INFORMATION TECHNOLOGY DIVISION

Memory Management

AICPA

American Institute of Certified Public Accountants

TECHNOLOGY BULLETIN

Notice to Readers

This technology bulletin is the first in a series of bulletins that provide accountants with information about a particular technology. These bulletins are issued by the AICPA Information Technology Division for the benefit of Information Technology Section Members. This bulletin does not establish standards or preferred practice; it represents the opinion of the author and does not necessarily reflect the policies of the AICPA or the Information Technology Division.

The Information Technology Division expresses its appreciation to the author of this technology bulletin, Liz O'Dell. She is employed by Crowe, Chizek and Company in South Bend, Indiana, as a manager of firmwide microcomputer operations, supporting both hardware and software applications. Liz is an Indiana University graduate with an associate's degree in computer information systems and a bachelor's degree in business management.

Various members of the 1992–1993 AICPA Information Technology Executive Committee were involved in the preparation of this technology bulletin. The members of the committee are listed below:

INFORMATION TECHNOLOGY DIVISION

Memory Management

AICPA

American Institute of Certified Public Accountants

TECHNOLOGY BULLETIN

1 2 3 4 5 6 7 8 9 0 IT 9 9 8 7 6 5 4 3

Library of Congress Cataloging-in-Publication Data
Memory management / Information Technology Division.
p. cm.
ISBN 0–87051–143–2
1. Memory management (Computer science) I. American Institute of Certified Public Accountants. Information Technology Division.
QA76.9.M45M46 1993 93–39671
004.5'3—dc20 CIP

Table of Contents

Chapter 1. Understanding Memory

Introduction

Today, manufacturers are cramming more tiny transistors on each silicon wafer, making memory an inexpensive commodity — and just in time, for applications are demanding more and more memory, forcing the average PC user to become more knowledgeable about computer memory and how to optimize it. It is becoming a major challenge to take full advantage of the computer's resources while balancing speed and maintaining reliability.

Running out of memory is becoming a common experience. The more complicated the task, the more memory is required. Applications such as graphic interfaces, graphics and presentation programs, and databases require a great deal of memory to operate. At the same time, personal computers are driving an increasing array of peripherals, such as CD-ROM readers, and loading drivers necessary for networking and pointing devices. Additional memory is required for loading memory-resident programs (TSRs) such as virus protection programs.

Memory management consists of a series of utilities and techniques that help squeeze the greatest use out of the memory available. One analogy for memory management is a company that is running out of office space. Instead of purchasing a larger office, the company can reassign and rearrange, making better use of the current space. For example, it could redesign the storage area on the second floor and move some of the staff into that area, which would give the first floor occupants the necessary space for better performance.

Memory management used to be the sole responsibility of system and network managers. Not any more. The first IBM personal computer possessed only 16K of memory. It would not be unusual nowadays to find 16Mb — 1000 times as much — in the computer sitting on your desk. But with this power comes complexity and the need for average users to understand and learn about memory. Unfortunately, it is difficult to understand memory management without basic knowledge of the various aspects of memory. With this in mind, let's first examine the difference between ROM and RAM, the existing processor modes, and define two very important files, the CONFIG.SYS and AUTOEXEC.BAT.

ROM and RAM

Actually, we are starting with the tough stuff — stay with me, it gets much easier. There are two physical types of memory found in a computer, ROM and RAM. ROM (*read-only memory*) stores information from the manufacturer. It consists of permanent instructions, such as the BIOS routines, that cannot be overwritten. Some adapter cards also contain ROM modules that supplement the system ROM routines.

New innovations in the desktop PC industry have given rise to nonpermanent kinds of ROM where BIOS can be updated without replacing the memory chip. Examples of these are EPROM (*erasable programmable read-only memory*), EEPROM (*electrically erasable programmable read-only memory*), and flash memory. One hears these variations of ROM being discussed, but they remain fundamentally inaccessible areas of memory.

RAM (*random-access memory*) is memory used to store data and program information. It can be read from and written to. The most common type, DRAM (*dynamic RAM*), consists of millions of capacitors that represent the binary 0s and 1s. DRAM must be refreshed periodically for the cells to hold their charge. A penalty in performance is paid while the cells are being recharged, but DRAM is used by most manufacturers to build affordable PCs. This is the physical type of memory that can be optimized through memory management.

There are a number of other types of RAM. SRAM (*static RAM*) does not require recharging and is approximately four times faster than DRAM. The cost of SRAM prohibits its use as the standard RAM in most PCs, but is being used for RAM caching in 386 and 486 computers. Another special type of RAM used to improve a PC's performance is VRAM (*video RAM*). VRAM is dual-ported and allows two sources to access it at the same time. While the microprocessor writes to VRAM, the video display circuitry can read VRAM and produce a character on the screen. Most new computers will offer both SRAM and VRAM.

Processor Modes

Three processor modes exist for personal computers: real, protected, and virtual. Each processor mode consists of a different set of processing instructions. An application must be able to read the set of instructions offered by that processor mode or it is not compatible with that mode. The processor mode is also dependent on the processor type. An 8086/8088 processor recognizes only real mode. The DOS operating system and most DOS applications are written for real mode.

The 80286 processor can operate in both real mode and protected mode. The first megabyte of memory is accessed in real mode and the remaining memory can be accessed in protected mode with the help of popular utilities.

The 80386 and 80486 processors add the capability of virtual mode. Virtual mode allows the system to address memory in the upper regions as if it were real mode. This allows DOS applications to access up to 4Gb of memory (1Gb = 1024 Mb).

The CONFIG.SYS and AUTOEXEC.BAT Files

The two most important system files that you have control over are the CONFIG.SYS and AUTOEXEC.BAT. You will read here how these play a central role in optimizing your system. When you boot your computer, DOS looks at the two hidden system BIOS files and the COMMAND.COM file. Then DOS analyzes your CONFIG.SYS file. Like the COMMAND.COM, the CONFIG.SYS must be in your root directory. The primary purpose of the config file is to load device drivers and reserve space in the system's memory for information processing. Since DOS accesses this file only when you start your computer, changes to this file will not take place until you reboot. Most of the commands in your CONFIG.SYS file cannot be typed at a DOS prompt.

