) contain code that can buffer direct memory access. Before QEMM 7.03, QEMM's VDS services were almost completely disabled when you entered Enhanced mode, so SMARTDRV's buffering was needed to ensure that no bus-mastering conflicts occured inside of Microsoft Windows. If you are using QEMM 7.5 or later, QEMM's VDS services and disk buffering will function properly while Microsoft Windows Enhanced mode is running, so loading SMARTDRV is not necessary.

a) The versions of Smartdrv that ship with Microsoft Windows 3.1, DOS 5, and DOS 6 have two functions: to provide disk caching through a module loaded in AUTOEXEC.BAT, and to provide buffering for SCSI hard drives through a module loaded in CONFIG.SYS. If you are using Windows 3.1 AND a bus-mastering hard drive and you are NOT using any of the options numbered 1 through 4 above, make sure that the following line appears in CONFIG.SYS:

DEVICE=C:\WINDOWS\SMARTDRV.EXE /DOUBLE_BUFFER

(If your path to SmartDrive differs, change C:\WINDOWS to the correct path.)

b) Windows 3.0 and DOS 5 shipped with SmartDrive version 3 or lower. If you are using one of these versions of SmartDrive, make sure that the following line appears in your CONFIG.SYS file:

DEVICE=C:\WINDOWS\SMARTDRV.SYS

(If your path to SmartDrive differs, change C:\WINDOWS to the correct path.)

Please read the section below titled "Making Sure Your Device Driver Loads Low".

6) QEMM has a DB=xx (DISKBUF=xx) parameter that can prevent QEMM-SCSI problems at the expense of a little conventional memory. "xx" is the number of K used for buffering. Any value for xx is sufficient to correct the problem. DISKBUF=2 is fine for most cases. Configuring QEMM with a DISKBUF greater than 2 might improve disk performance, but setting DISKBUF to more than 10 is probably a waste of memory.

Q: How does QEMM's detection of bus-mastering hard drives work?

QEMM will detect a bus-mastering hard drive and create a disk buffer automatically as long as QEMM itself is loaded from that bus-mastering hard drive. If you load a driver that provides VDS (Virtual DMA Services) support BEFORE you load QEMM, QEMM will not create the disk buffer. In cases where QEMM automatically creates this buffer, it does not add a DISKBUF parameter to the QEMM386.SYS line. QEMM's automatic detection of bus-mastering hard disks is active only when the RAM parameter is specified on the QEMM386.SYS line in the CONFIG.SYS file. You can disable QEMM's automatic disk buffering by using the QEMM386.SYS parameters DISKBUF=0, but there is usually no reason to disable this feature. If you have a bus-mastering hard disk that you do not load QEMM from, QEMM will not detect bus-mastering conflicts with it, and you must either use the disk controller's VDS support or specify the DISKBUF=nn (DB=nn).

Use of the DB= parameter will not help if the bus-mastering device is something other than a hard disk. If your bus-mastering device is not a hard disk then the solutions above, especially #1, are your only options.

If your bus-mastering hard disk controller uses a VDS device driver that is loaded after QEMM386.SYS, QEMM will still create a 2K disk buffer, because the VDS support will not be active when QEMM loads. This disk buffer will be necessary in most circumstances, because the Optimize program would otherwise fail when it tried to load the VDS driver into High RAM. However, you may wish in this circumstance to

reduce the size of the disk buffer as much as possible by placing the DISKBUF=1 parameter on the QEMM386.SYS device driver line. The smaller disk buffer is preferable here, because the disk buffer will never again be used after the VDS driver loads, and a bigger disk buffer uses valuable conventional memory.

If you are both disabling automatic disk buffering (with the DISKBUF=0 parameter) and creating a disk buffer for the page frame (with the DISKBUFFRAME=xx parameter), you must place the DISKBUF=0 parameter before the DISKBUFFRAME=xx parameter on the QEMM386.SYS line in the CONFIG.SYS file. If you reverse this ordering, automatic disk buffering will still be disabled, but the disk buffer for the page frame will not be created.

Q: I don't have a VDS driver, and I think that my proprietary device driver or my disk cache should be loaded low. How do I prevent it from loading high?

For double-buffering to work properly, the device driver for your bus-mastering hardware must be loaded in conventional memory, where physical and logical addresses are almost always the same. You must therefore make sure that it loads low if you are depending on it to provide DMA buffering. We will use SMARTDRV as an example of such a program. Change the instructions below to fit your device driver.

Ensure that there is no LOADHI command preceding SMARTDRV on the line which loads it.

OPTIMIZE will very likely try to load SmartDrive high, unless you instruct it not to do so. This is most easily done as follows:

1) Using a text editor, create a text file called OPTIMIZE.NOT in the \QEMM directory. Systems with DOS 5 and later can type "EDIT OPTIMIZE.NOT" from the \QEMM directory to create the file. If such a file exists already, simply open it for editing.

2) Put a new line in OPTIMIZE.NOT that says:

SMARTDRV

Do not specify a pathname nor an extension to the file name. Save the file and exit the editor.

3) From now on, OPTIMIZE will not affect the SMARTDRV line in either CONFIG.SYS or AUTOEXEC.BAT. As long as SMARTDRV is not being loaded high already, it will not load high during future OPTIMIZE sessions.

Q: I know I have a bus-mastering device on my computer, but I haven't seen any problem. Why not?

It's possible that your bus-mastering device uses a standard DMA channel for DMA operations. QEMM automatically supports bus-mastering when standard DMA channels are used.

Your bus-mastering device may have been shipped with a VDS driver in its ROM. Some bus-mastering hard disk controllers ship with drivers that make VDS calls, and these drivers do not require the DB parameter or any other buffering. We expect that most bus-mastering devices will eventually include VDS drivers and, therefore, will not exhibit any problems when run in Virtual 86 mode.

Return to Technotes Main Menu.

Compression Buffer Size

The QEMM MagnaRAM feature, which gives you more Windows memory by compressing data, depends on having a RAM buffer that it allocates out of the physical memory (installed RAM) on your system. By default, MagnaRAM takes a buffer equal to one-quarter of the physical memory that is available when it starts up. The minimum setting for this buffer is 32K; the maximum is the amount of available physical memory.

In general, if you increase the size of this buffer, MagnaRAM will be able to do more data compression and provide more extra Windows memory, but the system may run a bit slower, especially when starting a new program. Conversely, if you decrease the size of the buffer, MagnaRAM will be able to provide less additional Windows memory, but the system may speed up a bit. The ideal size of the buffer for your system will depend on which of MagnaRAM's benefits--speed or extra memory--means more to you.

Compression Threshold

The **Compression Threshold** field tells QEMM whether to compress all the data in the MagnaRAM RAM buffer, or just a part of it. The number in the field is a percentage that indicates how much of the RAM buffer should be left uncompressed. The default setting, 0, means that QEMM will try to compress all the data in the buffer. The maximum setting for this field is 100, which means that QEMM will only start compressing data when the buffer is 100% full, and that it will stop compressing when the contents of the buffer have shrunk to less than 100% of its capacity.

Any setting of this field is a tradeoff between more memory and better performance. A bigger value means that MagnaRAM can create slightly less additional Windows memory, but that performance will be improved slightly. A smaller value means a bit more Windows memory and a bit more performance overhead.

Contacting Quarterdeck's Technical Support Department

As a registered owner of QEMM, you are entitled to **90 days** of prepaid technical support. You can receive support by fax, mail, or phone. If you have a modem, you can get support by contacting Quarterdeck's CompuServe forum or through several other public message forums. Your 90 days of prepaid support starts with your first call, letter, fax, or online communication in reference to QEMM.

Before contacting technical support, we encourage you to see **Appendix A** of the QEMM Reference Manual for troubleshooting information, and **Appendix B** for a list of technical bulletins (technotes) included with QEMM. These technotes are placed in a directory called \QEMM\TECHNOTE during installation of QEMM. You can view technotes from this help file or from within QEMM Setup by selecting **Technotes** from the main menu.

The troubleshooting guide and the technotes give step-by-step solutions to several common problems. Also, be sure to see the file **READ.ME** for late-breaking information. You can view the READ.ME file from this help file or from within QEMM Setup by selecting **Technotes** from the main menu.

For additional information on contacting technical support see the **Passport** booklet included with QEMM or read the technote **CONTACT.TEC**.

Quarterdeck also offers VIP maintenance and support coverage. See your Passport brochure for information.

Return to Hints Main Menu.

Contacting Technical Support

Quarterdeck Technical Note #144

CompuServe

With over 2 million subscribers, CompuServe is the most popular computer information service in the world. And with good reason! CompuServe provides electronic mail, online reference material, up-to-date news services, travel services, weather information, online shopping, investor services, games, over 500 special interest forums, and much more.

If you are already a member of CompuServe, just type GO QUARTERDECK at any ! prompt to access our forum. Private e-mail is accepted at 76004,2310.

For more information about our CompuServe forum, refer to Quarterdeck Technical Note #134, "Technical Support viaCompuServe" (CIS.TEC).

Quarterdeck White Papers

Individual Quarterdeck Technical Notes cover a specific topic of interest to users of Quarterdeck products. Whether you need assistance in correcting a problem or you simply want a better understanding of how something works, these notes are an invaluable source of information.

Our complete technical note library is available on most of the online services listed above. Some of the more common technotes are included in this online help file. These technical notes are collectively referred to as the Quarterdeck White Papers.

QWHITE.COM, our complete technical note library and reader, is also available on most of the online services listed above. For more information about QWHITE.COM, refer to QuarterdeckTechnical Note #236, "Quarterdeck White Papers" (QWHITE.TEC).

Q/FAX

Our innovative Q/FAX system gives you access to our complete technical note library. Quarterdeck Corporation was the first software company to offer this 24-hour outbound, self-faxing system! Using the telephone attached to your FAX, you can access our Q/FAX service by dialing UK 01245-496931 or Ireland 353 1 2844383.

For an updated list of technical notes available, request document #100. This master list is updated as new technical notes become available.

Other Online Services

Online services provide a convenient way to contact QuarterdeckTechnical Support. Our electronic forums are staffed by our most senior technical support representatives. In these forums,you will find discussions of the latest issues, our complete technical note library, product upgrades, development tools, user-submitted utilities, and much more!

Although many of these forums offer private e-mail, we recommend posting your messages publicly whenever possible. In addition to Quarterdeck Technical Support, there are many users in these forums who are eager to help. Because these public messages can be read by anyone, you may get a faster response.

If you have a modem, the following options are available to you:

1. Quarterdeck BBS

The Quarterdeck Bulletin Board System (BBS) is available 24 hours a day, 7 days a week. The recommended modem settings are 8 bit word length, no parity, and 1 stop bit. You can reach the Quarterdeck BBS at UK 01245-496943 or Ireland 353 1 2844381. For more information about our BBS, refer to Quarterdeck Technical Note #105, "Using The Quarterdeck BBS" (BBS.TEC).

2. Internet - Messages and Anonymous FTP Site

Public messages: comp.os.msdos.desqview

Please note that the above message group is not run by Quarterdeck, but our senior technical support representatives do monitor the messages.

Private e-mail: qsupport@qdeck.com

To access our anonymous FTP site, use the following information:

Host: qdeck.com (149.17.8.10)
Login: anonymous
Password: Type your e-mail address here (e.g., johndoe@netcom.com).
Note: Refer to the README file in ~\pub for a list of files available for download from our FTP site.

3. BIX (Byte Information Exchange)

Public messages: JOIN DESQVIEW
Private e-mail: QOS.REP2

For more information about BIX, refer to Quarterdeck Technical Note #160, "Technical Support via BIX (Byte Information eXchange)" (BIX.TEC).

4. MCI Mail

Private e-mail: QUARTERDECK

5. SmartNet

Public messages: DESQview conference

SmartNet is a network of individual BBSes that exchange messages and files. The Quarterdeck BBS is a SmartNet node, or member. If a BBS in your area is a SmartNet node, and carries the DESQview conference, you can contact Quarterdeck Technical Support via that BBS. The advantage of these echoed conferences is that you can contact us via a local phone call, rather than dialing our BBS directly which may be long distance. If your local SmartNet node does not carry the DESQview conference, send a note to the Sysop. If enough interest is shown, the Sysop may consider adding the conference to that system.

For more information about SmartNet, refer to QuarterdeckTechnical Note #159, "Technical Support via Smartnet" (SNET.TEC)

6. Fidonet

Public messages: DESQview echo.

FAX

All FAXes are responded to within 24 hours of receipt (weekends and holidays excluded). Please include your telephone number and either your product serial number or customer VIP number on all FAX correspondence. FAX all technical support inquiries to UK 01245-496941 or Ireland 353 1 2844380.

Mail a Letter

All letters are responded to within 24 hours of receipt (weekends and holidays excluded). Please include your telephone number and either your product serial number or customer VIP number on all mailed correspondence. Mail all inquiries to the following address:

Quarterdeck International Ltd.
BIM House
Crofton Terrace
Dun Laoghaire
County Dublin
Ireland.

Telephone Support Hotline

Quarterdeck Technical Support can be reached at UK 01245-494940 or Ireland 353 1 2844144.
See the Quarterdeck Passport booklet for a complete directory of contact numbers.

Return to Technotes Main Menu.

QEMM Setup

Contents

For **Help** using the Online Help, see **Help About Help.**

Copy ROMs to RAM

This option enables or disables QEMM's ability to speed up **ROMs** by copying their program code into RAM where it will execute more quickly.

To enable or disable this feature:

- To have QEMM copy ROM code into faster RAM, select **Yes**.
- To prevent QEMM from copying ROM code into RAM, select **No**.

Yes adds the ROM parameter to the QEMM386.SYS line in CONFIG.SYS; No removes the ROM parameter.

Why you may want to copy ROMs to RAM:

- If your system does not already speed up ROMs by copying them into faster RAM, enabling this option may speed up some system operations, particularly writes to the screen by programs that use BIOS or DOS video calls (like DOS's COMMAND.COM).

Why you would *not* want to copy ROMS to RAM:

- Your system may already copy ROMs to RAM--this feature is already provided if your system has shadow memory.
- On some systems, ROMs may not work properly when copied to RAM; floppy disk drives may malfunction on a few systems if the ROM code that controls the floppy disk drives is speeded up. In this case, you can use the QEMM Analysis procedure to help determine which areas of ROM can be copied to RAM (for information see Chapter 9 of the QEMM Reference Manual).
- This feature diminishes QEMM's memory pool by the amount of memory taken up by your ROMs - usually about 96K.

On page 14, the QEMM User Guide describes the buttons on the bottom of the QEMM Setup screen. On the right side of the tab window (or property page), the Reset button restores your settings to the values that were in force when you entered that page. The Default button (which appears only on the Windows property page) sets the options for FreeMeg, Resource Manager, and MagnaRAM to their default settings. The Help button provides context-sensitive online help for the displayed property page.

On page 27, the QEMM User Guide incorrectly suggests that an exit button appears on the right side of the QEMM User Interface screen. To exit the program, select Exit from the File menu or press Alt - F4.

QEMM Setup Online Help

Designed and Written By

Phil Glosserman
Kathy Hand
Bob Parker
and
Dan Sallitt

QEMM's VCPISHARE:Y (VS:Y) is not compatible with the DESQview/X SERVER module. Do not use the VS:Y parameter if you are using DESQview/X.

Norton Cache (Symantec)
Cache86 (Aldridge)
PC-Kwik (PC-Kwik)
Hyperdisk (Hyperware)

If you are using real-mode Novell network drivers in Microsoft Windows 95, Microsoft recommends that you set the LOADTOP=0 option in the MSDOS.SYS text file. (Note that Windows 95's text-based MSDOS.SYS is NOT the same as the MSDOS.SYS DOS component in previous versions of DOS.) If you choose Windows 95's default LOADTOP=1 setting to load the command processor at the top of conventional memory, you may experience corruption of the DOS environment, which includes values for PROMPT, PATH, and SET statements.

This happens irrespective of the presence of Quarterdeck software. However, for similar reasons, if you are using QEMM's DOS-Up option, you should always choose to load COMMAND.COM low.

DOS=HIGH is a CONFIG.SYS statement that loads parts of the DOS kernel into the HMA (the first 64K of extended memory). The HMA is available only if you are using DOS version 5 or or higher (and is not available for DR DOS 6 users). The amount of DOS that gets moved to the HMA depends on your configuration, but is generally at least 40K.

The most common reason not to enable the DOS=HIGH feature is if you run a program that uses the HMA more efficiently than DOS, like DESQview or DESQview/X. By eliminating the DOS=HIGH statement in CONFIG.SYS you may be able to have more available memory inside DESQview and DESQview/X windows. For information on maximizing the memory inside DESQview and DESQview/X windows, select "Technotes" on the QEMM Setup menu. When the next menu displays, select "QEMM and DESQview or DESQview/X.

Exclude stealthing a particular ROM

You use this option to tell QEMM not to stealth a particular ROM. You should tell QEMM not to stealth a particular ROM only when attempting to solve problems with the **StealthROM** feature.

To use the feature that excludes stealthing of a particular ROM:

- Select **Address** to exclude a particular ROM from being Stealthed. Then, click in the adjacent field and type the starting address of the ROM you want to prevent from being Stealthed.
- Select **None** if you have excluded a particular ROM from being Stealthed and you now want to remove this exclusion.

If you specify that a particular ROM should not be Stealthed, QEMM Setup will place the XST=xxxx parameter on the QEMM386.SYS line in your CONFIG.SYS file, causing QEMM not to stealth that ROM. You can get the starting addresses of all stealthed ROMs from the Manifest QEMM Overview screen. In general, video ROMs are located at C000 (or at E000 on Micro Channel systems); system ROMs at F000. If you have a disk ROM (many systems do not), it will generally be located at an address between C800 and E000.

If possible, it is usually more memory-efficient to solve StealthROM problems with the EXCLUDE parameter than with the EXCLUDESTEALTH parameter.

DoubleSpace is the disk compressor that comes with DOS versions 6.0 - 6.20.

If you are using MS-DOS 6's **DoubleSpace** or **DriveSpace**, you can save 31K-49K of memory by using QEMM's **Stealth D*Space** feature to relocate the DoubleSpace or DriveSpace device driver in expanded memory. See Chapter 5 of the QEMM Reference Manual for details.

DriveSpace is the disk compressor that comes with DOS 6.22 (or later).

The EMS page frame is a 64K area, usually in upper memory, used by programs to access expanded memory. QEMM also uses the page frame to enable its StealthROM and Stealth D*Space features.

Edit QEMM device line

This selection lets you manually edit the QEMM device line. If the QEMM parameters do not fit in the visible field on screen, an arrow at the left or right of the field indicates the presence of off-screen parameters. The field will scroll when you use the arrow keys or type

Enable MagnaRAM Memory Compression

This check box determines whether QEMMs **MagnaRAM** feature, which gives you more Windows memory by compressing data, is active. If you disable this feature by clearing this box, you will need to restart Windows before the change takes effect.

Disabling MagnaRAMs memory compression is not the same thing as disabling MagnaRAM altogether. Even when memory compression is disabled, QEMM will still allocate MagnaRAMs RAM buffer, and it will still send swapped-out memory to the buffer until it is full. To remove MagnaRAM from memory, use the Uninstall MagnaRAM option.

Enable QuickBoot

This selection enables or disables QEMM's feature that speeds up warm reboots (i.e., when you reboot by pressing Ctrl+Alt+Delete).

To enable or disable QuickBoot:

- To enable QuickBoot, select **Yes**.
- To disable QuickBoot, select **No**.

If you enable QuickBoot, you can also enable the **Timeout** feature and specify a number of seconds in the adjacent field. The timeout feature tells QEMM to post a QuickBoot menu for xx seconds (where xx is a number from 1 to 99) when you warm boot your system. The QuickBoot menu lets you choose which drive to boot from. When the timeout value of xx seconds expires, QEMM automatically reboots the system without your intervention. The default timeout value is 0, which tells QEMM to warm boot without posting the Quickboot menu.

To enable the Timeout feature:

- Select **Timeout**.
- Click on the seconds field and type the number of seconds to wait for user input before automatically booting using the default boot drive.

Enable Resource Manager Option

QEMMs Resource Manager feature lets Windows 3.1 users fit more programs into memory before running out of precious system resources. The Resource Manager feature stores some system resources in a different place in memory, bring them back into the system resources area when they are needed.

If any program does not function properly when the Resource Manager feature is enabled, you can disable Resource Manager by clearing this check box, saving the change, and restarting Windows. Before disabling Resource Manager, you should try telling QEMM to disable Resource Manager for the particular program that is failing.

Enable or Disable DOS-Up

QEMM's DOS-Up feature loads into **High RAM** certain parts of DOS that would normally load into conventional memory. Depending on how your system is configured, DOS-Up can free between 7-70K of conventional memory for running DOS programs.

To enable or disable DOS-Up:

- Select **Do not use DOS-Up** to disable DOS-Up.
- Select **Use all the features of DOS-Up** to have DOS-Up load as much of DOS as possible into upper memory.
- Select **Use the specified features of DOS-Up** if you want to choose the parts of DOS that DOS-Up should load into High RAM. Then, click on the features you want to load into High RAM.

IMPORTANT: Once you enable or disable DOS-Up, you must reboot your PC for the change to take effect.

Parts of DOS that DOS-Up can move out of conventional memory are:

- **DOS resources** (FILES, BUFFERS, FCBS, STACKS, LASTDRIVE). The amount of memory that these resources take up varies with your configuration. See Manifest's DOS Overview screen for details.
- **COMMAND.COM** (the DOS command processor). Its size varies in different versions of DOS. It is normally smaller than 5K.
- **DOS data** (the DOS data structures that are not moved out of conventional memory by the **DOS=HIGH** statement). If you do not use DOS=HIGH, DOS-Up will additionally load into upper memory those parts of the DOS kernel that DOS=HIGH would have loaded into the HMA.
- **DOS=HIGH** is a feature of DOS version 5 and later (it is not a DOS-Up feature, but you can enable or disable it from QEMM Setup). DOS=HIGH loads the DOS kernel, buffers and part of COMMAND.COM into the HMA, the first 64K of extended memory. The amount of DOS that gets moved to the HMA depends on your configuration, but is generally at least 40K. We recommend that you use DOS=HIGH unless you routinely run a program (such as DESQview or DESQview/X) that can use the HMA more efficiently than DOS. If you use DOS=HIGH, you can still use the features of DOS-Up.
- If you are using DR DOS 6 or Novell DOS 7, you cannot use DOS=HIGH. For information on using DOS-Up with DR or Novell DOS, see **NW&DRDOS.TEC**.

DOS-Up makes three changes to your CONFIG.SYS file. The DOSDATA.SYS driver, which loads at the beginning of the CONFIG.SYS, prepares the system for DOS-Up. The DOS-UP.SYS driver loads the DOS kernel, data, and resources into High RAM. And your SHELL statement is modified so that LOADHI.COM can put COMMAND.COM in upper memory. If you have no SHELL statement, DOS-Up creates one for you.

QDPMI

QEMM Setup's QDPMI page lets you enable or disable the Quarterdeck DOS Protected Mode Interface for programs that support DPMI (e.g., Microsoft's C/C++ Development System for Windows version 7, Borland's C/C++ version 3, and Intel's Code Builder Kit version 1.1). QEMM's DPMI host is called QDPMI. Unlike other DPMI hosts, QDPMI provides virtual memory in the DOS environment.

To enable or disable QDPMI:

- Select **Do not use Quarterdeck's DPMI host** to disable this feature.
- Select **Use Quarterdeck's DPMI host** to enable this feature.

IMPORTANT: Once you enable or disable QDPMI, you must reboot your PC for the change to take effect.

QDPMI uses about 2K of RAM. If you do not have applications that support DPMI, you may want to disable QDPMI to free up 2K of memory. Protected-mode programs that are VCPI clients will run under QEMM even if QDPMI is not loaded.

If you enable QDPMI, QEMM Setup will place the QDPMI.SYS device line in your CONFIG.SYS file. You can specify the size in kilobytes of the DPMI swapfile, an area on disk that will be used as virtual memory for DPMI applications. The default swapfile size is 1024K (1 meg). The advantage of specifying a bigger swapfile is that more virtual memory will be available to DPMI programs. It is particularly important to have a large swapfile if you have a low-memory system and a memory-hungry DPMI application. The disadvantage of specifying a bigger swapfile is that more of your hard disk may be used up by your DPMI program. QDPMI does not use any of your hard disk for a swapfile until the DPMI program requests the memory, and the swapfile grows as needed up to the maximum size that you set.

Enable or Disable Stealth D*Space

QEMM can use its Stealth technology to move DOS 6's **DriveSpace** or **DoubleSpace** driver entirely out of conventional and upper memory, making it appear in the EMS page frame when it is needed. This saves about 31K-49K that would otherwise use up space in conventional memory or upper memory.

To enable or disable QEMM's Stealth D*Space feature:

- Select **Use QEMM's Stealth D*Space** to enable Stealth D*Space.
- Select **Do not use QEMM's Stealth D*Space** to disable Stealth D*Space.

IMPORTANT: Once you enable or disable Stealth D*Space, you must reboot your computer for the change to take effect.

If you enable Stealth D*Space, QEMM Setup will place the ST-DSPC.SYS driver in your CONFIG.SYS file to relocate the DoubleSpace or DriveSpace driver. ST-DSPC.SYS uses about 3K and can be loaded high. Optimize will add the necessary command to load this driver high if there is room for it in upper memory.

Suspend/resume laptop support

This option enables or disables QEMM's special support for the suspend/resume feature found on many portable computers. Suspend/Resume is a feature that allows you to run the computer on low power when it is not in use, and to restore the system to its previous state when you return to it. Many systems with the suspend/resume feature will work fine without special support from QEMM, but some systems will not return properly from a low power state if a 386 memory manager such as QEMM is active. If your system has a suspend/resume feature that is not working properly with QEMM installed, you should enable QEMM's support for suspend/resume.

To enable or disable QEMM's special support for the suspend/resume feature:

- Select **Auto** to enable QEMM's special support for suspend/resume.
- Select **No** to disable QEMM's special support for the suspend/resume feature.
- Select **Interrupt** if you have tried the Auto selection without success. After selecting Interrupt, click in the adjacent field and specify a hardware interrupt number for your PC's suspend/resume feature. 2, D, 72, 73, and 77 are the numbers most likely to be used by the Suspend/Resume feature. See your hardware documentation or contact the manufacturer for information on the appropriate interrupt number. If you choose Yes, QEMM Setup places the SUS parameter on the QEMM386.SYS line in the CONFIG.SYS file. This parameter makes QEMM search for the hardware interrupt that suspend/resume is using. If you select Interrupt, Setup adds the SUS:xx parameter, where xx is the interrupt number you specify.

Exception Reports Explained

Quarterdeck Technical Note #142

Q. What are processor exceptions? What is an Exception #6, #12, or #13? And what does the QEMM Exception message mean?

Users of QEMM may sometimes encounter a report that an attempt has been made to execute an invalid instruction. It is almost certain that QEMM, in and of itself, is not the cause of Exception problems, though QEMM's memory managment may come into conflict with other hardware and software on your system.

In this technical note, we explain in detail what a processor exception is, how you can interpret the information provided by the exception report, and what you can do to remedy the situation in the unhappy event that the techniques in **TROUBLE.TEC** don't provide relief from the problem.

To answer the questions above, it's worthwhile to examine the Exception report bit by bit.

"The processor has notified QEMM that an attempt has been made to execute an invalid instruction..."

Exceptions are the processor's response to unusual, invalid, or special conditions in the normal operation of the 80386 processor and others in its family. (The 80386 family includes the 80386SX, the 80386DX, the 80486SX, the 80486DX, and Pentium processors; their memory management architecture is essentially the same. In this document, the term "386" refers to any and all of these processors.) Exceptions cause the 386 processor to stop what it's doing and to try to react to the condition that caused the exception. QEMM is designed to capture some of these exceptions -- particularly those caused by protection faults or invalid instructions, which could cause a program or the entire system to crash -- and display a report to the user. When the processor encounters an instruction that it does not want to execute, it passes control to QEMM. QEMM's protected mode INT 6, INT 12, or INT 13 handler posts the Exception message. Neither DOS nor Microsoft's EMM386.EXE have as sophisticated protected mode handlers, so if an exception occurs using only DOS or EMM386.EXE, your system may simply crashes and leave you without a report.

Q. What causes an Exception?

"...This may be due to an error in one of your programs, a conflict between tw o pieces of software, or a conflict between a piece of hardware and a piece of software...."

The exception reported is most commonly #13, the General Protection Fault exception. This indicates that a program has tried to execute an invalid or privileged instruction. On the 386 processor, programs can run at varying privilege levels, so that the processor can better protect application programs (which generally run at lower privilege levels) from crashing the operating system or control program (which typically runs at the highest privilege level). DOS and QEMM do not enforce this protection, but QEMM can report when a program running at the lowest privilege level tries to execute a privileged instruction. The result may be a system crash, but QEMM does provide a report before the crash happens.

Invalid instructions are harder to classify, for indeed Exception #13 is something of a catch-all. Some examples of invalid instructions include:

- 386-specific instructions that are disallowed when the processor is in virtual 8086 mode. The processor is in this mode whenever QEMM is in an ON state -- essentially when it is providing expanded memory or High RAM.

- A program trying to write data to a segment that has been marked as executable or read-only (the data could overwrite program code).

- Trying to run program code from a data segment (if data is read as code, it will be a series of meaningless or nonsensical instructions -- which, if executed, could jump to invalid addresses or overwrite the operating system)

- Exceeding the limit of a segment. Segments in virtual 8086 mode are not permitted to exceed FFFFh (65535 decimal) bytes or to fall below 0 bytes. Neither a program instruction nor a memory reference may span the boundary of a segment.

It is this last error which is the most common; this is a problem also known as "segment wrap", which we will discuss later. Again, QEMM is designed to trap and report these errors, but it cannot defend against the system crashes that they may cause.

Occasionally Exception #12, indicating a stack exception, will be reported. This is a protection violation very similar to Exception #13, but is one in which the stack segment is involved in some way. Although generally no easier to solve, it is a somewhat less general report than Exception #13.

Exception #6 may also be reported. This indicates that a program has tried to execute an invalid opcode. Machine instructions are stored as sequences of bytes in memory. These sequences are fetched from memory and decoded by the processor into machine-language instructions. When the processor encounters a sequence of bytes for which there is no corresponding machine-language instruction, the processor generates an Exception #6 and QEMM reports the Exception to you.

Very infrequently, an Exception #0 is reported. This is not intentional; it is usually the result of QEMM's stack being corrupted while QEMM was trying to report another exception, or is the result of some other system error.

It is important to remember that in the vast majority of cases, QEMM is not involved with the problem, but is merely reporting it. Most often, the problem is simply a bug in the offending program.

Q. What do I do now?

"...It is likely that the system is unstable now and should be rebooted...."

QEMM is designed to offer the user the opportunity to terminate the offending program, or to reboot the computer, but often the damage has already been done by the time that the Exception is trapped and reported. In this instance, you may find the computer locked regardless of what you choose. If the computer is indeed hung, you should write down the information on the screen and then reboot the machine.

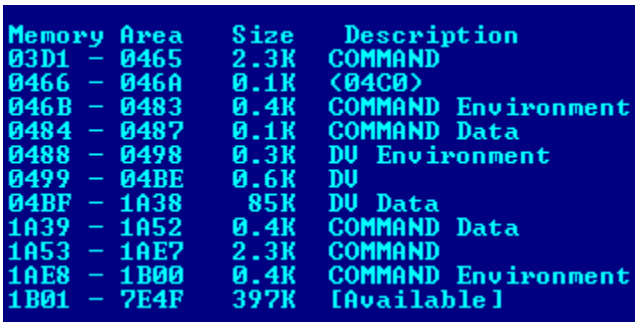
While QEMM's Exception reports can be cryptic to non-programmers -- or to programmers who have little experience with assembly language -- the information that they provide can sometimes be quite helpful. Exception reports can help you to identify which program has triggered the exception message, what the invalid instruction was, and the state of the processor's registers when the error occurred. Armed with this information, you may be able to help the developer of the offending application to determine the problem that led to the exception, and thus the developer may be able to provide a temporary workaround or a permananent fix.

The exception report is divided into three parts --

1) The vector or class of exception, and its location and error code. The location of the exception indicates the address in memory at which the invalid instruction was attempted. The program loaded at this address (if indeed a program is loaded there) should be noted by running Manifest.

Exception #13 at 1B12:0103, error code: 0000

In this example, the program loaded at address 1B12:xxxx is automatically your suspect. Reboot your system in the same configuration as you had when the Exception #13 occurred. If the problem happened during an application program, don't load the application just yet. Load Manifest instead, and have a look at First Meg / Programs.



The sample Exception #13 above happened in that Available range, so it was the program that would have been loaded had we not loaded Manifest -- that is, the application program. If you have a TSR loaded low, and the Exception #13 is occuring within that TSR's address space, then it is your suspect, rather than the application. In any case, the program whose code falls into the range in which the Exception #13 occurred likely has a problem of some type.

2) The second part of the Exception #13 message is the register dump:

AX=0000 BX=0000 CX=0000 DX=0000 SI=FFFF DI=0000 BP=0000 DS=1B12 ES=1B12 SS=1B12 SP=FFFE Flags=7246

The registers are the temporary storage areas on the 80386 chip which are used for calculations and addressing. Each register is two bytes (16 bits) in size, so each register is capable of holding a value from 0 to FFFF (hexadecimal), or from 0 to 65335 (decimal).

If any registers here are 0000 or FFFF, it's possible that you could be looking at a segment wrap. A segment wrap happens whenever a program attempts to access -- read from or write to -- something beyond the limit of a segment. A word value consists of two adjacent bytes; if a word value were to begin at FFFF (which is the last byte of a segment), the second byte of that value will be outside the segment -- and an attempt to read from or write to that word will thus cause a protection violation. Similarly, a doubleword is four adjacent bytes; if any of the last three bytes are outside of the segment limit, a segment wrap and a protection violation will occur when an access is attempted.

On an 8086 processor, it's actually possible for a segment wrap to occur without a protection violation, simply because the 8086 has no hardware protection at all. What is the byte after the last byte of a segment? On the 8086, it's the FIRST byte of the same segment. (Non-technical analogy for poker players: Queen - King - Ace - Two - Three is a straight in the penny-ante poker game played when the 8086 processor is dealing. The 386 processor is a very strict dealer, and does not permit this.) It is possible (though unlikely) for a program to continue without a crash on an 8086 processor when two "adjacent" bytes are actually a whole segment apart; it could theoretically be possible on a 386 too, but the exception is generated before the memory access can be completed.

This sort of problem is seen most commonly during a string move -- the program is copying a whole block of data from one range of addresses to another. You may not understand this, and actually it doesn't matter if you don't. Briefly, though, SI stands for Source Index; DI stands for Destination Index. These two registers are used for string instructions -- instructions that load or copy information sequentially. String instructions are extremely powerful and useful, since they allow the developer to

deal with large amounts of data in a single pass. A byte or a word value can be fetched from memory by one string instruction, dealt with, and then the result can be copied to a new memory location with a second string instruction -- and all this can be managed with an extremely tight, fast loop. An entire range of addresses (for example, in screen memory) can even be filled with a given value using a single instruction. The catch here is that the string instruction is only valid as long as the value of the SI or DI register does not fall outside the range addressable by these registers. If either one of these tries to exceed FFFF (or tries to fall below 0000), as a string is being copied from one region of memory to another, you'll get a protection violation.

3) Instruction: A5 CC 00 00 00 00 00 00 00 00 00 00 00 00 00 Do you want to (T)erminate the program or (R)eboot?

This is the invalid instruction that the program was trying to execute when the processor stopped it. Since most humans don't have a hope of interpreting machine language by looking at the opcodes, you can get a better interpretation of what is going on by examining this instruction with a program that can render machine codes into assembly language. (Well... it's better than nothing.) To do so, go into DEBUG; type DEBUG at the DOS prompt.

Enter the values from the Instruction line by typing

E 100

at DEBUG's hyphen prompt, and then entering each byte (pair of digits) from the instruction line. Follow each byte with a space.

(As a bonus -- if you're running under DESQview, you can Mark the information from the Exception #13 report, and Transfer it into DEBUG running in a different Big DOS window.)

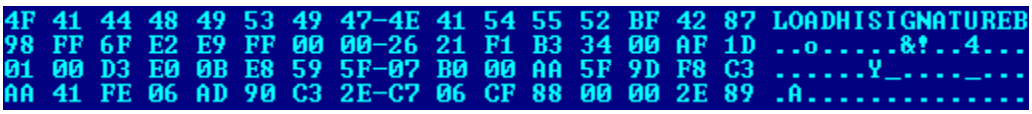
If most of the bytes begin with a 4, 5, 6, or 7, there's a good chance that you're seeing a program trying to execute text, thinking that text to be code. This can happen in several circumstances, but frequent offenders are those programs which load code at the top of conventional memory during boot -- and therefore during the OPTIMIZE process -- and presume that no program will allocate that memory. Programs which place parts of themselves at the top of conventional memory typically do so without protecting themselves from programs like LOADHI which may need to allocate all conventional memory at appropriate times; LOADHI (and programs like it) will overwrite the vulnerable code.

As a real-world example, PROTMAN, a program whose purpose in life is to manage the loading of various parts of 3Com and MS-LAN networks, did this in past versions, as explained in Quarterdeck Technical Note #173, PROTMAN.TEC. During the OPTIMIZE process, LOADHI would allocate all conventional memory while it was determining the size of the various drivers that were being loaded. PROTMAN would jump to what it thought was still its own code, but there would be LOADHI signatures there -- text -- and PROTMAN would crash.

You can see the contents of this string if you Dump the instruction you just entered; use DEBUG's D instruction to do this.

-d 100

At the leftmost edge of your screen, you'll see a list of addresses. At the center and right of your screen, you'll see this:



ASCII codes starting with 2 are generally punctuation marks; bytes 30-39 represent numeric digits; 3A-3F are punctuation, 41-5A are capital letters, 61-7A are small letters. Any instruction made up mostly of these numbers is almost certainly text -- and therefore not executable program code. The program that is trying to run such an instruction is doing so in error. When the instructions are NOT mostly in the 40-80 range, you should try to Unassemble them.

```
-u 100

20C0:0100 A5            MOVSW
20C0:0101 CC            INT             3
20C0:0102 0000          ADD             [BX+SI],AL
```

This is the killer instruction from the example Exception #13 above. It's performing a MOVSW (MOVe String Word) at a point when the SI register is FFFF, and that means that it's trying to write a word value to or from the last byte of a segment, which (as described above) is illegal.

Other invalid instructions are harder for the non-programmers of the world to interpret. Often the first byte of an invalid instruction is 0F -- which is a valid protected-mode instruction, but which the processor interprets as an invalid opcode if the machine is in Virtual 86 mode. Exceptions of this kind showed up more commonly in the past, with programs that were trying to enter protected mode without calling the Virtual Control Program Interface. VCPI is an industry-standard way for protected-mode software to coexist with 386 expanded memory managers such as QEMM; all 386 memory managers these days are VCPI-providers, and almost all protected-mode programs are VCPI users (or "clients"). Non-VCPI protected-mode programs include some memory- and hardware-diagnostic programs, and programs that use the DPMI memory management specification exclusively. Diagnostic programs typically recommend that you disable all memory-management software during diagnosis. DPMI programs will typically accept VCPI memory management; those rare programs that do not will simply refuse to start up under QEMM. In such cases, you may install QDPMI (the Quarterdeck DPMI Host) on your system; QDPMI is available on the Quarterdeck BBS at UK 01245-496943 or Ireland 353 1 2844381, Compuserve (!GO QUARTERDECK), or large local BBS systems.

Q. How can an Exception #13 be fixed?

Quarterdeck Technical Note #241, QEMM: General Troubleshooting (**TROUBLE.TEC**) is a good place to start. This note describes common problems and possible solutions, and will help if the cause of the Exception #13 is a memory conflict or bus-mastering issue.

If you follow the instructions in TROUBLE.TEC completely, and the Exception #13 persists, the prospects for a resolution are bleak, since the problem is almost certainly a bug in the offending program. If this is so, unless you can alert the developer of the program (and make him or her understand all this, which might be another task altogether), you can never really make the problem go away, although sometimes you may be able to make it subside.

Changing the location of the offending program in memory will sometimes help. If you're running under DESQview, and you're sure that you've given the program enough memory (i.e., all you can give it), try adding 16 to the size of the script buffer on page 2 of Change a Program. If you're not running under DESQview, try adding an extra file handle or two. The key here is to change the location of the program in memory, which can occasionally be enough to provide temporary relief from the Exception.

There is a substantial caveat: You're not fixing the problem by doing this; you're just making it submerge. There's still probably a bug in the offending program -- you've just changed it from a bomb to a landmine. If you can reproduce the problem consistently, you should still contact the publisher of the application with all of the data from the Exception message, and all of the data that you can supply about your system and its current configuration.

With the exception (no pun intended) of the techniques mentioned above and in TROUBLE.TEC, non-programmers can do little to fix the root cause or even the symptoms of Exception reports. If you are unsuccessful in resolving a conflict, the information provided by the report should be forwarded, along with a Manifest printout and a complete description of your system, to the developer of the program that you were running at the time.

Return to Technotes Main Menu.

Exception Reports: Advanced Troubleshooting

Quarterdeck Technical Note #232

OVERVIEW

This document addresses Exception #6, #12 and #13 error messages. These three Exceptions are so similar in cause, nature, and solution that they are all covered by the information below. Any reference to an Exception error in this document applies to an Exception #6, Exception #12, or Exception #13 error.

Please note that sometimes QEMM cannot report the error to the screen, so a blank screen or a lockup occurs. For troubleshooting purposes, treat a system lockup or a blank screen (one that accepts no input) as an Exception error.

This troubleshooting procedure will isolate and resolve the conflict most effectively if you are able to reproduce the conflict at will (i.e., you know of a specific action or seriesof actions that will cause the conflict to occur). The reason for this is that this procedure follows a logical set of tests to determine when a conflict is occurring and when it is not. A conflict that randomly occurs is difficult to troubleshoot because you do not know for sure when the configuration being used is actually resolving the conflict.

Q: What is an Exception error?

A: Exceptions are generated by the protection mechanisms on your PC's processor (and not by QEMM), in response to an invalid or unusual condition. QEMM has the ability to detect exceptions and display relevant information to the screen. The report informs you what has happened so that you may take the necessary steps to resolve the conflict. Without QEMM's warning, your system may simply have crashed or hung without a message, or may have become unstable. For more detailed information about Exceptions themselves, please refer to Technical Note #142, "QEMM: Exception 13 Explained" (**EXCEPT13.TEC**).

Q: How do I resolve my Exception error?

A: Follow these steps:

1) Run through **TROUBLE.TEC**. This technical note treats almost all potential causes of Exception reports with which QEMM might be involved, and does so in a very efficient way. Follow the steps in TROUBLE.TEC to determine which aspect of your system's configuration is the cause of the problem. TROUBLE.TEC is a general troubleshooting guide, and may refer you to still other technical notes to resolve specific conflicts.

2) If TROUBLE.TEC proves that QEMM is not involved, and Exceptions or crashes still persist with DOS' memory managers, contact the publisher of the application that is running when the problem occurs. Inform the publisher's technical support department that you have reproduced this problem using both QEMM and DOS's memory managers. This indicates a problem in which the specific memory manager is not involved. Contacting the manufacturer of the program is often the quickest way to resolve the conflict.

3) If TROUBLE.TEC shows that QEMM is involved with the problem but does not point you towards a solution, Quarterdeck's technical support department is easily reached via electronic channels such as CompuServe (GO QUARTERDECK), Internet (mail qsupport@qdeck.com or the USEnet newsgroup comp.os.msdos.desqview), the Quarterdeck BBS (UK 01245-496943 or Ireland 353 1 2844381), or fax UK 01245-496941 or Ireland 353 1 2844380. When contacting Quarterdeck, be sure to explain the symptoms of the conflict and the results of the tests performed while following TROUBLE.TEC. You can also call our Technical Support line at UK 01245-494940 or Ireland 353 1 2844144 for further assistance. When you call, please be at the machine

that is experiencing the conflict.

Return to Technotes Main Menu.

The driver FIXINT13.SYS (described in Chapter 12 of the Reference manual under the erroneous name INTFIX13.SYS) is a replacement for the driver ULTRAFIX.SYS, which was formerly distributed on the Quarterdeck bulletin board and other electronic forums. If you use ULTRAFIX.SYS, replace it with FIXINT13.SYS.

Find ROM holes

This option enables or disables QEMM's feature that finds "ROM holes" --unused areas in the system ROM (between F000 and FFFF)--and makes them available for **High RAM** or expanded memory mapping. This feature is only available when QEMM's **StealthROM** feature is not in use.

To enable or disable QEMM's ability to find and use ROM holes:

- Select **Yes** to enable QEMM's ability to find ROM holes and make them available for High RAM or expanded memory mapping.
- Select **No** to disable this feature.

The feature that finds ROM holes is on by default. If you choose No, QEMM Setup adds the RH:N parameter to the QEMM386.SYS line in CONFIG.SYS.

The most common reason to disable this feature is to troubleshoot floppy disk problems or other conflicts between QEMM and your system. If disabling this feature solves your problem, it may be more memory-efficient to use the EXCLUDE parameter on a section of the system ROM instead of using the RH:N parameter.

Fill Upper Memory with RAM

This option creates or removes **High RAM** in the upper memory area. When High RAM is present, you can load TSRs, device drivers and parts of DOS into upper memory. By loading these items into upper memory, you will have more conventional memory available for DOS programs.

To create or remove High RAM:

- Select **Yes** to create High RAM.
- Select **No** to remove High RAM.

Yes causes QEMM Setup to place the RAM parameter on the QEMM386.SYS line in CONFIG.SYS; No causes QEMM Setup to remove the RAM parameter. By default, QEMM's installation creates High RAM.

Freeing Additional Conventional Memory

If you use DOS text-based programs, you can extend conventional memory by as much as 96K by using QEMM's **VIDRAM** feature. For information on VIDRAM, see Chapter 6 of the QEMM Reference Manual.

The **Manifest** program may be able to tell you how to free up a bit more conventional memory. Run Manifest by typing **MFT** at the DOS prompt. When Manifest displays, type **H** to select **Hints**, and read the suggestions Manifest offers.

If you do not use any programs that require **DPMI** (DOS Protected Mode Interface) memory, you can free up 1-2K by deleting the **QDPMI.SYS** device driver line from your CONFIG.SYS file.

Return to Hints Main Menu.

Falcon (Spectrum Holobyte)
Flight Simulator (Microsoft)
Patriot (Three-Sixty)
Spear of Destiny (ID Software)
Strike Commander (Origin)
Ultima Underworld (Origin)
V for Victory (Three-Sixty)
Wing Commander (Origin)
Wolfenstein (ID Software)
X-Wing (LucasArts)

Help About QEMM Setup's Online Help

General information: For information on using the Windows online help system, press **F1** at any time.

▶

Secondary windows: The small graphic image of a printer and clipboard to the left of this paragraph appears in the upper left corner of most secondary windows. When a topic in a secondary window is accompanied by this graphic image, you can print the topic by clicking on the printer or copy the topic to the Windows clipboard by clicking on the clipboard.

Hints

Information is available on the following topics:

QEMM and Microsoft Windows

QEMM and DESQview or DESQview/X

QEMM and DR-DOS or Novell DOS

QEMM and Disk Compression Software

QEMM and Bus-mastering Devices

QEMM and the EMS Page Frame

QEMM's StealthROM Feature

QEMM's Optimize Program

Undoing an Optimize

QEMM's Manifest Program

VIDRAM: Extending Memory for Text-based Programs

Freeing Additional Conventional Memory

QEMM's New Parameter Names

Booting Your System Without QEMM

Contacting Quarterdeck's Technical Support Department

Switching Between MS-DOS 6's Memory Manager and QEMM

Return to Hints, Technotes, and README Menu

Some customers report less conventional memory available with QEMM 8 than with previous versions. In many cases, this is because QEMM attempts automatically to EXCLUDE a byte in the F000 region, in order to support better Microsoft Windows' MaxBPS= SYSTEM.INI setting. If you have set MaxBPS set to a value greater than 200, QEMM's default behaviour will likely be preferable to you. If you have MaxBPS set to a value of 200 or less, you might wish to try adding the SRBP:N parameter to the end of the QEMM 386.SYS line in CONFIG.SYS.

