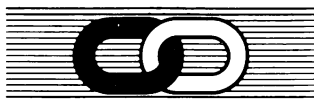


PC-MOSTM

MODULAR OPERATING SYSTEM

TECHNICAL REFERENCE MANUAL



a product of

THE SOFTWARE LINK, INCORPORATED

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

© Copyright 1987/1990 The Software Link, Incorporated
All Rights Reserved Worldwide

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

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


Hercules is a trademark of Hercules Computer Technology, Inc.

IBM is a registered trademark of International Business Machines Corporation

Intel is a registered trademark of Intel Corporation

LANLink is a trademark of The Software Link, Inc.

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



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

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

SunRiver is a registered trademark of SunRiver Corporation.

Televideo is a registered trademark of Televideo Systems, Inc.

WYSE 60 is a trademark of WYSE Technology.



Copyright Notice

Copyright 1987/1990, by The Software Link, Incorporated. All rights reserved worldwide.

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.

H190

PC-MOS TECHNICAL REFERENCE MANUAL

Table of Contents

CHAPTER 1: OVERVIEW 1 - 1

Why PC-MOS was Created	1 - 1
What PC-MOS Provides Today	1 - 1
The Future of PC-MOS	1 - 3
What it Means to You	1 - 3

CHAPTER 2: THE PC-MOS KERNEL 2 - 1

Introduction	2 - 1
Bootstrap Procedure	2 - 1
Interrupt Handling	2 - 4

CHAPTER 3: MEMORY ORGANIZATION 3 - 1

Introduction	3 - 1
Context Save Area	3 - 1
Video Areas	3 - 2
Memory Map without Memory Management	3 - 3
Memory Map in Systems with Memory Management	3 - 5
DOS Compatibility Considerations	3 - 7
Native Mode Considerations	3 - 8

CHAPTER 4: PROCESSOR DIFFERENCES 4 - 1

Differences between 80386 and Non-80386 Systems	4 - 1
---	-------

CHAPTER 5: INTERFACING APPLICATIONS 5 - 1

Introduction	5 - 1
16-bit Applications	5 - 1
32-bit Applications	5 - 2

CHAPTER 6: THE MULTI-TASKING ENVIRONMENT . 6 - 1

Introduction	6 - 1
File Sharing	6 - 2
Machine Sharing	6 - 4
I/O Programming	6 - 5
Efficient Console Input	6 - 8
Automatic Task Suspension	6 - 10
Direct Control of DIS Mode	6 - 12
Working with INT 8H and INT 1CH	6 - 14
Using MOS's Services for Time Measurement	6 - 16
Working with INT 9H	6 - 16
Intertask Communication	6 - 18
TSR Programs	6 - 19
General Product Design Considerations	6 - 19

CHAPTER 7: FILE I/O AND STANDARD DEVICES . . . 7 - 1

File and Path Names	7 - 1
File Handles	7 - 2
The Console Device	7 - 5
Other Standard Devices	7 - 9

CHAPTER 8: SYSTEM CALLS 8 - 1

Introduction	8 - 1
Calls Provided for Compatibility Only	8 - 3
Interrupt 21H (in detail)	8 - 4
Extended Error Codes for Int 21H & D4H	8 - 98
Standard Error Codes for Int 21H & D4H	8 - 99
The Extended Services Interrupt	8 - 110
Extended Services (in detail)	8 - 111

CHAPTER 9: RAM DATA STRUCTURES 9 - 1

Accessing MOS's Data Structures	9 - 1
The System Memory Pool	9 - 1
System Control Block	9 - 3
Task Control Block	9 - 3
Current Directory Block	9 - 7
Global File Block	9 - 8
Task File Block	9 - 9
Record Lock Block	9 - 10
Block Device Block	9 - 11
File Control Block	9 - 12
Program Segment Prefix	9 - 13
Native Context Area	9 - 14

CHAPTER 10: DISK DATA STRUCTURES 10 - 1

Introduction	10 - 1
Boot Sector	10 - 2
File Allocation Table	10 - 3
Directory Structure	10 - 7
Master Boot Record	10 - 9