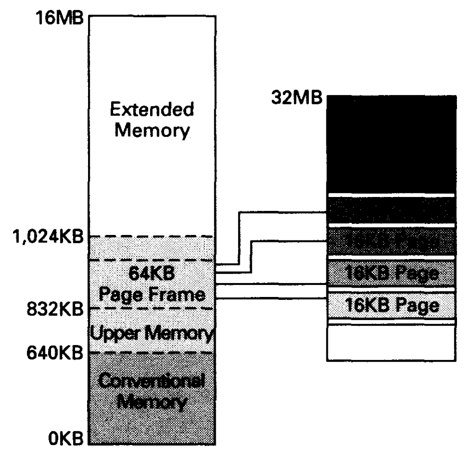
DOS allows you to define a start-up procedure by means of the AUTOEXEC.BAT file. Any of the commands in the autoexec file can be entered at the DOS prompt and you can reexecute the AUTOEXEC.BAT by entering autoexec. The autoexec file is a batch file, but special in the fact that DOS runs this batch file automatically each time the computer is booted. You might include programs such as the date, time, path, menu, and TSRs in your AUTOEXEC.BAT. TSRs are *terminate-and-stay-resident programs* — meaning they are always loaded in memory.

Chapter 2. Memory Types in the DOS Environment

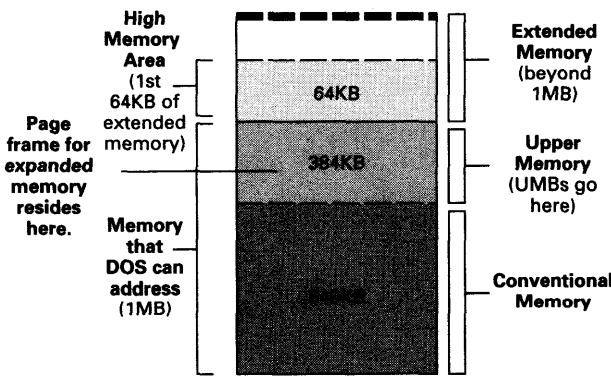
There are five main types of memory in the DOS environment: conventional, upper, extended, high, and expanded. The first four types differ mainly in the address location or specific range of the memory spectrum that they occupy. Expanded memory, however, lies outside the normal addressable space and must be accessed with a software memory manager. The delicate balance among these five types of memory can be achieved only if you understand what it is you are trying to utilize.

The schematic drawings presented below may be useful in conceptualizing the different types of memory. You can refer to these drawings as we discuss address locations.

Expanded Memory Page Frame*



Five DOS Memory Types*


High Memory Area (1st 64KB of extended memory)
Page frame for expanded memory resides here.
Memory that DOS can address (1MB)
64KB
384KB
640KB
Extended Memory (beyond 1MB)
Upper Memory (UMBs go here)
Conventional Memory

*Reprinted with permission from *PC Novice* (premiere edition, issue no. P3), Peed Corp., Lincoln, NB 68521.

Conventional Memory

Conventional memory, also referred to as base memory, is the region from 0 to 640K. MS-DOS uses this memory for operating and allocates the remaining portion to device drivers and applications. The 640K boundary was arbitrarily selected by IBM designers in the early 1980s. The original 8088 processor could address up to 1024K of data, but designers of DOS decided to allocate the first 640K to base memory and reserve the remaining 384K. (This seemed like an enormous amount of memory at the time. Remember, the first computer had only 16K, so the designers allowed for a forty-fold increase.)

To understand address locations, think of a computer's memory as a street with specific house addresses. The street was designed with a maximum of 640 house numbers. But, as technology advanced, houses became cheaper and in greater demand. Suddenly, we realize that people want to build 1000 houses on a street that is limited to 640 addresses. What do we do? We could start over with a completely new addressing scheme, such as OS/2 or Windows NT, or we can try to fix what we have. Since our beloved DOS applications are written for DOS (and DOS only), our efforts are centered around overcoming this limitation.

Upper Memory

The *upper memory area* (UMA) is located between 640K and 1024K. This area was originally reserved for the system hardware, such as the BIOS, video buffers, and adapter ROMs. Most computer systems do not consider upper memory part of total memory because it is not available for data storage. The parts of the upper memory that are not used by the system can be converted to *upper memory blocks* (UMBs). Since the need to optimize memory has increased, utilities have been created to allow the unused upper memory blocks to be accessed.

The EMM386.EXE driver that comes with DOS 5.0 can convert the unused space in the UMBs to usable RAM on 386 and 486 PCs. The area can then be used to load device drivers and TSRs, freeing the base memory for the execution of applications. The amount of UMBs available depends on the amount of memory required by the system and the expanded memory page frame, but can be as much as 128K.

This area is not *high memory,* although it is incorrectly referred to as this by many people. Read on to see where the high memory area is really located.

Extended Memory

The *extended memory area* includes all memory beyond the 1024K (1Mb) mark. Extended memory is not available on an 8086/8088 processor because of the limitation in the number of address lines. The 286 processor can access memory up to 16Mb and the 386 or 486 processors can access up to 4Gb of extended memory. The system accesses extended memory in either protected or virtual mode.

The most common way to access extended memory is by installing an extended-memory manager. The HIMEM driver is included with DOS and conforms to the *eXtended Memory Specification* (XMS) standard. HIMEM directs traffic above 640K, including the upper memory blocks.

Some high-end applications use built-in DOS extenders and can access extended memory in protected mode without the use of a memory manager. The DOS extender is built into the application and is not controlled by the user. There are also a few applications that access extended memory directly through an interrupt.

High Memory

The *high memory area* (HMA) is a special 64K area of extended memory located between 1024K and 1088K. Unlike the rest of extended memory, HMA can be accessed in real mode. This makes it the ideal location for parts of the operating system, thus freeing conventional memory. The high memory area is often confused with the upper memory blocks because of the terminology. When the LOADHIGH command is used, the TSR is loaded into the UMBs, not the high memory area.