QEMM's superior detection of Plug and Play BIOS and other forms of adapter RAM and ROM may cause QEMM to EXCLUDE automatically more address space than previous versions. This is safer than including these regions by default, but can result in less High RAM or less conventional memory than before, or can result in the EMS page frame being placed in conventional memory. The QEMM Analysis procedure, detailed in the manual, may allow you to reclaim this High RAM. You may also wish to check Manifest's Hints screen to confirm that Analysis will be helpful.

Finally, QEMM may have increased extended memory overhead in this version on some systems, which may result in a net loss in available extended or expanded memory. There is no remedy for this situation.

If you have a command in AUTOEXEC.BAT in the form GOTO <LABEL>, where there is no corresponding <LABEL> in the AUTOEXEC.BAT file, the OPTIMIZE process may simply terminate. Ensure that all GOTO statements in AUTOEXEC.BAT refer to valid labels.

MagnaRAM: General Information and Troubleshooting

Quarterdeck Technical Note #315

This technical note provides general information and troubleshooting tips for MagnaRAM 2.0. For information on MagnaRAM 1.0, refer to the Quarterdeck Technical Note #300, "MAGNARAM.TEC 1.0 General Information".

General MagnaRAM 2.0 Information

Quarterdeck's MagnaRAM is an easily configurable utility designed to increase your available memory, performance and multitasking capabilities within Windows 3.1x, Windows for WorkGroups, and Windows 95. MagnaRAM provides you with more available memory and performance by using powerful memory multiplying technology. MagnaRAM's memory multiplying techniques provide:

Significant amounts of additional Windows memory for your programs by compressing data, both in RAM memory and on your hard disk.

Performance benefits to your system by minimizing the use of slower virtual memory. The more you use your high-speed system RAM, the less you must access the much slower virtual memory.

System Requirements:

PC with a 386 or higher processor.
4 MB of RAM. (Performance improvement with additional installed RAM.)
Windows or Windows for WorkGroups 3.1x or Windows 95 running in 386 Enhanced mode
MS or IBM-DOS 3.1 or later, or Novell DOS.
Windows Virtual Memory feature enabled.
Permanent swap file recommended.

General Troubleshooting Issues

Q. Is MagnaRAM compatible with disk compression programs?

A. MagnaRAM is compatible with disk compression programs such as Stac Electronics' Stacker and Microsoft's DoubleSpace and DriveSpace. MagnaRAM does not interact directly with your hard disk, or with disk compression programs -- it lets Windows swap all data to the disk.

Q. Where is the Uninstall icon that is documented in the manual?

A. Some revisions of the MagnaRAM User Guide make anerroneous reference to an uninstall icon in the MagnaRAM folder for Windows 95. On Windows 95 systems, use the Add/Remove Programs feature located in the Control Panel.On Windows 3.1x / Windows for WorkGroups systems, use the MagnaRAM uninstall icon located in the MagnaRAM group.

Q. I have deleted MagnaRAM, but I can't reinstall it. My machine says that MagnaRAM is already installed. What can I do?

A. On Windows 95 systems, remove any lines referring to MagnaRAM in your System Registry. For example:

1. Click on "Start" and choose the "Run" option.
2. Type in REGEDIT <enter> to load the Registry Editor program.
3. Click on the Search menu, then click on the Find option.
4. Type MAGNA <enter>. If you find anything that refers to MagnaRAM, press <DEL> to remove that section.
5. Click on Find Next and continue searching until no more are references found.

On Windows 3.1x / Windows for WorkGroups systems, remove any lines referring to MagnaRAM located in the SYSTEM.INI and WIN.INI files. For example:

1. Click on File/Run from the menu of Program Manager.
2. Type in SYSEDIT <enter> to bring up the System Editor program.
3. Click on Window and select SYSTEM.INI.
4. Locate the line in the [386Enh] section that ends with MAGNARAM.VXD and remove it.
5. Click on Window and select WIN.INI.
6. Locate the line in the [windows] section that ends with LOGO31.EXE and remove it.

Q. I loaded MagnaRAM and now Windows 3.1x / Windows for WorkGroups stops responding when I am using it.

A. Below is a list of some things to check about your Windows configuration:

Make sure that you are using a PERMANENT Windows swap file. This can be verified by clicking on the 386Enh icon in Control Panel, which is by default in the Main program group.

Make sure that you are using the swap file on an UNCOMPRESSED drive. If you are using disk compression software such as Stacker or DriveSpace, use the configuration utilities included with those applications to increase the size of your uncompressed drive. Then you can create a permanent swap file on that drive, and still have room for any other programs that are needed on the compressed drive.

Please contact the manufacturer of the compression software you are using or consult the manuals if you are unsure of how to change the size of your uncompressed drive.

While you are checking the swap file settings in the 386Enh section of Control Panel, verify that 32-bit file access is not enabled OR lower the cache size to 1024K or less.

Using MagnaRAM's setup utility, lower the size of the compression buffer that MagnaRAM uses, and increase the threshold setting to 30-40%.

Edit the SYSTEM.INI file located in the Windows directory and change the order of the drivers in the DRIVERS= line. Try moving MAGNARAM.DLL to the beginning of that line. If that does not resolve the conflict, try placing MAGNARAM.DLL at the end of the DRIVERS= line.

Q. I installed MagnaRAM on my system and Windows appears to be running more slowly.

A. MagnaRAM automatically configures itself to your system, but sometimes it will require fine tuning in order to work at its peak performance.

For Windows 3.1x or Windows for WorkGroups users, try going into MagnaRAM's settings section (the screwdriver icon) and change the PageOverCommit setting from 8 to 3. This affects the amount of linear (Windows) memory that is available to your system. Reducing this number will lower the total amount of linear memory you have available; however, system performance may increase.

Also, for both Windows 95 and Windows 3.1x / Windows For Workgroups users, try decreasing the MagnaRAM Compression Buffer size at an increment of 20-30 percent until you get an improvement in performance.

Q. MagnaRAM does not appear to be doing much. Is it loading?

A. Check the following to see if MagnaRAM is loading:

On Windows 95 systems, use the Registry Editor (REGEDIT) to search for any lines containing "MAGNA". For Windows 3.1x / Windows for WorkGroups, edit the SYSTEM.INI and search for the DRIVERS= line that refers to MagnaRAM. If this entry is not present, MagnaRAM is not properly installed.

If there are Registry entries containing "MAGNA", check the MagnaRAM statistics window and make sure that Compression Buffer Size is set to 10% of the machine's physical RAM. MagnaRAM's compression techniques may not take effect immediately, and may not be readily noticed at Windows startup.

Q. MagnaRAM is loaded and working fine, but I no longer get any sounds in Windows 3.1x / Windows for WorkGroups.

A. After checking that your sound card is setup correctly, edit the SYSTEM.INI file located in the Windows directory. Check the DRIVERS= line that references MagnaRAM. It is possible that when MAGNARAM.DLL was added to that line during installation, the driver for the sound card was "bumped off" of the line, especially if the DRIVERS= line is unusually long. Reinstalling the driver(s) for the sound card may solve this problem. You should consult you sound card manual for the necessary files needed on this line.

Q. Why does MagnaRAM fail to load, and why can't I restart it?

A. You may find that MagnaRAM is is failing to reload some of its files properly. Uninstall MagnaRAM with its built-in UnInstall utility, and make sure that the lines added to the SYSTEM.INI shown on page 30 of the manual are removed. If not, manually remove them. Then re-install the program.

Q. I am trying to install MagnaRAM, but it will not accept my serial number nor proceed to the next screen when I click on the NEXT button.

A. If the NEXT button won't highlight to allow you to go to the next screen, please verify that the Company Name field is filled out.

Q. MagnaRAM doesn't give me enough memory to run an application.

A. Sometimes a program requires a certain amount of physical memory to run. For example, some computer assisted drawing (CAD) programs require 12 megabytes of physical memory in addition to any virtual memory they use. The system may simply need more physical RAM for such programs. If MagnaRAM does not provide you with enough LINEAR memory to run a program, try increasing the size of the permanent Windows swap file. You also try to decrease the Compression Buffer Size at an increment of 20-30 percent.

Return to Technotes Main Menu.

Maximizing Conventional Memory

Quarterdeck Technical Note #296

DOS and the programs that run under it are limited, except in special circumstances, to the first megabyte of the processor's address space. This has consequences for DOS programs, and for operating environments such as Quarterdeck's DESQview and DESQview/X, and Microsoft Windows.

Documentation for Quarterdeck products contains extensive information on memory management, and should be consulted for more detailed study. This technical note provides a list of the most important quick tips for assuring that you have the maximum amount of conventional memory available, whether you are working from DOS, DESQview, DESQview/X, or Microsoft Windows. Note that many of the QEMM features suggested below will be enabled for you automatically at the time you install QEMM.

1. Use QSETUP

The QSETUP program that comes with QEMM runs under DOS and Windows. Its job is to provide you with an easy-to-use way of configuring QEMM's features, which are described below. QSETUP will allow you to configure QEMM, to active the DOS-Up feature (including DOS=HIGH), and to enable Stealth D*Space. At the end of the QSETUP process, you will be given an opportunity to run OPTIMIZE.

2. Use OPTIMIZE

The OPTIMIZE process uses QEMM's RAM parameter and the LOADHI utilities to move as many programs as possible out of the conventional memory area and above 640K. If necessary, OPTIMIZE will offer the option to activate QEMM's Stealth parameter (detailed below). You should run OPTIMIZE

- after running QSETUP - after changing CONFIG.SYS or AUTOEXEC.BAT - after adding or removing hardware from your system

Regardless of the change that you have made, running OPTIMIZE will ensure that you get the maximum conventional memory.

3. Use Manifest.

The Manifest program that comes with QEMM provides you with a detailed view of the configuration of your PC. When run under DESQview, DESQview/X, or Microsoft Windows, Manifest can tell you about the state of the current window. Manifest also includes detailed hints on how to get more conventional memory on your PC; check the Hints Overview AND the Hints Detail screen.

4. Use StealthROM

Go into Manifest, and inspect the QEMM Overview screen. If the screen does not display "Stealth Type" and "StealthROMs" and if you have less conventional memory than you would like, run the QSETUP. OPTIMIZE program again. When OPTIMIZE offers to test for Stealth compatibility, answer Yes, and if OPTIMIZE finds that Stealth is compatible with your system, OPTIMIZE will activate StealthROM. StealthROM is quite robust and compatbile, and typically results in 83K of extra High RAM into which DOS programs can be loaded. If you are having compatibility problems with Stealth, consult your QEMM manual or "QEMM GENERAL TROUBLESHOOTING" (STEALTH.TEC). The time you take in troubleshooting a Stealth conflict is well worth the memory gain.

5. Use DOS-Up

QEMM's DOS-Up feature moves parts of DOS, including FILES, BUFFERS, LASTDRIVE, the DOS

Data Segment (which is a block of memory that DOS uses to keep track of its own operation), and the command processor (typically COMMAND.COM) high. The DOS kernel itself may also be loaded high (see the next section). Using DOS-Up can result in a saving of 7K-70K of conventional memory. To activate DOS-Up, go into QSETUP, choose DOS-Up Options, and enable Partial (if you are you a DESQviewor DESQview/X user) or All (if you do not use DESQview or DESQview/X). Save your configuration, and run OPTIMIZE.

6. Use DOS=HIGH

One of the features that you may activate from QSETUP is the DOS=HIGH feature. This loads the DOS kernel -- about 43K of code -- into the first 64K above the 1024K line, thanks to technology originally discovered by Quarterdeck in 1986. If you are a DESQview or DESQview/X user, it is typically worthwhile NOT to use this feature. DESQview and DESQview/X use this memory (called the High Memory Area, or HMA) more effectively than DOS does. Generally, you will receive maximum memory gains if you use DOS=HIGH, unless you are using DESQview or DESQview/X. To activate DOS=HIGH, go into QSETUP, choose DOS-Up Options, and enable All (or choose Partial and ensure that DOS=HIGH is activated. Save your configuration, and run OPTIMIZE.

7. Use ST-DSPC

QEMM 7.0 and later come with a feature for DOS's disk compression software (DoubleSpace, and in DOS 6.22, DriveSpace) for which Quarterdeck has special support. The code for the disk compression software can be placed into expanded memory, replacing a driver tht is typically over 40K with one that is typically under 4K. This generally represent substantial memory savings; although DoubleSpace can be loaded high, it still takes memory in the first megabyte of adddress space. Stealth D*Space removes most of this overhead. If you are using DOS's disk compression, activate QEMM's Stealth D*Space feature. To do this, go into QSETUP, choose ST*DPACE Options, and choose L for Enable or disable Stealth D*Space. On the next screen, choose Y to enable Stealth D*Space. Save your configuration, and run OPTIMIZE.

8. Use QEMM's Stacker Feature

QEMM 7.5 and 8 come with a feature that can allow Stacker 4 to put most of its code outside of conventional memory. Stacker's overhead in conventional memory or in High RAM to as little as 10K on MS-DOS 6 systems. This can save a great deal of memory indeed on Stac'd systems whose hard drives have large cluster sizes.

A program is available to enable users of Stacker 4 to enable this feature is available to all registered users of Stacker 4, either from Stac Electronics or from Quarterdeck, under the filename S4UP.EXE.

To get your copy of this file, join the CompuServe forum for either Quarterdeck or Stac, by typing GO QUARTERDECK or GO STAC at any CompuServe main prompt. Alternatively, using your modem, call

Stac Electronics BBS (619) 431-5956 Quarterdeck BBS UK 01245-496943 or Ireland 353 1 2844381

Oce you have acquired this file and run the update, you may activate the Stacker feature in this way:

1. If you are currently inside Windows, exit it.
2. At the DOS prompt, change to the Stacker directory.
3. Type ED /I
4. Press Enter to insert a new line.
5. On this new line, type /QD
6. Press Ctrl-Z to exit the editor, and save your changes.
7. Restart your system to put the changes into effect.

As usual, since this represents a change to your CONFIG.SYS, use the OPTIMIZE program again.

9. Use EMS or XMS

Many programs -- TSRs, device drivers, and applications -- are able to use expanded memory (EMS) or extended memory (XMS) reduce their DOS overhead. These features are sometimes automatic, but are sometimes enabled via parameters (or command-line switches) or initialization files. Additionally, parameters may exist that simply reduce the program's size -- sometimes by disabling unneeded features -- without using expanded or extended memory. This may require a little research on your part -- such parameters are found in the online help or in the documentation for the program. However, this research can reap rich rewards in memory savings. Some programs that run only in conventional memory do not have switches to activate EMS or XMS usage, but instead come with EMS-using equivalents. Novell's NETX network shell is an example of this; while it uses 43K of conventional memory, and does not use EMS, it is typically accompanied by EMSNETX (which requires less than 10K of coventional memory) and XMSNETX (which takes similar overhead). These programs are functionally equivalent to NETX, but save considerable amounts of memory.

Also consult the Quarterdeck Technical Note "Why the EMS Page Frame is Important" (**FRAME.TEC**)

10. Use VIDRAM

VIDRAM provides up to 96K of extra conventional memory if you are using text programs. VIDRAM does this by borrowing address space from the VGA graphics buffer on your video card. VIDRAM can be enabled or disabled on the fly, and can also run in text windows within Microsoft Windows. DESQview/X users in particular can take advantage of VIDRAM. (See Section 14 below.)

In order to activate VIDRAM, simply type:

VIDRAM ON

On some video cards, it may be necessary to add the parameter VIDRAMEMS (VREMS) to the QEMM386.SYS line in CONFIG.SYS for VIDRAM to work properly from the DOS prompt. This parameter is incompatible with Microsoft Windows. However, even on such cards, VIDRAM will work in a Microsoft Windows window without the VREMS parameter.

11. Streamline your CONFIG.SYS and AUTOEXEC.BAT

If you are loading many device drivers or TSRs, they will have a direct impact on conventional memory.

Programs such as disk caches, RAM drives, or video speedup utilities typically consume memory in the first megabyte. If you do not need these drivers, or are unsure of their usefulness, you may consider removing them to see the impact on performance. You may find that the benefit of additional memory outweighs the gains provided by such utilities. Arriving at optimal performance will require you to decide which features are more important than others.

If you are using DESQview, DESQview/X, or Microsoft Windows, TSRs or device drivers such as ANSI.SYS, DOSKEY or Sidekick should be removed from the AUTOEXEC.BAT and loaded in a window instead. Such programs only have a memory impact on the window in which they are loaded, but if loaded before your operating environment will reduce the size of all of the windows on your system.

12. Consider removing QDPMI

Many modern DOS-Extended programs use the DOS Protected Mode Interface (DPMI) in order to get access to extended memory. However, the DPMI specification suggests that, in the absence of a DPMI host, other strategies such as VCPI or XMS can and should be used to get access to extended memory. This means that the Quarterdeck DPMI Host (QDPMI.SYS) may not be necessary on your system. If so, use QSETUP to disable QDPMI, save your configuration, and run OPTIMIZE.

13. Use QEMM's Analysis Procedure to Gain High RAM

If you're not using QEMM's StealthROM feature, you may still be able to us e small amounts of address space that are marked as reserved by the System ROM, but which are acutally unused. The Analysis procedure is documented in the QEMM manual. You should not need to perform an Analysis if you are using StealthROM, which typically gains much more High RAM.

14. On DESQview/X Systems with 8514/A Graphics Adapters, Use VIDRAM

If you have an 8514 hardware-compatible video card and DESQview/X, you can use VIDRAM without disabling graphics! This is because 8514/A graphics adapters do not use the address space that VGA adapters do, and because DESQview/X specifically supports the 8514 adapter. This can create DESQview/X windows that can provide 64K-96K of conventional memory. Note that if your 8514 card requires a driver then it is not likely to be hardware compatible and that this hint may not be successful.

First, add the VREMS to the QEMM386.SYS line in the CONFIG.SYS file.

DEVICE=C:\QEMM\QEMM386.SYS RAM VREMS

Then, type the following line before you enter DESQview/X. You may choose to add this line to your AUTOEXEC.BAT file, or to any batch file that starts DESQview/X:

C:\QEMM\VIDRAM ON

This line will activate VIDRAM and extend conventional memory. When you start DESQview/X, it may report "GRFVGA.DVR does not find the correct video adapter". You may safely ignore this message; DESQview/X will function properly.

15. Reduce the overhead of your operating environment.

DESQview and DESQview/X's Setup programs contain some settings that may allow you to squeeze an extra few K of memory out of a heavily loaded system, at the potential cost of some speed or performance.

"Common Memory" is memory used by DESQview and DESQview/X to manage its windows, and the amount you need is usually proportionate to the number of windows you open, and the amount of text you intend to transfer with DESQview's Mark and Transfer feature. The default (and minimum) value under current versions of DESQview is 20K, and should permit the transfer of several screen's worth of data and several windows open simultaneously; if your common memory setting is higher than this, you may wish to reduce it. Under DESQview/X, the default value is 32K; this may be reduced by a few K to eke out an equivalent amount of window memory.

"DOS Buffers for EMS" (under DESQview) or "DOS I/O Buffer" is memory used by DESQview to manage file operations into expanded memory. The default value is 2K under DESQview, and 8K under DESQview/X. Users of QEMM who are not on a network can set this figure to 0K with no loss of performance. The value of this field can affect the speed of disk access; however, it is rarely worthwhile to choose a value higher than 10K or 15K.

If you wish to throw away a few DESQview features, you can probably scrimp a few more K from the Setup program.

On the Keyboard option, you can save as much as 12K if you tell DESQview that you don't wish to use the Learn feature. This will disable DESQview's very useful macro system.

On the Video Monitor option, you can save anywhere from 0K to 16K if you tell DESQview that you don't

wish to display text and graphics at the same time. This will disable DESQview's Video Options menu, prevent graphics programs from being seen when they are in background, and prevent virtualization of graphics. You can save another 2-9K by choosing 0 for "What Display Adapter do you have?". This causes DESQview not to load a video driver. This will keep DESQview from saving and restoring graphics screens or virtualizing graphics.

On the Performance option, you can save 2K by setting the "Manage Printer Contention?" field to its default value of N. This means that DESQview will not intervene to prevent two programs from printing at the same time.

On the Network option, you may disable the network support, or decrease the size of the buffer. This support is needed only for certain network-specific program, and not for most normal DOS applications that are merely run off the network.

The amount of memory you will save will be about 5K plus the size of the buffer reserved in the second field. Unless you know that you need this service, you should try running without it and seeing if you have problems without it that you do not have with it. You may also try decreasing the size of the buffer. The default is 8K.

Return to Technotes Main Menu.

Maximizing Memory with PCMCIA

Quarterdeck Technical Note #298

Q. On my notebook machine, the PCMCIA hardware and software seems to take up a lot of space in High RAM. What can I do to reduce this?

A. PCMCIA hardware and software, by default, may be configured to use more address space than they require. You can often adjust downward the amount of address space reserved for your PCMCIA devices if you are NOT using an ATA hard drive or a memory card.

The example techniques below have been applied successfully with **CardSoft PCMCIA Card Services Support software** from SystemSoft. Notes follow for **Phoenix PCMCIA Card Services Support**, **AMI PCMCIA Card Services Suppor**t, and **IBM's PCMCIA Card Services**. If your system uses some other vintage of PCMCIA Card Services support software, then consult the manual that came with your system for similar techniques.

CARDSOFT

1) In your current CONFIG.SYS file, remove any EXCLUDE parameters that refer to the D000 page on the QEMM386.SYS line. Then add the parameter X=D000:16K to the end of the QEMM386.SYS line. (Even a 16K EXCLUDE is probably excessive. Most PCMCIA network and modem cards require only a 4K EXCLUDE, and even this is only needed for the PCMCIA diagnostic software to work properly. Some experimentation may be called for, depending the PCMCIA devices in use on your system.)

2) Copy your CONFIG.SYS to another file name so that you can restore it easily later. Create a new CONFIG.SYS file with only QEMM386.SYS and FILES=40 in it. Place only the parameters RAM X=D000:16K on the QEMM386.SYS line in CONFIG.SYS. Don't put any other EXCLUDEs on the QEMM386.SYS line unless you know that they are required for some other aspect of your system. The idea is to get QEMM to create High RAM, but to leave a small range of address space unmapped. This range may have to be increased if it is insufficient to avoid conflicts; conversely, as noted above, the 16K may be reduced to as little as 4K if you are using a device that requires little or no upper memory space (for example, most PCMCIA modems). Copy AUTOEXEC.BAT to another file name. Create a new AUTOEXEC.BAT that preserves only your PROMPT and PATH lines.

3) Copy the PCMCIA software configuration file -- CSALLOC.INI, in our example -- to another name; call it CSALLOC.QDK.

4) Reboot with the minimal CONFIG.SYS and AUTOEXEC.BAT files you created in Step 1.

5) Change to PCMCIA driver directory (in our case, CARDSOFT), and run the PCMCIA configuration utility (in our case, CSALLOC) from the command line. (Some versions of CSALLOC also require a /G switch.)

6) Verify that the CSALLOC.INI file (which will be an ASCII text file) is created with a line that says MEM=D000-D3FF.

7) Restore the original CONFIG.SYS and AUTOEXEC.BAT, and test your PCMCIA devices. In your CONFIG.SYS file, make sure that on your QEMM386.SYS line, you have an EXCLUDE that matches the range that you EXCLUDEd above (e.g. X=D000:16K or X=D000:4K).

PHOENIX TECHNOLOGIES

If your system uses Phoenix Technologies PCMCIA Card Services Support software version 3.x, you may

follow the steps above. The only differences are:

The configuration file is named PCM.INI (instead of CSALLOC.INI).

The setup program is named PCMSETUP.EXE (instead of CSALLOC.EXE).

The default directory name is PCM3.

AMI

If your System has AMI PCMCIA Card Services Support software installed, you may follow the steps above. The only differences are:

The configuration file is named AMICS.CFG (instead of CSALLOC.INI).

The setup program is named AMISRU.EXE (instead of CSALLOC.EXE).

The default directory name is PCMCIA.

IBM

IBM PCMCIA Card Services Support software normally comes with the Thinkpad series of systems. The IBM PCMCIA drivers need the memory address to be specified via the Resource Map Utility, DICRMU01.SYS. This utility loads as a device line in CONFIG.SYS, and is described in the documentation that comes with your computer.

Specify an appropriate address range via the /MA parameter to DICRMU01.SYS. For example, to set up a 16K address range at D000, and assuming that DICRUM0.SYS is in the PCMCIA directory, the DICRMU01.SYS line would look like this:

```
DEVICE=C:\PCMCIA\DICRMU01.SYS /MA=D000-D3FF
```

If DICRUM01.SYS is in another directory, change the path accordingly.

Q. I installed QEMM on my system, and ran OPTIMIZE, and it seemed to identify almost all of my upper memory addresses as High RAM. Is this correct?

A. Probably not. QEMM and OPTIMIZE provide support for drivers that adhere to v2.1 of the PCMCIA specification. While the 2.1 spec is almost two years old as of this writing, many vendors provide drivers that support only PCMCIA v2.0. OPTIMIZE may, on such systems, identify many more areas of adapter RAM than are actually present. To work around this, pass OPTIMIZE the /PCMCIA=xxxx-yyyy parameter, where xxxx is the starting address and yyyy the ending address of the range used by your PCMCIA drivers. The starting and ending addresses can be determined from your configuration files -- in our examples above, CSALLOC.INI, PCM.INI, or AMICS.CFG -- or from the /MA parameter on the DICRUM01.SYS line if you're using IBM's PCMCIA implementation.

For example, if your PCMCIA setup is using 16K at address D000, you should use the OPTIMIZE parameter

/PCMCIA=D000-D3FF

to ensure that OPTIMIZE is aware of the PCMCIA implementation on the machine. Again, you may be able to reduce the size of this range.

Microsoft Windows and QEMM: Advanced Troubleshooting

Quarterdeck Technical Note #207

This Quarterdeck technical note has been written to help you troubleshoot and fix almost all Windows problems which relate to the use of QEMM. If you have an older version of QEMM or Windows, you should upgrade to the current versions before engaging in extensive troubleshooting.

TIPS AND QUICK FIXES FOR COMMON WINDOWS CONFLICTS

Conflict:
A CD-ROM drive will not function properly inside of MS Windows.

Solution:
Edit the AUTOEXEC.BAT file and add "/E" (no quotes) to the end of the line that contains the MSCDEX.EXE driver. This will move the CD-ROM buffers into expanded memory and will reduce the size of the MSCDEX drive, allowing it to be loaded properly and function in MS Windows.

Conflict:
MS Windows video is distorted or unreadable.

Solution:
Using MS Windows SETUP, configure MS Windows to use the standard VGA driver. If this allows MS Windows to display graphics correctly, then QEMM may be recovering parts of the video area that the video driver needs to access. Adding the following parameters to the end of the QEMM386.SYS device line in the CONFIG.SYS file will configure QEMM not to touch the video area on most systems:

```
XST=C000 X=A000-C7FF
```

These parameters may reduce the available upper memory by 64k, so if conventional memory is low, the troubleshooting flowchart (below) should be followed.

Conflict:
MS Windows fails to load, and displays an error message that begins as follows:

"Windows cannot set up an upper memory block at segment B000."

Solution:
This conflict occurs because the video driver that MS Windows is using needs to use the monochrome text area. Edit the CONFIG.SYS file and add the following parameter to the end of the QEMM386.SYS device line:

```
X=B000-B7FF
```

Alternately, you can install MONOUMB2.386 on your system. MONOUMB2.386 is a driver provided with Windows that allows memory managers like QEMM to use the monochrome text area for High RAM, even if the video driver is attempting to access that range. For instructions regarding the installation of MONOUMB2.386, please refer to the README.WRI file included with your Windows installation.

Conflict:
After using MS Windows for a while, the system either locks or a "General Protection Fault" error occurs.

Solution:
MS Windows may be running low on System Resources. Here are some tips to maximize the amount of

free resources:

Do not load fonts that are infrequently used.
Use a small tiled bitmap for the wallpaper instead of a large wallpaper bitmap.
Minimize groups not frequently used, or those with many icons in them.
Because resources cannot be returned once they are used, do not open and close resource-hungry programs.
If you are not loading QEMM 8's Resource Manager, do so by running QEMM Setup. Resource Manager is designed to alleviate resource shortages.
Keep an eye on your system resource with the QEMM memory reporting utility, or the Manifest program, both in your Windows QEMM program group.

Conflict:
MS Windows fails to start properly.

Solution:
Start MS Windows with the "/B" (no quotes) parameter. MS Windows will write any errors encountered while attempting to start to a file in the Windows directory called BOOTLOG.TXT. The BOOTLOG.TXT file lists the files and drivers that MS Windows loads when starting. The success or failure of a driver or file to load as recorded in this file may help you or a technical support representative determine the source of the problem.

TROUBLESHOOTING FLOWCHART

INSTRUCTIONS:
In each of the following steps you will either edit a file or run a program from the DOS prompt. Look for the **???** in each step, as these will guide you through the appropriate troubleshooting procedures. Section One consists of four TESTs, which will identify what is causing the conflict. Section Two consists of four STEPs which will help you resolve the conflict.

This troubleshooting procedure will isolate and resolve the conflict most effectively if you are able to reproduce the conflict at will (you know of a specific action or series of actions that will cause the conflict to occur). The reason for this is that this procedure follows a logical set of tests to determine when a conflict is occuring and when it is not. A conflict that randomly occurs is difficult to troubleshoot because you do not know for sure when a configuration is being used that resolves the conflict.

SECTION ONE: TESTS

TEST 1

CONFIGURING WINDOWS TO AVOID CONFLICTS WITH QEMM: NOEMMDRIVER

Edit the SYSTEM.INI file and locate the section titled [386enh]. Look in this section for a line that looks as follows:

NoEmmDriver=True

??? If this line exists in the [386enh] section, erase the line. QEMM is an expanded memory driver and this line conflicts with MS Windows when using any expanded memory driver. Try to reproduce the conflict.

??? If the conflict goes away, congratulations! You have resolved the conflict and are finished with this technical note.

??? If the conflict still exists then continue with the next step, *MS WINDOWS TROUBLESHOOTING PARAMETERS*.

??? If there is no line in the SYSTEM.INI that reads "NoEmmDriver=True," continue with the next step, *MS WINDOWS TROUBLESHOOTING PARAMETERS*. *MS WINDOWS TROUBLESHOOTING PARAMETERS*

To determine whether a MS Windows setup problem is causing a conflict:

Start Windows 3.1 by typing WIN /D:FSVX

or

Start Windows 95 by rebooting your system, pressing F8 during bootup, and selecting the Safe Mode option.

??? If MS Windows functions properly with the above switches, then adding the appropriate settings to the SYSTEM.INI file will resolve the conflict. Go to STEP A.

??? If the conflict still occurs then the MS Windows configuration is not causing the conflict. You should continue with the next step (WIN.INI).

WIN.INI

It it possible that a driver in the WIN.INI file is causing the conflict. To determine if this is the case, rename the WIN.INI file to WIN.OLD.

Start MS Windows and attempt to reproduce the conflict.

??? If the conflict goes away then go to STEP A.

??? If the conflict still exists then go to TEST 2.

TEST 2

CREATING A CLEAN ENVIRONMENT FOR QEMM

In order to ensure that the MS Windows conflict is not being caused by another program in memory, temporarily disable all lines in the CONFIG.SYS and AUTOEXEC.BAT files that are not a part of QEMM and are not needed to start MS Windows. To do this, place the word REM in front of the line and attempt to reproduce the conflict. Please note that if your hard drive is compressed or requires a driver to be loaded then those drivers should not be REMarked out.

Some conflicts require other drivers or TSRs to be loaded in order to attempt to reproduce the conflict; these drivers should NOT be REMarked out. For example, if the conflict is related to a CD-ROM, the drivers for the CD-ROM must be loaded. These drivers should NOT be REMarked out.

Once the REMark statements have been added to both the CONFIG.SYS and AUTOEXEC.BAT files, reboot the system and attempt to reproduce the conflict.

??? If the conflict goes away, go to STEP B.

??? If the conflict still exists, go to TEST 3.

TEST 3

SIMPLIFYING QEMM

Because QEMM is extremely thorough when recovering and using upper memory, it is possible that one of QEMM's advanced features is adding to the conflict with MS Windows. Please follow steps a) through e) to disable these features.

a) To ensure that there is no conflict in upper memory, you should eliminate all High RAM and test the system. Run QEMM Setup in Windows and choose the option "Fill upper memory with RAM" (In DOS, select "Review or change QEMM parameters", and then the "Fill upper memory with RAM" option. You should select "NO" to ensure that QEMM will not create High RAM.

b) To disable Stealth, select "Stealth system and video ROMs" from the QEMM Features tab in QEMM Setup ("Review or change QEMM parameters" in the DOS version of QSETUP). You will be presented with the following options:

Stealth Off
Stealth Mapping
Stealth Frame

??? If the "Stealth Off" option is highlighted then Stealth is not currently enabled and is not adding to the conflict. Make a note that you were not using Stealth and proceed to step c) below.

??? If either "Stealth Mapping" or "Stealth Frame" are highlighted then you should make a note of the Stealth mode that you were using and highlight the "Stealth Off" option to temporarily disable Stealth.

c) The QEMM Setup program can also place QEMM in a troubleshooting mode, adding many troubleshooting parameters to the QEMM386.SYS device line.

To add the troubleshooting parameters select Troubleshooting Parameters (In DOS, the "Review or change QEMM parameters" QSETUP menu, hit <Page Down> twice and select "Set up QEMM for troubleshooting".) This will add all of the general troubleshooting parameters to the QEMM386.SYS line.

d) QEMM's DOS-Up feature can load more of DOS into upper memory than DOS itself, freeing more conventional memory. It is possible that a program is expecting most of DOS to be in conventional memory, and disabling DOS-Up will eliminate that conflict. In Windows, select "Do not use DOS-Up" from the DOS-Up property page. In DOS, from the QSETUP Main Menu, select "Enable or disable DOS-Up," followed by "No."

e) To disable QEMM's Stealth D*Space feature from the QSETUP main menu, select "Enable or disable Stealth D*Space" and make note of whether you are using Stealth D*Space (if "Yes" is highlighted) or not (if "No" is highlighted). Set this option to "No."

f) To disable QEMM's MagnaRAM, FreeMeg, and Resource Manager, edit the SYSTEM.INI file, and search the [Boot] section for the line:

drivers=freemeg.dll rsrcmgr.dll

and remove "freemeg.dll" and "rsrcmgr.dll" from the line. Then search the [386Enh] section for the line:

device=C:\QEMM\MAGNA31.VXD

and place a semi-colon in front of the word "device". Save the file with the corrections.

After the changes in steps a) through f) have been made, save the configuration and allow the Optimize program to run.

Once Optimize has completed, try to reproduce the conflict.

??? If the conflict goes away then reconfiguring QEMM will resolve it. Go to STEP D below.

??? If the conflict still exists, go to TEST 4.

TEST 4

USING THE DOS MEMORY MANAGERS INSTEAD OF QEMM

If all of the above tests fail to resolve the conflict, there may be an incompatibility with DOS that is not related to QEMM. You need to ensure that the system is functioning properly with ANY memory manager.

The QEMM386.SYS driver should be REMarked out of the CONFIG.SYS to disable QEMM.

Next, add DOS's memory managers, HIMEM.SYS and EMM386.EXE, at the beginning of the CONFIG.SYS as follows:

EXAMPLE:

DEVICE=C:\DOS\HIMEM.SYS DEVICE=C:=DOS\EMM386.EXE RAM 1024 REM DEVICE=C:\QEMM\QEMM386.SYS <troubleshooting parameters>

NOTE: The RAM and 1024 parameters should be added to the EMM386.SYS device line.

Save the configuration, reboot the system, and test to see if the conflict still exists.

??? If the conflict goes away, go to STEP D.

??? If the conflict still exists then it is unrelated to QEMM. Since the conflict exists without QEMM loading, troubleshooting QEMM further will not affect the conflict. We recommend that you contact the manufacturer of the application that is failing, or contact Microsoft if MS Windows itself is not operating correctly. You are finished with this technical note.

SECTION TWO: STEPS FOR RESOLVING THE PROBLEM

STEP A: CONFIGURING MS WINDOWS

??? If the /D:FSVX switch resolved the conflict:

Each of the letters in the parameter "FSVX" represent a different SYSTEM.INI setting:

SWITCH	SYSTEM.INI SETTING
/D:F	32BitDiskAccess=FALSE
/D:S	SystemROMBreakPoint=FALSE
/D:V	VirtualHDIRQ=FALSE
/D:X	EMMExclude=A000-FFFF

Test MS Windows with only one of the above switches at a time to determine the setting that needs to be added to the [386enh] section of the SYSTEM.INI file. Once you have determined the switch that

resolves the conflict, add the corresponding SYSTEM.INI setting the the [386Enh] section of the SYSTEM.INI file. You are finished with this technical note.

??? If renaming WIN.INI to WIN.OLD resolved the conflict:

You have isolated the conflict to something loading in the WIN.INI file. To determine exactly which program is causing the conflict, rename the WIN.OLD file back to WIN.INI, edit the WIN.INI file, and locate the lines that begin with "LOAD=" (in the [Windows] section). Disable the "LOAD=" lines one at a time (by placing a ; at the beginning of the line) and try to reproduce the conflict. When the conflict goes away, the last line that was disabled contains the driver that is causing the conflict. You are finished with this technical note.

STEP B: RESTORING FROM A CLEAN BOOT

One of the drivers or TSRs that you have placed a REM statement in front of is causing the conflict. In order to identify which driver or TSR is causing the conflict you should remove the REMark statements from the beginning of each line in the CONFIG.SYS and AUTOEXEC.BAT files one at a time. By rebooting and trying to reproduce the conflict after each change you can identify the program that is conflicting with MS Windows.

When the conflict reoccurs, the program loading on the last line that you removed the REMark statement from is causing the conflict. Contacting the manufacturer of the program is the easiest way to resolve the conflict; however, Quarterdeck Technical Support has found that the following suggestions resolve many other programs' incompatibilities:

> Configure the program NOT to use expanded memory. Some programs misuse expanded memory, and by configuring them to use extended or conventional memory the conflict with MS Windows may be avoided. Consult the program's documentation for configuration options.
>
> Try loading the program low. Some programs are written with the assumption that they will be loaded in conventional memory and fail to function properly when loaded into upper memory. If the program is able to load itself into uppper memory, disable this feature to ensure that it is not adding to the conflict.
>
> Contact the manufacturer of the program to acquire the latest version of the program. Newer versions may contain compatibility fixes for known conflicts.

You are finished with this technical note.

STEP C: REBUILDING QEMM

HIGH RAM

The first step in rebuilding the system is to allow QEMM to create High RA M. To do this, run QSETUP, select "Review or change QEMM parameters" followed by "Fill upper memory with RAM." Save this configuration, reboot the system, and try to reproduce the conflict.

??? If the conflict does NOT occur then there is no upper memory conflict. Go to the next section, STEALTH.

??? If the conflict reappears then the conflict is related to upper memory. The technical note **EXCLUDE.TEC** will help you isolate the area in upper memory that is causing the conflict and help you correct the problem. You are finished with this technical note.

STEALTH

In this step you will determine if QEMM's STEALTH feature is causing the conflict.

??? If you noted in TEST 3 that Stealth was NOT being used then there is no need to test Stealth. Proceed to the next section, DOS-UP.

??? If you noted in TEST 3 that the Stealth feature was being used then the next step in rebuilding QEMM is to enable Stealth and see if the conflict occurs. Run QSETUP, select "Review or change QEMM parameters" followed by "Stealth system and video ROMs." Select the appropriate Stealth mode based upon the mode that you noted in TEST 3. Save this configuration, reboot the system, and try to reproduce the conflict.

??? If the conflict does not occur, go to the next step, DOS-Up.

??? If the conflict recurs then the conflict is related to Stealth. The Stealth troubleshooting technical note **STEALTH.TEC** should be followed in order to fine tune Stealth to avoid the MS Windows conflict. You have identified the conflict and are finished with this technical note.

DOS-UP

To determine whether the DOS-Up feature is adding to the conflict, you should enable this feature again.

From the QSETUP Main Menu, select "Enable or disable DOS-Up" followed by "Yes." This will fully enable the DOS-Up feature. Save the configuration, allow Optimize to run, and try to reproduce the conflict.

??? If the conflict does not occur then the DOS-Up feature is not adding to the conflict. You may proceed to next step, Stealth D*Space.

??? If the conflict recurs then DOS-Up is adding to the conflict. By partially enabling the DOS-Up feature you can still receive the benefits of DOS-Up while avoiding the conflict with MS Windows.

From the QSETUP main menu, press the "U" key to "Enable or disable DOS-Up," then the "P" key for a "Partial" DOS-Up configuration. This brings you to the DOS-Up Options screen which allows you to turn on or off the loading high of the 4 parts of DOS.

1 = DOS Resources No
2 = COMMAND.COM No
3 = DOS Data No
H = DOS=HIGH No

Test the system by enabling only one of these options at a time, rebooting, and trying to reproduce the conflict after each change to isolate which part of DOS-UP is causing the conflict. Once the option that is causing the conflict is isolated, setting the option to "No" in QSETUP will ensure that this part of DOS-Up will not be enabled by QEMM's Optimize program in the future. You are finished with this technical note.

STEALTH D*SPACE

This test will determine if the Stealth D*Space driver is adding to the MS Windows conflict.

??? If you noted in TEST 3 above that you were using the Stealth Doublespace feature then run QSETUP, select "Enable or disable Stealth D*Space," and set this option to "Yes." Save the configuration, reboot the machine, and try to reproduce the conflict.

??? If the conflict recurs then the Stealth D*Space feature is adding to the conflict. A common cause of conflicts with the Stealth D*Space driver is programs that misuse expanded memory. Try configuring other device drivers and TSRs on the system to NOT use expanded memory. You may need to consult the documentation for the programs that you are loading to learn how to do this.

??? If you noted in TEST 3 that you were NOT using Stealth D*Space OR if setting the Stealth D*Space option to "Yes" did not cause a conflict, the QEMM troubleshooting parameters resolved the conflict. The following is a list of the troubleshooting parameters that QSETUP added:

DB:2, RH:N, SH:N, TM:N, XBDA:N, TR:N, CF:N, FILL:N, MR:N

You may remove one of these parameters at a time, save the file, reboot the system, and attempt to reproduce the conflict. When the conflict occurs then you know that the LAST parameter that you removed was necessary to avoid the conflict and you should keep that parameter and remove the rest. You are finished with this technical note.

STEP D: CONTACTING TECHNICAL SUPPORT

The conflict that MS Windows is experiencing requires further troubleshooting techniques and investigation, and contacting our technical support department is the quickest and easiest way to resolve the conflict. Please mention that you have followed this technical note and were instructed by STEP D to contact Quarterdeck for additional troubleshooting steps.

Our technical support department is easily reached via the following electronic channels:

CompuServe (GO QUARTERDECK)
Internet (mail qsupport@qdeck.com or the comp.os.msdos.desqview Usenet newsgroup)
The Quarterdeck BBS (UK 01245-496943 or Ireland 353 1 2844381)
By fax (UK 01245-496941 or Ireland 353 1 2844144)

When contacting Quarterdeck, be sure to fully explain the symptoms of the conflict, the results of the tests performed while following this technical note, and include the following information:

CONFIG.SYS (in the root directory, usually C:\)
AUTOEXEC.BAT (in the root directory, usually C:\)
SYSTEM.INI (in the MS Windows directory, usually C:\WINDOWS\)
WIN.INI (in the MS Windows directory, usually C:\WINDOWS\)

You can also call our Technical Support line at UK 01245-494940 or Ireland 353 1 2844144 for further assistance. When you call, please be at the machine that is experiencing the conflict.

Return to Technotes Main Menu.

3COM TCPIP 2.1 (3COM)
DECNET (Digital Equipment Corp.)
PC/TCP 2 and 3 (FTP)
LANMAN 2.1 (Microsoft)
EMSNETX and VLM.COM (Novell)

QEMM's Optimize program does not support more than one level of INCLUDE statements in a CONFIG.SYS with multiple configurations. You can use INCLUDE statements with Optimize, but you cannot use an INCLUDE statement inside a CONFIG.SYS block that has already been included in another block.

If you have more than one level of INCLUDE statements, you must edit your CONFIG.SYS file before running Optimize and make sure that all INCLUDE statements below the first level are replaced with the actual CONFIG.SYS lines which the INCLUDE statement formerly invoked.

It is possible to load so many TSRs and device drivers on your system that you may run out of memory during the OPTIMIZE process!

Please refer to the technical note **MAXMEM.TEC** for suggestions on improving your pre-OPTIMIZE configuration. In cases where OPTIMIZE does not complete successfully, you may wish to try taking advantage of expanded memory by skipping the hardware detection phase as noted above.

OPTIMIZE's Stealth Testing process is the best way to ensure both maximum memory and maximum compatibility with your system. To take advantage of this feature, make sure that you're starting from a stable, bootable QEMM configuration. Typically the line "DEVICE=C:\QEMM\QEMM386.SYS ON" will allow you to boot your system safely. From this, you may run OPTIMIZE's Stealth Testing procedure as follows:

1) At the DOS prompt, type "OPTIMIZE /REMOVEALL" to remove LOADHI commands and parameters from all of the lines in CONFIG.SYS, AUTOEXEC.BAT, and any batch files called from AUTOEXEC.BAT.

2) Edit CONFIG.SYS and remove all parameters except ON from the QEMM386.SYS line in CONFIG.SYS.

3) At the DOS prompt, type "OPTIMIZE /ST".

The Stealth Testing process provides maximum compatibility with your system, at the possible expense of some High RAM or conventional memory. If this is the case, see the previous topic **LESS CONVENTIONAL MEMORY AVAILABLE**.

PAGEOVERCOMMIT

The PAGEOVERCOMMIT=n setting appears in the [386Enh] section of the Windows 3.1 SYSTEM.INI file. It determines the size of linear memory, which is the address space that Windows programs see when they allocate memory from Windows.

If the PAGEOVERCOMMIT statement does not appear in the SYSTEM.INI file, Windows 3.1 defaults to a value of 4, which means that the size of linear memory will be four times the amount of physical memory (installed RAM) on your machine. (The size of linear memory is fixed at two gigabytes in Windows 95, and the PAGEOVERCOMMIT statement is no longer needed or used.) QEMM will normally set PAGEOVERCOMMIT to 8, which is twice Windows 3.1's default. The allowable values of PAGEOVERCOMMIT are 1 through 20.

If the value of PAGEOVERCOMMIT is too small, then some of the physical or virtual memory on your system will go to waste, because there will be no addresses for Windows 3.1 to give out to programs, even if the memory is available. If PAGEOVERCOMMIT is too big, however, Windows 3.1 wastes precious physical memory to keep track of unneeded linear memory addresses.

The best way for you to determine how big PAGEOVERCOMMIT should be is to monitor MagnaRAM's performance on the QEMM screen. Open a number of programs until physical and virtual memory have been almost used up, and then check the amount of available linear memory. If there is quite a bit of linear memory left, try decreasing PAGEOVERCOMMIT. You want to arrive at a value of PAGEOVERCOMMIT that leaves some available linear memory behind when physical and virtual memory are used up, but does not leave too much.

There is a technical note included in this Help file on the subject of maximizing memory while using **PCMCIA** adapters.

Parity Errors

Quarterdeck Technical Note #128

Q. What is a parity error?

A. The memory controller chip on your PC reports a parity error when it reads a byte of data and the 9 bits it used to encode the byte do not add up to 1 (odd parity). Parity errors are always hardware-related. Software applications cannot cause parity errors, although an application may cause one to be detected.

