

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

With the advent of modern operating systems such as windows 95, Windows NT, and OS/2, memory management is becoming somewhat of a lost art. However, there are still many applications for it, particularly for those still using DOS or Windows 3.X, and older XT or AT systems. This white paper discusses what memory management is all about, and it explores in detail the application of memory management tools and techniques. So, let us begin.

BASE MEMORY

Also called **Conventional Memory** or **Lower Memory**, it is the first 640K available, which traditionally contains DOS, device drivers, TSR's, and any programs to be run. Thus, the less room DOS takes up, the more there is for the rest. Different versions of DOS were better or worse in this respect. In fact, under normal circumstances, you can expect the first 90K or so to consist of:

- An **Interrupt Vector Table**, which is 1K in size, including the name and address of the program providing the interrupt service. Interrupt vectors point to routines in the BIOS or DOS that programs can use to perform low level hardware access. DOS uses **io.sys** and **msdos.sys** for the BIOS and DOS respectively.
- **ROM BIOS Tables**, which are used by system ROMs to keep track of what is going on. This will include I/O addresses and possibly user defined hard disk data.
- **DOS itself**, plus any associated data files it needs to operate.

DOS was written to run applications inside the bottom 640K block simply because designers of the original IBM PC decided to. Other machines of the same era used more. Contrary to popular belief, Windows uses memory below 1 MB for administrative purposes, although it pools all memory above and below 1 MB and calls it **Global Heap**. Certain Windows structures must live below 1 MB, such as Task DataBase (TDB), which is necessary for starting new tasks.

Every Windows application needs 512 bytes of memory below 1 MB to load, but some will take much more if they can, even all that's available, thus preventing others from loading. This is one source for the "Out of Memory" messages that occur when running Windows. There are programs that purposely fragment base memory so it can't be taken by any single program.

Rather than starting at 0 and counting upwards, memory addressing on the PC uses a two step **segment:offset** addressing scheme. The **segment** specifies a 16 byte paragraph of RAM while the **offset** identifies a specific byte within it. The CPU finds a particular byte in memory by using two registers. One contains the starting segment value and the other the offset. The maximum that can be stored in one is 65,535 (FFF in hex). The CPU calculates a physical address by taking the contents of the segment register, shifting it one character to the left, and adding the two together (see high memory).

Sometimes you will see both values separated by a colon, as with FFFF:000F, meaning the sixteenth byte in memory segment FFFF. This can also be represented as the effective address

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

0FFFFFFh. When referring only to 16 byte paragraph ranges, the offset value is often left out. The 1025K of DOS memory is divided into 16 segments of 64K each.

Conventional memory contains the ten segments from 0000h to 9FFFh (bytes 0 to 65,167), and **Upper memory** contains the six segments ranging from A000h to FFFFh.

UPPER MEMORY

The next 384 is reserved for private use by the computer so that expansion cards with their own memory or ROMs can operate safely there without interfering with programs in base memory. Typical examples include network interface cards or graphics adapters. **There is no memory in it.** The space is simply reserved. This is why the memory count on older machines with only 1 MB was 640 + 384K of **Extended Memory**. The 384K was remapped above 1 MB so it could be used. When upper memory blocks are needed, that memory is remapped back again so you lose a little extended memory.

This area is split into regions A-F, which in turn are split into areas numbered from 0000 to FFF hexadecimally (64K each). With the right software, this area can be converted in Upper Memory for use by TSR programs to make more room below. The amount of upper memory available varies between computers and depends on the amount of space taken by the System BIOS or whether you have a separate VGA BIOS integrated into the system BIOS. It also depends on the number of add-in cards, which normally take around 16K.

Some chipsets will always reserve this 384K area for shadowing so it will not appear in the initial memory count on power up. Other chipsets have a **Memory Relocation** option which will readdress it above 1 MB as extended memory. Occasionally, some ROM space is not needed once the machine has booted and it might be useable. A good example is the first 32K of the System BIOS at F000 in ISA machines. It's only used in the initial stages of booting up, or before DOS gets to set up device drivers, so this area is often usable.

EXTENDED MEMORY

Memory above 1 MB is known as extended memory. It is not normally useable under DOS except to provide RAM disks or caches because DOS runs in real mode, and it can't access extended memory in protected mode which OS/2 and Windows 95 do. Some programs are able to switch the CPU from one to the other by using the **DOS Protected Mode Interface (DPMI)**. Although extended memory first appeared on the 286, and some software was written to take advantage of it, the 286 was used mostly as a fast XT because DOS wasn't rewritten for protected mode. It wasn't until the 386, with its memory paging capability, that extended memory came to be used properly.

