PC-MOS™

MODULAR OPERATING SYSTEM

ADVANCED SYSTEM MANUAL

by Martin J. Stitt

a product of

THE SOFTWARE LINK, INCORPORATED

3577 PARKWAY LANE
NORCROSS, GEORGIA 30092 USA
404/448-LINK

Limited Use License Agreement

You should carefully read the following terms and conditions before opening this package. Opening this package indicates your acceptance of these terms and conditions. If you do not agree with them, you should promptly return the package unopened to the person from whom you purchased it within fifteen days from date of purchase and your money will be refunded to you by that person. If the person from whom you purchased this package fails to refund your money, contact TSL immediately at the address set out below.

TSL provides computer software contained on the medium in the package (the "Program"), and licenses its use. You assume responsibility for the selection of the Program to achieve your intended results, and for the installation, use and results obtained from the Program.

LICENSE

a. You are granted a personal, non-transferable and non-exclusive license to use the Program under the terms stated in this Agreement. Title and ownership of the Program and documentation remain in TSL;

b. the Program may be used by you, your employees, or your agents only on a single computer; or, in a TSL-approved computer network where all of the Program's data resides at one disk file server;

c. you and your employees and agents are required to protect the confidentiality of the Program. You may not distribute or otherwise make the Program or documentation available to any third party;

d. you may not copy or reproduce the Program or documentation for any purpose, except you may copy the Program into machine readable or printed form for backup purposes in support of your use of the Program. (Any portion of this Program merged into or used in conjunction with another program will continue to be the property of TSL and subject to the terms and conditions of this Agreement);

e. you may not assign or transfer the Program or this license to any other person without the express prior consent of TSL; and

f. you acknowledge that you are receiving only a **LIMITED LICENSE TO USE** the Program and related documentation and that TSL retains title to the Program and documentation. You acknowledge that TSL has a valuable proprietary interest in the Program and documentation.

You must reproduce and include the copyright notice on any copy, modification or portion merged into another program. You may modify the Program for your own use, entirely at your own risk, provided that the Program is used as specified in Section (b) of this Agreement.

YOU MAY NOT USE, COPY, MODIFY, OR TRANSFER THE PROGRAM, OR ANY COPY, MODIFICATION OR MERGED PORTION, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS LICENSE.

IF YOU TRANSFER POSSESSION OF ANY COPY, MODIFICATION OR MERGED PORTION OF THE PROGRAM TO ANOTHER PARTY, YOUR LICENSE IS AUTOMATICALLY TERMINATED.

TERM

The license is effective until terminated. You may terminate it at any other time by destroying the Program together with all copies, modifications and merged portion in any form. It will also terminate upon conditions set forth elsewhere in the Agreement or if you fail to comply with any term or condition of the Agreement. You agree upon such termination to destroy the Program together with all copies, modifications and merged portions in any form.

LIMITED WARRANTY

EXCEPT AS STATED BELOW IN THIS SECTION THIS PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (AND NOT TSL OR AN AUTHORIZED DEALER) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

TSL does not warrant that the functions contained in the Program will meet your requirements or that the operation of the program will be uninterrupted or error free.

TSL does warrant as the only warranty provided to you, that the diskette(s) or cassettes on which the program is furnished, will be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your receipt.

LIMITATION OF REMEDIES

TSL's entire liability and your exclusive remedy shall be:

1. the replacement of any diskette or cassette not meeting TSL's "Limited Warranty" and which is returned to TSL or an authorized TSL dealer with a copy of your receipt, or
2. if TSL or the dealer is unable to deliver a replacement diskette or cassette which is free of defects in materials or workmanship, you may terminate this Agreement by returning the Program and your money will be refunded to you by the dealer from whom you purchased the Program.

IN NO EVENT WILL TSL BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAM EVEN IF TSL OR AN AUTHORIZED TSL DEALER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

GENERAL

You may not sublicense, assign or transfer the license or the Program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the State of Georgia.

Should you have any questions concerning this Agreement, you may contact TSL by writing to The Software Link, Incorporated, 3577 Parkway Lane, Atlanta, Georgia 30092, USA.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND TSL AND ITS DEALER ("US") WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

PC-MOS
GUIDE TO OPERATIONS
Preface

The PC-MOS Guide To Operations covers four general areas of the PC-MOS operating system:

1. Initial system configuration.
2. The use of system utilities and batch files.
3. Configuring and running applications.
4. Technical details of PC-MOS for hardware and software developers.

PC-MOS will frequently be referred to as MOS throughout this guide.

The guide is divided into two manuals:

Advanced System Manual

The Advanced System Manual covers the first three topics listed above and will be most useful for people who are involved with MOS at an operational level. This includes system administrators, systems integrators, consultants, advanced users, technical support technicians, and software and hardware developers. Novice computer users may find it helpful to first become more familiar with the basics of the PC/MS-DOS operating system and IBM PC/AT computer architecture.