In the digital world, all information is represented by the binary numbers 0 and 1. The binary digit, or bit, is the fundamental building block of digital information in a computer, and it stores information in two states: off or on (0 or 1, respectively). One bit can make a big difference. Here's why:

The binary number for the letter U is:

01010101

If you change just the fourth bit over from the left, from one state to the other, the binary number becomes the letter E.

01000101

Now while there are 8 bits in a byte, your memory controller handles information 9 bits at a time. This extra bit is called a "parity bit", and is the computer's way to verify the integrity of your data. Whenever you write data to memory, the memory controller adds up the number of 1's in each byte of information, and then sets the ninth bit to make the sum of all nine bytes odd. IBM, the original designers of the PC, could have chosen to make the sum of the nine bytes even (even parity), but they chose to store data in memory with odd parity and every other PC manufacturer followed suit.

In the example above, the letter U has the binary value of 01010101, which has 4 1s in it, and the letter E is 01000101, which has 3 1s in it. When your PC reads each byte of data, it sums the 9 bits to make sure the number of 1s in the byte is still odd. If the state of a single bit gets changed from 1 to 0, or 0 to 1, the parity of the nine bits becomes even and the memory controller asserts the NMI (Non-Maskable Interrupt). This signal is put directly on a pin of the CPU, then the code pointed at by Interrupt 2 posts a Parity Error message, which warns you that there is a problem with your RAM.

Q. Why am I getting parity errors on my system since I installed QEMM?

A. As stated above, parity errors are indicative of a hardware problem. The error may appear since you've installed QEMM because QEMM gives you and your applications access to memory that may never have been used before, and which could be marginally bad.

Q. How do I determine which piece of hardware is causing the problem?

A. First, check is the RAM in your system. An easy test is to disable everything that uses EMS and XMS memory so you can create a RAMDRIVE the size of all your system memory. (Refer to your DOS manual for information on creating a RAMDRIVE.)

Then:

a) Run CHKDSK on the RAMDRIVE

or

b) Copy files to the RAMDRIVE until it is full.

Either way, if you have bad memory on your system, eventually you will get a parity error or a General Drive Failure on the RAMDRIVE.

The first thing you can do to try to remedy this problem is to make sure that the RAM chips are seated properly in their sockets. If they are DRAMs or SIPPs, make sure the pins aren't broken off or bent. If they are SIMMs or the memory is on a card, you may just need to clean the contacts. If the chips physically check out ok, the chip speeds could be mismatched with memory that is too slow for the CPU/memory bus, or a controller chip could be bad. At this point the only sure way to test this is to swap out the chips for ones that you know are good.

Parity errors may also be caused by the presence of an autoswitching video card or one that is using 16-bit ROM access. Your motherboard could be assigning parity to the address space where your EMS page frame is located. Also there may be some special features of the computer in the CMOS Setup that could be causing problems. Try disabling the computer's shadowing of BIOS or video ROM or turning off memory caching or other features to see if one of them is involved. This may allow you to pinpoint the cause of the problem. In all these cases you should refer to the documentation that came with your hardware product to disable a particular feature.

Q. I ran a hardware diagnostic program on my machine, and it didn't report bad memory. Why not?

A. While there are several diagnostic programs on the market that will test your memory for errors, they may not duplicate conditions that would cause marginal memory to fail. Most are not even designed to be run with a memory manager. When parity errors are encountered, it is time to have the hardware components of the machine examined.

Q. Is there any software I can use to get around this problem?

A. No. Note that all of the parity operations are performed directly by your computer's hardware, regardless of which operating system (DOS, OS/2, UNIX) you use and regardless of which utility programs or application software you run. One exception is Macintosh computers, which use 8 bit SIMM chips that do not have parity. When errors occur, the system just malfunctions from the invalid data. Also remember that parity checking will only detect if one bit in a byte gets changed. If two bits in the same byte get changed it will accurately reflect that the sum is still odd and errors will not be detected.

Return to Technotes Main Menu.

Product Compatibility Information

Quarterdeck Technical Note #248

The following is a list of various hardware and software which our Testing and Compatibility Department has determined requires special attention in order to be compatible with QEMM and/or StealthROM. These notes are as accurate as possible at the time of writing, but as technology advances, this information may change from time to time. Also note that when an entry states that excludes within a certain area are needed, you can use QEMM's ANALYSIS procedure to determine the exact areas that need to be excluded from QEMM's use. See Chapter 9 of your QEMM manual for details.

HARDWARE

MISCELLANEOUS LAPTOPS

After a QuickBoot, laptops with Advanced Power Management will assume their default power-on speed. If you note that your laptop is running faster or slower than you expect after a QuickBoot, check the default power-on configuration settings, and adjust them accordingly.

ACER 1120SX

This system may need exclusions in the F000-FFFF area if you are using the StealthROM feature. On the particular systems that Quarterdeck tested, the parameters X=F300-F3FF and X=FA00-FAFF were needed with StealthROM. Current versions of OPTIMIZE will likely handle this machine more elegantly; or your system may require different exclusions. In addition, the Acer 1120sx may need X=C600-C7FF on the QEMM device line even without StealthROM.

ADAPTEC ASPI DRIVERS

If the ASPI4DOS.SYS or ASPI2DOS.SYS device driver is loaded before QEMM386.SYS, you may need to exclude up to 12K of the F000-FFFF range when using QEMM's Stealth function. This problem does not occur if ASPI4DOS is loaded after QEMM, but in this case QEMM will use 2K of conventional memory for a disk buffer to prevent bus-mastering problems when ASPI4DOS loads high. For more information on bus-mastering controllers, see BUS-MAST.TEC, included in the QEMM\TECHNOTE directory.

AT&T GLOBALYST

The AT&T Globalyst 360 Pentium (and potentially other machines with recent Award BIOSes) have large ROMs in the E000 region. OPTIMIZE's Stealth Testing process will complete without user intervention, but will not Stealth all of the ROMs on the machine. Much of this ROM is INCLUDEable through the QEMM Analysis procedure if you add the parameter

```
S=EF00:4K
```

to the end of the QEMM386.SYS line in CONFIG.SYS. The unStealthed ROMs have no interrupts pointing into them; see the technote **STLTECH.TEC** for an explanation.

ATI LOCAL BUS VIDEO CARDS

Current versions of QEMM include special support for ATI video cards, so exclusions required in the past may not be necessary.

COMPAQ LAPTOPS AND NOTEBOOKS WITH PCMSMIX.EXE

Some Compaq laptops and notebooks come with a program called PCMSMIX.EXE, the purpose of which is to allow the machine to receive incoming faxes while in power-saving mode. This program comes into conflict with QEMM's QuickBoot feature. For this reason, OPTIMIZE disables QuickBoot whenever

PCMSMIX is loaded in CONFIG.SYS.

COMPAQ SYSTEMS
If your Compaq machine does not recognize memory above the 16MB line, add the parameter

USERAM=1M:xxM

where xx is the amount of memory that you have on your machine. See the information on the USERAM parameter above for more details.

COMPAQ XL SYSTEMS AND PCNTNW.COM
Compaq XL desktop machines ship with a program called PCNTNW.COM, which is a driver for the built-in network interface hardware on XL systems. It is reported that the Compaq network hardware is bus-mastering, and that PCNTNW makes VDS calls to eliminate potential conflicts between the network hardware and memory managers like QEMM. This means that it may be necessary to make PCNTNW load low, by adding the word PCNTNW to the OPTIMIZE.NOT file in your QEMM directory. If no such file exists, create one that contains the single line PCNTNW. See Chapter 3 of the Reference Manual for more information on OPTIMIZE.NOT.

There are also unconfirmed reports that the QEMM parameter EXCLUDE=F600-FFFF is needed on some Compaq XL systems.

DEC CELEBRIS and VENTURIS
These machines may be affected by the "Plug and Play BIOS Machines" anomalies described below.

GATEWAY 2000
Some models of Gateway machines may be affected by the "Phoenix Green BIOS" anomalies described below. Others may have an STB PowerGraph 64 PCI video card, also noted below.

GRAVIS ULTRASOUND
The MegaEM emulator for Gravis Ultrasound may require the P:VME:N parameter to be added to the QEMM386.SYS line in CONFIG.SYS.

HEWLETT PACKARD OMNIBOOK 600
This machine will report an Exception #13 at startup until you apply the following information. QEMM will detect that the B000-B7FF and C000-EBFF regions on this machine contain Adapter RAM, and thus will not create High RAM in these areas by default. However, you may INCLUDE these regions by adding I= parameters to the QEMM386.SYS line in CONFIG.SYS. Power management routines exist in the EE00-EEFF region, and this area must be EXCLUDEd. Finally, the address range used by any PCMCIA hardware on the system must also be EXCLUDEd. The range in question depends on your PCMCIA configuration; refer to the PCMCIA.TEC technote in your QEMM directory for further information on determining the range appropriate for your system. A typical QEMM386.SYS line for an OmniBook will include these parameters

I=B000:32K I=C000:176K X=EE00:4K X=D000:8K

on the QEMM386.SYS line in CONFIG.SYS, where the starting point and the size of the last parameter will vary depending on your PCMCIA setup.

HEWLETT PACKARD VECTRA
Some models of this machine may require the QEMM386.SYS parameter S=F400:4K to support graphics modes properly when Stealth is active. Some may also require the parameter X=F000:8K for the Sleep mode to work properly.

IBM PS/2 MODEL L40SX

This laptop may need the following excludes in order to work with ST:M:

X=E000-E0FF X=E200-E3FF X=E600-E6FF.

The ST:F method requires no excludes. The Suspend/Resume feature works on this system automatically.

IBM TOKEN RING CARDS

Users of PS/2's with Token-Ring cards and QEMM may observe that conventional memory ends at 576K rather than 640K. A Token-Ring network card has both an adapter RAM and ROM in upper memory, either 8 or 16K in size. The default addresses for the RAM and ROM are D800 and CC00, respectively. This default configuration may not allow room in upper memory for the EMS page frame, especially on PS/2 systems. If the page frame does not fit in upper memory, QEMM will place the page frame in the last 64K of conventional memory, decreasing the memory in which programs can run. A message will be posted during boot up if the page frame has been placed in conventional memory.

On a PS/2, you can resolve this problem by using your PS/2 reference diskette to move the Token-Ring adapter RAM and ROM to one end or the other of available upper memory. The idea is to create a contiguous 64K area for the page frame rather than having your upper memory addresses broken up into small unusable regions. Moving both RAM and ROM as low as possible in the C000 area is usually a good choice. On non-PS/2 system, if the address ranges of the card are movable, contiguous address ranges starting at D000 are often a good choice.

If you use QEMM's StealthROM feature, the page frame will usually remain in upper memory even with the Token-Ring card hogging the address space because StealthROM clears the area of the ROMs between E000 and FFFF. However, in order to get the maximum memory available, you should still make the contiguous free areas in upper memory as large as possible by moving the adapter RAM and ROM to different locations.

When you use the StealthROM mapping method (ST:M) the default Token-Ring configuration will force the page frame to go to EC00, whereas QEMM would normally try to maximize memory by placing the page frame at C000. You can put the page frame at C000 by using the reference diskette to move the Adapter RAM and ROMs out of the C000-CFFF range. In this case, the best places to put the Token-Ring adapter RAM and ROM are probably at the beginning of the D000 area, as close to each other as possible.

INVISIBLE NETWORK

If you use the boot ROM on the Invisible Network card, it loads 32K of code into the top of the conventional memory address space and grabs interrupt 13. A much better solution than to use XSTI=13 and an appropriate exclude is to disable the ROM on the network card and load IS2BIOS instead. This will give you 32K of conventional memory (since IS2BIOS can be loaded high) and you will not have the network card's ROM breaking up your upper memory address space.

MICRO ELECTRONICS WINBOOK

Our testing has shown that on a certain call to the APM (Advanced Power Management) BIOS routines on the Micro Electronics WinBook XPS, the keyboard controller may send an escape key, which will cause the exit screen to appear, or which may cause an apparently locked keyboard. The latter symptom may be remedied by moving the
mouse while holding down a key. Micro Electronics has been informed of the problem.

NEC VERSA

This machine may be affected by the "Plug and Play BIOS Machines" section described below.

ORCHID TECHNOLOGY KELVIN 64 VIDEO CARD

The Kelvin EZ Setup utility that comes with this card permits video resolution switching while inside Microsoft Windows. This utility, when Stealth ROM is active, requires the EMS page frame to be placed

at C000, or requires the SVGA:256 parameter if the page frame is not at C000.

PHOENIX PCMCIA CARD MANAGER

The Phoenix Technologies PCMCIA Card Manager software includes a driver called CNFIGNAM.EXE that, in version 3.0 of the Card Manager (and perhaps in other versions), will give an error message and refuse to load if Optimize attempts to load it high. To prevent this problem, add a new line containing only the word CNFIGNAM to the OPTIMIZE.NOT file in your QEMM directory. If no such file exists, create one that contains the single line CNFIGNAM. See Chapter 3 of the Reference Manual for more information on OPTIMIZE.NOT.

PLUG AND PLAY BIOS MACHINES

The Plug and Play configuration driver DWCFGMG.SYS conflicts with OPTIMIZE and, in our experience, with MS-DOS's memory management utilties as well. By default, the OPTIMIZE.NOT file contains an entry which will cause the driver to load low, which we recommend. Unless you are using Plug and Play compatible hardware, you may consider removing this driver altogether.

PLUS IMPULSE/HARDCARD II

These hard drives may need the XST parameter applied to their ROM(s) when using Stealth. The default location for a Hardcard II ROM, for example, is C800. In this case adding XST=C800 to the QEMM386.SYS device line may be necessary. If you are unsure of the location of your Hardcard II or Impulse disk ROM, consult the documentation that accompanied your drive.

PSI HYPERSTOR 816/1600 HARD DISK CONTROLLER

Some versions of this controller may require that the page frame be located at the beginning of the controller's ROM (which is often at C800.) A small exclusion in the F000-FFFF range may also be necessary; use the QEMM Analysis procedure.

QLOGIC FAST SCSI CONTROLLER

Certain models of this controller may require the XBDA:N parameter to QEMM, and may also require the parameter XST=nnnn, where nnnn is the address of the ROM on the controller (typically C800 by default).

SETUP PROGRAMS, SETUP HOTKEYS

On some machines it is possible to access the computer's setup program at any time by pressing a hotkey. Other machines provide software programs for system configuration. On many of these systems you must EXCLUDE some portion of the F000-FFFF range in order to use these programs when QEMM's StealthROM feature is enabled, or treat the entry point to these programs with an S= parameter. Hotkey-based setups usually work without exclusions if you are using the ST:F feature. The most practical way to deal with this problem is to avoid EXCLUDEs and to prevent QEMM from loading on the rare occasions when you need to access your system setup program. If you prefer to sacrifice High RAM areas in order to run the system setup with StealthROM active, you can use QEMM's Analysis procedure to determine the areas you must exclude, providing that a reboot is not forced when exiting the setup program.

OPTIMIZE may be able to help you find an appropriate S= parameter to permit running the system setup, at a lower cost in High RAM. To find the right parameter, end the AUTOEXEC.BAT file with the command

```
PAUSE
```

and when your system pauses at the end of AUTOEXEC.BAT during the Software Detection Phase, press the hotkey combination for your system's Setup program. If possilble, exit the setup program by returning to DOS, rather than using the option to save settings and reboot. OPTIMIZE will regain control of the system, and should detect the appropriate S= parameter.

SMC NETWORK CARDS AND THE SMCPWR.COM DRIVER

Some SMC network cards ship with a driver called SMCPWR.COM. In some cases, this card may exhibit incompatibilities with QuickBoot, such that the machine may hang instead of resetting on a soft reboot. If you experience this problem, you may choose simply to reboot via the power switch, or to disable QuickBoot via the BE:N parameter on the QEMM386.SYS line in CONFIG.SYS.

STB 800/16 VGA CARD

This graphics card works well with the page frame at C000 when the StealthROM mapping method (ST:M) is in effect. QEMM places the page frame at C000 by default when ST:M is enabled. If ST:M is enabled and the page frame is not at C000, it is likely that some obstruction is preventing QEMM from putting the page frame there safely. In this case, you may need to exclude a portion of the C000-C7FF area.

STB POWERGRAPH VIDEO CARDS

On certain systems with shadow RAM, we have observed a conflict between the STB PowerGraph video card and systems with hardware ROM shadowing. QEMM does not cause the problem, but helps to expose it. This problem causes QEMM to report that it cannot find the ROM handler for INT 10 and to disable the StealthROM feature. The best workaround is to disable hardware video ROM shadowing on the motherboard of such systems, and to use QEMM's ROM parameter instead. Note that QEMM's ROM parameter can provide similar functionality, and write-protects the ROM in the process, avoiding the problem. An alternative is to disable QEMM's use of Shadow RAM with QEMM's SH:N parameter.

TOSHIBA LAPTOPS, POP-UP MENU

A feature of various Toshiba laptop computers is a pop-up menu that displays information on the status of the computer's battery. In order for this pop-up menu to work when the computer is in Virtual 8086 mode (that is, when QEMM is providing expanded memory or High RAM) a TSR called T386.EXE must be run. This tiny program and its accompanying doc file (T386.DOC) are included on the QEMM diskettes.

TOSHIBA 4400SXC

If you use the battery pop-up feature of this system, you may need the QEMM parameters X=F400-F7FF and X=FC00-FFFF. You may be able to narrow these excludes somewhat. (The T386.EXE file mentioned in the previous section may be needed for the battery pop-up feature to work.)

TOSHIBA 5100

This computer may be incompatible with the StealthROM mapping method (ST:M). OPTIMIZE should automatically detect and work around any such incompatibility.

ULTRASTOR DISK CONTROLLER

If you have an UltraStor SCSI disk controller and you are using QEMM's DOS-Up feature and you see a "device not found" message during bootup, you may be able to fix this problem with the FIXINT13.SYS driver which accompanies QEMM. This driver is misnamed INTFIX13.SYS in the QEMM manual. Load the driver as the line immediately before QEMM386.SYS in CONFIG.SYS, and add the /STACKSIZE parameter. For example:

```
DEVICE=C:\QEMM\FIXINT13.SYS
DEVICE=C:\QEMM\QEMM386.SYS RAM <your other parameters>
```

Some other SCSI disk controllers may also be fixed by this driver.

WESTERN DIGITAL SUPER VGA CHIPSETS AND WINDOWS DRIVERS

In at least one version of the Microsoft Windows Super VGA drivers for cards using the Western Digital WD90C11 chipset, the address range from B000-B7FF must be EXCLUDEd on the QEMM386.SYS line. Example:

```
DEVICE=C:\QEMM\QEMM386.SYS ST:M RAM X=B000-B7FF
```

ZENITH PC's
With some versions of DOS, you need the parameter XSTI=18 in order to print on a Zenith system when StealthROM is enabled. You will also need a small 4K exclusion somewhere in the F000-FFFF range. X=F500-F5FF works on some systems.

SOFTWARE

MISCELLANEOUS DRIVERS
QEMM's installation program adds several drivers by default to the OPTIMIZE.NOT file. This is because, for one reason or another, these drivers may not be loaded high, or may interfere with the running of OPTIMIZE. If you feel that your version of a given driver may load high successfully, even though it appears by default in OPTIMIZE.NOT, remove the line referring to your driver with an ASCII editor, or with a word processor in DOS Text mode.

The following programs are currently added to OPTIMIZE.NOT: CNFIGNAM, DPMS, MTDDRV, MINI406A, CASCMOD1, PCNTNW, MEMDRV, AUTODRV.

1DIR PLUS
Some versions of this program need the QEMM parameter UFP:N when Stealth is in effect if 1DIR Plus is using EMS. Another solution is to configure 1DIR Plus so that it does not put its stacks in the EMS page frame. (See the 1DIR Plus manual for details.)

ALLCLEAR
This charting software may need an exclusion in the C000-C7FF region if you use its View Chart or Print Preview options with StealthROM.

AVERY LABEL PRO
Some versions of this software may put display characters incorrectly on the screen when the StealthROM feature is enabled unless you use the X=F000-F0FF parameter.

BOOTCON
Bootcon (version 2.02) is a utility that allows a user to boot different configurations without having to constantly edit config files.

Bootcon is compatible with QEMM's Optimize program, but you must run Bootcon in STANDALONE mode to achieve this. This mode disables the MENU mode and boots the system with a single or flat configuration. Each configuration that is to be Optimized has to be booted as a Standalone.

To change the Bootcon program from Menu to Standalone, run the BCSETUP program, go to the Main Menu, and select SET MODE from the menu.

After the completion of all Optimizes, one may go back to MENU mode so that each time the system boots, one may select a configuration from the menu. The MENU mode is more akin to the DOS 6 multi-config setup.

BTRIEVE DATABASE PROGRAMS
Including DAC EASY, CLARION, AND pcANYWHERE
Btrieve is a database record manager sold by Novell and used by many applications to perform database activities. Btrieve is usually run before these applications as a TSR. It uses expanded memory, unless you prevent it from doing so by giving it the /E parameter (the E must be uppercased). Quarterdeck has seen many cases in which systems did not function properly unless Btrieve was stopped from using expanded memory with the /E parameter.

Among the applications that have used Btrieve in at least some of their versions are DAC Easy, Clarion, and pcANYWHERE.

- We have reports that DAC Easy versions 4 and 5 will fail when used with QEMM's D*Space feature unless Btrieve is using the /E parameter. In DAC Easy 5, the symptom is often a DoubleGuard alarm error when DAC Easy starts. DAC Easy loads Btrieve from a batch file called DEA4.BAT or DEA5.BAT; if this batch file does not already specify the /E parameter on its Btrieve line, you should place it there.

- Clarion 3 (and perhaps other versions) loads Btrieve from a batch file called CDD.BAT. If this batch file does not already specify the /E parameter on its Btrieve line, you should place it there to prevent failures when using Clarion with one of QEMM's Stealth features.

- pcANYWHERE 4 (and perhaps other versions) loads Btrieve automatically with its AWHOST program. To make sure that Btrieve does not use expanded memory, you must load Btrieve with the /E parameter, either manually or from a batch file, before loading AWHOST. AWHOST will see that Btrieve is already loaded and will use the already-active copy in memory.

CACHE86 (FROM THE ALDRIDGE COMPANY)

When using Cache 86's expanded memory cache with Stealth DoubleSpace, you must specify the EXPCACHE parameter to the ST-DSPC driver in CONFIG.SYS. Example:

```
DEVICE=C:\QEMM\ST-DSPC.SYS /EXPCACHE:4
```

Cache86, when using EMS, is also incompatible with OPTIMIZE's SqueezeTemp feature. The alternatives are to ensure that Cache86 is not using expanded memory, or to start OPTIMIZE with the /NT parameter:

```
OPTIMIZE /NT
```

CENTRAL POINT ANTI-VIRUS

As of this writing, it may be the case that a hang or a reboot will occur when Central Point Anti-Virus is used with QEMM and Windows for WorkGroups 3.11, and when a floppy or networ drive is scanned. Symantec (publishers of CPAV) and Quarterdeck are working co-operatively to resolve this problem. More up-to-date information may be available via the online support forums for both companies.

DELRINA DOSFAX

Delrina's DOSFAX program requires that you place the parameter X=B000-B0FF on the QEMM386.SYS line in your CONFIG.SYS file. Without this parameter, DOSFAX may fail after it captures a document to print.

GEOWORKS ENSEMBLE

As of version 1.2, Geoworks is incompatible with QEMM's Stealth ROM feature if Geoworks Ensemble is set up to use expanded (EMS) memory. If you set up Geoworks Ensemble to use extended (XMS) and conventional memory, it will work with StealthROM. See the accompanying documentation for details on configuring Geoworks to use XMS and conventional memory. (As of this writing, the recently released GEOWORKS v2.0 has not been tested to determine whether this information applies to that version, as well.)

GLYPHIX

Some versions of the font program Glyphix need the QEMM parameter UFP:N when Stealth is in effect if Glyphix is using EMS.

IBM PC-DOS 6.1

A quirk in PC-DOS 6.1's handling of CALLed batch files can cause problems for OPTIMIZE. In a CALLed batch file in most versions of DOS, a GOTO <label> statement for which there is no valid

destination <label> will cause the current batch file to terminate, and to return control to the batch file that CALLed it. In PC-DOS 6.1, a GOTO <label> statement will cause ALL batch files to terminate if there is no valid <label>. This will cause the OPTIMIZE process to terminate abnormally. There are two workarounds. One is to ensure that all GOTO statements point to valid desinations; the other is to upgrade your version of PC-DOS to any later version, including 6.3 and 7.0.

It is possible that installing from inside Microsoft Windows, and then choosing to run OPTIMIZE from QSetup in Windows may cause the message "invalid COMMAND.COM, system halted". In this case, reboot your system, verify that all GOTOs have valid destinations, and run OPTIMIZE from DOS.

INFINITE DISK

The QEMM directory (and especially any DOS device drivers or Windows .VxDs within it) should not be compressed by Infinite Disk, since the drivers might be required at boot time or during the OPTIMIZE process before the Infinite Disk software loads. Make sure that the files in the QEMM directory are uncompressed,
and then type the command

```
C:\INFINITE\PROTECT C:\QEMM\*.* /P
```

to protect the files from being compressed.

LANTASTIC 6.0

Two of the network drivers that ship with LANtastic 6.0, SERVER.EXE and REDIR.EXE, load into memory in such a way that QEMM's Optimize program assumes that they are larger than they in fact are. As a result, Optimize usually loads these programs low.

If your copy of LANtastic 6.0 contains versions of SERVER and REDIR that accept the /LOAD_HIGH parameter, you should specify this parameter to both LANtastic drivers. You should also make sure that the DOS=UMB or DOS=HIGH,UMB statement is in your CONFIG.SYS file; LANtastic requires the DOS=UMB interface (available in DOS 5 and later versions) in order to use upper memory. The /LOAD_HIGH parameter and DOS=UMB will allow Optimize to load SERVER and REDIR high if there is enough room for them in upper memory.

If your versions of SERVER and REDIR do not accept the /LOAD_HIGH parameter, you can still attempt to load them high by performing the following steps, although you will need more available High RAM than you would with the /LOAD_HIGH versions of the drivers:

1. For the best changes of loading the drivers high, reorder your AUTOEXEC.BAT file if necessary so that as many TSRs as possible are loaded after SERVER and REDIR.

2. Start Optimize and select Custom Optimize from the first Optimize screen.

3. Proceed with Optimize until you reach the Analysis Phase. When the Analysis Phase is complete, select O for Options on the Analysis Phase screen.

4. Select option 2 to modify the data used by the Optimize process.

5. Find SERVER and REDIR on the list of programs, use the arrow keys to go to the "Try to Load High?" field for these programs, and set the field to Y for both programs. Hit Enter to save each change.

6. Use the arrow keys to go to the Initial Size fields for the two programs. Enter 122880 in the Initial Size field for SERVER, and 61440 in the Initial Size field for REDIR. Hit Enter to save each change.

7. Hit Enter again. After Optimize recalculates, continue as usual with the Optimize process.

LOTUS 1-2-3

If you are using QEMM's VIDRAM feature, Lotus 1-2-3 may report that "123 cannot start because the driver set is invalid." VIDRAM works with DOS text-based programs, but does not allow EGA or VGA graphics. 1-2-3 is checking your graphics card's capabilities and VIDRAM is telling it that no graphics are allowed. The solution is to run 1-2-3's INSTALL program and make a driver set with no graphics entry. Use 1-2-3 INSTALL's "Advanced Options" and "Modify Current Driver Set" selections. Then select the "Graph Display" item. Press the Del key on the driver that is currently selected. Then press the "Esc" key and use "Save Changes" to save the driver set with a different name (we suggest 123VID.) Then when you want to use 1-2-3 with VIDRAM, type 123 123VID at the DOS prompt and the correct video driver will be used.

DESQview users may want to install a second version of 1-2-3 on the DESQview menu. The second version would include 123VID as a command line parameter to the 123 command.

LOGITECH MOUSE DRIVERS (LVESA.OVL)

Some versions of Logitech MOUSE.COM drivers load an overlay file (LVESA.OVL) that Optimize does not detect, and consequently the driver does not load high. A message appears saying "Insufficient Memory To Load Video Module" or "Error: Not enough memory to load Video Module" when LOADHI attempts to load the mouse driver high. Updated mouse drivers that do not exhibit this problem are available from Logitech. Alternatively, you may choose to remove the line

VideoModule=LVESA.OVL

from the LMOUSE.INI file, or use the NOVCI switch to the Logitech mouse driver.

MIRROR

MIRROR is written by Central Point Software and packaged with MS- and IBM-DOS version 5 and 6. MIRROR is used to recover deleted files. MIRROR first makes a backup copy of the your FATs (File Allocation Tables), then loads a resident portion of itself that tracks files as they are deleted in order to expedite their recovery. The file tracking feature is enabled by using the "/Tx" switch (where "x" is the letter of the drive to be monitored) to the MIRROR command line.

The copy of the FAT(s) that MIRROR makes may be too large to load into available High RAM. (This data cannot be spread over multiple High RAM regions.) If this happens when MIRROR loads, it will report that it has failed to perform this function. However, the undelete tracking feature may have installed successfully. Type LOADHI at the DOS prompt to make sure that MIRROR loaded successfully.

If there is insufficient High RAM to perform the first function of MIRROR above 640K, but there is enough High RAM to perform MIRROR's second function (the resident portion of MIRROR requires only 6.4K of memory), you may load MIRROR low once without the "/Tx" switch (to perform MIRROR's first function.) Then load MIRROR high with the "/Tx" switch in order to load its resident portion above 640K and make a successful copy of the FAT.

NORTON ANTI-VIRUS

Norton Anti-Virus version 2.00 is known to interfere with the ability of LOADHI.COM to load the command processor. Upgrade your software to NAV 2.1 or higher. In the interim, use QSETUP to load the command processor low by choosing DOS-Up Options from the main menu, and then Partial; on the following screen, set COMMAND.COM
to No.

NORTON BACKUP

If you frequently change your configuration from StealthROM enabled to StealthROM disabled, some versions of Norton Backup may require that you exclude X=FE00-FFFF and that you reconfigure the

backup program.

ORACLE AND VCPI
Oracle is a VCPI-compliant program, starting with version 2.1.34 of the SQLPME.EXE file. You may want to contact Oracle to find out the status of the VCPI support of your version. It is also important to choose the Oracle configuration option (machine type J) that tells Oracle that it is running on a VCPI system.

PCSA
PCSA's EMS loaders (DMNETHLD and EMSLOAD) do not work if Stealth ROM is enabled. The QEMM386.SYS parameter XST=F000 may solve the problem when it occurs. Some DEPCA cards may fail with the PCSA software and ST:M unless you place the page frame at the starting address of the DEPCA's card's 16K ROM.

PRINTQ
You should use this print spooler's /LSX parameter to make it use extended memory rather than expanded memory if you are using StealthROM.

REPEAT PERFORMANCE
Like other keyboard-enhancement programs that create a new type- ahead buffer, the Repeat Performance keyboard-enhancing program malfunctions if loaded above 63K. As a result, it cannot be loaded high with all of its features enabled. However, RP.SYS will load high if you use its BUFFERS=OFF parameter, which disables Repeat Performance's type-ahead buffer.

SPACEMANAGER
If your are using SpaceManager's SuperMount feature, DOS 6.0 and QEMM's StealthROM feature, your PC may hang at bootup time. (If you want to find out if you are using SuperMount, look for the SMOUNT or SMOUNT.EXE command in your AUTOEXEC.BAT file.) To fix the bootup problem, add the following parameter to the QEMM386.SYS device line in your CONFIG.SYS file: DBF=n (where n is a number; 1 and 2 are commonly-used values). For information on the DBF parameter, see DISKBUFFRAME in Chapter 7 of the QEMM manual.

SIDEKICK PLUS
SideKick Plus will not work with StealthROM unless it is prevented from using EMS. One workaround is to use QEMM's EMS.COM program to temporarily allocate all EMS before SKPLUS is loaded, then use EMS.COM again to free your machine's EMS memory after loading SKPLUS.

SUPER PC-KWIK
When Super PC-Kwik is using expanded memory and you are using Stealth D*Space and do not have StealthROM enabled, you must use the Super PC-Kwik parameter, EMSMapSaves=Always, which forces Super PC-Kwik to make the necessary EMS calls to be compatible with Stealth D*Space.

TALKING ICONS (Aristosoft)
The Talking Icons FX function can cause video display refresh problems when used with QEMM and Windows. It is recommended that the FX function not be used.

VENTURA PUBLISHER PROFESSIONAL
When QEMM's StealthROM feature is enabled and you have the line STACKS=0,0 in your CONFIG.SYS file, Ventura Professional Version 2 will not operate properly. Removing the STACKS=0,0 statement should solve the problem. DR DOS 6 does not use hardware interrupt stacks; as a result, you cannot use DR DOS 6 with Ventura Professional 2 if you are using StealthROM. Ventura Professional Version 3 does not put its stacks in the EMS page frame and works properly with StealthROM.

Ventura Publisher 2 will not work properly if the EMS page frame is located at an address higher than E000. To find out where your page frame is located, type QEMM at the DOS prompt. If you are using a

page frame, you will see its address listed. If the address is higher than E000, type QEMM again and look at the list of areas and sizes. Find the first High RAM area below E000 that is at least 64K in size and jot down its starting address, then add the FRAME=xxxx parameter to the QEMM line, replacing xxxx with the address you wrote down (e.g., FRAME=D000).

VIDEO ACCELERATOR DRIVERS

Several video cards come with programs such as **SPEED_UP.SYS**, **RAMBIOS.SYS**, or **FASTBIOS.SYS.** These programs make a copy of the video ROM in RAM in order to speed up your video. If loaded after QEMM on a system with StealthROM enabled, they may refuse to load, complaining that someone else has taken Interrupt 10. If loaded before QEMM on the same system, StealthROM will be disabled because QEMM cannot find the ROM handler for Interrupt 10.

You can solve both of these problems with XSTI=10. No exclusion is necessary because the video ROM is no longer being used. Speed_up.sys can then be loaded after QEMM (and can be loaded into upper memory.) However, we strongly recommend that you NOT load SPEED_UP.SYS, RAMBIOS.SYS, FASTBIOS.SYS, or any similar driver. Using SPEED-UP.SYS costs you 36K of memory. Instead use QEMM's ROM parameter, producing the SAME effect but using NO address space between 0-1024K.

VP PLANNER

Some versions of VP Planner spreadsheet need the parameter UFP:N when Stealth is in effect if VP Planner is using EMS.

XTRADRIVE

IIT's XTRADRIVE disk compression utility ships with a disk cache that is not compatible with QEMM's StealthROM feature. For information on using XTRADRIVE with QEMM, read **XTRADRV.TEC**.

Return to Technotes Main Menu.

QEMM Analysis Procedure: Solving Memory Conflicts

Quarterdeck Technical Note #219

Q. What is Analysis?

A. Whenever QEMM is "on" it monitors the use of the first megabyte of address space. The QEMM/ANALYSIS screen of Manifest shows what portions of the address space need to be EXCLUDEd to avoid High RAM conflicts.

Q. Why should I use Analysis?

A. If you are getting an Exception #13, are unable to access your network when QEMM is installed, cannot access a floppy, print, run some program, lock-up at some identifiable point in operating your computer (from booting to running your word processor), or have some other problem when running QEMM that you do not have when you do not run QEMM, then the ANALYSIS procedure may be a useful diagnostic process.

Q. Why is this necessary?

A. This procedure is necessary sometimes because Adapter ROMs and Adapter RAMs do not identify themselves in such a way as to be detected properly by QEMM. Adapter ROMs are supposed to identify their length in the third byte of the ROM itself but sometimes report a smaller size. Adapter RAMs that are not active at boot look exactly like unoccupied address space. QEMM maps unused portions of the system BIOS ROM and will map over such adapter ROMs and RAMs. Some special CGA video cards have two pages of video: one at B800-BBFF, the second at BC00-BFFF. QEMM may map over the second page, causing a conflict if you run a program that tries to use the second page. In rare circumstances there are programs that use portions of the high address space directly.

Q. How does Analysis work?

A. The QEMM/ANALYSIS screen of Manifest is a cross-reference between the QEMM/TYPE and QEMM/ACCESSED screens. The TYPE screen shows what QEMM thinks the address space is used for: Video, ROM, Page frame, High RAM, etc. The ACCESSED screen shows whether the address space has been accessed. When QEMM is neither accessing the high address space (by creating High RAM -- when the RAM parameter is on the QEMM386.SYS line in CONFIG.SYS), and when some portion of the address space is being accessed by something that QEMM has not detected automatically, a portion of the address space must be EXCLUDEd. Analysis displays the correct range to EXCLUDE.

Q. How do I use ANALYSIS to find EXCLUDEs?

A. First remove the RAM parameter from the QEMM386.SYS line of the CONFIG.SYS and add the ON and MA=0 parameters. This is to make sure that QEMM386 is not itself a user of the high address space, and that to ensure that QEMM on. Then reboot the machine and run the software that was causing the problem. The problem should not recur; if it does, a High RAM conflict is not the problem, and you should consult Quarterdeck Technical Note #241, QEMM General Troubleshooting (TROUBLE.TEC). Without rebooting, look at the QEMM / ANALYSIS screen in Manifest. If you see Xs, then these portions of the address space must be EXCLUDEd on the QEMM386.SYS line of the CONFIG.SYS. See the QEMM manual section on the syntax of the EXCLUDE parameter. If the QEMM is putting the page frame over a portion of the address space that QEMM should not be mapping then it may be necessary to put the parameter FR=NONE on the QEMM386.SYS line of the CONFIG.SYS during the ANALYSIS process. Once you are done with the ANALYSIS process you can restore the RAM parameter to the QEMM386.SYS line along with the appropriate EXCLUDEs.

Q. How can Analysis fail?

A. The only serious pitfall to the ANALYSIS process is that there are users of the high address space that use the high address space only momentarily. There is, for example, a Bernoulli drive that has an Adapter ROM in the high address space. At boot time, the device driver for the Bernoulli Box searches for the ROM at the beginning of every 8K portion of the address space beginning at C800. If the Adapter ROM is at DC00 then the device driver will access every other 4k of the address space from C800 to DC00. This causes Xs to appear in every other block in this area, even though the areas between C800-DBFF are only being used during the searching process. Most of these areas need not be excluded. The range from DC00-DEFF, where the Adapter ROM of the Bernoulli Box resides, mayy require an EXCLUDE.

When you go into enhanced mode of Microsoft's Windows then QEMM is not active and the ANALYSIS process is not useful for the period of time that you are in enhanced mode.

Q. What cost can there be in excluding an area?

A. EXCLUDEing a portion of the address space will only cost you, at the worst, a bit of usable high RAM. It will not make your system malfunction in any other way, and is likely to improve stability.

If you add an EXCLUDE you should run OPTIMIZE again because your available High RAM regions have been resized and perhaps renumbered.

Q. What about those green "I"s?

A. The green I you see on the QEMM/ANALYSIS screen indicates that this portion of the address space has not been accessed by anyone YET and QEMM is not mapping this portion of the address space. It is quite possible that this portion of the address space will be accessed later. (The portion of the system BIOS ROM that contains the code for controlling the floppy drive may report that it is INCLUDable until you actually use the drive. If you INCLUDE it you will have no problem until you access a floppy.) The QEMM manual discusses the use of the ANALYSIS process for this purpose; this document does not.

Q. How do I perform the Analysis procedure?

A. A thorough Analysis requires that you run all your programs. However, if a specific program seems to be exhibiting a High RAM conflict, you need only do a mini-Analysis by running the program that seems to be exhibiting the problem. As an example, suppose that you have installed an adapter card and software for a scanner. While the device driver for the scanner loads in CONFIG.SYS, part of the hardware is not accessed until you scan a document. This causes a crash, since QEMM has mapped High RAM over the address space that the adapter and the software expect to use. (It is unlikely, incidentally, that OPTIMIZE will fail to detect the card properly.)

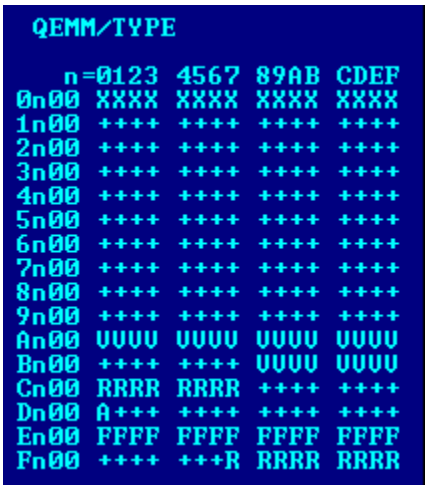
To begin the ANALYSIS process, modify your QEMM386.SYS line to
look like this:

DEVICE=C:\QEMM\QEMM386.SYS ON MA=0

Reboot your machine. In this example, we would access the hardware by scanning a document. After doing this, start Manifest by going to the DOS prompt and typing:

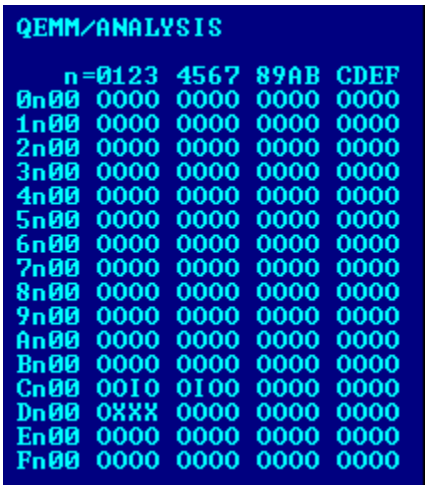
MFT

Select the QEMM / Type screen by typing Q, and then Y. Manifest will display a chart something like the following:

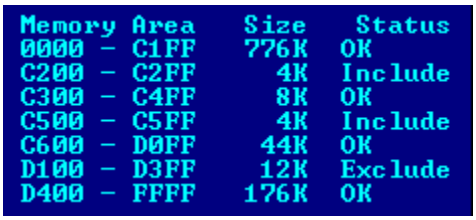


Look at the Dn00 line. This indicates that QEMM is identifying an Adapter ROM in D000-D0FF. Now select the QEMM / Accessed screen by pressing the C key. Manifest will display a screen that looks like this:

Look at the Dn00 line: QEMM is detecting that D000-D3FF is actually being accessed, D000-D1FF being written to and D200-D3FF only being read and is passing this information to Manifest. Now choose the QEMM / Analysis screen by pressing the N key:



Press the F3 key to display the information in List Mode. You will see a table that looks like this:



ANALYSIS is showing that D000-D3FF needs to be EXCLUDEd. This is the sort of address range associated with a scanner card at D000. In the example above, QEMM accurately identifies 4K adapter RAM on boot; in fact, the card has an unmarked RAM buffer as well. The ACCESSED map above shows that 16K is being accessed, and the ANALYSIS map is pointing out that the additional 12K must be EXCLUDEd. Note that unlike the display above, there should be no spaces within the EXCLUDE parameter. That is,

X=D100 - D3FF

is NOT a valid QEMM parameter, while

X=D100-D3FF

is valid.

Thus, a CONFIG.SYS line to account for the scanner, based on our mini-Analysis procedure, would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM X=D100-D3FF

(Increasing the range of the EXCLUDE such that it covers D000-D3FF is not harmful in any way.)

SUMMARY

QEMM does its best to identify users of the high address space, but hardware not accessed at boot time may not be detected by QEMM. ANALYSIS allows the user to find areas between 640K that are used by hardware.

Return to Technotes Main Menu.

QEMM Compatibility

The Compatibility page of QEMM Setup lets you review or change certain aspects of QEMM's behavior that may affect QEMM's compatibility with your particular system or configuration. When you select Compatibility by clicking on its tab you see a list of options. If you move the mouse pointer to an option, you see a brief description of what that option does in the **Parameter Information** area near the bottom of the window.

Each of the selections on this screen adds, deletes or modifies a parameter on the QEMM386.SYS device line in CONFIG.SYS. You can see QEMM's device line above the list of options. When you select an option, you will see how it modifies the device line. You can even edit the device line--just click at the point you want to edit.

IMPORTANT: Once you enable or disable any of the compatibility options, the change will not take effect until you reboot your PC.

Remove or Set Page Frame Address

Find ROM Holes

Exclude Stealthing a Particular ROM

Reclaim Top Memory

Enable Suspend/Resume Laptop

Relocate Extended BIOS Data Area

Setup QEMM for Troubleshooting

QEMM Installation: How it Modifies Your System

Quarterdeck Technical Note #297

When you install QEMM, the Install and Optimize procedures make changes to your CONFIG.SYS and AUTOEXEC.BAT files, and to any other batch file called by your AUTOEXEC.BAT file. The installation process also modifies your WIN.INI and SYSTEM.INI files to enable the Windows features in QEMM 8.0 and later. This technote is for those who want to know exactly what these changes are.

In this document, "Windows 3.1" refers to Microsoft Windows 3.10, 3.11, and Windows for WorkGroups; "Windows 95" refers to Microsoft Windows 95; "Windows" refers to both versions.

In some of the sample lines below, you will see the parameter /R:n which is used to load an item into a specified High RAM region. On your system, the n is replaced by a number indicating the region. You will also see the parameter /SIZE=n, where n is the number of bytes of memory the item needs to initialize.

Changes for All Systems

QEMM's INSTALL and OPTIMIZE programs make the following changesto your CONFIG.SYS file:

If you are using QEMM's DOS-Up feature, the following line is added at the beginning of CONFIG.SYS to prepare your system for parts of DOS to be loaded into upper memory:

```
DEVICE=C:\QEMM\DOSDATA.SYS
```

The following line is also added to CONFIG.SYS:

```
DEVICE=\QEMM\QEMM386.SYS RAM R:n
```

This is QEMM's device driver line; it is the line that loads QEMM whenever you boot your PC. Depending on your configuration, you may see additional parameters (e.g., ST:M or ST:F for StealthROM). For information on QEMM's parameters, see your QEMM manual.

If you are using DOS-Up, the following line, which loads the various parts of DOS into upper memory, appears directly after your QEMM386.SYS line:

```
DEVICE=C:\QEMM\DOS-UP.SYS
```

The INSTALL program also adds the following command (all on one line) to load QEMM's DPMI driver, which supports programs that use the DOS Protected Mode Interface:

```
DEVICE=C:\QEMM\LOADHI.SYS /R:n /SIZE=n C:\QEMM\QDPMI.SYS SWAPFILE=DPMI.SWP
SWAPSIZE=1024
```

(Note that the above should be a single line in your CONFIG.SYS file.)

The following syntax is added to the beginning of other device driver lines in CONFIG.SYS:

```
DEVICE=C:\QEMM\LOADHI.SYS /R:n
```

This command tells QEMM to load the device driver directly following this command into High RAM. (If Optimize has determined that a particular driver will not fit into upper memory, this syntax will not be added to that driver's line.) For example, if, before installing QEMM, your CONFIG.SYS file contained the

following line:

DEVICE=C:\DOS\ANSI.SYS

Optimize would change it to read as follows:

DEVICE=C:\QEMM\LOADHI.SYS /R:n SIZE=n C:\DOS\ANSI.SYS

If you are using DOS-Up, Optimize will add QEMM's LOADHI command to your SHELL line to load your command processor high (if you do not have a SHELL line, Optimize will add one for you). For example, if you have the following line in your CONFIG.SYS file:

SHELL=C:\DOS\COMMAND.COM C:\DOS\ /P

Optimize will change it to read as follows, all on one line:

SHELL=C:\QEMM\LOADHI.COM /R:n C:\DOS\COMMAND.COM /P

If you are using DOS version 5 or 6, QEMM's installation will add the following line to your CONFIG.SYS file if it is not already there:

DOS=HIGH

This is a DOS command that loads part of DOS's kernel and DOS buffers into the HMA, the first 64K of extended memory. If QEMM's installation detects DESQview or DESQview/X on your system and there is no DOS=HIGH statement in CONFIG.SYS, this statement will not be added. By omitting DOS=HIGH, DESQview or DESQview/X can use the HMA to load part of its own code.