In addition to part of the operating system, we will load our disk buffers into this area. There are some special utilities that allow access to the high memory area. But, after DOS and the buffers load into the HMA, there isn't any free area left to be concerned with.

Expanded Memory

Expanded memory is memory in excess of 640K that does not physically reside in the system memory address space. It was the first attempt by the PC industry to overcome the 640K handicap. On early PCs, expanded memory was physically located on an expanded-memory board, such as an Above-Board. A wide variety of applications were designed to use expanded memory, and the *Expanded Memory Specification* (EMS) is still strongly supported by software developers today.

With an 8086 processor, EMS could be located only on an expanded-memory board. With the 286, 386, and 486 processors, extended memory could be mapped as EMS. An expanded-memory manager conforming to the Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) must be installed for an application to access expanded memory on the system board. The LIM EMS version 4.0 will support up to 32Mb of expanded memory.

EMS is divided into 16K segments called *EMS logical pages.* When an application requires information stored in EMS, the expanded-memory manager maps the appropriate page into a physical 64K page frame located in the upper memory area. Mapping memory through EMS page frames makes the memory accessible to DOS applications.

Expanded memory can be used only by applications that are written to include the page swapping calls, such as Lotus 1-2-3. Because of the page swapping, expanded memory is slower than extended memory. If an application offers the option to use either expanded or extended memory, extended memory will render better performance. DOS 5.0 makes only limited use of expanded memory. The data buffered by FASTOPEN can be stored in EMS by adding the /X switch. RAMDrive and SMARTDrive can also use EMS.

Chapter 3. Determining Available Memory

The DOS MEM command will report your current memory statistics and help determine the effects that changes to the CONFIG.SYS and AUTOEXEC.BAT files will have on optimizing memory. The MEM command is available using DOS versions 4.x and 5.x. With earlier versions of DOS, the CHKDSK command can be used to report available conventional memory only.

You can gain more detailed information about memory by typing MEM/C at the DOS prompt. (Add the pipe command MORE if the display scrolls past one screen: MEM/C ¦ MORE.) A sample printout appears on page 7. The listing is broken into three sections. The first section shows the status of conventional memory. Each program or device drive loaded in conventional memory is listed with its corresponding size. Unused areas will be listed as "free" and the total of free areas will be displayed. Conventional memory is where most DOS applications run. The "free" conventional will need to be at a minimum for the application to load and at a maximum for the greatest performance.

The second section details how the upper memory blocks are being used. The total 384K of the upper memory region is not converted to upper memory blocks and the area occupied by "system" is hardware-dependent. The maximum you have to work with is 128K. Your objective is to load as much as possible into this area — without any conflicts. In our example, four device drivers have been loaded into the UMBs, leaving only 1248 bytes free.

The last section is a summary of each type of memory. The "largest executable program size" is the memory that is available for running your DOS applications. This number will correspond closely to the free conventional memory. If you have an expanded-memory manager loaded, the total and free EMS memory will be listed next.

The last four lines will contain the status of extended memory. The first line reports the total bytes of contiguous extended memory. The second line reports the bytes available, but don't be alarmed if this is 0. If you are running the HIMEM driver in your CONFIG.SYS, HIMEM is controlling the extended memory and the amount that is available is listed in the third line as "available XMS memory." The last line will show if you have MS-DOS loaded into the high memory area.

The MEM/P command will produce a more detailed map of conventional memory, but the address locations and program sizes are in hexadecimals, making the listing harder to decipher by the average user.

Sample Printout of the MEM/C Command

```
Conventional Memory:

          Name          Size in Decimal          Size in Hex
          MSDOS           15472   ( 15.1K)           3C70
          HIMEM            1184   (  1.2K)            4A0
          EMM386           8400   (  8.2K)           20D0
          STACKER         41792   ( 40.8K)           A340
          COMMAND          2880   (  2.8K)            B40
          MOUSE           11024   ( 10.8K)           2B10
          APPEND           9024   (  8.8K)           2340
          FREE               64   (  0.1K)             40
          FREE               64   (  0.1K)             40
          FREE              144   (  0.1K)             90
          FREE           565024   (551.8K)          89F20

          Total FREE:    565296   (552.0K)

Upper Memory:

          Name          Size in Decimal          Size in Hex
          SYSTEM         233472   (228.0K)          39000
          ATDOSXL          9520   (  9.3K)           2530
          ANSI             4192   (  4.1K)           1060
          SETVER            400   (  0.4K)            190
          SMARTDRV        13200   ( 12.9K)           3390
          FREE             1248   (  1.2K)            4E0

          Total FREE:      1248   (  1.2K)

Total bytes available to programs (Conventional + Upper):  566544 (553.3K)
Largest executable program size:  564800 (551.6K)
Largest available upper memory block:  1248 ( 1.2K)

1441792 bytes total EMS memory
1048576 bytes free EMS memory

1441792 bytes total contiguous extended memory
0 bytes available contiguous extended memory
4096 bytes available XMS memory
MS-DOS resident in High Memory Area
```

Chapter 4. Optimizing Memory

The objective in optimizing memory is to provide your application program with as much space as possible. Executing the following seven steps will increase the conventional memory available and provide maximum system performance.

Step One: Install HIMEM

The most important step in optimizing your configuration is to install the HIMEM extended-memory manager. It does not have to be the first line in your CONFIG.SYS, but it must come before any other commands that load device drivers. The HIMEM utility provides a central role in DOS's actions with memory and requires only 1K of space. It allows MS-DOS programs access to extended memory under XMS and the high memory area. If HIMEM is installed in conjunction with an expanded-memory manager, upper memory blocks are available.

```
DEVICE=C:\DOS\HIMEM.SYS
```

(Note: some OEMs, such as Compaq, call this driver HIMEM.EXE.)

Step Two: Load DOS High

The simplest and most effective way to increase your conventional memory is to load DOS into the high memory area. This can be done on any 286 or higher processor with at least 64K of extended memory. Loading DOS into the HMA can increase your available conventional memory as much as 60K. In the MEM/C example printout, if DOS had not been loaded high, the MSDOS and COMMAND entries would have increased to a total of 73,792 bytes. The difference — 55K — of the operating system was loaded in the high memory area.