Descriptions of the applications of various internal commands and system utilities appear throughout the Advanced System Manual in sections appropriate to their use. If you need information on a particular command, consult the index. A glossary of technical terms is also included at the end of the Advanced System Manual. For details on the proper syntax and usage rules for each command, refer to the PC-MOS USER GUIDE.

Technical Reference Manual

The Technical Reference Manual covers the technical details of MOS; how to construct applications programs and device drivers to interface with MOS's data structures, and system calls and programming techniques which are pertinent to a multitasking environment.

It is assumed that the user is already familiar with the general features of different CPU's and computer systems. The 80386 microprocessor itself will not be explained in detail. In addition, this book assumes familiarity with the hexadecimal numbering system. A brief review of this topic is included in Chapter 1 of the Advanced System Manual.

When the term multitasking is used, it may refer to the support of multiple tasks for multiple users (one task per workstation/user), multiple tasks for a single user, or multiple tasks for multiple users. Since the phrase "multitasking and multiuser" is too cumbersome for frequent use, where a distinction is important it will be stated. In the context of PC-MOS, the basic difference between a multitasking scenario and a multiuser one is that the console I/O of a task in the multiuser case is associated with its own workstation console.

Trademark Notice

ALL CARD and CHARGECARD are trademarks of All Computers, Inc.

AT-GIZMO is a trademark of The Software Link, Inc.

Fiber Optic Station is a trademark of SunRiver Corporation.

Hercules is a trademark of Hercules Computer Technology, Inc.

IBM is a registered trademark of International Business Machines Corporation.

IBM PC-AT is a trademark of International Business Machines Corporation.

Intel is a registered trademark of Intel Corporation.

LANLink is a trademark of The Software Link, Inc.

PC-Emulink is a trademark of The Software Link, Inc.

PC-DOS is a trademark of International Business Machines Corporation.

PC-MOS is a trademark of The Software Link, Inc.

SunRiver is a registered trademark of SunRiver Corporation.

WYSE 60 is a trademark of WYSE Technology.

Copyright Notice

NOTE: This information package contains TSL policies, procedures, programs and pricing that ar in effect at the time of publication. TSL reserves the right to change or cancel any policy, procedure, program or price at any time without notice.

In no event will TSL be liable to you for any damage, including any lost profits, lost savings or other incidental or consequential damages arising out of the use or inability to use any of the information provided in this package.

PC-MOS
ADVANCED SYSTEM MANUAL
Table of Contents

CHAPTER 1:
MEMORY MANAGEMENT

Hex Numbering Conventions

Before going any further, a little groundwork is in order concerning the use of hexadecimal numbers to specify memory addresses. Although some literature represents memory addresses in a segment:offset format such as A000:0 or 40:17, this manual will use the more absolute, 5 digit format. For example, the address which represents the start of the EGA/VGA video memory region is A0000 and the highest address within the base 640k of memory is 9FFFF. Likewise, the first address within the BIOS ROM address region is F0000 and the BIOS data area begins at 00400.

In many places within this book, there are references to ranges of addresses. Note that when a range of memory is specified as being from C0000 to F0000, this should be read as: "all memory starting with C0000 up to, but not including F0000". An alternate way of specifying this same range of addresses could be: "C0000 through CFFFF inclusive".

When using the 5 digit format, certain rules can be applied to make working with these addresses quick and easy. For instance, when the left-most digit increases by 1, a 64K region of memory has been traversed. Thus, the amount of memory contained in the range from C0000 to D0000 is 64K (remember to read this as C0000 up to, but not including D0000, or C0000 through CFFFF inclusive).

In addition, when working with the memory management techniques employed within MOS, there are numerous places where memory must be treated in blocks of 4K. Furthermore, it is important that the block starts on an address that is aligned upon a 4K boundary. This means that the block's starting address is a whole multiple of 4K. When using a 5 digit hex address, the right-most 3 digits must be zero for the address to be on a 4K boundary. In addition, when traversing a 4K region of memory, the 4th digit from the right will increase by 1. For example, 4K is spanned when going from address C4000 to C5000.

CPU Basics

The PC-MOS™ operating system can support multiple tasks running concurrently within a single computer system. Given the proper conditions, each one of these tasks can provide as large a memory space as you have come to expect from a stand-alone PC/AT with 640K of base memory installed. In certain cases, MOS can support PC-DOS™ style tasks which are effectively 20% larger than that which is available on a 640K machine running under PC-DOS.

To understand much of how the PC-MOS operating system functions, a fundamental understanding of memory management is essential. It is through the capabilities of memory management that MOS can support multiple tasks and multiple users. In addition, memory management plays a key role in optimizing the initial loading of the operating system's components into your computer when you boot up.

