

P C - **MOS**<sup>TM</sup>

## User Guide



# 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 the 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 this 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 this Program or this license to any other person without the express prior consent of TSL;
- f. you acknowledge that you are receiving only a limited license to use the program and the related documentation. You acknowledge that TSL retains title to the Program and documentation. You acknowledge that TSL has a valuable proprietary interest in the Program and documentation; and
- g. you may use the Program in a multiuser arrangement only if you license the multiuser versions of the Program and pay the applicable licence fee based on the number of users on the system.

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, 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 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) 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 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 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, Norcross, 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.**

## Trademark Notice

ALL CHARGE CARD is a trademark of All Computers, Inc.

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

CROSSTALK is a registered trademark of Digital Communications Associates.

Disk Manager is a registered trademark of Ontrack Computer Systems, Inc.

HAYES is a registered trademark of HAYES Microcomputer Products, Inc.

IBM is a registered trademark of International Business Machines Corporation.

Intel is a registered trademark of Intel Corporation.

Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation.

Maxpeed is a trademark of Maxpeed Corporation

Microsoft is a registered trademark of Microsoft Corporation.

MS-DOS is a trademark of Microsoft Corporation.

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.

ProComm is a trademark of PIL Software Systems.

WYSE-60 is a trademark of Wyse Technology

## **Copyright Notice**

Copyright 1987, 1988, 1989, 1990, 1991 The Software Link, Inc. All rights reserved worldwide. No part of this publication may be reproduced without the express written permission of The Software Link, Inc.

## **Notice**

This information package contains TSL policies, procedures, programs and pricing that are 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.

## **Thou Shalt Not Dupe**

### **Either Way It's Wrong**

People who would never walk into a store and shoplift a software product think nothing of making several copies of the same software. The results are the same. The act is just as wrong.

When it comes to unauthorized duplication of software, many people do not realize the costly impact on the software developer and the customer community. The relationship between customer and developer in a software transaction is one of mutual trust. The customer trusts that the developer has produced a product that will deliver the desired result, performs according to specifications, and is properly documented and supported. **The developer trusts that the customer will make use of only those copies for which he has purchased a license**, even though making additional, unauthorized copies is relatively easy.

Unauthorized duplication and use of software violates the U.S. Copyright Law, and unfairly deprives software developers of revenue they are entitled to receive for their work.

Software developers find that thousands of illegal copies have been made by customers who either innocently believe they are doing nothing wrong or simply choose to ignore the law.

### **The Law is Clear**

**Reproducing computer software without authorization violates the U.S. Copyright Law. It is a Federal offense. The money paid for a software product represents a license fee for the use of one copy. It does not represent an authorization to copy. Civil damages for unauthorized software copying can be as much as \$50,000 or more and criminal penalties include fines and imprisonment. Bills have been introduced in Congress to strengthen the law and increase penalties.**

## Myths and Facts of Software

Let's start by dispelling some myths with a few facts.

First, software developers *do not* condone unauthorized copying in order to gain market penetration.

Second, the price of software *does not* make unauthorized copying justifiable. The cost of a software product to a consumer represents only a small fraction of the publisher's marketing and developing costs.

Third, although the cost of *softlifting* is borne initially by the software developer, it is paid for ultimately by legitimate users.

## What It Means to You

It's obvious that legitimate software users are paying for theft along with software developers. It's obvious, too, that no one is going to put up with it for long.

Think twice before you ask someone to give you an illegal copy of their software. Think three times before you offer to do it for someone else. *Softlifting* or *software piracy* is not only a crime; it's simply wrong.

## Where to Get More Information

ADAPSO represents more than 750 corporate members that provide a wide spectrum of computer services and software including: micro, mini and mainframe software products; professional software services; batch and remote processing services; integrated hardware/software systems; education and training; and consulting.

For further information or questions on software protection, contact ADAPSO at Suite 300, 1300 North 17th Street, Arlington, VA 22209; or phone (703) 522-5055.

Copyright by ADAPSO, 1984. Permission to reproduce this article is granted by ADAPSO.

---

# HOW TO USE THIS MANUAL

---

This manual is provided as a guide for anyone who uses PC-MOS. You will find that MOS provides many options to let you customize the operation of your computer, and to maintain your information. The following is a brief summary of the information in this manual, and how it is organized.

**CHAPTER 1: OVERVIEW** - This chapter is an introduction to MOS. It explains the files provided with MOS, some special terms and how they are used, who the system administrator is and the responsibilities of that person, and how to make entries on the command line. Reading this chapter will give you a better understanding of MOS, and the rest of the information in this manual.

**CHAPTER 2: CONFIGURATION** - The command statements you may enter in the CONFIG.SYS file to define your environment are explained in this chapter. Following the commands is an explanation of the standard and optional device drivers provided with MOS.

**CHAPTER 3: FILES & DIRECTORIES** - MOS uses different types of files to execute commands and store information. This chapter explains how MOS treats files, and the directory structure that lets you organize and group files.

**CHAPTER 4: GENERAL COMMANDS** - The types of MOS commands and the conventions used to present them are explained in this chapter. The general processing commands in MOS are then explained in alphabetical order. Commands that have a specific function and require more detail are explained in the other chapters.

**CHAPTER 5: MULTI-TASKING/USER** - You will find all the detailed information to set up a multi-tasking and/or multi-user system with MOS in this chapter.

---

**CHAPTER 6: BATCH FILES** - The batch file commands that MOS recognizes as a job control "language" are explained in this chapter. Batch files are used to automate a sequence of commands and programs. This chapter provides many examples to help you write your own batch files.

**CHAPTER 7: MOS EDITOR** - The MOS Editor provides two modes of editing, a Command mode and a Visual mode. The modes of editing a file, editing commands and editing keys are explained in this chapter.

**CHAPTER 8: DEBUG** - MOS provides a .DEBUG utility that lets you make changes to existing programs and check for problems during execution. The use of DEBUG is explained in this chapter, along with the many commands that are available when .DEBUG is active. The use of DEBUG should normally be reserved to those with programming experience.

**CHAPTER 9: SECURITY** - The instructions and commands to initiate MOS security with security classes, User ID's and passwords are explained in this chapter, along with the effect security will have on the processing of files, directories and partitions on your computer.

**CHAPTER 10: PRINT SPOOLER** - If you have multiple users or multiple printers, you may want to use the MOS Print Spooler to control print operations. The commands to set up and use the Print Spooler are explained in this chapter.

**APPENDIX A: WARNING AND ERROR MESSAGES** - This appendix contains the warning and error messages issued by MOS, and what causes the warning or error to occur.

**APPENDIX B: NETBIOS EMULATION** - MOS provides NETBIOS emulation for intertask communication between applications. This chapter explains how to set up MOS to use NETBIOS emulation.

**APPENDIX C: CUSTOMIZING THE HELP FACILITY** - You may change or add to the text of the MOS Help Facility, which is the text that displays for the .HELP command. The instructions to modify the help facility are explained in this appendix.



---

**GLOSSARY** - This section contains an explanation of the terms used in this manual, listed in alphabetical order.

**INDEX** - An index is provided at the end of this manual to help you easily reference specific information.

**README** - There may be new features or commands in MOS that are not documented in this manual. Any new information is contained in the README file provided with MOS. To check this file for the latest information, place your MOS diskette in drive A and type the following command, including the periods:

.TYPE A:README | MORE

This will displays the contents of the file on your video screen, one page at a time. Checking this file will ensure that you are aware of all features and commands in MOS.

**This page intentionally left blank.**

---

# TABLE OF CONTENTS

---

## CHAPTER 1: OVERVIEW

Introduction . . . . .	1 - 2
Special MOS Features . . . . .	1 - 3
Optimization of Memory Managed Environments . . . . .	1 - 4
Support for non-Memory Managed Environments . . . . .	1 - 5
Video Handling . . . . .	1 - 6
VIDPATCH.COM . . . . .	1 - 6
MOS Files . . . . .	1 - 8
Understanding Terms in MOS . . . . .	1 - 10
The System Administrator . . . . .	1 - 12
Command Line Editing . . . . .	1 - 13
Command Recall Buffer . . . . .	1 - 13
Command Line Control Keys . . . . .	1 - 14

## CHAPTER 2: CONFIGURATION

Introduction . . . . .	2 - 2
Creating a CONFIG.SYS File . . . . .	2 - 2
Configuration Command Statements . . . . .	2 - 3
8087 . . . . .	2 - 4
CACHE . . . . .	2 - 5
COUNTRY . . . . .	2 - 15
DESNOW . . . . .	2 - 16
DEVICE & LDEVICE . . . . .	2 - 17
FREEMEM . . . . .	2 - 19
MEMDEV . . . . .	2 - 22
SHELL . . . . .	2 - 26

---

SLICE . . . . .	2 - 27
SMPSIZE . . . . .	2 - 28
USERFILE . . . . .	2 - 30
VTYPE . . . . .	2 - 31
Device Names . . . . .	2 - 37
Special MOS Device Drivers . . . . .	2 - 38
\$EMS.SYS . . . . .	2 - 38
\$MOUSE.SYS . . . . .	2 - 40
\$PIPE.SYS . . . . .	2 - 42
\$RAMDISK.SYS . . . . .	2 - 44
\$SERIAL.SYS . . . . .	2 - 46

## CHAPTER 3: FILES AND DIRECTORIES

Introduction . . . . .	3 - 2
Naming Files . . . . .	3 - 2
File Extensions . . . . .	3 - 3
Displaying File Names . . . . .	3 - 3
File Characteristics . . . . .	3 - 4
File Sizes . . . . .	3 - 4
Updated Date and Time . . . . .	3 - 4
File Attributes . . . . .	3 - 4
Class and User ID . . . . .	3 - 5
Creation Date and Time . . . . .	3 - 5
File Sharing . . . . .	3 - 5
Wildcard Characters . . . . .	3 - 5
File Maintenance . . . . .	3 - 7
Directories . . . . .	3 - 8
Directory Structure . . . . .	3 - 9
Directory Names . . . . .	3 - 10
Organizing Directories . . . . .	3 - 11
Directory Maintenance . . . . .	3 - 11
Displaying a Directory . . . . .	3 - 13
Displaying a Directory Structure . . . . .	3 - 14



---

## CHAPTER 4: GENERAL COMMANDS

Introduction . . . . .	4 - 3
Intrinsic and Extrinsic Commands . . . . .	4 - 3
Extrinsic MOS Commands . . . . .	4 - 4
Intrinsic MOS Commands . . . . .	4 - 5
Invoking Commands . . . . .	4 - 6
Understanding Command Conventions . . . . .	4 - 7
Input/Output Redirection . . . . .	4 - 9
General Commands . . . . .	4 - 11
.ADDDEV . . . . .	4 - 11
.ALIAS . . . . .	4 - 13
.BREAK . . . . .	4 - 16
.CD . . . . .	4 - 17
.CLS . . . . .	4 - 19
.COMMAND . . . . .	4 - 20
.COMPFILE . . . . .	4 - 22
.COPY . . . . .	4 - 25
.DATE . . . . .	4 - 32
.DIR . . . . .	4 - 33
.DIRMAP . . . . .	4 - 36
.DISKCOPY . . . . .	4 - 38
.DISKID . . . . .	4 - 40
.ENVSIZE . . . . .	4 - 42
.ERASE . . . . .	4 - 42
.EXCEPT . . . . .	4 - 45
.EXPORT . . . . .	4 - 49
.FILEMODE . . . . .	4 - 53
.FORMAT . . . . .	4 - 55
.HELP . . . . .	4 - 58
.IMPORT . . . . .	4 - 59
.MD . . . . .	4 - 61
.MORE . . . . .	4 - 63
.MSORT . . . . .	4 - 64

---

.MSYS . . . . .	4 - 67
.ONLY . . . . .	4 - 68
.PATH . . . . .	4 - 71
.PROMPT . . . . .	4 - 73
.RD . . . . .	4 - 76
.REL . . . . .	4 - 77
.REMDEV . . . . .	4 - 78
.RENAME . . . . .	4 - 79
.SEARCH . . . . .	4 - 82
.SET . . . . .	4 - 84
.TIME . . . . .	4 - 86
.TYPE . . . . .	4 - 88
.VERIFY . . . . .	4 - 90
.WVER . . . . .	4 - 93

## CHAPTER 5: MULTI-TASKING/MULTI-USER

Multi-Tasking Concepts . . . . .	5 - 4
Multi-Tasking Processing . . . . .	5 - 7
Multi-User Concepts . . . . .	5 - 7
Multi-User Processing . . . . .	5 - 8
Time Sharing . . . . .	5 - 8
Rebooting Tasks . . . . .	5 - 9
Multi-Tasking and Multi-User Commands . . . . .	5 - 10
.ADDTASK . . . . .	5 - 11
.REMTASK . . . . .	5 - 18
Partition Access . . . . .	5 - 19
Partition Access Keys . . . . .	5 - 19
.SWITCH Command . . . . .	5 - 20
Turning Partition Access On and Off . . . . .	5 - 20
MOS Utility Functions . . . . .	5 - 20
The .MOS Utility Commands . . . . .	5 - 21
.MOS MAP . . . . .	5 - 22
.MOS DIS . . . . .	5 - 23
.MOS NODIS . . . . .	5 - 23



.MOS USEIRQ . . . . .	5 - 23
.MOS FREEIRQ . . . . .	5 - 26
.MOS IRQ . . . . .	5 - 27
.MOS WAIT . . . . .	5 - 28
.MOS VMODE . . . . .	5 - 29
.MOS SERINIT . . . . .	5 - 30
.MOS ROUTE (COMn) . . . . .	5 - 31
.MOS ROUTE (LPTn) . . . . .	5 - 31
.MOS ROUTE (TERM) . . . . .	5 - 32
.MOS ROUTE (NOTERM) . . . . .	5 - 32
.MOS RESIZE . . . . .	5 - 32
.MOS INFO . . . . .	5 - 33
.MOS DSPORT . . . . .	5 - 35
.MOS MOUSE . . . . .	5 - 36
.MOS KEYB . . . . .	5 - 38
Foreign Keyboard Drivers . . . . .	5 - 39
.MOS FILES . . . . .	5 - 40
.MOS HOLD LPTn . . . . .	5 - 40
.MOS TSR . . . . .	5 - 41
.MOS ANSI . . . . .	5 - 42
.MOS DOSVER . . . . .	5 - 42
The .MOSADM Utility Command . . . . .	5 - 43
.MOSADM SLICE . . . . .	5 - 44
.MOSADM PRI . . . . .	5 - 45
.MOSADM CACHE . . . . .	5 - 46
.MOSADM SWITCH . . . . .	5 - 46
.MOSADM TMFACTOR . . . . .	5 - 46
.MOSADM HOLD LPTn . . . . .	5 - 47
.MOSADM VIRQ . . . . .	5 - 48
.MOSADM RESET . . . . .	5 - 48
.MOSADM TIME . . . . .	5 - 49
.MOSADM EMSLIMIT . . . . .	5 - 49
Connecting Terminals and Workstations . . . . .	5 - 52
Terminals vs Workstations . . . . .	5 - 52
Serial Ports . . . . .	5 - 53
Cable Connections . . . . .	5 - 53
Modem Cable Connections . . . . .	5 - 54

---

Terminal and Workstation Device Drivers . . . . .	5 - 55
Terminal Display Differences . . . . .	5 - 57
Escape Sequences for Non-PC Type Terminals . . . . .	5 - 58
Escape Sequence Chart . . . . .	5 - 61
.KEYMAP . . . . .	5 - 62
Modems . . . . .	5 - 64
MODEM.COM . . . . .	5 - 64
Defining Modem Types . . . . .	5 - 65
SunRiver Terminal Driver . . . . .	5 - 66
MOS System Monitor . . . . .	5 - 68

## CHAPTER 6: BATCH FILES

Introduction . . . . .	6 - 2
Batch File Features . . . . .	6 - 2
Creating Batch Files . . . . .	6 - 3
Batch File Commands . . . . .	6 - 3
.ABORT . . . . .	6 - 4
.AUTOCD . . . . .	6 - 5
.BATECHO . . . . .	6 - 6
.CALL/.RETURN . . . . .	6 - 7
.ECHO . . . . .	6 - 9
.FLUSH . . . . .	6 - 10
.FOR IN DO . . . . .	6 - 11
.GOTO . . . . .	6 - 13
.IF . . . . .	6 - 14
.INSERT . . . . .	6 - 17
.KEY . . . . .	6 - 18
.NEXT . . . . .	6 - 21
.PAUSE . . . . .	6 - 22
.REM . . . . .	6 - 23
.STOP . . . . .	6 - 24
.TEXT/.ENDTEXT . . . . .	6 - 25



---

Batch File Processing . . . . .	6 - 29
Automated Batch Files . . . . .	6 - 29
Nested Batch Files . . . . .	6 - 33
Replaceable Operand Files . . . . .	6 - 34
Compatibility . . . . .	6 - 36

## CHAPTER 7: MOS EDITOR - .ED

Introduction . . . . .	7 - 2
Invoking the Editor . . . . .	7 - 2
Editing in the Command Mode . . . . .	7 - 3
Command Mode Commands . . . . .	7 - 5
A - ASSIGN . . . . .	7 - 6
C - COPY . . . . .	7 - 7
D - DELETE . . . . .	7 - 8
E - END . . . . .	7 - 9
F - FILENAME . . . . .	7 - 10
H - HELP . . . . .	7 - 11
I - INSERT . . . . .	7 - 12
L - LIST . . . . .	7 - 13
M - MOVE . . . . .	7 - 14
P - PRINT . . . . .	7 - 15
Q - QUIT . . . . .	7 - 16
R - REPLACE . . . . .	7 - 17
S - SEARCH . . . . .	7 - 18
V - VISUAL MODE . . . . .	7 - 19
W - WRITE . . . . .	7 - 20
X - EXECUTE MACRO . . . . .	7 - 21
Editing in Visual Mode . . . . .	7 - 22
Visual Mode Editing Keys . . . . .	7 - 24
Using Macros . . . . .	7 - 26

---

## CHAPTER 8: DEBUG

Introduction . . . . .	8 - 3
.DEBUG Break points . . . . .	8 - 4
Entering .DEBUG Commands . . . . .	8 - 4
.DEBUG Commands . . . . .	8 - 6
A - ASSEMBLE . . . . .	8 - 7
AU - ASSEMBLE - UNASSEMBLE . . . . .	8 - 9
BC - CLEAR BREAKPOINT . . . . .	8 - 10
BD - BREAKPOINT DISABLE . . . . .	8 - 11
BE - BREAKPOINT ENABLE . . . . .	8 - 12
BL - BREAKPOINT LIST . . . . .	8 - 13
BS - BREAKPOINT SET . . . . .	8 - 14
C - COMPARE . . . . .	8 - 15
C? - DEBUG CONFIGURATION . . . . .	8 - 16
CO - CHANGE CONSOLE . . . . .	8 - 17
D - DUMP . . . . .	8 - 18
E - ENTER . . . . .	8 - 19
F - FILL . . . . .	8 - 21
G - GO . . . . .	8 - 23
H - HEX . . . . .	8 - 24
I - INPUT . . . . .	8 - 25
L - LIST . . . . .	8 - 26
M - MOVE . . . . .	8 - 27
N - NAME . . . . .	8 - 28
O - OUTPUT . . . . .	8 - 29
P - PROCEED . . . . .	8 - 30
Q - QUIT . . . . .	8 - 31
R - REGISTER . . . . .	8 - 32
S - SEARCH . . . . .	8 - 33
T - TRACE . . . . .	8 - 35
U - UNASSEMBLE . . . . .	8 - 36
V - VERIFY . . . . .	8 - 37
W - WRITE . . . . .	8 - 38



7 - COPROCESSOR . . . . .	8 - 40
\ - SWAP SCREEN . . . . .	8 - 41
! - SHELL . . . . .	8 - 42
" - PAUSE . . . . .	8 - 43
: - DELAY . . . . .	8 - 44
;* - REMARK . . . . .	8 - 45
? - HELP . . . . .	8 - 46
Using .DEBUG Commands . . . . .	8 - 47

## CHAPTER 9: SECURITY

Introduction . . . . .	9 - 2
How Security Works . . . . .	9 - 2
Initiating User Security . . . . .	9 - 3
Entering User Security Records . . . . .	9 - 4
Security Commands . . . . .	9 - 6
.CLASS . . . . .	9 - 7
.SIGOFF . . . . .	9 - 9
.SIGON . . . . .	9 - 10
Assigning Security to Your Items . . . . .	9 - 11
Partition Level Security . . . . .	9 - 12
Directory Level Security . . . . .	9 - 15
File Level Security . . . . .	9 - 17
MOS File Level Security and VERIFY Command . . . . .	9 - 18
Precautions for Security Users . . . . .	9 - 19
Advanced Security . . . . .	9 - 19
The MOS Encryption Key . . . . .	9 - 20
The Master Password Encryption Key . . . . .	9 - 20
Changing the Master Password . . . . .	9 - 22

## CHAPTER 10: PRINT SPOOLER

Introduction . . . . .	10 - 2
Print File Names . . . . .	10 - 2
Print File Extensions . . . . .	10 - 4
Disposition . . . . .	10 - 4
Priority . . . . .	10 - 5
Print Class . . . . .	10 - 5
MOS HOLD LPTn Commands . . . . .	10 - 6
PRINT.CTL File . . . . .	10 - 6
Print Spooler Commands . . . . .	10 - 7
.PRINT . . . . .	10 - 8
.SPOOL . . . . .	10 - 11
Initiating the Print Spooler . . . . .	10 - 15
Automating the Print Spooler . . . . .	10 - 16
The Print Spooler Menus . . . . .	10 - 17
Print Processor Menu . . . . .	10 - 17
Spooler Menu . . . . .	10 - 19
Multiple Printer Environments . . . . .	10 - 22
Print Spooler Example Batch Files . . . . .	10 - 22
Batch Files for Multiple Users . . . . .	10 - 22
Batch Files for Multiple Printers . . . . .	10 - 23

APPENDIX A: WARNING AND ERROR MESSAGES . . . . .	A - 1
--	-------

APPENDIX B: NETBIOS EMULATION . . . . .	B - 1
---	-------

APPENDIX C: CUSTOMIZING THE HELP FACILITY . . . . .	C - 1
---	-------

GLOSSARY . . . . .	G - 1
--------------------	-------

INDEX . . . . .	I - 1
-----------------	-------

---

# CHAPTER 1: OVERVIEW

---

Introduction . . . . .	1 - 2
Special MOS Features . . . . .	1 - 3
Optimization of Memory Managed Environments . . . .	1 - 4
Support for Non-Memory Managed Environments . . . .	1 - 5
Video Handling . . . . .	1 - 6
VIDPATCH.COM . . . . .	1 - 6
MOS Files . . . . .	1 - 8
Understanding Terms in MOS . . . . .	1 - 10
The System Administrator . . . . .	1 - 12
Command Line Editing . . . . .	1 - 13
Command Recall Buffer . . . . .	1 - 13
Command Line Control Keys . . . . .	1 - 14

# Overview

## Introduction

PC-MOS is an advanced operating system designed to provide Multi-Tasking and Multi-User capability to personal computers. MOS controls requests made on the computer's hardware resources, including the central processing unit (CPU), random access memory (RAM), disk drives, printers, serial ports, and other peripherals.

Application programs and utilities call upon MOS for communicating directly with the hardware to obtain the desired results. As a middle-manager, MOS must control keystrokes, screen and printer output, disk access, RAM allocations, time checks, priority changes and much more -- for multiple programs and users, all at the same time.

PC-MOS is offered in a single-user, multi-tasking version that provides the capability of performing multiple tasks on a single computer. MOS is also available in 5-user, 9-user, and 25-user versions. If you have a single-user version of MOS, you have the capability of upgrading to a 5-user, 9-user, or 25-user system. You may also upgrade from the 5-user to the 9- or 25-user version as your processing demands increase. Any files or programs you are currently running should not be affected as you upgrade.

This manual explains all the commands available in MOS, and how they may be used. If you have the single-user module, the multi-user information in this manual will not apply, although it will give you a preview of MOS's full potential. All other information in this manual applies to all versions of MOS.



## Special MOS Features

The multi-tasking capability of MOS lets you maximize your computer to its fullest processing capability. You may share your computer's resources among several tasks. MOS provides NETBIOS emulation as a means of intertask communication between applications.

The MOS kernel (or core) and the majority of the supporting device drivers and utilities were written in assembler for compactness and speed. MOS is modular in the fact that expansion is possible with device driver modules, and you may easily replace modules to expand the number of users, up to 25.

MOS provides support for monochrome, CGA and EGA video cards. EGA video cards are supported in the CGA mode for all tasks, while programs using the enhanced mode are supported at the main console only. Multi-user versions of MOS support monochrome and CGA video modes at appropriate CRT terminal workstations, or computer workstations running terminal emulation software.

You may optionally establish security with MOS at the user level. You may secure files or entire directories from unauthorized use, and assign your own passwords and user ID codes. MOS security lets you selectively determine levels of access for users to individual files, directories and multi-tasking partitions. An advanced security feature lets you further define your own encryption key for even greater security.

The MOS Print Spooler is available to manage your printing requirements. With multiple users, you may need to control the output to a printer so that only one file prints at a time. The Print Spooler may also be used to facilitate multiple printers.

An on-line help facility is built into MOS and is always available. You may use the .HELP command to display a menu of all the commands available in MOS, or display only the correct form and an explanation of a specific command. You may bypass the menu by entering .HELP and the name of the command to directly display the command form and an explanation.

# Overview

---

## Optimization of Memory Managed Environments

It is in a memory managed environment that the full power and capability of PC-MOS can be achieved.

When MOS is booting up, it must find places in memory to use for the MOS kernel, the SMP (System Memory Pool), the disk cache and the video save area.

Changes have been made in the way MOS relocates certain key components of the system into high memory to free more memory for individual tasks. The MOS kernel has been divided into two discrete segments that can be relocated independent of one another. Also, MOS's initialization procedure first relocates the two MOS kernel segments into FREEMEM, then the SMP, then the command processor, then the video save area, and finally the disk cache.

By relocating the largest items first, it is much more likely that items which won't fit up in FREEMEM will be the smaller items. These cause less encroachment into the memory available for tasks, resulting in larger task sizes.

You must tell MOS what type of memory management device you have in your computer for it to function properly. This is done with the MEMDEV configuration Command Statement. The MEMDEV statement must be part of the configuration set up in your CONFIG.SYS file. The VTYPE configuration Command Statement allows a user to control certain aspects of MOS's memory allocation, and thereby, task size limits. By telling MOS what type of video is being used, certain memory not in use for video may be assigned for use as task memory. The VTYPE statement is part of your configuration set up and is only possible when a memory management driver is being used.

When fine tuning MOS's memory allocation, the MOS INFO utility command is available to report what FREEMEM was found (if no FREEMEM statement was specified) and what portions of it are being used by each system component.



## Support for 8088, 8086 & 80286 (Non-Memory Managed) Environments

The memory allocation and task initialization procedures in MOS provide multi-user and multi-tasking capability even when no memory management driver is present. This means that PC-MOS is supported on an 8088 or 8086 type PC/XT or on an 80286 AT without an AT-GIZMO, ALL Computers CHARGE CARD, or similar memory management device. It could even be used on an 80386 system with no extended memory board installed.

Normally, in this situation, the concept of FREEMEM does not apply since there is no memory remapping capability to let you utilize the upper memory areas. As a result, low memory must be used for the foreground overhead items: the MOS kernel (\$\$MOS.SYS), the command processor (\$\$SHELL.SYS), the system memory pool (SMP), and the disk cache.

Also, when any tasks are added, they must use the same low memory area. Note that the memory for the new task will be taken from the top of the task from which the ADDTASK command is issued. On a 640K system, with a moderately sized SMP (SMPSIZE=40K), two tasks of approximately 240K should be possible.

In contrast to the memory use scheme when a memory manager is present, non memory managed multi-user/multi-tasking uses part of each task's memory space for the video save area. This means that in a 96K task on a CGA system, 16K of that 96K is used for the video save buffer. A monochrome task would only need 4K.

The system driver \$286N.SYS is available for use with 80286-based machines without a memory management board installed. When \$286N.SYS is used in this non-memory managed environment it provides better interrupt handling and allows for part of the MOS kernel to be relocated into extended memory, above 1 Megabyte. This helps to increase the memory available for tasks. Note that even though the \$286N.SYS driver is installed with the MEMDEV= statement, it does not provide memory management.

# Overview

## Video Handling

Some applications bypass the operating system and write output directly to the video screen in order to perform faster screen handling. These applications cause video problems in a multi-user environment. Whenever an application uses this direct video RAM addressing procedure, the screen output intended for the workstation will appear on the host computer's monitor. The background will remain active, but no screen will display.

PC-MOS will correct the video problems on 80386-based computers without the use of any special video handling programs.

When running PC-MOS on 80286, 8086, or 8088-based computers a memory management device, such as the AT-GIZMO or ALL Computers CHARGE CARD for an 80286, is required to provide video management for applications that write directly to the video memory. (See your dealer for the appropriate memory management device for your computer.)

## VIDPATCH.COM

Some applications also directly control the cursor and sound. In these cases (on 80286, 8086 and 8088 machines) the video will be correctly output to the background partition, but the foreground cursor will move and any beeps will be heard from the foreground.

To solve this, VIDPATCH is executed on the application that is directly controlling the cursor and sound, in the following format:

VIDPATCH [filename]

where "filename" is the program file which contains the undesirable cursor/sound handling logic.

VIDPATCH is run ONLY ONCE! Once a program has been fixed it will run in the foreground and background without any further changes, and need not be fixed again unless you reinstall the application.



BE SURE TO SAVE THE OLD VERSION OF ANY PROGRAM THAT YOU MODIFY WITH VIDPATCH... you will want it for back-up, and also to take full advantage of the application when run at the main console.

VIDPATCH supports only specific applications, primarily those in most popular use. There are other less popular applications with which VIDPATCH will not work.

AT GIZMO and CHARGECARD users: Note that VIDPATCH is frequently required for compatibility with applications running at terminals or in "background" tasks.

# Overview

## MOS Files

The files you receive with MOS and their purposes are:

- |               |  |
|---------------|--|
| \$\$MOS.SYS   | This file contains the MOS kernel, the core of the operating system, and several MOS device drivers.   |
| \$\$SHELL.SYS | The intrinsic commands provided with MOS are contained in this file. Each time you boot MOS, this file loads in memory and the commands are always available. Intrinsic commands are listed in the General Commands chapter of this manual.                        |
| COMMAND.COM   | This file allows \$\$SHELL.SYS to be activated for each partition.   |
| .COM and .EXE | Files provided with MOS that have a .COM or .EXE extension are the MOS extrinsic commands. These program files load from disk into memory each time they are invoked. A list of the MOS extrinsic commands appears in the General Commands chapter of this manual. |

There are several special device drivers provided with MOS. These device drivers are contained in files that are identified by a preceding dollar sign and an extension of .SYS. They are:

- |               |  |
|---------------|--|
| \$EMS.SYS     | This device driver will emulate the Lotus-Intel-Microsoft Expanded Memory Specification, and is used for bank switching "extended" memory as if it were "expanded" memory. |
| \$NETBIOS.SYS | This device driver supports applications written for the Network BIOS Emulation (NETBIOS) protocol by providing software emulation of network "adapter cards".             |
| \$PIPE.SYS    | Allows you to define character devices to buffer and "pipe" information between partitions.  |



**\$RAMDISK.SYS** This device driver lets you define a section of extended memory that MOS treats as a disk device. This is sometimes referred to as a virtual or RAM disk.

**\$SERIAL.SYS** This device driver sets up a buffered interface to standard serial ports. This driver is required for MOS to recognize more than the standard serial ports, COM1 and COM2, and must be present to support workstation terminals under MOS.

There are some files that are not provided with MOS, but MOS will recognize and use these files for special processing functions. These are files that you may create to define your own processing requirements and needs. The files you may create for special processing functions with MOS are:

**CONFIG.SYS** You may create this file to define your operating environment, such as the devices available to MOS, free memory available to MOS, etc. If you don't create your own CONFIG.SYS file, MOS uses built-in defaults to define your environment.