If you are using MS-DOS 6's DoubleSpace disk compressor, QEMM'sinstallation will add the following line to enable the StealthD*Space feature:

DEVICE=C:\QEMM\LOADHI.SYS r:n SIZE=n C:\QEMM\ST-DSPC.SYS

As mentioned earlier in this technote, QEMM's installation will remove device driver lines for other memory managers from CONFIG.SYS.

Optimize makes the following changes to AUTOEXEC.BAT and to any batch file called by AUTOEXEC.BAT:

QEMM's directory is added to your PATH statement.

The following syntax is added to the beginning of lines that load TSRs (i.e., memory-resident programs) high:

C:\QEMM\LOADHI /R:n

This command tells QEMM to load the TSR directly following this command into High RAM. For example, if, before installing QEMM, your AUTOEXEC.BAT file contained the line:

C:\NET\NETX.COM

Optimize would change it to read as follows:

C:\QEMM\LOADHI /R:n SIZE=n C:\NET\NETX.COM

Changes for Microsoft Windows

If you are using Microsoft Windows, QEMM adds the line

SystemROMBreakPoint=False

to the [386Enh] section of Windows' SYSTEM.INI file to make Windows run optimally with QEMM.

If you are using FreeMeg in Windows 3.1, FREEMEG.DLL will be added to the drivers= line in the [boot] section of the Windows SYSTEM.INI file.

If you are using Resource Manager in Windows 3.1, RSRCMGR.DLL will be added to the drivers= line in the [boot] section of the Windows SYSTEM.INI file.

If you are using MagnaRAM in Windows 3.1, the program LOGO31.EXE will be added to the Run= line in the WIN.INI file. The line

DEVICE=<QEMM path>\MAGNA31.VXD

will be added to the [386Enh] section of the SYSTEM.INI file, and a [Quarterdeck_MagnaRAM] section will be created. Within this new section, the lines

[Quarterdeck_MagnaRAM]
COMPRESSION=ON
COMPRESSION_BUFFER_SIZE=
COMPRESSION_THRESHOLD=0

will be created with values reflecting MagnaRAM's defaults or the settings that you chose via the QEMM Setup program.

If you are using MagnaRAM in Windows 95, configuration information will be stored in the Windows Registry. In the section

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\...Quarterdeck_MagnaRAM

the following values will be added:

COMPRESSION=ON
COMPRESSION_BUFFER_SIZE=""
COMPRESSION_THRESHOLD=""

Again, the actual values associated with each key will vary depending on your choice of configuration.

Return to Technotes Main Menu.

QEMM: Running Optimize with a Windows 95 Multiple Configuration Startup Menu

Quarterdeck Technical Note #312

If you installed Windows 95 over an older version of DOS, you may have a multiple configuration STARTUP MENU that displays when you boot your system. The Startup Menu lets you pick the configuration you want to use during bootup. You might, for example, have installed Windows 95 into a new directory rather than installing over Windows 3.1. At times, you may want to boot into Windows 3.1 instead of Windows 95.

If you use a WINDOWS 95 Startup menu, you must make sure that QEMM's OPTIMIZE boots into the same configuration choice each time it reboots your system. In this technote we tell you how to create a multiple configuration WINDOWS 95 Startup Menu and how to run Optimize when using such a menu.

1. Boot your computer and watch for the message "Starting Windows 95". When you see this message, press the **F8** key QUICKLY. If one of the choices is "Previous Version of DOS", note the number of that choice and write it down. Choose NORMAL and boot to WINDOWS 95.

2. When WINDOWS 95 has booted up, choose "Shutdown" from the Start Menu, and then choose "Restart to DOS".

3. Make sure you are in the WINDOWS 95 version of DOS by typing:

VER <enter>

If you see "Windows 95" on your screen, continue with Step 4. If you do not see the "Windows 95" message, boot to WINDOWS 95 or see your WINDOWS 95 documentation. DO NOT CONTINUE with this technote.

4. In order to enable the Windows 95 Boot Menu feature, you must make modifications to Windows 95's MSDOS.SYS configuration file. Type the following:

cd\ <enter>
attrib -s -h -r msdos.sys <enter>
copy msdos.sys msdos.qdk
edit msdos.sys <enter>

5. Add the following to the [Options] section:

BootMulti=1 ;Win95 Multi-Boot Configuration enabled
BootMenu=1 ;Win95 Multi-Boot Menu displayed by default
BootMenuDelay=5 ;Seconds to delay menu selection
BootMenuDefault=x ;Boots to item number x after the boot menu delay

Note:
If you did not make a note of the menu number for "Previous Version of DOS", do not add the BootMenuDefault= line. Instead, press the keys <ALT-F> <X> <Y> to save the file, then reboot the computer. When the Startup Menu displays, make note of the menu number for "Previous Version of DOS", then boot to the "Command Prompt Only" selection. Edit MSDOS.SYS again, and add the BootMenuDefault= line. Press the keys <ALT-F> <X> <Y> to save the file.

6. Type:

attrib +s +h +r msdos.sys <enter>

7. Re-boot your system. (If you end up in Windows 95, select "Shut Down" from the Start menu, then select "Restart in MS DOS mode".) Then type:

C:\QEMM\OPTIMIZE <enter>

to OPTIMIZE your previous version of DOS.

8. If you want to boot to WINDOWS 95 by default, follow these steps. (If you do not, you are finished with this technote.)

Re-boot and choose NORMAL (choice 1) on the Startup Menu.

When WINDOWS 95 appears, choose "Shut Down" from the Startup Menu, then choose "Restart to DOS".

Type the following:

CD\ <enter>
ATTRIB -S -H -R MSDOS.SYS <enter>
EDIT MSDOS.SYS <enter>

Place a semi-colon (;) before BootMenuDefault in the MSDOS.SYS file.

Press the keys <ALT-F> <X> <Y> one after the other to return to a DOS prompt, then type:

ATTRIB +S +H +R MSDOS.SYS <enter>

Now, when you boot your system the Startup Menu will be displayed for 5 seconds and then WINDOWS 95 will automatically start.

Return to Technotes Main Menu.

QEMM Programming Interface (QPI)

The QEMM Programming Interface (QPI) lets programs request information or services from QEMM. Programs can use the QPI to do the following:

- Determine QEMM's status, and change that status if the system configuration allows.
- Determine QEMM's version number.
- Determine whether QEMM's StealthROM feature is active, and if so what StealthROM mode is in use.
- Determine the number of ROMs that QEMM is Stealthing and the beginning segment address and length of each ROM.
- Determine whether QEMM is supporting the system's Suspend/Resume features, and if so what interrupt these features are using.
- Determine whether QEMM is allowing or suppressing the BIOS calls that make it possible to do work while waiting for disk activity to complete, and tell QEMM to allow or suppress these calls if the system configuration allows.
- Copy all or part of the contents of a Stealthed ROM into a buffer.
- Determine the physical memory mapped to any linear memory address,and change the page table so that any page of physical memory is mapped to any linear memory address.
- Read or write I/O ports, even if QEMM is trapping access to those ports.
- Determine whether QEMM is trapping access to I/O ports, and tell QEMM to trap access to ports on the calling program's behalf.
- Install a software routine that performs whatever actions the program requires when a given I/O port is accessed.
- Simulate a hardware interrupt in such a way that it goes to the correct DESQview or DESQview/X window.

You can find sample QPI code and programs on the Quarterdeck bulletin board and other electronic support locations.

Getting the QPI Entry Point

The first step in using the QPI is getting the double-word address of the QPI entry point.

The method of obtaining the entry point address that is described here is available only in versions of QEMM higher than 6.00. Programs that want to run with QEMM 5 must use a less straightforward INT 2F interface to get the QPI entry point; for more information, contact Quarterdeck via Compuserve (GO QUARTERDECK), the Internet (qsupport@qdeck.com) or our BBS and ask for the QDMEM interface document.

QEMM defines a DOS device driver called QEMM386$. To obtain the entry point for QPI, do an IOCTL Read Control String call (INT 21, function 4402h) to read four bytes from this device driver. Here is a code sequence that demonstrates the details:

```
QEMMDeviceName   db 'QEMM386$',0
QPIEntryPoint    dd ?

  GetQPIEntryPoint proc
      mov    dx,offset QEMMDeviceName
      mov    ax,3d00h
      int    21h                        ; Try to open QEMM386$
      jc     NoQEMM                        ; If CY, QEMM not present
      mov    bx,ax                       ; Save file handle in BX
```

```
        mov     dx,offset QPIEntryPoint   ; Store the entry point here
        mov     cx,4                      ; Set up to read 4 bytes
        mov     ax,4402h                    ; IOCTL Read Control String
        int     21h
        pushf                             ; Save the error code
        mov     ah,3eh                      ; Close the handle
        int     21h
        popf                              ; Restore the error code
        jc      NoQEMM                        ; If CY, QEMM is pre-6.00
        ret
NoQEMM: stc
        ret
GetQPIEntryPoint endp
```

QPI Functions

Once you have stored the address of the QPI entry point, you make all calls to QPI by loading AH or AX with the function number of the call, setting the other registers to values appropriate to the function, and making a far call to the entry point. The carry flag is set on return if there is an error, or if the function number is not valid in that version of QEMM.

The QPI calls of interest to third-party programmers are listed below. The version in which each of the calls was implemented is noted.

The QPI_GetStatus Call

The QPI_GetStatus call tells you whether QEMM is on or off, and whether it is in auto mode (see the AUTO/ON/OFF parameter in Chapter 7 in the QEMM Reference Manual for more information). All versions of QEMM support this call.

```
QPI_GetStatus           EQU       0
        ; Takes          AH = 0
        ; Returns        AL = 0 if on
        ;                AL = 1 if auto/on
        ;                AL = 2 if off
        ;                AL = 3 if auto/off
```

The QPI_SetStatus Call

The QPI_SetStatus call lets you set the status of QEMM. If QEMM is forced on by a parameter (like the RAM parameter) or other services that it provides, this call will have no effect. You should therefore make the QPI_GetStatus call after the QPI_SetStatus call to see if the first call was successful. All versions of QEMM support this call.

```
QPI_SetStatus           EQU       1
          ; Takes       AH = 1
          ;             AL = 0 if on
          ;             AL = 1 if auto/on
          ;             AL = 2 if off
          ;             AL = 3 if auto/off
```

The QPI_GetVersion Call

The QPI_GetVersion call returns the QEMM version number in Binary Coded Decimal form in AX and BX. For instance, the call will return BX = 0750 (not BX = 0732) for QEMM version 7.5. All versions of QEMM support this call.

```
QPI_GetVersion          EQU       3
        ; Takes         AH = 3
        ; Returns       BH = major version (in Binary Coded Decimal)
        ;                BL = minor version (in Binary Coded Decimal)
        ;                AX = same as BX
```

The QPI_GetInfo Call

In QEMM version 6.00 and later, the QPI_GetInfo call returns an ASCII letter in CL that tells QEMM's StealthROM mode (if any), and a number in CH that tells which interrupt (not IRQ) QEMM is monitoring (if any) to support Suspend/Resume features. In QEMM version 7.00 and later, the call also returns the size of QEMM's disk buffer in DL and a bit map of information about the disk buffer in BH. Bit 1 of BH will be on if the disk buffer has already been used; bit 0 will be on if QEMM is buffering only INT 13s into the page frame (DISKBUFFRAME) and off if all INT 13s into nonlinear memory are being buffered (DISKBUF). Note that other registers are not preserved by this call.

```
QPI_GetInfo             EQU       1E00h
        ; Takes         AX = 1E00
        ; Returns       BH = xxxxxxAB
        ;                    where A = 1 if disk buffer has been used
        ;                              yet, 0 if not
        ;                          B = 1 if DISKBUFFRAME buffer, 0 if
        ;                              DISKBUF buffer - not valid if
        ;                              DL = 0
        ;                BL = reserved
        ;                CL = StealthROM type (0 for no StealthROM,
        ;                                      "M" or "F" otherwise,
        ;                                      other StealthROM
        ;                                      types possible
        ;                                      in future)
        ;
        ;                CH = Suspend/Resume INT number (0 = none)
        ;                DL = size of QEMM disk buffer in K
        ;                     (if 0, disk buffer doesn't exist)
        ;                DH, DI, SI = reserved
        ;
```

The QPI_GetStealthCount Call

The QPI_GetStealthCount tells how many ROMs QEMM is Stealthing.
QEMM versions 6.00 and later support this call.

```
QPI_GetStealthCount equ   1E01h
        ; Takes       AX = 1E01
        ; Returns     BX = number of ROMs that are Stealthed
```

The QPI_GetStealthList Call

The QPI_GetStealthList call gives the same information as the QPI_GetStealthCount call, and also fills a buffer with information on the location and size of each Stealthed ROM. QEMM versions 6.00 and later support this call.

```
QPI_GetStealthList   equ 1E02h
        ; Takes      AX = 1E02
        ;            ES:DI= buffer to hold the list of Stealthed
        ;                   ROMs
        ; Returns    BX = number of ROMs that are Stealthed
        ;            Table at ES:DI will be filled in with:
        ;                 dw ROM start segment
        ;                 dw Length of ROM in paragraphs
        ;            for each ROM that is Stealthed
```

The QPI_GetPTE Call

The QPI_GetPTE call returns the page table entry for any logical page in the first 1088K of memory. In other words, if you pass this call the address of any page in the first 1088K of memory, the call will return (in an extended register) the doubleword page table entry for that address, which includes, among other things, information about which physical page of memory QEMM has mapped to the logical address that you provided. QEMM versions 6.00 and later support this call.

CX should contain the number of the logical page in memory that you want to affect; the highest valid CX is 010F. (Page numbers refer to consecutive 4K sections of memory, aligned on 4K boundaries: that is, 0000 refers to paragraph 0000-00FF, 0001 to 0100-01FF, etc.) EDX (the extended DX register) should contain a page table entry, in the following format:

Bit 0 is the Present bit. Any access to a page with this bit off causes a page fault.

Bit 1 is the Read/Write bit. Any writing to a page with this bit off causes a page fault.

Bit 2 is the User/Supervisor bit. Any access to this page when the processor is at privilege level 3 causes a page fault.

Bits 3 and 4 must be 0.

Bit 5 is the Accessed bit. Any read or write of the page causes the processor to turn on this bit.

Bit 6 is the Dirty bit. Any write to the page causes the processor to turn on this bit.

Bits 7 and 8 must be 0.

Bits 9, 10, and 11 are available for systems programmer use.

Bits 12 through 31 are the page number.

For instance, a page table entry 000FF007 means physical page number 000FF (paragraph FF00-FFFF), which is not yet accessed nor dirty, but which is present, writable and user-accessible.

```
QPI_GetPTE              equ 1F00h
```

```
        ; Takes       AX = 1F00
        ;             CX = page number
        ; Returns     EDX = page table entry for that page number
```

The QPI_SetPTE Call

The QPI_SetPTE call lets you set the page table entry for any logical page in the first 1088K of memory. In particular, this means that you can tell QEMM to map any 4K of memory to any 4K-aligned address below the 1088K mark. QEMM versions 6.00 and later support this call. See the section above on the QPI_GetPTE call for more information.

```
QPI_SetPTE          equ     1F01h
        ; Takes     AX = 1F01
        ;           CX = page number
        ;           EDX = page table entry to set at that page
        ;                 number
```

The QPI_GetVHIInfo Call

The QPI_GetVHIInfo call, in conjunction with the QPI_SetVHIInfo call, is primarily used by disk cache developers who wish to get information on QEMM's safety precaution of suppressing the BIOS INT 15 function 90 callout. This safety precaution, known as VirtualHDIRQ or VHI (see the section on the VIRTUALHDIRQ:N parameter in Chapter 7 in the QEMM Reference Manual for more information), is normally in effect only when the disk interrupt INT 13 is being Stealthed. If you have verified that the use of INT 15 fn 90 in the disk cache you are developing is compatible with QEMM's StealthROM feature, you will want to tell QEMM to allow INT 15 function 90.

The call returns flags in BL that give the VHI state. Bit 7 will be on whenever QEMM is Stealthing INT 13 (if INT 13 is not Stealthed, QEMM never suppresses INT 15 function 90); bit 0 will be on if QEMM is currently suppressing INT 15 function 90. Bits 1-6 of the VHI bit map are reserved. QEMM versions 6.00 and later support this call.

```
QPI_GetVHIInfo      equ     2000h
        ; Takes     AX = 2000
        ; Returns   BL = AxxxxxxB
        ;           where A = 1 if VHI is being paid attention to
        ;                 B = 1 if VHI is currently enabled
        ;           (i.e. INT 15 fn 90 is currently suppressed)
        ;                 x = reserved
```

The QPI_SetVHIInfo Call

The QPI_SetVHIInfo call lets you turn on or off QEMM's safety precaution of suppressing INT 15 function 90 whenever the disk interrupt INT 13 is being Stealthed. (See the section above on the QPI_GetVHIInfo call for more information.) To request a VHI state, set BL to 1 to suppress INT 15 function 90, or set BL to 0 to allow INT 15 fn 90. QEMM will return the previous VHI flags in BL. If bit 7 of the returned flags is off, then QEMM is not paying attention to the VHI state, and your request did not have an effect. QEMM versions 6.00 and later support this call.

```
QPI_SetVHIInfo      equ     2001h
```

```
        ; Takes       AX = 2001
        ;             BL = xxxxxxxB that you want (bit 7 is ignored)
        ; Returns     BL = AxxxxxxB of previous VHI state;
        ;             if A of output = 0, B of input was ignored
```

The QPI_CopyStealthRoms Call

The QPI_CopyStealthRoms call tells QEMM to copy the contents of part or all of a Stealthed ROM into a buffer in conventional memory. This is the only reliable to access the contents of a Stealthed ROM. QEMM versions 6.00 and later support this call.

```
QPI_CopyStealthRoms   equ       2100h
        ; Takes       AX = 2100
        ;             DS:SI = Original address of ROM to copy
        ;             ES:DI = Destination address
        ;                     in conventional memory
        ;             ECX = # of bytes to copy
        ; Returns     CY if no stealth or if DS:SI
        ;                     not within C000-FFFF
```

I/O Trapping

The following calls make up the Quarterdeck QEMM I/O Trapping Programming Interface. This interface allows a real-mode program to specify I/O ports that QEMM should trap access to, as well as I/O callback routines that QEMM will call whenever one of these I/O ports is accessed. Using this interface, you can emulate hardware devices that are accessible via I/O ports.

When QEMM traps an I/O port, all accesses of that port, whether input or output, are intercepted by QEMM. (QEMM traps certain I/O ports itself, for proper management of virtual-8086 mode.) Whenever an I/O port that a program has asked QEMM to trap is accessed, QEMM calls a real-mode I/O callback routine. The same callback routine is called for all trapped I/O ports.

A program that wishes to trap an I/O port should:

1) Use the QPI_GetVersion call to make sure that the version of QEMM is 7.03 or later. Earlier versions do not support most of the I/O Trapping Programming Interface. Alternatively, if the version of QEMM does not support the call you have made, the call will return with the carry flag set;

2) Issue a QPI_GetPortTrap call to determine that another program is not already trapping the port. If another program is trapping the port, it is generally advisable not to install your port trap;

3) Get the address of the existing callback routine with the QPI_GetIOCallback call. Because there is only one I/O callback routine, and because multiple programs may request I/O trapping, your callback routine must jump to the previous callback routine whenever your routine is not interested in the I/O port being accessed;

4) Install its own far routine as the new callback routine, using the QPI_SetIOCallback call. Your callback routine will be passed the following information:

```
AX = Data for output
CX = Type of I/O (see flag bits defined below)
DX = Port number
```

IF = 0 (interrupts are disabled)

When the callback routine has finished its work, it should return far with all registers other than CX and DX preserved. If the routine is called to get input from a port, AX should be modified. The bit-mapped word in CX contains the following information:

```
IOT_Output          equ     0000000000000100b
        ;           bit 2 is 1 if output,
        ;           0 if input

IOT_Word            equ     0000000000001000b
        ;           bit 3 is 1 if word I/O,
        ;           0 if byte I/O

IOT_IF              equ     0000001000000000b
        ;           bit 9 is the same as the
        ;           caller's interrupt flag
```

5) Specify which port to trap with the QPI_SetPortTrap call.

If your program does not stay resident forever, it should do the following before exiting:

6) Use the QPI_GetIOCallback call to make sure that no one has installed a callback routine after yours. If a handler is installed after yours, you should remain resident;

7) Remove its trap with the QPI_ClearPortTrap call. This call will clear all traps on a particular I/O port, unless QEMM is trapping that port for itself, in which case QEMM's trapping alone will remain in effect for that port;

8) Remove its callback by using the QPI_SetIOCallback call to set the previously existing callback routine.

The following restrictions apply to this interface:

Only INs and OUTs of words or bytes are supported, not INs and OUTs of doublewords. Also, string I/O (the INS and OUTS instructions) are supported only as of QEMM version 7.5 or later.

QEMM cannot trap the I/O of VCPI protected-mode programs. Furthermore, because Quarterdeck's DPMI driver (QDPMI) is implemented as a VCPI client, QEMM cannot trap the I/O of DPMI clients either. Furthermore, these traps are no longer in effect when Microsoft Windows 386 enhanced mode is running.

The following four calls allow you to bypass port trapping and read and write I/O ports directly. QEMM 5.00 and later versions support these four calls.

```
QPI_UntrappedIORead   equ     1A00h
        ; Takes       AX = 1A00
        ;             DX = port to read
        ; Returns     BL = value read
```

```
QPI_UntrappedIOWrite   equ       1A01h
        ; Takes       AX = 1A01
        ;             DX = port to write
        ;             BL = value to write


QPI_UntrappedIOReadIndexed    equ       1A02h
        ; Takes       AX = 1A02
        ;             DX = base port to read
        ;             BH = index into base port
        ; Returns     BL = value read


QPI_UntrappedIOWriteIndexed    equ       1A03h
        ; Takes       AX = 1A03
        ;             DX = base port to write
        ;             BH = index into base port
        ;             BL = value to write
```

The QPI_UntrappedIO Call

The QPI_UntrappedIO call performs the same functions as the QPI_UntrappedIORead and the QPI_UntrappedIOWrite calls, but it uses register values that are similar to the ones QEMM passes to your I/O callback routine (including the flags in CX that give information about the type and size of the I/O and the caller's interrupt flag). This call may therefore be easier to use from within an I/O callback routine. QEMM 7.03 and later support this call.

```
QPI_UntrappedIO       equ       1A04h
        ; Takes       AX = 1A04
        ;             BX = value to write
        ;             DX = port to read or write
        ;             CX = type of I/O (see description above
        ;                  of CX passed to callback routine)
        ; Returns     BX = value read
```

The following five calls are described in the introduction above. QEMM 7.03 and later versions support these calls.

```
QPI_GetIOCallback    equ       1A06h
        ; Takes       AX = 1A06
        ; Returns     ES:DI = previous I/O callback function


QPI_SetIOCallback    equ       1A07h
        ; Takes       AX = 1A07
        ;             ES:DI = new I/O callback function


QPI_GetPortTrap       equ       1A08h
        ; Takes       AX = 1A08
        ;             DX = I/O port number
        ; Returns     BL = 0 if port not trapped,
```

```
                        BL = 1 if port already trapped


  QPI_SetPortTrap       equ       1A09h
          ; Takes       AX = 1A09
          ;             DX = I/O port number


  QPI_ClearPortTrap     equ       1A0Ah
          ; Takes       AX = 1A0A
          ;             DX = I/O port number
```

The QPI_SimulateHWInt Call

The QPI_SimulateHWInt call can be used by callback routines that wish to simulate a hardware interrupt. When DESQview or DESQview/X is running, the interrupt handler that should receive the interrupt may be in a different process from the current one. Use QPI_SimulateHWInt to simulate an interrupt properly when DESQview or DESQview/X is running. The DESQview API Reference Manual describes how to determine when DESQview is running. QEMM 7.03 and later versions support this call.

```
  QPI_SimulateHWInt     equ       1C04h
          ; Takes       AX = 1C04
          ;             BX = interrupt number to generate


 QEMM_Get_QEMM_SubVer equ 1E05h

      ; Takes         AX = 30d * 100h + 5 (1E05h)dle a GameRunner issue.
      ;               CX = Length of buffer the QEMM Game Edition uniquely to
      ;            ES:DI = buffer QEMMs, so:
      ; Returns      CX = number of bytes which could not fit in buffer
      ;               CY if function not supported
      ; Fills in the given buffer with a null terminated string.   If
      ; the actual length of the string is greater than the length ofsion.   The
      ; the buffer (as passed in CX) then it clips the string to theit returns
      ; buffer length - 1 and puts a null in the last byte of the buffer
      ; and returns the number of extra bytes in CX.   If everything
      ; fits in the buffer, then the returned CX=0.
```

Return to Technotes Main Menu.

The technote PRODUCTS.TEC, in the TECHNOTE subdirectory of the directory into which you installed QEMM, contains a list of compatibility issues between QEMM and other hardware and software products. Please read this technote before beginning any troubleshooting procedure.

Other online documents, including QEMMUTIL.TEC, TESTPRGS.TEC, and QPI.TEC, descrbe utility programs and technical information for programmersand advanced users of QEMM. Still others provide background information, compatibility notes, or tips and tricks related to various types of hardware and software.

All of the QEMM technotes may be viewed by running QSETUP, either from DOS or from Windows; QSETUP incorporates a file viewer that allows you to read these notes easily. Technotes are also included in this online help file.

QEMM Utility Programs

Quarterdeck Technical Note #294

This technical note describes several miscellaneous utility programs included with QEMM. These programs let you:

Load device drivers before QEMM.
Load device drivers from the DOS prompt.
Ensure that Microsoft Windows runs properly with QEMM if you install Windows after QEMM.
Fix certain problems that occur on some Toshiba laptop PCs.
Fix problems that may occur if you are running LAN WorkPlace for DOS.
Fix bootup problems that occur with some Ultra Stor disk controllers.

Read this document if any of these topics concerns you.

DEVICE.COM: Loading Device Drivers from the DOS Prompt

DEVICE.COM is a program you can use to load certain device drivers from the DOS prompt instead of from CONFIG.SYS. DEVICE.COM will load character device drivers (e.g., a mouse driver, ANSI.SYS), but not block device drivers (e.g., drivers for disk compressors, RAM disks or CD ROM drives). In general, a block device is one that will be assigned a drive letter (e.g., E:, H:). You may want to use DEVICE.COM for the following reasons:

To load a device driver in a DESQview or DESQview/X window, or in a Microsoft Windows DOS window. For example, if you have a program that requires ANSI.SYS, you can load ANSI.SYS in that program's window without imposing ANSI's overhead on all your other windows.

To load a device driver from the DOS prompt when you need it.

To load a device driver in AUTOEXEC.BAT to help QEMM's Optimize program do a more efficient job of loading programs into upper memory. Occasionally a driver in CONFIG.SYS uses enough upper memory that there is not enough left to load a subsequent driver or TSR. In this case, you can try using DEVICE.COM to load the device driver in AUTOEXEC.BAT after the later driver or TSR has been loaded. This method is especially worth trying if Optimize is unable to load a very large TSR or driver into upper memory, after loading a preceding driver into upper memory.

The syntax for DEVICE.COM is:

DEVICE device_driver_pathname

For example, to load ANSI.SYS from the DOS prompt you would type: DEVICE C:\DOS\ANSI.SYS.

FIXINT13.SYS

FIXINT13.SYS prevents certain problems that can happen when the CONFIG.SYS file is being executed and problems occur on the DOS stack. FIXINT13's job is to switch away from the DOS stack and on to its own stack in conventional memory when a BIOS disk call occurs while the CONFIG.SYS file is being processed. If you give FIXINT13 the /STACKSIZE=xxxx parameter, you can also change the size of FIXINT13's stack, to prevent stack overruns. The default size of the FIXINT13 stack is 256 bytes; xxxx can be any value between 128 and 1024.

FIXINT13 is needed with some UltraStor disk controllers to prevent "Device not found" errors during the boot process. FIXINT13 with the /STACKSIZE=384 parameter also prevents "Configuration too large for memory" errors or crashes in the CONFIG.SYS file on some systems with Adaptec 1542c controllers.

If you think you need FIXINT13.SYS, load it in the CONFIG.SYS file, immediately before the QEMM386.SYS line (and after DOSDATA.SYS and any other programs loaded before QEMM386.SYS). For example:

DEVICE=C:\QEMM\FIXINT13.SYS

or

DEVICE=C:\QEMM\FIXINT13.SYS /STACKSIZE=384

HOOKROM.SYS: Loading Device Drivers before QEMM

HOOKROM.SYS is a device driver that allows you to load other device drivers before QEMM in your CONFIG.SYS file. You may need HOOKROM.SYS if you need to load a device driver before QEMM386.SYS and you are using QEMM's StealthROM feature (i.e., you have the parameter ST:M or ST:F on the QEMM386.SYS device line in CONFIG.SYS). Though it is usually best to load device drivers after QEMM386.SYS, there are some special drivers (like the ones that manage some 80386 conversion hardware) that must load before QEMM386.SYS. These drivers may obscure information that QEMM needs to enable the StealthROM feature. If this is the case, QEMM386.SYS will post an error message that reads, QEMM386: Disabling StealthROM because QEMM could not locate the ROM handler for INT x, where x is the number of an interrupt handler that QEMM needs to manage for the StealthROM process to work.

The solution to this problem is to place the line DEVICE=C:\QEMM\HOOKROM.SYS at the beginning of the CONFIG.SYS file, before the driver that needs to be loaded before QEMM386.SYS. HOOKROM will gather the necessary information for QEMM386.SYS, so that the special driver does not interfere with the StealthROM process.

LWPFIX: Fixing Problems with LAN WorkPlace

LWPFIX.COM is a TSR that works around problems with some versions of Novell's LAN WorkPlace for DOS. Specifically, some versions of Novell's TCPIP.EXE do not properly save and restore the state of two of the processor's 32-bit extended registers; this can cause malfunctions and crashes when other programs are using these registers. By adding the command C:\QEMM\LWPFIX.COM to your AUTOEXEC.BAT after TCPIP.EXE is loaded, you ensure that the original contents of these registers will be restored after TCPIP.EXE finishes using them.

You will need LWPFIX.COM if you are using versions 4.00 or 4.01 of LAN WorkPlace for DOS; you may need it with some later versions. LWPFIX.COM does no harm even if it is not needed, so it may be worth loading LWPFIX.COM if you are experiencing problems with any version of LAN WorkPlace for DOS. DESQview/X automatically loads a driver that performs the same function as LWPFIX.COM, so LWPFIX.COM is only needed to fix problems that occur outside of DESQview/X.

QWINFIX: Using Microsoft Windows with QEMM

QWINFIX.COM makes Microsoft Windows 386 enhanced mode compatible with QEMM. QWINFIX does this by adding the line SystemROMBreakPoint=false to the [386Enh] section of Windows' SYSTEM.INI file. If you have Windows installed on your PC at the time you install QEMM, QEMM's installation program

will run QWINFIX. If you install Windows after installing QEMM, you should run QWINFIX. To run QWINFIX:

Switch to Windows' directory (usually \WINDOWS).

Type QWINFIX and press Enter.

QEMMREG: Displaying QEMM's Version and Serial Number

QEMMREG.COM displays QEMM's version number and your serial number. To use QEMMREG:

Type QEMMREG and press Enter.

SCANMEM.COM: Checking for Memory Above the 16MB Line

SCANMEM.COM is a program that scans your PC's memory, looking for memory that is not reported by the BIOS, and reports a parameter you can use to make QEMM see this memory. This program may be useful if your system has more than 16 megabytes of memory and you cannot access the memory above 16 megabytes after installing QEMM. The USERAM:XX:YY parameter to QEMM performs a similar function, scanning all of the address range between XX and YY; running SCANMEM is not a prerequisite to using USERAM. However, you can use SCANMEM.COM to find the precise ranges of addresses that can be specified to USERAM; this may save a couple of moments when you boot your machine.

Some systems with more than 16 megabytes of memory do not report all of their memory through the appropriate BIOS call (the standard method for reporting how much memory is installed in a system). On such a system, QEMM will not automatically detect the memory above 16 megabytes. Certain Compaq and Dell PCs and PCs with older Micronics motherboards (e.g., some Gateways) with more than 16 megabytes of memory are the most notable examples. SCANMEM tries to locate regions of RAM that QEMM does not detect automatically when it loads. If you have a system with more than 16 megabytes of RAM and you suspect that all your memory is not available, follow the steps below:

First, run Manifest to see if the memory is recognized. Type

MFT

and press Enter.

Near the bottom of the Manifest System Overview screen, you will see a number for Total Extended Memory (pooled). If you have over 16 megabytes of RAM and the amount displayed is less than 16384K, your system's BIOS is not reporting the memory above 16 megabytes, and you should continue with the steps below.

Note: If you are having problems accessing memory above 16 megabytes on a Dell PC, contact Dell's technical support. They may be able to supply you with an updated version of the system BIOS that fixes this problem.

You should not run the SCANMEM program when QEMM386.SYS, DOS's HIMEM or EMM386, or any other memory manager is loaded. Similarly, you should not use any program that uses extended memory without the assistance of a memory manager; some disk caches or RAM disks may do this. The ideal environment for running SCANMEM is a completely clean boot with no CONFIG.SYS or AUTOEXEC.BAT.

Reboot your PC without any extended memory managers or consumers present. After rebooting, type

SCANMEM

and press Enter.

SCANMEM will scan your PC's memory, and if it finds a memory region that QEMM has not detected, it will post a message listing
the exact form of the USERAM=xxxxxxxx-yyyyyyyy parameter that you should put on the QEMM386.SYS device line in CONFIG.SYS. SCANMEM will list an address range in eight-digit hexadecimal format (e.g., USERAM=00100000-00206000). When you add the USERAM parameter to the QEMM386.SYS device line, use all the digits given in the address. This parameter will reclaim the memory; SCANMEM's only job is to suggest the appropriate USERAM parameter (for information on the USERAM parameter, see Chapter 7).

If SCANMEM lists a USERAM parameter, jot it down, when edit your CONFIG.SYS file and add the exact parameter SCANMEM reported to the QEMM386.SYS device driver line. Save your CONFIG.SYS file and reboot.

After rebooting, you should be able to access the memory above 16 megabytes. You can use Quarterdeck Manifest to verify that the memory is recognized (see the first step above).

SCANMEM may post various messages:

Address wrap at xxxxx, where xxxxx is a memory address, means that SCANMEM has detected that your PC's address space is smaller than the four gigabytes that the 386 processor can address. This message is for your information and does not invalidate SCANMEM's findings.

NOUSERAM=xxxxx-yyyyy, where xxxxx and yyyyy are memory addresses, means that SCANMEM does not detect memory in the address range xxxxx-yyyyy, even though your system's BIOS has reported enough extended memory to fill these addresses. If you see this message, you may wish to use your PC's system setup to reconfigure your machine so that the BIOS reports extended memory properly.

Error: Invalid USERAM due to memory cache! means that SCANMEM has detected that the USERAM=xxxxx-yyyyy parameter that it last printed to the screen is invalid and should not be used. You should ignore only the last USERAM message printed to the screen; previous USERAM messages are valid. This error may occur if an unusual memory cache makes the contents of memory appear to be variable.

T386.EXE: Displaying the Pop-up Menu on Toshiba Laptops

T386.EXE is a program for Toshiba laptop computers which allows the Toshiba pop-up menu to appear when QEMM is enabled. T386 works on many Toshiba laptops.

If the computer is in virtual-8086 mode, Toshiba's pop-up menu will display only if the expanded memory manager calls itself "T386." The computer is always in virtual-8086 mode when expanded memory is in use or High RAM has been created. Therefore, when QEMM is performing these services you will not be able to access the pop-up menu. The T386 program makes QEMM appear to be named T386 and allows the menu to work properly. To use T386.EXE:

Type T386 and press Enter

You can load T386.EXE into upper memory by typing LOADHI T386.

You may want to load T386 from your AUTOEXEC.BAT file, so it will run whenever you start your PC. We suggest you run Optimize after adding this or any other program to AUTOEXEC.BAT.

To remove T386 from memory (even if it is loaded into upper memory):

Type T386 R and press Enter.

Return to Technotes Main Menu.

QEMM and Bus-mastering Devices

Certain **SCSI** disk controller cards (and, less frequently, **ESDI** disk controllers and network cards) use a technique called **bus-mastering** to speed up disk access. This technique can cause a conflict when a memory manager (such as QEMM) attempts to load a device driver or TSR into upper memory.

QEMM automatically supports bus-mastering disk controllers. In the vast majority of cases, QEMM can detect a bus-mastering hard disk controller and will take steps to prevent problems. (This is not true if the card controls something other than a hard drive or if QEMM is not being loaded from the bus-mastering hard drive.)

For information on QEMM and bus-mastering devices, refer to our technical note "Bus-Mastering Devices and QEMM" **(BUS-MAST.TEC**).

Return to Hints Main Menu.

QEMM and DESQview or DESQview/X

If you are using **DESQview** or **DESQview/X**, you can increase the amount of memory in each window by using the **StealthROM** feature. To find out if you are using StealthROM, select "Review or change QEMM parameters" from the main QEMM Setup menu and look for the selection "Stealth system and video ROMs." If you see the word **Mapping** or **Frame** at the end of that line, StealthROM is already enabled. If you see the word **Off**, you can enable StealthROM by typing **"S"** or hitting the **Enter** key, then following the on-screen instructions. Online Help will tell you more about the Mapping and Frame methods of StealthROM.

If you have DOS version 5 or 6, QEMM's installation places the command **DOS=HIGH** in your CONFIG.SYS file (if it is not already there). This is a DOS command that loads part of DOS and DOS BUFFERS into the HMA (the first 64K of memory above 1MB).

You may be able to increase the amount of memory in DESQview or DESQview/X windows by deleting DOS=HIGH from CONFIG.SYS. To find out, first run **Memory Status** from inside DESQview or DESQview/X. Make a note of the figure in the bottom right under **Largest Available Expanded Memory**. Then delete DOS=HIGH from your CONFIG.SYS and run Optimize by typing OPTIMIZE at the DOS prompt.

When Optimize completes, run Memory Status from inside DESQview or DESQview/X and check **Largest Available Expanded Memory** again. If it is a larger number than before, you are better off without DOS=HIGH. Otherwise, add the line DOS=HIGH back to CONFIG.SYS and re-run Optimize.

For more suggestions on increasing the size of your DESQview or DESQview/X windows, see the Technical Note "Maximizing Conventional Memory" (**MAXMEM.TEC**).

Return to Hints Main Menu.

QEMM and DR-DOS or Novell DOS

DOS-Up is fully compatible with Novell DOS 7 and DR-DOS 5 and 6. For complete information on using QEMM with DR-DOS 6 or Novell DOS 7, see the text file "QEMM and Novell DOS 7 and DR-DOS 6" (**NW&DRDOS.TEC**).

Return to Hints Main Menu.

QEMM and Disk Compression Software

QEMM is fully compatible with current disk compression software and includes special features for both Stacker and MS-DOS's DoubleSpace and DriveSpace. QEMM's Stealth D*Space feature reduces the memory overhead of DoubleSpace or DriveSpace to as little as 3K. If you are using MS-DOS disk compression, this feature will be displayed on the QSETUP main menu; choose L for "Enable or disable Stealth D*Space", and Yes to enable the feature.

Adding the /QD parameter to the STACKER.INI file can reduce Stacker's overhead to as little as 10K. Please refer to Chapter 1 of the QEMM manual, or to the technotes listed below for details.

If you are using older disk compression software, you may need to take special steps to use QEMM.

Stacker
If you have Stacker versions from 2.01 through 4.0, no special steps are generally required; however, we do suggest you read the technote "QEMM and Stacker" (**STACKER.TEC**) before running Optimize.

SuperStor
Before running Optimize, read "QEMM and SuperStor Disk Compression" (**SSTOR.TEC**).

XtraDrive
Please see the technote "XtraDrive and QEMM" (**XTRADRV.TEC**).

DoubleSpace or DriveSpace
If you are using MS-DOS 6's DoubleSpace or DriveSpace, you can save 31K-49K of memory by using QEMM's Stealth D*Space feature to relocate the DoubleSpace or DriveSpace device driver in expanded memory. See Chapter 5 of the QEMM Reference Manual for details.

Return to Hints Main Menu.

QEMM and Games

Quarterdeck Technical Note #284

Q. Why do I need QEMM to run with my games?

A. In order to create some of the spectacular effects you see while playing your games, the authors sometimes "break the rules." That is, they do not comform to industry standards with regard to memory management. For this reason, you may come across an occasional game that refuses to run when any memory manager is present.

Most games will run with QEMM or another memory manager. Many of them, however, require an 80386 or faster processor, at least 2 megabytes of memory, and a VGA or better graphics card. You may experience problems if your system is not powerful enough to satisfy the needs of these games. In addition, some games refuse to run without 600k or more of available conventional memory. QEMM can make up to 634k conventional memory available on many systems, even after your necessary device drivers and TSRs are loaded. If your problem is related to insufficient conventional memory, QEMM's Optimize program, which loads drivers and TSRs into upper memory, may be the solution.

Q. How should I configure my system using QEMM to get the maximum performance that I require for my games?

A. If you are running low on memory, you should create a "minimal system". A minimal system configuration is one in which you load only the TSRs and drivers absolutely necessary to run the game in question, and nothing else. This will ensure that the game will not only have the memory that it needs to run, but hat the chances of another program or TSR interfering with the game are minimized.

This note will show you how to create a multiple configuration if you are using MS-DOS 6.x, PC-DOS 6.x. or later, or how to create a minimal system boot floppy. A minimal system boot floppy will allow you to insert a floppy disk in the A: drive, boot the computer, and load a configuration from the floppy that is optimal for the games that you are using. Users of MS-DOS 6 and PC-DOS 6 can alternatively set up a "multiple configuration" that allows different configurations to be chosen when booting the computer (thus, placing a floppy in the A: drive is not necessary). Please note that QEMM 7 or later is required to fully support multiple configurations.

CREATING A MULTIPLE BOOT CONFIGURATION

NOTE: THIS SECTION IS FOR PC-DOS & MS-DOS 6.x SYSTEMS ONLY

MS- and PC-DOS 6 and later support multiple configurations, which allow you to choose which group of drivers you would like to load. QEMM fully supports multiple configurations; this section is intended to help you create one quickly and painlessly. If you need any additional assistance, contact the manufacturer of the DOS that you are using.

To create a game configuration:

1) From the DOS prompt, type:

C: <Enter>
CD\ <Enter>
EDIT AUTOEXEC.BAT <Enter>

This will allow you to edit the AUTOEXEC.BAT file on the boot drive.

2) Add the following lines at the very top of this file:

```
GOTO %CONFIG%
:NORMAL
```

3) Go to the bottom of your AUTOEXEC.BAT (hit the down arrow until you are at the end of the file) and add the following lines:

```
GOTO END
:GAME
```

4) Using Copy and Paste, copy the following lines from your NORMAL configuration (everything between the :NORMAL line and the GOTO END line) to the GAME configuration (below the :GAME line):

CD-ROM (commonly MSCDEX.EXE)

Mouse (commonly MOUSE.EXE or MOUSE.COM)

Sound (most likely statements that start with the word SET and/or the lines that are added by your Sound board. Common examples include SBCONFIG and MVAUDIO.)

Path (usually looks like PATH=C:\DOS;C:\ ...)

Prompt (usually looks like PROMPT PG)

Joystick (if you need a driver to run your joystick)

EXAMPLE:

```
GOTO %CONFIG%

:NORMAL
@ECHO OFF
SET TEMP=C:\TEMP
SET NU=C:\NU
SET NORTON=C:\NORTON
REM THE "CHECK" LINE BELOW PROVIDES ADDITIONAL SAFETY FOR
REM STACKED DRIVES. PLEASE DO ONT REMOVE IT.
C:\STACKER\CHECK /WP
PROMPT $P$G
PATH=C:\PROAUDIO;C:\QEMM;C:\DOS;C:\WINDOWS;C:\STACKER;C:\;
SET MOUSE=C:\MOUSE
C:\QEMM\LOADHI /R:2 C:\DOS\MOUSE.EXE
C:\QEMM\LOADHI /R:2 C:\DOS\SHARE.EXE /L:500 /F:5100
C:\QEMM\LOADHI /R:2 C:\DOS\MSCDEX.EXE /D:MSCD001 /M:20
SET BLASTER=A220 D1 I5 T3
IMAGE
C:\QEMM\LOADHI /R:2 C:\DRIVERS\FASTLNK.EXE /Q
GOTO END

:GAME
@ECHO OFF
SET TEMP=C:\TEMP
REM THE "CHECK" LINE BELOW PROVIDES ADDITIONAL SAFETY FOR
```

```
REM STACKED DRIVES. PLEASE DO ONT REMOVE IT.
C:\STACKER\CHECK /WP
PROMPT $P$G
PATH=C:\PROAUDIO;C:\QEMM;C:\DOS;C:\WINDOWS;C:\STACKER;C:\;
SET MOUSE=C:\MOUSE
C:\QEMM\LOADHI /R:2 C:\DOS\MOUSE.EXE
C:\QEMM\LOADHI /R:2 C:\DOS\MSCDEX.EXE /D:MSCD001 /M:4 /E
SET BLASTER=A220 D1 I5 T3
GOTO END

:END
REM -- END OF MULTI --
```

On the line that reads MSCDEX.EXE, if there is an /M:xx, make sure that the number is less than or equal to 15 (/M:15). If it is not, please feel free to change it. Then, if there is not a /E on that line, please add one.

EXAMPLE: MSCDEX.EXE /D:MSCD001 /V /M:4 /E

Go to the very end of the AUTOEXEC.BAT. Hit <Enter> a couple of times to make a blank line and add the following line:

:ND

Save the file and exit.

5) Edit your CONFIG.SYS file by typing the following:

EDIT CONFIG.SYS <Enter>

6) Type the following as the first lines in your CONFIG.SYS:

```
[menu]
menuitem=NORMAL, Normal Configuration
menuitem=GAME, Games Configuration

[NORMAL]
```

7) Go to the bottom of your CONFIG.SYS file. (Press the down arrow until you get to the bottom of the file) and type

[GAME] <Enter>

Copy the following lines from your Normal Configuration:

```
DOSDATA.SYS
QEMM386.SYS
DOS-UP.SYS
SHELL=C:\DOS\COMMAND.COM
FILES
BUFFERS

CD-ROM Driver
(example - DEVICE=C:\CDROM\CDROMDRV.SYS)
```

Sound Driver
(example - DEVICE=C:\SOUND\SBMVAUD.SYS)

Disk Compression Drivers
(examples - DEVICE=C:\STACKER\STACHIGH.SYS
DEVICE=C:\STACKER\STACKER.COM
DEVICE=C:\QEMM\ST-DBL.SYS
DEVICE=C:\DOS\DBLSPACE.SYS)

EXAMPLE:

```
MENUITEM=NORMAL, Normal Configuration
MENUITEM=GAME, Games Configuration

[NORMAL]
DEVICE=C:\QEMM\DOSDATA.SYS
DEVICE=C:\QEMM\QEMM386.SYS RAM X=B000-B1FF R:1 ST:M
DEVICE=C:\QEMM\DOS-UP.SYS @C:\QEMM\DOS-UP.DAT
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\QEMM\QDPMI.SYS ...
DEVICE=C:\STACKER\DPMS.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\STACKER\STACHIGH.SYS
DEVICE=C:\CD-ROM\TSLCDR.SYS /D:MSCD001 /P:3 /S:330
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\ANSI.SYS
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\SETVER.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DRIVERS\SPEEDVID.SYS /VGA
FILES=40
BUFFERS=25
LASTDRIVE=Z
STACKS=0,0
SHELL=C:\QEMM\LOADHI.COM /R:2 C:\DOS\COMMAND.COM /P /E:512

[GAME]
DEVICE=C:\QEMM\DOSDATA.SYS
DEVICE=C:\QEMM\QEMM386.SYS RAM X=B000-B1FF R:1 ST:M
DEVICE=C:\QEMM\DOS-UP.SYS @C:\QEMM\DOS-UP.DAT
DEVICE=C:\STACKER\DPMS.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\STACKER\STACHIGH.SYS
DEVICE=C:\CD-ROM\TSLCDR.SYS /D:MSCD001 /P:3 /S:330
FILES=40
BUFFERS=25
LASTDRIVE=Z
STACKS=0,0
SHELL=C:\QEMM\LOADHI.COM /R:2 C:\DOS\COMMAND.COM /P /E:512
```

If any of the following parameters are on the QEMM386.SYS line, remove them:

```
ST:M ST:F XST=C000 XST=E000 XST=F000
```

Add the following parameters to the QEMM386.SYS line:

```
DMA=128 RH:N SH:N XBDA:L
```

Example:
(Before)

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XST=C000 R:1

(After)
DEVICE=C:\QEMM\QEMM386.SYS RAM DMA=128 RH:N SH:N XBDA:L R:1

Save the file and exit.