We'll begin with a quick review of how a computer's CPU (Central Processing Unit) interacts with the system memory in the simplest of cases - when no MMU (Memory Management Unit) is available. From here, we'll explore what this means in terms of loading the operating system into a computer's memory as well as loading in the code and data for applications programs. Once we've gotten an idea of what can and can't be done when there is no MMU, the concept of memory management will be introduced. This will be followed by a fairly in-depth examination of the role that memory management plays in enhancing both the loading of the operating system and of applications.

Figure 1-1 is a simplified representation of how a computer's CPU and RAM memory are interconnected. Each time the CPU needs to read or write information within memory, an address signal is generated in order to select the desired memory element. Once this address has been asserted by the CPU, the data transfer may take place. Note that additional control signals must be asserted by the CPU to stipulate whether a read or write operation is to be done. These are neglected in figure 1-1 for the sake of simplicity.

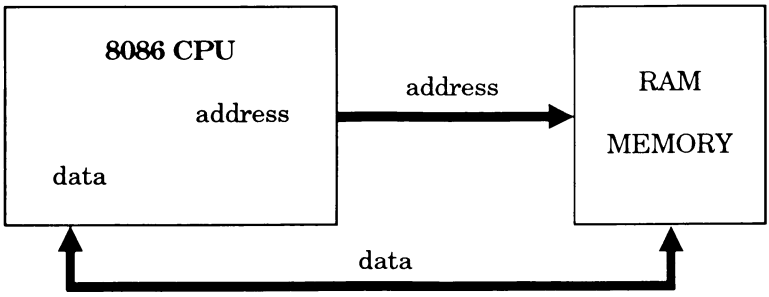

Figure 1 - 1

In a simple CPU/MEMORY relationship, there is a 1:1 correspondence between logical and physical addresses. The addresses generated by the CPU appear directly at the memory chips without any translation.

The PC-DOS Memory Map

The important thing to understand about a system which uses this simple type of CPU-to-memory interface is that there are definite limitations on how much memory is available and where it may be addressed. The range of addresses an 8086 type of CPU can access is from 0 to FFFFF inclusive, which means that this CPU can directly address a maximum of 1 megabyte of memory.

This address range provides for a base address region comprised of 640K of RAM memory which is used to hold the operating system and the code and data of applications programs. The remaining 384K within this 1 Megabyte holds various entities such as video display buffers and the ROM BIOS. These particular memory allocations are part of the IBM PC/AT machine standard rather than being any built-in assumptions within the 8086 CPU itself.

Figure 1-2 summarizes how the various components of the PC-DOS operating system fit into memory once a computer system has completed the boot up process.

Beginning at the bottom and working up, the interrupt vector table and the BIOS data area are the first two system components. These are fixed both in size and location and are independent of the operating system chosen. The next four components, the PC-DOS kernel, disk buffers, device drivers and command processor are specific to PC-DOS.