CHAPTER 11: PROGRAM FILE STRUCTURES 11 - 1

COM Files	11 - 1
EXE Files	11 - 2

CHAPTER 12: DEVICE DRIVERS12 - 1

Introduction	12 - 1
The Device Header	12 - 4
The Request Header	12 - 10
Device Driver Functions	12 - 14
Device Driver Design Considerations	12 - 23
Effects on Task Switching	12 - 24
Using a Task Table	12 - 25
Terminal Interface	12 - 27
Functions	12 - 31
Serial Device Interface	12 - 49
Functions	12 - 54
NETBIOS Interface	12 - 80
Functions	12 - 88
Error Recovery	12 - 90
Driver, calls supported	12 - 91
Memory Management Interface	12 - 95
Functions	12 - 97
INT 6 Handler	12 - 106
Using \$PIPE.SYS	12 - 107

CHAPTER 13: PROGRAMMING TECHNIQUES13 - 1

Introduction	13 - 1
Accessing System Data Structures	13 - 1
The PUSHF/FAR CALL Calling Method	13 - 3
Program Initialization	13 - 4
Determining the Data Access Method	13 - 7
Adding a Task	13 - 7
Establishing Raw Mode	13 - 8
The CHK_PORTNUM Procedure	13 - 9
The Wait For Event Call	13 - 10
Sample Program Listing	13 - 11

INDEX I - 1

CHAPTER 1: OVERVIEW

Why PC-MOS was Created

PC-MOS™ was created with three goals in mind:

1. Compatibility with existing DOS applications
2. Enhanced power, efficiency, and flexibility
3. Usability

What PC-MOS Provides Today

PC-MOS enhanced features include:

- multi-tasking support
- multi-user support
- 80386 native mode support
- user task, directory and file security
- print spooling
- volume & file sizes up to 256 megabytes

Many other unique features have been implemented while others will be developed as the product matures.

A great deal of effort was spent selecting the best balance of functionality for PC-MOS. Need, efficiency, power, consistency and simplicity were all considered.

CHAPTER 1

The Intel® 80386 microprocessor is especially desirable to use with PC-MOS. The following features of this processor make the key difference:

Virtual-8086 Mode

This mode provides the ability to run DOS-compatible applications while providing a more powerful, protected environment, including I/O protection.

Paging

A vital companion to the Virtual-8086 mode, this feature allows multiple 8086 applications to share the same address space, thereby providing multiple, concurrent "640K" applications.

32-Bit Protected (Native) Mode

The 80386 can handle 32-bit quantities in a single operation. Its predecessors worked, at best, with 16-bit quantities. This is a performance benefit above and beyond the increased clock speed of the 80386 versus the 80286. It is most noticeable where there are many computations involving numbers larger than 16-bits, or where there is much in-memory moving and comparing of data strings.

The 80386 also supports memory protection, I/O port protection, and 32-bit addresses which allow for much larger code and data segments.

The Future of PC-MOS

Like all software, PC-MOS will mature. This provides you, the software developer, a rare opportunity. You can help guide the evolution of PC-MOS by submitting your suggestions and needs to us. This is an open doorway for providing your own product with the features necessary to meet your customers' needs and to gain a competitive edge.

It cannot be guaranteed that your suggestions will become reality, but we do listen and react to market need.

What it Means to You

PC-MOS provides you, the software developer, with the power of the 80386 today. You can convert your application to take advantage of the native mode capability of PC-MOS with relative ease. Changing register references and data definitions is the fundamental conversion procedure.

Your existing customers can maintain their current investment in your software, while allowing them to upgrade to the more powerful version when you've got it ready. Your customers can have the best of the present and the future.

This page intentionally left blank.

CHAPTER 2: THE PC-MOS KERNEL

Introduction

The kernel is the "core" of MOS. It is loaded at bootup from the file `$$MOS.SYS` on the boot disk. Except for some code and data used only for initialization, it remains resident in memory until the computer is rebooted or turned off.

The kernel performs many functions, but all of them fall into one of two categories:

1. System Initialization (Bootstrap)
2. Handling System Interrupts

The following two sections discuss these categories in detail.

Bootstrap Procedure

It is unnecessary for a bootable disk to require that operating system files reside in specific areas of the disk. MOS allows the user to put the MOS kernel on his disk just as if it were an ordinary file.

MOS does not support booting from a disk with a sector size less than 512 bytes.

The only other special requirement of a bootable disk is that the appropriate logic be put into sector zero (the boot sector); this is the purpose of the `.MSYS` command.