HIGH MEMORY

The first 64K (less 16 bytes) of extended memory, which is useable only by 286 and up computers that have more than 1 MB of memory. It's a bug in the chip design that can be exploited by DOS to use that portion of extended memory as if it were below 1 MB, leaving yet

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

more available for programs in base memory. In other words, it is extended memory that can be accessed in real mode. It is activated with **HIMEM.SYS**.

HMA access is possible because of the segment:offset addressing scheme of the PC. Memory addresses on a PC are 20 bits long, and are calculated by shifting the contents of a 16 bit register 4 bits to the left, and adding it to a 16 bit offset. The 8088, with only 20 address lines, cannot handle the address carry bit, so the processor simply wraps around to address 0000:0000 after FFFF:000F. In other words, the upper 4 bits are discarded.

On a 286 or later, there is a 21st memory address line that was left open by accident. It can be operated by software, which gives a dirty bit. If the system activates this bit in 8088 (real) mode, the wraparound doesn't happen and the high memory becomes available.

EXPANDED MEMORY

This is the most confusing one of all. Once the PC was on the market, it wasn't long before 640K wasn't enough, particularly for people using Lotus and not having enough memory to load their spreadsheets. Users got onto Lotus, Intel, and Microsoft for a workaround. They came up with **LIM Memory**, or the **LIM Standard**, also known as Expanded Memory. It's a system of physical bank switching where several extra banks of memory can be allocated to a program, but only one will be in the address space of the CPU at any time, as that bank is switched, or paged, as required. In other words, the program code stays in the physical cells, but the electronic address of those cells is changed, either by software or circuitry.

In effect, LIM (4.0) directly swaps the contents of any 16K block of expanded memory with a similar one inside upper memory. No swapping takes place but the pages have their address changed to look as if it does. Once the page frame is mapped to a page on the card, the data of that page can be seen by the CPU. Some points to note about LIM:

- It is normally only available for data and not program code.
- Programs need to be specially written to use it.

In theory, LIM 4.0 doesn't need a page frame, but the programs expect to see one. In addition, there could be up to 64 pages so you could bank switch up to a megabyte at a time, effectively doubling the address space of the CPU and enabling program code to be run and multitasking. This was called large frame EMS, but it still used only four pages in upper memory. The idea was to remove most of the memory on the motherboard. The memory card backfilled conventional memory and used the extra pages for banking.

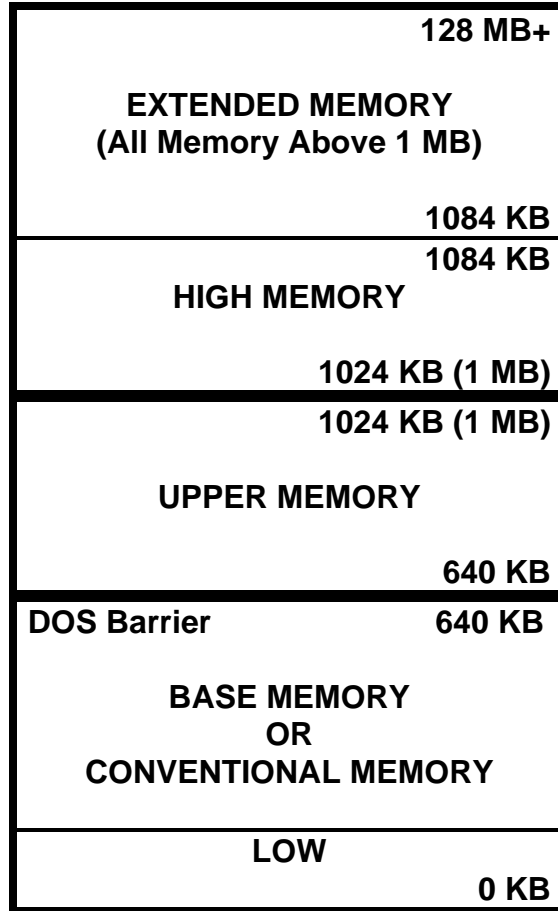
On an 8086 or 286 based machine, expanded memory is usually provided by circuitry on an expansion card, but there are some software solutions. 386 based machines have to emulate LIM with **emm386.exe**.