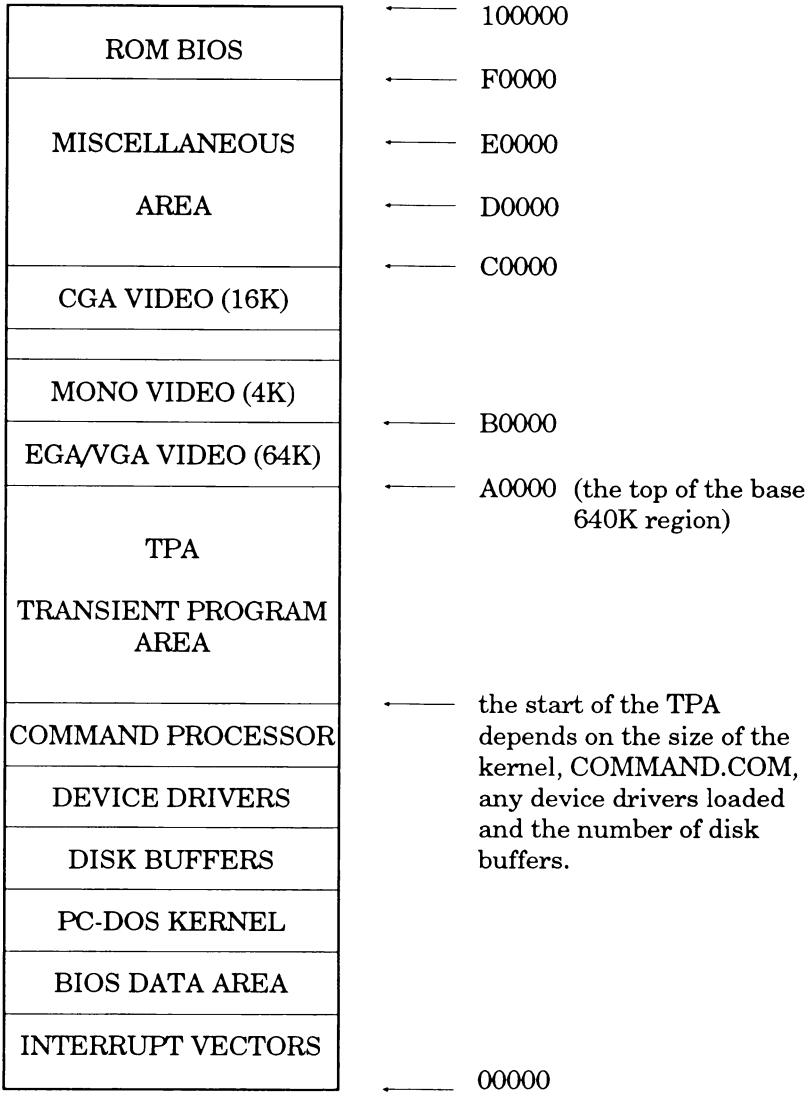

Figure 1 - 2

Memory map of a PC-DOS system after the boot up process has completed.

Following these four components is an area known as the Transient Program Area, or TPA for short. This is the region of the base 640K memory which is available for the code and data of applications programs. Maximizing the size of this area is, of course, an important consideration in setting up a computer system. Unfortunately, when using PC-DOS, about the only way to increase the TPA's size is to lower its lower boundary and the only way to do this is to reduce the amount of memory required for the operating system's components. The TPA is bounded on the top by the fixed address A0000 which is where the base 640K of memory ends and the video buffers start.

For a given version of PC-DOS, the sizes of the resident portion of the command processor and the kernel are fixed entities. The amount of room taken by device drivers is as fixed as your need for those drivers, and reducing the size of the disk buffer area too much will hurt your system's performance in a vital area. What this means is that your options are quite limited. Aside from EMS, which is only valid for applications which are designed to use it, little else can be done to increase the TPA's size.

PC-MOS Memory Map in the Non Memory Managed Case

When the PC-MOS operating system is booted in a computer system where the capabilities of memory management are not available, a similar situation exists as with PC-DOS. The operating system components take up what they must from the start of the base 640K with the remainder comprising the TPA. As can be seen in figure 1-3, MOS's components are quite similar to DOS's. A detailed description of each of MOS's system components appears later in this chapter.

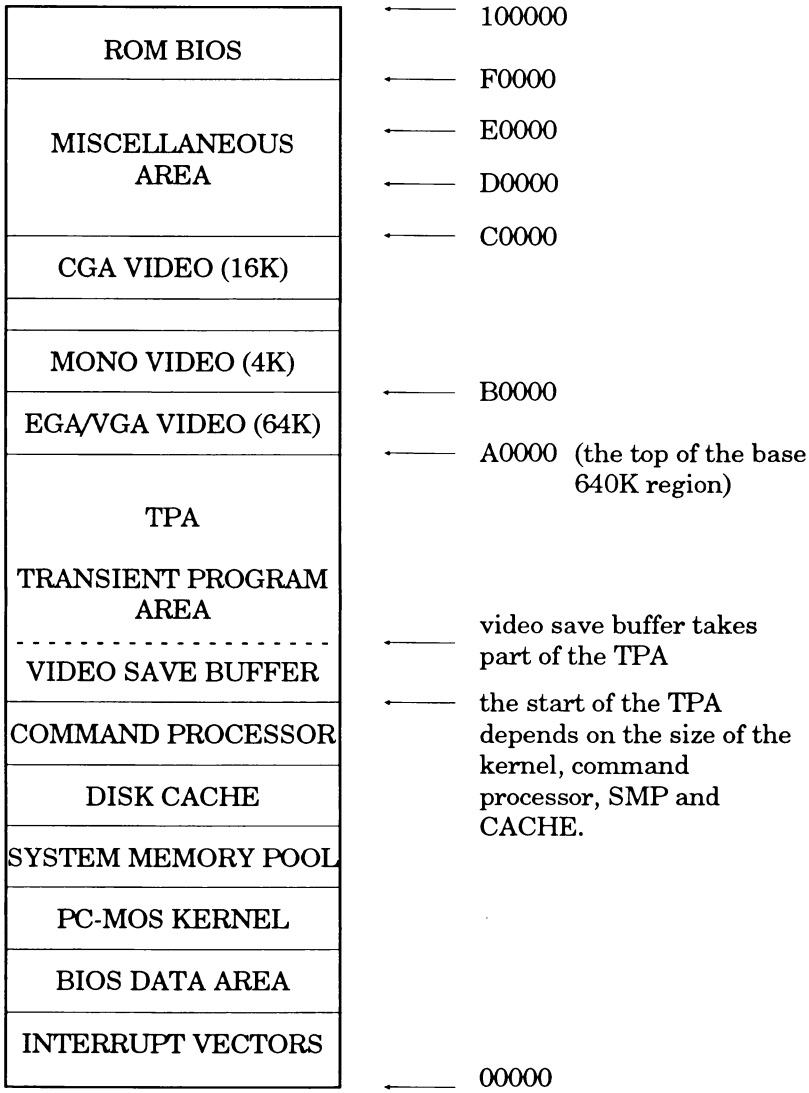

Figure 1 - 3

Memory map of PC-MOS when memory management is not used.