For further information on creating Multiple Boot Configurations please refer to your DOS Manual.

9) Type the following:

CD\QEMM <Enter>
Optimize /NOST <Enter>

Select the Games Configuration followed by the Express Optimize option.

10) You are finished with this part - go to ** CONCLUSION **.

CREATING A BOOT DISK

To create a game boot disk,

1) Find a blank, unformatted floppy that will fit in your A: drive, or a floppy disk containing nothing you wish to save.

2) Label this floppy disk "GAME FLOPPY".

3) From the DOS prompt, type:

FORMAT A: /S <Enter>

This will format the floppy disk, and will make it bootable.

4) After the format is complete and you are back at a DOS prompt, type:

COPY C:\CONFIG.SYS A:\ <Enter>
COPY C:\AUTOEXEC.BAT A:\ <Enter>
A: <Enter>

5) Edit your A:\AUTOEXEC.BAT file.

a) DOS 5 users will be able to type EDIT AUTOEXEC.BAT and <Enter>

b) DOS 3 or 4 users will need to use their favorite text editor.

c) DR-DOS users will be able to type EDITOR AUTOEXEC.BAT and <Enter>

6) Make the following changes:

a) Insert the letters "REM " (the word REM followed by a single space) in front of every line EXCEPT the following:

CD-ROM (commonly MSCDEX.EXE)

Mouse (commonly MOUSE.EXE or MOUSE.COM)

Sound (most likely statements that start with the word SET and/or the lines that are added by your Sound board. Common ones are SBCONFIG and MVAUDIO.)

Path (usually looks like PATH=C:\DOS;C:\ ...)

Prompt (usually looks like PROMPT PG)

Joystick (if you need a driver to run your joystick)

b) On the line that reads MSCDEX.EXE, if there is an /M:xx, make sure that the number is equal to or less than 15 (/M:15). If it is not, please feel free to change it. Then, if there is not a /E on that line please add one.

EXAMPLE: MSCDEX.EXE /D:MSCD001 /V /M:4 /E

c) Save the file and exit.

EXAMPLE:

Before

```
==================================
@ECHO OFF
SET TEMP=C:\TEMP
SET NU=C:\NU
SET NORTON=C:\NORTON
REM THE "CHECK" LINE BELOW PROVIDES ADDITIONAL SAFETY FOR
REM STACKED DRIVES. PLEASE DO NOT REMOVE IT.
C:\STACKER\CHECK /WP
PROMPT $P$G
PATH=C:\PROAUDIO;C:\QEMM;C:\DOS;C:\WINDOWS;C:\FUSION;C:\;
SET MOUSE=C:\MOUSE
C:\QEMM\LOADHI /R:2 C:\DOS\MOUSE.EXE
C:\QEMM\LOADHI /R:2 C:\DOS\SHARE.EXE /L:500 /F:5100
C:\QEMM\LOADHI /R:2 C:\DOS\MSCDEX.EXE /D:MSCD001 /M:20
SET BLASTER=A220 D1 I5 T3
IMAGE
C:\QEMM\LOADHI /R:2 C:\DRIVERS\FASTLNK.EXE /Q
```

After

```
==================================
@ECHO OFF
SET TEMP=C:\TEMP
SET NU=C:\NU
SET NORTON=C:\NORTON
REM THE "CHECK" LINE BELOW PROVIDES ADDITIONAL SAFETY FOR
REM STACKED DRIVES. PLEASE DO NOT REMOVE IT.
C:\STACKER\CHECK /WP
PROMPT $P$G
PATH=C:\PROAUDIO;C:\QEMM;C:\DOS;C:\WINDOWS;C:\FUSION;C:\;
SET MOUSE=C:\MOUSE
C:\QEMM\LOADHI /R:2 C:\DOS\MOUSE.EXE
REM C:\QEMM\LOADHI /R:2 C:\DOS\SHARE.EXE /L:500 /F:5100
C:\QEMM\LOADHI /R:2 C:\DOS\MSCDEX.EXE /D:MSCD001 /M:4 /E
SET BLASTER=A220 D1 I5 T3
```

REM IMAGE
REM C:\QEMM\LOADHI /R:2 C:\DRIVERS\FASTLNK.EXE /Q

7) Edit your A:\CONFIG.SYS file.

a) DOS 5 users will be able to type EDIT CONFIG.SYS and <Enter>

b) DOS 3 or 4 users will need to use their favorite text editor.

c) DR-DOS users will be able to type EDITOR CONFIG.SYS and <Enter>

8) Make the following changes:

a) Using the REM command, remark out all lines except the following:

DOSDATA.SYS
QEMM386.SYS
DOS-UP.SYS
SHELL=C:\DOS\COMMAND.COM
FILES
BUFFERS

CD-ROM Driver
(example - DEVICE=C:\CDROM\CDROMDRV.SYS)

Sound Driver
(example - DEVICE=C:\SOUND\SBMVAUD.SYS)

Disk Compression Drivers
(example - DEVICE=C:\STACKER\STACHIGH.SYS
DEVICE=C:\STACKER\STACKER.COM
DEVICE=C:\QEMM\ST-DBL.SYS
DEVICE=C:\DOS\DBLSPACE.SYS)

EXAMPLE:

Before
==========
DEVICE=C:\QEMM\DOSDATA.SYS
DEVICE=C:\QEMM\QEMM386.SYS RAM X=B000-B1FF R:1 ST:M
DEVICE=C:\QEMM\DOS-UP.SYS @C:\QEMM\DOS-UP.DAT
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\QEMM\QDPMI.SYS ...
DEVICE=C:\STACKER\DPMS.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\STACKER\STACHIGH.SYS
DEVICE=C:\CD-ROM\TSLCDR.SYS /D:MSCD001 /P:3 /S:330
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\ANSI.SYS
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\SETVER.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DRIVERS\SPEEDVID.SYS /VGA
FILES=40
BUFFERS=25
LASTDRIVE=Z
STACKS=0,0
SHELL=C:\QEMM\LOADHI.COM /R:2 C:\DOS\COMMAND.COM /P /E:512

After

```
====================================
DEVICE=C:\QEMM\DOSDATA.SYS
DEVICE=C:\QEMM\QEMM386.SYS RAM X=B000-B1FF R:1 ST:M
DEVICE=C:\QEMM\DOS-UP.SYS @C:\QEMM\DOS-UP.DAT
DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\QEMM\QDPMI.SYS ...
DEVICE=C:\STACKER\DPMS.EXE
DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\STACKER\STACHIGH.SYS
DEVICE=C:\CD-ROM\TSLCDR.SYS /D:MSCD001 /P:3 /S:330
REM DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\ANSI.SYS
REM DEVICE=C:\QEMM\LOADHI.SYS /R:2 C:\DOS\SETVER.EXE
REM DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DRIVERS\SPEEDVID.SYS /VGA
FILES=40
BUFFERS=25
LASTDRIVE=Z
STACKS=0,0
SHELL=C:\QEMM\LOADHI.COM /R:2 C:\DOS\COMMAND.COM /P /E:512
```

b) If any of the following parameters are on the QEMM386.SYS line, please remove them:

ST:M ST:F XST=C000 XST=E000 XST=F000

Please add the following parameters to the QEMM386.SYS line:

DMA=128 RH:N SH:N XBDA:L

Example:

(Before)
DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XST=C000 R:1

(After)
DEVICE=C:\QEMM\QEMM386.SYS RAM DMA=128 RH:N SH:N XBDA:L R:1

c) Save the file and exit.

9) Type the following:

C: <Enter>
CD\QEMM <Enter>
Optimize /B:A /NOST <Enter>

This will begin the Optimize process, which will provide as much conventional and upper memory as possible. When asked, select "Express Optimize" to speed the Optimize procedure up.

10) You are finished with this part - go to the ** CONCLUSION ** section at the end of this technical note.

** CONCLUSION **

You are now ready to play!

Q. My game says it *STILL* does not have enough memory. What do I do now?

A. Consult Quarterdeck Technical Note **MAXMEM.TEC** for suggestions.

SPECIAL NOTE: KNOWN INCOMPATIBILITIES

The following games are known to be either incompatible with QEMM, or require special handling. Some games will not run with any memory management software because the game designers are taking memory management into their own hands.

Tornado (by Spectrum Holobyte)
Do not load COMMAND.COM or Stacks high.

Links386 (by Access)
Make sure you are using their latest version, QEMM 7.04 (or later), and add VS:Y to the end of the QEMM386.SYS line in your CONFIG.SYS.

Rebel Assault, SimCity 2000, DOOM. and other DOS4GW v1.9 Extended Games
The DOS-Extender these games use does not function properly with AMI BIOS systems using the Hidden Refresh option. For the games to function properly, disable this option, or obtain the latest release from the game manufacturer.

Comanche (by NovaLogic)
Will not run with any Expanded memory manager by design. You must use HIMEM.SYS or equivalent XMS manager only.

Return to Technotes Main Menu.

QEMM and Microsoft Windows

QEMM is fully compatible with **Microsoft Windows 3.x and WIndows 95**. QEMM automatically gives you 8K-24K more memory for running DOS programs inside Windows 386 enhanced mode.

QEMM 8 includes three exciting new features to speed up Windows and allow users to run more programs simultaneously:

FreeMeg

Resource Manager

MagnaRAM

You can also use QEMM's **VIDRAM** feature to extend the amount of memory available to DOS text-based programs running in Windows by up to 96K. (For information on using VIDRAM, see Chapter 6 in the QEMM Reference Manual.)

In the unlikely event that you experience a Windows-related problem after installing QEMM, refer to the technote **WINFLOW.TEC**.

Return to Hints Main Menu.

QEMM and Stacker

Quarterdeck Technical Note #270

Q. Is Stacker 4 compatible with QEMM 8?

A. Yes. In fact, since Stacker 4 loads itself before DOS on MS-DOS 6 and Novell DOS 7 systems, Stacker 4 is quite transparent to QEMM, which represents a tremendous benefit over previous versions.

Not only is Stacker 4 compatible with QEMM 7.5 and 8, but these versions of QEMM and Stacker 4 include technology jointly developed by Quarterdeck and Stac Electronics, whereby Stacker can greatly reduces the amount of conventional memory it uses; thus Quarterdeck and Stac Electronics recommend that you take advantage of this technology by upgrading to Stacker 4.

The earliest versions of Stacker 4 did not include this technology; a program to update Stacker feature is available to all registered users of Stacker 4, either from Stac Electronics or from Quarterdeck, under the filename S4UP.EXE.

To get your copy of this file, join the CompuServe forum for either Quarterdeck or Stac, by typing GO QUARTERDECK or GO STAC at any CompuServe main prompt. Alternatively, using your modem, call

Stac Electronics BBS **(619) 431-5956**
Quarterdeck BBS UK 01245-496943 or Ireland 353 1 2844381

In order to obtain an upgrade to Stacker 4, or technical assistance from Stac Electronics, call the numbers below.

Stac Electronics Technical Support (619) 431-6712 (voice)

Once you have acquired this file and run the update, you may activate the Stacker feature in this way:

1. If you are currently inside Windows, exit it.
2. At the DOS prompt, change to the Stacker directory.
3. Type ED /I
4. Press Enter to insert a new line.
5. On this new line, type /QD
6. Press Ctrl-Z to exit the editor, and save your changes.
7. Restart your system to put the changes into effect.

Q. I installed Stacker 4 on my system and after running Optimize I found that I have 2K less available conventional memory. Why is this?

A. Stacker 4 now uses Novell's DOS Protected Mode Services (DPMS), through the driver DPMS.EXE, to place most of the Stacker program into extended memory. When you install Stacker 4 on your system, the DPMS.EXE driver will automatically be placed into the CONFIG.SYS file directly above the STACHIGH.SYS device driver line. The use of the DPMS.EXE driver will reduce the size of the Stacker program from about 44K (more or less, depending upon your configuration) to about 17K, and /QD line in STACKER.INI will reduce Stacker's overhead still further. However, the DPMS.EXE driver cannot be loaded into High RAM, so it must load into conventional memory. This will reduce your conventional memory by about 2K, but since your Stacker driver is now much smaller, you should be able to load more programs into High RAM. Further, the Stacker-QEMM technology mentioned above will still further reduce the amount of memory that Stacker uses.

Q. What if I don't want to use the DPMS.EXE driver?

A. If you don't want to use the DPMS.EXE driver, you may remove it from the CONFIG.SYS file. Keep in mind that after removing DPMS.EXE and rebooting, the Stacker driver will be about 44K in size, so you may need to run Optimize after making this change.

Q. How do I go about removing the DPMS.EXE driver from the CONFIG.SYS file?

A. Stacker 4 has a new configuration program called CONFIG.EXE. To see what changes you can make to Stacker's configuration, simply type:

CONFIG /? <ENTER>

Two of the listed options that will display are:

/D Adds Stacker DPMS driver to configuration files. /D- Removes Stacker DPMS driver from configuration files.

If you want to remove the DPMS.EXE driver from the CONFIG.SYS file, simply type:

CONFIG /D- <ENTER>

You will then be asked if you are sure you want the CONFIG.SYS changed. To save the changes made, press "Y".

Q. Is there any other way I can reduce the size of the Stacker driver?

A. If you want to reduce the size of the Stacker driver without using DPMS.EXE, you may still use the /EMS parameter to load Stacker into expanded memory. This is much less advantageous than the /QD parameter, and is recommended neither by Stac Electronics nor by Quarterdeck.

If you add the /EMS parameter to the STACKER.INI file and you want to use QEMM's STEALTH technology, you must add a DBF:2 parameter to the QEMM device line in the CONFIG.SYS file (this can be done from ED, too).

Q. Is Stacker 4 compatible with QEMM's ST-DSPC.SYS (Stealth D*Space) driver?

A. No. Even though Stacker uses a file called DBLSPACE.BIN in order to load before the CONFIG.SYS file loads, it is not sufficiently similar to Microsoft's DoubleSpace or DriveSpace programs to allow ST-DSPC.SYS to work. Stealth D*Space will only work with Microsoft's DOS 6 DoubleSpace or DriveSpace.

Q. After I installed Stacker 4 over my DoubleSpace program, I received the following message:

ST-DBL: DBLSPACE is not in use, so there is no need to load ST-DBL.SYS.

A. The Stacker 4 install does not detect or remove QEMM's ST-DBL.SYS or ST-DSPC.SYS drivers, one of which may be in your CONFIG.SYS file if you were using Microsoft's disk compression prior to installing Stacker. Upon installation, Stacker simply places a device line in the CONFIG.SYS file that reads

DEVICE=C:\STACKER\STACHIGH.SYS

at the end of the CONFIG.SYS file. If you were previously loading the ST-DBL.SYS device driver with a multi-config system, replace every ST-DBL.SYS device line with the STACHIGH.SYS device line

above. If you are loading it from a single boot CONFIG.SYS, simply replace the one incidence of ST-DBL.SYS.

Q. What are the different sizes of the Stacker driver?

A. The size of the driver is strongly dependend on the size of our hard drive and the size of Stacker's compressed clusters. If you are using Stacker with DPMS.EXE and the /QD parameter, the driver's resident size will be as little as 10K. Without the /QD parameter, the driver will typically be at least 8K larger. If you are using Stacker's /EMS switch, the driver should be at least 25K. If you are not using DPMS.EXE or the /EMS switch, the driver should be at least 44K. The initialization size, the size necessary to load the driver before it shrinks down to its resident size, is 87K no matter what parameters you use.

Stacker 3.1 and earlier:

As mentioned above, both Quarterdeck and Stac Electronics strongly recommend upgrading to Stacker 4. This portion of this document discusses issues related to Stacker versions 2.01 through 3.1. Most references to Stacker will be without a version number, except in those instances where it is necessary to specify a particular version.

Any version of Stacker 2.01 or later properly detects the presence of QEMM, regardless of whether or not you install QEMM first. If you install QEMM after installing Stacker and you run OPTIMIZE, Stacker will detect OPTIMIZE and copy the pertinent QEMM files from the COMPRESSED drive to the UNCOMPRESSED boot drive.

1. This copy process is usually successful, but if you do not have enough room on your UNCOMPRESSED drive to hold the QEMM files, you must use the Stacker utility STAC.COM to increase the size of the UNCOMPRESSED drive and then either run OPTIMIZE again or manually copy over the correct files to the drive. The following is a list of those files needed on the UNCOMPRESSED boot drive:

QEMM386.SYS
OPTIMIZE.COM
LOADHI.SYS
TESTBIOS.COM
LOADHI.COM
BUFFERS.COM
RSTRCFG.SYS
WINHIRAM.VXD
WINSTLTH.VXD
MCA.ADL (if a Micro Channel machine, typically an IBM PS/2)

To increase the size of the UNCOMPRESSED partition, through Stacker, type:

STAC <ENTER>

at the DOS prompt. For further information regarding the STAC.COM program, please refer to your Stacker manual.

2. IF YOU ARE NOT USING SSWAP.COM

If you are NOT using the SSWAP.COM program to swap drive names, then Stacker will not detect the presence of OPTIMIZE and copy the correct files to the UNCOMPRESSED drive. However, this should not be a problem because you will most likely have already installed QEMM on the UNCOMPRESSED drive.

3. "/SYNC" PARAMETER WITH SSWAP.COM

If you are using the SSWAP.COM program, in order to maintain compatibility with OPTIMIZE, you MUST have the "/SYNC" parameter at the end of the SSWAP.COM line. The parameters differ slightly between version 2.01 and 3.0. Examples of the two versions are below:

DEVICE=C:\STACKER\SSWAP.COM C:\STACVOL.DSK /SYNC (VERSION 2.01)

DEVICE=C:\STACKER\SSWAP.COM C:\STACVOL.DSK /SYNC+ (VERSION 3.00)

Stacker places the "/SYNC" parameter at the end of the SSWAP.COM line during installation. It is only discussed in this document because sometimes it is accidentally deleted.

The /"SYNC" parameter for Stacker 3.0 has a "+" sign at the end. The "+" tells SSWAP.COM to AUTOMATICALLY update any changed files, such as CONFIG.SYS, that are supposed to be on both drives. If you delete the "+" from the "/SYNC" parameter, SSWAP.COM will only NOTIFY you of changes to files and ask if you want to synchronize them.

As of Stacker version 3.0 some compatibility issues with OPTIMIZE remained unresolved. These issues may require some troubleshooting as well as editing of your CONFIG.SYS and AUTOEXEC.BAT files.

Stacker detects the presence of OPTIMIZE and allows you to run it without having to first edit your CONFIG.SYS and AUTOEXEC.BAT files. If you are using SSWAP.COM, it will detect when OPTIMIZE is being run, make changes to the drive references in the CONFIG.SYS and AUTOEXEC.BAT files, then prompt you to press a key to reboot the machine again for OPTIMIZE. This will occur during the Detection and Final Phases of OPTIMIZE. Don't worry if your machine reboots itself several times during OPTIMIZE; this is normal.

If you are using DOS 5 with Stacker, and you run OPTIMIZE, the number of buffers may disappear from the line in the AUTOEXEC.BAT file or CONFIG.SYS file, depending on whether or not you are loading DOS into the HMA. To fix this problem, simply edit the AUTOEXEC.BAT or CONFIG.SYS file after the OPTIMIZE process is complete and add the number of buffers you want to the C:\QEMM\LOADHI line in the AUTOEXEC.BAT or the BUFFERS= line in the CONFIG.SYS file.

4. "INCORRECT QEMM OPTIMIZE"

Although it is not common, there may be times during OPTIMIZE when you will receive the message: "Incorrect QEMM Optimize". To fix this problem, you must edit your CONFIG.SYS file on the UNCOMPRESSED drive and remove the line that reads:

DEVICE=C:\QEMM\RSTRCFG.SYS **** OPTIMIZE D%etection %P%hase ****

After removing the line and saving the file, reboot your machine and re-run OPTIMIZE.

5. OPTIMIZE.EXC and SSWAP.COM

Some earlier versions of QEMM LOADHI.SYS driver may have a conflict with Stacker's SSWAP.COM. Although Stacker now creates an OPTIMIZE.EXC file which tells the OPTIMIZE program to NOT place a LOADHI.SYS line in front of SSWAP.COM, you must make sure that a LOADHI.SYS line is not ALREADY in front of SSWAP.COM. If it is, you must remove it before running OPTIMIZE.

6. STACKER WITH "/EMS" SWITCH

Stacker can put its built-in cache into EMS, which reduces the amount of conventional memory the STACKER.COM driver requires. You can select this option when you are installing Stacker on your

hard drive. If you are using the STEALTH option with QEMM, however, you must make sure that you have a "DBF=2" parameter at the end of the QEMM line. This is because when STACKER.COM uses EMS, it accesses the disk via the EMS Page Frame at the same time that STEALTH is using the Page Frame. "DBF=2" buffers all disk read and writes that directly access the Page Frame and thus prevents a conflict.

Stacker 3.0 should automatically place this parameter at the end of the QEMM line for you, but 2.01 does NOT. If you are using Stacker 2.01, you MUST add this parameter manually. Below is a sample QEMM device line with the "DBF=2" parameter:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M DBF=2

7. "LOADHI: This program took over INT 67!"

If you are using STACKER.COM with the "/EMS" parameter, under certain circumstances you might get the above error message. To get this message, your QEMM / Stacker configuration has to be as follows.

1. STACKER.COM is using the "/EMS" parameter which puts Stacker's built-in cache into EMS.

2. You are NOT using QEMM's "STEALTH" parameter.

3. QEMM's LOADHI.COM driver is loading, from the COMPRESSED Stacker drive, a driver or TSR into HIGH RAM, with the LOADHI.SYS "SQUEEZEF" (Squeeze Frame) parameter. "SQUEEZEF" uses the Page Frame temporarily, to give a driver or TSR enough room to initialize. Since ANY driver or TSR loading from the COMPRESSED Stacker drive becomes an EMS user, and subsequently uses the Page Frame at the same time as "SQUEEZEF", the two become incompatible. Hence, the error message above.

To remedy this problem, you have a number of options:

1. Use QEMM with the "STEALTH" parameter. Since "STEALTH" uses the Page Frame, "SQUEEZEF" does NOT work with it. However, the "SQUEEZET" (Squeeze Temp) parameter which temporarily uses areas of HIGH RAM to give a driver or TSR enough room to initialize does work.

2. If you do not want to use STEALTH, the easiest remedy is to rerun OPTIMIZE with the "/NF" parameter. This tells OPTIMIZE to NOT use the "/SQUEEZEF parameter.

3. A final option would be to simply load all your drivers and TSR's from the UNCOMPRESSED boot drive. For instance, if you are using Stacker with SSWAP.COM, and a sample line in your AUTOEXEC.BAT file looks like the following:

C:\MOUSE\MOUSE

then you must change it to read:

D:\MOUSE\MOUSE

After SSWAP.COM has swapped your drive names, the D: drive is your UNCOMPRESSED drive. If you are loading a driver from the CONFIG.SYS file, and SSWAP.COM is the last line, there is no need to make any changes to that file.

The suggestions included in this technote should take care of most of the problems you are likely to encounter with either Stacker version 2.01 or 3.0.

Return to Technotes Main Menu.

QEMM and SuperStor

Quarterdeck Technical Note #249

This troubleshooting guide is designed to help the individual who has QEMM and SuperStor 2.04 or SuperStor Pro installed on his or her machine and has run, or wants to run, OPTIMIZE. If you have not yet run OPTIMIZE, please read the General Information section below, then proceed to either Section 1 or Section 2 depending on if you installed QEMM BEFORE or AFTER installing SuperStor.

GENERAL INFORMATION

1. To avoid confusion we will refer to COMPRESSED and UNCOMPRESSED drives whenever possible. The COMPRESSED drive, which is actually a very large hidden file with a drive ID, will always be the one that SuperStor creates to hold most of your programs and data.

2. Since you cannot boot from the COMPRESSED drive, SuperStor sets aside a small portion of the UNCOMPRESSED drive to boot from. Depending upon which version of SuperStor you have, this partition may or may not be easily accessed.

3. Your UNCOMPRESSED drive is always drive C: before bootup. When the SuperStor driver (SSTORDRV.SYS) executes during bootup it then allows access to the large COMPRESSED partition and gives it the next available drive ID, usually D: This partition is really a very large hidden file called SSPARTSS.ADD (if you have access to both the COMPRESSED and UNCOMPRESSED drives, the file will be called SSPARTSS.SWP).

4. After the SuperStor driver executes, another driver (DEVSWAP.COM) usually follows it. When this driver executes, it swaps drive names so that the COMPRESSED partition becomes C: and the UNCOMPRESSED partition becomes D:. Basically, it is just reversing the drive names so that your applications will still think that they are on drive C:

5. If you have allowed SuperStor to compress the entire hard drive, you will not be able to access the UNCOMPRESSED partition after boot, unless you run a utility called ADD2SWP.EXE, found on your SuperStor program diskette. From the diskette, just type :

ADD2SWP C:<Enter>

This will then give you access to both your COMPRESSED and UNCOMPRESSED drives.

6. If you have configured SuperStor to reserve space on the UNCOMPRESSED drive to be accessed by a different drive ID you will already be able to access the UNCOMPRESSED partition easily. You are given the option to reserve this space when you are running the SSTOR program to create your COMPRESSED drive(s). In version 2.04 the default is NO. If you press ENTER when encountering this option SSTOR will set-up SuperStor to compress your whole hard drive.

7. If you have not yet installed SuperStor 2.04, Quarterdeck recommends that you select YES to override the default. Beginning with SuperStor Pro version 1.0, the default is YES. You only have to press ENTER at this prompt. When the program then asks you for a size to make the UNCOMPRESSED drive enter 2048 to add two megabytes to that drive size.

8. To determine which drive is COMPRESSED or UNCOMPRESSED start the SSTOR.EXE program in the C:\ADDSTOR directory. Under the heading "System Device List" will be information pertaining to all drives on your computer.

9. If before you installed SuperStor your hard disk drive had more than one partition, i.e., a C: and D: partition, SSTORDRV.SYS will select the next available drive name for its container file. If you had a C:

and D: partition and you COMPRESSED the C: drive only then your drives will be set up as follows when SuperStor is installed:

C: becomes E: D: is still D: unless you compress that as well. If you do compress D:, it becomes F:

The troubleshooting guide below is based on a one-partition drive. If your drive has multiple partitions then you must determine what drive C: swaps to and use the appropriate drive name.

10. If you are using a disk caching program with SuperStor, be very careful to determine whether it is compatible with SuperStor or not. For instance Microsoft Windows' 3.1 SMARTDRV.EXE will lock up your system upon bootup, if you load the SSTORDRV.SYS driver into High Ram with LOADHI.SYS. At the time of this writing, Norton NCACHE version 6, will not work under any circumstances with SuperStor installed on your machine.

****

BE ABSOLUTELY CERTAIN THAT YOU READ THE README.TXT FILE IN YOUR ADDSTOR DIRECTORY.

****

11. You should always have a bootable floppy diskette available which will execute SuperStor and give you access to your COMPRESSED drive. To create this format a floppy diskette with the /S parameter so that it becomes a boot disk.

When this disk has been formatted copy the following files from the C: drive to the floppy diskette: SSTORDRV.SYS, DEVSWAP.COM, CONFIG.SYS, AUTOEXEC.BAT. Once those files have been copied over check the contents of them. The CONFIG.SYS file should contain at least the following lines:
FILES=40 DEVICE=\SSTORDRV.SYS /NOHIGH (SuperStor Pro requires /NOUMB instead)
DEVICE=\DEVSWAP.COM

12. If you want QEMM's LOADHI.SYS driver to load SuperStor version 2.04 into High RAM you must place the "/NOHIGH" parameter at the end of the SuperStor device line. If you are using SuperStor Pro you must use the "/NOUMB" parameter instead of "/NOHIGH." SuperStor does not automatically place either of the parameters on the line. The line should read as follows:

DEVICE=\SSTORDRV.SYS /NOHIGH (SuperStor 2.04)

or

DEVICE=\SSTORDRV.SYS /NOUMB (SuperStor Pro)

The AUTOEXEC.BAT file should contain at least the following lines:

PATH=C:\;C:\DOS;C:\QEMM;C:\ADDSTOR PROMPT=PG

These files on the boot floppy will allow you to access your SuperStor COMPRESSED drive and navigate the hard drive.

Section Two: SuperStor Installed First

If SuperStor is already installed on your hard drive and you now install QEMM and run OPTIMIZE then you must follow the section below.

1. If you install QEMM and run OPTIMIZE on a system with SuperStor already installed, SuperStor will fail to execute during the first reboot of the OPTIMIZE program. This is because OPTIMIZE places device statements on the SSTORDRV.SYS line, but the files necessary to complete the boot cannot be found on the UNCOMPRESSED boot drive.

2. Since SuperStor fails to load, you will only have access to your UNCOMPRESSED drive after bootup. It is an easy process to correct this situation. All you have to do is create a new, temporary, CONFIG.SYS file on the UNCOMPRESSED drive, now drive C:, and then reboot. To do this first rename the old CONFIG.SYS file by typing:

RENAME CONFIG.SYS CONFIG.XXX<ENTER>

Now, create a new CONFIG.SYS file by typing the following:

COPY CON CONFIG.SYS<ENTER>

The cursor will now be below the "C" in COPY. Now, type:

DEVICE=\SSTORDRV.SYS<ENTER> DEVICE=\DEVSWAP.COM<ENTER>

NOTE: IF YOU HAVE ANY OTHER DRIVERS, SUCH AS A DISK PARTITIONER, THAT ARE ESSENTIAL TO BOOT YOUR MACHINE INCLUDE THOSE IN THE TEMPORARY CONFIG.SYS FILE. IF YOU DO NEED TO BOOT WITH A DISK PARTITIONER MAKE SURE IT LOADS BEFORE SSTORDRV.SYS. DON'T WORRY ABOUT ANY OTHER DRIVERS YOU MAY HAVE HAD IN THE ORIGINAL CONFIG.SYS FILE: ONCE YOU HAVE HAVE REBOOTED THE MACHINE YOUR ORIGINAL CONFIG.SYS FILE WILL BE INTACT.

Now, press the F6 key and <ENTER>. You will see the following message:

1 file(s) copied.

3. Reboot the machine.

4. Because of SuperStor program designs the untouched CONFIG.SYS file on the COMPRESSED drive will automatically overwrite the temporary CONFIG.SYS file on the UNCOMPRESSED drive. The contents of both CONFIG.SYS files will then be the same as before you installed QEMM and ran OPTIMIZE.

********** FULL DISK COMPRESSION **********

IF YOU INSTALLED SUPERSTOR TO COMPRESS YOUR WHOLE HARD DRIVE, PROCEED TO STEP 2 IN "OPTIMIZING WITH QEMM AND SUPERSTOR."

********** PARTIAL DISK COMPRESSION **********

IF YOU INSTALLED SUPERSTOR TO COMPRESS ONLY A PORTION OF YOUR HARD DISK DRIVE, PROCEED TO STEP 2 IN "OPTIMIZING WITH QEMM AND SUPERSTOR."

ONCE YOU HAVE COMPLETED STEP 2, PROCEED TO EITHER STEP 4 OR 6, DEPENDING UPON WHETHER OR NOT YOU HAVE LEFT ADEQUATE ROOM ON YOUR UNCOMPRESSED PARTITION TO HOLD ALL QEMM AND DOS FILES.

Section Three: QEMM Installed First

If QEMM is currently installed on your hard disk drive and you now install SuperStor, you then need to follow the instructions in the section below.

IMPORTANT NOTE: As of SuperStor Pro 1.0, Addstor recommends that if you have a memory resident program, such as a memory manager, installed and running, you must disable the memory resident program BEFORE installing SuperStor. At Quarterdeck, we have installed SuperStor Pro successfully with QEMM running. If you choose to follow their advice, disable QEMM, and your memory resident programs, by placing a REM statement at the beginning of the line that loads such a program.

1. After you install SuperStor you then have to run SSTOR.EXE to do the actual compression of your hard drive. The default of SuperStor 2.04 is to compress all of the hard drive space. As of SuperStor Pro 1.0, the default is to leave a portion (the size is specified by the user) of the hard drive uncompressed. No matter which version you are using, you should make sure you leave at least a 2 MB portion uncompressed.

2. If you accept the default of SuperStor 2.04 to compress the whole hard disk drive or if you override the default of SuperStor Pro, SuperStor leaves a tiny boot partition with only COMMAND.COM and other system files, as well as the SuperStor drivers resident. If QEMM is installed when you install SuperStor, the QEMM386.SYS device driver will be placed into the root directory of the boot drive.

Also, the QEMM line in the CONFIG.SYS will be changed from:

DEVICE=C:\QEMM\QEMM386.SYS RAM

to

DEVICE=C:\QEMM386.SYS RAM

3. Upon installation, SuperStor does not create a QEMM directory on the UNCOMPRESSED drive, nor does it copy over the pertinent QEMM files from the COMPRESSED drive.

4. If you have accepted the default of compressing the whole drive, the drive letter D: is not readily accessible. How to access the UNCOMPRESSED drive will be explained in detail later in this troubleshooting guide. If you have only compressed a portion of the hard disk drive you already have access to the UNCOMPRESSED portion of the drive.

5. The net result of the above is that after installing SuperStor, compressing the hard disk, and rebooting the machine, QEMM will execute, but any lines in the CONFIG.SYS that precede SSTORDRV.SYS and that have LOADHI.SYS statements will fail. After following the instructions below in "Optimizing with QEMM and SuperStor" those files will then be in the correct directory for OPTIMIZE to perform.

********** FULL DISK COMPRESSION **********

If you installed SuperStor to compress your whole hard disk drive, proceed to Step 1 in "Optimizing with QEMM and SuperStor."

********** PARTIAL DISK COMPRESSION **********

If you have installed SuperStor to compress only a portion of your hard disk drive, proceed to Step 1 in "Optimizing with QEMM and SuperStor." Once you have completed Step 2, proceed to either Step 4 or Step 6, depending on whether you have left adequatre room on your uncompressed partition to hold all QEMM and DOS files.

Section Four: Optimizing with QEMM and SuperStor

BEFORE FOLLOWING THE STEPS BELOW MAKE CERTAIN THAT YOU HAVE READ ALL THE INFORMATION ABOVE PERTAINING TO THE ORDER AND METHOD USED TO INSTALL QEMM

AND SUPERSTOR ON YOUR COMPUTER. ALSO MAKE SURE ANY IMPORTANT FILES HAVE BEEN BACKED UP. THIS INCLUDES BOTH THE CONFIG.SYS AND AUTOEXEC.BAT FILES AND ANY OTHER FILES YOU FEEL ARE IRREPLACEABLE.

* IF YOU HAVE CONFIGURED SUPERSTOR TO COMPRESS YOUR WHOLE HARD DISK DRIVE, PLEASE FOLLOW THE STEPS BELOW FROM 1 TO 16. IF YOU INSTALLED QEMM AFTER INSTALLING SUPERSTOR YOU MAY SKIP STEP 1.

* IF YOU HAVE CONFIGURED SUPERSTOR TO COMPRESS ONLY A PORTION OF YOUR HARD DISK DRIVE BUT DID NOT LEAVE ENOUGH ROOM ON THE UNCOMPRESSED PORTION TO COPY ALL QEMM AND DOS FILES TO THAT PORTION, PLEASE FOLLOW THE STEPS BELOW FROM 4 TO 16.

1. On the C: COMPRESSED drive, edit the QEMM device line in the CONFIG.SYS file to read:

DEVICE=C:\QEMM\QEMM386.SYS RAM

Save the CONFIG.SYS file.

2. Place the SuperStor program diskette into drive A:. Log onto drive A: and type:

ADD2SWP C: <ENTER>

The program will then report:

ADD2SWP has completed, your PC needs to be rebooted. Press the [ENTER] key to reboot the PC

3. Remove the diskette and press ENTER. After the computer completes its reboot you will then have access to both your COMPRESSED and UNCOMPRESSED drives.

4. Start SSUTIL.EXE by typing:

SSUTIL<ENTER>

at the C: prompt, and run SHRINK DISK from within that program. Select an amount to shrink the COMPRESSED disk by. Quarterdeck recommends a minimum of 1024 Kb with 2048 Kb preferred.

5. EXIT and REBOOT.

THE NEXT STEPS CAN BE FOLLOWED IF YOU HAVE CONFIGURED SUPERSTOR TO COMPRESS ONLY A PORTION OF YOUR HARD DISK DRIVE AND LEFT ENOUGH ROOM FOR ALL QEMM AND DOS FILES OR IF YOU HAVE JUST FOLLOWED STEPS 1-4.

6. Log onto drive D: the UNCOMPRESSED drive. Create a QEMM directory by typing the following:

CD\ <ENTER> MD QEMM <ENTER>

7. Copy all QEMM files from the COMPRESSED drive into the QEMM directory on the UNCOMPRESSED drive.

8. While still on the UNCOMPRESSED drive D: create directories pertaining to all other drivers loading from the CONFIG.SYS file. For example, if you have a couple of lines in your CONFIG.SYS file that read:

DEVICE=C:\MOUSE\MOUSE.SYS DEVICE=C:\DOS\ANSI.SYS

then create a MOUSE directory and a DOS directory on the UNCOMPRESSED drive. Then, copy the MOUSE.SYS and ANSI.SYS drivers from the CONPRESSED drive to their respective directories on the UNCOMPRESSED drive.

Do the same for all other drivers loading from the CONFIG.SYS file.

9. Log back on to drive C:, the COMPRESSED drive. Edit the CONFIG.SYS file in the following ways:

A. Move the two SuperStor device lines to the bottom of the CONFIG.SYS file.

B. If you haven't already done so, place a blank space and the "/NOHIGH" parameter (for SuperStor 2.04 only) or the "/NOUMB" parameter (for SuperStor Pro only) at the end of the SSTORDRV.SYS device line.

C. Place a REM and a blank space in front of the line that reads:

DEVICE=\DEVSWAP.COM

so it now reads:

REM DEVICE=\DEVSWAP.COM.

Save the CONFIG.SYS file but do NOT reboot yet.

IMPORTANT NOTE: YOU MUST PLACE THE "/NOHIGH" PARAMETER (for SuperStor 2.04) OR THE "/NOUMB" PARAMETER (for SuperStor Pro) AT THE END OF THE \SSTORDRV.SYS DEVICE LINE TO ALLOW OPTIMIZE TO SUCCESSFULLY LOAD THE DRIVER INTO HIGH RAM. THE SIZE OF THE DRIVER, SSTORDRV.SYS, CAN VARY WHEN IT IS LOADED INTO HIGH RAM. THE LARGEST SIZE WE HAVE SEEN AT QUARTERDECK IS 71K.

10. Still on the COMPRESSED C: drive, edit the AUTOEXEC.BAT file and reverse all drive designations from C: to D: or D: to C:. For example if a line reads:

C:\MOUSE\MOUSE

change it to read:

D:\MOUSE\MOUSE

Make sure you also reverse the drive designations in your path statement, i.e.,

PATH=C:\;C:\DOS;C:\QEMM;C:\ADDSTOR

becomes:

PATH=D:\;D:\DOS;D:\QEMM;D:\ADDSTOR

If you are using the CALL command to CALL batch files from the AUTOEXEC.BAT file, make sure that the drive designations in the CALLed batch file are also reversed. For instance, if you have a line in your AUTOEXEC.BAT file that reads:

@CALL C:\NETWORK\LOADNET.BAT

and logs you onto a network make sure you reverse the drive designations in both the above line and in the batch file LOADNET.BAT.

11. Save the AUTOEXEC.BAT file and copy to the D:\ drive.

12. Reboot the computer.

13. Run OPTIMIZE.

14. After OPTIMIZE is complete edit the CONFIG.SYS file on the UNCOMPRESSED drive and remove the REM and blank space in front of the DEVSWAP.COM device line. It's very important that you edit the CONFIG.SYS file on the UNCOMPRESSED drive which will be C:. After editing save the file and copy it to the COMPRESSED D: drive by typing:

COPY C:\CONFIG.SYS D:\ <ENTER>

15. Edit the AUTOEXEC.BAT file on the UNCOMPRESSED C: drive and once again reverse the drive designations. Be very careful that you find ALL drive designations. Some lines may have more than one drive designation in the same command so watch out for that.

After editing the AUTOEXEC.BAT file on the UNCOMPRESSED C: drive save and copy it to the COMPRESSED drive D:.

16. The OPTIMIZE process is now complete. Reboot the computer which will once again swap your drive names and your work is done.

Return to Technotes Main Menu.

QEMM and XtraDrive

Quarterdeck Technical Note #199

XtraDrive is a disk compression program published by Integrated Information Technology. While this program is generally compatible with QEMM, some issues must be addressed. Below are some of the most frequently asked questions about XtraDrive and QEMM.

STEALTH ROM:

Q: When I try to install XtraDrive on a system that has QEMM and Stealth ROM running, I get the following error message:

> **"Drive 1 is being controlled by a program that appropriates INT13. You cannot install XtraDrive on this drive."**

What do I have to do to fix this?

A: Remove the StealthROM parameter (ST:M or ST:F) from the QEMM device line and try to install XtraDrive again. You should only need to do this during the installation of XtraDrive. Once XtraDrive is installed, you can restore the StealthROM parameter.

Q: So, I can use XtraDrive with QEMM's StealthROM parameter?

A: Under most circumstances, StealthROM works fine with XtraDrive, but if you are using the XtraDrive EMS cache you cannot use either of QEMM's StealthROM parameter (ST:M or ST:F). The XtraDrive EMS cache accesses the disk via the page frame and this causes a conflict with StealthROM. Normally, if a program accesses the disk via the page frame, QEMM's DBF=2 parameter will fix the problem. This is not possible with XtraDrive. The EMS cache will also cause problems with drivers such as EMSNETX.COM and Microsoft's MOUSE.COM using EMS.

Q. Are there any other possible conflicts with XtraDrive and QEMM's StealthROM parameter?

A. Under certain rare circumstances you may need to exclude a 4K region in the system ROM BIOS area. If, after installing XtraDrive and QEMM with the StealthROM parameter, you get an Exception 13 or lockup you should refer to Technical Bulletin #205, **STEALTH.TEC**. This technote will help you determine which area must be excluded. Keep in mind, however, that the need to exclude the 4k block of High RAM is very rare.

EXCLUDES:

Q: When I install QEMM on a system that has XtraDrive currently installed, I get a lockup or Exception 13 when I reboot the machine, loading QEMM. What do I have to do to allow me to boot the system?

A: Simply reboot the machine and when it beeps after checking the memory, hold down the ALT key until prompted to press ESC to unload QEMM. (If you are using QEMM DOS-UP feature, you will first be prompted to press ESC to unload DOSDATA.) Once you have unloaded DOSDATA and/or QEMM, load your CONFIG.SYS file into a text editor. If you have MS-DOS 5 or MS-DOS 6, you may load the DOS Editor, by typing EDIT C:\CONFIG.SYS and <ENTER>. At the end of the QEMM device line, place the following parameter: X=9000-9FFF.

A sample QEMM device line would then read as follows:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=9000-9FFF

After adding the parameter, simply save the CONFIG.SYS file and reboot your machine.

Q: When I installed XtraDrive after QEMM, XtraDrive placed an exclusion from 8000-A000 on the QEMM device line. Do I need this exclusion and will it take away any memory from my system?

A: Given the way that XtraDrive is currently designed, an exclusion is needed. If you are using DESQview or DESQview/X, the exclusion will tell QEMM to not map memory into that area, so the amount of memory available to each DESQview window will be smaller. There are a couple of things you can do to increase your window size if you are using DESQview or DESQview/X:

> **1.** The default exclude is 8000-A000, a 128K block. Our testing has determined that usually an exclusion of 9000-9FFF will be sufficient. This will decrease your window size by only 64K. If you are using the CONVENTIONAL MEMORY disk cache, and have selected more than two 16K blocks of conventional memory, you will need to change the exclude back to 8000-9FFF.
>
> **or**
>
> **2.** Load the XtraDrive driver BEFORE QEMM and take the exclude off the QEMM device line. This should decrease your window size by only about 46K. Once again, if you are using the CONVENTIONAL MEMORY disk cache, your driver size will increase with each 16K block you add. The default is one 16K block. If you select two blocks, your driver will be 66K. If you select three it will be 82K. The maximum, four 16K blocks will result in a 98K driver. Your DESQview and DESQview/X window sizes will decrease accordingly.

BUS-MASTERING:

Q: If I have a SCSI bus-mastering controller, can I load the XtraDrive device driver into High Ram?

A: No. Even with double buffering, on a system with a SCSI bus-mastering device you will not be able to load XtraDrive into High Ram. This is NOT a memory manager issue, but a limitation of XtraDrive.

Return to Technotes Main Menu.

QEMM and the EMS Page Frame

Most programs that use expanded memory (**EMS**) access up to 64K of expanded memory at a time (in 16K units called pages) at a special area of upper memory called the **page frame**. An expanded memory manager (QEMM) makes memory from outside the first megabyte of memory appear in the page frame.

QEMM's installation normally reserves 64K of upper memory for use as a page frame. You can use QSETUP to specify that no page frame should be used; however, we strongly recommend that you do NOT do this. See the technote "Why the EMS Page Frame is Important" (**FRAME.TEC**).

The following applications use (or can be configured to use) expanded memory:

Applications

Networking Software

Utilities

Disk Caches

Games

Return to Hints Main Menu.

QEMM with MS-DOS 5.0

Quarterdeck Technical Note #200

Q. Is QEMM compatible with DOS 5?

A. The shipping versions of all Quarterdeck products are compatible with IBM and Microsoft DOS 5.

DOS 5 contains a number of enhancements over previous versions. Among the most notable features is more advanced memory management--specifically, the ability to create and use regions of upper memory above 640K. These regions can be used for loading resident programs, drivers, and parts of DOS itself.

In many ways the facilities for managing memory that are built into DOS 5 are similar to those first made available to users of DOS 2.x through 4.x with the release of QEMM version 4.1 in the spring of 1988. QEMM, now at version 8, has evolved considerably from that original product, incorporating improvements with each new version. These improvements have progressively resulted in more available memory for the user, higher reliability and enhanced ease of use.

While the memory management features of DOS 5 represent an advance for DOS, Quarterdeck's memory managers still provide significant advantages for DOS 5 users.

Q. What advantages do Quarterdeck's memory managers have over DOS' memory managers?

A. The differences between the current release of Quarterdeck memory managers and those built into DOS 5 are as follows:

1) The combined size of DOS 5's memory managers (HIMEM.SYS and EMM386.SYS) is 8 to 10K. QEMM provides the facilities of both these drivers in less than 1K!

2) QEMM typically provides 96K MORE High RAM by default than the DOS 5 memory managers on non-PS/2 systems, and 128K MORE by default on PS/2 systems. The DOS 5 memory manager allows these areas to be included manually, but this requires some expertise.

3) QEMM's Squeeze technology allows larger resident programs to be loaded by allowing them to squeeze--temporarily--into the EMS page frame or areas used by Adapters when loading. The DOS 5 memory managers do not provide a similar feature.

4) Using its Stealth option, QEMM can provide EVEN MORE high RAM (211k total is common) by hiding machine ROMs and allowing High RAM to be mapped over them. Neither DOS 5's memory managers nor any other third-party memory manager currently provides this patent-pending technology.

5) DOS 5 has no equivalent to Quarterdeck's Manifest program. Manifest is a memory analysis program included free with QEMM, DESQview, and DESQview 386. It provides extensive information about the computer on which it is running and is an invaluable tool when optimizing a system or diagnosing a memory problem or conflict.