**AUTOEXEC.BAT** You may create this file to automate the execution of programs and utilities each time you boot MOS. You may include commands for application programs or MOS utilities in this file.

**\$\$USER.SYS** If you decide to use MOS security, you must create this file. It contains the user records that define a user's ID and password, the access level each user has to each class, and each user's default output class. Be sure you read and understand the Security chapter in this manual before creating this file.

**\$\$MASTER.SYS** This file is necessary only if you decide to use the master password level of security on your system. Be sure you read and understand the Security chapter in this manual before creating this file, or you may find your system unusable.

# Overview

## Understanding Terms in MOS

When you begin to install and set up MOS, there are a few terms you need to understand. The following is an explanation of these terms. If you find a term in this manual that you don't understand, look in the Glossary at the end of this manual.

class	A class in MOS may be a security class or a print class, and is usually identified as being one or the other.
command	A command is an instruction that causes MOS to perform a specific action, such as copy a file or display a list of files.
computer	The hardware pieces that comprise your system, including the central processing unit, disk drives, a video display screen and a keyboard.
disk	The term "disk" is used in this manual to represent any disk media, which may be a diskette, a hard disk, a removable cartridge, etc.
main console	The workstation consisting of the monitor and keyboard that are directly associated with the central processing unit is referred to as the main console.
multi-tasking	The ability to have several programs share the central processing unit in your computer at the same time is referred to as multi-tasking.
multi-user	The ability to have more than one user running the programs that share the central processing unit in your computer is referred to as multi-user.
operand	A syntax element that further defines how a command in MOS is to perform an action is referred to as an operand.



partition	A partition is a portion of your computer's memory where a program may be processing independently of any other program, i.e., multi-tasking. Each partition is identified by a unique task ID number.
single-user	An environment where only one user may run the programs that share the central processing unit.
workstation	A CRT terminal or reasonable substitute that is connected to your computer to share the central processing unit is referred to in MOS as a workstation.

# Overview

## The System Administrator

Because MOS provides many sophisticated capabilities, a technically knowledgeable person should be in charge of setting up and maintaining your computer. The term "system administrator" is used in this manual when referring to anyone that performs these functions.

The functions of the system administrator may be performed by one or more individuals. If you have a multi-user version of MOS, you may have one person, such as your dealer, install the computer hardware. There may be another person, perhaps someone from your dealership or someone in your company, who sets up your operating environment.

Some of the responsibilities of the MOS system administrator are:

- Installing MOS on your computer.
- Setting up the CONFIG.SYS file with the \$SERIAL.SYS or other device driver to define port addresses. Also defining, and other drivers necessary for your environment.
- Setting up the AUTOEXEC.BAT file to define your operating environment. This file should include the .ADDTASK commands necessary for partition support. The system administrator should also set up the startup batch files for each user's partition.
- The system administrator is responsible for installing workstations, and establishing communications between the computer and workstations.
- Establishing and maintaining the organization and structure of directories.
- Setting up the MOS Print Spooler (optional).
- Setting up and maintaining MOS Security in the \$\$USER.SYS file (optional).
- Creating any necessary batch files.
- Maintaining a schedule for backing up your files.



## Command Line Editing

When you boot MOS, the system prompt appears and MOS is ready to accept a command entry. The first time you boot MOS, the prompt appears on your screen similar to the following:

[A:\]\\_

The underscore following the prompt is the cursor, which is positioned at the beginning of the line where you will enter all commands. The cursor gives you an indication of your location on the screen.

When entering commands, MOS provides some useful editing control keys and two editing modes, insert and typeover. When you first boot MOS, it is in the typeover mode and the cursor appears as a blinking underscore. You may change to the insert mode by pressing the Ins key and the cursor appears as a block. You may place the .INSERT command in your AUTOEXEC.BAT file if you always want to be in the insert mode when you boot MOS.

## Command Recall Buffer

Each time you enter a command on the command line, it is saved in the MOS Command Recall Buffer. As you enter new commands, the oldest entries in the buffer are dropped and the newest are saved.

The number of command lines saved in the buffer will vary. If your command lines average about 10 characters, the buffer will hold an average of 45 to 50 command line entries. If your command lines average 20 to 30 characters, there will be fewer lines saved in the buffer. The more characters there are in the command line, the fewer the number of command lines that are saved.

You may retrieve the most recent line in the buffer by pressing the Up-Arrow key, or retrieve the oldest line in the buffer by pressing the Down-Arrow key. You may clear the buffer by pressing the CTRL PgDn keys.

# Overview

## Command Line Control Keys

The following are the command line editing keys in MOS:

<b>INS</b>	Toggles between the insert and typeover mode. An underlined cursor identifies the typeover mode and a standard block cursor means insert mode.
<b>DEL</b>	Deletes the character at the cursor and shifts remaining characters one position to the left.
<b>BkSp</b>	Deletes the character to the left of the cursor and shifts remaining characters one position to the left. This key may appear as a left arrow on some keyboards, but do not confuse it with the left arrow located on the numeric key pad.
←	Moves the cursor to the left, but does not overwrite an existing entry.
→	Moves the cursor to the right, but does not overwrite an existing entry.
<b>CTRL</b> ←	Moves the cursor to the first position of a word to the left of the cursor's current position.
<b>CTRL</b> →	Moves the cursor to the first position of a word to the right of the cursor's current position.
↑	Beginning with the most recent entry in the command recall buffer, recalls one line at a time. Each additional time you press this key, it recalls the preceding entry. After the last entry has been recalled, the most recent entry appears again.



↓

Beginning with the oldest entry in the command recall buffer, recalls one line at a time. Each additional time you press this key, it recalls the next succeeding entry. After the most recent entry has been recalled, the oldest entry appears again.

**CTRL PgUp** Copies the current line to the command recall buffer without having to enter the line as a command.

**CTRL PgDn** Deletes the contents of the command recall buffer.

**ENTER** Executes the current line and adds it to the command recall buffer.

**CTRL C** Aborts execution of the current command, and  
or  
returns control to the MOS system prompt.

**CTRL BRK**

**CTRL S** Causes an output display to halt until any other key is pressed.

**CTRL P** Causes data echoed to the video screen to also echo to the printer.

**HOME** Moves the cursor to the first position of the line.

**END** Moves the cursor to the last position of the line.

**CTRL BkSp** Delete the current word at the cursor position.

**CTRL END** Deletes all data from the cursor position to the end of the line.

**ESC** Ignores the current line and begins a new one.

**CTRL PrtSc** Causes data echoed to the video screen to also echo to the printer.

This page intentionally left blank.

---

# CHAPTER 2: CONFIGURATION

---

Introduction . . . . .	2-2
Creating a CONFIG.SYS File . . . . .	2-2
Configuration Commands Statements . . . . .	2-3
8087 . . . . .	2-4
CACHE . . . . .	2-5
COUNTRY . . . . .	2-15
DESNOW . . . . .	2-16
DEVICE & LDEVICE . . . . .	2-17
FREEMEM . . . . .	2-19
MEMDEV . . . . .	2-22
SHELL . . . . .	2-26
SLICE . . . . .	2-27
SMPSIZE . . . . .	2-28
USERFILE . . . . .	2-30
VTYPE . . . . .	2-31
Device Names . . . . .	2-37
Special MOS Device Drivers . . . . .	2-38
\$EMS.SYS . . . . .	2-38
\$MOUSE.SYS . . . . .	2-40
\$PIPE.SYS . . . . .	2-42
\$RAMDISK.SYS . . . . .	2-44
\$SERIAL.SYS . . . . .	2-46

# Configuration

## Introduction

After installing MOS on your computer, you will need to create a CONFIG.SYS file to define your operating environment. The information necessary to create the configuration file is explained in this chapter. Following the configuration information is an explanation of the standard device drivers supplied with MOS, and the special device drivers that you may select to use.

## Creating a CONFIG.SYS File

The CONFIG.SYS file contains the information MOS needs to know about your computer's configuration. For example, what hardware devices are available, how many cache buffers to allocate, what memory management driver to use, what areas in high memory (above 640K) are available for MOS to use, etc. The CONFIG.SYS file may also contain information necessary for an application program that runs under MOS.

The CONFIG.SYS file should always reside at the root directory on your MOS boot disk. If MOS does not find a CONFIG.SYS file when you boot your computer, MOS will use the default values for the necessary command statements and boot up in the Real mode. In this mode, no matter how much extended memory your system has, only the first 640K of RAM is accessible for both MOS and your applications to use.

It is in a memory managed environment that the full power of PC-MOS can be achieved. On a computer with memory management capabilities (an 80486, 80386, or an 80286 with an AT-Gizmo or CHARGECARD installed) the MEMDEV and FREEMEM statements can be specified in the CONFIG.SYS file. Once this is done, MOS will boot up in Protected mode with memory management active and try to relocate as much of itself as possible into high memory between 640K and 1MB. This greatly increases the memory available for your applications provides the environment necessary for MOS's multitasking and multiuser capability.



The command statements and default values that are necessary to MOS are explained in this chapter. MOS also contains built-in device drivers that you do not have to define in the CONFIG.SYS file, and some optional device drivers that you must define if you want to use them. The built-in and optional device drivers are explained following the CONFIG.SYS file command statements.

You can create a CONFIG.SYS file with the MOS system editor, .ED. The MOS editor lets you create a file and add, delete, or insert lines in the file. The MOS editor is explained in detail in a separate chapter of this manual.

## Configuration Command Statements

There are specific command statements in PC-MOS that you may want to set up in the CONFIG.SYS file. Some statements are optional and will depend on the type of computer you have.

The following pages explain the command statements you may want to set up, along with any default values used when a CONFIG.SYS is not present, or when a specific command statement is not present. The command statements are explained in alphabetical order, but may be entered in the file in any order.

# Configuration

## 8087

This command statement is only necessary if multiple applications will be using a math co-processor (8087, 80287, or 80387) simultaneously.

**Type:** Intrinsic.

**Form:** 8087=yes

### **Operands:**

**yes** enter yes to configure your system for simultaneous use of a math co-processor by multiple applications.

### **Explanation:**

In order for more than one application to use the math coprocessor simultaneously in a multi-tasking environment the 8087=yes command statement must be entered in your CONFIG.SYS file.

The statement form is always "8087=yes" regardless of whether you are using an 8087, 80287, or 80387 math co-processor.



## CACHE

CACHE lets you set up disk caching to speed up file processing on your computer. It gives you the ability to fine tune your system by establishing the size of the cache, and how often read requests are written to the disk. Disk caching is noticeably faster than accessing data directly from disk since it uses the computer's memory for storing and retrieving data. Also, in a memory managed environment, CACHE uses extended memory and does not take away from available base memory like a BUFFERS statement. (The BUFFERS statement has been replaced by the CACHE statement.)

PC-MOS supports hard disk partitions of greater than 32MB that are set up using Disk Manager® v4.02 and above by OnTrack Computer Systems, Inc. Since the Disk Manager software changes the standard 512 byte/sector setting for large hard disk volumes, you must include the /BPS (bytes per sector) operand with the CACHE= statement to inform MOS of the new Bytes/Sector value.

**Type:** Intrinsic.

**Form:** CACHE=*nnnn*,*{unit}*,*{firstw}*,*{lastw}*,*{drives}* /BPS=*nnnnn*

### Operands:

*nnnn* enter the total size of the cache in kilobytes. This is the total amount of memory reserved for caching read and write requests. Defaults to 16K if not specified.

*unit* enter the size in kilobytes of one unit of data that may be read at one time. This is the size of each individual buffer in the cache. Defaults to 2K if not specified.

*firstw* enter the maximum number of seconds that are to elapse before writing data from the cache to disk, i.e. how long any data may be held in memory. Enter a 0 (the default value) if you do not want to use this operand.

# Configuration

---

*/lastw* enter the maximum number of seconds to elapse after the last write request before all data is physically written to the disk from the cache. After the specified number of seconds without any write requests, all data in the cache will be physically written to disk. However, if another write request occurs before time expires, *lastw* will reset and start timing over again before writing any data to disk. Enter a 0 (the default value) if you do not want to use this operand.

*drives* enter the letter of each drive to make use of the cache. Use a comma as a delimiter between each drive letter, for example: C,D,E. Leaving this operand blank causes MOS to use caching on all available drives. (Do NOT leave this operand blank if network drives other than LANLink drives are active.)

*/BPS=nnnnn* This option allows you to specify the Bytes Per Sector value for your hard drive when OnTrack's Disk Manager was used to partition the drive. The nnnn values are as follows:

<u>Hard Disk Volume</u>	<u>Bytes/Sector</u>
over 32MB to 64MB	1024
over 64MB to 128MB	2048
over 128MB to 256MB	4096
over 256MB to 512MB	8192
over 512MB to 632MB	16384

## Explanation:

A CACHE statement must be in your CONFIG.SYS file in order for disk caching to be active on the system. If MOS does not find the CACHE= command statement (or the DEVICE= \$CACHE.SYS statement used in earlier releases of MOS) in the CONFIG.SYS file, MOS will default to using a cache size of 16K, a unit size of 2K, and both the firstw and lastw operands will default to 0. The cache area will be available to all drives, but it will only be used by MOS for buffers for read and write requests without the benefits of disk caching.



If MOS does not find the CACHE= command statement but there is an old \$CACHE.SYS driver installed, caching will still be active on the system since the DEVICE=\$CACHE.SYS statement is supported as an internal command. The size operand from the \$CACHE.SYS statement will set the cache size. The old \$CACHE.SYS buffer address operand is ignored. The remainder of the operands that would have been specified by the CACHE= statement will assume their default values.

If both the CACHE= and DEVICE=\$CACHE.SYS statements appear in the CONFIG.SYS file, MOS will issue an error message saying that it is using the CACHE= statement values and ignoring the old \$CACHE.SYS statement.

The "nnnn" operand of the CACHE= statement specifies the size in kilobytes of the cache to be used. The total memory used by caching includes the cache plus a cache descriptor list. There is one cache descriptor for each cache buffer. The number of cache buffers is approximately "nnnn" divided by "unit". Since each cache descriptor is 16 bytes, the memory used by the descriptor list is 16 bytes times the number of cache buffers. The cache descriptor list is always located in the lower megabyte of memory. The cache itself is located in extended memory when memory management is available. In general, the larger the cache, the faster the cache. Therefore, set aside as much memory as you can afford for the cache.

The size of each individual buffer in the cache is specified by the "unit" operand of the CACHE= statement. This defines the largest unit of data that may be read at one time. For example, with a cache size of 256K, an assigned unit size of 4K results in 64 buffers ( $256/4=64$ ). The optimum unit size, and number of cache buffers, is best determined by trying different values for your particular applications.

Applications that perform random reads and writes, like data bases, generally perform better with a smaller unit size (more buffers). Applications that perform sequential reads and writes generally perform better with a larger unit size (fewer buffers). However, too small a unit size (very large number of buffers) may slow the system down, since it may take longer to search all the buffers for the record than to read it directly from disk.

## Configuration

For hard disks we suggest you start with 4K, and adjust this number down if your processing speed does not seem fast enough. For floppy disks we recommend you use a unit size of at least one to two tracks of your particular disk, as follows:

1.2M, 5.25":	track size = 15K
360K, 5.25":	track size = 9K
1.44M, 3.5":	track size = 18K
720K, 3.5":	track size = 9K

Firstw and lastw are used to specify that everything to be written to the disk should first be written to the cache (held in memory for a specified time), and only later physically written to the disk. Any write to disk will clear both first and lastw, if set. Firstw and lastw both default to 0, in which case everything is written immediately to disk, as well as being written to the cache. The higher the values, the faster the cache, but the more likely that there will be data lost in memory in the event of a power failure.

If either firstw or lastw is used and the cached drive is a floppy disk drive, MOS will cause the drive light to stay on from the time an application wants to write something until everything has been physically written to disk. This is to remind you not to remove the floppy disk from the drive until the drive light goes out, since all your data may not have been written from the cache to the floppy disk.

Firstw specifies the maximum number of seconds during which something might be held in the cache without being physically written to disk. This operand acts as a "security blanket" by insuring that at some specified interval data will be written from the cache to the disk regardless of how often write requests occur or how often lastw postpones the writes to disk. The largest number you may enter is 1800 seconds (30 minutes). Data may be held for less time than specified if the cache becomes full, in which case MOS would write the data to disk sooner to free cache space.



The firstw "clock" starts timing the first time an application wants to write something to disk. If and when the specified time expires, MOS looks through the cache and physically writes everything to disk, clearing firstw (and lastw, if set). Firstw will also be cleared if the cache writes to disk because the cache is full or because the lastw timer expires. "Cleared" means that the clock stops timing and any remaining time on the clock is erased. Once cleared, the firstw "clock" will not reset until the next time an application makes a write request. "Reset" means to start timing over again from the original setting.

Lastw specifies the maximum number of seconds to elapse after the last write request occurred before it will cause all data to be physically written to the disk from the cache. This operand improves the performance of your computer by letting you postpone disk writes to reduce the amount of head movement and total number of writes.

The lastw "clock" starts timing EACH TIME an application wants to write something to disk. If lastw expires, data is written to disk and both lastw and firstw are cleared. Upon the next write request both will reset and start timing again. If another write request occurs before lastw expires, lastw resets and the lastw "clock" starts timing over again (while firstw keeps on timing.) Therefore, if write requests continue to occur more frequently than the lastw specified time, lastw will keep resetting with each write request and never expire. In this case, firstw will eventually expire and cause all data to be written to disk.

Though the largest number you may specify for lastw is 1800, you should always enter a number that is less than that of the firstw operand, or lastw will be rendered inoperative. If the cache becomes full, MOS will override this operand and write to disk to free cache space for more data.

The "drive" operand is used to specify which drives to cache. If no drives are specified, all drives are cached. If \$RAMDISK.SYS is used, MOS will recognize and not cache it. If any other RAMDISK drive is used, all drives except the RAMDISK drive should be specified by the "drive" operand to ensure that the RAMDISK drive is not cached.

## Configuration

---

Network drives should NOT be cached or data corruption will result. If LANLink drives are active, MOS will recognize them and they will not be cached. If network drives other than LANLink drives are active, all drives except the network drives should be specified by the "drive" operand to ensure that the network drives are not cached.

In the following example firstw is set to 600 (10 minutes) and lastw is set to 120 (2 minutes):

```
CACHE=1024,4,600,120,C
```

Upon the first write request, both the firstw and lastw "clocks" start timing. If no other write requests are made within two minutes, lastw will expire and all data in the cache will be written to disk, clearing both lastw and firstw. Both will reset and start timing again upon the next write request.

If another write request did occur before the two minute lastw "clock" expired, the lastw "clock" would reset to two minutes and start timing over again (while the firstw "clock" would keep on counting from its original ten minute setting). If all time intervals between write requests are less than two minutes, lastw would keep resetting to two minutes with each request and never expire. If this continues, the ten minute firstw "clock" will ultimately expire and cause all data in the cache to be written to disk, clearing both "clocks". Upon the next write request, both "clocks" will reset and start timing again.



## Large Volume Support with OnTrack Disk Manager and /BPS

The PC-MOS HDSETUP utility can partition hard disks with 16 or less heads and 1024 or less cylinders. Very large hard disks may have more heads and/or cylinders.

Disk Manager by OnTrack Computer Systems, Inc. can partition hard disks that have greater than 16 heads and/or greater than 1024 cylinders. PC-MOS supports hard disks of greater than 32MB that are set up using Disk Manager v4.02 and above. You can not run the PC-MOS HDSETUP utility on drives partitioned with Disk Manager.

The Disk Manager software changes the standard 512 byte/sector setting for large hard disk volumes. Therefore, you must include the /BPS (bytes per sector) operand with the CACHE= statement. Also, the "unit" operand of the CACHE= statement MUST be set to at least the same number of kilobytes as the BPS value for the setup to work correctly.

The following are the BPS values that Disk Manager uses for various hard disk volumes:

Hard Disk Volume	Bytes/Sector	minimum cache "unit" size
over 32MB to 64MB	1024	1K
over 64MB to 128MB	2048	2K
over 128MB to 256MB	4096	4K
over 256MB to 512MB	8192	8K
over 512MB to 632MB	16384	16K

For example, the CACHE statement to include in your CONFIG.SYS file for a 200MB hard disk with a 2MB (2048K) cache size would be:

CACHE=2048,4,0,0,C,D /BPS=4096

## Configuration

Note that 2048 is the cache size, 4 is the unit size (in kilobytes) and the /BPS=4096 sets the bytes/sector value. Note also that the two zeros are the settings for the firstw and lastw timers, which effectively disables write caching. C and D are the drives to be cached.

If you have more than one logical disk, say a 32MB, and 168MB on the same physical hard disk you must use the Bytes/Sector value for the largest one, in this case /BPS=4096.

Note that Disk Manager only allows one bootable partition when you have multiple logical disks (partitions). This means that you can not set up a dual bootable DOS and MOS system on one hard disk if it was partitioned with Disk Manager. Another Disk Manager limitation is that the first logical partition can not exceed 32MB. If you require multiple bootable partitions, you must use HDSETUP to partition the hard disk.

While HDSETUP only supports up to four logical disk partitions, Disk Manager supports up to sixteen. Dividing a hard disk up into several smaller partitions will save disk space as compared with having just a few large logical drives since the cluster size is smaller on smaller logical drives. Practically, this means that small files will take up less space on a smaller logical drive.

For installation instructions follow the OnTrack documentation. The installation MUST be performed when running under a PC-DOS or MS-DOS version 3.x since Disk Manager installs PC/MS-DOS on the drive when it prepares the drive. Once the drive is totally functional as a normal PC/MS-DOS system, install PC-MOS by following the appropriate "Installation" section instructions.

**NOTE:** If you use Disk Manager (or HDSETUP) to set up your hard disk and you want both PC/MS-DOS and PC-MOS to be able to read all disk partitions, the C drive must be less than or equal to 32MB and you must NOT use a DOS version greater than 3.x. (DOS 4.x versions no longer support the same disk partitioning structures as previous versions of DOS.)



## IMPORTANT CACHING RULES:

1. NEVER open a floppy disk drive door with the drive light on.
2. NEVER reboot or turn off your computer without first running MOSADM CACHE OFF **or** being POSITIVE that you've waited the lesser of firstw or lastw seconds following the last time anything was done on the system.
3. If you don't understand items 1 and 2, or anything else about firstw and lastw, always set them both to 0.
4. NEVER cache a network drive. Only cache drives that are local to each machine.

If there are any errors in the CACHE= statement the following message will appear:

Invalid CACHE parameter, using default.

The BUFFERS command statement is no longer used and has been replaced with the CACHE= command statement. If you enter a BUFFERS statement in your CONFIG.SYS file, the following error message will appear each time you boot your computer:

BUFFERS= command ignored, use CACHE= command instead.

When writes are postponed, critical error messages may occur when information is physically written to disk rather than immediately when the action is requested.

## Configuration

If this happens, the critical error message will appear on the screen of the partition which last updated the data that could not be written. This will be described as a flushing error, as follows:

**CRITICAL ERROR DETECTED**

Error: Write protect while flushing A:  
Enter A to abort or R to retry.

Choosing retry will try to flush the data to disk again. Choosing abort will clear the error without flushing all of your changed data to that disk. (Abort should only be used as a last resort since all of your changes for that disk will be lost.)

If too many tasks have pending flushing errors, the system may become tied up to the point that it can't get any free cache buffers. If that happens, the following message may appear:

**CRITICAL ERROR DETECTED**

Error: Critical error block while writing/reading A:  
Enter A to abort or R to retry.

This message may also occur when an operation is trying to clear out all the cache buffers for a drive, and there is a cache buffer for that drive waiting for a response to a flushing error.

If this error message appears, first clear as many tasks as possible that have a pending flushing error. Then come back and choose retry to try to clear this error message.



## COUNTRY

The COUNTRY command statement establishes the format MOS uses for the date, time, currency symbol, and number format for currency decimal positions. MOS defaults to using 001, the telephone system code for the USA, if the COUNTRY command statement is not found in the CONFIG.SYS file.

**Type:** Intrinsic.

**Form:** COUNTRY=*nnn*

**Operands:**

*nnn* enter the three digit telephone system code of the country to define.

**Explanation:**

The value of *nnn* is a three digit country code that corresponds to the three digit telephone system code for a country. If you do not set up a country code, MOS defaults to the USA telephone system code of 001. The other codes supported by MOS are:

Australia	061	Middle East	785
Belgium	032	Netherlands	031
Canada	002	Norway	047
(French)			
Denmark	045	Portugal	351
Finland	358	Spain	034
France	033	Sweden	046
Germany	049	Switzerland	041
Italy	039	United Kingdom	044
Israel	972	United States	001

# Configuration

## DESNOW

DESNOW corrects a snow-like effect that appears on color video screens with some color adapter cards. If MOS does not find the DESNOW command statement in the CONFIG.SYS file, the special logic for rewriting to the video screen is not used.

**Type:** Intrinsic.

**Form:** DESNOW=yes/no

### **Operands:**

- |     |  |
|-----|--|
| yes | enter yes to use the special logic that avoids snow when rewriting to the video screen.  |
| no  | enter no if you do not need to use the special logic when rewriting to the video screen. |

### **Explanation:**

Some color adapter boards cause a snow-like display when writing to the video screen. Using the DESNOW command statement will correct this condition, but video output will be slightly slower.



## DEVICE & LDEVICE

DEVICE gives you a means to load device drivers to install support for non-standard devices. You can have any number of device command statements set up in the CONFIG.SYS file. Normally, the device drivers are relocated up into high memory as part of the SMP to save memory space for applications.

However, if for some reason a device driver is required to be run in low memory the LDEVICE statement can be used to keep it from relocating into high memory.

You can also use the .ADDDEV (add device) and .REMDEV (remove device) commands at the MOS system prompt to add or remove certain drivers without having to reboot your computer.

**Type:** Intrinsic.

**Form:** DEVICE={*d:*}|{*path*}|*filename* {*operands*}

LDEVICE={*d:*}|{*path*}|*filename* {*operands*}

### Operands:

*d:* enter the letter of the drive from which you want to load the device driver. MOS defaults to the current drive if a drive is not entered.

|*path*| enter the path of the directory where the device is located. MOS defaults to the current directory if a directory is not entered.

*filename* enter the file name of the device driver.

*operands* enter any operands that are necessary for the driver.

### Explanation:

The standard MOS device drivers are automatically loaded and do not have to be entered in the CONFIG.SYS file.

## Configuration

There are optional device drivers supplied with MOS that you must define with a DEVICE command statement in the CONFIG.SYS file if you want to use them. The standard and optional devices are explained later in this chapter.

There are also other non-industry standard device drivers that you may need to enter in the CONFIG.SYS file. For example, if your computer has a hard disk that is not bootable with MOS, you probably need a device driver supplied by the manufacturer of the hard disk.

If for some reason a device driver is required to be run in low memory the letter L can be added to the beginning of the DEVICE= statement in your CONFIG.SYS file to keep it from relocating into high memory. Enter the DEVICE= statement exactly the same as you would normally with the exception that the letter L is added to the beginning of the word DEVICE as shown above.



## FREEMEM

The FREEMEM statement allows the system administrator to specify what memory address ranges between 640K and 1MB (A0000 and 100000 hex) are not used by hardware, and are free for use by MOS. This allows MOS to remap extended memory into these address spaces for use a task memory.

**Type:** Intrinsic

**Form:** FREEMEM=*m,n|N*

### Operands:

- m* enter the beginning address of the free memory range. This must be a byte address, expressed in hexadecimal.
- n* enter the ending address of the free memory range. This must be a byte address, expressed in hexadecimal.
- N* tells MOS that no free memory space is available.

### Explanation:

The memory addresses between 640K (A0000 hex) and 1MB (100000 hex) of your computers RAM is typically reserved for use by certain hardware devices installed in your computer. Some of these are: video adapter boards, network interface cards, and the system BIOS. However, many systems have unused blocks of memory addresses in this space that may be used by MOS.

MOS uses these memory ranges to move or relocate itself out of the primary 0 to 640K address range. This relocation leaves more memory available for application program use. For example, if memory addresses from C8000 to F0000 in your system was unused, the following command statement in your CONFIG.SYS file would tell MOS that this memory space is free for its use.

## Configuration

If the free memory is fragmented, then up to five separate FREEMEM command statements can be defined in the CONFIG.SYS file. If a FREEMEM command statement is not found, MOS looks at memory between C0000 and F0000, tries to guess which parts of it are available, and relocates itself accordingly.

Entering a FREEMEM=N statement is a good way to determine if you are having any sort of address conflict with other devices in your system. If your system doesn't boot correctly or locks up trying to load MOS, try entering a FREEMEM=N statement in your CONFIG.SYS file and rebooting. If the system boots now, you probably have two components in your system trying to use the same address space.

Video adapters typically use the area between A0000 and C8000. VGA boards normally use all of that area, while EGA boards normally only use A0000 to C4000. Hercules mono-graphics adapters usually use from B0000 to C0000. CGA normally uses from B8000 to C0000. MONO non-graphics adapters usually use from B0000 to B4000. These are not always the case. Check with your board manufacturer to determine the exact addresses used by your video adapter.

Since many systems load some of their system BIOS routines from F0000 to 100000, it is not normally a good idea to use that area as FREEMEM since conflicts may result.

**NOTE:** It is your responsibility to check with your add-in board manufacturers to determine what addresses locations above 640K are being used by those specific products. There is no way for us to tell exactly what areas of high memory are already in use in your specific computer system. If you have an EGA or VGA device driver, an EMS device driver, or a RAM disk driver in your CONFIG.SYS file, or hardware add-ons you will need to set your FREEMEM statement(s) to avoid a conflict of address assignments.

For example, the \$EMS.SYS device driver supplied with MOS defaults to the address E0000 to F0000 for memory paging. If you do not define a FREEMEM command statement to restrict MOS from also using this memory address, your computer will not function properly.



To be sure MOS does not use this address, the FREEMEM command statement you should enter in the CONFIG.SYS file is:

```
FREEMEM=C0000,E0000
```

This tells MOS that the memory segments from C0000 to E0000 are available, and MOS will not attempt to use any memory before or after this address range. The memory segments from E0000 to F0000 are then available for use with the \$EMS.SYS device driver.

# Configuration

---

## MEMDEV

MEMDEV specifies a System Driver, which tells MOS what type of Memory Management hardware is in your computer system. The MEMDEV driver is also used to tell PC-MOS if your hard disk requires a DMA (Direct Memory Access) buffer. The MEMDEV statement must be part of the configuration set up in your CONFIG.SYS file in order to access extended memory.

**Type:** Intrinsic

**Form:** `MEMDEV=[d:]{[path]}[filename [/f][/c][/e][/m]{/p}{/x}{/d=nn}`

### Operands:

*d*: enter the letter of the drive from which the system driver is to be loaded. If a drive is not specified, MOS defaults to the current drive.

*path* enter the path of the directory from which the system driver is to be loaded. If a path is not specified, MOS defaults to the root directory.

*filename* enter the complete file name of the system driver used for your computer system. There is no default, you must specify the correct system driver file name. The system drivers supplied with PC-MOS are:

\$386.SYS -	for 80386 and 80486 machines
\$GIZMO.SYS -	for IBM PC/AT or compatible with an AT-GIZMO
\$CHARGE.SYS -	for 80286 machines with a CHARGECARD
\$ALL.SYS -	for 8088 machines with an ALL CARD
\$286N.SYS -	for 80286 machines WITHOUT a memory management device installed

These drivers are explained on following pages. Other system drivers may also be used.



/f enter the /f operand if you want to force the memory check portion of the power-on self-test (POST) routine to be done during a warm reboot (CTRL-ALT-DEL). Normally the memory check is not done during a warm reboot so the reboot process take less time.