To support PC-DOS style tasks the operating system code must be within the first megabyte along with the code and data of the applications programs which will be used. One drawback of using MOS in a non-memory managed configuration is that since MOS provides more features (e.g. multiple tasks, large disk volumes, security, etc.) MOS's operating system overhead is necessarily larger than that of PC-DOS. In addition, although MOS can support multiple tasks when memory management is not available, the main TPA memory area must be split up to provide memory for each task.

The basic limitation in a non-memory managed configuration stems from the fact that there is a one-to-one correspondence between the address information generated by the CPU and the address which actually arrives at the pins of the memory chips. In other words, when a program calls for a read of the memory location at address 30000, address and memory read signals are generated which cause the physical memory element at 30000 to present its contents to the CPU.

When saddled with the limitations of a one-to-one association between the logical address generated by the CPU and the physical address which appears on the address pins of the memory chips, attempts at multitasking are fraught with sacrifices. Splitting up one TPA provides too small of a task size and swapping the contents of the TPA to disk, as used in mainframes, is too slow. Furthermore, the additional operating code used to support multiple tasks takes up more memory itself.

Memory Management to the Rescue

Now that we've had a look at some of the limitations when there is no MMU support - lets move forward and see what can be done when we do have MMU support. By introducing the ability to selectively define which physical memory location will correspond to which logically generated address, great benefit can be achieved. However, to call this new association between logical and physical addresses a one-to-many relationship would be confusing unless certain clarifications are made. At any given moment, when the CPU asserts a memory address, in order to fetch the next program instruction or to read or write some data, this logical address generation must result the selection of only one physical memory element. There's nothing to be gained but chaos from a literal one-to-many transformation.

Where advantage can be gained is when logical address L can be associated with physical address P1 during the execution of one task; and then the same logical address L can be made to correspond to another physical address P2 when a different task is to be given its slice of execution time. This type of relationship is one-to-many in the overall sense and one-to-one at any given instant.

An efficient and versatile way of achieving memory management is through a technique which involves translation of the CPU's address signals before they reach the actual memory hardware. The 80386 chip contains built-in hardware to support this method. There are also add on devices available to bring this capability to 80286 and 8088 based systems.

Figure 1-4 illustrates, in simplified form, the inner workings of an 80386 chip - showing how the MMU gets first chance at the address asserted by the CPU portion of the chip. For the most part, this book will discuss the operation of the 80386 chip. Specific differences that pertain to the 80286 and 8088 will be stated where appropriate.

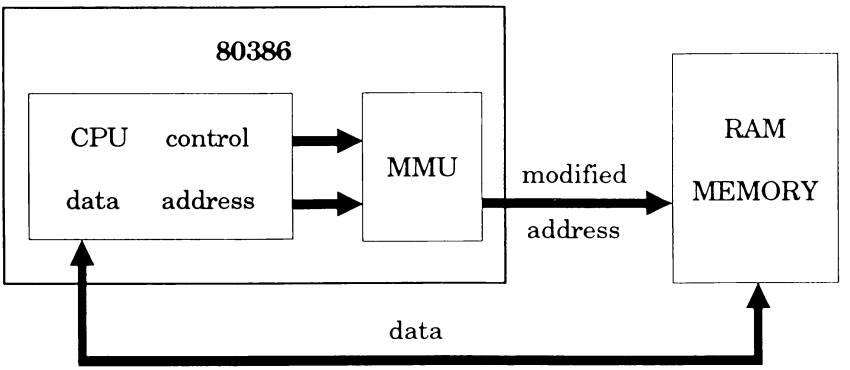

Figure 1 - 4

Under control of the CPU, the on-chip memory management hardware can modify the CPU's address signals before they reach the outside world.

Two terms often used to describe the type of memory management used in MOS are memory paging and remapping. The meaning of these two terms is considered synonymous for our purposes. Paging (or remapping) is possible in an 80386 system when the $386.SYS memory management driver is used. As mentioned above, paging is also available when the appropriate add on hardware and corresponding software driver is used in 80286 and 8088 based systems (see Chapter 3 for more details on memory management drivers).