To load MS-DOS into the high memory area, add the following line to your CONFIG.SYS file:

```
DOS=HIGH,UMB
```

The UMB switch dynamically links the base memory with the upper memory area and should be included if you are using DEVICEHIGH or LOADHIGH commands. This statement can appear anywhere in the CONFIG.SYS file.

Step Three: Evaluating Buffers

Next let's discuss the BUFFERS= line in the CONFIG.SYS file. This setting determines the space available for buffering I/O from your disk drives. Each buffer over the minimum of 3 requires 532 bytes of memory. Increasing buffers past a certain value will use more memory without increasing the performance of your system.

When you load DOS high, the disk buffers will automatically be redirected to the HMA. Typically, there is room for 48 buffers in the HMA. If there is not enough room for DOS to load the buffers in the HMA, all of the disk buffers will be loaded in conventional memory. Experimenting with the buffers command will also show you that there is a decrease in available conventional memory between BUFFERS=44 and BUFFERS=45. At this breaking point, DOS will move part of the COMMAND.COM back to conventional memory. To confuse the issue even more, the BUFFERS setting will directly impact disk performance. To find the optimal BUFFERS setting follow these rules:

- Restrict BUFFERS to 44 or less when DOS is loaded in HMA.
- If you do not use a disk cache, set your buffers according to your hard disk size —

Hard Disk Size	*Buffers*
<40Mb	20
40–79Mb	30
80–120Mb	40

- If you use a caching program, such as SMARTDrive, use a BUFFERS setting from 10 to 20.

Step Four: Consider Lowering the Files Setting

The FILES= statement is the next most common configuration statement that reduces the amount of conventional memory by 59 bytes for each file. The FILES setting determines the size of the system file table used to store information about open files. Even though the FILES statement is usually blocked with the BUFFERS statement, the directives have nothing in common. The FILES directive cannot be loaded high without a third-party utility, such as UMBFILES. To maximize available memory, the FILES should be kept at the minimum necessary to run your applications. However, reducing the FILES= statement to a number that is too low can cause an application to crash without warning.

Step Five: Load Device Drivers and TSRs High

Loading device drivers and TSRs into the upper memory blocks is next on the list for optimizing conventional memory. First you need to configure your system to make the UMBs available: run HIMEM.SYS, install EMM386.EXE, and set DOS=UMB.

The HIMEM utility is required to ensure that one program does not use the area already occupied by another program. The EMM386 driver converts the empty areas between 640K and 1Mb to UMBs and the UMB directive adds the upper memory blocks to DOS's list of system resources. The entries in a typical CONFIG.SYS would look like this:

```
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\EMM386.EXE NOEMS
DOS=HIGH,UMB
```

After adding these commands to your CONFIG.SYS, the MEM/C command will report the amount of available upper memory blocks.

The DEVICEHIGH command is provided with DOS to load device drivers into the UMBs and can be used only within your CONFIG.SYS file. The DEVICEHIGH and DEVICE commands have the same syntax and are interchangeable as long as there are free UMBs. The DEVICEHIGH command should always follow and cannot be used with the HIMEM and EMM386 drivers. Any other DOS device driver can be load into the UMBs. There are some non-DOS drivers that are not UMB compatible.

If there is insufficient memory to load a driver into the UMBs, DOS will load the driver into conventional memory. DOS determines the memory needed for the device driver by examining the disk space used by the driver. However, a close look at our memory printout will show that the driver may actually take less or more space when loaded into memory. Some drivers also require extra memory when executed. If you know that a driver will require additional space after loading, you can use the SIZE= switch to specify a minimum number of bytes.

The LOADHIGH command can be used in the AUTOEXEC.BAT file to load TSR programs into the upper memory blocks. The order in which the TSRs are loaded is important. As a rule, LOADHIGH will load a TSR in the largest available UMB space. It then follows that your largest TSR should be loaded first, if possible. Some programs, such as those used for networking, must be loaded in a predefined order.

Loading TSRs into the UMBs is somewhat of a trial-and-error procedure. As with the device drivers, some TSRs require more space at execution time than when they were loaded, and others simply are not UMB compatible. All of the TSRs shipped with DOS can be loaded high.

Step Six: Add a STACKS Command

A small amount of memory can be regained by adding a STACKS command to reduce the memory area used to handle hardware interrupts. The default for 8088/8086 based computers is STACKS=0,0. The default for all other computers is STACKS=9,128. You can conserve 1K of memory by setting the stacks to zero on all but a few systems. If your PC crashes with an error message of "Internal stack overflow," you will know that your PC requires the overflow buffering.

Step Seven: Adjust the LASTDRIVE Setting

Each logical drive letter uses 88 bytes. By setting the LASTDRIVE equal to the minimum number of drive letters required, you can save memory. DOS will automatically set aside five logical drives. If your system does not use drive "E," for example, then you can save 88 bytes by adding this statement to your config file:

```
LASTDRIVE=D
```

Watch out for some applications that set the LASTDRIVE equal to Z even though twenty-six drive letters are not required.

These seven steps do not exhaust the possibilities for optimizing your computer's memory. They are, however, the most common and most effective, both in terms of the memory gained and the amount of effort required by the user to implement them.

Chapter 5. Sample Configurations

After all this discussion, maybe some sample configuration files will be helpful.

If you have a 386- or 486-based computer system with 2Mb of memory and a 40Mb hard drive installed, your CONFIG.SYS might look like this:

```
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\EMM386.EXE NOEMS
DOS=HIGH,UMB
SHELL=C:\DOS\COMMAND.COM C:\DOS /P /E:512
FILES=40
BUFFERS=20
STACKS=0,0
LASTDRIVE=D
```

If we increase our 386 or 486 system to 4Mb of memory and install a 120Mb hard drive, the configuration might be set up like this:

```
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\EMM386.EXE RAM 1024 FRAME=E000
DOS=HIGH,UMB
DEVICEHIGH=C:\DOS\SMARTDRV.SYS 1024
SHELL=C:\DOS\COMMAND.COM C:\DOS /P /E:512
FILES=40
BUFFERS=10 (Using Smartdrv)
STACKS=0,0
```

Chapter 6. Additional Memory Management Techniques

Disk Caches

A disk cache can be one of the best ways to make use of the extra memory in your PC by speeding up the operations of your hard drive. Requested data is buffered in RAM and can be accessed in a matter of nanoseconds rather than the milliseconds it takes to read from a hard disk. Caching can improve hard disk performance tenfold plus reduce wear on the disk.