/c When running PC-MOS on the COMPAQ 386/20e, you must use the /c operand with the \$386.SYS system driver. (This prevents the MOS Kernel segment #2 from loading into extended memory.) For example:

```
MEMDEV=$386.SYS /c
```

One of the next operands is necessary only if you have a hard disk that uses the DMA (Direct Memory Access) channel:

/e enter the /e operand if your computer system's hard disk requires 64K of DMA buffering, such as an Emulex hard disk.

/m enter the /m operand if your computer system's hard disk requires 32K of DMA buffering.

/p enter the /p operand if your computer system's hard disk requires 16K of DMA buffering, such as IBM PS/2 Models 50, 60, 70 or 80.

/x enter the /x operand if your computer system's hard disk requires 9K of DMA buffering, such as an IBM PC-XT.

/d=nn enter the /d= operand followed by up to two digits indicating the desired DMA buffer size in kilobytes. This allows you to select a specific size buffer to suit your hard disk's needs. The minimum setting allowed is 4 and the maximum is 64.

NOTE: If these DMA operands are needed and not included the system may stop booting and lock up when it reaches the MEMDEV statement. You might also receive a "cannot open \$\$MOS.SYS file" message during boot-up if your hard disk needs a DMA buffer and one was not specified.

# Configuration

## **Explanation:**

You must tell MOS which type of memory management you have on your system for it to function properly. (There may be other system drivers besides those provided with PC-MOS.) The drivers supplied with MOS are as follows:

### **\$386.SYS**

The \$386.SYS system driver is necessary if your computer uses an 80386 or 80486 processor. An example of the correct command statement for an IBM PS/2 Model 80 would be:

```
MEMDEV=$386.SYS /p
```

On an 80386-based machine with the \$386.SYS system driver installed, part of the MOS kernel can be relocated into high memory (above the 640K boundary) and the other part can be relocated into extended memory, above 1 Megabyte. This allows users to have larger task sizes and larger SMPs.

### **\$GIZMO.SYS**

The \$GIZMO.SYS system driver is necessary if your computer system has an AT-GIZMO memory expansion board. An example of the correct command statement for a computer with an AT-GIZMO is:

```
MEMDEV=$GIZMO.SYS
```

If you have an IBM PC-AT with an AT-GIZMO and an Emulex hard disk, you would need to include the /e operand as follows:

```
MEMDEV=$GIZMO.SYS /e
```

### **\$CHARGE.SYS**

For users of the ALL Computers, Inc. CHARGECARD, the following is the correct MEMDEV syntax:

```
MEMDEV=$CHARGE.SYS
```



## \$ALL.SYS

Users of the ALL Computers, Inc. ALL CARD should use the \$ALL.SYS system driver, as follows:

```
MEMDEV=$ALL.SYS
```

## \$286N.SYS

The \$286N.SYS system driver is provided for use with 80286-based machines WITHOUT a memory management board installed. An example of the correct command statement for an IBM PC-AT with no memory management board installed is:

```
MEMDEV=$286N.SYS
```

When the \$286N.SYS driver is used in this non-memory managed environment it provides for better interrupt handling and allows part of the MOS kernel to be relocated into extended memory, above 1 Megabyte. This increases the memory available for tasks.

## Hard Disk DMA Buffer Size

Your hard disk manufacturer or documentation should tell you if your hard disk uses the DMA channel and how large a DMA buffer is required. The DMA buffer has to be large enough to handle the largest possible data transfer operation. Selecting a buffer size smaller than the required value reduces disk performance yet using too large a buffer reduces your available task size.

If you use a DMA buffer that is too small the disk read or write is broken up into as many pieces as required. For example, if a hard disk requires a 64K DMA buffer (the /e operand) but you set a 16K buffer with the /p operand, a 52K read will be broken up into three reads of 16K and one of 4K. Disk operations which are broken up to work with a smaller buffer will, of course, take more time. However, you get more memory space for tasks.

Therefore, you must choose your DMA buffer size with these trade offs in mind. Too large a buffer = reduced task size. Too small a buffer = reduced disk performance.

# Configuration

## SHELL

SHELL is a user interface that specifies the location and name of a user defined command processor that loads in place of MOS's command processor. MOS defaults to using COMMAND.COM if the SHELL command statement is not found in the CONFIG.SYS file.

**Type:** Intrinsic.

**Form:** SHELL = {*d:*}|{*path*}|*filename*

### **Operands:**

*d:* enter the letter of the drive from which to load the user defined command processor. MOS defaults to the current drive if a drive is not entered.

{*path*} enter the path of the directory where the user defined command processor is located. MOS defaults to the current directory if a directory is not entered.

*filename* enter the filename of the user defined command processor.

### **Explanation:**

The SHELL command statement is necessary only when a user defined command processor is used in place of MOS's command processor. It is important that the user defined command processor contains the same interrupt levels, intrinsic commands and batch processor as the MOS command processor.



## SLICE

SLICE lets you define the amount of processing time a partition is given at the time it is added. This is a default for time slicing that may be otherwise adjusted for each partition. If MOS does not find the SLICE command statement in the CONFIG.SYS file, the number of time ticks for each partition is set to 1.

**Type:** Intrinsic.

**Form:** SLICE=nnn

**Operands:**

*nnn* enter a 0 if time slicing is not to be used, or enter a number between 1 and 255 for the number of time ticks each task is to receive.

### Explanation:

The multi-tasking capability of MOS is based on time-sharing, where each task (partition) at a given priority level receives an equal share of the processor's time. The SLICE command statement instructs MOS to either not use time slicing, or to force the sharing of the processor. If time slicing is not used, a task will monopolize the processor until it makes a system call. If a program running in a partition requires constant control of the processor, other partitions may temporarily be inactive until the processor is freed.

The default for SLICE is 1. This instructs MOS to give each partition one time tick (approximately 1/18 of a second) of processing time before MOS moves to the next partition. Processing time is continually shared among the partitions in turn. For example, if there are three partitions, 1, 2 and 3, of equal priority, MOS shares processing time among all three. First 1 receives one tick of processing time, then 2 receives one tick, then 3 receives one tick, then 1 again receive one tick, then 2, etc.

From within a partition, the .MOSADM command lets you assign the number of time ticks that partition will receive during its turn, up to 255. However, 4 time ticks (approximately 4/18 of a second) is a practical limit.

# Configuration

---

## SMPSIZE

MOS uses a system memory pool (SMP) that is a fixed amount of memory to dynamically supervise all tasks and track all activity on your computer. MOS defaults to using 64K for the SMP if the SMPSIZE command statement is not found in the CONFIG.SYS file.

**Type:** Intrinsic.

**Form:** `SMPSIZE=nnnK`

`SMPSIZE=nnn,nnnK`

### **Operands:**

*nnn* enter the number of kilobytes (1024 bytes) to allocate to the SMP.

*K* you must enter a K to document that the SMPSIZE size is in kilobytes.

### **Explanation:**

Each time a file is opened or a new task is started, MOS uses memory from the system memory pool to track the activity. When the file is closed or the task is completed, the memory used to track the activity is returned to the system memory pool and is available for other tasks.

The value of *nnn* is the number of kilobytes to be allocated to the system memory pool. For example:

`SMPSIZE=128K`

This command statement allocates 128K bytes of memory for use by the system memory pool.

A split SMP statement is available to optimize the memory allocation process. For example, if a total of 100K of SMP is required but the largest contiguous block of available FREEMEM is 80K, the following statement could be used in your CONFIG.SYS file:



SMPSIZE=80,20K

This will result in two separate SMP blocks, an 80K block and a 20K block. This split SMP statement will now allow the SMP to be relocated up in high memory, increasing available task size. (This example presumes, of course, that no single device driver requires more than 80K of memory.)

If you do not specify the size of the system memory pool, MOS defaults to 64K of memory. This is a reasonable value for many systems.

A substantially larger SMPSIZE will be required if you are:

1. Running a large number of background tasks.
2. Loading large (or many) device drivers into the system.
3. Opening a large number of files in the system.

To estimate the size of the SMP required for your installation total the following items:

1. The sum of the size of each device driver loaded with the DEVICE= statement in your CONFIG.SYS file.
2. The maximum number of tasks on the system times 4K.
3. The total of all buffer space reserved by \$SERIAL.SYS and any other serial drivers such as VNA.SYS (if using IONA) or VGNA.SYS.
4. Total the first three items and add an additional 30% for file handles.

Don't make your SMP larger than necessary since it may take away from task size if it can not totally relocate itself into high memory. The maximum allowable value for SMPSIZE varies with system configuration. In a system configured to allow MOS to relocate (see FREEMEM), the maximum SMPSIZE is 440K bytes. In systems that do not allow MOS relocation, the maximum is 360K bytes.

# Configuration

## USERFILE

USERFILE lets you define the directory where the \$\$USER.SYS file is located. This command statement is necessary only when user security is invoked for your computer, and the \$\$USER.SYS file is not located at the root directory.

**Type:** Intrinsic.

**Form:** USERFILE=*d:\path\\$\$USER.SYS*

### **Operands:**

*d:* enter the letter of the drive from which the \$\$USER.FILE is loaded.

*|path|* enter the path of the directory where the \$\$USER.SYS file is located.

\$\$USER.SYS you must enter \$\$USER.SYS as the file name operand for this command.

### **Explanation:**

If a USERFILE command statement is not found in the CONFIG.SYS file, MOS looks for the \$\$USER.SYS file at the root directory of the boot drive when user security is active. If you create the \$\$USER.SYS file at a directory other than the root, and do not set the path in the CONFIG.SYS file, MOS will not find the file.



## VTYPE

The configuration command statement VTYPE allows a user to control certain aspects of MOS's memory allocation, and thereby, task size limits. The VTYPE statement is part of your configuration set up and is only possible when a memory management driver is being used. EGA and VGA systems can NOT use the optional F operand.

**Type:** Intrinsic

**Form:** VTYPE=1|2|3|4|5 {F}

### Operands:

- 1 VGA/EGA/CGA/MONO TEXT mix. A 16K video save area is reserved. (The area from A0000 to B0000 can be used as task memory by using the F operand.)
  - 2 MONO - TEXT ONLY. A 12K video save area is reserved. (The area from A0000 to B0000 can be used as task memory by using the F operand.)
  - 3 CGA only. A 16K video save area is reserved. (The area from A0000 to B4000 can be used as task memory by using the F operand.)
  - 4 CGA only. A 16K video save area is reserved. (The area from A0000 to B8000 can be used as task memory by using the F operand.)
  - 5 HERCULES MONO GRAPHICS. A 32K video save area is reserved. (The area from A0000 to B0000 can be used as task memory by using the F operand.)
- F this optional operand can only be used in combination with one of the previous operands. If used, the foreground task will be filled out to the new upper memory limit by allocating from extended memory, thus increasing the available task size. To use the F operand , no other hardware device (including your video adapter) can be mapped to the fill area address space.

# Configuration

---

## Explanation:

The VTYPE statement allows you to specify the maximum amount of memory PC-MOS is to reserve for the main console's video context area and all subsequent task's video save areas.

PC-MOS saves the video output from each task in their own video save area. This allows processing to continue to take place and screen updates to occur, even if the task is not currently being viewed. When the task is eventually brought to view the screen will be updated with the contents of this save area. If you are using EGA or VGA level graphics adapters, and your application software is running in an EGA/VGA graphics mode, PC-MOS does not save the video data when the task is not in view. Instead, MOS suspends that task until someone switches back into the task to view it.

The highest video mode that is saved under PC-MOS is Hercules (mono-graphics). This mode requires 32K of RAM to save the video graphics data.

The F operand can be used to instruct PC-MOS to fill in the unused area between A0000 and the new upper memory limit (where your video card starts mapping) with extended memory for use as task memory space. This will increase the size of the task available for your applications. If you intend to use the F operand, you MUST make sure that no other hardware devices on your system are attempting to use this memory area. For example, if you have an EGA or VGA board in your system, you can NOT use the F operand since these video adapters use memory space beginning at A0000.

The following table lists typical video adapter memory mapping addresses. (These are NOT always the same for different board manufacturers. Consult the manufacturer or your video adapter for more specific information.)

<u>Video Adapter</u>	<u>Memory Addresses</u>
Mono-Text Only	B0000 - B4000
CGA	B8000 - C0000
Hercules Mono Graphics	B0000 - C0000
EGA	A0000 - C4000
VGA	A0000 - C8000



In a PC-MOS system that has a variety of video types to emulate, select a VTYPE that will accommodate the highest video graphics mode that is saved under PC-MOS. The following table summarizes the various VTYPE options. A detailed explanation of each VTYPE follows the table.

VTYPE	Video Save Area Size	F option Task Fill Area
1 - VGA/EGA/CGA/Mono mix	16K	A0000 - B0000
2 - Mono (Text only)	12K	A0000 - B0000
3 - CGA only	16K	A0000 - B4000
4 - CGA only	16K	A0000 - B8000
5 - Hercules Mono Graphics	32K	A0000 - B0000

**If VTYPE is NOT used, the defaults are:**

The upper memory limit for tasks is available base memory (e.g. A0000 for a 640K system).

A 16K memory area is reserved for the video save area to allow for the largest possible need. Even if the video adapter in the master console is monochrome, which would only require a 12K video save area, a 16K area is reserved to handle CGA tasks which could be started using EMULINK or a CGA type terminal. (Monochrome tasks still only allocate 12K of extended memory for their video save buffer).

# Configuration

## VTYPE=1

VTYPE=1 is normally used on a system with a mix of VGA, EGA, CGA, and MONO (Text Only) workstations.

The new upper memory limit for tasks is B0000. This means that the area from A0000 up to B0000 can be used as task memory by using the F operand (if no other hardware conflicts).

A 16K memory area is reserved for the video save area.

## VTYPE=2

VTYPE=2 is used to tell MOS that only MONO (Text Only) tasks will be active on the system.

The new upper memory limit for tasks is B0000. This means that the area from A0000 up to B0000 can be used as task memory by using the F operand (if no other hardware conflicts).

Only a 12K memory area is reserved for the video save area.

If you have a monochrome adapter at the main console, and will NEVER have a CGA background task (using EMULINK or a CGA terminal) then using VTYPE=2F will raise the upper memory limit for tasks and reduce the memory needed for the video save area.

## VTYPE=3

VTYPE=3 can be used on systems with only CGA workstations. For example, a CGA host computer with PC EmuLink tasks.

The new upper memory limit for tasks is B4000. This means that the area from A0000 up to B4000 can be used as task memory by using the F operand (if no other hardware conflicts).

A 16K memory area is reserved for the video save area.

This VTYPE leaves the area from B4000 to B8000 available for the \$RAMDISK.SYS driver to use. (B4000 is the default area for this driver. Another area may be specified.)



NO MONOCHROME TASKS (e.g. terminals using a non-graphics terminal driver such as PCTERM) MAY BE USED WITH THIS OPTION SINCE THE MONOCHROME VIDEO MEMORY AT B0000 CAN BE USED AS TASK MEMORY.

In addition, there is a possibility that certain applications will test for memory at B0000 and B8000 to determine where they can do their direct video writing. Such an application would see task memory at B0000 and be misled into thinking it was on a monochrome system. A system crash would most likely result. Anyone who wants to use such applications will NOT BE ABLE TO USE VTYPE=3 OR VTYPE=4.

#### VTYPE=4

VTYPE=4 can be used on systems with only CGA workstations. For example, a CGA host computer with PC EmuLink tasks.

The new upper memory limit for tasks is B8000. This means that the area from A0000 up to B8000 can be used as task memory by using the F operand (if no other hardware conflicts).

A 16K memory area is reserved for the video save area.

The same considerations hold true for this option as for VTYPE=3 (no MONO/PCTERM tasks, etc.) with the following exception:

The area from B4000 to B8000 is NOT available as the default location for \$RAMDISK.SYS. This driver may still be used as long as another area is specified.

## Configuration

VTYPE=5

VTYPE=5 is used for handling Hercules mono graphics adapters. In order to select MOS VMODE HG1 or HG2, VTYPE must be set to 5.

The new upper memory limit for tasks is B0000. This means that the area from A0000 to B0000 can be used as task memory by using the F operand (if no other hardware conflicts).

A 32K memory area is reserved for the video save area.

The area from B4000 to B8000 is not available as the default location for \$RAMDISK.SYS. This driver may still be used as long as another area is specified for it.

The area from B8000 to C0000 is not available if VMODE HG2 is used, or if VMODE HG1 and CGA are both used at the same time.

VMODE HG1 is suitable for applications that only require the first hercules graphics page. HG2 is suitable for applications that require both hercules graphics pages.



## Device Names

A device may be any piece of hardware connected to your computer, such as a printer or CRT terminal. A device may also be a program that emulates hardware, such as a RAM disk. A device driver is a special routine or program that controls the action of the devices.

Several standard device drivers are built into MOS and you do not have to define them in the CONFIG.SYS file. These standard device drivers are:

- COMn - where n is 1 to 24 to define an asynchronous channel (only COM1 and COM2 are defined by default).
- CON - defines a console connected to your processor.
- LPTx - where x can be 1, 2 or 3 to define the available parallel ports (LPT1, LPT2, and LPT3) normally used for printers.
- AUX - same as COM1.
- PRN - same as LPT1.
- NUL - defines a null device that discards output and appears empty when read from.
- d - where d can be a letter from A to Z that defines a disk drive.

Several special device drivers are provided with MOS, but are not built-in. In order to use them, they must be set up in the CONFIG.SYS file with a DEVICE= command statement, in the same way as the drivers supplied with non-industry standard devices. The following pages explain each special device driver provided with MOS that you may want to set up.

A list of devices active under PC-MOS can be viewed by entering:

REMDEV

at the system prompt. Devices in the list with a preceding \$\$ were loaded through a statement in your CONFIG.SYS file.

# Configuration

---

## \$EMS.SYS

\$EMS.SYS sets up a device driver for emulating the Lotus-Intel-Microsoft Expanded Memory Specification (EMS). This driver now supports both EMS 4.0 and 3.2 specifications. The memory buffer size for paging is 64K. The actual amount of EMS memory set up for bank switching is assigned in each task with the MOSADM EMSLIMIT command and is only limited by the available memory in your computer. The \$EMS.SYS device driver is loaded with a DEVICE= statement in your CONFIG.SYS file.

**Form:** DEVICE={d:}{|path}|\$EMS.SYS {bufadr}

### Operands:

*d*: enter the drive letter of the drive on which the \$EMS.SYS file resides. MOS defaults to the current drive if a drive is not entered.

|*path*| enter the path of the subdirectory where the \$EMS.SYS file is located. MOS defaults to the current directory if a directory is not entered.

*bufadr* you may optionally enter the buffer hex address range to be used for paging. The 64K EMS buffer address range defaults to E0000 (through F0000) if not entered. (The buffer address must always be on a 4K boundary, where the last three digits are zeros.) Be sure the default address, or the address you specify, is not used by MOS as free memory (see the FREEMEM command statement).

### Explanation:

The \$EMS.SYS device driver lets you access and use extended memory for processing as though it were expanded memory. This eliminates the need for additional hardware, such as an expanded memory board, for products such as Lotus 1-2-3®, Rel. 2 and up, that may need to access expanded memory.



For example, if you want to set up the \$EMS.SYS driver to use extended memory for EMS bank switching beginning at address C0000, and the \$EMS.SYS file resides in the PCMOS subdirectory on the C drive, the command statement in CONFIG.SYS would be:

```
DEVICE=C:\PCMOS\$EMS.SYS C0000
```

To be sure the memory for EMS paging is not used by MOS, you would need to set up a FREEMEM command statement in your CONFIG.SYS file to omit memory from C0000 to D0000 from free memory. The FREEMEM command statement for this example could be:

```
FREEMEM=D0000,F0000
```

# Configuration

## \$MOUSE.SYS

\$MOUSE.SYS is used for installing a mouse on the host computer or on a VNA or SunRiver workstation. You MUST use this driver instead of the MOUSE.SYS driver version supplied with your mouse. (You still, however, use the MOUSE.COM program that was provided with your mouse.)

**Form:** DEVICE={d:}\|path\|\$MOUSE.SYS

### **Operands:**

- d:* enter the drive letter of the drive on which the \$MOUSE.SYS file resides. MOS defaults to the current drive if a drive is not entered.
- |*path*| enter the path of the subdirectory where the \$MOUSE.SYS file is located. MOS defaults to the current directory if a directory is not entered.

### **Explanation:**

The driver must be entered in your CONFIG.SYS file as follows:

```
DEVICE=$MOUSE.SYS
```

If the driver resides in a subdirectory on your disk, you must enter the subdirectory path with the statement, for example:

```
DEVICE=C:\PCMOS\$MOUSE.SYS
```

Since the mouse driver must be loaded after all serial drivers (\$SERIAL.SYS, SRTERM.SYS, VNA.SYS, etc.), it must be listed after all of those device driver statements in your CONFIG.SYS file.

Once the mouse driver is active on your system you must initialize the mouse in the task in which it will run. This is done with the MOS utility command:

```
MOS MOUSE n[,r]
```



where "n" is the COM port number that the mouse is connected to and "r" is the baud rate.

The MOS MOUSE command must be entered BEFORE you run the mouse program (MOUSE.COM) in your task to use the mouse.

The MOS utility command "MOS MOUSE off" may be entered in a task if it is necessary to turn off mouse support in that partition.

See the Multi-Tasking/User chapter for more information on these MOS utility commands and installing mice.

# Configuration

## \$PIPE.SYS

\$PIPE.SYS lets you define a character device to pipe information between partitions. \$PIPE.SYS must be set up in the CONFIG.SYS file with a DEVICE= command statement. You can define as many of these devices as you need, and assign the name of the driver and the buffer size for each one. Your only limitation is the amount of memory available in your computer. These devices allow partitions to communicate and share information.

**Form:** DEVICE={*d:*}/{*path*}|\$PIPE.SYS *devname*,{*bufsize*} [/n]

### **Operands:**

- d:* enter the drive letter of the drive on which the \$PIPE.SYS file resides. MOS defaults to the current drive if a drive is not entered.
- [*path*] enter the path of the subdirectory where the \$PIPE.SYS file is located. MOS defaults to the current directory if a directory is not entered.
- devname* you may assign up to 8 alphanumeric characters for each device name, but the name must begin with an alpha character. For example, PIPE1 is a valid device name where 1PIPE is not. The name you assign to a pipe device must be unique; it cannot be a name assigned to any other files or devices.
- bufsize* the buffer size for each pipe device may be from 1 to 16384 bytes of memory, taken from the System Memory Pool. The PIPE buffer size defaults to 64 bytes if not entered.
- /n if you use the /n operand, the pipe driver will return an "end of file" message if a read request is sent to an empty buffer, rather than waiting for input to the file.



### Explanation:

When pipe devices are defined, any user may send input to the device in the same manner as sending input to any other device. The device holds the input until it is retrieved. For example:

```
.COPY MYFILE.DOC PIPE1
```

This command copies the content of MYFILE.DOC to the device PIPE1. The information remains in the buffer for PIPE1 until it is cleared by another command.

Note that if PIPE1 has a buffer size of 100 bytes and MYFILE.DOC is 200 bytes, the first 100 bytes are output to PIPE1. MOS will halt and wait until more space is available in the buffer to send the next 100 bytes.

If the buffer for PIPE1 is not cleared, the partition sending MYFILE.DOC will lock in a waiting state.

Another user may retrieve the information in PIPE1 by copying from the device. For example:

```
.COPY PIPE1 REVIEW.DOC
```

MOS does not control any locking of pipe devices. If two users are sending data to a device at the same time, the data will be mixed together. In the same way, two users accessing the data at the same time will receive scrambled data.

# Configuration

## \$RAMDISK.SYS

\$RAMDISK.SYS is a device driver that allocates extended memory to be used as a virtual disk, which may also be referred to as a RAM disk. The RAM disk will take the next available drive letter identifier by which it may be accessed.

**Form:** DEVICE=[*d:*]\[*path*]\\$RAMDISK.SYS *nnnnK*,[*bufadr*]

### **Operands:**

- d:* enter the drive letter of the drive on which the \$RAMDISK.SYS file resides. MOS defaults to the current drive if a drive is not entered.
- [*path*] enter the path of the subdirectory where the \$RAMDISK.SYS file is located. MOS defaults to the current directory if a directory is not entered.
- nnnnn* enter the amount of extended memory in kilobytes, which may be from 1 kilobyte (1K) to 16 megabytes (16384K). Defaults to 64K if not entered.
- K* you must enter a K to document that the memory size is in kilobytes.
- bufadr* you may optionally enter the buffer address to be used for paging. Defaults to address range B4000 to B8000 if a buffer address is not entered. (The buffer address must always be on a 4K boundary, where the last three digits are zeros.) A 16K address range is used for paging regardless of RAM disk size. Be sure the default address, or address you specify, is not used by MOS as free memory (see the FREEMEM command statement).



### Explanation:

The memory defined in the command statement directly emulates a disk drive, which may be accessed as any other drive. Because a RAM disk allows data to be stored directly in the computer's memory, data access time is much quicker than accessing data from a physical drive. Any data stored on the RAM disk is lost when power to the computer is turned off.

If you want to set up the \$RAMDISK.SYS device to use 128K of extended memory beginning at address DC000, the command statement in CONFIG.SYS would be:

```
DEVICE=$RAMDISK.SYS 128K,DC000
```

To be sure the memory for \$RAMDISK.SYS paging is not used by MOS, you would need to set up a FREEMEM command statement in your CONFIG.SYS file to omit memory between DC000 and E0000 from free memory. The FREEMEM command statements for this example could be:

```
FREEMEM=C0000,DC000
```

```
FREEMEM=E0000,F0000
```

# Configuration

## \$SERIAL.SYS

\$SERIAL.SYS is a device driver that sets up a standard buffered interface to non-intelligent serial port devices. The DEVICE= command statement for \$SERIAL.SYS must be present in your CONFIG.SYS file to set up serial ports to support local terminals, printers, mice, and modems for remote terminals.

If you are using only the standard serial ports, COM1 and COM2, \$SERIAL.SYS defaults to the standard assignments for COM1 and COM2 and you do not need to enter the driver. (Entering a DEVICE= command statement for \$SERIAL.SYS allows you to change the defaults and/or define additional ports and will override the standard assignments.)

Ports that are to be used for modems with standard DOS communications packages (like CrossTalk® and ProComm™) must NOT be defined with \$SERIAL.SYS.

**Form:**      DEVICE=[*d:*]\[*path*]\\$SERIAL.SYS /AD=*nn*,IB=*nnnnnn*, ~  
OB=*nnnnnn*,HS={N|D|X|P|R},IN=*n*,CN=L|R|T|...

### **Operands:**

- d:***      enter the drive letter of the drive on which the \$SERIAL.SYS file resides. MOS defaults to the current drive if a drive is not entered.
- [*path*]**      enter the path of the subdirectory where the \$SERIAL.SYS file is located. MOS defaults to the current directory if a directory is not entered.
- AD=**      the port address may be from 0 to 8000 hex. Defaults to 03f8 (COM1) and 02f8 (COM2) if not entered.
- IB=**      the input buffer may be from 32 to 65536 bytes. Defaults to 64 bytes if not entered. This is sufficient for most incoming keyboard or mouse data.



**OB=** the output buffer can be from 16 to 65536 bytes. Defaults to 1024 bytes if not entered. This is correct for ASCII terminals. PC EmuLink™ terminals should normally be set to 16384 bytes. If you are using a serial printer that has too small an internal buffer you could increase OB to from 2048 to about 8192 bytes (depending on the printer) to improve your printer performance.

**HS=** Handshaking defaults to N - None if not entered, but can also be set to:

- N - None
- D - DTR (data terminal ready, data set ready)
- X - XOFF (receiver controlled on and off)
- P - XPC (receiver controlled on and off using different characters: 65H=xon, 67H=off)
- R - results in RTS (request to send, clear to send) protocols

**IN=** the hardware interrupt level (IRQ) may be set at 2 through 7. Defaults to IRQ 4 and IRQ 3 for ports 1 and 2 respectively. Some serial adapters that provide multiple ports may use one interrupt for all the port's addresses.

**CN=** defines communication for a local or remote terminal. CN=L defines a LOCAL (directly connected) terminal, the default. CN=R defines a REMOTE terminal connected through a modem. CN=T also defines a remote terminal, but with the TASK-RESTART option.

**~** a ~ preceded by a space tells MOS that the next line is a continuation of the attributes.

### Explanation:

Data coming into non-intelligent serial port devices must be buffered or the system will be slowed down since the computer's microprocessor (CPU) must handle all the serial I/O activity. The \$SERIAL.SYS driver sets up a standard buffered interface for non-intelligent serial ports for local terminals, printers, mice, and modems for remote terminals.

Intelligent serial devices do not require buffering since they have their own on-board CPUs to handle the serial I/O.

# Configuration

Ports that are to be used for modems with standard DOS communications packages (like CrossTalk and ProComm) must NOT be defined with \$SERIAL.SYS. These programs write directly to the port and wait for a response, which they won't get if the port is buffered. Instead, use the MOS USEIRQ command to reserve the interrupt for that port before running the communications program. When done, use the MOS FREEIRQ command to release the interrupt.

PC-MOS defaults to using COM1 (03f8) and COM2 (02f8) if no \$SERIAL.SYS statement is entered. If you require more than the standard COM1 and COM2 serial ports, or need to change some of the default settings for those ports, you should enter a DEVICE=\$SERIAL.SYS statement in your CONFIG.SYS file to set up the ports.

Only one DEVICE= command statement for the \$SERIAL.SYS device driver is entered in the CONFIG.SYS file, followed by the attributes for each serial port assignment. You may enter a maximum of 24 port assignments with \$SERIAL.SYS.

NOTE: You must enter a space before the forward slash in the first port assignment, but not before any subsequent port assignments.

A comma or a space is used as a delimiter between each attribute. If you omit an operand (accept the default), DO NOT enter a comma in place of the operand. Instead, continue the command line with the next operand you want to specify. For example:

```
DEVICE=$SERIAL.SYS /AD=03F8,IN=4
```

The attributes for each port are separated by the forward slash (/). If you need to enter the attributes for several devices, the attributes will likely wrap on several 80 column lines. To let MOS know that the next line is a continuation of the attributes, a tilde (~) is used at the end of the line. Always include a space before the ~. For example:

```
DEVICE=$SERIAL.SYS /AD=03f8,IB=128,OB=2048,HS=N, ~  
IN=4/AD=02f8,IB=64,OB=1024,HS=N,IN=3
```



**IMPORTANT:** Optional operands entered for one port carry over to all subsequent ports defined unless manually changed back to the default or another value. In the above example, the IB and OB values for the second port would have also been 128 and 2048 had they not been changed back to the defaults by entering the new values for those operands.

The order of port address assignments corresponds to the port addresses you enter with the .ADDTASK command. The first assignment entered with \$SERIAL.SYS is port 1, the second is port 2, the third is port 3, etc. For example:

Port 1

Port 2

Port 3

```
$SERIAL.SYS /AD=03f8,IB=64.../AD=02f8,IB=.../AD=03e8,IB=...
```

When you invoke the .ADDTASK command and enter port numbers for specific terminals, you must be sure the number corresponds to the correct serial port assignment. The operand AD=03f8 assigns COM1 as the serial port to which the terminal for this partition is connected. The operand AD=03e8 would assign COM3 as the serial port to which the terminal for this partition is connected.

Because \$SERIAL.SYS contains the serial port assignments, there may be port numbers you will never assign in an .ADDTASK command. Using the previous example, if you have a modem connected to address 02f8, you would never assign port 2 with an .ADDTASK command. That port is reserved for the modem connection.

The "CN=" operand is used to indicate whether your connection is direct to a local terminal or through a modem to a remote terminal. CN=L defines a LOCAL (directly connected) terminal. CN=R defines a REMOTE terminal connected through a modem. CN=T defines a remote terminal with the TASK-RESTART option.

For example:

```
DEVICE=$SERIAL.SYS /AD=03f8,IN=4,CN=R/AD=02f8,IN=3,CN=L
```

defines port 1 as a REMOTE connection and port 2 as a LOCAL connection. If the CN= operand is not used, the ports default to the LOCAL definition. Remember, however, that an operand specified for one port will be in effect for all subsequent ports, unless changed. In the above example, port 2 would have used the CN=R definition from port 1 if it hadn't been changed with the CN=L entry.