VIRTUAL MEMORY

LEGACY MEMORY MANAGEMENT

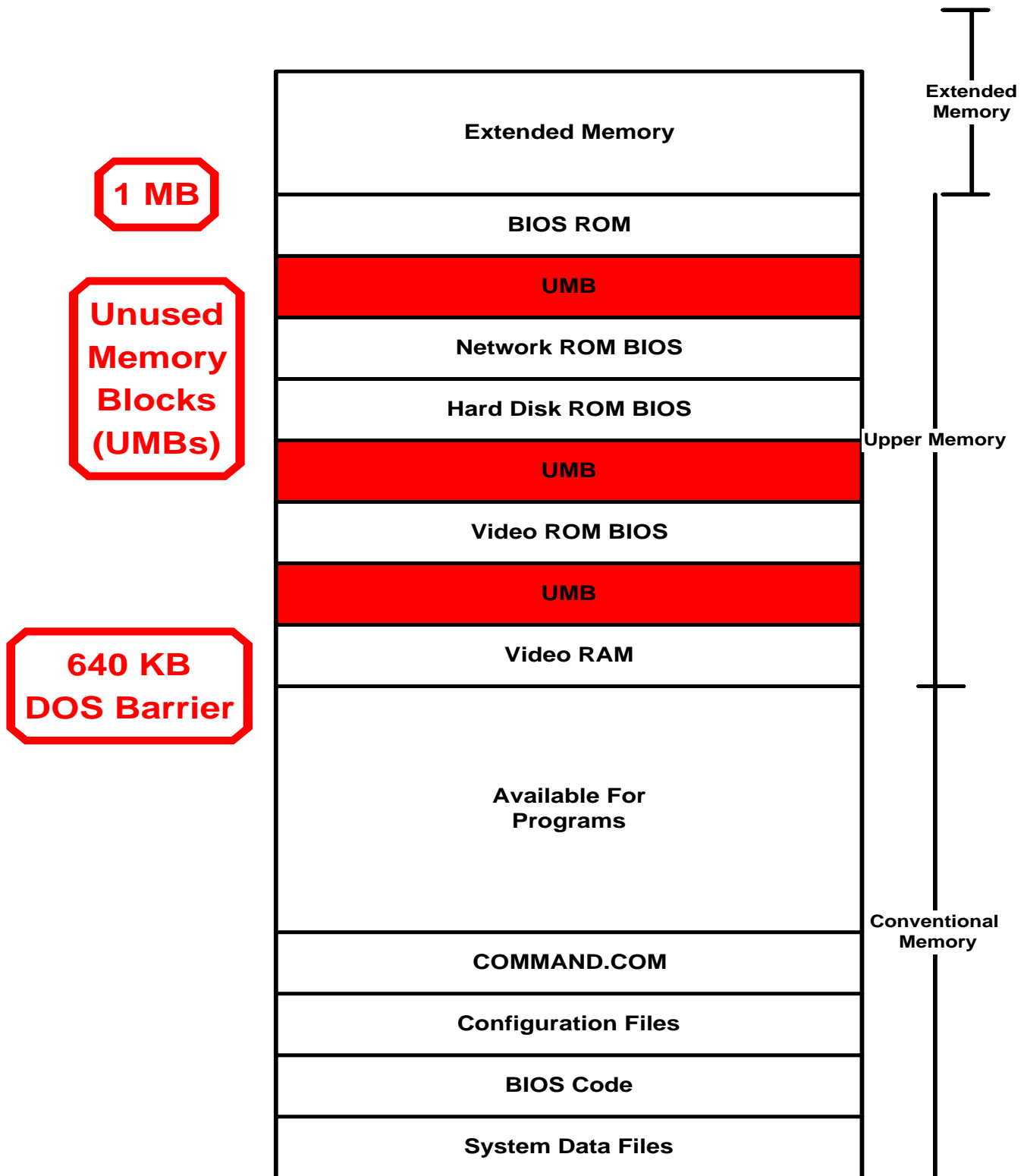
Mark E. Donaldson

Virtual in the computer industry is a word meaning that something is other than what it appears to be. In view of that, Virtual Memory isn't memory at all, but hard disk space made to look like it is. It is the opposite of RAM disk. Windows and System 7 uses virtual memory for swap files used when physical memory is exhausted. Like disk caching, VM was used on mainframes for some time before migrating to the PC. VMS, the OS used on DEC VAX's, actually stands for Virtual Memory System. There is a speed penalty as you have to access the hard disk to use it.