CHAPTER 2

Specifically, bootup involves the following steps:

1. The ROM BIOS (operating in Real Mode) loads the boot sector (the first sector of the boot drive) to address location 07C00, and transfers control to it with CS:IP set to 0000:7C00.
2. The code from the boot sector relocates itself to an area just below the 128K boundary, to address 1FB00 (MOS cannot boot in less than 128K).
3. The relocated routine looks for the file \$\$MOS.SYS in the root directory of the boot disk; if the file is found, its first cluster is read into memory location 00700, and control is passed to that code with CS:IP set to 0070:0000.
4. The remaining clusters of \$\$MOS.SYS are loaded, at which point the kernel is fully in memory.
5. The kernel then relocates itself to the top of available RAM and initializes interrupt vectors.
6. If \$\$MASTER.SYS exists in the root directory of the boot disk, the master password is acquired from the console.
7. CONFIG.SYS parameters are read into memory (this works even if the file is secured).
8. The MEMDEV driver, if specified, is loaded at address 00700 and initialized as a standard device driver. If the driver is \$386.SYS, it will put the 80386 processor into protected mode and return control to the kernel in Virtual 8086 mode.

9. The kernel relocates itself once more, to its final resting place. If a MEMDEV driver is present and sufficient FREEMEM space is available, then RAM is allocated and mapped into the FREEMEM area for it (FREEMEM is normally the unused address space between the video RAM and the ROM BIOS). Otherwise it relocates to the lowest available memory address (just after the MEMDEV driver, if one exists). Interrupt vectors are modified as required to suit the new location.
10. Additional RAM is allocated and initialized for the System Memory Pool (SMP), disk buffers, and video buffers. FREEMEM and/or extended memory are used for these when possible.
11. Built-in device drivers are initialized. External drivers are loaded into the SMP and initialized. The resident command interpreter, \$\$\$SHELL.SYS, is loaded.
12. Task 0 is initialized. The program COMMAND.COM is loaded into task 0 and is given control (it just transfers to the resident command interpreter; some of it remains resident as a data area for the task).

Note that MOS is designed in such a way that most of the resident code specific to the 80386 microprocessor is confined to the MEMDEV driver \$\$\$SHELL.SYS.

CHAPTER 2

Interrupt Handling

On 80386-based systems the \$386.SYS driver intercepts all interrupts, and then in most cases passes control on to the routine addressed by the corresponding 8086-level interrupt vector in low RAM. The discussions in this section apply only to interrupt handling at the 8086 level, and are relevant to all microprocessors supported by MOS. Features of the operating system peculiar to the 80386 microprocessor are covered separately in a later chapter.

The MOS kernel modifies various interrupt vectors at initialization time in order to gain control of those interrupts. In fact, interrupt vectors are the only means by which the kernel may receive control after it is initialized. Therefore it is important that the assembler-level programmer have an understanding of which interrupts are trapped by the kernel, and for what purposes.

Many interrupts that MOS intercepts were more properly the domain of the ROM BIOS, and were originally intended to be called only by the operating system.

However, because of the simplicity of (and deficiencies in) previous operating systems, it became common and accepted practice for applications programmers to call the BIOS interrupts directly in order to attain desired features and performance levels. And because many of these interrupts deal with task-specific information (for example, keyboard and display parameters) it is absolutely vital for any multi-tasking operating system to take over those interrupts if it wants to support such applications.

THE PC-MOS KERNEL

The following table summarizes the system interrupts that are handled by the MOS kernel.

Interrupt Number	Vector Address	Description
00	000-003	Divide by zero
08	020-023	IRQ0 - System timer
09	024-027	IRQ1 - Keyboard hardware
0A	028-02B	IRQ2 - Multi-purpose
0B	02C-02F	IRQ3 - Communications (COM2)
0C	030-033	IRQ4 - Communications (COM1)
0D	034-037	IRQ5 - Multi-purpose (often hard disk)
0E	038-03B	IRQ6 - Diskette drive
0F	03C-03F	IRQ7 - Printer (seldom used)
10	040-043	Video I/O
13	04C-04F	Disk I/O
14	050-053	Serial I/O (\$SERIAL.SYS)
15	054-057	Miscellaneous services
16	058-05B	Keyboard I/O
17	05C-05F	Printer I/O
1A	068-06B	Time of day services
1C	070-073	Timer tick handler
20	080-083	Terminate application
21	084-087	Basic operating system services
22	088-08B	Program terminate address
23	08C-08F	Program control-break exit address
24	090-093	Program critical error exit address
25	094-097	Read disk sector(s)
26	098-09B	Write disk sector(s)
27	09C-09F	Terminate and retain application in memory