Using the REMOTE option affords better control of remote communications by telling the serial driver to monitor the carrier detect line. When carrier detect is off, the driver will ignore any characters from the port. When carrier detect is on, the driver will accept incoming characters from the port. Characters can be output to the port regardless of the state of the carrier detect line.

The CN=T TASK-RESTART option offers the same control as the REMOTE option with one additional feature. If your connection is broken and the carrier detect signal drops, the TASK-RESTART option will automatically restart the task. This will make it much more likely that the modem will be properly set up to receive the next incoming call, since the task was not left hanging in the middle of an application. Also, if security was active in the task, restarting the task will re-run the batch file or security command that establishes security in that task. This will ensure that no one calling in can gain access to the task without entering the necessary user ID and password.

---

# CHAPTER 3:

## FILES AND DIRECTORIES

---

Introduction .....	3-2
Naming Files .....	3-2
File Extensions .....	3-3
Displaying File Names .....	3-3
File Characteristics .....	3-4
File Sizes .....	3-4
Updated Date and Time .....	3-4
File Attributes .....	3-4
Class and User ID .....	3-5
Creation Date and Time .....	3-5
File Sharing .....	3-5
Wildcard Characters .....	3-5
File Maintenance .....	3-7
Directories .....	3-8
Directory Structure .....	3-9
Directory Names .....	3-10
Organizing Directories .....	3-11
Directory Maintenance .....	3-11
Displaying a Directory .....	3-13
Displaying a Directory Structure .....	3-14
MOS-D191 .....	3-1

# Files & Directories

## Introduction

MOS lets you maintain files as the method to store and track information. A file may contain text, system information, programs, or commands, such as those found in batch files. You may create a file with any text editor, a word processor, or with the MOS Editor, .ED. You may also copy files from an application program onto your computer's hard disk.

You may further organize your files under MOS with a directory structure. The directory structure lets you group files, and define a path to where the files are located.

This chapter explains specific information about creating and maintaining files, and the directory structure for grouping and accessing files.

## Naming Files

The complete name of a file consists of two elements, the file name and an extension. The name of the file may contain up to eight characters, and the extension may contain up to three characters. A period is used to separate the file name from the file extension. For example:

filename.ext

Each file you create must have a unique name. Duplicate file names may exist as long as they are in different directories. You may use any combination of alphanumeric characters for the name and extension of your file. However, you may not use any of the following characters in any part of a file name.

> < [ ] : " ; = + ! . \ / \* ? ,

The names of the files provided with MOS are reserved in the sense that MOS will "look" for them, and when found will execute them. You should not name a file with the same name as a MOS file. Unless you are technically knowledgeable, these files should not be overwritten or modified.



You should never name a file with the same name as any of the reserved devices provided with MOS. Some of the reserved device names in MOS are:

AUX	CON	NUL
COMn	LPTn	PRN

### File Extensions

MOS uses the filename extension to identify several special types of files, and will treat these files accordingly. You may choose any three-character string for your file extension. The special extensions that MOS identifies and their uses are:

- .BAT    MOS treats any file with a .BAT extension as a batch file, which is a collection of commands. When this file is invoked, MOS will execute the commands contained in the file.
- .COM    MOS treats any file with a .COM extension as if it contained executable machine language. Most .COM files are program files, and may be up to 640K in size.
- .EXE    MOS treats any file with a .EXE extension as if it contained executable machine language. MOS looks for a header record at the top of the file that tells MOS how and where to load the file. The size of these files is limited by the size of your disk volume.
- .SYS    MOS treats any file with a .SYS extension as a MOS system file. These files have various special purposes.

### Displaying File Names

MOS tracks and maintains information associated with each file. This information appears with the file name when you display a directory with the .DIR command. A directory display appears on your screen similar to the following:

# Files & Directories

filename.ext	size	---updated---	attrib	cls	user	---created---
LETTER.TXT	4,152	4-18-87 11:25				4-16-87 14:34
MEMO.TXT	3,196	5-01-87 18:32				5-01-87 18:32
CONTRACT.TXT	15,342	5-25-87 17:33	S	A	SUE	4-21-87 10:45

## File Characteristics

A directory will display the files along with their characteristics. The following is a description of these characteristics.

### File Sizes

Following the file name and extension is the file "size", which is the number of bytes the file occupies. The size of a file under MOS is limited by the size of the volume or media on which it resides. However, you should not make a file larger than you really need.

### Updated Date and Time

MOS tracks the date and time when a file is accessed and has information written to it. The last date and time when information was written to an existing file appears in a directory display in the "updated" column. If you use MOS to merely display the file and nothing is actually written to the file, the date and time remain unchanged. The date displays in the format you select with the COUNTRY command statement in your CONFIG.SYS file.

### File Attributes

MOS tracks several attributes for each file. These attributes also appear on the directory display in the "attrib" columns. An asterisk (\*) appears to indicate that the file has been changed since the last time it was backed up with the .EXPORT command. An R appears for a file that has a read-only attribute, and nothing can be written to the file. An S appears as an attribute for any secured file. If any hidden files exist on your computer, MOS will display an H attribute.



## Class and User ID

The "cls" column displays the class that is assigned to a file when MOS security is active. The "User" column displays the user ID of the user who created the file.

## Creation Date and Time

MOS tracks the date and time a file is created. This is the internal system date and time known to MOS, and appear on a directory display in the "created" column in the format determined by the COUNTRY command statement in your CONFIG.SYS file.

The creation date and time will always be associated with the file and may not be changed. If you create a copy of the file, the copy will carry the same creation date and time as the original file. If you copy files and concatenate them, the resulting file will show a new creation date and time.

Note that some applications may rename or delete a file while you are using it, and then create a new file when you save the output back to disk. When this occurs, MOS will always recognize the output as the creation of a new file and assign the current date and time as the creation date and time of the file.

## File Sharing

Whether or not two users can access the same file at the same time depends on how the file was opened, and the type of access. If an application program is controlling the file, the application controls which users may access the file at the same time.

## Wildcard characters

MOS recognizes a ? and a \* as wildcard characters. You may place either of these characters in a file name to indicate that the position in the filename may contain any characters. Wildcards are generally used to search for files when only a partial name is known, or to search for files that have a common name characteristic.

## Files & Directories

For example, you may want to use the directory command and display the names of all files that begin with M and have a .DOC extension. The ? indicates that any character may occupy a single position. You could enter the file name as:

m???????.doc

This would display a directory of all files that begin with m and carry a .DOC extension. The ?????? represent the positions that could be occupied by any characters. If you knew that the 8-character name of the file you wanted began with m and ended with t, you could enter the file name as:

m??????t.doc

MOS would display a directory of all 8-character file names that begin with an m, end with a t and have a .DOC extension.

The \* is equivalent to the number of ?'s required to complete the file name or extension. For example:

m\*.doc = m???????.doc

abc.x\* = abc.x??

a\*c.xyz is not a legal filename. The "c" is ignored.

If you enter a file name as:

\*.doc

MOS would display all files that have a .DOC extension. You could include all files on a disk or in a directory by entering the file name to include with a command as:



## File Maintenance

MOS provides several commands that you may use to maintain your files. The .ERASE command lets you delete a file from the system when it is no longer needed. The .RENAME command lets you rename a file or move it from one directory to another. The .COPY command lets you create a new copy of an existing file.

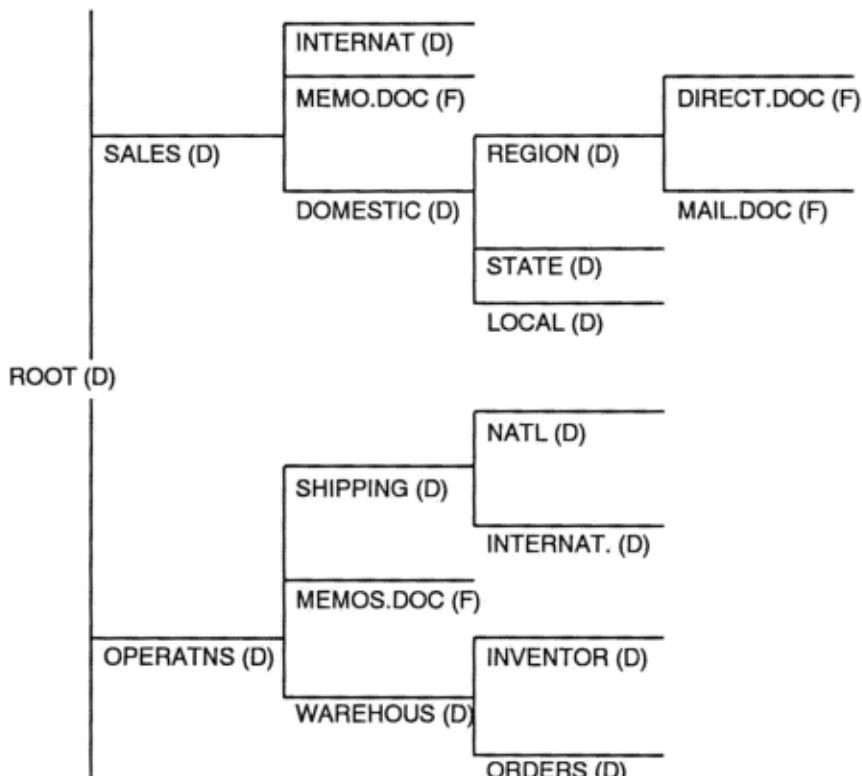
You may want to periodically make back up copies of your files for safekeeping. You should use the .EXPORT command to make your back up copies. .EXPORT differs from .COPY in that it creates a compressed copy of the file, and allows the file to span multiple diskettes. You may restore exported files only with the .IMPORT command. These commands are explained fully in the General Command chapter of this manual.

# Files & Directories

## Directories

MOS provides a directory structure for grouping files. This structure begins with a base called the root directory. From the root you may add other directories. There are no other directories in MOS until you create them. Each disk you use with MOS may have its own unique directory structure.

You may create a directory at the root that contains several other directories, and those directories may also contain additional directories. However, all directories you create have a path back to the base level of the root. The following diagram illustrates this directory structure.



D = Directory

F = File



## Directory Structure

Note that we define the base level of MOS as the root. If you never create a directory, all files will then reside at the root. With a large capacity disk, this would make it difficult for you to locate a specific file, and may take MOS a long time find the file.

In our illustration, a directory name SALES has been created at the root. The root directory is the parent of SALES. The SALES directory is the parent for two more directories, DOMESTIC and INTERNAT. MOS will recognize each level of directory only if the levels are separated with a backslash. For example:

\SALES\DOMESTIC

The first backslash indicates that the path begins from the root directory. The second backslash indicates the path from SALES to DOMESTIC. MOS follows this format as the path to the requested directory. However, if you are in the parent directory (here, the root), it is not necessary to enter the first backslash as part of the path. For example:

SALES\DOMESTIC

In our illustration, DOMESTIC is also a parent directory to three more directories, LOCAL, STATE and REGION. If you are at the root and want to access a file in the REGION directory, the path would be:

SALES\DOMESTIC\REGION\MAIL.DOC

If DOMESTIC is the current directory, the path may be:

REGION\MAIL.DOC

If you want to move from the REGION directory to the ORDERS directory, the path must begin from the root and must therefore include the beginning backslash. For example, the following path instructs MOS to return to the root (indicated by the first backslash) and move through the OPERATNS and WAREHOUS directories to get to the ORDERS directory:

\OPERATNS\WAREHOUS\ORDERS

# Files & Directories

From the REGION directory, or any directory, you may return to the root by entering only the first backslash. For example, if you are currently working in the ORDER directory, the following path returns you to the root.

\

You may move to a directory that is on another disk by including the letter of the drive where the directory exists as part of the path. For example, to move from a directory on drive A to the ORDERS directory on drive C, you could enter the path as:

C:\OPERATNS\WAREHOUS\ORDERS

## Directory Names

You may create a directory and give it any name you want, subject to the same limitations as file names. A directory name, like a file name, may be up to eight characters and may include a period and an optional extension of up to three characters.

All the levels of directory names included in a path from the root may not exceed 64 characters. The number of different directory paths you may create from the root is determined by your disk media. From any directory level other than the root, there is no limitation to the number of different paths you can create. At the root you may create:

- 512 directory entries at the root of a hard disk.
- 112 directory entries at the root of a 320K or 360K double-sided floppy disk.
- 224 directory entries at a 1.2 megabyte floppy disk.
- 64 directory entries at the root of a 160K or 180K single-sided disk. Note that you may read this disk media with MOS, but MOS will not format or write to this media.



## Organizing Directories

There are three commands in MOS that let you make (.MD) and remove (.RD) directories, and change (.CD) from one directory to another. These commands are explained in detail in the General Command chapter of this manual. The following are some general guidelines for using these commands to organize your directories.

The directory structure lets you group files, and define a path to where the files are located. When you want to access a file that is not in the current directory, you may enter the correct path to the directory where the file is located.

You may want to group specific files by their type, or group specific application programs and files separately. For example, you may want to place a spreadsheet program in a separate directory from a communications program. Or, you may have a word processing application and want to create separate directories for each users' files.

## Directory Maintenance

You may make a directory with the .MD command. You may then copy any files into that directory, or create your own files. If you want to copy existing files into a different directory, you may use the .RENAME command to move them. Note that the name of each file in a particular directory must be unique.

Note that you can only create one level of a directory path at a time. For example, you may want to create a MEMOS directory at the root, and from the MEMOS directory create another level that includes directories for each month of the year. You must first make the directory named MEMOS. For example:

```
.MD MEMOS
```

## Files & Directories

When this directory exists, you may then create the next level of directories. You may want a directory for each month of the year. From the root you may create a directory for January with the following command:

```
.MD MEMOS\JAN
```

If you were already at the MEMOS directory, you could make the JAN directory with the following command:

```
.MD JAN
```

To move from the root directory to the JAN directory, you could enter the command:

```
.CD MEMOS\JAN
```

You may display the path of the current directory by typing the .CD command with no operands. If you were working in the JAN directory and entered .CD with no operands, MOS would display:

```
\MEMO\JAN
```

Note that two periods are used to move from a current directory to its parent directory. For example, if you are currently working in the JAN directory and want to move to the MEMOS directory, you would enter:

```
.CD ..
```

If you made a directory for each month of the year at the MEMOS directory, you could move from the JAN to the FEB directory by instructing MOS to change to the parent and then move to FEB. For example:

```
.CD ..\FEB
```



When the directory for January files is no longer necessary, you may want to remove the JAN directory to free disk space. You must first delete any files in the directory. If there are other levels of directories from the JAN directory, you must also remove them. You may then enter the following command to remove the January directory:

```
.RD MEMOS\JAN
```

## Displaying a Directory

The .DIR command lets you display the contents of a directory. For example, to display the content of the current directory, you could enter:

```
.DIR
```

The directory will appear on your screen similar to the following:

filename.ext	size	---updated---	attrib	cls	user	---created---
LETTER.DOC	2987	1-19-87 11:25				1-11-87 14:54
LETTER.SAV	3256	1-12-87 15:21				1-12-87 15:21
JAN		(directory)				1-11-87 14:58

Note that the file names in the current directory appear on the display. The next directory level created at the current directory also appears. The word "directory" appears on the display to let you know this is a directory name, along with the time and date the directory was created.

## Displaying a Directory Structure

The .DIRMAP command lets you display any directory structure on a disk. You may select to include file names in the display, or only directory names. You may also begin the display from the root, or from any level of directory.

For example, you may display the directory structure beginning at the root by entering the .DIRMAP command without specifying a path. The directory will appear similar to the following:

Directory	Files	Bytes
C:\	93	1441916
C:\MOS	12	56337
C:\WORK	14	1883
C:\WORK\JIM	21	213649
C:\MEMOS	18	62312
C:\MEMOS\JAN	10	89329
C:\MEMOS\FEB	6	9362

The directories appear in the map in the order they are found on the disk. You may use the path with the .DIRMAP command to display all levels of a specific directory. For example, the following command will display the MEMO directory as well as the MEMO\JAN directory on the default drive:

```
.DIRMAP \MEMO
```

The .DIRMAP command is explained in detail in the General Command chapter of this manual.

---

# CHAPTER 4: GENERAL COMMANDS

---

Introduction .....	4-3
Intrinsic and Extrinsic Commands .....	4-3
Extrinsic MOS Commands .....	4-4
Intrinsic MOS Commands .....	4-5
Invoking Commands .....	4-6
Understanding Command Conventions .....	4-7
Input/Output Redirection .....	4-9
General Commands .....	4-11
.ADDDEV .....	4-11
.ALIAS .....	4-13
.BREAK .....	4-16
.CD .....	4-17
.CLS .....	4-19
.COMMAND .....	4-20
.COMPFILE .....	4-22
.COPY .....	4-25
.DATE .....	4-32
.DIR .....	4-33
.DIRMAP .....	4-36
.DISKCOPY .....	4-38
.DISKID .....	4-40

## General Commands

---

### General Commands (continued)

.ENVSIZE . . . . .	4 - 42
.ERASE . . . . .	4 - 43
.EXCEPT . . . . .	4 - 45
.EXPORT . . . . .	4 - 49
.FILEMODE . . . . .	4 - 53
.FORMAT . . . . .	4 - 55
.HELP . . . . .	4 - 58
.IMPORT . . . . .	4 - 59
.MD . . . . .	4 - 61
.MORE . . . . .	4 - 63
.MSORT . . . . .	4 - 64
.MSYS . . . . .	4 - 67
.ONLY . . . . .	4 - 68
.PATH . . . . .	4 - 71
.PROMPT . . . . .	4 - 73
.RD . . . . .	4 - 76
.REL . . . . .	4 - 77
.REMDEV . . . . .	4 - 78
.RENAME . . . . .	4 - 79
.SEARCH . . . . .	4 - 82
.SET . . . . .	4 - 84
.TIME . . . . .	4 - 86
.TYPE . . . . .	4 - 88
.VERIFY . . . . .	4 - 90
.WVER . . . . .	4 - 93



## Introduction

MOS contains many commands that you will use on a routine basis, and many commands for specialized functions. The routine commands are explained in this chapter. Specialized commands are explained in other chapters along with the necessary details for using them.

### Intrinsic and Extrinsic Commands

All commands in MOS fall into one of two categories; either intrinsic or extrinsic. Intrinsic commands are loaded into memory when you boot MOS and are always available. Extrinsic commands reside in the .COM and .EXE files that are provided with MOS, and are actually programs that load from disk when the commands are invoked.

The intrinsic commands and the MOS batch language are contained in the \$\$SHELL.SYS file that is loaded into memory each time you boot MOS. The COMMAND.COM file also loads into memory when you boot MOS, and provides the method of access to intrinsic commands. You can execute an intrinsic command regardless of what disks are available.

Extrinsic command files should be loaded on your hard disk, and may reside at the root directory, or in a different directory. You may simplify the execution of extrinsic commands by placing them in a separate directory on your hard disk. For example, you may create a directory on drive C named MOSCOM. Then copy all the .COM and .EXE files provided with MOS to this directory. In your AUTOEXEC.BAT file, set the .PATH for MOSCOM so that these files are always available, for example:

```
.PATH=c:\moscom;
```

Following are the MOS commands listed by their type, intrinsic or extrinsic. Within each type, the commands are listed by usage, either general commands explained in this chapter, or specialized commands that are explained in other chapters.

# General Commands

## Extrinsic MOS Commands

### Installation Commands

.HDSETUP

### General Commands

.ADDDEV	.EXPORT	.MSORT
.ALIAS	.FILEMODE	.MSYS
.COMPFILE	.FORMAT	.REMDEV
.DIRMAP	.HELP	.SEARCH
.DISKCOPY	.IMPORT	.VERIFY
.DISKID	.MORE	

### Multi-Tasking/Multi-User Commands

.ADDTASK  
.KEYMAP  
.REMTASK  
.MOS (including all utility functions)  
.MOSADM (including all utility functions)

### MOS System Editor

.ED (and all commands in the Editor)

### MOS Debug

.DEBUG (and all commands in .DEBUG)

### Security Commands

.CLASS

### Print Spooler

.PRINT  
.SPOOL



## Intrinsic MOS Commands

### Configuration Command Statements (used only in CONFIG.SYS file)

8087	DEVICE	SLICE
CACHE	FREEMEM	SMPSIZE
COUNTRY	MEMDEV	USERFILE
DESNOW	SHELL	VTYPE

### General Commands

.BREAK	.ENVSIZE	.REL
.CD	.ERASE	.RENAME
.CLS	.EXCEPT	.SET
.COMMAND	.MD	.TIME
.COPY	.ONLY	.TYPE
.DATE	.PATH	.WVER
.DIR	.PROMPT	
.DOT	.RD	

### Batch File Commands

.ABORT	.FOR IN DO	.PAUSE
.AUTOCD	.GOTO	.REM
.BATECHO	.IF	.STOP
.CALL/.RETURN	.INSERT	.TEXT/
.ECHO	.KEY	.ENDTEXT
.FLUSH	.NEXT	

### Security Commands

.SIGNOFF	
.SIGNON	

# General Commands

## Invoking Commands

With the exception of the command statements used in the CONFIG.SYS file, all MOS commands are shown in this manual with preceding periods. When you first boot MOS, the requirement for the preceding periods is turned off and you need not enter them.

When you enter a command, MOS first checks for an intrinsic command that is loaded in memory. If the command is not found, MOS then searches for an extrinsic command. If not found, the command is not invoked.

The .DOT command lets you control the checking for a preceding period with MOS commands. This command lets you keep the MOS command names from being confused with files or executable programs that have the same name. You may use the .DOT command to turn on the requirement for the preceding periods. If you do so, you should then always enter the preceding period when you enter a MOS command.

If .DOT is on and you enter an intrinsic command without the preceding period, MOS first looks on the current drive and directory to find the command. MOS then searches each directory defined in the current path, looking for a .COM, .EXE, or .BAT file, in this order, with the same name as the command. If the command is not found, MOS will then look for an intrinsic command loaded in memory. Invoking an intrinsic command without the preceding period may therefore cause a slower response.

If .DOT is on and you enter an extrinsic command without the preceding period, MOS checks for the existence of an extrinsic command first. If not found MOS then looks for an intrinsic command.

### .DOT

Each time you boot MOS, the .DOT command defaults to OFF, and MOS will not expect a period to precede commands. MOS will look for intrinsic and extrinsic commands in the correct order.

The form for the .DOT command is:

.DOT {on|off}



If you want the preceding periods to be required with commands you may enter the .DOT on command. You should then enter the period with all MOS commands, but not the command statements used in the CONFIG.SYS file.

You may place the .DOT on command in your AUTOEXEC.BAT file if you always want the preceding period required. Enter .DOT with no operands to display the current setting of the .DOT command.

## Understanding Command Conventions

Every command in MOS has a specific purpose, and different options that may be used when the command is invoked. To help you understand the way commands are presented, several conventions have been developed. They are:

- |              |   |
|--------------|---|
| COMMAND      | all command names throughout this manual are shown in capital letters. Intrinsic commands are always in memory and never require a preceding drive and path. Extrinsic commands require a preceding drive letter and/or path only if not included with the .PATH command in the AUTOEXEC.BAT file, and if the command is not at the current drive and/or directory. |
| d:           | the drive letter may be included before a command (in the path) to tell MOS where the command is located. The d: is included with the operands of the command, only when a drive may be part of the command.  |
| delimiter    | delimiters are used to separate a series of operands in a command. A delimiter may be a comma, a semi-colon, an equal sign, a tab, a space or a backslash. The command form includes the delimiters that are valid for that command.  |
| dirname      | if a specific directory name is required in a command, it is represented in the command form as dirname.  |
| editing keys | when entering a command at the system prompt, you may use any of the editing keys explained in the first chapter of this manual.  |

## General Commands

---

filename	includes a drive letter if not the default drive, the path if not the current directory, and the name of the file including the file extension. For example, a filename entered as an operand could be:  D:\PAYROLL\HAROLD\PRMENU.BAS
	If D is the default drive and HAROLD is the current directory, the filename could be PRMENU.BAS.
operand	all available operands are shown with each command. An operand may be a drive letter, a filename, a switch that sets a command to process a certain way, or some other parameter. Some commands have a large choice of operands. In the examples shown under "Form:" the text of some commands may wrap to the following line. However, when you type the operands for a command at your keyboard, DO NOT force the operands to wrap to a new line. The operands for some commands are required with the command and some are optional.
\path\	the \path\ is included in a command form only when a path is required, and includes all levels of directories in the path, and optionally a drive designation. For example, \ACCT\PAYROLL\HAROLD is the path to access the directory HAROLD. The path could also be C:\ACCT\PAYROLL\HAROLD to indicate a drive other than the current drive. A path may or may not require the ending backslash, depending on the individual command form.
wildcard	wildcards may be used to represent that a position in the filename may contain any characters. Some commands allow the use of wildcards within filenames and some commands do not. A question mark (?) indicates that a single position may contain any character as long as all other positions match. An asterisk (*) indicates that the current position and all following positions may contain any characters.



- { } operands enclosed in braces { } are optional. If an operand is not enclosed in braces, the operand is required.
- | the vertical bar indicates that there is a choice of operands for a position, but only one of those operands may be entered. For example, on | off indicates that you may enter either on or off as the operand for that position in the command form.
- ... means that you may repeat an operand or series of operands in the command statement.
- . indicates that there are lines that follow containing the same type of information, or that there are lines omitted in an example that are not necessary to the example.

## Input/Output Redirection

MOS lets you redirect the input or output of commands. For example, the .TYPE command normally sends output to the video screen. There may be times when you want to redirect the output of the .TYPE command to a printer. This will give you a hard copy of the file contents.

There are several conventions for redirection of input and output that you need to understand. The term "standard input" usually refers to input from the keyboard, but can also mean input from a file or device. The video screen is the normal destination for "standard output".

There are four ways to accomplish redirection with MOS. They are:

- | This is the pipe symbol. It instructs MOS to redirect the standard output of the command that precedes it, and use that output as the standard input of the command that follows it.
- < The less than symbol is preceded by a command, and tells MOS to use the file or device that follows the symbol as the standard input for that command.

## General Commands

- > The greater than symbol is preceded by a command, and tells MOS to use the file or device that follows the symbol as the standard output for that command. If the file specified for standard output does not exist, MOS will create it. If the file does exist, standard output will overwrite any information the file contains.
- >> Two greater than symbols redirect standard output and append it to the information in an existing file. If the file specified for standard output does not exist, MOS will create it.

There are many ways you can use redirection. For example, the standard output of the .DIR command is a directory listing on the video screen. The following command will redirect the output of the directory listing to a printer.

```
.DIR > LPT1:
```

The following example tells MOS to use the standard output of the directory command as the standard input for the sort command.

```
.DIR ! .MSORT
```

The result is a display of the sorted directory to the video screen. You may further redirect the output to the .MORE command so that the standard output displays 24 lines at a time. The commands to accomplish this are:

```
.DIR ! .MSORT ! .MORE
```

Redirection lets you predefine the input for a specific command. In the next example, redirection is used for input and output.

```
.MSORT < DATAFILE > NEWDATA
```

The .MSORT command is using the file named DATAFILE for standard input. MSORT will sort the data in this file in ascending alphabetic order, beginning at column 1 to the end of the record. The resulting data is sent to the file named NEWDATA as the standard output of .MSORT. Other examples of I/O redirection are given with the individual MOS command explanations.



## .ADDDEV

.ADDDEV lets you add a device driver at any time. You may make a driver active on the whole system or only in a specific task.

**Type:**      Extrinsic

**Form:**      .ADDDEV *n|t,device*

### Operands:

- n*            enter a number representing the amount of memory in kilobytes that is required by this device driver to run with MOS. (1 for 1024 bytes, 2 for 2048 bytes, etc.)
- t*            use the *t* operand instead of *n* if you want the device to be active only in the task in which the .ADDDEV command is being entered.
- device*      enter the device driver name and extension, with any buffer size or other operands, as you would normally enter it in the CONFIG.SYS file. Include the drive and directory if not the current defaults. (Do not include the DEVICE= command statement with the .ADDDEV command.)

### Explanation:

You may add a device at any time by invoking the .ADDDEV command at the system prompt. For example, if you want to add a pipe device, you could enter the following command:

```
.ADDDEV 2,$PIPE.SYS PIPE3,2048
```

MOS will check to be sure there is sufficient memory to run the PIPE3 device before adding it. If not, MOS will display a message.

You may use the *t* operand to invoke a task-specific device driver, thus isolating other tasks from the effects of the driver. For example, the command form:

```
.ADDDEV t,device
```

## General Commands

---

would add the device driver only in the task in which the command was entered. Note that a separate copy of the driver must be installed in each task that requires access to it.



## .ALIAS

.ALIAS lets you substitute a drive letter for a directory name. You may find ALIAS useful as a shortcut to access directories that are removed from the root directory by several levels. You may assign floppy disk drive letters A: and B:, but not an existing hard disk drive letter such as C:.

**Type:** Extrinsic.

**Form:** .ALIAS *d:* {*d:*}/{*dirname*}[/*d*]

### Operands:

*d:* enter the drive letter you want to substitute for a directory name. This must be a drive letter that is not used to access a physical or logical hard disk on your computer.

*d:* enter the letter of the drive where the directory you are substituting resides. If you do not enter a drive letter, MOS uses the current default drive in the substitution.

*dirname* enter the name of the directory you are substituting. You may enter the complete name of the directory from the root, or only a portion of the directory name. If you do not enter a directory name, MOS uses the current directory as the substitution directory.

/*d* delete aliases. If an existing alias drive is specified with the command, only that alias drive will be deleted. If no drive is specified MOS will ask if you want to delete all aliases.

### Explanation:

You may use .ALIAS to shortcut the access to a directory which is several layers removed from the root directory. For example, you may have a directory similar to the following that is four levels from the root:

\wp\mywp\myfiles\mywork

## General Commands

You may use the .ALIAS command to assign access to the MYWORK directory with a drive letter. If drive letter E is not used to access a drive on the computer, you could enter the .ALIAS command as follows:

```
.ALIAS e: c:\wp\mywp\myfiles\mywork
```

If drive C is your current drive and MYWORK is the current directory, you could enter the following command and MOS will use the current drive and directory for the substitution:

```
.ALIAS e:
```

When the substitution is complete, MOS will display a message similar to the following:

```
E: is C:\wp\mywp\myfiles\mywork
```

If you enter a directory with or without a drive that MOS cannot trace back to the root, MOS will display a message similar to the following:

```
Cannot find directory c:\letters
```

If you attempt to assign an alias to an existing drive (other than A: or B:), MOS will display a message similar to the following:

```
C: may not be used as an alias
```

If your current drive is C, and your current directory is \wp\mywp, you could enter the following command to assign an alias to the next level of a directory:

```
.ALIAS F: myfiles
```



When the substitution is complete, MOS will display the complete substitution as follows:

```
F: is C:\wp\mywp\myfiles
```

To see if an .ALIAS is active on your computer, you may type .ALIAS with no operands and press ENTER. If no substitution is active, MOS displays the following message:

```
No aliases defined
```

If one or more substitutions are active, MOS will display a message similar to the following:

```
E: is C:\wp\mywp\myfiles\mywork  
F: is C:\sales\mkt\adver\reps
```

You may delete previously assigned aliases by using the /d operand. If an existing alias drive is specified along with the /d operand, only that alias drive will be deleted. For example, the following command will delete the assigned alias for drive letter E:

```
.ALIAS E: /d
```

You may delete all existing alias assignments by entering the command without specifying any drive, as follows:

```
.ALIAS /d
```

MOS will then prompt with:

```
Delete all aliases? (Y/N)
```

Responding with Y will cause all existing aliases to be deleted.

# General Commands

## .BREAK

.BREAK instructs MOS to check for CTRL - BRK key sequences issued from the keyboard during processing. The control break key sequence instructs MOS to halt processing.