To say that memory is "mapped" simply means that each actual physical memory element is assigned a unique logical address. In a simple system where there is no memory management, the mapping of memory is determined by the wiring of the address decoding circuits, and, in some cases, by DIP switches or jumpers. Regardless of the method used to define the mapping, it is a fixed relationship once your computer is booted up.

When memory management hardware is present it is possible to change the mapping of memory to dynamically re-define the logical address by which a physical memory address element can be accessed. In order to better understand the concept of memory paging, you may find it helpful to consider what can be done with pages in a three-ring binder. When you first receive a ring bound book, the pages are in an initial default order. By pulling the rings open you can remove a group of pages from its initial location and re-insert it into a different one. For instance, you could move pages 50 through 59 to be in between pages 19 and 20.

How does this apply to MOS and the support of multiple tasks? When each task is given a slice of execution time, the program code and operating data for that task must reside within the base 640K of the system's memory space. However, trying to keep too much in this one area at the same time is not practical. Through remapping, MOS can shuffle pages of memory around at a high rate of speed and give each task a more respectable TPA size.

The memory which is initially mapped into the base 640K area, that which is available when there is no memory management capability, can comprise the memory for one of our tasks - the foreground task. When memory is required for additional tasks it must be available in the form of extended memory. Extended memory is that which is initially addressable in the address region above the first megabyte; outside the range that a processor running in an 8086 type of mode can directly access.

When an 8086 mode of operation is used, as is the case when MOS operates the 80386 in its virtual 8086 mode, memory located above the 1 megabyte boundary is not directly addressable. MOS's memory management device driver acts as a system supervisor, using a blend of VM86 mode (virtual 8086 mode) and 386 protected mode operations to provide memory remapping capabilities.

Through control of the memory management unit's address translation circuitry the CPU can logically access a block of extended memory as if it were actually within a portion of the 1st megabyte region. Any physical memory which is normally mapped into this location in the 1st megabyte is, of course, still present in the system; it is just dormant while extended memory is mapped over it.

If it were not possible to dynamically remap memory, multiple full size tasks could not be supported without resorting to a swapping scheme (e.g. disk swapping or swapping from expanded memory). Although the 80386 chip supports paging to and from a disk, MOS does not use this method as paging extended memory is much faster and simpler. Re-mapping of extended memory pages can occur at a very high speed.

Unlike swapping techniques, there is never any need to copy the contents of memory from one place to another. When one task's share of processor time has expired there is no need to do anything but map the memory for the next task into place over the memory of the current task. Each time a task switch occurs and a new set of memory pages is mapped into selected portions of the first megabyte region, the previous mapping is automatically displaced.

In an 80386 based system, a page of memory consists of 4096 bytes (4K), this size being fixed by the internal circuitry of this CPU. The term "page frame" is used to describe each specific slot in the memory address map into which a page of memory may be accessed -- where it can be mapped and logically addressed. The region from physical address 0 through 0FFF (0 to 4095 in decimal, the first 4K) comprises the first page frame. Addresses 1000 through 1FFF make up the second page frame with this pattern continuing throughout the remainder of the system's address map. The first address of each page frame is a value which is a whole multiple of 4096. A page frame is thus said to be aligned on a 4K boundary.

Memory management involves the maintenance of what are called page tables. A page table contains one entry for each page frame with each entry designating which memory page is mapped into the frame. When the page table is initialized, a one to one correspondence is established between physical and logical addresses. What will appear at logical address 60000 will be what is physically allocated at 60000.

In a system with 640K of base memory, this will be usable RAM. Likewise, what will appear at D0000 will be what is actually at D0000. Therefore, if no actual memory exists at address D0000 then no memory will be available on the logical side either. This is typically the case in this higher address region.

When a change is made in the page table entry which controls what logically appears in the 4K block beginning at address 40000, all references to addresses 40000 through 40FFF will be re-directed to the corresponding addresses in a 4K block of RAM which is actually located in extended memory. The memory map in figure 1-5 depicts a situation where the first 4K page of extended memory has been remapped into the first 4K page frame of the TPA area. In actual practice, a set of extended pages will be remapped into page frames within the TPA. A task of only 4K would not be too useful.