6) The DOS 5 memory manager provides no program to automatically and OPTIMALLY load TSR's and device drivers into the regions of upper memory. Novice users may experience considerable difficulty achieving good results from the DOS 5 LOADHIGH program, but even advanced users will appreciate the speed and accuracy with which OPTIMIZE sets up a system.

7) The DOS 5 memory manager provides no Analysis feature. QEMM's Analysis is extremely useful in determining areas of upper memory that can safely be used. Analysis also lets QEMM users reclaim unused addresses in the system ROM and in other areas in upper memory--a great advantage to memory-hungry users.

8) The DOS 5 memory manager cannot map ROMs into faster RAM. QEMM provides this option which can result in substantially better performance, especially where screen update speed is important.

9) The DOS 5 memory managers have no option to sort memory. On machines where some sections of memory run slower than other sections, QEMM can sort the memory so that the fastest memory is used first.

10) The DOS 5 memory manager cannot manage ShadowRAM or Top Memory, a feature QEMM users on limited-memory systems depend heavily upon. Many 1MB systems turn 384K of the first megabyte of memory into ShadowRAM or Top Memory. This memory is unavailable when using the DOS 5 memory managers.

11) Microsoft Windows 3.x Standard mode won't run under the DOS 5 memory manager when the memory manager is active (in virtual 8086 mode), for example when using a disk cache. QEMM is the only currently shipping memory manager that can run Windows 3.x in all modes whether or not QEMM is active.

12) The DOS 5 memory manager provides no control over the region of High RAM that can be used to load programs high. This means that even expert users may be unable to use High RAM efficiently in situations where TSR's and drivers must be loaded in a specific order. The Quarterdeck LOADHI programs allow TSR's or drivers to be directed to specific high memory locations, giving complete control to the user. Of course, as mentioned before, this feature is used expertly by the OPTIMIZE program in order to provide the optimum configuration.

13) On PS/2s and other microchannel systems, QEMM can automatically detect the addresses used by any adapter listed in our MCA.ADL file. This is especially valuable on systems with adapter RAM (used by many network cards, among other adapters). Adapter RAM can be particularly hard for 386 memory managers to detect. The DOS 5 memory manager has no such feature. As the addresses used by network cards vary from machine to machine and card to card, QEMM's MCA.ADL file can save considerable work for network administrators in companies with large installations of PS/2s or micro channel compatibles on networks. Users of PS/2 machines that are not on a network will also benefit from this "ease of use" feature.

14) DOS 5 has no equivalent for the VIDRAM utility that allows users to extend conventional memory on EGA/VGA systems when running programs that are not using EGA/VGA graphics.

15) The DOS 5 memory manager is less flexible for configuring expanded (EMS) memory. Many DOS programs support EMS memory; others use XMS. To have both types of memory, the DOS memory manager requires you to divide extended memory, part as EMS, part as XMS. To change the amounts available you must edit your CONFIG.SYS file and reboot. QEMM allows EMS and XMS to "share" extended memory. With QEMM, applications can use up to the maximum amount of memory available as EMS, XMS, or a combination of the two without editing configuration files or rebooting the system.

16) QEMM is required for full support of Quarterdeck's DESQview. While DESQview can run with the DOS 5 memory managers, as it can with other EMS drivers, only by using QEMM can you get the special features of DESQview that provide for memory protection and the multi-tasking of "ill-behaved" DOS programs. In addition, QEMM moves more of DESQview's data out of conventional memory, resulting in larger DV windows.

17) QEMM now supports the Suspend and Resume features of some portables and notebook computers that these computers use to minimize battery drain.

Q. How can I install DOS 5 on a system already running QEMM?

A. If you are installing DOS 5 on a system that already has QEMM installed, simply leave the

Quarterdeck memory managers in place and run the DOS SETUP program. SETUP will detect that a compatible memory manager is installed and will not replace it. SETUP typically makes two or three changes to your CONFIG.SYS file. First, it places a "DEVICE=SETVER.EXE" statement at the beginning of your CONFIG.SYS file (before the QEMM device line.) You may want to move this line below QEMM so that Optimize will load it into upper memory. Second, it puts the line "DOS=HIGH" at the end of the CONFIG.SYS. Third, it puts a "SHELL=COMMAND.COM" statement in the CONFIG.SYS if you do not already have one.

Once you have installed the DOS 5 upgrade, switch to your QEMM directory and run the OPTIMIZE program. Since the installation of DOS 5 will change your memory configuration, running Optimize will take care of any rearrangement of programs in upper memory that is necessary, giving you the most conventional memory possible.

If you are running DESQview, you should Optimize with the /STEALTH option, even though Optimize may not suggest it automatically. DESQview can take advantage of the extra memory provided by StealthROM.

Also, in most cases, DESQview users car run larger programs memory in their DOS windows by removing the "DOS=HIGH" from the CONFIG.SYS file. This allows DESQview to use the High Memory Area (HMA) that DOS would otherwise use. DESQview is more efficient at using the HMA than DOS, so giving DESQview the HMA usually results in more memory in your DESQview window. The only way to be sure which is better is to try it both ways (running Optimize and then running DESQview's Memory Status program with each configuration) to determine which provides the most memory.

Q. How do I install QEMM on a DOS 5 system?

A. If you are installing QEMM onto a system that already has DOS 5, simply put the distribution disk in your floppy drive and run the INSTALL program. The INSTALL routines are fully aware of DOS 5 and will automatically run the Optimize program to configure the system optimally. There is no need to do any "pre-configuration" to the system or to remove any of the DOS 5 memory management utilities. If the StealthROM feature is needed to get your resident programs loaded into upper memory, StealthROM will be offered automatically by the Optimize program.

As suggested above, if you are running DESQview you can force the StealthROM feature by running OPTIMIZE /STEALTH, since DESQview can use the extra memory provided by StealthROM.

Q. I run Microsoft Windows in Enhanced Mode. Is there anything I need to know?

A. If you will be running Microsoft Windows in Enhanced mode and plan to use the DOS=HIGH parameter, you cannot use QEMM's "EXT=" or "MEM=" parameters. When these parameters are used, Windows cannot properly take over management of the High Memory Area and will fail to start in Enhanced mode. These parameters do not affect operation in Real or Standard modes, as QEMM remains in control of memory when those modes are used. Further information concerning Windows 3.x and Quarterdeck products can be found in our technotes specific to Windows.

Return to Technotes Main Menu.

QEMM with MS-DOS 6.x

Quarterdeck Technical Note #166

Q. Is MS-DOS 6 compatible with QEMM?

A. Absolutely! The Microsoft DOS 6 README.TXT file states that "Quarterdeck's QEMM memory manager is compatible with MS-DOS 6." In addition, there are no known incompatibilities between MS DOS 6.0 and Quarterdeck's DESQview or DESQview/X multitaskers.

Q. As a QEMM user, what information do I need to install DOS 6?

A. As a QEMM user, the most important information that you can take with you in the installation and configuration of Microsoft DOS 6 is the knowledge that you are already running the most effective memory management system available for the IBM-compatible computer. If you are already using QEMM, follow these steps:

1) Run the SETUP program from the DOS 6 installation diskette and follow the instructions on the screen.

2) Install any of the new DOS 6 utilities that you desire. DOS 6 provides you the opportunity to add virus protection and other utilities to your system. Remember: the default installation of DOS 6 installs only the MS Windows versions of these utilities; you have to tell it to install the DOS versions as well. This is done on the Utilities screen of the SETUP process.

3) Run QEMM's OPTIMIZE to load those drivers into Upper Memory.

If you should desire to experiment with Microsoft DOS 6's MemMaker (the program that attempts to provide more memory), we can recommend a couple of safeguards. MemMaker will remove ALL the QEMM commands from your CONFIG.SYS and AUTOEXEC.BAT files (with the exception of DOSDATA.SYS and DOS-UP.SYS, which you would have to remove manually). We are confident that you will want to return to Quarterdeck's QEMM, so we urge you to save a copy of your CONFIG.SYS and AUTOEXEC.BAT files. Before you begin to experiment, copy these files from the root directory to some other directory or to a floppy diskette. This will allow you to restore them easily.

Our customers have reported problems related to running MemMaker on multi-path CONFIG.SYS files. Unlike QEMM's OPTIMIZE and QSETUP programs, MemMaker does not offer a multi-CONFIG menu.

Next, be aware that even if you save the configuration created by MemMaker, you can usually reverse its changes by running MEMMAKER /UNDO. If the final screen of the MemMaker process is NOT a numerical breakdown of how MemMaker got you more memory (and every test that we have run says it will not be), select the default exit by pressing the <ENTER> key. Do NOT press F3 to save the current MemMaker configuration.

Note that MemMaker does NOT handle CALLed batch files. A CALLed batch file is executed with the DOS keyword "CALL" which tells DOS to execute the commands in the "CALLed" batch file and then resume executing the remaining commands in the current batch file (the AUTOEXEC.BAT in this case). The significance of this is that Quarterdeck's Optimize process DOES handle CALLed batch files, loading any TSR's in those batch files into Upper Memory instead of just ignoring them. This means that when MemMaker deletes the QEMM commands from the AUTOEXEC.BAT and CONFIG.SYS, it leaves them in the Optimized, CALLed batch files. These would have to be removed by editing the CALLed batch files and deleting the LOADHI information. If NO changes have been made since the last time that you Optimized your system, you could also run UNOPT.BAT. UNOPT is a batch file, created by Optimize, that returns your system to the condition it was in before the most recent

Optimize. If you have Optimized more than once, use the OPTIMIZE /RESTORE parameter instead; this allows you to restore any of your last nine configurations, or the pre-QEMM configuration.

With the greater selection of features in QEMM and Optimize, MemMaker does not stand a chance of creating more upper memory. QEMM's StealthROM feature adds 96K of Upper Memory, 64K more than MemMaker's best attempt to find unused space in your System BIOS. Optimize has the ability, through Quarterdeck's Squeeze technology, to shoehorn TSR's and device drivers into areas that are large enough for them to reside, but too small for them to initialize. (It is common for drivers and resident programs to require larger areas during initialization than they need once they have loaded.) Optimize has a "What-If" feature that lets you see the effect that rearranging the loading order of your programs and drivers will have on your memory usage WITHOUT making any changes to your configuration. None of this is possible with MemMaker.

Q. Are any of my Quarterdeck products affected by DoubleSpace (or DriveSpace)?

A. The most talked-about feature of Microsoft DOS 6 is its disk compression -- DriveSpace in DOS 6.22; DoubleSpace in earlier releases of DOS 6. Both DoubleSpace and DriveSpace were designed to be compatible with QEMM. Disk compression utilities, including Stacker, XtraDrive, and now DoubleSpace, have gone to great lengths to be compatible with StealthROM as well as Optimize.

(In the following paragraphs, DBLSPACE is used to refer to DBLSPACE or DRVSPACE.) DBLSPACE.BIN is a driver that allows your system to recognize your DoubleSpace drive. It is loaded by IO.SYS during boot time, BEFORE DOS has even thought about loading QEMM. DBLSPACE.BIN uses about 43K of your memory, and when the CONFIG.SYS has completed, the memory used by the resident portion of DBLSPACE.BIN appears to be added to the memory used by the LAST driver loaded in the CONFIG.SYS. In other words, the last driver loaded appears to be 43K larger than it actually is. When you install DoubleSpace, the following line is added to your CONFIG.SYS file:

DEVICE=C:\DOS\DBLSPACE.SYS /MOVE

DBLSPACE.SYS has only one purpose, and that is to make DBLSPACE.BIN appear as a "real" driver, separate in memory. DoubleSpace REQUIRES that DBLSPACE.SYS be loaded in order for any memory manager to load DBLSPACE.BIN into upper memory.

QEMM 8 includes a feature called "Stealth D*Space," which moves the DoubleSpace driver out of conventional or upper memory and maps it into the expanded memory Page Frame whenever it is needed. By using Stealth DoubleSpace you save approximately 41K of memory. If DoubleSpace is installed on your system when you install QEMM, the following line will be added to your CONFIG.SYS file:

DEVICE=C:\QEMM\ST-DSPC.SYS

If you install DoubleSpace AFTER installing QEMM, you should run QEMM's QSETUP program (by typing QSETUP at the DOS prompt). QSETUP will remove the DBLSPACE.SYS line in your CONFIG.SYS and replace it with the ST-DSPC.SYS line shown above.

The Stealth D*Space feature, like the StealthROM feature, requires the presence of an EMS page frame. If you have used QEMM386.SYS's FRAME=NONE; FRAMELENGTH=0, 1, 2 or 3; or EMS:N parameter to eliminate the page frame, the ST-DSPC.SYS program will act exactly like the DOS 6 driver DBLSPACE.SYS with its /MOVE parameter: in other words, it will move DBLSPACE.BIN from the top of conventional memory to low conventional memory. If you choose to leave ST-DSPC .SYS in your CONFIG.SYS file without a page frame, you can use O ptimize to load the DoubleSpace driver into High RAM, just as you could with DBLSPACE.SYS /MOVE.

Q. How can I restore my QEMM configuration after MemMaker has removed QEMM from my CONFIG.SYS and AUTOEXEC.BAT files?

A. The steps that you must follow in order to return to QEMM after running MemMaker depend on whether you have:

1. Not yet completed MemMaker.

or

2. Have completed MemMaker.

For the Number 1's who have not yet completed MemMaker:

When MemMaker completes and DOES NOT provide a better configuration than the one you already had, it will tell you "Your computer's memory was optimally configured before you ran MemMaker". At this juncture you can press <ENTER> to restore your original configuration or F3 to save the MemMaker configuration. Your choice at this time will be <ENTER>. Your existing QEMM configuration will be restored.

For the Number 2's who have completed MemMaker:

Since you have completed the MemMaker process, it has probably become evident to you that nothing provides more upper memory for loading your TSR's and device drivers than QEMM. Just type QSETUP from the DOS prompt. QEMM's QSETUP program will remove HIMEM.SYS and EMM386.EXE from your CONFIG.SYS file and replace them with a QEMM386.SYS device line. From the QSETUP menu you can also enable other options (such as QDPMI and DOS-UP.) Once QSETUP has enabled the options you choose, run QEMM's Optimize program to load your device drivers and TSR's into upper memory. That is all there is to it.

Q. MemMaker does not support multi-path CONFIG.SYS files. Does QEMM support this DOS 6 feature?

A. Yes -- much better than DOS 6's memory management programs.

DOS 6 provides the ability to build menus of configurations in the CONFIG.SYS and AUTOEXEC.BAT. This is accomplished by building "blocks" in the CONFIG.SYS, and having the name of the block selection that you make on boot passed to the AUTOEXEC.BAT as an environment variable -%config%. The use of the environment variable, for IF's and GOTO's, will then process a particular portion of the AUTOEXEC.BAT file that is appropriate to that portion of the CONFIG.SYS.

Multiple configurations (as implemented via the CONFIG.SYS blocks) have to be MemMakered one configuration at a time. The DOS 6 documentation discusses the process of converting your CONFIG.SYS and AUTOEXEC.BAT into multiple copies and then MemMakering them one at a time. Microsoft warns users to avoid [common] blocks and "first entries" in the AUTOEXEC.BAT.

If you are using QEMM however, you will find the process of Optimizing a multi-config system much easier than with MemMaker. QEMM's Optimize program handles multiple configurations with ease. When you run Optimize, it will detect any multiple configurations you have set up and will post a message prompting you to choose the CONFIG.SYS configuration you want to Optimize. (If Optimize is launched automatically by the Install program or by QEMM Setup, these programs will also detect multiple configurations and pass the information along to the Optimize program.) Optimize will then execute normally, booting the system with the configuration that you have chosen.

When you are not using multiple configurations, Optimize places the /R:n (REGION:n) parameter on lines that load TSRs and device drivers to specify which High RAM region the driver or TSR will load into. When you are using multiple configurations, instead of placing /R:n parameters on the

QEMM386.SYS and LOADHI lines, Optimize will place /RF (/RESPONSEFILE) parameters. The /R:n parameters would not work in a multiple configuration situation, because a program might be part of two or more CONFIG.SYS configurations, each requiring a different region number.

The /RF parameter gets around this problem by directing QEMM386.SYS and the LOADHI programs to look in a resource file called LOADHI.RF that Optimize has created in the \QEMM directory. (Optimize places a LOADHIDATA environment variable in the CONFIG.SYS file that tells QEMM386.SYS and the LOADHI programs the name and location of this resource file.) LOADHI.RF will contain several "config blocks," each corresponding to a CONFIG.SYS configuration and containing the appropriate /R:n statements for that configuration. QEMM386.SYS and the LOADHI programs check the current CONFIG environment variable (created at boot time by DOS 6 to indicate which CONFIG.SYS configuration is being used), and then choose the appropriate config block to get information about what High RAM region they should use to load high.

Here is an example of a resource file:

```
[Vanilla]
QEMM386.SYS /R:1
C:\DOS\SETVER.EXE /R:1
C:\DOS\COMMAND.COM /R:1

[Development]
QEMM386.SYS /R:1
C:\DOS\SETVER.EXE /R:1
C:\QEMM\QDPMI.SYS /R:1
C:\DOS\COMMAND.COM /R:1

[Full]
QEMM386.SYS /R:2
C:\DOS\SETVER.EXE /R:3
C:\QEMM\QDPMI.SYS /R:3
C:\DOS\COMMAND.COM /R:2
C:\PCKWIK\SUPERPCK.EXE /R:2
C:\NET\IPX.COM /R:1
C:\NET\EMSNETX.COM /R:2

[All Others]
QEMM386.SYS /R:1
C:\DOS\SETVER.EXE /R:1
C:\QEMM\QDPMI.SYS /R:1
C:\DOS\COMMAND.COM /R:1
C:\PCKWIK\SUPERPCK.EXE /R:2
```

In this example, the blocks named Vanilla, Development, and Full will be used only when you choose their names off the menu that DOS 6 offers when you boot with multiple configurations. The All Others section will be used only if none of the blocks before it were chosen. If you place a line before the first block, it will be used no matter what configuration name you choose.

This file format is also supported by QEMM's parameter files (see Chapter 7 of the QEMM manual for information) and by the DOS-Up resource file DOS-UP.DAT, which the driver DOS-UP.SYS uses to determine where different parts of DOS go in High RAM. Normally, you do not need to edit these files yourself. Optimize creates and maintain the files.

Q. What else is Quarterdeck doing for DOS?

A. Quarterdeck's commitment to adding functionality to DOS and DOS-based programs has kept our

products at the forefront of memory management and multitasking technology. With each new version of DOS comes an opportunity for Quarterdeck to design new features and offer the DOS user an even greater implementation of the world's most widely used operating system. Quarterdeck intends to continue this pattern.

Return to Technotes Main Menu.

QEMM with Novell DOS and DR-DOS

Quarterdeck Technical Note #269

All shipping versions of Quarterdeck products are essentially compatible with Novell DOS 7 and Digital Research's DR-DOS 6, alternative versions of DOS to Microsoft and IBM DOS offerings. There are, however, a few configuration issues that will be addressed in this note. In the first section, Novell DOS 7 issues are addressed; in the second DR-DOS 6 is discussed, and in the third, notes pertaining to both versions detail QEMM's advantages over the Novell DOS memory managers. We recommend strongly that you read all of the section pertaining to your version of DOS before installing QEMM and running the OPTIMIZE program.

For the purposes of this note, "Novell DOS" will be used to denote either DR-DOS 6 and Novell DOS 7; when there is a distinction between the two, the name of the operating system will be specified in full. Note that Novell DOS 7 is an updated version of DR-DOS 6, and contains significant and worthwhile new features. Some of these features make using QEMM much easier on Novell DOS systems, and while not disparaging DR-DOS 6, Quarterdeck would not discourage DR-DOS 6 users from considering an upgrade to Novell DOS 7.

Part One: Novell DOS 7

Q. Novell DOS 7 promises better memory management than other versions of DOS. Is QEMM useful on Novell DOS 7 systems?

A. While the memory management features of Novell DOS 7 represent an advance for DOS, Quarterdeck's memory managers still provide significant advantages for Novell DOS 7 users. These advantages are detailed at the end of this note.

Q. Novell DOS provides a DPMS driver. What is DPMS?

A. DPMS stands for "DOS Protected Mode Services". Effectively, this is a memory management specification of the same genre as EMS, XMS, VCPI, and DPMI, but is different from all of these. DPMS allows several of the utilities that come with Novell DOS 7 to load into extended memory, reducing conventional memory overhead. DPMS does not conflict with any of the memory management specifications offered by QEMM; in fact, DPMS allocates its memory from QEMM's memory pool.

In the earliest versions of Novell DOS 7, the DPMS driver (DPMS.EXE, version 1.0 and 1.1) would not work properly when loaded high by any memory manager. Updates to Novell DOS 7 are available on CompuServe (GO NETWARE) and from Novell. The initialization size of DPMS.EXE is very large in these updated versions; thus DPMS.EXE can never be loaded high and will work properly in all cases. The size of the driver is only 2K, so this will not have a significant impact on conventional memory, but will greatly reduce the DOS memory overhead of DPMS-aware device drivers and TSRs.

If you have a version of DPMS.EXE version 1.1 or earlier, you should strongly consider an update. If you cannot arrange to acquire an update, place the line

```
DPMS
```

in the OPTIMIZE.NOT file before running OPTIMIZE. Consult your QEMM manual for information on OPTIMIZE.NOT.

Q. How do I install Novell DOS 7 on a machine that is already running QEMM?

A. Novell DOS 7 may have problems installing properly if COMMAND.COM is loaded into upper memory. Before installing Novell DOS 7, please take the following precautionary steps to ensure that QEMM's

DOS-Up feature is configured to load the command processor low:

1) At the DOS prompt, type QSETUP.

2) When the QEMM Setup welcome screen appears, press Enter.

3) At the QEMM Setup Options menu, type U to select Enable or Disable DOS-Up.

4) At the Enable or Disable DOS-Up menu, type P to select Partial.

5) At the DOS Up Options menu, type 2 until the word No appears after COMMAND.COM. Then press Enter.

6) At the QEMM Setup Options screen, press S to select Save Configuration and Quit.

7) You will be prompted to run Optimize. Follow the prompts on the screen.

When Optimize completes, you can run the Novell DOS 7 installation. The installation program will detect that a compatible memory manager (QEMM) is already installed and will not replace it. Follow the procedures outlined for you in the Novell DOS 7 Installation Guide.

Q. How do I install QEMM and run OPTIMIZE on a Novell DOS 7 machine?

A. If you are running the Novell 7 Taskswitcher or Multitasker, shut down all of your programs and exit Task Manager before installing QEMM. Put the QEMM distribution disk in your floppy drive and run the INSTALL program as instructed in the QEMM manual.

Before running OPTIMIZE, ensure that the DPMS driver is in OPTIMIZE.NOT as noted above.

Q. Is the QEMM DOS-Up feature compatible with Novell DOS 7?

A. Yes. The DOS-Up feature has been compatbile with Novell DOS 7 since QEMM 7.04. There are a couple of differences between DOS-UP on MS- or PC-DOS systems and DOS-Up under Novell DOS 7:

- The line DEVICE=C:\QEMM\DOSDATA.SYS appears at the beginning of the CONFIG.SYS file on MS-DOS and IBM DOS systems to prepare the loading of the DOS kernel's data segment into upper memory. On Novell DOS 7 systems, the DOS data segment remains low, so QSETUP does not insert the DOSDATA.SYS line. If DOSDATA.SYS exists in your CONFIG.SYS file, it may be removed. It uses no memory and it will do no harm if you leave it in.

- The SHELL statement in CONFIG.SYS will not contain QEMM's LOADHI command, which is used to load COMMAND.COM into upper memory on MS-DOS and IBM DOS systems only. On Novell DOS 7 systems, DOS-Up does not load COMMAND.COM into upper memory.

Q. Are there any considerations for NWCache?

A. NWCache, Novell DOS 7's disk cache utility, defaults to loading its 16k lookahead buffer into conventional memory. If you need to free more conventional memory for your programs, you may want to use the /BU or /BE parameters on NWCache, which will load the look ahead buffer into either Upper or Expanded memory. If you have a system with a SCSI hard drive, you may need to keep the buffer in conventional memory.

Q. Can I use the INSTALL= and HIINSTALL= commands in CONFIG.SYS?

A. Novell DOS's INSTALL and HIINSTALL commands are incompatible with DOS-Up. If you are using either of these commands in your CONFIG.SYS file to load programs, load the programs from

AUTOEXEC.BAT instead.

Novell DOS 7 can load the command processor into upper memory (High RAM). The command processor's presence in upper memory may cause OPTIMIZE to miscalculate the amount of High RAM available at the end of the boot process. You can prevent such problems by adding the /MH parameter to the command processor line in CONFIG.SYS. For example:

```
SHELL=C:\COMMAND.COM /P /MH
```

The /MH parameter causes the command processor to load into the HMA or conventional memory, depending on other configuration options you may have set, which avoids any potential conflict with OPTIMIZE.

NOTE: If you are using Quarterdeck's DESQview or DESQview/X, please skip the following step which tells you how to free up conventional memory by loading parts of DOS into the HMA. We suggest that DESQview and DESQview/X users not do this because these products can make more efficient use of the HMA than DOS can.)

You can have Novell DOS relocate the DOS kernel into the HMA, freeing space in conventional memory for DOS applications. We recommend that you do this unless you are using DESQview or DESQview/X. To load the parts of DOS into the HMA, add the following line anywhere in your CONFIG.SYS file:

```
DOS=HIGH
```

To get the most free conventional and upper memory, if you use DOS=HIGH to load parts of DOS into the HMA, put the command processor there as well. See step 2 above.

For DESQview and DESQview/X Users Only:
Various Novell DOS 7 utilities, including SHARE and NLSFUNC, will put themselves into the HMA by default, even if the DOS=HIGH statement is not present in CONFIG.SYS. This will prevent DESQview and DESQview/X from using the HMA, which will almost always result in a decrease in the size of DESQview and DESQview/X windows. To prevent SHARE and NLSFUNC from using the HMA, give them one of the following parameters: /MU (which loads the program into upper memory) or /ML (which loads the program into conventional memory). The following example loads SHARE into upper memory:

```
SHARE /MU
```

Consult your Novell DOS 7 manual or the online documentation for further details.

Q. Why does Manifest tell me that I have DOS 6 when I know that I have Novell DOS 7?

A. Programs may request information on the DOS version from the system via the DOS Get Version call. To retain maximum compatibility with MS-DOS 6, Novell DOS 7 answers that its version number is 6.0 whenever a program asks.

Part Two: DR-DOS 6

We recommend strongly that you read ALL of this section of this note before installing QEMM and running the OPTIMIZE program on a DR-DOS 6 system. Pay special attention to the section on SuperStor below if you are using SuperStor disk compression.

Q. How do I get the maximum amount of conventional memory with DR-DOS 6?

A. There are several steps involved, for which a little background information is needed. However, you

can recover as much as 629K of conventional memory using the features of QEMM and DR-DOS.

1) To get the most conventional memory available in DR-DOS 6, use the DOS-Up feature in QEMM in combination with the DR-DOS HIDOS option. To do this, run QSETUP, and from the main menu select U for DOS-Up Options. Choose Y for Yes to enable all of the supported DOS-Up features, and Enter to return to the QSETUP main menu.

2) In MS- and PC-DOS, the DOS=HIGH command in CONFIG.SYS allows the DOS kernel to be loaded into the first 64K of extended memory (the HMA), which eliminates most of DOS's overhead. DR-DOS also permits the kernel to be loaded into the HMA, although in a slightly different way. This is advantageous for all DR-DOS users except those who use DESQview or DESQview/X, which use the the HMA more efficiently than DOS can. Thus if you're a DESQview or DESQview/X user, skip this next step. To load the DR-DOS kernel into the HMA, choose C for "Edit the Proposed CONFIG.SYS" and add the following two lines to CONFIG.SYS AFTER the QEMM386.SYS line:

```
DEVICE=C:\DRDOS\HIDOS.SYS /BDOS=FFFF HIDOS=ON
```

3) Use the DR-DOS HIBUFFERS command, which loads BUFFERS into the HMA, rather than into conventional memory or High RAM. Add this line to CONFIG.SYS:

```
HIBUFFERS=20
```

Since there might not be room in the HMA to load an excessive number of BUFFERS, we recommend limiting BUFFERS to 20.

Q. Why doesn't OPTIMIZE arrange to load the HISTORY or FASTOPEN commands high?

A. LOADHI.SYS will not load HISTORY or FASTOPEN because they are internal instructions to DR-DOS, rather than discrete programs as they are in MS-DOS. This causes no problems and does not increased conventional memory overhead.

Q. What is the HIBUFFERS command?

A. HIBUFFERS is the DR-DOS command to load buffers into the HMA. It will use the HMA even if the DR-DOS kernel is not loaded there. The BUFFERS command of DR-DOS causes the buffers to be loaded into High RAM if HIDOS=ON is in CONFIG.SYS. Each buffer takes an individual UMB. The BUFFERS.COM program from Quarterdeck works with DR-DOS; using the LOADHI.COM program with BUFFERS.COM will cause the buffers to be loaded into a single UMB. Use HIBUFFERS only if you are loading the DR-DOS kernel into the HMA, otherwise the HMA will be used for nothing but buffers.

Q. Should I use the DR-DOS HILOAD, HIDEVICE, and HINSTALL commands in CONFIG.SYS?

A. HILOAD, HIDEVICE, and HINSTALL are the DR-DOS internal commands that load TSRs and device drivers high. These commands are incompatible with QEMM. QEMM's LOADHI.COM, LOADHI.SYS, and INSTALL=LOADHI.COM /TSR which, respectively, perform the same functions, should be used instead.

Q. Can I use the MEMMAX program that comes with DR-DOS?

A. QEMM's VIDRAM program allows the user to extend conventional memory by using the address space normally associated with VGA graphics. DR-DOS comes with a similar program called MEMMAX that works only with the DR-DOS memory managers. Users of any Quarterdeck memory manager or enhancer should use VIDRAM instead of MEMMAX.

Q. Can I use the DR-DOS CHAIN command with QEMM and OPTIMIZE?

A. The DR-DOS CHAIN command allows the CONFIG.SYS to pass control to another CONFIG.SYS-like file. DR-DOS uses it on installation if you choose to install SuperStor. OPTIMIZE does not follow this passing of control to another file. If you are using CHAIN you must combine your configuration files into one for the duration of the OPTIMIZE process.

Q. What about the DR-DOS utilities that load themselves high?

A. DR-DOS utilities such as Super PC-KWIK and DELWATCH are polite enough to allow LOADHI.COM to load them high in the same manner as they would load themselves high; this allows them to be included by OPTIMIZE in its calculations. This is done automatically and does not require any attention from the user.

Q. Can SuperStor be loaded high?

A. SuperStor (SSTORDRV.SYS) is a disk-compression utility that is bundled with DR-DOS. It loads part of itself high, and leaves the rest of itself in conventional memory. The remainder cannot be loaded high with any memory manager. If you use DR-DOS's EMM386.SYS, SuperStor merely loads low without notification; if you use QEMM's LOADHI.SYS, SuperStor does not load at all. To prevent OPTIMIZE from trying to load SSTORDRV.SYS high, place the line

```
SSTORDRV
```

in the OPTIMIZE.NOT file. Consult your QEMM manual for more information on OPTIMIZE.NOT.

Q. Does SuperStor present any complications for QEMM's installation?

A. SuperStor works by creating a large file on the physical hard drive, making that file look like a hard drive, and compressing the data in the file. SuperStor may also swap the drive designations such that the compressed file (which by default would appear to be drive D) appears to be drive C, while the uncompressed portion of the hard drive becomes drive D. This can be convenient, since you will not have to re-write batch files or reconfigure software to run from a different drive. However, the DEVSWAP command poses complications for QEMM's installation and OPTIMIZE processes. However, thanks to the DR-DOS CHAIN command, and to the fact that DR-DOS reads a DCONFIG.SYS file before CONFIG.SYS if the former is present, things can be simplified even if drive swapping is going on. Note that the Stacker software that comes with Novell DOS 7 loads before CONFIG.SYS is processed, and is much easier on the user than the SuperStor approaches presented here.

Method One:

Start with QEMM386.SYS on an uncompressed drive. Leave about one megabyte of space on the uncompressed drive. No other Quarterdeck files but QEMM386.SYS are needed from the QEMM directory.

The first line in DCONFIG.SYS, which is the first CONFIG.SYS file read on bootup, should be the QEMM386.SYS line. QEMM386.SYS is in the QEMM directory of the same drive that DCONFIG.SYS is on. The entire DCONFIG.SYS will normally be

```
DEVICE=C:\QEMM\QEMM386.SYS RAM DEVICE=C:\DRDOS\SSTORDRV.SYS DEVICE=C:
\DRDOS\DEVSWAP.COM CHAIN=C:\CONFIG.SYS
```

Now, move DEVICE=C:\QEMM\QEMM386.SYS to the CONFIG.SYS chained to from DCONFIG.SYS.

MAKE CERTAIN THAT THE QEMM DEVICE LINE IS NO LONGER IN DCONFIG.SYS.

Run OPTIMIZE.

When you're through running OPTIMIZE, move the DEVICE=C:\QEMM\QEMM386.SYS line back to the DCONFIG.SYS file.

That's all there is to it. You don't have to get rid of DEVSWAP or change the drive mappings as you do in Method Two below.

Method Two:

When running OPTIMIZE, DEVSWAP.COM must be remarked out or removed from the CONFIG.SYS. Also all references to drives C: in the AUTOEXEC.BAT and in the CONFIG.SYS after the DEVSWAP.COM line need to be changed to drive D:. Conversely, all references to drive D: should be changed to drive C:.

The next step is to create a QEMM sub-directory on the uncompressed drive. This is typically drive D: when the DEVSWAP.COM device driver is loaded in your CONFIG.SYS. The following files need to be in the QEMM sub-directory in order to run OPTIMIZE properly: DOS-UP.SYS, QEMM386.SYS, OPTIMIZE.COM, OPTIMIZE.EXE, LOADHI.SYS, LOADHI.COM, LOGOPT.COM, BUFFERS.COM (if you are using DOS 2.x or 3.x), RSTRCFG.SYS, MCA.ADL (if you are running on a Microchannel machine), and all .VXD files, if you are planning on running Windows 3.0 in enhanced mode.

Once you have done this, you should reboot before running OPTIMIZE so that the drives are set up correctly. Now you will be able to run OPTIMIZE normally. After OPTIMIZE has run, you may edit your CONFIG.SYS and restore the DEVSWAP.COM line. After you do this, you must edit your CONFIG.SYS and AUTOEXEC.BAT to restore the drive specifications to what they were before; e.g. change all references to drive D: to drive C: and all references to drive C: to drive D:. As mentioned above, if SSTORDRV.SYS is being loaded high, you must change it to load low because it won't work when loaded high. Reboot again. You are now getting the most out of your conventional memory.

Q. Are there any more considerations with DR-DOS 6?

A. Unlike MS- and PC-DOS, DOS hardware interrupt stacks are not provided in DR-DOS 6 -- that is, there is no STACKS command for CONFIG.SYS. There are programs that may malfunction when DOS does not allocate STACKS. As an example, Ventura Publisher 2.0 will allocate its own hardware interrupt stacks when DOS does not do so. When Ventura Publisher uses expanded memory, it puts its stacks in the EMS page frame (a violation of the EMS spec). This comes into conflict with EMS-using software, including QEMM's Stealth feature. The only real resolution is to update your version of DOS to some later version -- Novell DOS 7, or one of the IBM or Microsoft versions.

Q. If I'm using DR-DOS 6, why does Manifest report that I have DOS version 3.31 on the DOS overview screen?

A. DR-DOS 6's API (Application Programming Interface) returns the version number 3.31 when a program using a DOS Get Version call. This is done for compatibility reasons. Manifest makes this call, and thus returns version 3.31 on the DOS Overview screen. DR-DOS loads an environment variable that causes the VER command to report DR-DOS Version 6, but Manifest reports the level of API support. For all programming purposes, DR-DOS 6 is version 3.31. There is no SETVER command, nor is it necessary.

Q. Manifest reports more FILES than I have specified in CONFIG.SYS. Why?

A. DR-DOS converts FCBS to FILE handles so Manifest and the FILES.COM program that comes with QEMM will report the total number of FILE handles to be the sum of the two. The minimum number of FILE handles is 20 and the minimum number of FCBS is 4. Another effect of this conflation of FCBS and FILE handles causes Manifest to report that there is only one FCB.

Part 3: QEMM's Advantages

Q. What are the advantages of QEMM over Novell DOS's memory managers?

A. There are at least ten good answers to this question.

1) QEMM typically provides 64K more High RAM by default than the Novell DOS 7 memory managers, HIMEM.SYS and EMM386.EXE. The Novell DOS 7 products allow similarly includable areas to be included manually, but this requires some expertise, and QEMM is accompanied with better tools for this purpose.

2) QEMM's Squeeze technology allows larger resident programs to be loaded by allowing them to squeeze--temporarily--into the EMS page frame or areas used by Adapters when loading. The Novell DOS 7 memory managers do not provide a similar feature.

3) Using its Stealth option, QEMM can provide EVEN MORE High RAM (211K total is common) by hiding ROMs and allowing High RAM to be mapped over them. Neither Novell DOS 7's memory manager, nor any other third-party memory manager, currently provides this patented technology.

4) Novell DOS 7 has no equivalent to Quarterdeck's Manifest program. Manifest, included free with QEMM, provides extensive information about the computer on which it is running and is an invaluable tool when optimizing a system or diagnosing a memory problem or conflict.

5) The Novell DOS 7 memory manager provides no program to load automatically and optimally TSR's and device drivers into the regions of upper memory. Novice users may experience considerable difficulty achieving good results from the Novell DOS 7 LOADHIGH program, but even advanced users will appreciate the speed and accuracy with which QEMM's OPTIMIZE sets up a system.

6) The Novell DOS 7 memory manager provides no Analysis feature. QEMM's Analysis is extremely useful in determining areas of upper memory that can safely be used. Analysis also lets QEMM users reclaim unused addresses in the system ROM and in other areas in upper memory--a great advantage to memory-hungry users.

7) The Novell DOS 7 memory manager has no equivalent for QEMM's QuickBoot feature.

8) The Novell DOS 7 memory manager provides no control over the region of High RAM that can be used to load programs high. This means that even expert users may be unable to use High RAM efficiently in situations where TSR's and drivers must be loaded in a specific order. The Quarterdeck LOADHI programs allow TSR's or drivers to be directed to specific high memory locations, giving complete control to the user. Of course, as mentioned before, this feature is used expertly by the OPTIMIZE program in order to provide the optimum configuration.

9) On PS/2s and other Micro Channel systems, QEMM can automatically detect the addresses used by any adapter listed in our MCA.ADL file. This is especially valuable on systems with adapter RAM (used by many network cards, among other adapters). Adapter RAM can be particularly hard for 386 memory managers to detect. The Novell DOS 7 memory manager has no such feature. As the addresses used by network cards vary from machine to machine and card to card, QEMM's MCA.ADL file can save considerable work for network administrators in companies with large installations of PS/2s or micro channel compatibles on networks. Users of PS/2 machines that are not on a network will also benefit from this "ease of use" feature.

10) Novell DOS 7's EMM386.EXE is not compatible with Quarterdeck's DESQview multitasking products. QEMM is compatible with the Novell DOS 7's TASKMAX program, although TASKMAX may require Novell's EMM386.EXE for multitasking. Since DESQview allows much more sophisticated multitasking and windowing features than TASKMAX, Quarterdeck recommends using DESQview with QEMM.

Return to Technotes Main Menu.

QEMM's EMS Utility Programs

Quarterdeck Technical Note #293

This technical note describes three advanced EMS utility programs that can give you more control over EMS memory. This document is provided for programmers and advanced users who want to control EMS memory allocation.

EMS.COM and EMS.SYS

The EMS.COM and EMS.SYS programs provide several informative and powerful functions to help you make the best use of your EMS memory in cases in which you have special or unusual requirements. You use EMS.SYS in the CONFIG.SYS file to manipulate expanded memory during the system boot sequence. You use EMS.COM in the AUTOEXEC.BAT file or directly from the DOS prompt, as needed. Although anyone may benefit from seeing the EMS status report and the details of expanded memory allocation, other uses of EMS which will be described in these sections are for technically sophisticated users.

Most of the functions of EMS.SYS and EMS.COM involve the manipulation of expanded memory handles. An EMS handle is the information that the expanded memory manager uses to identify a block of memory that it allocates. A handle is represented by a number and may optionally have a name.

An expanded memory handle is the token of interaction between an EMS-using program and an expanded memory manager. EMS.SYS and EMS.COM give you command-line control of some of the EMS functions that are usually available only at the programming level. Since these EMS utilities are capable of granting you access to handles which may belong to other programs, you should exercise caution when using these utilities.

With the EMS programs, you can allocate and name a block of memory with the CREATE option, and optionally specify that this block of memory consists of the fastest or slowest memory on your system. You can use the FREE option to free the expanded memory associated with a handle. You can read data from a file into expanded memory or write the data from expanded memory to a file with the LOAD or SAVE options. You can rename an EMS handle and change the amount of memory associated with it.

The most common reason for using the EMS programs is to prevent a specific application from using all of the memory in your system. By issuing an EMS CREATE command before running an application, you effectively "hide" the specified amount of memory from that application. Many programs (e.g., Microsoft Windows, AutoCAD, Quattro, Lotus 1-2-3 version 3) allocate a great deal of available memory to themselves at startup timeþsometimes as much as you have on your system. By creating an EMS handle in the following fashion:

EMS CREATE handle_name 2048K

you reserve 2 megabytes of memory, identified by the name HANDLE_NAME, that other programs will see as already assigned, and therefore will not touch. Once your program has started, you could go to the DOS prompt and issue the command:

EMS FREE handle_name

to release the 2 megabytes of memory, which would leave 2 megabytes available after your application is running. Because QEMM gives out both expanded and extended memory from the same memory pool, you can use this method to withhold memory from programs that allocate their memory through EMS, XMS, VCPI, or DPMI. This method is particularly useful for preventing Microsoft Windows 3.1

standard mode from allocating all memory, so that you can run programs that get their memory through EMS, VCPI, or DPMI inside Windows.

If parts of the expanded memory in your system run at different speeds, you can use EMS to allocate memory of one speed before you load a device driver or TSR so that it can only use the faster or slower memory that remains; then you can free the memory for use by your other applications. Manifest can show you if your memory runs at different speeds.

If you are a programmer using expanded memory, you can use the LOAD and SAVE functions when you need to save and restore the contents of expanded memory during development and debugging.

To get a summary report of your expanded memory:

At the DOS prompt, type EMS and press Enter.

EMS will report the total amount of expanded memory, the amount currently available and the address of the page frame.

EMS Parameters

Both EMS.SYS and EMS.COM respond to the same parameters. The parameters are described below. Some parameters have an abbreviation you can use instead of the full name; abbreviations are listed in parentheses below. Brackets ([ ]) in a statement indicate that the enclosed item is optional.

CREATE name amount (CR) allocates expanded memory. CREATE requires two arguments: a name for the block of memory you are allocating and the amount of memory. The name may be one to eight characters long. The name need not be enclosed in quotation marks unless it contains blanks. You can express the amount of memory to allocate several ways: Use a number by itself to express the amount of memory in EMS pages (16K per page). Use a number directly followed by the letter K (e.g., 2048K) to express the amount in kilobytes. If you specify the number of kilobytes, the memory manager will round the number up if necessary to a multiple of 16. You can use the letter M instead of K to express a value in megabytes. You can use the argument ALL to allocate all available memory. You can use the argument ALL-nnnnnn, ALL-nnnnnnK or ALL-nnnM to allocate all available memory minus a specified number of EMS pages, kilobytes or megabytes.

Follow the EMS CREATE command with the EMS DIR command to confirm the allocation and to determine the handle number assigned to the name.

CREATEFAST name amount (CFAST) and CREATESLOW name amount (CSLOW) are alternate forms of the CREATE option (see above) that instruct the memory manager to allocate the memory from either faster or slower memory. Use Manifest's Expanded Memory Timings to determine if any speed difference does in fact exist.

DIR displays a breakdown of the current expanded memory allocated. For each allocated handle, DIR gives the number of expanded memory pages associated with it, the number of kilobytes of memory those pages represent, and the name assigned to that handle, if any.

FREE name or number frees memory and deallocates a handle. FREE requires that you specify a handle to deallocate, either by its name or number. Beware of doing this to someone else's handle.

HELP displays help on the EMS programs and their options.

LOAD name or number filename allows you to restore the contents of expanded memory pages that have been stored in a file. This option requires that you specify the handle name (or number) and the name of the file containing the data you want to restore. The number of pages required will be

automatically allocated based on the file's size.

RENAME name or number new_name (REN) lets you assign a new name to a handle. The first parameter to RENAME is the original handle. You may refer to this handle by its number or its name. The second argument is the new handle name. RENAME can be useful to name an unnamed handle to help you keep track of it.

RESIZE name or number amount (RES) lets you increase or decrease the amount of memory assigned to a handle. Its two arguments are the same as those of CREATE (see above).

SAVE name or number filename allows you to save the contents of the expanded memory pages associated with an EMS handle to a file. This option requires that you specify the handle name (or number) and the filename.

? lists the EMS programs' parameters.

EMS2EXT.SYS

EMS2EXT.SYS converts expanded memory to extended memory, for programs that rely upon the old INT 15 method of accessing extended memory. This method is no longer widely used, and has been replaced by XMS (the Extended Memory Specification). Older versions of DOS shipped with utilities which relied upon the old INT 15 interface, most notably VDISK.SYS. These drivers have since been replaced by programs that use XMS instead, and as a result EMS2EXT is rarely useful.

EMS2EXT is not needed for programs that access memory through XMS, VCPI, or DPMI. It is intended only to provide on-the-fly control over extended memory allocated through the older INT 15 interface. Programs which support XMS, VCPI or DPMI can allocate and deallocate memory directly from QEMM's memory pool and have no need for EMS2EXT.

Even if you have an old extended memory utility, you cannot use EMS2EXT if your program expects to access extended memory directly at physical addresses above 1024K. Quarterdeck's QEXT.SYS driver, supplied with DESQview, cannot use memory supplied by EMS2EXT. Likewise, Microsoft's HIMEM.SYS cannot use memory supplied by EMS2EXT.

If you do have an old extended memory program that uses the INT 15 interface, EMS2EXT lets you allocate memory for that program out of QEMM's memory pool. The advantage of allocating this memory with EMS2EXT instead of with QEMM parameters is that the memory allocation can later be increased or decreased with the EMS.COM program without rebooting your system.

EMS2EXT is a device driver and therefore needs to be loaded with a DEVICE= statement in your CONFIG.SYS file. The statement to load EMS2EXT should look like this:

DEVICE=C:\QEMM\EMS2EXT.SYS MEMORY=nnn speed

The nnn parameter in MEMORY=nnn is the number of kilobytes of expanded memory to allocate initially (e.g., MEMORY=512). EMS2EXT will allocate an EMS handle named EMS2EXT for a block of memory nnnK in size. You can also load EMS2EXT without specifying any MEMORY parameter. EMS2EXT will be resident, but it will not allocate any memory. It will, however, reserve for itself a handle with the name EMS2EXT.

The optional SPEED parameter tells EMS2EXT to allocate faster or slower memory if there are different speeds of memory on your system. You may specify FAST, SLOW or no SPEED option at all. You can, as needed, grow, or shrink the amount of extended memory for the EMS2EXT handle using EMS.COM. You can use this capability to give a program INT 15 extended memory only while it is running. For instance, if you loaded EMS2EXT with no MEMORY parameter, you could make a batch

file which included the line:

EMS RESIZE EMS2EXT 128K

before running an application that needs 128K of extended memory through the old INT 15 interface. When the program terminates, another EMS statement could free the memory:

EMS RESIZE EMS2EXT 0

The memory is then returned to QEMM's memory pool for the use of other programs.

Return to Technotes Main Menu.