**Type:** Intrinsic.

**Form:** .BREAK on|off

### **Operands:**

**on** turns on control break checking. MOS will check for the CTRL - BRK key sequence.

**off** turns off control break checking. This is the default state of .BREAK each time MOS is booted.

### **Explanation:**

You may press CTRL - BRK, or CTRL - C to halt processing of a program or a command. When .BREAK is set to OFF only a few program calls are sensitive to .BREAK, i.e. control break checking is only done when output is written to the screen or printer, or when input is read from the keyboard. When .BREAK is ON, most calls are sensitive to .BREAK, i.e. most program's requests of MOS to carry out functions can be interrupted by pressing CTRL - BRK or CTRL - C.

You may turn .BREAK on by typing:

.BREAK ON

Typing .BREAK with no parameters displays the current setting for .BREAK. For example:

.BREAK set to on

If .BREAK is on, you have a greater opportunity to break out of a program. However, you should note that when .BREAK is on, your system may run a little slower because every operating system call will check for .BREAK.



## .CD

.CD sets the directory that will be used as your default directory.

**Type:** Intrinsic.

**Form:** .CD [*path*]

### Operands:

[*path*] enter the name of the directory to which you want to change. You may enter two periods (..) to change to the parent of the current directory.

### Explanation:

You may create several levels of directories in MOS. To change from one directory to another, you may enter the complete directory name while at the root directory as follows:

```
.CD MYTEXT\MYFILES\LETTERS
```

If you are not at the root, and you want to change to the LETTERS directory, you must enter a preceding slash with the first level of directory in the path. For example:

```
.CD \MYTEXT\MYFILES\LETTERS
```

If you were currently in the MYFILES directory, you could access the LETTERS directory without any preceding directory levels or the backslash. For example:

```
.CD LETTERS
```

To move from the LETTERS directory to the MYFILES directory, you may enter two periods to indicate that you want to move to the parent of the current directory. For example:

```
.CD ..
```

## General Commands

You may move to the root directory by entering only the backslash, regardless of the level of directory you are in. For example, if you are currently in the LETTERS directory, the following command returns you to the root directory on your current drive:

```
.CD \
```



## .CLS

.CLS clears the current display on the video screen, and positions the system prompt at the top of the screen.

**Type:** Intrinsic.

**Form:** .CLS

### **Explanation:**

The .CLS command clears any display currently on the video screen, creating a clean screen. The MOS system prompt will appear on the first line at the top of the screen.

You may want to include the .CLS command in your AUTOEXEC.BAT file so you always start with a clean screen when you boot your system. If you call up a menu in your AUTOEXEC.BAT, it is a good idea to place a .CLS command prior to the menu call. This causes the screen to clear so that only the menu displays when you boot your computer.

# General Commands

## .COMMAND

Invoking .COMMAND at the system prompt lets you move out of the current command processor leaving it dormant, and move to a new layer of the command processor. The new layer carries a copy of the environment with it. The environment is a reference area in memory for frequently accessed strings, such as the prompt and path. You may find .COMMAND useful to test strings of data that you want to set into the environment. You may then exit from this layer of the command processor by typing .EXIT at the prompt and pressing ENTER. You will then return to the previous layer where the command processor's environment has not changed.

**Type:** Intrinsic.

**Form:** .COMMAND {d:}/{path} {/c string|/p}

### **Operands:**

*d:* enter the letter of the drive that contains the new command processor, or omit it to use the default drive.

*|path* enter the directory that contains the new command processor you want to invoke, or omit to use the current directory.

*/c string* lets you pass a string that follows the /c operand and execute that string in the new layer of the command processor and then return to the current command processor. The string to execute may be a command or a batch file. If a batch file, do not enter the .BAT extension as a part of the string.

*/p* makes the new layer of the command processor permanent in memory. You cannot exit back to the current layer when you use the /p operand. You must reboot your computer to remove this new permanent layer. If you want to run a startup batch file in the new layer, you must first enter a .SET STARTBAT=filename command. The "filename" could be AUTOEXEC or any batch file of your choice. This batch file must be located at the root directory of your boot drive.



### **Explanation:**

You may type .COMMAND with no operands at the system prompt to move to a new layer of the command processor. The COMSPEC= statement in the current environment identifies the new command processor that is loaded. Invoking a duplicate command processor, whether the current one or a new user-created processor, reduces available memory. If you invoke .COMMAND with the /p operand, the new layer becomes permanent.

# General Commands

## .COMPFILER

.COMPFILER compares the contents of two files on a byte-to-byte basis. Any discrepancies in the files are displayed on the video screen.

**Type:** Extrinsic.

**Form:** .COMPFILER *filename filename*

### **Operands:**

*filename* enter the name of the first file you want to compare, including the drive letter and/or path if not the current defaults. If wildcards are used .COMPFILER will operate on more than one set of files.

*filename* enter the name of the second file you want to compare, including the drive letter and/or path if not the current defaults. If wildcards are used all or part of the first file's name will be used to form the second file's name.

### **Explanation:**

.COMPFILER lets you compare the contents of one file to that of another file one byte at a time. You may want to compare the contents of a file you copied to be sure the copy was successful. For example, if you copied a file from a floppy diskette to your hard disk you could enter the following command to compare the files:

```
.COMPFILER A:myfile.txt C:myfile.txt
```

The following message displays as the comparison is running:

```
PC-MOS File Comparison Utility v#.##  
Copyright 1988 The Software Link, Incorporated  
All rights reserved worldwide.
```

```
-----  
FILE1 : A:myfile.txt  
FILE2 : C:myfile.txt
```



If any discrepancies are found, MOS will display a message similar to the following:

OFFSET	FILE1	FILE2
-----	-----	-----
0000000H	61 'a'	EOF N/A

The position of the discrepancy is expressed as "0000000H", meaning the hexadecimal offset position in the first file. If there are no discrepancies, MOS displays the following message when the comparison is complete:

No differences encountered in the two files

If wildcards are used in the first operand .COMPFILER will operate on more than one set of files. For example:

.COMPFILER \UTILS\\*.BAT TEST.BAT

will compare all batch files found in \UTILS with the file TEST.BAT in the current directory.

If wildcards are used in the second operand all or part of the first file's name will be used to form the second file's name. For example:

.COMPFILER JEFF.LTR \WP\\*.\*

will compare JEFF.LTR in the current directory with \WP\JEFF.LTR.

An example in which only part of the name is carried over may be:

.COMPFILER ABC ??DE

## General Commands

This will compare ABC with ABDE in the current directory. The ?'s causes the corresponding characters of the first operand to be carried over to the second operand.

Wild card characters may be used in both operands at the same time, for example:

```
.COMFFILE \UTILS\*.BAT *.*
```

will compare each batch file in \UTILS with any corresponding batch file of the same name in the current directory.



## .COPY

The .COPY command lets you create a copy of a file or a group of files. You may copy a file and give the copy a different name, or copy the file and give it the same name on a different drive or in a different directory. You may copy several files into one file, or you may copy a file from or to a device. .COPY may also be used to assign or change the security class of a file.

Some rules apply to all uses of .COPY. Wildcards may be used in both the original and new file names. .COPY preserves the user ID and dates and times of creation and last update in the new file. The security class is preserved unless you include the "/c" operand in the command.

Two optional operands may be entered with the original and the new file to accommodate both ASCII and binary files. The /a operand is used with ASCII files. If /a is used with the original file, MOS copies to the end of file character, not the physical end of the file. If /a is used with the new file, MOS places an end of file character at the end of the copy. MOS uses a CTRL Z as an end of file character. The /b operand is used with binary files. If /b is used with the original file, MOS ignores any end of file characters. If /b is used with the new file, MOS will not place an end of file character on the new file.

**Type:** Intrinsic.

**Form:** .COPY *filename* [/a|/b] {*d:\path\}/*filename*} [/a|/b]  
[/v]{/c|/cn|/c-}*

### Operands:

*filename* enter the name of the file to be copied. You may include a drive and path if the original file is not at the current drive or directory.

/a copies to the end of file character (CTRL Z), on the original.

/b copies to the physical end of file on the original.

## General Commands

- d:[path] enter the drive and/or path to contain the new file, if you want them to be different from the current drive and/or directory.
- filename enter the name you want to give the copy, or omit to use the same name as the original file.
- /a on the new file, adds an end of file character (CTRL Z).
- /b on the new file, omits placing an end of file character (CTRL Z).
- /v verifies each aspect of the copy operation to be sure the information may be read from the copy.
- /cn enter /cn, where n is a security class of A to Z, to assign a specific security class to the file.
- /c- enter /c- to remove an existing security class, and assign the file a blank class.
- /c enter /c to change the class of the file to your current default output class.

### **Explanation:**

You may copy a file to a different drive and directory without specifying a second filename. In this case, MOS will assign the original filename to the new file. For example:

```
.COPY memo.doc d:\work\
```

If you specify a new directory for the copy, that directory must already exist. If it does not, MOS treats the directory name as the name of the new file.

You may copy a file at your current drive and directory to another drive and give the new file the same name with the following command:

```
.COPY memo.doc a:
```



You may copy a file from a different drive and directory to your current drive and directory and give the new file the same name as the original file. The following command illustrates how this may be done, using C:\ as the current drive and directory.

```
.COPY a:letter.doc
```

### Device Input/Output

.COPY may also be used to copy input from a device or output to a device. However, do not copy input or output from serial ports because there is no control over the handshaking transmission of data. Devices are explained in the Configuration chapter of this manual. The command form to copy input from a device is:

```
.COPY DEV filename
```

DEV is the name of the device from which you want copy input. The device name may be CON for the main console, PRN for a printer, or a user defined name for a MOS pipe that allows you to copy characters from one partition to another. For example, you may use the following .COPY command to create a file from keyboard input.

```
.COPY CON testfile.doc
```

When you press ENTER, the TESTFILE.DOC file is created. Anything you then type at the console is placed in the file. When the file is complete, press CTRL - Z to create an end of file character. MOS will then display:

Files copied: 1

You may also copy to a device with the following form of .COPY:

```
.COPY filename DEV
```

In this example, DEV is treated the same as if it were a file. For example, you may enter:

```
.COPY testfile.doc PRN
```

## General Commands

This will send the TEXTFILE.DOC file to the printer. You may also send direct keyboard input straight to the printer with the following command:

```
.COPY CON PRN
```

### File Concatenation

There is a special form of .COPY that lets you copy and combine a group of files. This is called concatenation. The command form for .COPY is:

```
.COPY file1.doc + file2.doc + file3.doc bigfile.doc
```

Note the use of the + symbol between the files that are to be combined. The new file, BIGFILE.DOC contains the contents of the concatenated files in the order you enter them. If no name is specified for the copy, MOS defaults to the first file in the command and adds the other files to it in the order you enter them. For example:

```
.COPY file1.doc + file2.doc + file3.doc
```

This command copies the contents of FILE2.DOC and FILE3.DOC, into FILE1.DOC. The contents of FILE2.DOC is added at the end of FILE1.DOC, and then the contents of FILE3.DOC is added.

When the copy in the previous example is complete, MOS will display a message indicating the number of files created as a result of the copy. If you concatenated several files into one file, the message will reflect that one file was created. If a new file is being created by a user who is signed on to security, the new file will carry the default output class of the user who invoked the .COPY command.

You may concatenate with global file names. For example:

```
.COPY *.asm + *.lst *.tot
```

The order of the first item in the series of files controls what will happen during the concatenation. In this example, MOS will search for the first file with an .ASM extension. If A.ASM is found, then MOS looks for a file named A.LST. If it is found, A.ASM and A.LST are written to a file MOS names A.TOT. If MOS does not find a file named A.LST, then only A.ASM is written to A.TOT.



In either case, the new file named A.TOT is closed and MOS begins looking for the next file with an extension of .ASM. This continues until MOS cannot find any more files with an .ASM extension.

Be careful of how you concatenate with global file names. If you enter the following command, MOS will copy the content of all files with a .DOC extension to a new file named X.BIG.

```
.COPY *.doc x.big
```

However, the following command could cause problems:

```
.COPY *.doc x.doc
```

If a file named X.DOC does not currently exist, MOS will concatenate all files with a .DOC extension into the new file. If X.DOC does exist, and MOS finds it as the first file, the concatenation will take place. If X.DOC is not the first file found, when MOS does find X.DOC it appears as an attempt to write a file on top of itself. MOS will concatenate the file, but will display the following warning message to let you know that something strange occurred during processing:

New file content lost before copy complete.

You may control this situation by using the following command to instruct MOS to use the file named X.DOC as the new file, and to concatenate other files with a .DOC extension into the new file.

```
.COPY x.doc + *.doc
```

### Changing File Dates and Times

The .COPY command has an operand (+,,) that lets you change the creation and updated date and time of a text file to the current date and time. For example, you may change the last updated and creation date and time of a file named CONTRACT.SAV by entering the following command:

```
.COPY contract.sav +,,
```

## General Commands

You may include this operand with the .COPY command to assign the current date and time to only the new copy of a file. For example:

```
.COPY x.doc +,, b:
```

### Copying ASCII and Binary Files

If the file you want to change is not a text file, you must enter the /b operand. For example:

```
.COPY .help.com /b +,,
```

If you copy one file to another file, the /b is the default to copy the entire content of a file. MOS will ignore any end of file characters (CTRL Z) in the file. For example, the following command uses /b as the default.

```
.COPY *.* A:
```

If you concatenate files, the default is /a because MOS assumes the files to concatenate are text files. For example, the default for the following command is /a:

```
.COPY text1.doc + text2.doc newtext.doc
```

Note that you may enter /a or /b before the names of any files in the copy so that the option applies to all the files. For example, the following command uses /a to ensure that all files with an .ASM extension are copied only up to an end of file character (CTRL Z):

```
.COPY /a *.asm new.asm
```

When a device (CON, PRN, etc.) is used in a .COPY command, MOS uses /a as the default. You may include /b for a target device, but not for the source device. For example:

```
.COPY /b CON x.doc
```

Because the /b operand is included with a device as the source, MOS will display the following error message:



Binary read from a device not allowed.

## Copying Secured Files

Normally, you may copy a file or group of files within the current drive and directory only if you give the copy a new name. However, you may copy a file with the same name within the current drive and directory if you use the /c operand to assign a different security class to the file.

If you enter only an original filename and the /c operand, MOS rewrites the file back to disk using the same name, but the file is converted to the security class specified after the /c. For example, MOS allows the following command to assign security class A to the file:

```
.COPY contract.doc /ca
```

MOS would not allow this command without the /c, because it would be an attempt to write a new file on top of an original file. Note that you may only copy a secured file if you have an access level that allows you to access the class assigned to the file.

If you are currently signed on to security, you may assign the file your current default class by entering the /c operand without the letter of a security class. For example, if your current user default output class is B, the following command will secure the file named FORMS.DOC with security class B:

```
.COPY forms.doc /c
```

You may change a secured file to an unsecured file by copying the file with a /c- operand. This operand assigns the file a blank security class to indicate that the file is not secured. For example:

```
.COPY memo.doc /c-
```

The file named MEMO.DOC is written back to the disk in an unsecured state.

## General Commands

### .DATE

.DATE sets the MOS system date in the format defined by the COUNTRY command statement in your CONFIG.SYS file. This is the internal date that is known to MOS.

**Type:** Intrinsic.

**Form:** .DATE {mm-dd-yy}

#### **Operands:**

*mm-dd-yy* enter the date in the form used on your computer. The default form for MOS is mm (month) dd (day) and yy (year).

#### **Explanation:**

If your computer does not have a perpetual calendar, you may want to include .DATE without an operand in your AUTOEXEC.BAT file. Each time you boot your system, MOS will prompt you to set the date.

You may set the current date by entering the .DATE command at the system prompt, followed by the correct date. For example:

```
.DATE 10-21-87
```

You may type .DATE at the system prompt without any operands to display the current date that is known to the system, and to optionally enter a new date. The date display appears similar to the following:

```
Wed 12-31-86 is the current date  
New date is (mm-dd-yy):
```

You may enter a new current date, or press ENTER without making an entry to leave the date unchanged.



## .DIR

.DIR displays the contents of a directory. There are several display options that let you list specific directories, sort by file name, sort by file extension, sort by creation date and time, or sort by size. Only one sort option at a time can be active. You may select to include hidden and system files in the display. You may choose additional options for a condensed format, or to pause between pages.

**Type:** Intrinsic.

**Form:** .DIR {[path] [/d]/[sn]/[se]/[sd]/[ss]/[a]/[w]/[p]}

### Operands:

[path] enter the complete name of the directory to display if not the current directory, and/or current drive.

/d includes only directories in the display.

/sn sorts the display alphabetically by file name.

/se sorts the display alphabetically by file extension.

/sd sorts the display by date and time last updated, with the oldest date and time appearing first.

/ss sorts the display by file size, with the smallest file appearing first.

/a includes hidden and system files in the list.

/w displays only the names of the files in the directory in an 80-column wide format.

/p displays the directory 24 lines (one screen) at a time.

## General Commands

### **Explanation:**

You may display the contents of the current directory on the current drive by entering the command:

.DIR

You may display the contents of a different directory by including the directory name in the .DIR command. If the directory is on a different drive, include the drive letter with the directory name. For example:

.DIR D:\WORK3

To display only the directories within the selected directory, enter the /d operand in the command.

You may include only a specific file in the directory by entering the name of the file to display. You may also use the wildcard characters to display specific groups of files. For example, the following command will display only the files in the current directory that have an extension of .DOC.

.DIR \*.DOC

There are four sort options you may select for displaying the directory. Only one sort option may be active at a time. The following is an example of a directory displayed by file size.

filename.ext	size	----updated----	attrb	cls	user	----created----
COMMAND.COM	6	12-30-86 12:00	* S B	MARK	12-30-85	12:00
AUTOEXEC.BAT	179	02-20-87 09:13	* S E	SYS	1-19-87	09:13
CONFIG.SYS	201	02-25-87 14:19	*		2-25-87	14:19
.						
KIM.BAT	436	01-30-87 10:44	*		1-30-87	10:44
DISKID MOS	55,329 Bytes	31 Files	11,134,976 Bytes	remaining		



The first column displays the file name and extension. The file size in bytes is then shown. The date and time a file was last updated appears next. For a directory entry, the word directory always appears in this column.

Following the time are four columns that display the attribute settings of the file. The first column displays an asterisk if the archive attribute is set. The second column displays an R if the read-only attribute is set, and the fourth column displays an S for any file that has a security class. If the /a operand is included in the command, the third column displays an H if the file is a hidden file.

If security is used, the next two columns display the security class assigned to the file, and the ID of the user who created the file. The last two columns display the date and time the file was created.

The last line of the display shows the disk ID if one has been assigned. The total number of bytes used by the files in the display appears next, along with the total number of files in the display and the remaining bytes available on the drive.

## General Commands

### .DIRMAP

.DIRMAP scans a disk and displays the directories on that disk along with the current number of files and total bytes used in each directory. The directories display in the order they are found on the disk.

**Type:** Extrinsic.

**Form:** .DIRMAP {path}{/f}{/a}

#### **Operands:**

**|path** you may enter the letter of the drive and/or the name of a specific directory to display, or omit this operand to begin the display at the root directory on the current drive.

**/f** enter the /f option to include the names of the files in each directory in the display. Omit this option if you don't want to display file names.

**/a** enter the /a option to include a list of only those files in the directories which have the archive attribute set.

#### **Explanation:**

Entering the .DIRMAP command without any operands displays the directories on the default drive, beginning at the root directory. Subsequent directories are listed in the order they are found on the default drive. The following is an example of the .DIRMAP display:

Directory	Files	Bytes
-----	-----	-----
C:\	93	1441916
C:\MOS	12	56337
C:\WORK	14	1883
C:\WORK\JIM	49	213649



You may use the path option to display all levels of a specific directory. For example, the following command will display the WORK directory as well as the WORK\JIM directory on the default drive:

```
.DIRMAP \WORK
```

To include the name of each file in a specific directory on a specific drive, the command could be:

```
.DIRMAP C:\WORK\JIM /f
```

The /a option is useful in determining what files on your disk have been changed since the last time you made a backup copy with the .EXPORT command. The following command:

```
.DIRMAP C:\WORK /a
```

would list all files in the WORK directory of the C drive that have the archive attribute set. (Changing a file sets the archive attribute, while running .EXPORT clears the archive attribute.)

If you have a lot of directories or files to display, you may want to use the .MORE command to display only 24 lines at a time. An example of the .DIRMAP command used with .MORE could be:

```
.DIRMAP \WORK\JIM /f | .MORE
```

# General Commands

## **.DISKCOPY**

.DISKCOPY copies the entire contents of one diskette onto another diskette. If the original diskette contains the boot sector, it is also copied onto the destination diskette. .DISKCOPY works with either a one- or two-diskette drive computer.

**Type:** Extrinsic.

**Form:** .DISKCOPY *d:* *d:*

### **Operands:**

*d:* enter the letter of the drive that contains the original diskette you want to copy.

*d:* enter the letter of the drive that contains the destination diskette on which the copy is to be placed.

### **Explanation:**

.DISKCOPY produces a copy of the original diskette. The files and directories on the copy appear in the same order as they do on the original. If you have two diskette drives, you may copy the contents of a diskette in one drive to a diskette in the other drive by entering the following command:

.DISKCOPY *a:* *b:*

A message appears prompting you to place the original diskette to copy in drive A, and to place the diskette to contain the copy in drive B. You could copy a diskette in drive B to a diskette in drive A by entering the following command:

.DISKCOPY *b:* *a:*

If you only have one diskette drive, you will need to swap the original and destination diskettes in that one drive until the entire copy is complete. The command to copy a diskette when you have only one diskette drive is:

.DISKCOPY *a:* *a:*



MOS will prompt you to place the original diskette in drive A. When you press ENTER, MOS copies as much of the original diskette as possible into memory, and then prompts you to remove the original and place the destination diskette in the drive. When you press ENTER this time, MOS copies the original content in memory onto the destination diskette.

If the entire contents of the original diskette could not be copied into memory in one pass, a prompt appears instructing you to return the original diskette to the drive. You will be prompted to continue swapping diskettes until the content of the original diskette is completely copied onto the destination diskette.

If you are copying a 360K diskette on a single drive computer, the following prompt appears if the diskette was copied in one pass:

Do you wish to copy this disk again? Y/N

You may enter Y to make another copy of this diskette. If you enter N, or for any other type the media, the following prompt always appears:

Do you wish to make another copy? Y/N

If you want to make a copy of another diskette, enter Y. If not, enter N and you return to the system prompt.

## General Commands

### .DISKID

.DISKID lets you assign an 11-character name to a diskette or hard disk. The .DISKID is displayed by the .DIR and .VERIFY commands.

**Type:** Extrinsic.

**Form:** .DISKID {d:}{name}

**Operand:**

*d:* enter the letter of the drive that contains the disk you want to name, if not on the current drive.

*name* enter up to 11 characters for the .DISKID name.

#### **Explanation:**

You may enter the .DISKID command and assign a name as part of the command form. MOS assigns the name to the disk as the .DISKID and then returns to the system prompt.

If you enter the .DISKID command without a name, MOS displays whether or not the disk already contains a name. For example, if the disk does not contain a name, the following message appears:

Disk ID for drive A is not set.  
Enter Disk ID (11 characters max or ENTER for none):

If the disk contains a name, the following message appears:

Disk ID for drive A is DATA FILES  
Enter Disk ID (11 characters max or ENTER for none):



---

You may type a different name, or press **ENTER** and MOS displays a message asking if you want to delete the current name. You must respond with a Y (yes) or an N (no) to continue.

**NOTE:** Never use **.DISKID**, or any other method, to change the Disk ID of any **.EXPORT** disk. Doing so will render **.IMPORT** inoperable and you will not be able to restore the data from that **.EXPORT** set.

## General Commands

### .ENVSIZE

The .ENVSIZE command lets you specify the minimum size of your environment. You may use this command to reserve space for your environment prior to loading any resident programs.

**Type:** Intrinsic.

**Form:** .ENVSIZE *nnnn*

#### **Operand:**

*nnnn* enter the number of bytes you want to reserve for the environment. If you do not reserve the environment size, MOS uses 128 bytes.

#### **Explanation:**

You must specify a size operand as part of the .ENVSIZE statement. The normal system default for environment space is 128 bytes. Because the .ENVSIZE statement reserves the memory for your environment, it is ideally included in your AUTOEXEC.BAT. For example:

```
.ENVSIZE 256
```

.ENVSIZE must be specified prior to loading any resident programs, such as .SPOOL, the resident program for the MOS Print Spooler. If you do not specify the .ENVSIZE, you may find the space you could reserve for your environment diminished.

You may invoke the .ENVSIZE command at the system prompt to replace a previously invoked .ENVSIZE. The new .ENVSIZE command will overwrite the space allocation you made with a .ENVSIZE statement in your AUTOEXEC.BAT file.



## .ERASE

.ERASE deletes a file or group of files from a disk.

**Type:** Intrinsic.

**Form:** .ERASE {path} {filename} {/v | /y}

### Operands:

*filename* the name of the file to be erased. Include the drive and directory if not the current defaults. Wildcards are permitted in the filename.

/v displays a verification prompt that lets you select whether or not to erase each file.

/y enters a "y" response to the "Erase all files" prompt when you delete the entire contents of a directory.

### Explanation:

You may use the /v option with .ERASE to display a verification prompt and selectively delete files. For example, the following command will find files with an extension of .BAK, and displays the verification prompt with each file found:

```
.ERASE *.BAK /v
```

MOS will display the following verification prompt:

ERASE MEMO.BAK (Y/N)?

You may enter a Y (yes) to erase this file, or enter an N (no) to skip this file. MOS then displays the prompt for the next file it finds with an extension of .BAK.

## General Commands

The following command deletes all files in the current directory:

```
.ERASE *.*
```

MOS will display the following prompt when it finds this command:

ERASE ALL FILES (Y/N)?

You may enter Y (yes) to erase all files, or enter N (no) to return to the system prompt.

You may include the /y operand with the command to erase all files without displaying the verification prompt.

NOTE: You may also use the command .DEL to delete files.



## .EXCEPT

.EXCEPT DO lets you exclude a file or a group of files from a command. You may use .EXCEPT DO with a command or with a batch file. You may also use .EXCEPT DO with any programs or applications that use filename operands rather than operating directly on the disk. Be careful using .EXCEPT DO because it offers a global functionality with destructive commands. You may want to preview what you'll be deleting or erasing prior to invoking the command. .EXCEPT CANCEL may be used to cancel the .EXCEPT command when it is used in a batch file, and free the file or group of files for use with other commands.

**Type:** Intrinsic.

**Form:** .EXCEPT (*filename ...*) DO *command | batfile*

.EXCEPT CANCEL

**CANCEL** If an .EXCEPT DO statement is used within a batch file, you may cancel it with an .EXCEPT CANCEL statement. This will free the files in the .EXCEPT DO group for use with other commands in the batch file.

### Operands:

*filename* enter the name of the file or group of files without a drive letter or directory. All filenames must be enclosed in parentheses. Wildcard characters are permitted in the filename. You must enter a comma or space as a delimiter between filenames. You may enter as many filenames as space permits on the command line. Do not enter a drive or a path with any file names you enter in the group.

*command* enter the command you want to perform on the files. You may enter any operands associated with the command, including the drive and path for the file or group of files.

*batfile* enter the name of the batch file to perform on the file or group of files, but do not include the .BAT extension.

## General Commands

### **Explanation:**

You may use .EXCEPT DO to exclude a group of files from a global command statement. For example:

```
.EXCEPT (MYWORK.DOC TESTWORK.DOC) DO .COPY *.* A:
```

In this example, all files except MYWORK.DOC and TESTWORK.DOC will be copied onto the diskette in drive A. You may also use .EXCEPT to limit the access to files. For example:

```
.EXCEPT (MYWORK.DOC) DO WORDPROC
```

This command statement will invoke a word processor, and let you edit any file except a file named MYWORK.DOC.

You may also use .EXCEPT DO to invoke a new command processor and limit the files which may be accessed when the new command processor is active. For example:

```
.EXCEPT (*.DOC, *.TXT) DO COMMAND
```

This statement excludes files with .DOC and .TXT extensions from the new command processor.

**NOTE:** Be careful when you limit access to files. You may create files excluded from an application or a new command processor that may already exist. Although it appears as if you are creating a new file, the existing file may be overwritten without your knowledge.

You may also use .EXCEPT DO to exclude specific files from a batch file execution. For example:

```
.EXCEPT (MYWORK.DOC TESTWORK.DOC) DO CLEAN
```

All files except MYWORK and TESTWORK would be included when CLEAN.BAT is invoked. The batch file named CLEAN.BAT contains:

```
.CD\WORK.DIR  
.ERASE *.* /y  
.CD\ASM.dir  
.ERASE *.* /y
```



NOTE: Be careful if you invoke a globally destructive batch file such as this one. You may want to preview the files you'll be deleting. To preview the files, you could write CLEAN.BAT as follows:

```
.CD\WORK.DIR  
.DIR /w/p  
.ECHO Erase these files in \WORK.DIR?  
.KEY %%a IN (y,n) DO .GOTO p1%%a  
:p1y  
.ERASE *.* <c:/y  
:p1n  
.  
.
```

The use of .KEY requires you to respond to the prompt. The key you press as your response, either Y or N, determines the action that will be taken. See .KEY for more information.

There may be times when you want to remove the .EXCEPT limitation during the execution of a batch file. For example:

```
.EXCEPT (*.doc *.txt) DO FLIST
```

All files except those with an extension of .DOC and .TXT will be included when FLIST.BAT is invoked. The batch file named FLIST.BAT includes:

```
.DIR  
.TEXT      This displays a limited set of files based on the group  
           defined in the .EXCEPT DO statement.  
.ENDTEXT  
.EXCEPT CANCEL  
.DIR  
.TEXT      This is a complete list of files because .EXCEPT  
           CANCEL permanently removes the limitation of the  
           .EXCEPT DO statement at the time this file was invoked.  
.ENDTEXT
```

## General Commands

NOTE: The .EXCEPT CANCEL statement permanently removes the .EXCEPT DO limitation. For example, you may invoke a batch file with the .EXCEPT DO limitation, and that batch file may call a nested batch file. If the nested batch file contains a .EXCEPT CANCEL statement, the limitation is removed for all further processing within the nested batch file. This carries back to any processing within the batch file from which the nested file was called.

You may couple the .EXCEPT statement with a .ONLY statement to further limit the effect of a command. For example:

```
.ONLY (*.doc) DO .EXCEPT (special.doc) DO .EXPORT *.* A:
```

In this example, SPECIAL.DOC is excluded from the .EXPORT, but all other .DOC files are included. In a situation like this, the order of the .ONLY and .EXCEPT command does not matter. The same effect could have been achieved with:

```
.EXCEPT (special.doc) DO .ONLY (*.doc) DO .EXPORT *.* A:
```

If the groups defined in the .EXCEPT and .ONLY statements conflict, no files will be found. For example:

```
.ONLY (x.doc) DO .EXCEPT (*.doc) DO .EXPORT *.* A:
```



## .EXPORT

.EXPORT creates a compressed copy of one or more files for backup purposes. .EXPORT is similar to the .COPY command, but data is compressed and a header is placed at the beginning of the exported file. We recommend that you use .EXPORT for backing up files. Unlike .COPY, .EXPORT will spread files across multiple diskettes. You may reinstate exported files only with the .IMPORT command.

Type: Extrinsic.

Form: .EXPORT {d:[path]} d: [/s] {D:mm-dd-yy} [/m] [/q] [/a] [/?]

### Operands:

d:[path] enter the name of the file or directory to export. Include the drive letter and directory name if not the current defaults. Wildcards are permitted in the name of the file. Defaults to the current drive and directory.

d: enter the letter of the drive that contains the target disk on which the exported files are to be placed. Only a drive letter may be specified, not a subdirectory.

/s begins the export for all files in the specified or default directory, and continues to export all files in any following levels of directories.

/D:mm-dd-yy export only those files with file dates after the specified date. If not entered, defaults to exporting all files regardless of file date.