LEGACY MEMORY MANAGEMENT

Mark E. Donaldson



CONFIG.SYS

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

The CONFIG.SYS is used to customize DOS. The CONFIG.SYS can load special device drivers for optional PC equipment, such as a mouse, or for software, such as a virus detection program. The CONFIG.SYS can change DOS system parameters, such as FILES and BUFFERS. The CONFIG.SYS file does not exist until it is created with a text editor. The CONFIG.SYS must be placed in the root directory of the boot drive. While the computer is booting, DOS checks for the presence of the CONFIG.SYS file. The specified device drivers are then loaded into RAM, and the system parameters are changed as requested. If no CONFIG.SYS exists, the default values for the system parameters are used.

Syntax:

DEVICE=drive:\directory\devicedrivename

Example:

DEVICE=C:\MOUSE.SYS

The **BUFFERS** statement sets the number of disk buffers. Disk buffers create a cache in RAM, containing information recently read from disk. This data is scanned by DOS prior from transferring data from the hard drive. A complex program may require 20 or even 30 buffers. A simple one, such as a word processor, will probably only require 10 or 15. If another disk cache program, such as SMARTDRV is being used, the buffers statement can be set very low. A typical buffers statement looks like this:

BUFFERS=20

The **FCBS** (File Control Block System) is an area in RAM that stores information about open files. The default is 4 files control blocks which is normally sufficient. Therefore, it is unlikely that the FCBS statement will be needed in CONFIG.SYS. If it is included, consider setting it to 1 with the command:

FCBS=1

The **FILES** statement is also used to track open files. The default is 8. Most programs specify the number of file handles they need. Set the FILES statement to the highest of those numbers, such as:

FILES=20

The **STACKS** statement is used to track interrupts. If the stack area is not large enough to record all of the interrupts that occur (but have not been handled by the CPU), the error message *Internal Stack Failure, System Halted* may occur. The default number of stacks is frequently set to STACKS=9,128, which creates a stack area of nine stacks of 128 bytes each. Most programs create their own private stacks in memory, so the stack area may be set to zero to save RAM.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

Syntax:

STACKS=numberofstacks, sizeofstacks

Example:

STACKS=9,128

The **SHELL** statement specifies where **COMMAND.COM**, the DOS command interpreter, is located on the boot disk. It also may be used to determine the amount of environment space that is required (**PATH** and **SET** options).

Syntax:

SHELL=drive:\directory\commandinterpeter \p \E:environmentspace

Example:

SHELL=C:\COMMAND.COM \p \E:256

A typical CONFIG.SYS may appear as follows:

```
DEVICE=C\WINDOWS\HIMEM.SYS  
DOS=HIGH,UMB  
DEVICE=C:\WINDOWS\EMM386.EXE NOEMS  
DEVICHIGH=C:\BIN\NAV_.SYS  
DEVICEHIGH=C:\DOS\ANSI.SYS  
DEVICEHIGH=C:\WINDOWS\MOUSE.SYS  
FILES=20  
BUFFERS=30  
SHELL=C:\DOS\COMMAND.COM /P /E:256  
BREAK=ON
```

AUTOEXEC.BAT

The AUTOEXEC.BAT is a special file that automatically executes commands when the system is booted. It can be used to make changes to the DOS environment or to start programs. Some commands typically found in the AUTOEXEC.BAT use RAM. The AUTOEXEC.BAT is created with a text editor and placed in the root directory of the boot drive.

The **PATH** statement provides DOS with a list of directories to search for program files not located in the current directory. The PATH statement is copied into the DOS environmental space. Each directory in the PATH statement is separated by a semi-colon.

Example:

PATH=C:\WINDOWS;C:\DOS;C:\NC;C:\WORD;C:\FASTBACK;C:\NORTON;C:\NAV

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

RAM cache, with certain programs such as **SMARTDRV**, may be created and loaded through the AUTOEXEC.BAT. The **PROMPT** command, used to customize the DOS prompt, is also loaded with AUTOEXEC.BAT. TSRs may also be loaded through AUTOEXEC.BAT.

A typical AUTOEXEC.BAT file appears as the following:

```
C:\WINDOWS\SMARTDRIVE.EXE
@C:\WINDOWS\AD_WRAP.EXE
@echo off
cls
PATH=C:\BIN\EXEL4;C:\WINDOWS;C:\DOS;C:\BIN\NC;C:\BIN\WORD;C:\FASTBACK;C:\BIN\NORTON;C:\BIN\NAV
prompt $P$G
set TEMP=C:\WINDOWS\TEMP
SET TMP=C:\WINDOWS\TEMP
SET FASTBACK=C:\FASTBACK
SET PCPLUS=C:\PCPLUS
C:\WINDOWS\AD_DOS.COM
```

ACCESSING HIGH MEMORY (HMA) and EXTENDED MEMORY

The high memory area (HMA), found in the first 64K of extended memory, can be used for DOS, device drivers, and TSRs with the use of device driver **HIMEM.SYS**.