QEMM's Manifest Program

Manifest is a powerful system reporting program that is automatically installed on your hard drive when you install QEMM. While Manifest can be extremely useful when troubleshooting a problem, it is much more than a diagnostic tool. Whether you are new to computers or a power user with years of DOS experience, Manifest will help you take full advantage of the memory that is installed in your computer.

To run Manifest type **MFT** at the DOS promp and then, at your own pace, explore the world of memory as it relates to your own PC. From within Manifest, press **F1** at any time for **context-sensitive online help**.

Return to Hints Main Menu.

QEMM's New Parameter Names

Many QEMM parameters have new names. If you are upgrading from an earlier version, you can still use the old parameter names if you like. Below is a list of the old parameter names, cross-referenced with the new names. Abbreviations are listed in parentheses.

Old Name	New Name
COMPAQ386S (C386S)	COMPAQ386S:Y (C386S)
COMPAQEGAROM (CER)	COMPAQEGAROM:Y (CER)
COMPAQHALFROM (CHR)	COMPAQHALFROM:Y (CHR)
COMPAQROMMEMORY (CRM)	COMPAQROMMEMORY:Y (CRM)
DONTUSEXMS (DUX)	USEXMS:N
DOS4 (D4)	DOS4:Y (D4)
FORCEEMS (FEMS)	FORCEEMS:Y (FEMS)
FORCESTEALTHCOPY (FSTC)	FORCESTEALTHCOPY:Y (FSTC)
IGNOREA20 (IA)	TRAP8042:Y (T8) **
LOCKDMA (LD)	LOCKDMA:Y (LD)
NOCOMPAQFEATURES (NCF)	COMPAQFEATURES:N (CF)
NOEMS	EMS:N
NOFILL (NO)	FILL:N
NOHMA	HMA:N
NOPAUSEONERROR (NOPE)	PAUSEONERROR:Y (PE)
NOROM (NR)	MAPREBOOT:N (MR)
NOROMHOLES (NRH)	ROMHOLES:N (RH)
NOSHADOWRAM (NOSH)	SHADOWRAM:NONE (SH)
NOTOKENRING (NTR)	TOKENRING:N (TR)
NOTOPMEMORY (NT)	TOPMEMORY:N (TM)
NOVDS	VDS:N
NOVIDEOFILL (NV)	VIDEOFILL:N (VF)
NOVIDEORAM (NVR)	VIDEORAM:N (VR)
NOWINDOWS3 (NW3)	WINDOWS3:N (W3)
NOXBDA (NX)	XBDA:N
NOXMS	XMS:N
UOLDDV (ODV)	OLDDV:Y (ODV)
UNUSUALEXT (UX)	UNUSUALEXT:Y (UX)

**** Default has changed.**

Return to Hints Main Menu.

QEMM's Optimize Program

Optimize is a program that determines how to load TSRs, device drivers and selected parts of DOS into upper memory. Optimize analyzes the memory requirements of device drivers and TSRs that you are loading from **CONFIG.SYS** and **AUTOEXEC.BAT** and any batch files called by AUTOEXEC.BAT. Then, Optimize determines the most efficient way to load items into High RAM by testing all possible locations. There may be millions of possibilities.

The object is to free up as much conventional memory as possible for your DOS programs. If you are using QEMM's **DOS-Up** feature, Optimize also experiments with different ways of loading parts of DOS into upper memory

When you install QEMM, **INSTALL** offers to run Optimize. You should run Optimize again if you add new hardware devices or modify your AUTOEXEC.BAT or CONFIG.SYS files.

Optimize normally reboots your machine two or three times as it analyzes your configuration. If your system is particularly complex, however, you may be asked to reboot your machine several more times. These additional reboots are necessary to detect non-standard hardware and software and result in greatly improved compatibility between such systems and QEMM.

Optimize changes the lines that load device drivers and TSRs by adding a LOADHI command to the beginning of those lines. For example, a device driver line that looks like this:

DEVICE=C:\DOS\SETVER.EXE

would look something like this when Optimize completes:

DEVICE=C:\QEMM\LOADHI.SYS /R:1 C:\DOS\SETVER.EXE

A TSR line in AUTOEXEC.BAT that looks like this:

C:\DOS\SMARTDRV.EXE

would look something like this after running Optimize:

C:\QEMM\LOADHI /R:2 C:\DOS\SMARTDRV.EXE

For detailed information on Optimize, see Chapter 3 of the QEMM Reference Manual. For information on the LOADHI command, see Chapter 8.

Return to Hints Main Menu.

QEMM's StealthROM Feature

StealthROM is an exclusive QEMM feature that can typically create an additional 48K to 115K of High RAM on almost any PC. StealthROM hides your PC's ROMs and makes their memory addresses available for High RAM or expanded memory mapping. The advantage of having the additional High RAM is that QEMM can load TSRs, device drivers and selected parts of DOS there instead of in conventional memory. By freeing up conventional memory, you will have more room for running DOS programs.

Depending on your configuration and the installation options you chose, StealthROM may have been enabled on your system when you installed QEMM. When you run the **Optimize** program, Optimize will try to load your TSRs, device drivers and selected parts of DOS into High RAM. If all of them will not fit, Optimize will test your system for compatibility with StealthROM and will determine which StealthROM method is best for your system.

For more information on StealthROM, consult the technote "An Overview of QEMM's StealthROM Technology" (**STLTECH.TEC**). In the unlikely event that you are having trouble that you think is related to Stealth, consult "General Troubleshooting" (**TROUBLE.TEC**) or "Troubleshooting StealthROM" (**STEALTH.TEC**).

Return to Hints Main Menu.

QEMM's StealthROM Technology: An Overview

Quarterdeck Technical Note #168

Q: What is StealthROM?
Q: How does StealthROM work?
Q: What is the difference between ST:M and ST:F?
Q: Which StealthROM strategy is preferable?
Q: Does StealthROM slow down my system?
Q: How can StealthROM fail?
Q: If I'm having problems with StealthROM, what should I do?

Note: for the purposes of this note, "386" refers to any processor in the 80386 family -- the Intel 80386 SX and DX; the i486 SX and DX in all their flavors; the Pentium processor, and all processors compatible with these chips.

Traditionally, 386 memory managers such as QEMM have been able to create extra memory for DOS by associating physical extended memory (memory above the 1MB line, which is outside of DOS' address space) with unused addresses between 640K and 1MB. This extra memory is called High RAM. Quarterdeck's StealthROM technology (which is included with QEMM versions 6.00 and higher) is QEMM's method of creating more High RAM than previously thought possible, by mapping memory to addresses that are used by system, video, disk, and other ROMs.

Q. How does StealthROM work?

To understand how StealthROM works, it is useful to understand the concept of MAPPING. When a program needs more memory than what is normally available to it under DOS, it can request that some expanded memory be allocated from either an EMS board, or from the EMS memory created by a 386 expanded memory manager. MAPPING is the process by which memory management hardware and software can make memory appear in appropriate places at appropriate times; it is the process of associating memory with an address other than its actual one. A convenient place to make memory appear is a 64K window of addresses above the 640K line; this window is called the EMS page frame. The expanded memory specification (EMS) uses mapping to make portions of expanded memory appear inside the EMS page frame when that memory is requested by a program.

Expanded memory has no addresses of its own, but can be made to appear at a valid address -- "mapped in". Expanded memory pages can be filled with code or data by a program; when that code or data is not needed, the pages can may be "mapped out" -- relieved of their addresses and put back into the expanded memory pool, with the code and data still intact. When the application needs these pages, they are "mapped in" to the EMS page frame again. It is therefore possible for a program that uses expanded memory to have access to much more memory than DOS itself can see of its own accord. You may know this technology as "bank switching," which is one of the techniques used to extend and add power to everything from mainframe computers to high-end UNIX systems... to DOS machines!

Mapping is also useful for creating High RAM; in addition the the page frame, memory can be associated with other unused addresses between 640K and 1MB. The 386 hardware and QEMM cooperate to make memory appear where there is otherwise none.

StealthROM uses mapping for a new purpose. The 386 chip can be made to map memory in or out of DOS' address space at any time. StealthROM uses 386 mapping to map system, disk, or video ROMs in and out of DOS' address space when appropriate, using one of two strategies -- Mapping mode or Frame mode. These two features are activated by parameters on the QEMM line -- ST:M, for Stealth Mapping, or ST:F, for Stealth Frame.

Q. What is the difference between ST:M and ST:F?

"BIOS" stands for "Basic Input Output Services", programs that are built right into the hardware of your system in a form called "Read-Only Memory". Your system communicates with various parts of itself and with its peripherals via the ROM-BIOS, often referred to as "ROMs". The ROMs on your system are accessed via interrupts -- which are conceptually similar to BASIC subroutines. When your system boots up, it sets up something called an interrupt vector table. This is a list of addresses where specific ROM subroutines can be found. When a program on your system needs a certain ROM function (for example, writing colored text to the screen), it sets up some data in appropriate places, and then calls the interrupt with a processor INT instruction. The processor then looks at the interrupt vector table to find out the address where the ROM function can be found. The processor transfers control to that address, the ROM subroutine gets run, and then control is returned to the calling program.

When you use StealthROM, as your system boots QEMM takes control of interrupts that are in use by the ROMs on your system and points those interrupts into QEMM itself. This way, QEMM can monitor exactly when a ROM interrupt occurs, and can manage the interrupt appropriately.

When you use ST:M ("Mapping Method"), QEMM maps system, video, and disk ROMs and any other "Stealthable" ROMs out of the first megabyte. (For information on what is "Stealthable," see "How can StealthROM fail?" below.) When the ROM is needed by the system, QEMM maps the appropriate ROM code into the expanded memory page frame. The ROM code now has a valid address at which it can execute, and it does so normally. When the ROM routine is finished, QEMM then remaps the ROM elsewhere out of the address space.

When you use ST:F ("Frame Method"), QEMM leaves the system, video, and disk ROMs where they are normally found. QEMM then places the EMS page frame at the same address as -- or "on top of" -- a ROM. Expanded memory can then be mapped into the EMS page frame. When the ROM that has been hidden by the page frame is needed, QEMM maps the page frame away, and maps ROM back into the addresses that were occupied by the page frame. The ROM code then executes normally. When the ROM routine is finished, QEMM can then restore the contents of the page frame, and the ROM is effectively hidden again.

Q. Which StealthROM strategy is preferable?

Since ST:M is capable of mapping almost all ROMs out of DOS' address space, and thus provides much more High RAM, it is the better of the two options. ST:F should only be needed on a very small number of systems; its object is to ensure compatibility with machines that have ROMs that jump to each other without using an interrupt to do so, or with ROMs that need to execute at their original addresses.

Q. I have to have a special version of QEMM so that StealthROM will work on my system, right? My system has to be one that StealthROM knows about, right? I have to disable some of QEMM's memory management features to take advantage of StealthROM, right?

No to all three questions! StealthROM is designed to work on ANY system, regardless of brand, model, or ROM BIOS revision. You do not need a special version of QEMM or StealthROM that has been customized for your machine, because StealthROM's strategy merely relocates your ROMs instead of replacing them. StealthROM does not modify, compress or replace your ROM BIOS, and it does not depend on being aware of the brand or revision of your ROMs. Additionally, StealthROM will typically create more High RAM on your system than any other memory management technique. You do not have to disable any of QEMM's features -- EMS, XMS, DPMI, or VCPI memory management. Other memory managers force you to sacrifice features or compatibility as they try to match QEMM's prowess in squeezing every last byte of High RAM from your system.

Q. Does StealthROM slow my system down?

StealthROM does add some tiny amount of overhead to ROM BIOS interrupts. Since most application

programs spend very little time calling ROM code, the slowdowns are usually imperceptible or insignificant to the user. Ironically, since benchmark programs often call ROM interrupts repeatedly (some do almost nothing but this), the greatest slowdown will be seen in some benchmark results; these results rarely have much to do with the actual speed of useful programs, however. Since your application programs typically have much more conventional memory to deal with when StealthROM is invoked, you are more likely to observe faster -- not slower -- performance. Furthermore, QEMM optimizes some ROM video functions with its own faster techniques when StealthROM is active, and QEMM's ROM parameter (see the QEMM documentation) can provide additional performance increases. Using StealthROM with the ROM parameter is typically significantly faster than not using QEMM at all.

Q. How can StealthROM fail?

StealthROM is a robust and proven technology. However, it is possible for programs or system ROM implementations to interfere with StealthROM's strategies. Note that the problems described here are infrequent and/or system-specific, and that most users will experience no difficulty at all with StealthROM.

In the above description of how StealthROM works, each strategy depends on a processor interrupt being referenced. This is the normal way of accessing ROM code; processor registers are loaded with data and with information which denotes exactly which ROM service is being requested, and then a processor INT instruction is called. BASIC programmers will recognize that this is similar to the process of initializing a few variables with data, and then calling a subroutine with a GOSUB instruction; most good texts favor this method of programming. However, it is possible (though relatively uncommon) for a piece of code to JUMP to a specific ROM address, without branching via an interrupt. This is analogous to a BASIC GOTO, rather than a GOSUB; dependencies on GOTOs are generally frowned upon by expert programmers, since a GOTO presumes that the address to which the code is jumping will remain constant and unchanging. This is less of a problem if one person writes all the code, since it is easier for one person to keep track of the proper destination addresses; when more than one person is involved, it's more difficult to determine why and where the code should branch.

A few addresses in the interrupt vector table are used to point to tables of BIOS data, rather than to executable code. StealthROM is designed to account for these sorts of addreses as well; as with program code, QEMM points the processor to appropriate data if an address in the interrupt vector table points to system configuration information, rather than to BIOS program routines.

If an application or utility jumps directly to a ROM address when StealthROM is invoked, QEMM will not be able to intercept an interrupt, and thus may not have a chance to make sure that the appropriate portion of the ROM code is mapped into the page frame. If QEMM's Optimize program detects this behavior, it can make the application work properly with StealthROM by applying the STEALTHTHUNK parameter, sacrificing a small amount of High RAM (usually 4K) in order to intercept the direct jump, map the appropriate ROM into the page frame, and divert the direct jump to the proper address. If the behavior does not occur during the Optimize process, it will probably be necessary to EXCLUDE a portion of the ROM on the QEMM386.SYS line in the CONFIG.SYS. More information on STEALTTHUNK parameter can be found in the QEMM Reference Manual (or in the README file in some releases).

In the case of system setup programs and installation routines for video cards (many of which access ROM addresses directly), it is far better to disable QEMM temporarily than to use EXCLUDEs or sacrifice the large amounts of extra High RAM that ST:M can provide. Setup programs should need to be run infrequently, and typically require a reboot before the modified settings take effect. High RAM is generally much more useful. It is worth weighing the benefits of instant access to your setup program against the extra High RAM that StealthROM can provide; the decision should not be a difficult one.

The easiest way to deal with this is to disable QEMM, run your Setup program, and reboot with QEMM active again. To disable QEMM temporarily, hold down the <Alt> key immediately after you hear a beep

on bootup. QEMM will post a message telling you to press <Escape> to unload QEMM, or any other key to continue with QEMM. Press <Escape>, and run your Setup program. (If you are using QEMM's DOS-UP feature, you will first see a message asking if you want to unload DOSDATA. Press <Escape> to unload DOSDATA, then hold down the <Alt> key again until you see the message telling you to press <Escape> to unload QEMM. After unloading QEMM, run your Setup program, then reboot the machine normally (without holding down <Alt>); your revised Setup will be in effect, and so will QEMM.

If you are using DOS 6.0, you can also boot without loading either your CONFIG.SYS or AUTOEXEC.BAT file by pressing F5 before the CONFIG.SYS is processed. Also, if you press F8 before the CONFIG.SYS is processed, you will be given the option of processing your configuration files on a line-by-line basis. Choose "No" when asked if you want to load DOSDATA, QEMM386, DOS-UP, and QDPMI. If you choose to step through your configuration files line-by-line, you will see error messages stating that some drivers and TSRs cannot load high and will be loaded low, and suggesting that you re-run Optimize. These error messages are normal when booting without QEMM.

* Some ROMs are written in such a way that they jump internally to addresses that are "hard-wired" into the ROM code. For instance, a ROM that lives at address C000 may jump within itself using a full address like C000:AD91, where a jump to offset AD91 would have had the same effect. Jumps to explicit addresses can confound StealthROM, as the ROM does not always execute at its original address. The best way of getting around this problem is often to use ST:M, and to place the EMS page frame to be placed on top of the ROM in question; this means that the ROM will execute at its original location without any sacrifice of High RAM. If you cannot put the page frame over the offending ROM, the ST:F option is another method of guaranteeing that all ROMs will execute at their original addresses.

* Sometimes one ROM will jump directly into another ROM's code instead of accessing the other ROM through interrupts. In such circumstances, ST:F may be helpful, since the ROMs will all execute at their original addresses, making both inter-ROM and intra-ROM jumps safe.

* Some programs find the address of a given piece of ROM at startup, and then jump directly to that address later on, at a time when the ROM may not be mapped into memory. Programs like these will often require that a portion of the ROM be EXCLUDEd on the QEMM386.SYS line in CONFIG.SYS. Quarterdeck Technical Note #205, Troubleshooting StealthROM (**STEALTH.TEC**) can assist in finding the appropriate EXCLUDE quickly.

* Some ROMs do not have any interrupts pointing to them at startup. If this is the case, QEMM will not be able to detect where a given interrupt should point, and thus may not invoke StealthROM for that ROM. Again, Quarterdeck Technical Note #205, Troubleshooting StealthROM (STEALTH.TEC) may help to determine which addresses within this ROM must be EXCLUDEd (for compatiblity) or can be INCLUDEd (for more High RAM).

* Some device drivers refuse to load unless they see an interrupt pointing to its normal location. Quarterdeck Technical Note #233, "QEMM and the XSTI parameter" (**XSTI.TEC**) explains another way to resolve this problem which usually results in more conventional memory saved than if the driver is loaded before QEMM. The DEVICE= lines that refer to these programs may also be loaded before the QEMM386.SYS line in CONFIG.SYS (though after DOSDATA.SYS) if necessary.

* Some programs make invalid assumptions about the EMS page frame. In some cases, programs assume that the state of the EMS page frame will remain unchanged even after they decide to release their claim to a page of expanded memory; this is akin to assuming that you can get your property back after leaving it at the end of the driveway on garbage pick-up day. This fails with Stealth ROM because, by default, the page frame is immediately un-mapped after a handle has been abandoned -- as if, in the above example, the city picks up the garbage pretty much immediately -- as soon as you get back into your house. The UFP:N parameter suppresses this feature and can make such careless programs work with StealthROM, perhaps at the expense of some speed.

* Some applications assume that the contents of the page frame will be the same at hardware interrupt

time as they are when the main body of the application is executing -- like assuming that your coat will never get moved from the place in which you saw the cloakroom attendant put it. This is an invalid assumption, and can cause problems not only with StealthROM, but with EMS-using TSRs as well. This ignores the guidelines in the Expanded Memory Specification, which governs the proper use of expanded there.

* Other programs outright violate the Expanded Memory Specification by placing their interrupt stacks -- effectively the program's means of keeping track of its current state -- in the page frame. This is not simply a problem for StealthROM or for QEMM; this can cause a conflict with any using expanded memory and ANY expanded memory manager.

Fortunately, the programs that exhibit these problems are rare. If you experience difficulty that is found to be Stealth-related, you might wish to encourage the developer of the faulting program to make the program more compatible with StealthROM. Quarterdeck is very happy to assist the developer of any commercial hardware or software who wishes added compatibility with our products.

Q. What does the ROM parameter have to do with StealthROM?

ROM code is normally read 8 or 16 bits at a time, and 32-bit RAM is therefore much faster. (You can see this in action by looking at Manifest First Meg / Timings, first without the ROM parameter on the QEMM386.SYS line in CONFIG.SYS, and then with ROM added to the end of that line.) Some video ROM speed-up drivers work by copying the contents of video ROM to conventional RAM. These programs (such as TVGABIO.SYS, RAMBIOS.SYS, FASTBIOS.SYS, and SPEED_UP.SYS, typically shipped on the utilities diskette provided with your video card) will often conflict with StealthROM. If loaded after QEMM, such programs may refuse to load because they detect that a program loaded before them (QEMM) is intercepting the video interrupt, INT 10. Conversely, if loaded before QEMM, these programs may divert interrupts into RAM, so that QEMM cannot locate the ROM handler for those interrupts. In these cases, the video speed-up program will function properly, but StealthROM will be disabled. XSTI.TEC explains how to resolve this problem if you really want to load the video enhancement program. However, QEMM's ROM parameter generally provides the same feature these drivers do, with three important advantages. First, QEMM copies the video ROM into 32-bit RAM and then write-protects the RAM so that some errant program does not overwrite the ROM code. Second, QEMM's ROM parameter costs neither conventional memory nor High RAM to provide this feature -- the video drivers mentioned above will typically take 32K of one or the other. Finally, the ROM parameter is fully compatible with StealthROM.

Q. If I'm having problems with StealthROM, what should I do?

StealthROM problems can be resolved by consulting Quarterdeck Technical Note #205, "Troubleshooting StealthROM" (**STEALTH.TEC**).

SUMMARY

StealthROM is a robust and proven technology. It is an easy-to-use, safe, and efficient way of creating more High RAM on your system, providing more memory for your TSRs, your device drivers, DESQview 386, MS Windows, and your application programs. It is likely to speed up your system rather than slowing it down. It is designed to be effective on any 386 or higher processor, regardless of the ROM's manufacturer or version. Many programs that cause conflicts with StealthROM can cause problems with other programs and memory managers. Stealth conflicts are rare, and troubleshooting is straightforward. StealthROM is the easiest way to provide the optimal amount of High RAM on your system.

Return to Technotes Main Menu.

QEMM's XSTI StealthROM Parameter

Quarterdeck Technical Note #233

Q. Why do I see this message when I start my computer?

> **QEMM386: Disabling Stealth because QEMM could not locate the ROM handler for INT XX"**

A. There are three possible causes:

1) You are loading a driver before QEMM which is grabbing interrupt XX; OR

2) A ROM is loading a handler for interrupt XX into RAM.

3) You are using a computer which was upgraded to an 80386 with an add-in board, such as the Intel "Inboard PC."

There are several potential solutions:

1) Load the driver in question after QEMM. If it must be loaded before QEMM, load HOOKROM.SYS before you load this driver.

During installation of QEMM, HOOKROM.SYS is installed in the QEMM directory. Assuming that QEMM is installed in a directory called QEMM on your "C" drive, the new line would look like this:

DEVICE=C:\QEMM\HOOKROM.SYS

HOOKROM is a device driver that may be needed if you use the StealthROM feature and are loading one of your device drivers before QEMM386.SYS in the CONFIG.SYS file. Though it is usually best to load device drivers after QEMM386.SYS, there are some special drivers (like the ones that manage some 80386 conversion hardware) that must load before QEMM386.SYS. These drivers can obscure information that QEMM needs to enable the StealthROM feature, in which case QEMM386.SYS will post the above error message.

Placed before QEMM386.SYS in the CONFIG.SYS, HOOKROM will gather the necessary information for QEMM386.SYS and prevent this special driver from interfering with the StealthROM process.

2) Add the parameter "XSTI=XX" (where "XX" is the number of the interrupt reported in the message) to the QEMM386.SYS line of the CONFIG.SYS, then add the appropriate eXclude to this same line in order to keep QEMM from mapping over the portion of the address space where the ROM handler for interrupt XX resides. (See "HOW DO I FIND THE APPROPRIATE EXCLUDE?" below.)

It may also be possible to reconfigure your system in such a way that the ROM no longer redirects an interrupt into RAM. This is the case with the Invisible Network. (See "KNOWN USES FOR XSTI" near the end of this technical bulletin.) It is also possible that a program you are trying to run, or even your operating system, wants to have a particular interrupt remain unStealthed. XSTI, with the appropriate eXclude, is necessary to get your program, or operating system, working with StealthROM.

3) Add the following parameters to the QEMM device line in your CONFIG.SYS file:

XSTI=70 XSTI=74 XSTI=75 XSTI=76

A typical QEMM line would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=70 XSTI=74 XSTI=75 ... XSTI=76

(Note that the preceding two lines should be on a single line in CONFIG.SYS.)

Q. How do I find the "appropriate exclude"?

A. First, note that QEMM's Stealth Testing will find automatically the majority of circumstances that will require XSTI, and will make the appropriate exclusions or S-pages. If the conflict you experience does not happen as part of the boot process. You find the appropriate eXclude by excluding all the address space occupied by ROMs, using the parameter FSTC, and doing an Analysis. First, locate all your ROMs. You can do this by looking at the First Meg/Overview screen of Manifest. Those with non-Micro Channel machines and VGA video typically have a system ROM at F000-FFFF and a video ROM at C000-C7FF. Those with PS/2s or other Micro Channel machines typically have one ROM at E000-FFFF. Add-on devices, such as some disk controller cards and network cards, may also have ROMs, which you must eXclude as well.

A typical QEMM line for a non-Micro Channel machine is:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=XX X=F000-FFFF ... X=C000-C7FF FSTC

(again, all on one line).

On a PS/2 or most Micro Channel machines, the line will look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=XX X=E000-FFFF FSTC

In the above examples, XX is replaced with the interrupt reported in the QEMM error message.

Reboot your computer with this CONFIG.SYS. StealthROM should work this time. Use your computer for a while, then look at the QEMM/Analysis screen of Manifest. You will see a chart that looks something like this:

Consulting the ANALYSIS section of your Manifest or QEMM manual, you will read that an "I" indicates a portion of the address space that HAS NOT been accessed and an "O" indicates a portion of the address space that HAS been accessed. You must eXclude that portion of the address space in the

eXcluded ROMs where you now see "O"s.

In this example (which presumes that the ROMs were located from C000-C7FF and F000-FFFF), the appropriate eXclude is "X=F800-F9FF", an 8K portion of the address space. This is the portion of the address space where the ROM handler for the interrupt XX resides. Our QEMM line, with appropriate excludes, would read as follows:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=XX X=F800-F9FF

PLEASE NOTE: The FSTC parameter is used only during this analysis process and should be removed afterward. Because the last 64 bytes of the First Meg address space (in FFFC-FFFF) is still addressed directly with StealthROM, the last 4K piece of the QEMM/Analysis screen will always have an "O" in it, whether an eXclude is appropriate or not.

ALSO NOTE: This procedure IS NOT used to find INCLUDES in portions of the address space NOT occupied by Stealthed ROMs. If you wish to experiment with INCLUDES (in order to gain additional High RAM) you must perform a complete analysis as described in the ANALYSIS section of the QEMM or Manifest manual.

Q. What if there are no "O"s?

A. It is possible that there are no "O"s at all: this is because the ROM handler for interrupt XX has been replaced by a new interrupt handler and the one in the ROM is not being accessed at all. No eXclude is necessary in this case.

Q. What are the known uses for XSTI?

A. There are several known instances of a need for XSTI. In many cases, these parameters will be found automatically by QEMM

INVISIBLE NETWORK
If you use the boot ROM on the Invisible Network cards, it loads 32K of code into the top of the conventional memory address space, and grabs interrupt 13. A much better solution than to use XSTI=13 and the appropriate eXclude is to disable the ROM on the network card and load IS2BIOS instead. This will give you 32K more conventional memory (since IS2BIOS can be loaded high), and you will not have the network card's ROM breaking up your high address space.

MS-DOS 5 ON SOME ZENITH MACHINES
XSTI=18 and the appropriate eXclude is necessary to print on some Zenith machines. This is due to an obscure method used only in some Zenith BIOSes. A Zenith version of DOS 5 may not have this problem.

WORDSTAR 2000 version 1.01
XSTI=15 and the appropriate eXclude is necessary. This is due to an ancient method of jumping directly to the code that an interrupt vector points to. This version of Wordstar 2000 was written in 1985. Newer versions may not have this problem.

VIDEO ACCELERATOR DRIVERS
SPEED_UP.SYS is a driver that comes with the Orchid Prodesigner video card. It makes a copy of the video ROM in RAM in order to speed up your video. If it is loaded after QEMM on a system with StealthROM enabled, it refuses to load, complaining that someone else has taken Interrupt 10. If loaded before QEMM on the same system, StealthROM will be disabled because QEMM cannot find the ROM handler for Interrupt 10.

You can solve both of these problems with XSTI=10. No exclusion is necessary because the video ROM is no longer being used. Speed_up.sys can then be loaded after QEMM and (and can be loaded into upper memory). However, we strongly recommend that you NOT load SPEED_UP.SYS, RAMBIOS.SYS, FASTBIOS.SYS, or any similar driver. Using SPEED-UP.SYS costs you 36K of memory. Instead use QEMM's ROM parameter, producing the SAME effect but using NO address space between 0-1024K.

TECHNICAL BACKGROUND

All you need to know to use the XSTI parameter is contained above. If you REALLY want to understand what you are doing, keep reading. Otherwise, go sit out on the back porch and watch the sun set.

Q. What does StealthROM do to interrupts?

A. The StealthROM feature of QEMM allows you to map High RAM over ROMs by intercepting the interrupts that point into those ROMs and restoring the ROM into the Page Frame when the interrupt comes in, allowing the ROM's code to be run from the Page Frame. QEMM must divert all interrupts that point into a ROM it Stealths. Otherwise, when an undiverted interrupt comes in, control will pass to whatever QEMM has mapped into the High RAM in that portion of address space, rather than to the ROM that originally resided there.

Q. In what cases might QEMM not find an interrupt handler?

A. If a program you have loaded before QEMM or a ROM (all ROMs load before the CONFIG.SYS) loads an interrupt handler into RAM, then, when QEMM loads, QEMM will find this interrupt's handler not pointing into a ROM. An interrupt handler pointing into RAM cannot be Stealthed. If a device driver diverts this interrupt, you can load it after QEMM. If a ROM diverts this interrupt into RAM, you should see if there is a way to reconfigure the ROM so that it does not. On the INVISIBLE NETWORK, for instance, it is possible to reconfigure the network card (by means of a jumper) so that the ROM is no longer active and network services are provided by a program. In other cases, there may be a configuration program that performs this service.

If you cannot reconfigure the ROM to stop diverting this interrupt, then QEMM must be told not to try to Stealth this interrupt. This is what XSTI=XX does. Since the new interrupt handler may eventually call the ROM's interrupt handler, the ROM's interrupt handler for this interrupt may have to be left in place. This is done by eXcluding the portion of the address space where the ROM's handler for this interrupt resides. When you eXclude a portion of the address space of a ROM that QEMM Stealths, the underlying code that was formerly there returns.

You can get an idea where this interrupt is by looking at the First Meg/ Interrupts screen of Manifest, as it reports the beginning address of this interrupt. The acid test is to do an ANALYSIS with all the ROMs eXcluded, which will report what portion of the ROM's address space is being addressed directly. Typically, only an 8K eXclude is needed. If the handler for the target interrupt is being replaced entirely by the new interrupt handler, then the ROM's interrupt handler is never called. In this case, no eXclude is necessary. To be sure of this, you should still run an Analysis. (See the ANALYSIS section of your Manifest or QEMM manual.)

Q. What if some other program complains about StealthROM's interrupt diversion?

A. Some programs, when they load, check to see where the interrupt handlers they expect to use point. If an interrupt handler they expect to use is not pointing into a ROM, they think that an interrupt they wish to manage is already used by another program, and incorrectly assume that there is a conflict. Such programs will see Stealthed interrupts pointing into QEMM's code, rather than ROM, and may refuse to run. If such a program cannot be configured to ignore QEMM's diversion of the interrupt in

question, then this interrupt must be XSTIed and the appropriate eXclude found, by the means described above.

Some programs make a copy of the video ROM in RAM, and divert interrupt 10 (the video interrupt) into this new location for the video ROM's code. Such programs (RAMBIOS.SYS, FASTBIOS.SYS, RAPIDBIO.SYS are some examples) may refuse to load if interrupt 10 has been diverted. The best solution to this problem is to instead use QEMM's ROM= parameter, which instructs QEMM to perform this same service without using any addresses in the first megabyte of address space. If you wish to use such a program anyway, and it has the above complaint, then you must use XSTI=10. No eXclude is necessary, because such drivers usurp the video ROM entirely and INT 10 is never called again.

Q. What is FSTC?

A. The purpose of the FSTC parameter is to make the ANALYSIS procedure accurate. When QEMM Stealths a ROM, certain tables have to be stored by QEMM in its own data area. For a video ROM, this table occupies 12K; for a disk ROM, this table occupies 0.1K (If you have no explicit disk ROM, this table is in the system ROM.) When a ROM is being Stealthed, but the address in which the ROM resides is eXcluded, as with X=C000-C7FF, then QEMM won't need to make copies of these tables in its own data area. QEMM will automatically save memory by NOT making copies of the tables. This means that when you do eXclude the portion(s) of the ROM where these tables are stored, the ROM will be accessed directly. (This only holds true when you have used an eXclude.) This will cause Analysis to report that a portion of the address space is OK (when eXcluded) even though it would not be accessed directly were it not eXcluded.

FSTC (FORCESTEALTHTABLECOPY) forces QEMM to make copies of these tables so that inappropriate eXcludes are not recommended for the above reason. FSTC should only be used when you are testing a portion of a ROM's address space for direct access by eXcluding the whole ROM. It is not an appropriate parameter for a final configuration.

SUMMARY

The XSTI parameter is rarely needed. If you are loading any driver OTHER THAN QEMM 7.0's DOSDATA.SYS before QEMM in your CONFIG.SYS file, move QEMM above this driver. This step alone may solve the problem without the use of XSTI.

If you decide to use XSTI, you MUST determine the appropriate eXclude that will return the ROM code for handling the XSTIed interrupt to the address space it formerly occupied, because QEMM will no longer restore the ROM's code for the interrupt to the Page Frame and divert the interrupt there when it comes in.

Return to Technotes Main Menu.

QEMM: General Troubleshooting

Quarterdeck Technical Note #241

This is a very general guide to troubleshooting QEMM, and provides either quick fixes or references for additional information. It does not provide the detail available in the QEMM manual, which you should also consult. The troubleshooting section in Appendix A has many quick fixes for common problems.

As you proceed through this guide, please record carefully the results of each step. This is important; Quarterdeck Technical Support may need this information, and if you can provide a record of it, we can address your problem much more efficiently. In any case, you will find that this saves you time and trouble in further troubleshooting.

If your problem relates to one of the following, refer to the referenced technote:

A product-specific conflict (**PRODUCTS.TEC**)

Exception #6, #12, #13 (**EXCEPT13.TEC** and **EX13FLOW.TEC**)

StealthROM (**STLTECH.TEC** and **STEALTH.TEC**)

Microsoft Windows (**WINFLOW.TEC**)

MagnaRAM 2.0 **(MAGNARM2.TEC)**

Stacker (**STACKER.TEC**)

SuperStor (**SSTOR.TEC**)

MS-DOS 5 (**DOS5.TEC)**

MS-DOS 6 (**MSDOS6.TEC**)

DR-DOS or Novell DOS (**NW&DRDOS.TEC**)

High RAM Conflicts (**EXCLUDE.TEC**)

"Cannot find ROM Handler for INT xx" Error Message (**XSTI.TEC**)

Bus-mastering Device or SCSI Hard Drive (**BUS-MAST.TEC**)

Maximizing Conventional Memory (**MAXMEM.TEC**)

Parity Errors (**PARITY.TEC**)

Consult the note **CONTACT.TEC** or the Passport Support Brochure that accompanies your copy of QEMM for more information on contacting Quarterdeck.

Beginning the Troubleshooting Process:

If your system will not boot normally after installing QEMM, begin with Section A below.
If your system does boot normally, but you experience problems later on, begin with Section B below.

For the purposes of this troubleshooting guide, QEMM is comprised of the QEMM386.SYS driver (which provides EMS, XMS, VCPI memory management, High RAM, and miscellaneous other services) and

three other significant features, installed as separate drivers. These are:

DOS-Up, which includes DOSDATA.SYS and DOS-UP.SYS

QDPMI Host, which utilizes QDPMI.SYS

Stealth DoubleSpace, which utilizes ST-DSPC.SYS (ST-DBL.SYS in v7.0)

QSETUP and Manifest should be very helpful as you troubleshoot any problems that you may have with the QEMM package. Manifest provides detailed reporting on various aspects of your system's configuration, and, on its Hints / Overview and Hint / Detail screens, suggestions for improving your system's use of memory. You may use QSETUP to review or change QEMM parameters, to enable or disable the other drivers that come with the QEMM package, and/or to edit CONFIG.SYS and AUTOEXEC.BAT. We will use QSETUP in many of the steps below. To run QSETUP, simply change to the QEMM directory, and type QSETUP at the DOS command prompt. Though QSETUP runs as a Windows program, you may find it quicker to run QSETUP from the DOS prompt.

OPTIMIZE's /RESTORE parameter will allow you to restore past configurations quickly and easily. See the QEMM manual for more details on OPTIMIZE /RESTORE.

Section A -- Recovering Easily from a System Failure

If your machine fails to boot properly after QEMM has been installed, you may recover easily.

1) Reboot your machine. Use the power switch if necessary.

2) Wait until you hear a beep; then hold down the Alt key until the boot sequence stops. If your system does not beep on bootup, hold down the Alt key after you hear the floppy drive being accessed. When the boot sequence stops, you will see a message from DOSDATA or from QEMM. If the message is from QEMM, proceed directly to item (4) below. If the message is from DOSDATA, proceed to item (3).

3) If the message is from DOSDATA, prepare to hold down the Alt key again. Tap the Escape key to unload DOSDATA, and immediately hold down the Alt key again.

4) You will see the following message "QEMM: Press Esc to disable QEMM or any other key to continue with QEMM." Press the Escape key. Your system should then proceed with the boot sequence. QEMM will not be loaded, and no programs will be loaded into High RAM. You will likely see messages noting that there is not enough room to load your programs high; these messages are expected and no harm should result to your system as a consequence. Proceed to Section B.

Section B -- Determining if QEMM is the Problem

The first thing to determine is whether your difficulty is associated with the QEMM package at all. There are two lines in your CONFIG.SYS that read:

DEVICE=C:\QEMM\DOSDATA.SYS

DEVICE=C:\QEMM\QEMM386.SYS [parameters]

Using QSETUP, Manifest, or a text editor, disable QEMM entirely by placing the word "REM" before the word "DEVICE" on each line; reboot your system and try to reproduce the problem.

If the problem persists in exactly the same way as it always has, you can be reasonably sure that neither QEMM nor its associated drivers are the cause (since neither QEMM nor its drivers are active at this point). Make a note of this, and contact the vendor of the faulting application for assistance.

If this does relieve the problem, note that the problem does not persist when QEMM is inactive, and proceed to section C.

Section C -- Conflicts with DOS-Up, QDPMI, and Stealth D*Space

1) Disable the DOS-Up, QDPMI, and Stealth DoubleSpace features if any of them are active. Do this by running QSETUP, going to the main menu, and selecting each feature in turn. Answer "No" when you are asked if you would like to enable each one. Note that you should choose "No", and not "Partial" in response to the DOS-Up option. Note also that if you are not using DOS 6's DoubleSpace, the option to enable or disable Stealth DoubleSpace will not appear. Return to QSETUP's main menu, and select S for Save Configuration and Quit. If you are offered the opportunity to run OPTIMIZE, do NOT do so at this time. Reboot your machine without running OPTIMIZE.

2) If your problem is now solved, one of the QEMM features you have just disabled is likely in conflict with some other aspect of your system. Re-enable each feature, one at a time, and write down which feature you are enabling. It is likely that your system will fail before you re-enable the last feature. Write down the one that appeared to cause the failure; it is likely that this feature is the cause of the conflict. To be sure of this, re-eanble all features except the one that seems to be causing the conflict. Write down the results of this testing, and then proceed to section E below.

3) If your problem persists, but was solved by disabling QEMM in Section A above, the problem is likely related to the QEMM386.SYS driver. Write this down, and proceed to Section D below.

Section D -- Troubleshooting with the QEMM386.SYS Driver

Again, in your CONFIG.SYS file, there is a line that reads:

DEVICE=C:\QEMM\QEMM386.SYS [parameters]

Steps 1-5 in this section involve editing the [parameters] on this line, and nothing else. You may use a text editor such as DOS' EDIT, or the CONFIG.SYS editor in Manifest or QSETUP to make these changes. Every time you change the parameters on this line, you must reboot your computer for them to take effect. Write down the results of each step.

1) If there is a Stealth parameter ("ST:M" or "ST:F"), remove it and reboot. If this solves your problem, refer to the QEMM parameter STEALTHROM for an explanation of the parameter, and then refer to the technote "STEALTH TROUBLESHOOTING" (under the filename STEALTH.TEC) and follow its instructions. If removing the Stealth parameter fails, note the failure and proceed to Step 2.

2) Add the parameter "DB=2" to this line and reboot. If this solves your problem, refer to the QEMM parameter DISKBUF, and to the technote "BUS-MASTERING DEVICES AND QEMM" (under the filename BUS-MAST.TEC) for an explanation. If adding the DB=2 parameter fails, note the failure and proceed to Step 3.

3) Add the parameter "X=A000-FFFF" to this line and reboot. If this solves your problem, it is likely tha there is a conflict between QEMM's placement of High RAM and some piece of hardware on your system. To resolve the problem, refer to the QEMM Analysis Procedure (page xxx), or refer to the technote QEMM ANALYSIS PROCEDURE FOR SOLVING MEMORY CONFLICTS ("EXCLUDE.TEC" and follow the instructions for the Analysis procedure. If this EXCLUDE parameter fails, note the failure and proceed to Step 4.

4) Remove all the parameters on the QEMM386.SYS line and add:

APM:N BE:N BOOTKEY:Y CF:N DB=2 DM=128 FILL:N IOTRAP=64 LD MR:N P:VME:N RH:N SH:N TM:N TR:N VDS:N WC:N XBDA:N ON

then reboot. (Note that all of these parameters should be on the same line, the QEMM386.SYS line.) If this does not solve your problem, go to Step 5.

If this does solve your problem, it is probable that one (and only one) of the parameters above is required. All of these parameters, even taken together, do not seriously handicap the usefulness of QEMM. All together, they cause QEMM to use only 2K more conventional memory, 116K more extended memory, and will not cause QEMM to be any slower, except on a Pentium. You can find the one(s) you need by eliminating some and retaining others, noting the changes that you make each time. An efficient way of doing this is to remove half the list, writing down the parameters that you have removed. If the problem returns, one of the parameters that you have removed is the likely solution. If the problem does not return, one of the parameters still on the line. Continue to remove and restore parameters in this manner until you find the one that is required to solve your problem. When you are finished, you may consult the parameters section of the QEMM manual for an explanation.

5) If your system is still not working properly, add the parameter "NOEMS" and reboot. If this does not solve your problem, proceed directly to Step 7. If this solves your problem, some program that uses expanded memory is probably misbehaving, since this parameter causes QEMM to cease providing expanded memory. Write this information down, and go to Step 6.

6) In order to verify that the problem is with a program that is abusing expanded memory in general, try to reproduce the problem with DOS' memory managers. Add REM to the beginning of the QEMM386.SYS line in CONFIG.SYS, and add the following three lines immediately beneath the QEMM line:

DEVICE=C:\DOS\HIMEM.SYS

DEVICE=C:\DOS\EMM386.EXE RAM ON 1024

DOS=UMB

Reboot your system, and try to reproduce the problem. If the problem recurs, the problem is unrelated to QEMM, but instead is caused by some program that is mishandling expanded memory in some way. Note this important information, and contact the vendor of the faulting application.

7) Rename your AUTOEXEC.BAT to TEST.BAT, and copy your CONFIG.SYS file to another called C.SYS. Edit your new CONFIG.SYS with just the QEMM386.SYS line and "FILES=40", then reboot and attempt to reproduce the problem. If this solves the problem, run TEST.BAT. If the problem recurs, there was a conflict with something in your old AUTOEXEC.BAT. If the problem does not recur after you run TEST.BAT, there was likely a conflict with something in your old CONFIG.SYS. Restore all of the file that WASN'T a party to the conflict, and then restore, one line at a time, the lines in the file that WAS a party to the conflict, rebooting and testing after adding each line. You should be able to determine quickly which line was causing the problem.

It is possible that in this section, various elements of your system may not work properly, since there may be drivers in both CONFIG.SYS and AUTOEXEC.BAT that are essential for the operation of a given device. In this case, restore the lines necessary for the device in both CONFIG.SYS and AUTOEXEC.BAT.

In any case, if you have not found a solution to the problem, check Step 8, and then proceed to Section E.

8) It is possible that you have more than one problem, and that consequently you may need more than one of these solutions. When you have solved one problem, and are still having others, keep that solution and start over.

Section E -- If You Have Not Yet Resolved the Problem

Quarterdeck Technical Support is willing and ready to assist you with any compatibility problems that you might experience. However, you can help us to help you better by making sure that you have clear notes on all of the steps you have taken above. Even if these steps did not solve the problem for you, a record of your troubleshooting will put you in a much better position to get help, and will save you time.

If you are calling from the United States and you have a touchtone phone, we suggest you try 1-800-ROBOTECH, Quarterdeck's toll-free, automated technical support hotline. 1-800-ROBOTECH can assist with the most common technical questions and offer a variety of solutions. Navigation through 1-800-ROBOTECH is accomplished by pressing numbers on your phone's keypad to jump directly to the topic that you are interested in hearing about. The system will instruct you every step of the way. Call 1-800-ROBOTECH (1-800-762-6832), toll free, 24 hours a day, 7 days a week, including Holidays.

If you contact us by mail, fax or on one of our BBS systems, please include the following information:

Your Quarterdeck customer VIP number which you receive when you register your copy of QEMM with Quarterdeck.

The version number and serial number of QEMM. To find these out, type QEMM /REG at the DOS prompt.

If you are contacting us by mail or fax include a printout from Quarterdeck's Manifest. If you are using the DOS version of Manifest, press F2 to print, and select "All Manifest" from the "What to Print" portion of Manifest's print menu. If you are using the Windows version, select Print from the File menu, then select All Manifest. If you have other important hardware in the system, or if Manifest's list is incomplete, please include any additional information you think may help us diagnose your problem.

If you cannot run Manifest, print out your CONFIG.SYS and AUTOEXEC.BAT files, and write down what hardware (include the make and model) and software (include the version) you are using.

Give a precise description of the problem that is occurring, and the exact text of any error messages. Describe in detail the results of your troubleshooting efforts.

Please tell us how to respond to you via mail, fax or one of the other methods we support. See your Quarterdeck Passport booklet for information on contacting Quarterdeck Technical Support.

If you are contacting a technical support representative by telephone:

- Be at your computer.

- Please gather the information listed above.

- When you contact our technical support representative, you need only give your customer VIP number or product serial number and a brief description of your hardware, software and the problem you are encountering. If the support technician requires additional information, he or she will ask for specific details.

Return to Technotes Main Menu.

MS-DOS 6 and PC DOS 6 support multiple paths of execution through CONFIG.SYS. DOS 6 can use the CONFIG environment variable and the GOTO %CONFIG% batch statement to support separate paths of execution in the AUTOEXEC.BAT that correspond to the different CONFIG.SYS configuration paths. When you use QSETUP to add a new path to your DOS 6 multiple configuration CONFIG.SYS file, QSETUP does not create an entirely new branch in the AUTOEXEC.BAT to correspond to your new CONFIG.SYS path. Instead, QSETUP makes sure that the new configuration path and the existing one that it was based on will execute the same commands in AUTOEXEC.BAT.

If you want the new path to execute different AUTOEXEC.BAT commands than the path from which it was created, you must edit your AUTOEXEC.BAT file to create two separate branches to replace the common branch that QSETUP creates. See the DOS 6 manual for more information on the CONFIG variable.

With some expanded-memory-using RAM disks, the QuickBoot feature may not clear the contents of the RAM disk when you quickboot. Quickboot does not intentionally preserve the contents of any RAM disk and should not be relied upon for this purpose. To ensure that your RAM disk is cleared, warm boot normally by pressing Ctrl-Alt-Del twice in quick succession, thus bypassing quickboot.

READ ME File

This file includes tips to help you get the most out of QEMM. For that reason alone, it is worthwhile reading! It also contains last-minute information that did not make it into the manual and a few corrections to the manual.