/m exports only files which have been modified since your last .EXPORT operation, i.e. those files with the archive attribute set. The archive attribute indicates these files have been modified since the last .EXPORT operation.

/q enter alone for Query Mode. This mode ignores the command line syntax and prompts the user to enter the desired .EXPORT operation instructions manually.

## General Commands

- /a causes the exported files to be appended to any existing files on the archive disk(s).
- /? entered alone, displays a review of .EXPORT command line syntax on the screen.

### **Explanation:**

You can export files to any type of drive, but removable drives, such as a removable cartridge or diskette, are recommended. The diskettes (or cartridges), must already be formatted. When you invoke the .EXPORT command, a message similar to the following appears:

**WARNING: All files on drive A will be DELETED!**  
Do you wish to continue? (Y/N)

You may enter a Y to erase any files that may exist on the first export disk and begin the export operation. You may enter an N to exit the .EXPORT and the following message appears.

**EXPORT terminated.**

You may enter the /a operand to use the diskettes "as is" without erasing any existing files. The .EXPORT command will append data to any existing data on the archive disks without erasing any existing files. If you invoke the .EXPORT command with the /a operand, a message similar to the following appears:

**NOTE: Exported files will be appended to the data already  
existing on drive A:  
Do you wish to continue? (Y/N)**



MOS prompts you to change disks when exporting data that won't fit on one disk. If some space is available on the disk, MOS will copy as much of the file as the diskette can hold and then will continue copying the file on the next diskette in the set. The archive attribute is automatically removed from each file as it is exported. (See the .FILEMODE command.)

Sometimes the end of an exported file will coincide with the end of a disk's space. .EXPORT will create a directory entry with a zero file length for the next file to indicate that the .EXPORT sequence continues.

When an export operation spans multiple disks, be sure to write the sequence number on the label of each disk, since you'll be prompted to insert them in order when running .IMPORT.

The .EXPORT command normally backs up only one directory at a time. If you don't specify a directory, MOS defaults to the current directory. You can export files from all subsequent levels within a directory by including the /s operand. For example, you may have several levels of directories as follows:

\WP\BOB\JUNE91\PROJECT1

To begin the .EXPORT with directory BOB, and include all files from JUNE91 and PROJECT1 in the export, you would enter:

.EXPORT \WP\BOB\ \*.\* /s

You may export all files on a disk by using the asterisk wildcard. If the root directory is the current default, you could enter the following command:

.EXPORT \*.\* A: /s

This command will begin the export at the root directory, and export all files from all directories on the default drive. The name of each file appears on the screen as it is being exported. When .EXPORT is complete, a message similar to the following displays:

## General Commands

EXPORT: ##### files backed up to A:

This message includes information on how many files were backed up and to which drive.

If you want to make an .EXPORT set to supplement an existing set, you can use the /m operand to backup only those files that were modified since your last .EXPORT set was made. Of course, if you later need to import all of these files, you will have to run multiple .IMPORT operations - one for each .EXPORT set. When doing so, always import the oldest set first so that the newer files overwrite the older ones.

If you forget the command line syntax for .EXPORT, you can enter .EXPORT/? to get an on-screen review. If you don't like dealing with command line syntax, enter .EXPORT/q to run .EXPORT in the manual Query mode. In this mode you will be asked a series of on-screen questions about the .EXPORT operation you want to perform. Once you answer the question, the .EXPORT operation will run. This is a very easy way to use the .EXPORT program.

### Security and .EXPORT

Note that .EXPORT follows the same rules for security as any other commands. If you export files and are not signed on, only unsecured files are included in the export. If you are signed on, only the files to which you have unrestricted access are included in the export. To create a complete back up of all files on your computer, the user who invokes the .EXPORT command should be signed on with unrestricted access to all security classes.

NOTE: Never use .DISKID, or any other method, to change the Disk ID of any .EXPORT disk. Doing so will render .IMPORT inoperable and you will not be able to restore the data from that .EXPORT set.



## .FILEMODE

.FILEMODE lets you change the "read-only" or "archive" attributes of a file. The read-only attribute determines whether or not a file may be modified. The archive attribute is used with the .EXPORT command to determine whether or not a file has been changed since the last time it was .EXPORTed. The .EXPORT command clears the archive attribute. You may use .FILEMODE without any operands to display the current read-only setting of a file. The .DIR command displays files along with the current setting for both of these attributes.

**Type:**      Extrinsic.

**Form:**      .FILEMODE {+R | -R | +A | -A} *filename*

### Operands:

+R      sets the file attribute to read-only.

-R      removes the read-only attribute.

+A      sets the archive attribute of a file.

-A      removes the archive attribute setting.

*filename*      enter the name of the file, including the letter of the drive and/or directory if not the current defaults. Wildcards are permitted in the name of the file.

### Explanation:

You may change the attribute of a file to read-only to prevent changes to the file. It also prevents the file from being deleted. For example:

```
.FILEMODE +R \wp\contract.doc
```

This changes the attribute of the file, CONTRACT.DOC, in the WP directory to read-only. You may remove the read-only attribute by using the same command form and specifying -R instead of +R.

## General Commands

You may display the current attributes of a file to check the read-only setting. For example, the following command will display the current read-only setting of the LETTER.DOC file in the WP directory:

```
.FILEMODE \wp\letter.doc
```

The archive attribute is used by .EXPORT, which copies files for safekeeping and compresses their data. The archive attribute serves as a flag that is set if the file has been changed since the last .EXPORT.

The .DIR command displays the current attribute settings of a file. An asterisk (\*) appears following the last updated time for any file that has the archive attribute set. An R appears following the time of last update for any file that has the read-only attribute set. The following is an example of a directory display:

filename.ext	size	----updated----	attrb	cls	user	----created----
COMMAND.COM	6	12-30-86 12:00	*	S	B MARK	12-30-85 12:00
AUTOEXEC.BAT	179	2-20-87 09:13	*	S	E SYS	01-19-87 09:13
CONFIG.SYS	201	2-25-87 14:19	*R			02-25-87 14:19
MOS.DIR	(directory)		S			12-05-86 08:15
.	.	.				
KIM.BAT	436	1-30-87 10:44	*R			01-30-87 10:44
WORK.DIR	(directory)					01-30-87 10:49
DISKID MOS	55,329 Bytes	31 Files				11,134,976 Bytes remaining



## .FORMAT

.FORMAT prepares a disk for recording information with MOS. You must .FORMAT a new disk before you can write or read information to it. You may also use .FORMAT on a disk that already contains information to clear the information on the disk. .FORMAT will prepare up to 4 gigabytes of disk space. .FORMAT automatically places a boot sector on the disk.

**Type:** Extrinsic.

**Form:** .FORMAT *d:* [/4 | /7 | /8] [/V]

### Operands:

*d:* enter the letter of the drive that contains the disk to be formatted.

/4 creates a 9-sectored double-sided format on a 320K or 360K drive.

/7 use to format a 720K disk with a 1.4M drive.

/8 creates an 8-sectored format on a 320K or 360K drive.

/V lets you enter a disk ID of up to 11 characters.

### Explanation:

This chart shows diskette format options:

<u>Drive</u>	<u>No Operand</u>	<u>Use Operand</u>
360K 5-1/4"	360K Disk	/8 for 320K Disk
1.2M 5-1/4"	1.2M Disk	/4 for 360K Disk
720K 3-1/2"	720K Disk	
1.4M 3-1/2"	1.4M Disk	/7 for 720K Disk

## General Commands

You may use .FORMAT with hard or floppy disks. .FORMAT builds the logical format of the disk, creates the File Allocation Table (FAT) for that disk and prepares the disk to receive a directory structure. .FORMAT automatically places a boot sector on the disk. The boot sector contains information that describes the disk to the device driver in the CONFIG.SYS file. (The .MSYS command may be used independently to place the boot sector on the disk.) You must also copy the MOS system file, \$\$MOS.SYS, and the command processor files, \$\$SHELL.SYS and COMMAND.COM to the formatted disk for it to be bootable.

When you enter .FORMAT for a floppy diskette drive, say drive A, the following prompt appears:

Insert diskette in drive A:  
Press RETURN to continue.

Place the diskette to format in drive A and press ENTER. If you are formatting a hard disk, the following message will appear:

\*\*\*\*\* WARNING \*\*\*\*\*  
Formatting non-removable drive C: will erase all data it contains!  
Do you wish to format (Y/N)?

This message appears as a warning any time you select to format a hard disk. Enter a Y if you want to format the hard disk. You may enter an N to return to the system prompt without formatting the hard disk.

As the disk or diskette is being formatted, the following message appears:

Percentage complete 1 %



The percentage complete is constantly updated until the .FORMAT is complete. When the format is complete, MOS displays the total number of bytes on the disk, how many bytes are in bad sectors (if any) and the total number of bytes free. The following message then appears:

Format another (Y/N)?

You may enter Y and repeat the procedure to format another diskette. You may enter N to end the .FORMAT command and return to the system prompt.

## General Commands

---

### .HELP

.HELP displays a menu of all commands available in MOS. From the menu you may select to display a help screen for a specific command. The .HELP screen shows the command form with all available operands, and an explanation of the command.

**Type:**      Extrinsic

**Form:**      .HELP *{command}*

#### **Operands:**

*command*    you may optionally enter the name of the command for which you want to display the help screen and by-pass the menu display.

#### **Explanation:**

You may use the .HELP command at any time to display an on-line explanation of a MOS command. If you do not enter a command name as an operand, a menu listing all the commands in MOS will display.

At the menu, you may use the arrow keys to move to the command for which you want to display a help screen. Your current selection appears highlighted on the screen. Press ENTER to select the command that is highlighted. You may press ESC at the menu to exit .HELP and return to the system prompt.

If an explanation of a command is more than one screen long, you may use the arrow keys to scroll a screen up or down one line at a time. The PgUp and PgDn keys will scroll the screen one page at a time. The HOME key moves you to the beginning of the present command description, and the END key moves you to the end of the command description. When you are ready to exit the help screen, press ESC to return to the menu.

You may also use the plus (+) key on the numeric key pad to move to the next command's help screen without returning to the menu. In the same way, you may also use the minus (-) key on the numeric key pad to move to the previous command's help screen without returning to the menu.



## .IMPORT

.IMPORT must be used to restore and decompress files that were backed up with the .EXPORT command. .IMPORT lets you reinstate a single file, a group of files, a single directory or a group of directories. You may import files to any location, which need not be the directory from which they were exported.

**Type:** Extrinsic.

**Form:** .IMPORT *d:[path] {d:[path]}* [/o] [/q] [/s] [/?]

### Operands:

*d:[path]* enter the letter of the drive that contains the files you want to import. Also enter the name of the file(s) to import, including all levels of a directory name to begin the import with a specific directory. Wildcards are permitted in the name of the file.

*d:[path]* enter the drive and path of the target directory where the import is to begin. Defaults to the current drive and directory if not entered.

/o Indicates that files already existing at the target location are to be overwritten by any imported files with the same name(s).

/q enter alone for Query Mode. This mode ignores the command line syntax and prompts the user to enter the desired .IMPORT operation instructions manually.

/s Restore all nested subdirectories. Restores all levels of subdirectories from the .EXPORT set.

/? entered alone, displays a review of .IMPORT command line syntax on screen.

## General Commands

### **Explanation:**

You may reinstate all files from a set of .EXPORT diskettes (or cartridges) using the asterisk wildcard character in the .IMPORT command. For example:

```
.IMPORT a:\*.* /s c:\
```

This command will import all files and directories beginning with the first directory on the first .EXPORT disk in the set, from drive A to the root directory of drive C. You will be prompted to insert the first and subsequent archive disks until the import operation is complete.

You may use .IMPORT to restore a specific file or directory, and begin the .IMPORT with the first diskette in a set. .IMPORT will always prompt you for the next archive disk. A selective .IMPORT may require multiple .IMPORT commands. The file directory structure on the archive disk are created on the destination disk, where necessary.

If you don't use the /o operand, any attempt to import a file over an existing file with the same name will cause MOS to prompt you for verification before importing the file:

**WARNING:** The output file (filename) already exists.  
Do you want to overwrite it? (Y/N)

If you forget the command line syntax for .IMPORT, you can enter .IMPORT/? to get an on-screen review. If you don't like dealing with command line syntax, enter .IMPORT/q to run .IMPORT in the manual Query mode. In this mode you will be asked a series of on-screen questions about the .IMPORT operation you want to perform. Once you answer the question, the .IMPORT operation will run. This is a very easy way to use the .IMPORT program.

**NOTE:** the .IMPORT command follows the same rules for security as any other command. (Also see the .EXPORT command.)



## .MD

.MD lets you create or make directories. You may create as many levels of directories as you wish so long as the full name of the directory beginning at the root does not exceed 64 characters.

**Type:** Intrinsic.

**Form:** .MD |path

### **Operand:**

|path enter the complete name of the directory including all levels of directories that precede it, and the drive letter if not the current drive.

### **Explanation:**

MOS lets you create multiple levels of directories, beginning from the root directory. You may use directories to organize information.

For example, you may set up a directory called MYDIR at the root directory of a disk, and use it to store certain types of files. From the MYDIR directory you may make another directory called MYINFO in which you store text files. From MYINFO you may make yet another directory called MEMOS in which you store only memos. This directory is three levels deep and the directory name is 21 characters long, for example:

\MYDIR\MYINFO\LETTERS

You may only make one level of directory at a time. For example, you must make the MYDIR directory first. You may then make the MYINFO directory as the next directory level from MYDIR. (MYDIR is the parent directory for the MYINFO directory.) You may then make the next directory level for the directory named LETTERS. MYINFO is the parent directory for the LETTERS directory.

## General Commands

If the current directory is LETTERS, you may make a fourth level of directory by entering the following command:

```
.MD CLIENTS
```

If the current directory is the root directory, you must enter the preceding levels when you want to make a directory at a different level. To add this fourth level of directory from the root, you would enter the following command:

```
.MD \MYDIR\MYINFO\LETTERS\CLIENTS
```



## .MORE

.MORE is a filter that massages standard input to produce standard output in 24 line increments. The standard output displays on the video screen with the message, CONTINUED..., on every 25th line. Pressing any key causes the next 24 line increment of standard output to display.

**Type:** Extrinsic.

**Form:** .MORE

### Explanation:

.MORE is commonly used to filter long files or directories to display only one screen of information at a time. .MORE uses input/output redirection to accomplish the display. For example, you may use the .TYPE command to send the contents of TEXT.DOC to standard output. The standard output may be redirected as standard input to the .MORE command. .MORE filters TEXT.DOC and sends it to the video screen as its standard output, 24 lines at a time. The commands to accomplish this are:

```
.TYPE TEXT.DOC | .MORE
```

This causes the first 24 lines of TEXT.DOC to display on the video screen. Pressing any key displays the next 24 lines. You may press CTRL - C or CTRL - BRK to stop the display and return to the system prompt.

You may find .MORE useful to display a long directory map. To display a directory map one screen at a time, you would enter:

```
.DIRMAP | .MORE
```

# General Commands

## .MSORT

.MSORT lets you sort records in a file on a field-by-field basis. Records may be up to 5000 bytes long, and must occupy one line with a hard carriage return at the end of the line. You may also specify the starting column number of the field on which the records will be sorted. It is not necessary to separate the fields by spaces or commas. You may sort in ascending order or descending order. When the .MSORT is complete, the output is written back to the video screen, unless you redirect it to an output file.

**Type:** Extrinsic.

**Form:** .MSORT [/r [/a] [/dd:[path]] [/fn [/+n] [/fn,n] [/m]]{<infile>} {>outfile}

### **Operands:**

/r sorts the file in reverse order.

/a sorts characters in the file in true ASCII order, regardless of their use. If /a is not specified, characters in the range of decimal 128 to 226 will be sorted in a "weighted" format. In a weighted sort, the "a" character and the "A" take on the same value.

/dd:[path] enter /d immediately followed by the drive and/or path where temporary work files generated by .MSORT are to be placed. You may omit this operand to place temporary work files on the current drive and/or directory.

/fn enter /f followed by the number of the column where the sort string begins. The sort will occur through the end of the line (record).

/+n same effect as /fn above.

/fn,n enter /f and the number of the first column where the sort string begins, followed by a comma and the total number of characters in the string.



- /m displays several messages that you may find helpful during lengthy sorts.
- <infile enter the name of the file to sort. If you do not make an entry, MOS defaults to direct keyboard input.
- >outfile enter the name of the file where the resulting data is to be written. If you do not enter an output file name, MOS displays the sorted output on your video screen.

#### Explanation:

.MSORT creates a temporary work file to hold the sorted data. The work file is written to the default path and drive unless you specify a path and drive. The resulting sorted output will display on the video screen, unless you specify a file (the outfile) for the resulting output.

You may want to use redirection with the .MORE command to display the output file one screen at a time. .MSORT is useful for doing multiple field sorts. For example:

```
.MSORT /a /dc:\wp\ /f8,4 /m < MYFILE.DOC > DOCSORT
```

In this example, MOS will sort the records in MYFILE.DOC in ascending order, with a string of text that begins with column 8 and includes 4 positions; columns 8, 9, 10 and 11. Including the /m operand causes MOS to display the following message when the sort begins:

Sorting has begun on record nnnn.

The nnnn in this example represents the number of records read into memory for the sort. If your system does not have enough memory to sort all records in one pass, there will be multiple passes through the file and the following message will appear:

Completed Pass: nnnn nnnn records were sorted this pass!

## General Commands

The first nnnn indicates the pass number, and the second nnnn is the actual number of records sorted and written to the temporary sort file. .MSORT will continue until all records have been sorted, creating as many temporary work files as are required for the sort. .MSORT may display:

The temporary files are merging.

After merging the temporary files, MOS begins the final sort. The following message displays the number of remaining temporary files that need to be sorted during the final sort phase:

nnnn temporary files remain!

If the input file is sorted in one pass, .MSORT displays the following message during its last phase:

Creating output file.

The output in this example is redirected to the DOCSORT file. You may display the contents of DOCSORT with the MOS Editor or with the .TYPE command. Note that the performance of .MSORT is directly related to the number of records in the file; the more records to sort, the longer .MSORT runs. .MSORT is disk-intensive and may slow processing in other partitions.



## .MSYS

.MSYS writes a boot sector on a disk. This is one of the steps in creating a bootable MOS disk. The .FORMAT command does this for you automatically. However, .MSYS may be run to place the boot sector on a disk at any time.

**Type:** Extrinsic.

**Form:** .MSYS *d:*

**Operand:**

*d:* enter the letter of the drive that contains the disk on which you want to place the MOS boot sector.

**Explanation:**

The MOS boot sector must be placed on a disk for it to be a bootable MOS disk.

If you format a disk with the MOS .FORMAT command the boot sector is automatically placed on the disk for you. However, .MSYS can be used separately.

For example, you can use the .MSYS command to convert a bootable DOS disk to a bootable MOS disk without having to reformat the disk. Insert the DOS disk in drive A and enter the following command at the MOS system prompt:

.MSYS A:

**NOTE:** You must also copy the system file, \$\$MOS.SYS, and the command processor files, \$\$SHELL.SYS and COMMAND.COM to the disk before it is truly bootable.

# General Commands

## .ONLY

The .ONLY command lets you limit the action of a command or subsequent sequence of commands to a specified file or group of files. You may use .ONLY with MOS commands or with batch files. .ONLY can also be used with any programs or applications that do not operate directly on the disk, but use filename operands. Be careful using .ONLY since it offers a global functionality. If you use it with destructive commands, you may want to preview what you'll be deleting or erasing.

**Type:** Intrinsic.

**Form:** .ONLY (*filename filename ...*) DO *command | batfile*

.ONLY CANCEL

**CANCEL** If a .ONLY statement is used within a batch file, you may cancel it with a .ONLY CANCEL statement. This will free up the files in its group for operation by other commands.

### **Operands:**

*filename* enter the name of the file or files on which you want the command to perform. The file or group of files must be enclosed in parentheses. Do not enter its path. Wildcard characters are permitted. You must use a comma or space as a delimiter between filenames. You may enter as many filenames as space permits on the command line. Note that no drive or path specifications are permitted.

*command* enter the command you want to perform on the files. You may enter any operands associated with the command, including the drive and path for the file or group of files.

*batfile* enter the name of the batch file to perform on the file or group of files, but do not include the .BAT extension.

### **Explanation:**

.ONLY can be used to limit the files on which a utility command is performed. For example:



```
.ONLY (*.map *.exe *.obj *.asm) DO .ERASE *.* /v
```

In this example only those files with the extensions listed will be grouped for erasure. You may also use .ONLY to limit the files that may be processed within an application. For example:

```
.ONLY (*.doc *.txt) DO wordproc
```

This command statement invokes a word processing application on a system, but limits the files which may be accessed under it. If you choose to use this limitation, beware of any files that the application may need. For example, if you choose to place this limitation on the MOS Editor, .ED, the DEFKEYS.ED file, which is used to store previous assignments of function keys, will not be found unless it is included in the .ONLY file set.

Additionally, you can use .ONLY to invoke a new command processor and limit the files which may be accessed under it. For example:

```
.ONLY (*.doc *.txt) DO command
```

Within this new command processor, you can only access files with a .DOC and a .TXT extension.

**NOTE:** Be careful when you exclude access to files. You may create files excluded from an application or a new command processor that may already exist. Although it appears as if you are creating a new file, the existing file may be overwritten without your knowledge.

.ONLY can also be used to create a set of files and invoke a batch file to perform on those files. For example:

```
.ONLY (tl*.doc) DO CLEAN
```

This .ONLY statement will let you invoke the batch file, CLEAN. However, in this example, .ONLY's exclusion properties would permit no other batch files to be invoked from CLEAN until processing is complete since files with the .BAT extension are not included in the filename operand. The batch file CLEAN.BAT contains:

## General Commands

---

```
.CD\WORK.DIR  
.ERASE ** /y  
.CD\ASM.DIR  
.ERASE *.* /y
```

Note that this will erase all files which begin with t1 and carry a .DOC extension in the WORK.DIR and ASM.DIR directories. Had you wanted to preview the files before erasing them, you could have written CLEAN as follows:

```
.CD\WORK.DIR  
.DIR /w /p  
.ECHO Erase these files in \WORK.DIR?  
.KEY %%a IN (y,n) DO .GOTO p1%%a  
:p1y  
.ERASE *.* /y  
:p1n  
.CD\ASM.DIR  
.DIR /w /p  
.ECHO Erase these files in ASM.DIR?  
.KEY %%a IN (y,n) DO .GOTO p2%%a  
:p2y  
.ERASE *.* /y  
:p2n
```

The use of .KEY requires you to respond to the prompt. The key you press, either Y or N determines the action that will be taken. You may couple the .ONLY statement with an .EXCEPT statement to further limit the effect of a command. For example:

```
.ONLY (*.doc) DO .EXCEPT (special.doc) DO .EXPORT *.* A:
```

In this example, SPECIAL.DOC is excluded from the .EXPORT, but all other .DOC files are included. In a situation like this, the order of the .ONLY and .EXCEPT commands does not matter. The same effect could have been achieved with:

```
.EXCEPT (special.doc) DO .ONLY (*.doc) DO .EXPORT *.* A:
```

If the groups defined in the .EXCEPT and .ONLY statements conflict, no files will be found. For example:

```
.ONLY (x.doc) DO .EXCEPT (*.doc) DO .EXPORT *.* A:
```



## .PATH

The .PATH command specifies the drives and directories MOS is to search when you enter a command. When you invoke a .PATH command, MOS places the .PATH string in the environment space, which is an area of memory used to keep such items.

**Type:** Intrinsic.

**Form:** .PATH {*d:*}/{*dirname*}|;...|

### Operands:

*d:* enter the letter of the drive that MOS is to search for an executable file.

*dirname* enter the complete directory name MOS is to search, include any levels of directories that precede it from the root directory.

; enter the semicolon to link together a series of drives and directories to be searched.

... any number of drive and directory entries may be included with the .PATH command.

### Explanation:

When you invoke a command, MOS searches the current drive and directory for the executable command file. If the command is not found, MOS then looks for a search path in the environment and searches for the executable command file in the order of drive and directory entries.

You should place the .PATH command in your startup batch files to specify the path each time you boot MOS. You may want to organize the order of entries in the path with the directories of your most frequently used commands first. This will cause MOS to locate the executable command file quicker, and give you a little faster performance. For example, the .PATH for a command that invokes a word processing application you frequently use could be:

## General Commands

.PATH c:\mos;c:\work\wordproc;c:\mywork

You may enter .PATH at the system prompt without any operands to display the current path setting.

You may delete the current path by entering .PATH ; (the space and semicolon after .PATH are necessary parts of the command).

Entering a new .PATH command will delete any current path in the environment and replace it with your new entry.



## .PROMPT

.PROMPT lets you change the appearance of the MOS system prompt. The default prompt is [d:\], which is the current drive and directory enclosed in brackets. You may use any of the following operands and/or any characters to create a different prompt. Note the use of the \$ that serves as a special delimiter for the .PROMPT operands.

**Type:** Intrinsic.

**Form:** .PROMPT {\$b}{\$d}{\$e}{\$g}{\$h}{\$i}{\$l}{\$n}{\$p}{\$q}{\$t}  
{\$u}{\$v}{\$\_}{\$}{\$x}{text}

### Operands:

\$b displays the ! character.

\$d displays the system date in the format chosen by the COUNTRY command statement in your CONFIG.SYS file.

\$e displays the ESCape character; primarily used with the ANSI mode of operation.

\$g displays the > character.

\$h sends a destructive backspace to the current prompt.

\$i displays the number of the current partition (task ID).

\$l displays the < character.

\$n displays the letter of the current default drive.

\$p displays the current default drive and directory.

\$q displays the = character.

\$t displays the system time in the format chosen by the COUNTRY command in your CONFIG.SYS file.

## General Commands

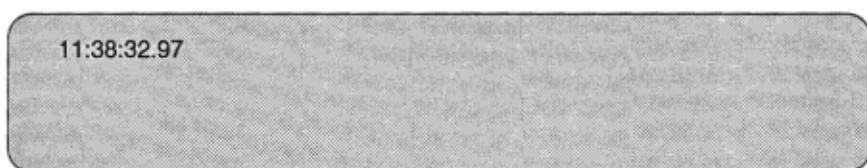
- \$u displays the user ID of the user signed on in the current partition, if any.
- \$v displays the operating system name and version number.
- \$\_ ends current prompt line and continues the prompt on the next line.
- \$\$ displays a single \$.
- \$x To include leading delimiters in the prompt, enter \$ and an x or any other character not used as the second character in the previous operands.
- text you may enter any text in the prompt that you want.

### **Explanation:**

You may set the .PROMPT for partition 0 in your AUTOEXEC.BAT file. You may set the .PROMPT that will appear for each partition in the startup batch file that is invoked from the .ADDTASK command for that partition. You may also modify the prompt display at any time with entries at the system prompt. For example, to set the prompt of the current partition to display the system time, enter the following at the system prompt:

```
.PROMPT $t
```

The prompt at the current partition will display:



```
11:38:32.97
```

The brackets display only with the default prompt, unless you enter them as text in your .PROMPT command. To display only the hours and minutes in the prompt, you may include the \$h operand six times to backspace over :ss:hh and omit them from the prompt. For example:

```
.PROMPT $t$h$h$h$h$h$h
```



If you want your display to include the word TIME and a colon prior to the time, you could enter:

```
.PROMPT TIME:$t
```

To return the prompt to its default, enter .PROMPT in the partition without any operands.

The first entry with the .PROMPT command cannot contain a delimiter, such as a space, a comma, etc. You may create a null or blank position in your prompt by typing a \$ followed by a character that is not used as the second character of a valid operand.

For example, \$x is not a valid operand, but can be entered as the first operand in the PROMPT command. Any characters, including delimiters may follow this operand. For example, the following command displays three blank spaces, and then JOE> as the prompt.

```
.PROMPT $x  JOE$g
```

You may use any combination of operands for a prompt display. For example, the following creates a prompt that displays the current drive and directory (\$p), the greater than symbol (\$g), the characters JOE, and another greater than symbol (\$g):

```
.PROMPT $p$gJOE$g
```

The prompt may appear similar to the following:

```
C:\WORK>JOE>
```

Note that .PROMPT produces a string which is stored in the environment (an area of memory used to hold frequently called items).

## General Commands

### .RD

.RD lets you remove an empty directory from the system. The directory to remove cannot contain any files, and cannot be a parent to another level of directory.

**Type:** Intrinsic.

**Form:** .RD \path

**Operand:**

**\path** enter the complete name of the directory, including the letter of the drive and/or any preceding levels of directories.

#### **Explanation:**

Before a directory may be removed, you must delete all files in the directory, or use .RENAME to move them to a different directory. You cannot delete a parent directory that contains any other levels of directories.

You may only remove one directory level at a time. For example, you may have a directory named \MYTEXT\MYFILES\LETTERS. In order to remove the MYFILES directory, you must first remove the LETTERS directory.

Normally, you must specify all levels of directories that precede the directory you want to remove. For example, you could remove the LETTERS directory with the following command:

```
.RD \MYTEXT\MYFILES\LETTERS
```

If MYFILES is the current directory, you could remove the LETTERS directory with the following command:

```
.RD LETTERS
```

Note that you cannot remove a directory that is your current directory, nor can you remove a directory that is currently being used by someone in another partition.



## .REL

.REL creates a display on the video screen of the release of MOS that is currently running on your computer.

**Type:** Intrinsic.

**Form:** .REL

### **Explanation:**

Entering .REL at the system prompt displays the current release of MOS that is on your computer. The display appears similar to the following:



PC-MOS Version #.##

## General Commands

### .REMDEV

.REMDEV lets you remove a device driver. You may enter this command at the system prompt to remove the device.

**Type:** Extrinsic.

**Form:** .REMDEV *devname*

**Operand:**

*devname* enter the device driver you want to remove.

**Explanation:**

You may use the .REMDEV command at any time to remove a device that is no longer necessary. For example, you may have a pipe device defined that you want to remove to free memory for other uses. If the pipe device name is PIPE4, the command you would enter to remove it is:

.REMDEV PIPE4

There may be devices on your computer that are active or are required. If you attempt to remove one of these devices, MOS will display a message stating that the device cannot be removed. (Also see the .ADDDEV command and the DEVICE= command statement.)



## .RENAME

.RENAME lets you change the name of a file or group of files and optionally move them to a different directory. You may also use .RENAME to relocate or change the name of a subdirectory.

**Type:** Intrinsic.

**Form:** .RENAME *name name [/m]*

### Operands:

*name* enter the name of the file or directory you want to rename or move. You may include the letter of the drive and/or the directory name if not at the current defaults. Wildcards are permitted with file names but not with directories.

*name* for renames enter the new name of the file or directory. If not specified, MOS defaults to the current directory. If you want to move the file(s) to a different directory, enter the name of the directory where the file(s) are to be moved. When moving a directory, you must enter the complete path name for its new location. You may rename and move at the same time. Wildcards are permitted with file names but not with directories.

*/m* you must enter this operand if you want to move the file(s) or directory.

### Explanation:

.RENAME is useful for changing the name of a file. For example, you may want to temporarily change the contents of a batch file, without altering its original contents. You can rename the original batch file with a different extension, for example:

```
.RENAME AUTOEXEC.BAT AUTOEXEC.SAV
```

## General Commands

You can then create a new AUTOEXEC.BAT to use when you boot the computer. When you want to revert to the original file, delete the new AUTOEXEC.BAT and then rename AUTOEXEC.SAV with the .BAT extension.