Once high memory is made available by loading **HIMEM.SYS** through CONFIG.SYS, DOS can be moved out of conventional memory into the HMA. The command DOS=HIGH is used after the driver **HIMEM.SYS** is loaded.

Unused areas of upper memory (UMBs) can be used to load TSRs and device drivers into expanded memory through the following process:

- Load **HIMEM.SYS** (the driver that provides access to the high memory area).
- Load EMM386.EXE (the driver that controls expanded memory).
- Use the **DOS=UMB** command (which transfers the command of the expanded memory areas in upper memory to DOS).
- Once the UMBs have been created, use the DOS command **LOADHIGH** to load TSRs, and the DOS command **DEVICEHIGH** to load device drivers into high memory.

To make extended memory available, load the **EMM386.EXE** device driver.

Extended memory can be made to simulate expanded memory through a two step process:

- Load **HIGMEM.SYS** to provide access to the high memory areas.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

- Load **EMM386.EXE** to create the page frame in upper memory and to control expanded memory.

HIMEM.SYS

The device driver HIMEM.SYS is used in the CONFIG.SYS to provide access to the high memory area. The command that loads the HIMEM.SYS driver should be the first within CONFIG.SYS, unless the system loads a device driver that provides access to the hard disk drive (it would then be the first).

Syntax:

```
DEVICE=[drive:\directory\]HIMEM.SYS  
[/machine:xxxx]  
[/A20CONTROL:ON|OFF]
```

HIMEM.SYS has several advanced options that can be used to specify:

- The minimum number of kilobytes a program must request before being allowed to use high memory.
- The number of extended memory block handles that can be used at one time.
- The amount of memory allocated to the 15H interrupt interface (applications use the 15H interrupt to access extended memory).
- Whether shadow RAM is enabled.
- Whether a great deal of information is displayed on the screen when HIMEM.SYS is loaded and run.

Examples:

```
DEVICE=C:\DOS\HIMEM.SYS  
DEVICE=C:\DOS\HIMEM.SYS /MACHINE:7  
DEVICE=C:\DOS\HIMEM.SYS /A20CONTROL:ON  
DEVICE=HIMEM.SYS /V
```

EMM386.EXE

The EMM386.EXE driver:

- Fills UMBs (upper memory blocks) with expanded memory so device drivers can be loaded there.
- Provides access to extended memory for programs that use it.
- Simulates expanded memory with extended memory.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

- Excludes Windows from accessing UMA.

EMM386.EXE is loaded through CONFIG.SYS, and it must be located after the HIMEM.SYS command, but before any expanded memory commands.

Syntax:

DEVICE=[drive:\directory\]EMM386.EXE [mem] [RAM|NOEMS]

Mem is the number specifying the amount of extended memory (in K) that simulates expanded memory from 16 to 32,786. The default is 256K.

RAM or **NOEMS** activates upper memory blocks. If **RAM** is used, upper memory blocks and expanded memory are available. If **NOEMS** is used, upper memory blocks are available, but no expanded memory is simulated and all extended memory remains extended.

EMM386.EXE has several advanced options that can be used to specify:

- The exact upper memory address where the page frame starts.
- The exact upper memory address for a page.
- A range of addresses in upper memory to exclude when creating the page frame.
- The lowest address in extended memory to be used.
- The amount of extended memory (in K) that is not to be used as expanded memory.
- The number of fast alternate register sets to be used for tracking expanded memory.
- The number of page handles used.
- How much of the UMA is to be dedicated for use by Windows and not EMM386.
- Whether EMM386 is to load itself entirely into conventional memory.
- Whether EMM386 should exclude the use of the UMA in the event that a memory error messages from an application that resides in the UMA.
- If a detailed display should be given when EMM386 is executed.