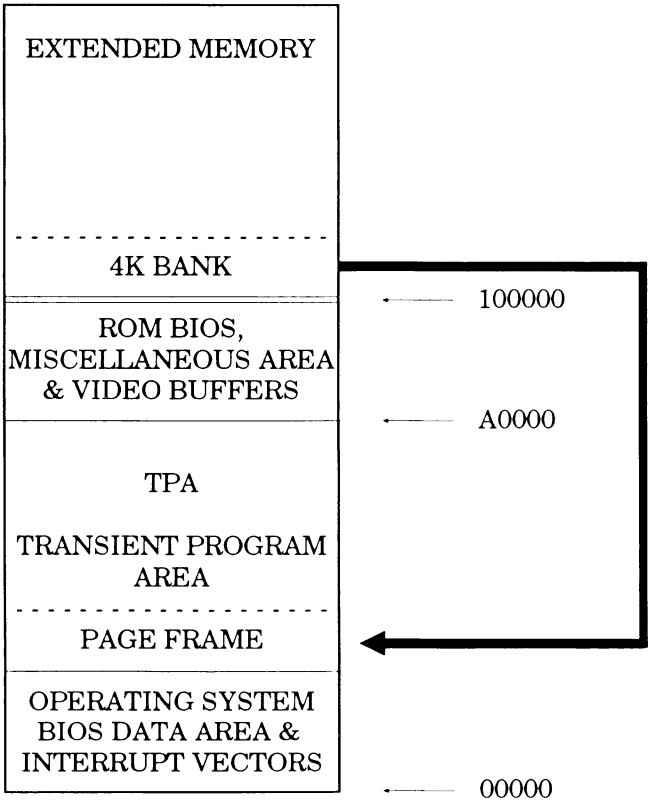

Figure 1 - 5

Through the control of the MMU, a 4K bank of extended memory can be remapped into a page frame within the TPA.

FREEMEM - Relocating MOS into High Memory

By taking advantage of the paging capabilities of memory management, a PC-MOS system can, in many situations, provide an even larger area for applications than PC-DOS. This is possible because pages of extended memory can be remapped into a typically unused area within the 1st megabyte region so that the operating system's code can be loaded there instead of using so much of the valuable base memory. This unused area is within the upper part of the 1st megabyte, in the address region from C0000 to F0000. Figures 1-3 and 1-5 show this area labeled as the "miscellaneous area".

The term FREEMEM is used to refer to address regions which are available for this special use. The size and number of these gaps in the memory map will vary depending on the machine and on any add-on hardware which may be installed. The ideal case is to have the entire 192K range from C0000 to F0000 available. There are also special cases in which other areas will be available as will be detailed in Chapter 2.

Figure 1-6 shows the effect that mapping extended memory into the FREEMEM area has on the TPA. Locating the system overhead completely out of lower memory area often results in the TPA's lower boundary being positioned even lower than in the PC-DOS case. As we will see in the next chapter there are also ways that the TPA's upper boundary can be raised through another memory management trick.

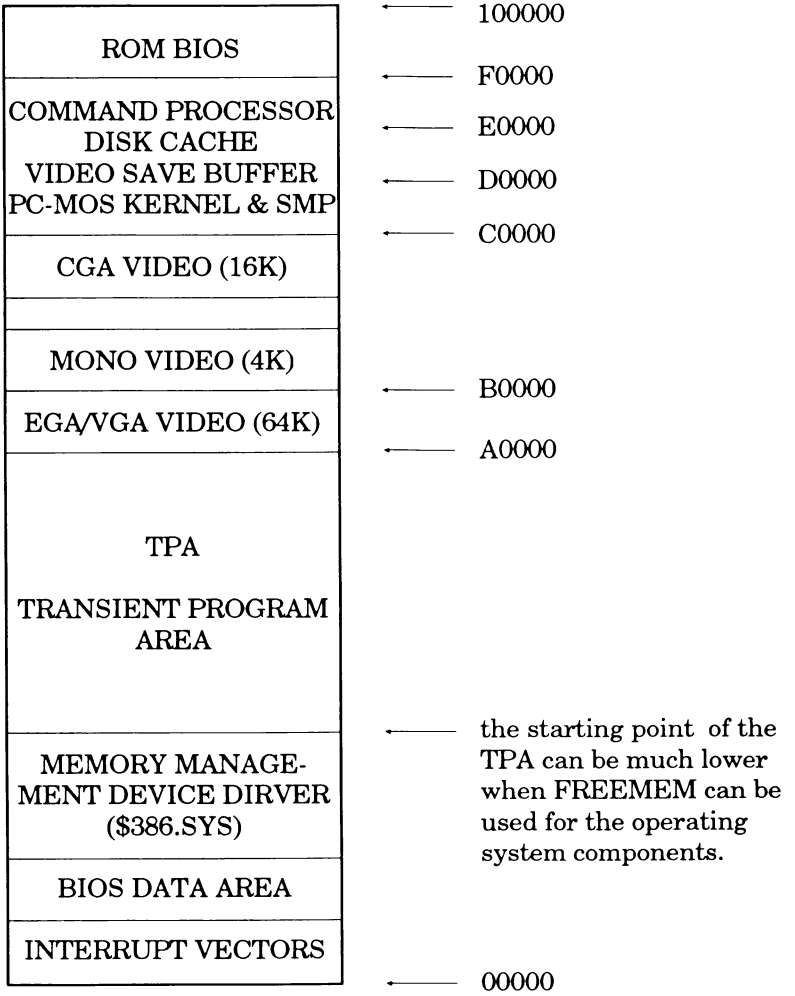

Figure 1 - 6

Memory map of PC-MOS when memory management is being used. What was formerly the MISCELLANEOUS AREA, from C0000 to F0000, can be used to hold MOS's system components.