You may use an asterisk or a question mark as wildcard characters to rename or move a group of files. For example, you may have several files in a directory with an extension of .BAK, and you want to change the name of all these files to contain an extension of .DOC. You may enter the following command to rename the entire group of files and leave the group in the current directory:

```
.RENAME *.BAK *.DOC
```

You can also move a file or a group of files to a new directory. However, .RENAME cannot move files from one drive to another drive. The new directory must exist on the same drive as the original directory. To move the group of .BAK files to a different directory without changing the file names, you could enter the following command:

```
.RENAME \MYDIR\*.BAK \NEWDIR /m
```

To move files to a different directory, you must include the /m operand in the command. You may also give the file or files that you are moving to a different directory a new name, for example:

```
.RENAME \MYDIR\LETTER.TXT \NEWDIR\FORM.TXT /m
```

.RENAME is also useful for renaming and moving directories. For a rename only, the second operand defaults to a directory with the same path as the first operand if not specified. For example, the following command would rename the \SALES\JAN directory to the \SALES\JANUARY directory:

```
.RENAME \SALES\JAN JANUARY
```

When moving a directory, the second operand must specify the complete path name for the new location of the directory that is being moved. For example, the following command will move the LETTERS directory from its present location in the root to a new position as a child of the WP directory:



---

```
.RENAME LETTERS WP\LETTERS /m
```

Note that all files and subdirectories within the LETTERS directory are carried along with it. This makes it possible to easily restructure an entire disk, since whole limbs of the directory tree structure can be pruned from one location and grafted onto another.

In all previous examples the commands were entered at the root directory of the C drive, [C:\]. The next command is entered at the \WP subdirectory prompt, [C:\WP], as follows:

```
.RENAME \DOCUMENT MANUAL /m
```

In this example DOCUMENT is presently a subdirectory off of the root. Since the second operand (MANUAL) does not start with a "\", the current directory (\WP) is assumed and the \DOCUMENT directory is moved and renamed to \WP\MANUAL.

When using .RENAME with directories the operands must be specific directories by name, no wildcards can be used. The root directory, of course, cannot be renamed or relocated. A directory cannot be relocated to be a child of itself. You also cannot rename and/or relocate a directory if that directory is presently the current directory of any task or if it is a parent of the current directory of any task.

# General Commands

## .SEARCH

.SEARCH searches one or more files to locate a specific string of characters. If a filename is not specified, then .SEARCH reads from standard input as a filter to produce redefined output.

Type:      Extrinsic.

Form:      .SEARCH {/n}{/v}{/c}{/i} "string" {filename ...}

Operands:

/n      causes the number of the line that matches the character string to display on the output device along with the character string.

/v      causes the lines where the character string does not match to display on the output device.

/c      counts and displays on the output device the total number of lines in which the character string is matched in each specified file. The /c option overrides the two previous options.

/i      causes the search to ignore upper or lower case.

"string"      string is the characters to locate. The string must be enclosed in double quotes. If a double quote ("") appears within a character string, MOS will read it as the end of the string. You may, however, enter two consecutive double quotes ("") and MOS will recognize them as one double quote within the character string. For example, the string "the "command" name" is not valid, but "the ""command"" name" is a valid string.

MOS will find only character strings that match explicitly. For example, "text" and "text" match, while "Text" and "text" do not match because of capitalization. The .SEARCH will find strings contained within words.



*filename* enter the name of the file to search, including the drive letter and/or directory if not the current defaults. Wild cards are not permitted in the file name.

... multiple filenames may be present in the command. A space must be used between each filename.

#### Explanation:

Besides searching for character strings in a file, .SEARCH can also be used as a filter. For example, to find all files created or last changed on a specific date, you can use the .SEARCH command as follows:

```
.DIR | .SEARCH /n "10-05-87"
```

The first command tells MOS to display the directory on the current default drive. The pipe symbol (|) tells MOS to treat what is on each side of the symbol as separate commands, but to use the standard output of the command on its left (.DIR) as the standard input for the command on its right (.SEARCH). The result is a display to the video screen of only the files in the directory created or last updated on 10-05-87.

.SEARCH will locate a specific character string in multiple files. If the files are on different drives, or reside in different directories, then the drive letter and directory must be specified. In the following example, MOS will locate and display all occurrences of the word "terminal" in the two files that reside in different directories.

```
.SEARCH "terminal" \wp\doc1.txt \wp3\glos.txt
```

## General Commands

### .SET

The .SET command lets you place character strings for specific variables in the environment space, which is a reference area for frequently accessed strings of information. A copy of the information contained in the environment is made available to all programs.

**Type:** Intrinsic.

**Form:** .SET *variable*=*string*

#### **Operand:**

*variable*= enter the name of the environment variable you want to set. Although you may enter any variable string, the most commonly used environment variables and their uses are:

COMSPEC - tells MOS where the Command Processor is located so it knows where to look to reload the command processor after a program has overwritten it in RAM.

PATH - tells MOS the drives and directories to search when you invoke an executable file. Executable files carry an extension of .COM, .EXE and .BAT. MOS searches for executable files in that order. The .PATH command is an alternative to .SET PATH=.

PROMPT - defines the prompt that will be displayed for the partition in which the .SET command is invoked. The .PROMPT command is an alternative to .SET PROMPT=.

*string* enter the value to set for the variable.

#### **Explanation:**

When you boot MOS, it automatically sets COMSPEC=C:\COMMAND.COM into the environment, which is approximately 127 bytes in size. PATH and PROMPT are automatically set into the environment space when they are invoked from a batch file or at the system prompt. The .SET command lets you change any of these values.



You may display the current variables that are set in the environment by entering the .SET command without any operands. MOS will display the current settings similar to the following:

```
COMPSPEC=C:\MOSBIN.DIR  
PROMPT=$P$G  
PATH=C:\WP\WORK
```

You can delete items set into the environment by entering the .SET command, the variable name, an equals symbol and then pressing ENTER. The following example would delete the prompt setting for the current partition:

```
.SET PROMPT=
```

Although the environment is approximately 127 bytes in size, it may grow larger with each .SET command. When you install a resident program, the environment is restricted from any further growth. Invoking a nested batch file may also restrict further growth of the environment while the nested batch file is active. For these reasons, you should invoke all .SET commands prior to installing a resident program or invoking a nested batch file.

**NOTE:** You may want to include all .SET commands at the beginning of your AUTOEXEC.BAT file to be sure there is no restriction on the environment growth while variables are being set. Do not invoke a resident program or a nested batch file from the AUTOEXEC.BAT file prior to entering a .SET command.

You may also use the .ENVSIZE command to specify a minimum size for your environment. .ENVSIZE lets you reserve the space for your environment prior to loading any resident programs.

## General Commands

### .TIME

.TIME lets you set or display the MOS system time. The format for time is defined with the COUNTRY command statement in your CONFIG.SYS file. The .TIME is the internal time that is known to MOS. You will need to set the system time immediately after booting MOS if your system does not have a perpetual clock. The .TIME a file was created and last updated appears with each file when a directory is displayed.

**Type:** Intrinsic.

**Form:** .TIME {hh:mm{ss{.hh}}}

#### **Operands:**

*hh:mm* enter the hour (hh) and minutes (mm) for the correct time.

*:ss* enter the number of seconds for the correct time.

*.hh* enter the number of hundredths of a second for the correct time.

#### **Explanation:**

If your system does not have a perpetual system clock, you may want to include .TIME in your AUTOEXEC.BAT file. This will cause MOS to prompt you to enter the current time when you boot your computer.

You may set the current time by entering .TIME at the system prompt, followed by the current hour, minutes, seconds and hundredths of a second.

You may type .TIME at the system prompt, without any operands, to display and optionally reset the current time. The display appears similar to the following:



14:24:03.42 is current time

New time is:

You may press **ENTER** to leave the current time setting, or you may enter a different time. The format for time is hours (hh), minutes (mm) seconds (ss) and hundredths of a second (hh).

# General Commands

## .TYPE

.TYPE displays the contents of a file on the video screen. You may use the /h option for a display in a hexadecimal form.

### Type:

Intrinsic.

**Form:** .TYPE *filename [/h]*

### Operands:

*filename* the name of the file you want to display.

/h displays the file in a hexadecimal form.

### Explanation:

If you select to display a text file, it appears on the video screen similar to the following example:

this is an example of a normal text file display

If you choose the hexadecimal form, the actual hexadecimal codes for the contents of the file display. If the file is a text file, or any text appears in the file, the hexadecimal codes will appear and their "text equivalents" will be displayed to the right. The following is an example of a CONFIG.SYS file displayed in hexadecimal form.

```
00000000 0D 0A 64 65 76 69 63 3D 6D 6F 73 64 64 64 63 6C ..device=moscddcl  
00000010 6B 2E 73 79 73 0D 0A 64 65 76 69 63 65 3D 71 75 k.sys..device=qu  
00000020 69 63 6B 76 69 64 2E 73 79 73 0D 0A 64 65 76 69 ickvid.sys..devi
```



Note the periods that appear in the text equivalent column to the right. Periods represent actual periods in the file or letters in the ASCII character set that could not be translated into hexadecimal codes.

If the file you are typing is very long, you may want to use the MORE command to display the file one screen at a time. For example, the following command will type the contents of the MYFILE.DOC to the .MORE command. The .MORE command will display the file one screen at a time.

```
.TYPE MYFILE.DOC | MORE
```

## General Commands

### .VERIFY

.VERIFY is a diagnostic command that verifies a disk by checking for file allocation errors. You may select to display any errors found, display non-contiguous files or automatically fix the errors. .VERIFY checks for allocation errors; it is not a surface analysis. It is wise to run .VERIFY if your system fails due to a power outage or other cause. .VERIFY displays the allocation status showing the number of lost clusters, and any cross allocated clusters (allocated to more than one file). .VERIFY should be run from task 0 with all other tasks removed.

**Type:** Extrinsic.

**Form:** .VERIFY [*d:*] [/c /f /v] | [/h]

#### **Operands:**

- d:* enter the letter of the drive that contains the disk to be checked, or omit to check the current drive.
- /c* displays any non-contiguous files as they are being checked.
- /f* fixes the lost allocation clusters by combining them into files named LOSTDATA.nnn. The nnn in the file name is a consecutive number assigned to each new file as it is created.
- /v* displays the directory path and file names on the disk as they are being checked.
- /h* displays a help screen for the .VERIFY command.

#### **Explanation:**

.VERIFY should be run from task 0 with all other tasks removed. .VERIFY may be run on any type of disk. If you include the /c operand, any files with non-contiguous clusters display on the screen as they are being checked, for example:



A:\WORKFILE.SAV has 2 non-contiguous clusters

If you specify /v, the file and path names on the drive being checked appear on the screen. The .VERIFY allocation status display notes incorrect file sizes by comparing the actual file size to the number of clusters it occupies, for example:

A:\MYFILE.SAV allocation chain truncated to match file size

If you specify /f, .VERIFY writes the data in the lost file allocation clusters into files. These files are named LOSTDATA.nnn, and are numbered in the order in which they are encountered. You may then display and check the files to decide whether or not they should be deleted. Deleting these files will free their disk space for other uses.

A sample allocation status display appears on the screen similar to the following:

Deleting any previous LOSTDATA.nnn files.

A:\WORKFILE.SAV has 2 non-contiguous clusters

A:\MYFILE.SAV allocation chain truncated to match file size

A:\TEXT.DOC has 2 non-contiguous clusters

Current DISKID of device: WORKDISK

Total files in system:	39	
Total subdirectories:	0	
Total fragmented files:	2	
Bytes in bad sectors:	0	OK
Bytes available on drive:	44,032	43K
Total bytes on drive:	362,496	354K

## General Commands

### MOS SECURITY and VERIFY COMMAND CAUTION

When using MOS SECURITY for file level security, care must be exercised when using the VERIFY command to check a disk for file allocation errors! ONLY users with "level 3 - unrestricted access" assigned for ALL file classes should be allowed to run VERIFY! (This level of access is usually reserved for someone at the system administrator level.)

**CAUTION:** If the user running VERIFY does not have level 3 access for all file classes, VERIFY will not be able to properly read the disk.

\* **WARNING \*** If the user runs VERIFY with the /f operand to fix lost allocation clusters, there could be some disk corruption if they do not have level 3 access assigned for all file classes!



## .WVER

.WVER causes any subsequent disk writes to be verified. Note that the verify function may slow processing as a verification check is performed with each operation.

**Type:** Intrinsic.

**Form:** .WVER {on | off}

### Operands:

*on* turns on the verification check.

*off* turns off the verification check.

### Explanation:

The default status of .WVER is off when you boot your computer. You may display the current status of .WVER by typing the command at the system prompt without any operands. This will display a message similar to the following:

WVER IS ON

When .WVER is on, MOS performs a check of all output sent to a disk to be sure it is identical to the copy still in memory. If the original in memory and the output on disk are not the same, MOS displays a critical error message similar to the following:

DATA ERROR READING DRIVE A  
Enter A to abort, R to retry

You may enter an A to abort the current function and return to the system prompt, or enter an R to retry. If an error appears while .WVER is active, you may have a problem with your disk or a hardware problem with your computer.

---

# **CHAPTER 5:**

## **MULTI-TASKING / MULTI-USER**

---

Multi-Tasking Concepts .....	5 - 4
Multi-Tasking Processing .....	5 - 7
Multi-User Concepts .....	5 - 7
Multi-User Processing .....	5 - 8
Time Sharing .....	5 - 8
Rebooting Tasks .....	5 - 9
Multi-Tasking and Multi-User Commands .....	5 - 10
.ADDTASK .....	5 - 11
.REMTASK .....	5 - 18
Partition Access .....	5 - 19
Partition Access Keys .....	5 - 19
.SWITCH Command .....	5 - 20
Turning Partition Access On and Off .....	5 - 20
MOS Utility Functions .....	5 - 20

# Multi-Tasking / Multi-User

---

The .MOS Utility Commands .....	5 - 21
.MOS MAP .....	5 - 22
.MOS DIS .....	5 - 23
.MOS NODIS .....	5 - 23
.MOS USEIRQ .....	5 - 23
.MOS FREEIRQ .....	5 - 26
.MOS IRQ .....	5 - 27
.MOS WAIT .....	5 - 28
.MOS VMODE .....	5 - 29
.MOS SERINIT .....	5 - 30
.MOS ROUTE (COMn) .....	5 - 31
.MOS ROUTE (LPTn) .....	5 - 31
.MOS ROUTE TERM .....	5 - 32
.MOS ROUTE NOTERM .....	5 - 32
.MOS RESIZE .....	5 - 32
.MOS INFO .....	5 - 33
.MOS DSPORT .....	5 - 35
.MOS MOUSE .....	5 - 36
.MOS KEYB .....	5 - 38
Foreign Keyboard Drivers .....	5 - 39
.MOS FILES .....	5 - 40
.MOS HOLD LPTn .....	5 - 40
.MOS TSR .....	5 - 41
.MOS ANSI .....	5 - 42
.MOS DOSVER .....	5 - 42
The .MOSADM Utility Commands .....	5 - 43
.MOSADM SLICE .....	5 - 44
.MOSADM PRI .....	5 - 45
.MOSADM CACHE .....	5 - 46
.MOSADM SWITCH .....	5 - 46
.MOSADM TMFACTOR .....	5 - 46
.MOSADM HOLD LPTn .....	5 - 47
.MOSADM VIRQ .....	5 - 48
.MOSADM RESET .....	5 - 48
.MOSADM TIME .....	5 - 49
.MOSADM EMSLIMIT .....	5 - 49



---

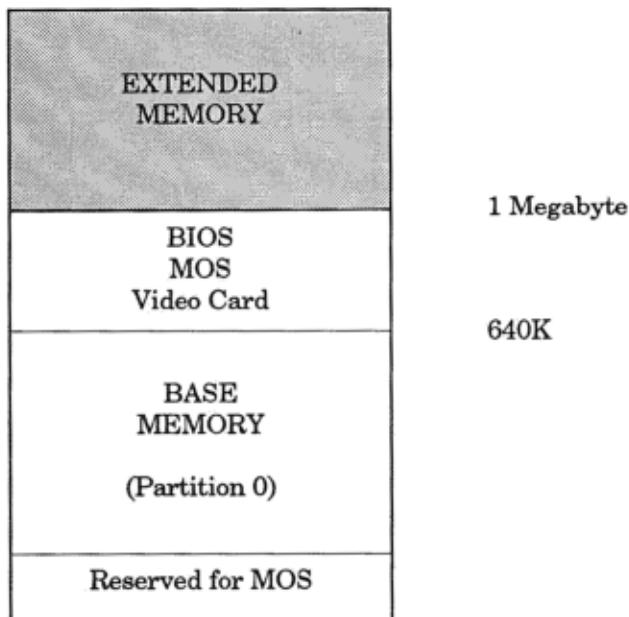
Connecting Terminals and Workstations . . . . .	5 - 52
Terminals vs Workstations . . . . .	5 - 52
Serial Ports . . . . .	5 - 53
Cable Connections . . . . .	5 - 53
Modem Cable Connections . . . . .	5 - 54
Terminal and Workstation Device Drivers . . . . .	5 - 55
Terminal Display Differences . . . . .	5 - 57
Escape Sequences for Non-PC Type Terminals . . . . .	5 - 58
Escape Sequence Chart . . . . .	5 - 61
.KEYMAP . . . . .	5 - 62
Modems . . . . .	5 - 64
MODEM.COM . . . . .	5 - 64
Defining Modem Types . . . . .	5 - 65
SunRiver Terminal Driver . . . . .	5 - 66
MOS System Monitor . . . . .	5 - 68

# Multi-Tasking / Multi-User

## Multi-Tasking Concepts

The first time you boot your computer with MOS, the command processor loads and all remaining base memory (memory below 1 megabyte) is available in a single-tasking environment. MOS gives you the ability to allocate the available extended memory (memory above 1 megabyte) into partitions to create a multi-tasking environment. A partition in MOS is a portion of memory which is given processing time for running programs or functions.

By dividing your computer's memory into partitions, you can simultaneously run more than one program. The following illustration shows how your computer's memory is organized before multi-tasking is set up, with only MOS loaded on your computer.



MEMORY BEFORE ADDING PARTITIONS



This diagram shows the first megabyte of addressable memory, which is contiguous RAM that we call base memory. The base memory is typically 640K, but depending on your system can often be less, and sometimes can be as much as 704K.

The upper portion of this first megabyte from 640K to 1MB is called high memory and is typically used for BIOS, the MOS command processor, video card memory and other add-in board memory.

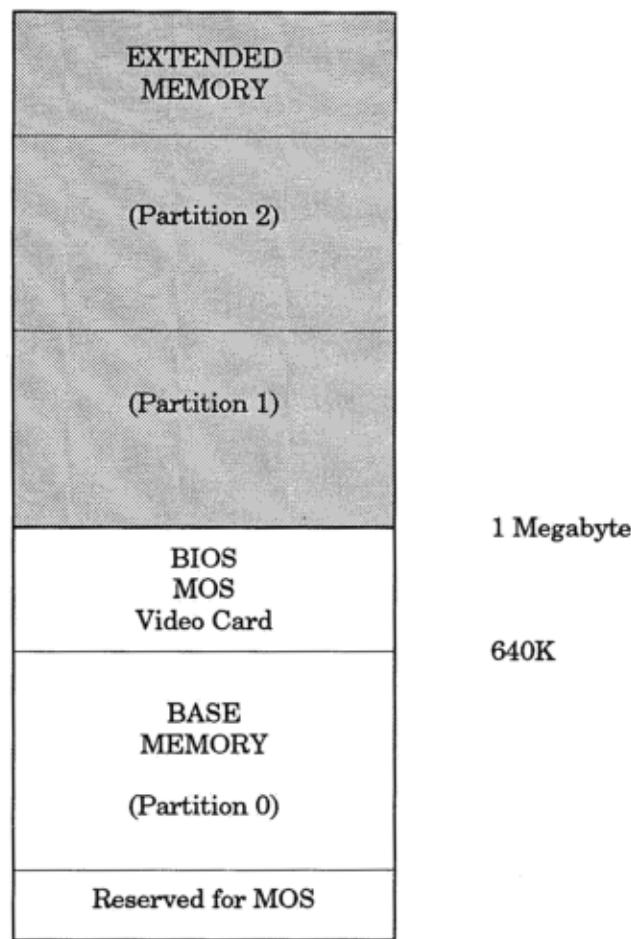
Note that a small portion of low base memory is used by the MOS command processor, and about another 16K is reserved for use by MOS. Sometimes more than 16K is reserved for MOS if FREEMEM space is not large enough. (See FREEMEM in the Configuration chapter of this manual.)

The first time this computer is booted with MOS, most of the base memory is available for processing in a single-tasking environment. This area of base memory is referred to as partition 0. Only one program or function can be active in partition 0 at a time.

MOS lets you create additional partitions from the available extended memory (or base memory) in your computer. Each partition you create can run programs and functions separate from any other partition. The .ADDTASK command is provided with MOS to let you add tasks (which we call partitions), and define the amount of memory to allocate for each partition. The .ADDTASK command is explained in detail later in this chapter.

For example, you could add two additional partitions in the extended memory of this computer. Each partition is identified by a number. The two partitions we add will be partitions 1 and 2. The following diagram illustrates the memory divisions for multi-tasking.

## Multi-Tasking / Multi-User



### MEMORY AFTER ADDING PARTITIONS

When you load MOS, a portion of the command processor is available for sharing among the partitions. When you add a partition, the portion of the command processor that cannot be shared loads in the memory allocated for that partition.

The command processor will occupy a small portion of memory in each partition you add. You may set up as many partitions as the amount of memory in your computer will accommodate.



## Multi-Tasking Processing

There are many advantages to processing in a multi-tasking environment. For example, you may be working in partition 0 and process a lengthy report. While the report is running in a single-tasking environment, you can do nothing but wait.

With multi-tasking capability, you could switch to partition 1 and load and process a spreadsheet. While both partitions 0 and 1 are active, you can switch to partition 2 and perform any other type of processing you desire.

If you have a modem, you may set up one partition that does nothing except transmit and receive files. While the modem is active, you may be processing in other partitions.

Although processing may be active in several partitions, you may access and display the processing in only one partition at a time. You may easily switch from one partition to another by pressing the access keys that are assigned to each partition.

To access and display more than one partition at a time requires a multi-user environment.

## Multi-User Concepts

The multi-user versions of MOS let you connect either terminals or workstations to the host computer so that more than one user has access to the resources of your computer. Each workstation functions in a similar manner as the main console.

Terminals communicate with the computer through serial ports. The .ADDTASK command lets you assign the serial port to which each terminal is connected, and the terminal type and baud rate for communication.

Workstations consisting of PC monitors and keyboards connect to the computer through additional hardware and add-in boards. The .ADDTASK command is used to assign the logical port number to which each workstation is connected and the workstation type.

# Multi-Tasking / Multi-User

A user at any workstation or terminal can access any partition set up on the computer. The users are sharing the computer's processing power and all available devices; disk drives, printers, etc. Each user appears to be processing as if they were a separate computer, although they are sharing the same computer's resources.

## Multi-User Processing

The multi-user capability of MOS provides many advantages. For example, while one user is running a word processing application, another user may be working with a spreadsheet application. With an accounting application, one user may be entering orders while another user is printing invoices and another user is entering inventory receipts.

If you run an application in a multi-user environment, you should be aware if the application was written to facilitate record and file sharing. If not, you may corrupt your files if two or more users attempt to write to the same file at the same time.

There are many features in MOS that let you define the processing environment. Security lets you restrict a user's access to partitions, directories and files. Security may be set up for only a few users, or for all users. You should review the Security chapter in this manual to determine whether or not you want to use security on your system.

The MOS Print Spooler provides a method to control the printing operations of your computer. If two users send output to the same printer at the same time without spooling, the output will be garbage. The spooler intercepts printer output and sends one file at a time to the printer.

## Time Sharing

The multi-tasking capability of MOS is based on the principle of time sharing. The SLICE command statement in the CONFIG.SYS file controls the amount of time each partition receives. If your computer has only partition 0, then partition 0 is using all the processing time. If you add partition 1, then processing time is shared between the two partitions.



When you first boot your computer, the default value for the SLICE command statement is one time tick, which is approximately 1/18 of a second. MOS constantly gives each partition in turn an equal share of the processing time. Partitions that are waiting, or "at rest" always give up their turn.

As you add more partitions or more users, you may notice slightly slower processing time in each partition. This is because the more active partitions there are, the longer it takes MOS to get to each partition to give it processing time.

There are several utility functions in MOS that let you control the processing time allocated to each partition, as well as whether or not a specific partition is to use disk caching. The command that provides these functions is .MOSADM.

There is also a .MOS command that provides utility functions to let you control keyboard checking, the use of IRQ's, the initialization of serial ports, the video mode and several other necessary utilities. You may also use a utility function in the .MOS command to adjust the size of a partition.

## Rebooting Tasks

Multiuser partitions may be rebooted independently of the host computer by pressing the CTRL - ALT - DEL keys at the same time on the workstation's keyboard.

The workstations own task MUST be displayed on the screen before pressing the keys to reboot, i.e. you may not have switched out of your task to view another task's display. If you have switched out of your own task, pressing CTRL - ALT - DEL will have no effect. To reboot, switch back into your own task and press the keys again.

Workstations viewing a rebooted task on their monitors will be switched back to their own task upon reboot of the viewed task.

As usual, pressing the CTRL - ALT - DEL keys simultaneously on the host computer keyboard will reboot the entire system. In addition to rebooting the host partition (task 0), this is of course the only way to reboot any multitasking background partitions on the host computer.

## Multi-Tasking / Multi-User

### Multi-Tasking and Multi-User Commands

The .ADDTASK command lets you create a partition for multi-tasking capability. If you have a multi-user version of MOS, you may include the appropriate operands to define the environment for communicating with a workstation or terminal.

The .REMTASK command may be used to remove any partitions that are currently defined.

The .SWITCH command allows you to switch to a specific partition with a command line entry rather than with the ALT - number key partition access method.

The following explains the .ADDTASK command, .REMTASK command and partition access in detail.



## .ADDTASK

The .ADDTASK command lets you create a memory partition (task). If you have multi-user capability, you may assign a terminal ID, port number and baud rate or workstation number for the partition. The .ADDTASK command may be included in the AUTOEXEC.BAT file to automatically add a task when you boot your computer. You may also enter an .ADDTASK command at the MOS system prompt from a workstation or the main console to add a task at any time.

**Type:**      Extrinsic.

**Form:**      .ADDTASK *memsizeK | MAX,[task ID],[class],[startup],{term ID},{port,baud rate} | {workstation number}*

### Operands:

*memsize*      enter the amount of memory you want to initially allocate to the task, which must be at least 32K. The maximum is determined by the amount of free memory that is remaining on your computer, and cannot be larger than approximately 640K. The exact maximum depends on your system configuration. The memory size should be large enough to facilitate whatever applications are to be run in the partition.

*K*                you may enter a K to document that the memory size for the partition is in kilobytes. However, the K with the memory size in the .ADDTASK command is optional.

*MAX*              entering MAX instead of a memory size will add a task of maximum possible size for your system configuration.

*task ID*          enter up to two digits for the task number to assign to the partition. Zero is reserved for the partition in base memory. The number entered corresponds to the access key(s) by which the partition is accessed. If not entered, MOS assigns the first available task number in sequence.

## Multi-Tasking / Multi-User

---

- class* enter the security class to assign to the partition.  
Defaults to a blank class if not entered. (If security is used you must create the file \$\$USER.SYS in your root directory to define security parameters for each user. See the Security chapter in this manual.)
- startup* enter up to eight characters for the name of a startup batch file for the task. The startup file must be on the root directory of the drive from which the .ADDTASK command is invoked. Enter only the file name without a drive identifier, directory name or .BAT extension. If a name is not entered, MOS does not use a startup batch file. (This file is similar to the host computer's AUTOEXEC.BAT file in that it is used to customize the path, prompt, or other variables for the task.)
- term ID* in a multi-user environment, enter up to eight characters for the file name of the terminal or workstation device driver for the partition. Enter the file name without the .SYS extension. All drivers must be defined with the DEVICE= command statement in your CONFIG.SYS file. The following are the terminal/workstation device drivers provided with MOS:
- PCTERM - for PC type terminals, Ampex 232, Falco 5500, Kimtron KT-7 PC and KT-70, Link Technologies MC1, MC3, MC5 and PCTerm, TeleVideo PCS1, Wyse Wy-60 and Wy-150.
- SH - for MaXtation SH-4/M workstations
- VNA - for Video Network Adapter workstations
- TTTERM - for Teletype type terminals
- AVTERM - for ADDS Viewpoint type terminals
- 3ATERM - ADM 3A
- T1TERM - Televideo 910



52TERM - DEC VT-52  
31TERM - IBM 3101  
19TERM - Zenith Z-19  
TVTERM - Televideo 912C, 920C, 925  
EXTERM - Excel 42/44  
ANTERM - ANSI terminals  
ATTTERM - AT&T 605 Business Communications Terminal  
ELTERM - for EmuLink Terminal Emulation Software.  
SRTERM - SunRiver Fiber Optic Workstation

*port*

The logical port number of the serial port through which the terminal for this task is connected to the host computer. This port number is based on the order it is listed in the \$SERIAL.SYS statement in your CONFIG.SYS file. For example, 2 represents the second serial port address listed in the DEVICE=\$SERIAL.SYS statement. See \$SERIAL.SYS in the Configuration chapter.

**NOTE:** Other add-in serial port devices may also be used to connect terminals, such as a Maxspeed intelligent serial port board. If so, the port addresses are assigned sequentially as the various drivers are listed in the CONFIG.SYS file. You can use the MOS INFO command to display a list of the serial drivers loaded on the system and what logical port numbers have been assigned to each device.

*baudrate*

in a multi-user environment, enter the baud rate at which the terminal and the host computer communicate. You may set the baud rate at up to 115,200. However, most terminals work reliably only when the baud rate is set at 9600, 19200 or 38400.

## Multi-Tasking / Multi-User

**workstation number** Enter the number of the workstation, e.g. 1 for the first workstation, 2 for the second, etc. (This operand is used for hardware-based workstations like VNA and Matrix workstations, and replaces the "port" and "baud rate" operands used with terminals.)

### **Explanation:**

When you boot MOS the first time, only partition 0 in base memory is available. You must add each additional partition you want on the system by using the .ADDTASK command.

To add a multi-tasking partition, only the memory size or "MAX" is required with the command. For example:

ADDTASK MAX

This will add a background task of maximum size with the next sequential task ID number, a blank security class and no startup batch file for customizing the path, prompt, or other variables for that task.

You may add tasks of a specific size by including the memory size, in Kilobytes, with the command. For example:

ADDTASK 512

If other than the default settings and/or a startup batch file is needed, they can be entered with the command. You must enter a comma in the command line as a delimiter between the operands. You may omit an operand in the .ADDTASK command by entering a comma in place of the operand. For example, to add a partition and assign only a memory size the startup batch file TASK.BAT, the command is:

.ADDTASK 256,,TASK

Note that a comma is entered in the positions for the task ID and security class. You do not need to enter any commas after the last operand in the command line.



You must use the partition access keys associated with the task number to access the partition. Partition access keys are explained following these commands.

If you add three partitions with task numbers 1, 2 and 3, and then use the .REMTASK (remove task) command to remove task 2, only tasks 1 and 3 are defined. If you then add another task and do not assign a task number, MOS will assign the new partition task number 2, which is the first available number. For better control, you may want to specify a task number, for example:

```
.ADDTASK 256,5
```

The class operand is used only when user security is active on your computer. Each partition may have one class assigned to it. Only users that are allowed access to that class can access the partition after they sign on. The access keys to a partition that a user is restricted from accessing will not work for that user. See the Security chapter of this manual for more detail.

The startup file is a batch file that you must create prior to invoking the .ADDTASK command. The file must be located on the root directory of the boot drive. MOS will automatically invoke the batch file from the .ADDTASK command as a nested batch file. The startup batch file may contain any commands you want to automatically invoke for a partition.

For example, you may want to assign the system prompt in each partition to display the task number. In a multi-user environment, you may want to have the prompt display a user's name or initials or the task ID number.

The startup file may also contain a .PATH command for a specific user, or automatically bring up a menu on the video screen at a user's workstation. If security is active, the startup file must contain a .SIGNON command to bring up the partition with the signon screen.