Examples:

DEVICE=C:\DOS\HIMEM.SYS

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

```
DOS=HIGH ,UMB  
DEVICE=C:\DOSEMM386.EXE NOEMS
```

```
DEVICE=C:\DOS\HIMEM.SYS  
DOS=HIGH ,UMB  
DEVICE=C:\DOS\EMM386.EXE 4096 RAM
```

DOS=HIGH

The DOS=HIGH command is used somewhere in the **CONFIG.SYS** after the **HIMEM.SYS** command to load DOS into high memory. This command can also be used to load device drivers and TSRs into upper memory.

Syntax:

```
DOS=[HIGH|LOW] , [UMB|NOUMB]
```

HIGH or **LOW** controls whether DOS loads into high memory. If **HIGH** is specified, then DOS loads into high memory. If **LOW** is specified, then DOS loads into conventional memory. The default is **LOW**.

UMB or **NOUMB** controls whether device drivers and TSRs load into upper memory. If **UMB** is specified, DOS controls upper memory blocks and lets device drivers and TSRs load there. If **NOUMB** is specified, extended memory controls any UMBs created, and device drivers and TSRs are not allowed to load into upper memory.

Examples:

To load DOS into high memory, two commands are needed:

```
DEVICE=C:\DOS\HIGHMEM.SYS  
DOS=HIGH
```

To load device drivers and TSRs into upper memory, several commands are needed:

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

DEVICEHIGH

The DEVICEHIGH command is used in the **CONFIG.SYS** to load device drivers into upper memory if there is enough room for them. If a device driver is too large to fit into upper memory, it loads into conventional memory automatically. DEVICEHIGH won't work without **HIMEM.SYS**, **EMM386.EXE**, and **DOS=HIGH,UMB**.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

Use the **MEM/C** command to see which device drivers in conventional memory are the largest and to load them into upper memory according to size (from largest to smallest). **DEVICEHIGH** estimates the size of a device driver and loads it into an appropriate open space in upper memory. However, some device drivers request additional memory after they have been loaded and cause the system to lockup.

Syntax:

DEVICEHIGH=[SIZE=size][drive:\directory\]driverfilename

SIZE=size *size* is used to specify (in hexadecimal) the minimum amount of upper memory required to load this device. A device will not be loaded if the minimum amount specified is not available. If *size* is not specified, DOS will estimate the amount of upper memory to allocate.

Example:

To load the device driver, **ANSI.SYS** into upper memory, use the following commands:

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

LOADHIGH

LOADHIGH is the **AUTOEXEC.BAT** file's equivalent of **DEVICEHIGH**. The **LOADHIGH** command is used by both DOS 5 and DOS 6 to load TSRs and certain device drivers into upper memory through **AUTOEXEC.BAT**. The same conditions for use apply with **DEVICE=HIGH:HIMEM.SYS**, **EMM386.EXE**, and **DOS=HIGH,UMB** must first be set up.

Syntax:

LOAD HIGH=[drive:\directory\]driverfilename

Example:

To load the DOS TSR **DOSKEY.COM** into upper memory, include these commands in the **CONFIG.SYS**:

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

Then include this command in the **AUTOEXEC.BAT**:

```
LOAD HIGH=DOSKEY
```

WINDOWS 3.X

There are several things that can be done to streamline memory useable under the Windows 3.X environment:

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

- Use all of extended memory as extended memory, not expanded. Windows cannot make use of expanded memory unless it is running in standard, and not enhanced, mode. Use the following in the **CONFIG.SYS**:

DEVICE=C:\DOS\EMM386.EXE NOEMS

- Set **FILES** at 30.
- Set **BUFFERS** at 30 if not using **SMARTDRV**, or from 1 to 15 if using **SMARTDRV**.
- If loading a device driver, put it in UMB as follows:

C:\DOS\HIMEM.SYS
DOS=HIGH,UMB
DEVICE=C:\DOSEMM386.EXE NOEMS
DEVICEHIGH=C:\DOS\MOUSE.SYS

- Set **STACKS** to zero because Windows does not need a **STACKS** statement.

STACKS=0,0

- In the **AUTOEXEC.BAT**, load TSRs into UMBs (or consider loading them with the **WINSTART.BAT** if the TSRs are to be used with Windows programs only).

LOADHIGH =C:\DOS\SOSKEY

SMARTDRV

SMARTDRV is a program that creates a disk cache in RAM. A disk cache is a buffer in memory that stores copies of information read from the hard disk. IF the same information is requested later, it can quickly be read from RAM, rather than from the Hard disk. When the cache gets full, the least requested information is overwritten, ensuring that only the most needed information is kept on the cache. **SMARTDRV** do not cache floppy disk operations, so a **BUFFERS** statement is still needed in the **CONFIG.SYS** for those purposes.