SMARTDrive is shipped with DOS 5 or Windows and can be used to create a cache in either expanded or extended memory. Extended memory is recommended. The cache size should be set as large as possible but within the range of 256K to 2048K. A cache larger than 2048K will not improve performance significantly. By loading the SMARTDrive driver high, you can reserve 15K–20K in conventional memory.

```
DEVICEHIGH=C:\DOS\SMARTDRV.SYS 1024
```

Smartdrive 4.0, when shipped with Windows 3.1, has a feature called *double buffering* that should be enabled if the hard disk controller performs bus mastering (accessing memory directly without using the standard DMA channels).

Some other popular disk caching programs, such as Multisoft's Super PC-Kwik Disk Accelerator, will also cache floppy disk data.

RAM Disk

A RAM drive is very similar to a physical drive, except it is created in memory. The data stored in the RAM drive will be lost each time the computer is turned off. A RAM drive will improve performance while running disk-intensive programs, but the SMARTDrive disk caching program will increase speed more than a RAM drive. Therefore, it is suggested that you set up a RAM drive only if you have additional memory after setting a disk cache.

DOS 5 comes with a RAMDRIVE device driver for creating a RAM drive. The driver consumes 1.4K of conventional memory and can be loaded high with the DEVICEHIGH command:

```
DEVICEHIGH=C:\DOS\RAMDRIVE.SYS 1024 /e
```

The RAM drive can be created in conventional, extended (/e), or expanded memory (/a) and will assume the next available drive letter.

The most common use of a RAM drive is to store temporary files. You can use it to set the TEMP environmental variable to direct applications to store their temporary files to the RAM drive. Because of the DOS limitation of files created at the root directory, the TEMP variable should be set to a subdirectory. For example, you might include these lines in your AUTOEXEC.BAT:

```
MD E:\TEMP
SET TEMP=E:\TEMP
```

Chapter 7. Avoiding Problems With Memory Management Programs

Customizing your computer system's memory is a trial-and-error procedure. Not all hardware or software is compatible with a standard system configuration. Before adjusting your configuration, save a copy of your CONFIG.SYS and AUTOEXEC.BAT files. Then create a bootable diskette that can be used if your system locks up.

HIMEM

There are some conflicts with the HIMEM utility that you may need to be aware of. If HIMEM reports that it is "unable to control the A20 line," use the /MACHINE:n switch. A20 is the address line that controls access to the high memory area. For example, on a Toshiba use /MACHINE:7 or on a Zenith with ZBIOS use /MACHINE:10.

After loading DOS in HMA, an application may try to load unsuccessfully into the first 64K of memory. You may receive the error message "Packed File is Corrupt." This is due to a bug in EXEPACK program. DOS has supplied a LOADFIX program to load the program above the 64K line:

```
LOADFIX program_name
```

Another problem: applications that do not use the HIMEM utility for access to extended memory report that there is no extended memory available. The application is trying to access extended memory directly through interrupt 15h. With the HIMEM driver loaded, all of the extended memory is being converted to XMS. You can keep HIMEM from controlling and converting all the memory to XMS by adding the switch INT15=nnnn.

LOADHIGH

If a program will not load high even though the total upper memory is sufficient, chances are there is not enough contiguous memory available. This is the time to experiment with the load order of your drivers and TSRs. You can also remove the EMS page frame, if you are not using expanded memory. With the help of third-party memory managers, the BIOS ROM can also be relocated to give more contiguous space. Also with the help of third-party utilities, TSRs can be loaded and then unloaded as needed.

EMM386

A conflict can occur when running EMM386.EXE and an application using DOS extenders. Both EMM386 and the application must comply with the Virtual Control Program Interface (VCPI) specification. EMM386 supports the VCPI standard and is compatible with DOS extenders only if the program is supplying expanded services. Do not use the NOEMS switch if you run applications with DOS extenders. If the application does not support the VCPI standard, the EMM386 driver must be removed from the system configuration.

EMM386 will not create UMBs in the E000h segment since BIOS ROM could be stored in this area. On most systems (the IBM PS/2 and IBM/VP are two exceptions), this memory segment is free and can be used for the page frame:

```
DEVICE=C:\DOS\EMM386.EXE RAM 2048 FRAME=E000
```

Part of the video segment, A000h–B000h, can also be used on CGA, EGA, MDA, or Hercules adapters. A VGA adapter is capable of using all this address space, but if you have a color monitor you may be able to use B000–B7FF. To include an area as useable RAM add the I= switch:

```
DEVICE=C:\DOS\EMM386.EXE RAM I=B000–B7FF
```

EMM386 does not search for memory conflicts. If your PC locks up when loading the EMM386, use the X= switch to exclude the upper memory region, then cut the excluded range in half until you have narrowed down the memory conflict. Some memory managers fail to detect RAM on a network adapter. You can use the exclude switch to protect the address range of the network adapter:

```
DEVICE=C:\DOS\EMM386.EXE RAM X=D000–DFFF
```

Some EMM386 switches available are:

[NOEMS, RAM, <blank>]	UMBs only, UMBs & expanded, expanded only
[memory]	Size for expanded memory
[FRAME=nnnn]	Default D000h
[H=handles]	Default 64, valid 2–255, handles in XMS

Chapter 8. Where Do I Go From Here?

There are plenty of resources available, if trial-and-error problem diagnostics have failed or you wish to become more knowledgeable of memory management.

Read. Read the Read.me files that came with the drivers or software that you are having problems with. Read the manuals. Yes, the manuals! There are also a wide variety of memory management books that can turn you into an expert.