CHAPTER 2

28	0A0-0A3	Safe entry interrupt
2E	0B8-0BB	Captured for compatibility
2F	0BC-0BF	Multiplex interrupt
38	0E0-0E3	Extended operating system services (old vector)
70	1C0-1C3	IRQ8 - Interval timer
71	1C4-1C7	IRQ9 - BIOS redirected to interrupt 0A
72	1C8-1CB	IRQ10 - Undefined
73	1CC-1CF	IRQ11 - Undefined
74	1D0-1D3	IRQ12 - Undefined
75	1D4-1D7	IRQ13 - NMI
76	1D8-1DB	IRQ14 - Undefined
77	1DC-1DF	IRQ15 - Undefined
D4	350-353	Extended operating system services (new vector)

Interrupts not included in the preceding table are either initialized and completely handled by the BIOS, initialized by MOS to point to an IRET instruction, or left uninitialized. Some may be reserved for non-kernel MOS utilities or even applications software, and are therefore documented elsewhere.

The following discussion covers specific interrupts in more detail:

Interrupt 0 is trapped by MOS in order to provide a default handler for divide overflow conditions. If a divide overflow interrupt occurs and the application has not installed its own handler, MOS will terminate the program with an appropriate message.

Interrupts 8 through 0FH correspond to hardware interrupts IRQ0 through IRQ7, respectively - they are initiated by hardware external to the microprocessor, and may occur at any time. MOS will normally not permit a user application to directly trap a hardware interrupt vector, largely because in a memory-managed multi-tasking environment the application's handler (or information required by it) might not be available when the interrupt occurs.

If the application does try to replace a hardware interrupt vector, MOS will save the address of the application's handler in the TCB and then change the vector so that MOS retains control of the interrupt. When the interrupt occurs, MOS can then make sure that conditions are right before giving control to the application's handler. This usage of a hardware interrupt by an application generally requires that the MOS USEIRQ command first be issued (MOS USEIRQ is not needed for device drivers that trap interrupts), though in many situations MOS can determine what to do with an interrupt without help from MOS USEIRQ.

Interrupt 8 is issued approximately 18.2 times per second. It is trapped by MOS to ensure that task selection and a few other internal routines are serviced regularly. The timer hardware should never be disabled, as MOS cannot run without it. See Chapter 6 for more information on this interrupt.

Interrupt 9 is issued by the keyboard hardware every time a key on the main console is pressed or released, or if a key is pressed long enough for the "auto-repeat" feature to engage. It is trapped by MOS to support the console driver.

CHAPTER 2

In addition, MOS goes to considerable lengths to handle applications that trap into the interrupt 9 vector, particularly when these applications are not associated with the console, i.e. if a key is hit at a terminal attached to a serial port, MOS will (if possible) simulate the situation created by the computer's keyboard hardware for that application. See Chapter 6 for more information on this interrupt.

Interrupt 0AH is caused by hardware interrupt level IRQ2 (or IRQ9 on AT-class machines). It does not have a "standard" assignment, but is frequently used for multi-ported serial adapters in multi-user systems. It may be selected for use by the \$SERIAL.SYS driver.

Interrupt 0BH is for hardware interrupt level IRQ3, normally used by serial port COM2 (I/O address 2F8) and may be trapped by \$SERIAL.SYS.

Interrupt 0CH is for hardware interrupt level IRQ4, normally used by serial port COM1 (I/O address 3F8) and may be trapped by \$SERIAL.SYS.

Interrupt 0DH (IRQ5) is often trapped by firmware (ROM) supporting a hard disk. If your hard disk does not use IRQ5, then you may use it for other hardware such as multi-ported serial adapters. This interrupt is also used internally in 80386 systems to trap various types of exceptions. This usage does not conflict with its support of IRQ5.

Interrupt 0EH (IRQ6) is commonly used by diskette controllers, and is usually not available for other purposes.

Interrupt 0FH (IRQ7) is reserved for use by the parallel printer adapter. However, in most systems neither printer drivers nor firmware ever enable this feature, and IRQ7 may be used with impunity for other hardware (such as serial ports).

Interrupt 10H is the BIOS video interface. A large portion of the MOS kernel is dedicated to performing appropriate actions when this interrupt is invoked by applications.