The **SMARTDRV** cache is created using extended memory, unless the **/A** parameter is used, and then expanded memory is used. **SMARTDRV** provides two services. It acts as a disk cache when loaded from the **AUTOEXEC.BAT** file, and it provides double buffering if it is loaded from the **CONFIG.SYS**. If the hard drive is SCSI or ESDI, or another drive identified as needing double buffering, **SMARTDRV** should be loaded into both the **AUTOEXEC.BAT** and **CONFIG.SYS** files to enable both disk caching and double buffering.

Syntax:

[drive:\directory\]SMARTDRV.EXE
[DOScache]{WINcache}

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

DOScache specifies the size (in K) of extended memory the cache should occupy prior to Windows starting. **WIN**cache specifies the maximum size (in K) of extended memory that the cache can occupy once Windows has started.

Examples:

To set up double buffering in the **CONFIG.SYS**:

```
DEVICEHIGH=C:\DOS\SMARTDRV.EXE /DOUBLE_BUFFER
```

To create a disk cache with the **AUTOEXEC.BAT**:

```
C:\WINDOWS\SMARTDRV.EXE 2048 1024
```

RAMDRV.SYS and VDISK.SYS

RAMDRV.SYS is a program that creates a RAM disk in memory. A RAM disk looks and acts like any other disk drive (such as C), but the computer can quickly read and write information to it because it is made in RAM. Treat a RAM disk like any other disk drive by copying files, deleting files, making directories, and so on. RAMDRV comes with DOS 5, DOS 6, and Windows 3.1. Prior versions provided the utility VDISK.SYS which accomplished the same thing. Both RAMDRV.SYS and VDRIVE.SYS are loaded through the **CONFIG.SYS** file.

Syntax:

```
DEVICE[HIGH]=[drive:\directory\]RAMDRV.SYS  
[disksize]  
[sectorsize] [direntries] [/E] [/A]
```

disksize is the size of the RAM disk in K. Allowable values are from 16 to 4,096. The default is 64. **sectorsize** is the size of the sectors (in bytes) used for the RAM disk. Allowable values are 128, 256, and 512. The default is 512. If this parameter is used, **disksize** must be specified. **direntries** puts a limit on the number of files and directories that can be placed in the root directory of the RAM disk. Allowable values are from 2 to 1,024. The default is 64. If this parameter is used, **disksize** and **sectorsize** must also be specified. **/E** tells RAMDRV to place the RAM disk in extended memory, while **/A** tells RAMDRV to put it in expanded memory.

Once a RAM disk is created, be sure to include this statement (the drive letter of the RAM disk) in the **AUTOEXEC.BAT** file:

```
SET TEMP=D:\
```

Example:

```
DEVICEHIGH=C:\DOS\RAMDRV.SYS 2048 /E
```

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

MEMMAKER

MemMaker is a DOS program that will set up the system's memory management program. MemMaker actually interviews to determine the type of work done with the computer, and then configures the **CONFIG.SYS** and **AUTOEXEC.BAT** files for optimum performance. It can analyze hardware and software to determine the best fit for programs, device drivers, and TSRs that can be loaded into upper memory. To run MEMMAKER type:

MEMMAKER

It will then ask to choose between Express or Custom setup options. Express Setup makes the choices for you while the Custom Setup prompts for informed choices. Some of the questions that may be asked include:

- Whether or not Windows is used.
- Whether any programs that require expanded memory are used. If not sure, answer NO.
- Which drivers and TSRs (currently listed in the configuration files) are to optimized, if not all of them.
- Whether or not to scan aggressively. This may result in better memory management, but if MemMaker makes an error, the system may lockup.
- Whether or not to use the region of memory reserved for monochrome display adapters as available memory for programs.
- Whether or not to keep current command lines for both EMM386 and HIMEM.
- Whether to copy BIOS instructions from conventional memory to the UMA.

DEFINITIONS

640 Barrier - The DOS barrier.

ANSI.SYS - A DOS device driver used to enhance the capabilities of a PC's keyboard and monitor.

AUTOEXEC.BAT - A special file that automatically executes commands when the system is started.

BUFFERS - A statement place in the CONFIG.SYS which sets the number of disk buffers.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

CONFIG.SYS - A file which is used to customize DOS by loading special device drivers for optional PC equipment, or to change DOS system parameters, such as FILES, BUFFERS, and so on.

Conventional Memory - Also known as Base Memory or Lower Memory, this is the area of memory (0K to 640K) in which programs run.