Ask a friend or colleague. If you are a beginner in the world of DOS and PCs, friends and co-workers are eager to show off their great knowledge.

Check the Bulletin Board System (BBS). Most software and hardware manufacturers now have public bulletin boards with the latest bugs and fixes.

Call. Pick up the phone and call the technical support line. Believe me — they have already heard all the dumb questions. The tech line can really be helpful when you have exhausted all your own ideas.

There are very few jobs in the world today that are not affected by the PC. And, there is nothing more frustrating than a PC with memory conflicts. I hope this bulletin has enlightened you to some of the deep, dark secrets of RAM.

Glossary

AUTOEXEC.BAT

Special DOS batch file that is automatically executed whenever the computer is started or restarted. It must be stored in the root directory. It is used to load terminate-and-stay-resident (TSR) programs that stay in memory and "pop up" whenever you call them. It's also used to start an application, such as a menu program that launches a variety of applications, when the computer is turned on.

Batch File

File containing data that is processed or transmitted from beginning to end. File containing instructions that are executed one after the other.

Bulletin Board System (BBS)

Computer system used as an information source and message switching system for a particular interest group. Users dial into the BBS, review and leave messages for other users as well as communicate to other users on the system at the same time. BBSs are used to distribute shareware and may provide access (doors) to other application programs.

Buffer

Reserved segment of memory used to hold data while it is being processed. In a program, buffers are created to hold some amount of data from each of the files that will be read or written. A buffer may also be a small memory bank used for special purposes.

Buffer=

528-byte areas of RAM reserved for input and output (1–99). Default is usually 15, but this is often set to 20 or 30. The more buffers, the faster the I/O.

Bus Mastering

Bus design that allows add-in boards to process independently of the CPU and to be able to access the computer's memory and peripherals on their own.

Cache

Pronounced *cash.* A reserved section of memory used to improve performance. A disk cache is a reserved section of normal memory or additional memory on the disk controller board. When the disk is read, a large block of data is copied into the cache. If subsequent requests for data can be satisfied in the cache, a slower disk access is not required. If the cache is used for writing, data is queued up in memory and written to the disk in larger blocks. A memory cache is a high-speed memory bank between memory and the CPU. Blocks of instructions and data are copied into the cache and instruction execution and data updating are performed in the higher-speed memory.

Note: Definitions have been excerpted from the *Electronic Computer Glossary* by Alan Freedman with permission of The Computer Language Company Inc., Point Pleasant, PA (215) 297–8082.

CHKDSK

External command that reports free memory and disk space. To display memory and disk status, type:

```
chkdsk
```

Improperly closed files, caused by rebooting from a frozen application (for example), generate lost clusters, which are unidentifiable files. Usually, these are temporary files not worth recovering. To reclaim these lost clusters, run chkdsk with the /f switch. When you're asked "Convert lost chains to files?," answering Y for yes will convert lost clusters to FILE0000.CHK files, which you can examine. Answering N will remove them. For example:

```
chkdsk /f
```

To list recovered files, type:

```
dir *.chk
```

Important! Don't go to the DOS prompt (shell out) from within Windows or any other program and then run the chkdsk utility. You may get invalid results and possibly destroy data.

COMMAND.COM

Command interpreter that displays the DOS prompt and accepts and executes your typed-in commands. If a command interpreter other than COMMAND.COM is used, it is specified with the Shell command.

DOS loads COMMAND.COM from the disk at start-up. Part of COMMAND.COM is always resident in memory. The rest of it (the transient part) may be overwritten when a program is executed. When the program is done, the transient portion is reloaded into memory.

CONFIG.SYS

Configuration file that DOS looks for in the root directory upon start-up. It is used to load drivers and change system settings. Adding a new type of peripheral to the computer usually requires installing the driver program to make it operate. Example of a CONFIG.SYS file:

```
files=30
buffers=40
lastdrive=k
device=qemm386.sys
device=ansi.sys
device=mouse.sys
```

Conventional Memory

In a PC, the first megabyte of memory. The term may also refer only to the first 640K. The top 384K of the first megabyte is called *high DOS memory* or *upper memory area.*

Device

A program routine that links a peripheral device or internal function to the operating system. (Also called a *device driver.*) It contains the precise machine language necessary to activate all device functions and includes detailed knowledge of its characteristics, such as sectors per track or the number of pixels of screen resolution.

Basic drivers come with the operating system, and drivers are added when new peripheral devices are installed. For example, if you add a mouse or CD-ROM player to your personal computer, you have to install the appropriate driver so that the operating system knows how to handle it. In the DOS world, applications provide their own screen and printer drivers in order to provide complete control over the display and printing of a document.

Memory managers, RAM disks and disk caches are also activated by drivers.

Common Drivers	*Purpose*
ansi.sys	Screen and keyboard control
display.sys	Supports code-page switching
driver.sys	Identifies third and fourth floppy and allows copying from/to same drive
mouse.sys	Mouse driver
printer.sys	Code-page support for printers
himem.sys	Extended memory (XMS) manager
emm386.exe	386 EMS manager
qemm386.sys	Quarterdeck's 386 EMS manager
ramdrive.sys	RAM disk (extended or EMS memory)
smartdrv.sys	Disk cache (extended or EMS memory)

Double Buffering

Programming technique that uses two buffers to speed up a computer that can overlap I/O with processing. For example, data in one buffer is being processed while the next set of data is read into the second buffer.

DRAM

Dynamic RAM, the most common type of computer memory. It usually uses one transistor and one capacitor to represent a bit. The capacitors must be energized hundreds of times per second to maintain the charges. Unlike firmware chips (ROMs, PROMs, etc.), both major varieties of RAM (dynamic and static) lose their content when the power is turned off. Contrast with static RAM.

EEPROM

Electrically erasable programmable read-only memory, a memory chip that holds its content without power. It can be erased, either within the computer or externally, and usually requires more voltage for erasure than the common +5 volts used in logic circuits. It functions like nonvolatile RAM, but writing to EEPROM is much slower than writing to RAM. EEPROMs are used in devices that must keep data up-to-date without power. For example, a price list could be maintained in EEPROM chips in a point-of-sale terminal that is turned off at night. When prices change, the EEPROMs can be updated from a central computer during the day.