Interrupt 13H is the BIOS-level interface to diskette and hard disk drives. MOS intercepts it in order to resolve certain limitations of the DMA channel used by the disk controller, including the fact that the DMA channel has no communication with the paging feature of the 80386, nor with hardware memory management similar to that of the AT-GIZMO™. MOS does depend on the original BIOS for actually performing disk I/O.

Interrupt 14H is the MOS and applications interface to serial ports. This interface is supported by the \$SERIAL.SYS driver. It is buffered, interrupt-driven, device-independent and fast, and is considerably enhanced beyond the standard BIOS offering. If you write software that accesses serial ports, you must support this interface in order to avoid restrictions on its use with MOS. See Chapter 12 for details about using \$SERIAL.SYS.

Interrupt 15H supports several system functions related to event waits, access to extended memory, and access to the System Request key. MOS intercepts these for compatibility purposes and to make it possible to overlap task processing with disk I/O wait time.

CHAPTER 2

Interrupt 16H is the BIOS-level keyboard I/O interface. MOS provides appropriate support and emulation of these functions as they relate to the task's workstation.

Interrupt 17H is the BIOS-level parallel printer interface. It is intercepted by MOS to provide overlap of printer-busy time with task processing, to support the spooler, and for printer device redirection.

Interrupt 1AH provides time of day services in the same manner as the ROM BIOS version of this service.

Interrupt 1CH is issued by the BIOS interrupt 8 (IRQ0) handler at each timer interrupt. MOS intercepts it to provide a default handler, and to ensure that order is maintained when an application's handler is installed.

Interrupt 20H is equivalent to interrupt 21H function 0 (terminate application).

Interrupt 21H is the basic interface between the operating system and user applications. It also provides compatibility with applications written to work with previous operating systems for PC's and compatibles. See "System Calls" for details.

Interrupt 22H is not actually used as an interrupt; its vector contains the address in the invoking process to which control returns when an application terminates. This field is maintained by MOS.

Interrupt 23H is used to point to the application's control-break handler. MOS provides a default control-break handler.

Interrupt 24H is used to point to the application's critical error handler. MOS provides a default error handler.

Interrupt 25H is provided by MOS for sector-level disk reads. See "System Calls" for details. Use of this interrupt is strongly discouraged, as it will cause incompatibilities with LAN environments.

Interrupt 26H is provided by MOS for sector-level disk writes. See "System Calls" for details. Use of this interrupt is strongly discouraged, as it will cause incompatibilities with LAN environments.

Interrupt 27H (Terminate and Stay Resident) is provided for compatibility with applications written for earlier operating systems. It is preferable to install resident code as either a device driver or via interrupt 21H function 31H. The device driver method is necessary if the resident code contains a polling routine for task selection, and highly desirable if it contains code to handle hardware interrupts. These are because "TSR" programs may be swapped out of addressable memory when other tasks are processing, whereas device drivers are not. See "System Calls" for details about the interrupt 27H call.

Interrupt 28H provides an indication as to when TSR type programs may safely make system calls to the MOS kernel.

Interrupt 2EH is trapped by MOS to prevent attempts at illegal entry into the command processor.

Interrupt 2FH provides an indication that SHARE is loaded. Note that under PC-MOS, there is actually no separate SHARE module since the file sharing logic is intrinsic to the kernel.

Interrupt 38H provides additional interfacing to user applications that is unique to MOS. In most cases the functions supported by interrupt 38H relate to either multi-tasking or to 80386 native mode support. See "System Calls" for details. NOTE: all future developments should use vector D4 instead.

Interrupts 70H-77H are supported on AT-class machines for hardware interrupts IRQ8-IRQF, respectively. The above general discussion for interrupts 8-0FH (IRQ0-IRQ7) applies also to IRQ8-IRQF.

Interrupt D4H provides additional interfacing to user applications that is unique to MOS. In most cases the functions supported by interrupt D4H relate to either multi-tasking or to 80386 native mode support. See "System Calls" for details.

NOTE: Interrupts 8 and 9 are handled in a unique manner by MOS. They are often trapped by more than one application within the same task. Because they can occur "asynchronously", without regard to which task is currently processing, MOS must maintain first-round control of these interrupts. Therefore, MOS must be aware of all programs within a task that trap such interrupts. This is handled in a rather complex manner through an internal "chain table" which contains information about each level of trapping. Up to five such levels are supported per task.