Return to Hints, Technotes, and README Menu

ROM stands for Read-Only Memory--memory that is fixed in content and cannot be changed. The contents of ROM memory are not lost when the power is turned off. ROMs generally occupy addresses in upper memory. The BIOS and video services are among the programs contained in ROM.

Reclaim top memory

This feature enables or diables QEMM's ability to reclaim **top memory,** adding that memory to QEMM's memory pool. By default, QEMM reclaims unused top memory on Compaqs and other systems on which QEMM recognizes the presence of top memory. This feature typically adds 256K to 384K of RAM to QEMM's memory pool. Manifest's QEMM Memory screen will include a Top Memory row if QEMM is reclaiming top memory on your system.

To enable or disable QEMM's ability to reclaim top memory:

- Select **Yes** to enable reclamation of top memory.
- Select **No** to disable this feature.

This feature is on by default. If you disable it, QEMM Setup adds the TM:N parameter to the QEMM386.SYS line in CONFIG.SYS.

If QEMM has a problem reclaiming top memory on your system, you may experience a hang or reboot when QEMM386.SYS loads.

Reclaim unused shadow memory

This option lets you enable or disable QEMM's feature that reclaims unused portions of **shadow memory.** When QEMM reclaims shadow memory, it adds that memory to QEMM's memory pool for general use, typically giving you about 192K more usable RAM. By default, QEMM reclaims unused shadow memory, giving you more expanded or extended memory on systems that have any of the following: Chips & Technologies LEAP, AT386, NEAT, or SCAT ShadowRAM; or NEC, OPTI, PEAK or TOPCAT shadow memory,

To have QEMM reclaim unused shadow memory:

- Select **Yes** to reclaim shadow memory.
- Select **No** if you do not want QEMM to reclaim shadow memory.

QEMM reclaims shadow memory by default. When you disable the feature that reclaims shadow memory, QEMM Setup adds the SH:N parameter to the QEMM386.SYS line in CONFIG.SYS.

Manifest's QEMM Memory screen will include Shadow RAM information if QEMM is reclaiming shadow memory on your system. On some systems with unusual types of shadow memory, QEMM may have a problem reclaiming the unused portion. A common symptom is continual rebooting when QEMM loads, although other symptoms can occur. Disabling the shadow memory feature is a common troubleshooting technique.

Relocate Extended BIOS Data Area

This selection tells QEMM how to treat the **XBDA** (Extended BIOS Data Area).

To relocate the XBDA:

- Select **No** to tell QEMM not to move the XBDA.
- Select **Auto** to have QEMM determine where to most effectively place the XBDA.
- Select **Low** to move the XBDA to low conventional memory.
- Select **High** to force the XBDA into High RAM.

Below is a more detailed summary of these options:

AUTO is the default. QEMM moves the XBDA into **High RAM** unless it detects that you have a suspend/resume feature, that you have a machine (like some IBM ThinkPads) that fails with the XBDA in High RAM, or you place the **SUSPENDRESUME (SUS)** parameter on the QEMM386.SYS line in CONFIG.SYS. In these cases, QEMM moves the XBDA into low conventional memory.

If you select No, QEMM Setup adds the XBDA:N parameter to the QEMM386.SYS line in CONFIG.SYS. The XBDA will remain at the top of conventional memory where it will prevent video filling or the use of **VIDRAM** and will decrease the size of windows in DESQview and DESQview/X. You should choose No if you have a system or a program that expects the XBDA to be at the top of conventional memory. The symptom of this problem is usually a system crash, which can occur at boot time or later.

Low gives most of the benefits of moving the XBDA, and so is a less drastic way to try to solve any XBDA-related problem than choosing No.

You may want to select High (to save 1K of conventional memory) if QEMM is loading your XBDA low. If you do this on a laptop PC that has a Suspend/Resume feature, or on an IBM ThinkPad, your system may not work properly.

Moving the XBDA into High RAM lets VIDRAM and video filling work, increases the size of windows in DESQview and DESQview/X, and saves 1K of conventional memory. Moving the XBDA to low memory does not save conventional memory but gives all the other benefits listed above.

To find out where your XBDA is loaded, see Manifest's First Meg BIOS Data screen. If the third line on this screen does not say "0E: Extended BIOS Segment," then you do not have an XBDA.

If you do have an XBDA, check the four-digit hexadecimal address of the XBDA. If this address is 9FC0, then the XBDA has not been moved at all. If the address starts with 0 or 1, the XBDA has been moved to low conventional memory. If the address starts with a letter (A through F), then the XBDA is in High RAM.

Remove or set address of page frame

This option lets you specify the starting address of the **EMS page frame** or specify that QEMM should not create a page frame.

To specify the EMS page frame select one of the following:

- Select **None** to eliminate the page frame. This will disable the StealthROM and Stealth D*Space features and make expanded memory unavailable for programs. If you do not use Stealth or programs that use EMS, eliminating the page frame will make 64K of upper memory addresses available for **High RAM**, at the expense of all the benefits of having a page frame. We strongly recommend that you leave the page frame enabled.
- Select **Auto** to have QEMM choose the page frame address based on your system configuration.
- Select **Address** if you want to specify a particular address for the page frame. Then, click in the adjacent field and enter the 4-digit hexadecimal segment address for the beginning of the page frame. The address must be on a 16K boundary (i.e., its last 3 digits must be 000, 400, 800 or C00). You can specify the starting segment address of the page frame if a different location will consolidate two smaller High RAM regions into one large one, or if you need to place the page frame at the starting address of one of your ROMs to make the ROM work with the StealthROM feature. However, you should not set the page frame address if you do not know how to avoid conflicts between the page frame and ROM, adapter RAM, or video RAM.

Depending on your selection, QEMM Setup places the FR=NONE or FR=xxxx (where xxxx is a hex address) parameter on the QEMM386.SYS line in CONFIG.SYS, or removes the FR parameter from the QEMM386.SYS line.

QEMM Features

The Features page of QEMM Setup lets you review or change certain aspects of QEMM's behavior. When you select QEMM Features by clicking on its tab, you see a list of options. If you move the mouse pointer to an option, the option will become highlighted and you will see a brief description of what that option does in the **Feature Information** area near the bottom of the window.

Each QEMM Feature adds, deletes or modifies a parameter on the QEMM386.SYS device line in CONFIG.SYS. You can see QEMM's device line above the list of options. When you select an option, you will see how it modifies the device line. You can even edit the device line--just click at the point you want to edit.

IMPORTANT: After enabling or disabling any QEMM Feature, you reboot your PC in order for the change to take effect.

The options on the QEMM Features page are:

Fill Upper Memory with RAM

Copy ROMs to RAM

Enable QuickBoot

StealthROM Method

Set Size and Type of Disk Buffer

Reclaim Unused Shadow Memory

Reviewing and Editing Proposed Configuration Files

Options selected from the various QEMM Setup screens can result in changes to your CONFIG.SYS, AUTOEXEC.BAT, SYSTEM.INI, WIN.INI, and FREEMEG.INI files. Changes that result from your selections are not saved until you choose **Save** at the bottom of the QEMM Setup display.

To review proposed changes to these files, select one of the following options from the **File** menu on the menu bar:

Proposed CONFIG.SYS

Proposed AUTOEXEC.BAT

Proposed SYSTEM.INI

Proposed WIN.INI

Proposed FREEMEG.INI

The file you select is displayed and can be reviewed and edited as desired. At any time during the editing process, you can click **Reset** to discard any changes you have made and continue editing.

When you finish reviewing and editing the file, click **OK** to close the editor window or **Cancel** to discard any changes you have made to the file and close the window. If you choose **OK**, you must still select **Save** when you return to QSetup's main screen in order to save your changes.

If you work exclusively in Microsoft Windows, you can delete the contents of the QEMM\TECHNOTE subdirectory, at a savings of about 400K of disk space. All the QEMM technotes are also included in this help file, and can be read by selecting the Technote option from QEMM Setup for Windows Help menu.

Set size and type of disk buffer

This option allocates additional memory to buffer disk reads and writes. Buffering may be necessary if you are experiencing problems with QEMM on a system with a **bus-mastering hard disk** or if there are conflicts between an EMS-using disk utility and QEMM's StealthROM feature.

To set the size and type of disk buffer:

- Select **None** to remove any disk buffering.
- Select **Auto** to have QEMM attempt to determine whether you need a disk buffer to resolve problems with a bus-mastering hard drive. If it detects a bus-mastering conflict with the drive from which QEMM loads, QEMM Setup will add the parameter DB=2 to the QEMM386.SYS line in CONFIG.SYS.
- Select **Full** to have QEMM intercept all disk reads and writes to resolve problems with a bus-mastering hard disk. This adds the DB=xxx parameter to the QEMM386.SYS line in CONFIG.SYS. If you select the Full option, you should set the number of kilobytes to reserve for the disk buffer; just click in the box on the right side of the field and type a number. 2 and 10 are commonly-used values. This type of disk buffering eliminates problems with bus-mastering hard disks, but with a penalty in conventional memory and disk performance.
- Select **Frame** to resolve conflicts between an expanded memory-using disk utility and StealthROM. This selection tells QEMM to buffer only disk reads and writes into the EMS page frame. It adds the DBF=xxx parameter to the QEMM386.SYS line in CONFIG.SYS. If you select the Frame option, you should set the number of kilobytes to reserve for the disk buffer; just click in the box on the right side of the field and type a number. 2 and 10 are commonly-used values.

Set up QEMM for troubleshooting

If you wish to troubleshoot a particular problem, you can set up QEMM for troubleshooting. When you set up QEMM for troubleshooting, QEMM Setup will place the following ten parameters on the QEMM386.SYS line in CONFIG.SYS:

DB=2 SH:N TM:N TR:N CF:N FILL:N MR:N RH:N XBDA:N BE:N

To set up QEMM for troubleshooting:

- Select **Yes** to add the troubleshooting parameters,
- Select **No** to remove the troubleshooting parameters.

These are not the only QEMM386.SYS parameters that can solve problems, but they are the easiest to try as part of a one-step troubleshooting process. If your problem goes away after you enable the troubleshooting parameters, you should try eliminating the parameters one by one until you find the parameter that solved the problem, then take all the other troubleshooting parameters off the QEMM386.SYS line. Be sure to reboot whenever you add or delete parameters.

SkipFile List

QEMMs **FreeMeg** feature normally safeguards the first megabyte of memory whenever any Windows program loads. However, in case a particular program does not load properly when FreeMeg is active, QEMM keeps a list of programs for which it deactivates FreeMeg. Only the programs on this SkipFile list load without FreeMegs protection; all other programs will still be unable to monopolize precious first-megabyte memory when they are loading.

To add a program to the SkipFile list, click the **Add** button next to the list, and type or select the name of the program, complete with file extension, into the Add to SkipFile List dialog box. If you do not see the name of the file you want to add to the list, select a new drive or directory

To edit the Skip File list, select the entry on the list that you want to change and click the **Edit** button next to the list. Use the Edit Skip File List dialog box to change the entry.

To delete an entry from the Skip File list, select the entry and click the **Delete** button next to the list.

StealthROM is a QEMM feature that creates additional mappable areas at the addresses used by your PC's ROMs. By default, QEMM will turn these areas into High RAM that can be used to load TSRs, device drivers and selected parts of DOS. When StealthROM is enabled, QEMM monitors the interrupts pointing into those ROMs. When those interrupts occur, QEMM maps the appropriate ROM into the page frame and passes the interrupts to the ROM's location in the page frame. In general, the ROMs targeted are your system ROM, video ROM and disk ROM, although certain other ROMs may be "Stealthed" as well.

With the ROMs out of the way, the amount of usable upper memory is greatly increased. Depending on the location of the ROMs, High RAM regions can become quite large and able to accommodate more or larger device drivers and TSRs.

Stealth Troubleshooting

Quarterdeck Technical Note #205

StealthROM may seem mysterious and cryptic, but it really is not. This note tells you how to diagnose and solve problems related to the StealthROM feature of QEMM. Although this note may appear lengthy, it is detailed and informative. For those who are more interested in fast solutions, the path to resolving a problem is quite straightfoward and quick.

Q. What do I need to know first?

Before beginning the steps outlined in this technote, please review the information in Quarterdeck Technical Note #248, "Product Compatibility Information" (**PRODUCTS.TEC**), which contains information on various hardware and software products that may require specific treatment with StealthROM.

If you are reading this technote because QEMM displayed the message "Disabling StealthROM becauase QEMM cannot find the ROM handler for INT xx", you should instead refer to Quarterdeck Technical Note #233, "QEMM and the XSTI Parameter" (**XSTI.TEC**).

You may also wish to consult Quarterdeck Technical Note #168, "QEMM's StealthROM Technology" (**STLTECH.TEC**) for background information on StealthROM and how it works.

Q. How do I resolve a conflict with StealthROM?

In almost all cases, the OPTIMIZE program that comes with QEMM will detect and resolve automatically any aspect of your system's hardware or basic configuration that is incompatible with Stealth. OPTIMIZE cannot, however, anticipate the behaviour of programs that are not run as part of your CONFIG.SYS, AUTOEXEC.BAT, or other batch files that are CALLed as part of your system's startup process. The following troubleshooting procedure is divided into simple steps, contained in several sections, to address compatibility issues with programs that are not run as part of the startup process, or that manage to evade OPTIMIZE's automatic handling of StealthROM.

SECTION ONE

1) The first step is to ascertain whether StealthROM is involved with the problem. Remove the StealthROM parameter (ST:M or ST:F) from the QEMM device line in your CONFIG.SYS file and rerun Optimize. When Optimize completes, try to duplicate the problem. If the problem still happens, then StealthROM is not causing the problem, and you should refer to the troubleshooting section of your QEMM manual for further information. However, if removing ST:M or ST:F solves the problem, proceed to Step 2.

2) Now that we have determined that StealthROM is involved in the problem, add StealthROM parameters ST:M and XST=F000 to the QEMM device line in the CONFIG.SYS file. Your QEMM line would look something like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XST=F000

Reboot the computer. If this step solves the problem, go on to Step 3 immediately below; if it does not, go to SECTION TWO.

3) If XST=F000 solves your problem replace this parameter with X=F000-FFFF, reboot your computer, and try again to create the problem. The QEMM line in your CONFIG.SYS file would look something like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=F000-FFFF

If the problem persists, go to Step 4 below. If you cannot recreate the problem with the above line, add the parameter FSTC to the end of the line as follows:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=F000-FFFF FSTC

and reboot. If this step solves the problem, continue on. If it does not (and FSTC may not work in all circumstances) then remove the FSTC parameter and reboot with the previous QEMM line. In either case, run Manifest (by typing MFT from the DOS prompt) and look at the QEMM/Analysis screen. The last line should look something like this:

Fn00 IIII IIII IIII IIOO

Since the entire F000-FFFF range is EXCLUDEd, Analysis here is suggesting that the last two 4K pages of the F region are Okay (that is, the EXCLUDE is appropriate for the region FE00-FFFF) and that the other pages are INCLUDEable (that is, the EXCLUDE is not needed for these pages). This is so because some program or piece of hardware is trying to read the contents of the last two pages of ROM directly, rather than accessing them through interrupts. QEMM must be prevented from mapping High RAM over this ROM to allow the ROM to be be accessed directly. This is done by using the EXCLUDE parameter. In our example the target region is FE00-FFFF; thus appropriate EXCLUDE is X=FE00-FFFF and the QEMM line would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=FE00-FFFF

This EXCLUDE allows StealthROM to do its job and costs only 8K of High RAM. If you are at this point, your problem is solved and you do not need to continue with this technical note.

4) If XST=F000 solves your problem while X=F000-FFFF does not, try changing ST:M to ST:F, and remove XST=F000. You may end up with more High RAM with ST:F than with ST:M and XST=F000. However, if ST:F fails, remove ST:F and place ST:M and XST=F000 back on the QEMM386.SYS line.

SECTION TWO

This section is ONLY for systems that have video ROMs--that is, systems with an EGA or VGA video card. CGA color and Hercules-compatible monochrome systems do not have video ROM; If you have one of these systems, proceed to Section Three.

Manifest, on its First Meg Overview screen, will identify the ROMs on your system when StealthROM is inactive. On most systems, the Video ROM is located at C000-C7FF and uses 32K of upper memory address space. Some machines (particularly Micro Channel machines such as the IBM PS/2 family) have a video ROM elsewhere, generally at E000-E7FF. If this is true of your system, you should use XST=E000 (or wherever your video ROM begins) instead of XST=C000 in the QEMM386.SYS lines that follow.

5) If XST=F000 does not solve your problem, try XST=C000. The QEMM line of the CONFIG.SYS would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XST=C000

If XST=C000 does not work, go on to Section Three. If XST=C000 does work, proceed to Step 6 immediately below.

6) If XST=C000 solves the problem, try placing the page frame at C000 by removing XST=C000 and adding FR=C000 to the QEMM386.SYS line. Do this only if the entire C000 range is available--that is,

no device is using address space between C000 and CFFF for Adapter ROM or RAM. The QEMM line of the CONFIG.SYS would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M FR=C000

If this works, you may find this an acceptable solution. All the address space in which High RAM can be created is being used in this configuration, and you will want to stop troubleshooting here. If this step does not work (or if you cannot put the page frame at C000) go on to Step 7 below.

7) If XST=C000 solves the problem but you do not want to put the page frame at C000 (or cannot for some reason) then try the parameter F10:N (or FASTINT10:N). By default QEMM uses its own code for some video functions, rather than the video ROM's own code. The F10:N parameter tells QEMM not to perform this function. The QEMM line of the CONFIG.SYS would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M F10:N

If this step works, all ROMs are being Stealthed at the expense of some speed. You may find this satisfactory, at which point you may stop here. If you do not find the modest decrease in speed acceptable, or if this step does not work, proceed to Step 8 immediately below.

8) If XST=C000 solves the problem but FR=C000 or FASTINT10:N does not (or you either cannot put the Page Frame at C000 or do not want to use FASTINT10:N) then replace XST=C000 with X=C000-C7FF. The QEMM line of the config.sys would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=C000-C7FF

If replacing the XST=C000 parameter with X=C000-C7FF does not work then you should replace the X=C000-C7FF with XST=C000, which will avoid all Stealth conflicts with your system's video ROMs; you may stop troubleshooting here.

If replacing the XST=C000 parameter with X=C000-C7FF works, add the parameter FSTC to the QEMM line:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=C000-C7FF FSTC

Reboot your computer. If this step works, continue on; if this step does not work (and FSTC may not work in all circumstances) then remove the FSTC parameter and reboot with the previous QEMM line.

In either case, run Manifest (by typing MFT at the DOS prompt) and look at the QEMM/Analysis screen. The Cn00 line should look something like this:

Cn00 OIII IIII OOOO OOOO

This indicates that the first 4K region of the C000 range (in the video ROM) is being accessed directly. This portion of the address space must be EXCLUDED from QEMM's use when StealthROM is enabled. The appropriate QEMM line in the CONFIG.SYS would be:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M X=C000-C0FF

SECTION THREE

9) On some machines there are other ROMs that can be Stealthed. Typically these are disk ROMs; sometimes they are associated with network cards or with other peripherals. To determine if your system has one of these other ROMs, boot your system without ST:M or ST:F and check the First Meg/Overview screen in Manifest. If you see more than two ROMs listed, than your system has another ROM.

Troubleshooting StealthROM with these ROMs involves a similar procedure to those above; That is, add XST=??00 (using the beginning address of that particular ROM). If this solves the problem, replace the XST=??00 with the appropriate EXCLUDE (X=??00-!!FF, replacing ??00 and !!FF with the beginning and ending addresses of the ROM) to determine whether the problem is related to Stealth or to the fact that some portion of the ROM's address space needs to be directly accessible.

If some portion of the address space must be EXCLUDED for StealthROM to work you should check Analysis with the FSTC and X=??00-!!FF parameters on the QEMM line in a manner similar to that detailed in Step 3.

The trick of placing the page frame at the beginning of the ROM may also work here as well. If XST=??00 solves your problem, try replacing it with FR=??00. This is possible if there is a 64K portion of the address space that is either ROM or RAM beginning at ??00, and if ?? is a multiple of 16K.

10) Use XST=F000, XST=C000, and XST=??00 simultaneously for all ROMs being Stealthed. Then replace the XSTs one by one with the appropriate regular Exclude (X=F000-FFFF, X=C000-C7FF, X=??00- !!FF...) and look at the QEMM/ Analysis screen of Manifest to see what portions of the address space need to be directly available.

11) If ST:M with XST on all Stealthed ROMs fails, add the following parameters to the QEMM line we used in Step 10:

FB:N UFP:N VHI:N F10:N

If this works, remove the XSTs one at a time to determine which one(s) you actually need. If an XST alone does not solve the problem, turn back to SECTION ONE for help in determining the appropriate XST or EXCLUDE while keeping the above parameters. The parameters are explained later in this technote as well as in your QEMM manual.

12) If ST:M does not work, try ST:F instead. If ST:F does not work, try ST:F XST=C000 (and XST=??00) for Stealthed ROMs other than the one(s) the page frame overlays.

13) If none of these steps solves the problem, Quarterdeck's Testing and Compatibility Department wants to know what program or device is failing (and at what point). Please contact our Technical Support staff so that our technicians can get all the necessary information about your problem.

The following section contains questions and answers on various aspects of Stealth and its parameters. Again, you may wish to consult Quarterdeck Technical Note #168, "QEMM's StealthROM Technology" (**STLTECH.TEC**) for background information on StealthROM and how it works.

Q. I have a ROM that is not being turned in High RAM by StealthROM. Why?

QEMM's StealthROM feature relocates ROMs which are accessed through the use of interrupts. A ROM that is not Stealthed may still have regions that are never used during the system's normal operation. Some disk ROMs, video ROMs, and devices that only use their ROMs during bootup (before the CONFIG.SYS is loaded) fit into this category. You may be able to get more High RAM by INCLUDING these unused regions on the QEMM386.SYS device driver line in CONFIG.SYS. To find out what addresses you can include, run the Analysis procedure, then use the INCLUDE parameter to include these addresses. Refer to your QEMM manual for further information on Analysis.

Q. What is FASTINT10:N?

When Stealthing a video ROM, QEMM replaces some of the video ROM's routines with code written by Quarterdeck. This replacement code is suitable for almost all video cards. The FASTINT10:N parameter (which may be abbreviated F10:N) tells QEMM not to use its code but the code of the video ROM instead. This in no way limits the amount of High RAM StealthROM creates and may be an

acceptable solution for those users who need it. It should only be necessary on unusual video cards. If placing the page frame at the beginning of the video card's ROM works or if a small regular exclude also solves the problem you may choose to use F10:N instead.

Q. What is FSTC?

In addition to program code, ROMs contain various tables of information. A ROM may itself access these tables directly. When the ROM is being Stealthed normally, StealthROM will prevent direct access to the ROM by copying various tables into QEMM's own data area, which uses some extra memory -- usually in High RAM. When a ROM is being Stealthed but the address in which the ROM resides is EXCLUDEd (as with X=C000-C7FF), QEMM cleverly figures out that it does not need to make copies of these tables in its own data area. QEMM therefore saves memory by not making copies of the tables. However, if the copy is not made, portions of the ROM where these tables are stored will be accessed directly. This will cause Analysis to report that a portion of the address space is OK when EXCLUDED even though it would not be accessed directly were it not EXCLUDED.

FSTC (which stands for ForceStealthTableCopy) makes the Analysis procedure accurate by forcing QEMM to make copies of these tables so that inappropriate EXCLUDEs are not recommended.

Some system and video ROMs may not function properly with the FSTC parameter. If this is the case on your system you will have to perform the Analysis procedure without the FSTC parameter. However, you should be aware in this case that some of the EXCLUDE statements that Analysis prompts you to use may not be necessary. You can try reducing these EXCLUDES on a trial-and-error basis if you wish.

FSTC should only be used when you are testing a portion of a ROM's address space for direct access by excluding the whole ROM. It is not an appropriate parameter for a final configuration.

Q. What is FB:N?

FB:N, short for FRAMEBUF:N, disables QEMM's feature of breaking up disk reads into the page frame and disk writes from the page frame. FB:Y is the default with StealthROM; FB:N is the default without StealthROM.

Q. What is UFP:N?

When StealthROM is active, and when no program is using expanded memory, QEMM unmaps the memory left in the page frame. This allows the ROM underneath the page frame to be visible, in case some program reads that ROM directly. However some programs use expanded memory and then free an EMS handle (which is tantamount to ceasing to use that expanded memory) and yet expect the page frame to contain the memory that they last mapped there. The UFP:N (abbreviation for UNMAPFREEPAGES) parameter tells QEMM not to unmap the EMS page, which will make such programs work with StealthROM. Of course such programs violate the EMS specification by abandoning the EMS handle (and apparently common sense) but expecting the page frame to contain the memory just released. UFP:N is the default without StealthROM.

Q. What are advanced disk features? What is VHI:N?

The BIOS has a set of function calls intended for use by multitasking programs. These are INT 15h, functions 90h and 91h. The system ROM or disk ROM may issue the INT 15h, Fn 90h call while it is waiting for the disk controller to read or write a sector, allowing other programs to execute during this wait. When the sector is ready, the disk interrupt handler issues an INT 15h, Fn 91h, signalling the multitasking program that the disk information is ready to be processed by the system or disk ROM.

The "advanced features" of some disk caches hook this call to allow your system to go ahead and execute your current program while the system or disk ROM is waiting for its requested sector.

Although such caches properly preserve the stack and register state for the BIOS and the application when doing this pseudo-multitasking, they do not preserve the mapping of the page frame. Therefore, if a BIOS call causes the page frame to be used (as is the case with StealthROM active), conflicts and system failures could result. Since most disk caches do not preserve the page frame properly, QEMM automatically suppresses INT 15h, Fn 90h calls from the BIOS, effectively disabling advanced disk features. Caches that save and restore the page frame when using advanced disk features can use a programming interface to QEMM to re-enable advanced disk features.

You may defeat QEMM's defeating of this feature with the VIRTUALHDIRQ:N (VHI:N) parameter on the QEMM device line in your CONFIG.SYS file. If your cache uses INT 15h Fn 90h as one of its advanced features, and does not save and restore the page frame you will crash or corrupt data on the cached drive(s).

Q. Can I load a driver that uses ROM before QEMM?

You can load a device driver that uses a ROM before QEMM and still Stealth that ROM by loading the driver HOOKROM.SYS (found in your QEMM directory) before you load this driver. directory. Here is an example of a CONFIG.SYS file that loads HOOKROM, then a driver that uses a ROM, then QEMM with StealthROM enabled:

DEVICE=C:\QEMM\HOOKROM.SYS DEVICE=C:\DISK\ROMDRIVER.SYS DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M

For a more detailed explanation of HOOKROM, refer to XSTI.TEC in your QEMM\TECHNOTE directory.

Q. Why does my computer's BIOS Setup program fail when I use ST:M?

Many machines have a built-in System Setup program in the BIOS ROM that can be popped up via a keystroke. StealthROM may make this feature inaccessible to you after your system has booted. Because the Setup program accesses the ROM directly, you must EXCLUDE the portion of address space where it is stored in order for it to work after QEMM has been loaded and StealthROM enabled. On most machines this Setup program is found in F000- F7FF.

You may decide that you would rather have access to the Setup program ONLY on bootup and use this portion of the address space for High RAM. Since you must reboot your computer after making changes in your CMOS Setup anyway, most users consider this a fair trade.

Q. How does StealthROM work? What can cause it to fail?

Refer to consult Quarterdeck Technical Note #168, "QEMM's StealthROM Technology" (**STLTECH.TEC**) for background information on StealthROM and how it works and what can interfere with it.

SUMMARY

The StealthROM technology has been exhaustively tested. Due to the wide variety of software and hardware in the PC world, however, StealthROM cannot be guaranteed to work with every configuration. The actual Stealthing of interrupts is very successful. Most failures are due to programs (or other ROMs) trying to access a portion of the ROM directly rather than by means of interrupt, and most of these failures can be resolved via the techniques in this note.

If you experience a problem that you are unable to resolve using the steps outlined in this technical bulletin, Quarterdeck would very much like to hear about it.

Return to Technotes Main Menu.

StealthROM method

This selection lets you enable or disable QEMM's **StealthROM** feature. StealthROM can typically free 48K-115K of upper memory addresses which can then be used for **High RAM** or expanded memory mapping.

To enable or disable StealthROM:

- Select **Mapping** to enable StealthROM using the **mapping** method.
- Select **Frame** to enable StealthROM using the **frame** method.
- Select **None** to disable StealthROM.

If you select Mapping, QEMM Setup will add the ST:M parameter to the QEMM386.SYS line in CONFIG.SYS. If you select Frame, Setup will add the ST:F parameter. If you select None, Setup will remove the ST parameter.

QEMM offers to enable StealthROM during the installation process or the Optimize process if it sees that you need additional High RAM. If you use DESQview or DESQview/X, you should use the StealthROM feature even if QEMM has not enabled it for you.

Switching Between MS-DOS 6's Memory Manager and QEMM

QEMM provides all the functionality of MS-DOS 6's memory manager, and much more. For a list of QEMM's features and how they stack up against DOS 6's memory manager, see "QEMM Benefits and Features" in Chapter 1 of the QEMM Reference Manual.

If you are using MS-DOS 6 and you have run its MemMaker memory utility, you can switch back to QEMM by running QEMM's Optimize program (assuming you have already installed QEMM on your hard disk). To run Optimize, type **OPTIMIZE** at the DOS prompt.

If you ever want to switch back to MS-DOS's memory manager, simply run MemMaker again. If you are using QEMM's **DOS-Up** feature, be sure to run QEMM Setup and disable DOS-Up before you run MemMaker.

If you are using QEMM's **Stealth D*Space** feature and you decide to switch back to MS-DOS 6's memory manager, the **ST-DSPC.SYS** driver will perform the same function as DOS's **DBLSPACE.SYS /MOVE** or **DRVSPACE.SYS /MOVE**. It will allow **DBLSPACE.BIN** or **DRVSPACE.BIN** to be moved into upper memory. You can replace ST-DSPC.SYS with DBLSPACE.SYS or DRVSPACE.SYS, but you will suffer no ill effects if you do not.

Return to Hints Main Menu.

Technotes

Technotes are available on the following subjects:

Contacting Technical Support **(CONTACT.TEC)**

Return to Hints, Technotes, and README Menu

PC Tools (Symantec)
DoubleDisk (Vertisoft)
Fastback (Symantec)
NDOS (Symantec)
Btrieve (Novell)
QRAM (Quarterdeck)
Stacker (Stac Electronics)
MSCDEX CD ROM Driver (Microsoft)
XtraDrive (Integrated Information Technologies)

Undoing an Optimize

When you run QEMM's **Optimize** program, modifications are made to your CONFIG.SYS and AUTOEXEC.BAT files and any batch file called by AUTOEXEC.BAT.

If for some reason you want to restore these files to their pre-optimized states, type **UNOPT** at the DOS prompt. The files will be restored to the state they were in before you last ran Optimize.

Optimize saves your last nine configurations as well as the configuration you were using when you installed QEMM 8. (Your original configuration files will be saved regardless of the number of times you run Optimize.)

To see a list of these saved configurations, type **OPTIMIZE /RESTORE** at the DOS prompt. You can then select which configuration you would like Optimize to restore.

Return to Hints Main Menu.

VIDRAM is a QEMM program you can use to extend conventional memory when running DOS text-based programs on a system with an EGA or VGA adapter. By using VIDRAM, you can get an additional 64K-96K of memory to run such programs. However, you cannot use EGA or VGA graphics while VIDRAM is on. For information on VIDRAM, see Chapter 6 of the QEMM Reference Manual.

VIDRAM: Extending Memory for Text-based Programs

QEMM's **VIDRAM** program can extend conventional memory by as much as 96K for running DOS text-based programs. VIDRAM even extends conventional memory for DOS text programs running in Microsoft Windows.

To use VIDRAM, your system must have an **EGA** or **VGA** video adapter or an adapter with EGA or VGA capability (this includes VGA- compatible 8514A video adapters). Your PC must have 640K of conventional memory and the programs that you run while using VIDRAM must not use EGA or VGA graphics.

If your PC has an EGA or VGA video adapter, the 64K memory area just above conventional memory (640K-704K or A000-AFFF hex) is reserved for use by graphics modes. When you run text-based programs, that area is unused, so VIDRAM can appropriate it to extend the contiguous conventional memory for running programs.

It is important to understand that you cannot run EGA or VGA graphics operations while VIDRAM is in use. If you routinely use both large text-based programs and graphics programs, you can turn the VIDRAM feature on when you need it for a text program and off before you run a graphics program. If you are using an 8514A adapter, you can still use 8514 graphics programs while VIDRAM is enabled.

To turn VIDRAM on type **VIDRAM ON** at the DOS prompt.

This command will extend conventional memory into the EGA/VGA graphics area for a total of 704K conventional memory.

To turn VIDRAM off so you can use graphics programs again, type **VIDRAM OFF**.

For more information on VIDRAM, including using VIDRAM with Microsoft Windows, DESQview, and DESQview/X, and extending conventional memory an additional 32K, see Chapter 6 of the QEMM Reference Manual.

Return to Hints Main Menu.

While manufacturers of video cards often recommend EXCLUDEing large regions of address space -- for example, A000-C7FF -- this is almost never necessary, and these recommendations should be taken with a good deal of skepticism. QEMM is intelligent enough to recognize the addresses used by video cards, and makes the
appropriate exclusions automatically. QEMM INCLUDEs the B000-B7FF region by default, and EXCLUDEs video RAM and ROM automatically when appropriate. A few video cards require that the address space between B000-B7FF be EXCLUDEd when using Microsoft Windows high-resolution video drivers. However, it may be possible instead to use QEMMExclude=B000-B7FF in the [386Enh] section of the Windows SYSTEM.INI file. This parameter is documented in the Technical Reference section of the QEMM manual.

Hints, Technotes, and READ ME File

The following technical information is available from this page:

- **Hints**

  Getting the most out of QEMM. Includes information on using QEMM with various combinations of hardware and software.

- **Technotes**

  Detailed technical and troubleshooting information.

- **READ ME**

  QEMM's READ ME file that contains last-minute changes not covered in the documentation.

Why the EMS Page Frame is Important

Quarterdeck Technical Note #295

This document explains how the EMS page frame can save you much more than the 64K of High RAM than it requires, and why disabling the page frame is a bad idea.

Technical support staff at some companies will sometimes suggest that you disable the expanded memory page frame in order to get 64K more High RAM. This is short-sighted and wasteful. The expanded memory page frame is one of the most valuable resources available to increase the amount of memory available to your DOS programs.

To understand the usefulness of the EMS page frame in a non-technical way, suppose an empty space, 16" x 12", on an otherwise blank wall in your living room. Some people might put up a painting (which displays one thing, all the time), but most would prefer a television screen (in which you can see what you want, when you want to see it).

On a more technical level, the page frame is a 64K window of address space, typically located above the 640K line, that can be shared and used by multiple programs to reduce their overhead. To understand how expanded memory works, it is most useful to understand the concept of mapping. Mapping is the process by which memory management hardware and software can make memory appear in appropriate places at appropriate times; it is the process of associating memory with an address other than its actual one. The expanded memory specification (EMS) uses mapping to make portions of expanded memory appear inside the EMS page frame when that memory is requested by a program. When a program needs more memory than what is normally available to it under DOS, it can request that some expanded memory be allocated from either an EMS board, or from the extended memory managed and made to appear as expanded memory by a 386 memory manager such as QEMM.

Expanded memory has no addresses of its own, but can be made to appear at a valid address -- "mapped in". Expanded memory pages that are not currently needed may be "mapped out" -- relieved of their addresses and put back into the expanded memory pool, with code and data still intact. When the application needs these pages, they are "mapped in" to the EMS page frame again. It is therefore possible for a program that uses expanded memory to have access to much more memory than DOS itself can see of its own accord. This is similar in concept to bank switching and paged memory systems, techniques used to extend and add power to everything from mainframe computers to high-end UNIX systems to DOS machines. Any program loaded on your system may use EMS at any time, even while other programs have access to it.

Mapping is also useful for creating High RAM; in the same way as detailed above, memory can be associated with unused addresses between 640K and 1MB. The 386 hardware and QEMM cooperate to make memory appear where there is otherwise none; this memory is called High RAM. Programs can be loaded into High RAM instead of conventional memory. This allows more room in conventional memory for DOS programs. Unlike the page frame, however, only one program at a time can occupy a block of High RAM.

QEMM's StealthROM feature uses mapping for yet another purpose. The 386 chip can be made to map memory in or out of DOS' address space at any time. StealthROM uses the page frame and 386 mapping to map system, disk, or video ROMs in and out of DOS' address space when appropriate. More information on StealthROM is available in Quarterdeck Technical Note #168, QEMM's StealthROM Technology (**STLTECH.TEC**).

The Quarterdeck Expanded Memory Manager, QEMM, provides expanded memory services, allowing any EMS-using program on your system to take advantage of expanded memory. QEMM itself also takes advantage of expanded memory for its StealthROM, SqueezeFrame, and Stealth D*Space features.

Thus any advice to remove the page frame is penny-wise and pound-foolish. Remember that the page frame is 64K of address space that can be used any program, at any time, to access effectively as much memory as it likes. Some view the page frame as 64K of address space that could be used to hold up 64K of programs, but it is much more useful to consider the page frame as a place to access up to 32 megabytes of code and/or data for the programs that use it. The distinction is very similar to the difference between a TV and a painting.

On an example system, with the page frame enabled, StealthROM can create an 83K of extra High RAM. This alone justifies the investment in the page frame, returning an extra 19K. Stealth D*Space can also use the page frame, reducing the overhead for Microsoft's DoubleSpace or DriveSpace disk compression utilities by 40K. Stacker's EMS feature can permit similar memory gains.

This example system is on a Novell network. If the page frame is enabled, one may use EMSNETX as the network redirector instead of NETX. The overhead for the latter is 44K; for the former it's a little less than 10K. When EMS is available, VSAFE, on that system, reduces its overhead from 22K to 6.5K; MSCDEX goes from 35K to 15K, and so on. Thus 194K of code is loaded for an investment of 64K, at a net savings of 130K.

In addition to these savings, EMS is also available to DOS application programs that can use it. If an application uses EMS, it can reduce its conventional memory overhead dramatically, and/or improve its performance. The Lotus 1-2-3 Release 2 series, the most widely-installed version of Lotus, uses expanded memory; WordPerfect 5.1 similarly uses expanded memory. Neither of these programs uses XMS (or any other flavour of) extended memory. VCPI, a memory management specification for DOS Extended applications, depends on an expanded memory manager to be present. Not all VCPI applications require a page frame, but many of them attempt to map a page in the page frame, and refuse to run if they can't.

In summary, it is imprudent to disable the EMS page frame in order to create more High RAM. For a 64K investment, you can typically recover a good deal more memory.

Return to Technotes Main Menu.

Windows

The Windows page of QEMM Setup lets you review or change the configuration of QEMMs Windows features: **FreeMeg**, **Resource Manager**, and **MagnaRAM**.

If you move the mouse pointer to one of the configuration options, the option will appear in highlighted text and you will see a brief description of what that option does in the Information area near the bottom of the window.

IMPORTANT: After making changes to the Windows tab, you must restart Windows for the changes to take effect.

The following options are available on the Windows page:

FreeMeg

- Allocation Method
- Block Size
- SkipFile List

Resource Manager

- Enable Resource Manager Option

MagnaRAM

- Enable MagnaRAM Memory Compression
- Compression Buffer Size
- Compression Threshold
- PAGEOVERCOMMIT

XSTI.TEC

This abridged QEMM technote is available in its entirety from the following sources:

Quarterdeck Technical Support BBS: **XSTI.TEC**
CompuServe: **XSTI.TEC**
Q/FAX: **#233**

PROBLEM:
When starting up your computer you see the following message:

QEMM386: Disabling StealthROM because QEMM could not locate the ROM handler for INT XX"

POSSIBLE CAUSES:
A) You are loading a driver before QEMM which is grabbing interrupt XX.
B) A ROM is loading a handler for interrupt XX into RAM.
C) You are using a computer which was upgraded to an 80386 with an add-in board, such as the Intel "Inboard PC."

SOLUTIONS:
A) Load the driver in question after QEMM. If it must be loaded before QEMM, load HOOKROM.SYS before you load this driver.

During installation of QEMM, HOOKROM.SYS is installed in the QEMM directory. Assuming that QEMM is installed in a directory called QEMM on your "C" drive, the new line would look like this:

DEVICE=C:\QEMM\HOOKROM.SYS

Though it is usually best to load device drivers after QEMM, some drivers (like the ones that manage some 80386 conversion hardware) must be loaded before QEMM. These drivers can obscure information that QEMM needs to enable the StealthROM feature.

Placed before QEMM386.SYS in the CONFIG.SYS, HOOKROM will gather the necessary information for QEMM386.SYS and prevent this special driver from interfering with the StealthROM process.

B) Add the parameter "XSTI=XX" (where "XX" is the number of the interrupt reported in the message) to the QEMM386.SYS line of the CONFIG.SYS, then add an appropriate exclude statement to keep QEMM from mapping over the portion of the address space where the ROM handler for interrupt XX resides. (See the section "HOW DO I FIND THE APPROPRIATE EXCLUDE?" later in this document.)

It may also be possible to reconfigure your system in such a way that the ROM no longer redirects an interrupt into RAM. This is the case with the Invisible Network. (See "KNOWN USES FOR XSTI" near the end of this technical bulletin.) It is also possible that a program you are trying to run, or even your operating system, wants to have a particular interrupt remain unStealthed. XSTI, with the appropriate exclude, is necessary to get your program, or operating system, working with StealthROM.

C) Add the following parameters to the QEMM device line in your CONFIG.SYS file: XSTI=70 XSTI=74 XSTI=75 XSTI=76

A typical QEMM line would look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=70 XSTI=74 XSTI=75 XSTI=76 [all on the same line]

HOW DO I FIND THE "APPROPRIATE EXCLUDE?"
You find the appropriate exclude by excluding all the address space occupied by ROMs, using the parameter FSTC, and doing an Analysis. The first thing you need to do is locate all your ROMs. You can do this by looking at the First Meg/Overview screen of Manifest. Those with non-Microchannel machines and VGA video typically have a system ROM at F000-FFFF and a video ROM at C000-C7FF. Those with PS/2s or other Microchannel machines typically have one ROM at E000-FFFF. Add-on devices, such as some disk controller cards and network cards, may also have ROMs, which you must exclude as well.

A typical QEMM line for a non-Microchannel machine is:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=XX X=F000-FFFF X=C000- C7FF FSTC [all on the same line]

On a PS/2 or most Microchannel machines, the line will look like this:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=XX X=E000-FFFF FSTC

In the above examples, XX is replaced with the interrupt reported in the QEMM error message.

Reboot your computer with this CONFIG.SYS. StealthROM should work this time. Use your computer for a while, then look at the QEMM/Analysis screen of Manifest. You will see a chart that looks something like this:

```
          n=0123 4567 89AB CDEF
0n00 OOOO OOOO OOOO OOOO
1n00 OOOO OOOO OOOO OOOO
2n00 OOOO OOOO OOOO OOOO
3n00 OOOO OOOO OOOO OOOO
4n00 OOOO OOOO OOOO OOOO
5n00 OOOO OOOO OOOO OOOO
6n00 OOOO OOOO OOOO OOOO
7n00 OOOO OOOO OOOO OOOO
8n00 OOOO OOOO OOOO OOOO
9n00 OOOO OOOO OOOO OOOO
An00 OOOO OOOO OOOO OOOO
Bn00 OOOO OOOO OOOO OOOO
Cn00  II  I  I  I  III  OOOO OOOO
Dn00 OOOO OOOO OOOO OOOO
En00 OOOO OOOO OOOO OOOO
Fn00  I  I  I  II  II  I  OOII  I  III O
```

Consulting the ANALYSIS section of your Manifest or QEMM manual, you will read that an "I" indicates a portion of the address space that HAS NOT been accessed and an "O" indicates a portion of the address space that HAS been accessed. You must exclude that portion of the address space in the eXcluded ROMs where you now see "O"s.

In this example (which presumes that the ROMs were located from C000-C7FF and F000-FFFF), the appropriate exclude is "X=F800- F9FF", an 8K portion of the address space. This is the portion of the address space where the ROM handler for the interrupt XX resides. Our QEMM line, with appropriate excludes, would read as follows:

DEVICE=C:\QEMM\QEMM386.SYS RAM ST:M XSTI=XX X=F800-F9FF

PLEASE NOTE: The FSTC parameter is used only during this analysis process and should be removed afterward. Because the last 64 bytes of the First Meg address space (in FFFC-FFFF) is still addressed

directly with StealthROM, the last 4K piece of the QEMM/Analysis screen will always have an "O" in it, whether an exclude is appropriate or not.

ALSO NOTE: This procedure IS NOT used to find INCLUDES in portions of the address space NOT occupied by Stealthed ROMs. If you wish to experiment with INCLUDES (in order to gain additional High RAM) you must perform a complete analysis as described in the ANALYSIS section of the QEMM or Manifest manual.

WHAT IF THERE ARE NO "O"S?
This would mean the ROM handler for interrupt XX has been replaced by a new interrupt handler, and the one in the ROM is not being accessed at all. No exclude is necessary in this case.

KNOWN USES FOR XSTI:

INVISIBLE NETWORK

MS-DOS 5 ON SOME ZENITH MACHINES

VIDEO ACCELERATOR DRIVERS (such as SPEED_UP.SYS, RAMBIOS.SYS, FASTBIOS.SYS.)

If you are using any of these products, please refer to **PRODUCTS.TEC** for additional information.

All you need to know to use the XSTI parameter is contained above. A long, highly technical explanation of the above issues can be found in the unabridged version of this technote which is available through our standard support channels.

A bus-mastering hard drive does its own direct memory access (DMA) without going through the PC's processor or its DMA controller. The most common bus-mastering hard drives are SCSI drives. Because bus-mastering drive controllers transfer information without going through the PC's processor, they circumvent QEMM's memory mapping which works at the processor level.

The frame method leaves the system, video, and disk ROMs in place. QEMM places the EMS page frame so that it lies on top of a ROM's address space. When the ROM at the location of the page frame is needed, QEMM saves the current contents of the page frame and restores the ROM to its original location. The ROM code then executes normally. When the ROM routine is finished, QEMM restores the previous contents of the page frame. The frame method typically provides 48K-64K of extra High RAM and is provided for systems that are incompatible with the mapping method.

StealthROM's mapping method maps system, video, and disk ROMs and any other Stealthable ROMs out of the first megabyte of memory. When the system needs the ROM, QEMM maps the appropriate ROM code into the EMS page frame. The ROM code then has a valid real mode address at which it can execute normally. When the ROM routine is finished, QEMM restores the previous contents of the page frame. This mapping method typically provides 83K-115K of extra High RAM. If your system is not compatible with the mapping method, try the frame method.

Multiple Configurations

QEMM Setup has detected that your CONFIG.SYS file contains multiple configuration paths. Because the QEMM386.SYS device driver may be in more than one of these paths, you need to tell QEMM Setup which path you want to modify, in the event that the changes you specify will alter the QEMM386.SYS device line.

To select an existing configuration path:

Choose the configuraton you want, then select **Continue**.

To create a new configuration path:

Select highlight one of the existing paths and select **Create a new path from the selected existing path**. Then type a unique name for the new path in the field below. The name can be up to 32 characters long and can consist of more than one word. When you choose this option, QEMM Setup will add the new path to your CONFIG.SYS file. See your DOS manual for information on how to modify your CONFIGF.SYS and AUTOEXEC.BAT files for multiple configurations.

If you create a new configuration path, you should run Optimize and select the new configuration path. You will be prompted to run Optimize when you exit QEMM Setup.

Certain PCs devote 384K of RAM to shadow memory. Shadow memory is a hardware feature for speeding up the execution of ROM code by copying that code from ROMs to faster RAM.

Top memory is a kind of memory found on certain systems--notably, some Compaqs and some machines with Micronics motherboards. Top memory is used to speed up ROMs and also used by some pieces of system software.