EMM386.EXE

Memory manager for 386s and up that comes with DOS 5.0 and Windows 3.0. It converts extended memory into EMS memory and also allows programs to be stored in the UMA (area between 640K and 1M). It is activated with a statement in the CONFIG.SYS file. The HIMEM.SYS driver must also be activated before EMM386.EXE:

```
device = himem.sys
device = emm386.exe
```

To provide access to the UMA, either the RAM or NOEMS parameters must be added:

```
device = emm386.exe ram      UMA and EMS
device = emm386.exe noems    UMA only
```

EMM386.EXE is an executable program, which can be run after it has been initialized in order to change settings.

EMS

Expanded Memory Specification. Technique for increasing memory in DOS PCs. EMS Version 4.0 allows DOS to work with up to 32MB of extra memory by bank switching 16K segments of EMS memory, known as the "page frame," into conventional memory.

In order to use EMS, the application is either written to use it directly (Lotus 1-2-3 Ver. 2.x, AutoCAD, etc.) or the application is run in an environment that uses it, such as DESQview.

In XTs and ATs, EMS is installed by plugging in an EMS memory board and adding an EMS driver. In 386s and up, EMS is created by expanded-memory-manager (EMM) software that turns extended memory into EMS.

EPROM

Erasable programmable ROM. Reusable PROM chip that holds its content until erased under ultraviolet light.

Expanded Memory

Memory in excess of 640K that does not physically reside in the system memory address space. It consists of two parts: an expanded-memory board and a program called an expanded-memory manager. A program designed to use expanded memory does not have direct access to the information in expanded memory; instead, expanded memory is divided into 16K segments called *pages.* When a program requests information that is in expanded memory, the manager maps or copies the appropriate page to an area called a *page frame,* from which the program gets the information.

Extended Memory

In Intel 286s and up, standard memory above 1Mb used for RAM disks, disk caches, and applications using DOS extenders. Windows also uses extended memory. Expanded memory and extended memory are not the same. Expanded memory can be installed in XT-class machines and up, whereas extended memory requires at least a 286. With 286s now the low-end CPUs, extended memory is increasingly being used with DOS extended applications and Windows 3.0.

FASTOPEN

External command starting in DOS 3.3 that reopens hard disk files quickly. If a drive is specified with Fastopen, the locations of the files opened are stored in memory. When opened again within the same session, their exact location is known.

Fastopen is put in the AUTOEXEC.BAT file with the number of files you want to hold in memory:

```
fastopen c:=50          hold 50 file names on C:
fastopen c:=50 d:=75    50 on C:, 75 on D:
```

DOS 4.0 can optionally keep track of all the file fragments. This second number is usually four times greater than the number of names:

```
fastopen c:=(50,200)
```

Caution! Fastopen cannot be used on network drives, and it can be used only once per computer session.

FILES=

Files open at one time (8–255). Default is 8, but this is often set to 20–40. Some applications open a lot of files.

Flash Memory

Memory chip that holds its content without power, but must be erased in bulk. Originally coined by Toshiba, the term comes from its ability to be erased "in a flash." Derived from EEPROMs, flash memory chips are less expensive and provide higher bit densities.

High Memory

The uppermost end of memory. In PCs, the area between 640K and 1M, or the 64K HMA area between 1024 and 1088K.

HIMEM.SYS

Extended memory (XMS) driver included with DOS 5.0 and Windows 3.0. HIMEM.SYS must be activated in CONFIG.SYS in order to load DOS into the HMA, to use extended memory for RAM disks and disk caches, and to use EMM386.EXE to turn extended memory into EMS memory.

Only one program can reside in the HMA at one time. To reserve it for the largest program, you can use the HMAMIN switch to specify the minimum size the program must be.

Interrupt

Signal that gets the attention of the CPU and is usually generated when I/O is required. For example, hardware interrupts are generated when a key is pressed or when the mouse is moved. Software interrupts are generated by a program requiring disk input or output.

An internal timer may continually interrupt the computer several times per second to keep the time of day current or for time-sharing purposes.

When an interrupt occurs, control is transferred to the operating system, which determines the action to be taken. Interrupts are prioritized; the higher the priority, the faster the interrupt will be serviced.

LASTDRIVE

Used in the CONFIG.SYS file, it specifies the maximum number of drives that can be accessed. The following line in CONFIG.SYS sets the last drive letter to M:

```
LASTDRIVE=m
```

LIM EMS

In 1984, Lotus, Intel, and Microsoft introduced EMS (LIM EMS), which allowed up to 8MB of EMS memory. By Version 3.2, it was widely supported, but limited to one 64K page frame (four 16K pages) only in the UMA (640K–1M region).

LOADFIX

External command in DOS 5.0 that loads a program beyond the first 64K of RAM, solving a problem with some earlier programs. With DOS 5.0, part of DOS can be loaded into high memory, thus freeing up more lower memory. Some programs cannot run in this lower RAM and generate a "packed file corrupt" error message. *Packed* refers to an older method for compressing EXE files, and this method had a bug in it that prevents it from running in lower RAM.

The following example loads the program ABC beyond the first 64K:

```
loadfix abc
```

LOADHIGH

Internal command in DOS 5.0 that loads a program into the UMA (upper memory area: 640K–1M) in 386s and up. It requires that the HIMEM.SYS and EMM386.EXE memory managers be loaded and the dos=umb command be present.

If you have programs that need expanded memory (EMS), the following three lines in CONFIG.SYS are required to use Loadhigh. The RAM parameter informs EMM386 to manage both the UMA and expanded memory:

```
device = \dos\himem.sys
device = \dos\emm386.exe ram
dos = umb
```

The following example uses the NOEMS parameter to inform EMM386 to manage only the UMA and not expanded memory:

```
device = \dos\himem.sys
device = \dos\emm386.exe noems
dos = umb
```

If the lines above are in your CONFIG.SYS file, you can load programs into upper memory. To load a device driver into upper memory, use the DEVICEHIGH command in CONFIG.SYS instead of DEVICE; for example:

```
DEVICE = ANSI.SYS          below 640K
DEVICEHIGH = ANSI.SYS      above 640K
```

MEM

External command starting with DOS 4.0 that displays the amount of memory currently used and free. Type:

```
mem
```

As of DOS 5.0, to display the contents of memory by program name and show the available free memory blocks (UMBs) in upper memory (UMA), type:

```
mem /c
```

OEM

Original equipment manufacturer, a manufacturer that sells equipment to a reseller. OEM is also used to refer to the reseller. OEM customers often purchase equipment and resell it under their own names. They may combine units from several vendors as well as add software. The terms OEM and VAR are often used synonymously.