DEVICE - A DOS command that, when used in the CONFIG.SYS file, loads a device driver into lower memory.

DEVICEHIGH - An MS DOS command which is used to load device drivers into upper memory.

DOS Barrier - Also known as the 640K barrier, it is the arbitrary division between conventional (program) memory (0K to 640K) and upper (system) memory (640K to 1 MB).

EMM368.EXE - An MS DOS device driver that provides access to extended memory, fills UMBs with memory so that device drivers and TSRs can be loaded there, and simulates expanded memory with extended memory.

EMS - Expanded Memory Specification.

Expanded Memory - Special memory that's accessed through a page frame in upper memory. The page frame swaps memory back and forth between upper memory and expanded memory.

Expanded Memory Specification - Abbreviated EMS, the standard for accessing expanded memory. Also known as the LIM Standard.

Extended Memory - The area of RAM above the 1 MB mark. Extended memory can be used for data storage by programs that are designed for it. Extended memory can be made to simulate expanded memory.

FILES - A statement placed in the CONFIG.SYS which limits the number of file handles (indexes which track open files).

High Memory Area - Also known as HMA, it is located in the first 64K of extended memory. The HMA is only available to PC's with a 286 microprocessor and above.

HIMEM.SYS - An MS DOS device driver used to provide access to the high memory area.

INSTALL - A DOS command that loads programs named in your CONFIG.SYS file without creating additional environment space for that program. Some programs don't require environment space, so some system memory can be saved by using the INSTALL command in certain cases.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

Interleaved Memory - Memory is divided into two groups, even and odd memory addresses. While an even address is refreshing, the odd address is ready to be read, and vice versa. The CPU does not have to wait for the memory to refresh because the next memory address to be read is most likely ready.

LOADHIGH - An MS DOS command that is used to load TSRs (and certain device drivers) into upper memory.

MEM - A DOS command (available in DOS 4.0 and above) that is used to determine the PC's memory configuration.

MemMaker - A DOS 6 utility that automatically sets up startup files to manage memory.

Page Frame - An unused area of upper memory which forms the gateway to expanded memory. Data is copied from expanded memory into the page frame for processing, and then back again.

PATH - A statement in the AUTOEXEC.BAT that provides DOS with a list of directories to search for when program files are not located in the current directory.

Protected Mode - In order to address memory areas above 1 MB, the CPU must be switched to protected mode. These additional addresses are accessed with the help of the extended memory manager.

RAM Disk - A disk drive created in memory. A RAM disk looks and acts like any other disk drive. Information can be quickly read and written to it because it is made out of memory. However, because a RAM disk is made of memory, its data is erased when the PC is turned off.

RAMDRIVE.SYS - An MS DOS device driver that comes with DOS 5. Prior versions used VDISK.SYS. RANDRIVE can be used to create a RAM disk in either conventional, expanded, or extended memory.

Real Mode - The CPU's normal operating mode. In real mode, only memory addresses up to 1 MB can be accessed by the CPU.

SET - A statement placed in the AUTOEXEC.BAT used to place notes into the environmental area for programs.

Shadow RAM - Lets a PC run faster by copying the ROM information into RAM. When basic Input/Output instructions are needed, those instructions are read from RAM, which is faster than trying to read them from ROM.

SHELL - A statement place in the CONFIG.SYS which specifies where COMMAND.COM is located on the boot disk, or to increase the environment space.

LEGACY MEMORY MANAGEMENT

Mark E. Donaldson

SMARTDRV.SYS - An MS DOS device driver used to create a disk cache.

STACKS - A statement placed in the CONFIG.SYS limiting the stack area in memory which keeps tracks of interrupts.

Standard Mode - A Windows operating mode using conventional memory and extended memory only. In standard mode, existing expanded memory is ignored by Windows as it acts as an intermediary between DOS and the application, passing the requests for expanded memory into DOS.

UMB - Also know as Upper Memory Blocks, these are areas of upper memory (640K to 1 M) which normally go unused. Through the use of an upper memory device driver, you can utilize this seldom used area for memory.

Upper Memory - Also known as adapter memory, upper memory (640K to 1M) was designed for DOS and the computer's hardware use, but can be used for TSRs and device drivers.

VDISK.SYS - A DR DOS and MS DOS (versions prior to DOS 5) device driver. VDISK can be used to create a RAM disk in either conventional, expanded, or extended memory.

Virtual Memory - Additional memory capacity that's simulated by using hard disk space.