PC-MOS's System Components

Maximizing the benefit attainable through the use of FREEMEM will be covered in detail in Chapter 2. To close out our introductory examination of memory management and how it affects the loading of the PC-MOS operating system, we'll examine what MOS's system components are. In order to do the best job of optimizing the initial load process, its important to be familiar with exactly what is being loaded in.

PC-MOS is comprised of a number of components, many of which have counterparts in DOS. MOS also has some components which do not have counterparts in PC-DOS such as the memory management driver and the video buffers. The following section introduces each of these components and illustrates its similarities and differences with its DOS counterpart, where applicable. Most of these components are loaded into upper memory (FREEMEM) whenever possible.

Memory Management Driver

One of several different memory management drivers can be used in a MOS system depending on the particular hardware configuration of the system. In the case of an 80386-based system, the file $386.SYS contains the program code and data for the memory management driver.

$386.SYS makes the advanced system control features of the Intel® 80386 chip available. This includes paging (the remapping of memory), I/O trapping and the processing of certain types of processing exceptions. Paging is what makes it possible to support multiple full-size tasks and make use of FREEMEM.

The page tables that dictate which block of memory is mapped into which page frame are located within the memory management driver along with the program code to manage these tables.

The memory management driver is the one system component which is never relocated up into a FREEMEM area. This driver is always located at the low end of the base 640K base memory, just above the BIOS data area. A standard PC-DOS system has no equivalent to MOS's memory management driver. Further detail on memory management drivers is provided in Chapter 3.

$$MOS.SYS

The contents of the file $$MOS.SYS comprise the operating system kernel. This module is most similar to the IBMDOS.SYS and IBMBIO.SYS files of PC-DOS. One important difference is that MOS's kernel file does not need to have hidden and system attributes and it does not need to be located in a certain position on the disk. You can just copy it into the root directory of the boot drive as any normal file. See Chapter 5 for details on making a bootable volume.

The operating system kernel serves two different kinds of functions at two different times. When a computer system is first booted up, the kernel manages the following functions:

1. The loading and initialization of the memory management driver

2. The initial configuration of the system hardware

3. The loading of device drivers specified in CONFIG.SYS

4. The invocation of the command processor shell

Once the initial boot up process has completed, the program code and data responsible for those functions is discarded. The remaining portion of the kernel manages the on-going runtime operations of the system. This includes:

1. Responding to system calls made by the command processor and applications programs

2. Communicating with workstations

3. Adding and removing tasks and maintaining each task's virtual machine environment

4. Managing IRQ type interrupts - for serial communications and special purpose devices

$$SHELL.SYS AND COMMAND.COM

The files $$SHELL.SYS and COMMAND.COM work in conjunction to comprise MOS's command processor and are similar to DOS's COMMAND.COM. The command processor interprets user input from the command line as well as from batch files.

As you'll notice by the file size, MOS's COMMAND.COM is quite small - on the order of 20 bytes. In MOS, the actual program code for the command processor is contained in the $$SHELL.SYS file. COMMAND.COM exists to provide an entry into the command processor in a way which is compatible with what a number of applications expect. While each task contains its own individual command processor data area, all tasks share one copy of the program code in $$SHELL.SYS.

SMP

The SMP (System Memory Pool) is roughly equivalent to DOS's device driver loading area. However, in MOS the SMP is also used to hold other items such as data structures the kernel must maintain on each task and temporary buffers involved in certain file operations. Sizing of the SMP is discussed in Chapter 2.

Disk Caching Buffers

PC-MOS comes with a built-in disk caching subsystem. Two buffer areas are associated with this feature which occupy memory space. The disk cache descriptor buffer is relatively small and will vary in size depending on the setting of certain caching parameters. The main disk cache buffer area is actually only one 4K page frame when paging capable memory management support exists. This 4K area is used as a window into which blocks from the actual extended memory disk cache are mapped. When paging support is not available, the main disk cache buffer area must exist entirely within a portion of the base 640K memory area. See Chapter 5 for more information on MOS's caching subsystem.

Video Areas

PC-MOS uses a pair of video buffers to maintain a virtual machine environment for each task. In addition, a memory area which is used in conjunction with these buffers is known as the master video context area. Just as with the main cache buffer, the master video context area changes the way it affects memory allocation depending on the availability of paging support.

In a paging environment, the master video context area acts as a mapping window for a special video save buffer for each task. When there is no paging support, there is no central master video context area and each task's video save buffer must be allocated from that task's memory. Virtualization of video is covered in more detail in Chapter 3.