OS/2

Single user, multitasking PC operating system for 286s and up. The 16-bit versions have been developed jointly by Microsoft and IBM. The 32-bit versions are developed independently.

OS/2 is an advanced operating system that is not confined to DOS's infamous 1Mb limit. Although new commands have been added, many OS/2 commands are the same as in DOS. The 16-bit versions can address 16Mb of RAM and 1Gb of virtual memory. OS/2 requires 4Mb of RAM (except for version 1.3), but is often found running in computers with 8Mb and more.

Protected Mode

In Intel 286s and up, an operational state that allows the computer to address all of memory. It also prevents one program from entering into the memory boundary of another, thus enabling multiple programs to run in a guarded environment.

RAM

Random-access memory, the computer's primary workspace. Although true of most memory chips (ROMs, PROMs, etc.), *random* means that the contents of each byte can be directly accessed without regard to the bytes before or after it. RAM chips require power to maintain their content.

RAMDRIVE

RAM disks simulate a disk drive in memory and provide fast retrieval of programs and data. If operations are disk-intensive, they can be speeded up using RAM disks; for example, copying files within a RAM disk is almost instantaneous. However, some operations may benefit only slightly. You'll have to try it to find out. RAM disk contents are lost if the power fails or the computer is turned off, thus, data updated in RAM disks should be periodically copied to real disks. RAM disks take on the next available drive letter.

Real Mode

Operational state in Intel 286s and up in which the computer functions as an 8086/8088. It is limited to 1Mb of memory.

ROM

Read-only memory, a memory chip that permanently stores instructions and data. Its contents are created at the time of manufacture and cannot be altered. Used extensively to store control routines in personal computers (ROM BIOS) and in peripheral controllers, it is also used in plug-in cartridges for printers, video games and other systems.

ROM BIOS

ROM basic input output system, instructions contained in a ROM chip that activate peripheral devices in a PC. ROM BIOS includes routines for the keyboard, screen, disk, parallel and serial ports, and for internal services such as time and date. It accepts requests from the device drivers in the operating system as well as from application programs.

It also contains autostart functions that test the system on start-up and prepare the computer for operation. It searches for other BIOSs on the plug-in boards and sets up pointers (interrupt vectors) in memory to access BIOS routines. It loads the operating system and passes control to it.

SMARTDrive

Disk cache program that comes with DOS 4.0 and Windows 3.0. Driver is SMARTDRV.SYS.

SRAM

Static RAM, a memory chip that requires power to hold its content. A bit is made up of a pretzel-like flip-flop circuit that lets current flow through one side or the other based on which one of two transistors is activated. Chips have access times in the 10- to 30-nanosecond range, whereas dynamic RAMs are usually above 30 and Bipolar and ECL memories are under 10.

SRAMs do not require refresh circuitry as do dynamic RAMs, but they do take up more space and use more power.

Stack

Set of hardware registers or a reserved amount of memory used for arithmetic calculations or for keeping track of internal operations. Stacks keep track of the sequence of routines that are called in a program. For example, one routine calls another, which calls another and so on. As each routine is completed, the computer must return control to the calling routine all the way back to the first routine that started the sequence.

Stacks usually work on a LIFO basis; the last item, or address, placed (pushed) onto the stack is the first item removed (popped) from the stack.

An *internal stack failure* is a fatal error which means that the operating system can't find the next routine to process. Restarting the computer usually corrects this; otherwise, the operating system may have to be reinstalled.

Stack Overflow

Error condition that occurs when there is no room in the stack for a new item. A stack underflow occurs when an item is called for but the stack is empty.

TSR

Terminate-and-stay-resident. Refers to programs that remain in memory so that they can be instantly popped up over some other application by pressing a hot key. The program is displayed either as a small window on top of the existing text or image, or it takes up the full screen. When the program is exited, the previous screen contents are restored.

On DOS PCs, TSRs provide quick access to a calculator, calendar or dictionary; however, conflicts may arise when multiple TSRs are loaded. Older ones may not always work with newer ones.

The term refers to loading a program, and terminating its action but not removing it from memory.

UMA

Upper memory area, PC memory between 640K and 1024K.

UMB

Upper memory block, an unused block in the UMA (640K–1Mb). A UMB provider is software that can load drivers and TSRs into this area.

Virtual 8086 Mode

Operational mode in Intel 386s and up that allows them to perform as multiple 8086 CPUs. Under direction of a control program, each virtual machine runs as a stand-alone 8086 running its own operating system and applications, thus DOS, UNIX and other operating systems can be running simultaneously. All virtual machines are multitasked together.

This mode divides up the computer into multiple address spaces and maintains virtual registers for each virtual machine. This is not the same as the 386's virtual memory mode, which extends main memory to disk.

VRAM

Video RAM, specially designed memory circuits on a video display board that are used to hold the image that appears on the video screen. The pixel data or character data stored in the video RAM by the program is translated to the monitor in the required format. Often uses dual-ported RAM, which allows simultaneous reads and writes.

Windows NT

Windows New Technology, an advanced 32-bit operating system for 386s and up from Microsoft scheduled for 1993. It runs applications written for DOS, Windows 3.x, and NT. POSIX support is also planned, but OS/2 compatibility is unclear. NT does not use DOS, it is a self-contained operating system.

XMS

eXtended Memory Specification, an interface that allows DOS programs to use extended memory in 286s and up. It provides a set of functions for reserving, releasing, and transferring data to and from extended memory without conflict, including the high memory area (HMA).

043001