For example, the startup batch file TASK.BAT might contain the following commands to customize the environment for that task:

## Multi-Tasking / Multi-User

### **TASK.BAT**

```
BATECHO OFF  
PATH=C:\C:\PCMOS  
PROMPT=[TASK $I] $P$G
```

**NOTE:** The \$I operand causes the task ID number of the task to appear in the prompt. This allows users to easily determine what task they are presently in.

Terminal ID's are required only in a multi-user environment when terminals or workstations are connected to the main console. MOS must know what driver to use for communicating with each terminal or workstation.

If a terminal or workstation does not function properly with one of the device drivers supplied with MOS, refer to the documentation that came with the workstation to identify and define the correct device driver.

MOS must also know which port a terminal is connected to, and the baud rate for communications. There is no default for these two operands. The \$SERIAL.SYS device driver that defines serial ports must be defined with a DEVICE= command statement in your CONFIG.SYS file.

For example, the following command:

```
ADDTASK 512,1,A,USER1,PCTERM,2,38400
```

would add task number 1 with a 512K memory size, security class A, and start-up batch file USER1.BAT. PCTERM.SYS would be the terminal driver used for communication. The terminal would be connected to port 2, the second port defined with your \$SERIAL.SYS statement. The baud rate for communications would be 38400.



Workstations differ slightly from terminals. The ADDTASK command to add a 600K task for a MaXtation SH-4/M workstation with the next sequential task ID number, a blank security class and the startup batch file TASK1.BAT might be:

```
ADDTASK 600,,,TASK1,SH,1
```

where SH.SYS is the device driver, and 1 represents the first MaXtation workstation.

**NOTE:** Notice that for workstations the "workstation number" replaces the "port" and "baud rate" operands normally used for terminals.

A user that is working in a partition may invoke the .ADDTASK command to add up to 99 other partitions, limited by available memory. After adding the partition, the user may use the partition access keys to access the new partition.

If user security is active, a user may also change the security class of the partition in which the user signed on, or may change the default output class that is assigned to any files or directories the user creates in that partition. See the .CLASS command in the security chapter of this manual.

## Multi-Tasking / Multi-User

### .REMTASK

.REMTASK lets you remove a memory partition from the system. You may use .REMTASK to remove partitions added from the AUTOEXEC.BAT file when you booted your computer, or partitions that you have added at the system prompt.

**Type:** Extrinsic.

**Form:** .REMTASK *task ID | {ALL}*

#### **Operand:**

*task ID* enter the task number of the partition to remove.

*ALL* entering ALL instead of a task ID number removes all tasks (except task 0)

#### **Explanation:**

You may use .REMTASK to remove any task to which you have unrestricted access, except partition 0. Partition 0 is absolute and may never be removed. To remove a partition with a task ID of 4, you would enter:

.REMTASK 4

Entering the following command will remove all tasks on the system (except task 0):

.REMTASK ALL

**NOTE:** The REMTASK ALL command can only be run from task 0.

When you remove a partition, the task ID of that partition is then available for assignment to another partition. If you then add a partition without specifying an ID, MOS will automatically assign the first task number available.



For example, you may have partitions 1, 2, 3 and 4 set up on your system. You use .REMTASK to remove partition 2. You then add another partition without specifying a partition number. The new partition then becomes partition 2.

You can remove a partition in which only one user is currently processing if you have unrestricted access to that partition. If more than one user is currently accessing a partition, MOS will not allow you to remove it, even if you have unrestricted access.

When you remove a partition, MOS closes all files that are open in that partition. If the partition you remove is running at a workstation, the workstation is rendered inactive. You can even remove a partition in which you're currently processing, but to do so will render your workstation inactive, and you will not be able to access any other partitions.

You may reactivate any inactive workstation by adding a partition for that workstation from the main console or from any other active workstation.

## Partition Access

Partition Access Method (PAM switching) allows a user to switch into any partition from the main console, or from any workstation. If partition level security is active, users may only access the tasks that they have the proper security class for.

### Partition Access Keys

The ALT key and the numbers on the numeric key pad are used to switch between tasks. To switch tasks, hold down the ALT key, type the number(s) of the task ID for the partition you want to access, and then release the ALT key. (The task ID numbers are assigned by the .ADDTASK command and can be viewed with the MOS MAP command.)

To access the partition assigned task ID 12, hold down the ALT key and type a 1 and then type a 2 on the numeric key pad, then release the ALT key, for example:

## Multi-Tasking / Multi-User

ALT 1 2

To return to the host partition (task 0) hold down the ALT key, type 0 on the numeric key pad, and then release the ALT key.

### .SWITCH Command

The .SWITCH command allows you to switch to a specific partition with a command line entry rather than with the ALT - number key Partition Access Method. The command form is:

.SWITCH {task ID}

where task ID is the number of the task to which you want to switch. This is helpful for use in batch files. If you do not specify a task number, you will be switched to the next task number in sequence.

### Turning Partition Access On and Off

When you first boot MOS, partition access is on. If the ALT and numeric key pad sequence conflicts with an application program's use of the keyboard, you may turn off partition access. To toggle partition access off, hold down the ALT key and type the nine on the numeric key pad three times. For example, press ALT 9 9 9 to turn off partition access. When you need to turn partition access back on, you may then press ALT 9 9 9 again.

### MOS Utility Functions

There are several utilities in MOS that you may find helpful in a multi-tasking or multi-user environment. There are two commands that invoke the utilities. The .MOS command provides several utilities that should be available to all users. The .MOSADM command allows several additional utilities that should be reserved for the system administrator.



## The .MOS Utility Command

You may display a list of the utility functions available in .MOS by entering .MOS at the system prompt. The display appears on your video screen similar to the following:

PC-MOS MOS UTILITY v#.##

Copyright 1991 The Software Link Incorporated.

All rights reserved worldwide.

Available functions are:

.MOS MAP	- display partition map
.MOS DIS NODIS	- disable [enable] keyboard ready looping
.MOS USEIRQ n [hhhh]	- reserve IRQ for application
.MOS FREEIRQ n [hhhh]	- free IRQ reserved by application
.MOS IRQ	- list IRQs reserved on system
.MOS WAIT event	- wait for event before continuing
.MOS VMODE mode	- set video mode for terminal
.MOS SERINIT n,r,p,b,s,h	- initialize a serial port
.MOS RESIZE nnnK	- adjust partition size
.MOS INFO	- displays memory allocation information
.MOS DSPORT n	- disables a port from \$SERIAL.SYS
.MOS FILES on off	- limit number of open files
.MOS TSR on off	- turn on off enhanced TSR support
.MOS KEYB AT PC EN SB LB	- defines keyboard type and buffer size
.MOS MOUSE n [,r] [,s]	- initialize mouse for a task
.MOS MOUSE off	- turn mouse support off
.MOS HOLD LPTx [nnnn]	- set printer reservation time
.MOS HOLD LPTx OFF	- release printer reservation for task
.MOS ROUTE LPTn [to] COMn LPTn TERM NOTERM	- redirect printer I/O to another device
.MOS ANSI on off	- turn on off ANSI support
.MOS DOSVER 3.2 3.3	- set DOS version level

These functions are invoked at the system prompt from within a partition by typing .MOS, followed by the function and any necessary operands. To invoke the MAP function, you would type .MOS, a space, MAP, and then press ENTER. For example:

.MOS MAP

# Multi-Tasking / Multi-User

Only the first three characters of a function are required to invoke the function. For example either form of the following will invoke the .MOS FREEIRQ function:

.MOS FRE 2 or .MOS FREEIRQ 2

A utility function is active only for the partition you are in when the function is invoked. If there is a function that you always want to have active for a partition, you may want to include it in the startup batch file for that partition.

The following explains each utility function along with any operands that must be entered with the function.

## .MOS MAP

The MAP function displays a map of statistical information about the partitions set up on your computer, and then returns control to MOS. You may not change any of the statistical information with this function. The map display is similar to the following:

PC-MOS USER TASK STATISTICS											
Task	Start	Size	Video	User	Program	Port	Baud	Pri	Slice	Files	Status
0*	18000	512K	CGA		MOS.COM	N/A	N/A	2	1	3	ACTIVE
1	18000	512K	MONO		COMMAND.COM	1	19200	2	1	2	ACTIVE
2	18000	512K	MONO		COMMAND.COM	N/A	N/A	2	1	2	WAIT

191K of 2048K Memory available      256K of 256K expanded memory available  
65K of 80K SMP allocated

An asterisk (\*) appears next to the task ID number from which the map was loaded. "Start" indicates the memory address where this partition starts in memory. "Size" is the memory size of the partition. "Video" displays the current video mode for each partition.

If a user has signed on to a partition with a user ID, it is displayed. "Program" displays the program currently being executed. The current port and baud rate appear in the next two columns if the task is a multiuser task for a terminal; if not, "N/A" is displayed.



"Pri" is the priority assigned to each partition. "Slice" is the number of time ticks assigned to each partition. "Files" displays the number of files currently open in each partition. "Status" displays whether the partition is active or in a waiting state.

The bottom of the display shows how much system memory is available, how much expanded memory has been assigned to the task if any, and how much memory has been assigned/used for the System Memory Pool (SMP).

#### .MOS DIS

The DIS function lets you disable anything running in a partition that constantly takes processor time just to check for keyboard input. The constant checking for keyboard input may take processing time away from other programs that actually require use of the processor.

Invoking the .MOS DIS function in a partition will speed throughput by freeing the processor until an actual keyboard request is issued.

Note that DIS cannot be used for all applications because they may be disabled when you don't want them to be. Some trial and error is needed to determine when DIS is appropriate.

#### .MOS NODIS

The NODIS function returns keyboard checking back to its normal state under MOS. This function is necessary only when a previous DIS function has been invoked.

#### .MOS USEIRQ n {hhhh}

An IRQ is a hardware interrupt line or vector. The USEIRQ command lets you reserve control of an interrupt vector or optionally a port specific interrupt vector, where "n" is the IRQ number and "hhhh" is a four digit hexadecimal number representing the address of a serial communications port.

## Multi-Tasking / Multi-User

Serial port COM1 normally uses IRQ 4 as its default, and COM2 normally uses IRQ 3 as its default. Some applications and hardware require the use of a specific interrupt vector, which may be 2, 5, 6 or 7. If a specific interrupt is required by the application or hardware product, it will be explained in the documentation that comes with the product. The valid IRQ numbers you may reserve are 2 through 7.

For example, many tape backup units require the use of IRQ 6, which is normally used by the system floppy disks. Before using such a tape backup unit you should reserve IRQ 6 with the MOS USEIRQ 6 command. When your backup is finished, enter the command MOS FREEIRQ 6 to return use of IRQ 6 to the floppy disks.

There are three different entities that can be associated with an IRQ reservation: a device driver, a task, and a task/port combination. Each has a different action and effect on the system. In the first two cases an IRQ is specifically reserved by a device driver or a task. In the last instance one IRQ may be split between more than one task based on the use of different serial ports.

When an IRQ is reserved by a device driver such as \$SERIAL.SYS, no other reservation can be made for that same IRQ, and the device driver will always receive control when that IRQ occurs.

When an IRQ is reserved by a task in a general, non-port specific manner, no other reservation can be made for that same IRQ. For example the command:

```
.MOS USEIRQ 3
```

entered in a task would mean that task will always receive control when an IRQ 3 occurs.

MOS will respond with the following message if the requested interrupt vector is available:

IRQ3: now reserved for application use.



If the interrupt vector is already assigned to another task or device, one of the following messages appears:

IRQ3: already in use by another task.  
IRQ3: already in use by a device driver.

When an IRQ is reserved by a task in a port specific manner, no other type of reservation may be made except another port specific one. For example, if the following command was entered in task 0:

.MOS USEIRQ 5 06A0

when an IRQ 5 occurs, MOS will test the serial port at address 06A0 to see if it caused the IRQ to occur. If it did then task 0 will receive control.

MOS will respond with the following message if the interrupt vector and port requested above are available:

Port specific IRQ reservation registered.

If you then tried to make a general reservation for IRQ 5 by entering:

.MOS USEIRQ 5

the following message would appear:

Duplicate entry in IRQ reservation

indicating that a port specific IRQ reservation was already on record.

## Multi-Tasking / Multi-User

Although you could make more than one port specific IRQ reservation in the same task, the typical case would be for separate tasks to share an IRQ based on different serial ports.

For example, if you now entered the following command in task 1:

```
.MOS USEIRQ 5 06A8
```

when an IRQ 5 occurs, if the serial port at address 06A8 caused the IRQ, then task 1 would receive control.

Using this method you could provide for multiple mouse support. Each mouse would communicate using the same IRQ but through a different port. (Note that your mouse software would have to be able to be configured for IRQs and ports that are available in your specific hardware environment.)

```
.MOS FREEIRQ n {hhhh}
```

The FREEIRQ function lets you release an interrupt vector when it is no longer required for use in a partition.

You can only release an interrupt vector while in the partition where it was originally reserved. For example, if the release of IRQ 3 is successful, the following message appears:

IRQ3: now freed from application use.

If the release is not successful, one of the following messages may appear:

IRQ3: cannot be freed because it is reserved by another task.  
IRQ3: cannot be freed because it is reserved by a device driver.  
IRQ3: cannot be freed because it is not reserved by any application.



In order to free a port specific IRQ reservation, the port must be specified along with the IRQ. For example:

.MOS FREEIRQ 5 06A0

If the release is successful, the following message will appear:

Port specific IRQ reservation removed.

If that port was not previously reserved, the following message will appear:

Port address not found in table.

.MOS IRQ

The IRQ function will display the status of the interrupt vectors and ports that are currently reserved on the system. For example:

IRQ	TASK	PORT
-----	-----	-----
2	--	-- (reserved by device driver)
3	0*	-- (non port specific reservation)
4	1	-- (non port specific reservation)
5	0*	06A0
5	1	06A8

If a device driver is using an IRQ there is no task or port association so no entry appears in those columns and the message: (reserved by device driver) appears to the right.

## Multi-Tasking / Multi-User

For general IRQ reservations, the port column has no entry and the message: (non port specific reservation) appears.

The asterisk by task number 0, indicates that task #0 is the task in which this IRQ listing is being done (the same as with the MOS MAP display).

Interrupt vectors that are still available for use will not appear in the display. From our example display you can tell that interrupt vectors 6 and 7 are available for use.

.MOS WAIT nnnnn or .MOS WAIT mm-dd-yy hh:mm:ss

The WAIT function lets you instruct MOS to wait for an event before continuing. The event may be a number of seconds (nnnnn) up to 99999, or a specific date and time that must be entered in the form mm-dd-yy hh:mm:ss. The date is optional and may be omitted to set only a specific time. The hh:mm:ss is the time known to the computer where hh is 0 to 24 for the hour, mm is 0 to 60 for the minutes, and ss is 0 to 60 for the seconds.

For example, you may want to set up a batch file that will export files to a back up device late in the evening when there is no one on the computer. You may include .MOS WAIT in a batch file with the .EXPORT command so that the export occurs at a specific time. You may invoke the batch file at the end of the day, and the export will be complete the next morning.

When the WAIT function is active, the following message displays on the video screen of the workstation where the function was invoked:

Wait Active...Press any key to abort

Any user may cancel the wait function by pressing any key. Aborting the WAIT function sets an error level one. If the WAIT function is processing from within a batch file, control returns to the next statement in the batch file. You may want to set the next statement in the batch file for errorlevel checking.



## .MOS VMODE mode

The VMODE function lets you change the video mode to meet the requirements of an application program. This function may be used at any workstation, and affects only the workstation where it is invoked. There are some restrictions on using certain modes as explained below. The values you may enter for mode are:

- MONO - monochrome card
- CGA - 80 column color graphics adapter
- C40 - 40 column color graphics adapter
- HG1 - 32K Hercules type 1 support
- HG2 - 64K Hercules type 2 support
- E43 - 43 Line EGA Main Console
- EGA - Multi-tasking EGA graphics - 80 column
- E40 - Multi-tasking EGA graphics - 40 column
- VGA - Multi-tasking VGA graphics - 80 column
- V40 - Multi-tasking VGA graphics - 40 column

If you run a program at a terminal and the program "sees" a graphics card, the terminal may attempt to use graphics mode. This will cause undesirable results if the terminal does not support graphics.

You may use the VMODE function at a workstation to make an application think you have monochrome card. For example:

### .MOS VMODE MONO

You may have a color monitor attached to the main console. Many terminals will not support color. If an application that runs in color causes problems with the video display at a workstation, you may need to change the mode at the workstation to monochrome before running the application that causes the problem.

HG1 and HG2 set the video mode to recognize and work with a hercules type video card.

E43 sets the video display for 43 lines in graphics mode instead of 25 lines. This mode will require at least 256K of memory in each task where it is set. The resolution of some graphics applications may require the 43 line display.

## Multi-Tasking / Multi-User

EGA, E40, VGA and V40 set the video mode to recognize a graphics adapter card, and will require at least 256K of memory in each task where they are set.

VGA and V40 are basically the same as the corresponding EGA modes, except that they use a new technology introduced with the IBM Personal Series/2.

Note that you may set up EGA and E40 on any computer, but you can only switch in and out of EGA tasks on an 80386-based computer or greater (80486, etc.). VGA and V40 can also be set up on any computer, but the 80386 requirement for task switching does not apply. Also, these four multi-tasking modes will only work at the main console, not at terminal workstations. When you switch out of any EGA or VGA task, any active processing in that task is suspended. The .MOS MAP display will show a status of "HOLD" for all suspended tasks.

.MOS SERINIT n,r,p,b,s,h

This utility lets you initialize a specific serial port by the port number defined with the \$SERIAL.SYS device driver, and define the attributes for the port as follows:

- n - the port number
- r - the baud rate for communications
- p - the parity may be n (none), e (even) or o (odd)
- b - the number of data bits
- s - the number of stop bits
- h - handshaking protocol

The handshaking "h" operand can be set to:

- N - none
- D - DTR
- X - XON/XOFF
- P - XPC
- R - RTS

These are the same handshaking protocols that are used with the \$SERIAL.SYS driver. See \$SERIAL.SYS in the Configuration chapter for more information.



For example, you may have a modem and need to initialize a serial port to set up the mode for communications. The command to initialize a port with its attributes could be as follows:

```
.MOS SERINIT 4,19200,n,8,1,d
```

This utility command will initialize port 4 at a baud rate of 19,200 with no parity, 8 data bits and 1 stop bit, and DTR handshaking..

```
.MOS ROUTE LPTn {to} COMn
```

This utility lets you redirect printer output that is sent to an LPTn device so that it will be sent to a COM port. This can be LPT1, LPT2 or LPT3. The COM port for redirection can be COM1 or COM2. This is useful if you have a serial printer.

For example, the following command will intercept any printer output sent to LPT1 and direct the output to COM2, where the printer is actually connected to the computer:

```
.MOS ROUTE LPT1 TO COM2
```

The "TO" in the this command is optional. You could also enter this command as:

```
.MOS ROUTE LPT1 COM2
```

```
.MOS ROUTE LPTn {to} LPTn
```

This utility lets you redirect printer output that is sent to an LPTn device to a different LPTn device. This is useful to redirect printer output that an application sends to LPT1 to a different print device (LPT2 or LPT3).

For example, the following command will intercept any printer output sent to LPT1 and direct the output to LPT2.

```
.MOS ROUTE LPT1 TO LPT2
```

The "TO" in the this command is optional. You could also enter this command with only a space between LPTn and LPTn.

## Multi-Tasking / Multi-User

.MOS ROUTE LPTn {to} TERM

This utility lets you redirect printer output that is sent to an LPTn device to a printer attached to your terminal. This is only true for most of the PCTERM terminal types. Your terminal must have a serial or parallel port for the printer connection. For example, if you have a printer connected to your terminal, the following command redirects output normally sent to LPT1 to the locally attached printer:

.MOS ROUTE LPT1 TO TERM

If a previous .ROUTE command has been invoked to redirect printer output to a COMn device, the COMn device will override the redirection to a terminal. The "TO" in this command is optional.

.MOS ROUTE LPT1 {to} NOTERM

This command will cancel a previous command that redirected output to a printer. For example, to cancel the redirection of printer output in the previous example, the command would be:

.MOS ROUTE LPT1 TO NOTERM

Any printer output for this partition will then be directed to the previously defined LPT1 device. The "TO" in this command is optional.

.MOS RESIZE nnnK

This utility lets you adjust the size of your partition at any time. The nnnK represents the size of the partition in kilobytes (K). This size may be from a minimum of 32K to a maximum of 640K, depending on your available memory. You must enter the K after the size to document that it is in kilobytes.

You may adjust the size of your partition at any time. For example, if you need to increase the size for a partition before you can run an application program, you could enter the following command:

.MOS RESIZE 256K

This utility will adjust the size of only the partition you are currently working in.



## .MOS INFO

The .MOS INFO display indicates specifically where in memory certain key components of the system are loaded. When you make a change in the memory ranges available for use by MOS with the FREEMEM command statement in your CONFIG.SYS file, this display will show you what has been able to relocate to high memory. This is useful when trying to maximize your application task size.

If memory management is present on the system, entering .MOS INFO at the system prompt will bring up a display similar to the following:

PC-MOS System Information	Start	End
FREEMEM-	C4000	E8000
FREEMEM-	ED000	EE000
MOS Kernel Segment #1	D4030	DFA30
MOS Kernel Segment #2	101000	10C470
System Memory Pool (SMP)	C4030	D4030
Disk Cache Descriptors	DFA60	DFAE0
Disk Cache	E4000	E5000
Command Processor	07050	0EA40
Master Video Context Area	E0000	E4000

If memory management is NOT used, a display similar to the following will appear:

PC-MOS System Information	Start	End
FREEMEM-	NO <NOT RELOCATED>	
MOS Kernel Segment #1	00700	0C100
MOS Kernel Segment #2	0C100	17570
System Memory Pool (SMP)	17570	27570
Disk Cache Descriptors	27580	27600
Disk Cache	27600	2B600
Command Processor	2B600	32FF0

## Multi-Tasking / Multi-User

With memory management, the location where the cache is mapped into low memory is listed in the display for "Disk Cache". With no memory management, the actual location of the cache is listed.

Additional information will be shown in the MOS INFO display if necessary to indicate which communication ports are being controlled by MOS's \$SERIAL.SYS device driver and which ports are being controlled by other serial drivers such as the SunRiver serial driver or the Maxspeed intelligent serial board driver. This information will appear at the bottom of the MOS INFO display, similar to the following:

PC-MOS System Information	Start	End
FREEMEM-	C4000	E8000
FREEMEM-	ED000	EE000
MOS Kernel Segment #1	D4030	DFA30
MOS Kernel Segment #2	101000	10C470
System Memory Pool (SMP)	C4030	D4030
Disk Cache Descriptors	DFA60	DFAE0
Disk Cache	E4000	E5000
Command Processor	07050	0EA40
Master Video Context Area	E0000	E4000
Communications Driver Description	First	Last
\$SERIALSYS Driver PC-MOS v3.10	COM1	COM3
SunRiver Serial Driver	COM4	COM11

In this example the first three COM ports are under control of the PC-MOS \$SERIAL.SYS driver. There are three PC-type terminals connected to the host computer through these ports. There are also four SunRiver workstations installed. Since there are two serial ports on each SunRiver workstation, the first SunRiver workstation's serial ports are therefore COM4 and COM5, the second SunRiver workstation's are COM6 and COM7, etc. You need to know these numbers when specifying the port number "n" with the MOS MOUSE command to initialize the mouse for the task.



Also note in this example that the FREEMEM statements conflict with the default location of the SRTERM.SYS terminal driver's workspace at E4000 to E8000. Either the FREEMEM statements would have to be changed to avoid using that address range, or the SRTERM.SYS driver's workspace would have to be located at a different address.

If an eight-port Maxspeed serial port board was installed for terminal connections instead of the SunRiver workstations, the eight ports on it would be COM4 through COM11.

.MOS DSSPORT n

Sharing a port between \$SERIAL.SYS control and use by a communications package can be accomplished with .MOS DSSPORT n.

When the following statement is used in the CONFIG.SYS file, 03f8 becomes logical port number 1 of the serial driver:

```
DEVICE=$SERIAL.SYS /AD=03f8,IN=4/AD=06a0,IN=3/AD=06a8
```

If you connect a modem to this port and set it up for auto answer and add a task for port 1, you would then have a multi-user environment in which a remote dumb terminal and modem could dial in to access the host machine.

However, if you occasionally wanted to use the modem from the master console with a communications package, such as CrossTalk, to call up a bulletin board service you couldn't. Since port 03f8 is under \$SERIAL.SYS's control it would not permit serial communications to work. You would have to edit your config.sys file, remove 03f8 from the \$SERIAL.SYS statement, and re-boot the machine. In fact, on some machines if you try to use CrossTalk without disabling the port, it will lock up the system.

To address this inconvenience, the .MOS DSPORT n command has been added to the MOS utility to tell \$SERIAL.SYS to temporarily disable a port in such a way that a communications package can use it. For port 1 the command would be:

```
MOS DSPORT 1
```

## Multi-Tasking / Multi-User

The requirements for successful use of this command are that \$SERIAL.SYS (or a similar serial communications driver) be installed, that the specified port number be a valid port, and that the irq associated with that port not be shared by any other ports defined with the serial driver. Also, the port must not be in use as the ADDTASK port of any task, i.e. the port that is used by a terminal which is the console of a task.

Once a port has been disabled with .MOS DSPORT it can be used again later for a dumb terminal workstation by initializing that task with the .ADDTASK command. When a call is made to \$SERIAL.SYS to initialize a port, as .ADDTASK does when a port is specified, \$SERIAL.SYS reverses the effect of the disable action. This could be done with either a local direct cabled workstation or a remote workstation via modem.

If you intend to use a port with a modem on it for a remote LANLink connection, you must re-initialize the port with .MOS SERINIT to reverse the effects of the disable command before running LAN ON.

### .MOS MOUSE

Once the mouse driver \$MOUSE.SYS (see the Configuration chapter) is active on your system you must initialize the mouse in the task in which it will run. This is done with the MOS utility command:

MOS MOUSE n[,r]

where:

n = the COM port number that the mouse is connected to. (1 for COM1, 2 for COM2, etc.) Can be obtained from the Communications Driver Description list in the MOS INFO display.

,r = the baud rate at which the mouse will communicate. Defaults to 1200. You must include the comma if you specify a baud rate.

For example:

MOS MOUSE 1,2400



---

will initialize a 2-button mouse on COM1 at 2400 baud for the task in which the command was entered.

The command can be entered manually in each partition or invoked automatically from a startup batch with the .ADDTASK command for each partition. A startup batch file for a partition is similar in function to the AUTOEXEC.BAT file on the host computer. See your MOS User Guide for more information on this command and on using startup batch files with the .ADDTASK command.

The MOS MOUSE command must be entered BEFORE you run the mouse program (MOUSE.COM) in your task to use the mouse.

Finally, enter the command "MOUSE" to run MOUSE.COM in the task to start the mouse session. (Note that with applications with their own mouse logic, such as Windows/286-based applications, you don't need to execute MOUSE.COM since it is loaded and executed automatically.)

For example, to install one mouse in the foreground task on the host computer using the first serial port (COM1), the following commands/statements would be necessary:

CONFIG.SYS file:

```
DEVICE=$SERIAL.SYS /AD=03F8,IN=4
DEVICE=$MOUSE.SYS
```

(Note that the \$MOUSE.SYS driver must be listed after all other serial drivers in the CONFIG.SYS file.)

AUTOEXEC.BAT file:

```
MOS MOUSE 1,2400
MOUSE
```

(This will initialize the mouse on the first serial port with a 2400 baud rate and then run MOUSE.COM to start the mouse session. The commands must be in this order.)

## Multi-Tasking / Multi-User

Remember that these are the additional commands and statements that are required to install the mouse. Others may already exist in either of these files.

The MOS utility command:

MOS MOUSE off

may be entered in a task if it is necessary to turn off mouse support in that partition. This might be necessary for some unusual application software. Once turned off, you must re-enter the MOS MOUSE command in the task to re-initialize the mouse for further use.

.MOS KEYB

In order to use one of the foreign keyboard drivers you must inform the system of the physical type of keyboard that is connected to the workstation. The keyboard type is defined with the following command:

MOS KEYB AT|PC|EN|SB|LB

where:

AT = AT-Style keyboard

PC = PC-Style keyboard

EN = Enhanced keyboard

SB = Small type-ahead Buffer

LB = Large type-ahead Buffer

The default condition is for the Enhanced keyboard. If, for example, an AT-Style keyboard was attached to the workstation you must enter the following command in that workstation's task:

MOS KEYB AT

The MOS KEYB command also includes "type-ahead" buffer size operands. The "LB", or large type-ahead buffer is the default condition. When "SB" is used, the type-ahead buffer that MOS uses is made small to match that of DOS.



The type ahead buffer allows you to start typing your next command while the current one is executing. However, under DOS, if you type more than 15 characters ahead, you will hear a beep. This is because the type ahead buffer is full.

Within a system which is running under DOS, this buffer is located at a standard memory address. Under MOS, the type ahead buffer was enlarged to 127 characters. In order to do this, the memory location of this buffer had to be changed.

Some applications programs set themselves up to interact with this buffer and have been designed to expect the buffer to be in its standard memory location. Since there is a defined way to tell where the buffer is, this is more of a weakness in the design of these applications rather than a problem with MOS.

In order to support such applications, MOS will need to be able to switch back to using the same SMALL type-ahead buffer that DOS does and use the same memory location. This is what the MOS KEYB SB command is for.

The following command should therefore be entered in the task where a small type-ahead buffer is required:

MOS KEYB SB

This change is task-specific and reversible (enter MOS KEYB LB).

#### Foreign Keyboard Device Drivers

The following device drivers can be used to remap or change the key assignments of the standard U.S. English keyboard to those of the required national keyboard layout:

\$KBBE.SYS - Belgium (Flemish)	\$KBNO.SYS - Norwegian
\$KBDK.SYS - Danish	\$KBPO.SYS - Portuguese
\$KBFR.SYS - French	\$KBSP.SYS - Spanish
\$KBCF.SYS - French Canadian	\$KBSV.SYS - Swedish
\$KBGR.SYS - German	\$KBSF.SYS - Swiss French
\$KBIT.SYS - Italian	\$KBSG.SYS - Swiss German
\$KBLA.SYS - Latin American	\$KBUK.SYS - United Kingdom
\$KBNL.SYS - Netherlands	

## Multi-Tasking / Multi-User

Enter the appropriate driver in your CONFIG.SYS file with a DEVICE= statement and reboot your computer to install the new keyboard driver. For example:

DEVICE=\$KBFR.SYS

If it is necessary to switch back to the U.S. English keyboard layout for some reason, press the CTRL - ALT - F1 keys at the same time. When you are ready to return to the installed foreign keyboard layout press the CTRL - ALT - F2 keys simultaneously. (Note that these are system drivers, not task-specific drivers.)

.MOS FILES on | off

The MOS utility command MOS FILES is used to place an upper limit (20) on the number of files that may be opened in the task in which the command is entered. The default is "off" since PC-MOS itself does not set any limit on the number of files that may be opened on the entire system. (A present application package that requires the use of this command is ENABLE v3.0.)

Enter the following command to set this limit:

MOS FILES on

To remove the limit, once set, enter:

MOS FILES off

MOS HOLD LPTx {nnnn} | OFF

The "MOS HOLD LPTx nnnn" command can be used to automatically reserve a printer for use by a specific user for a specified length of time. "x" specifies the printer, and "nnnn" specifies the reservation time in seconds. The maximum is 9999 seconds (about 166 minutes or 2 & 3/4 hours).

This automatic reservation prevents other users from using the printer for the specified time after the reserved user has accessed it. This scheme allows both direct printing and spooled printing on the system.



If a user accesses a printer that is not in use by any other task (by printing or checking the status), the printer will be reserved for that user for 15 seconds. Any other user will receive a busy status from that printer while it is reserved. The time between access and release of the device (reservation time) can be set with the "nnnn" operand.

For example, entering:

```
MOS HOLD LPT1 60
```

only sets the reservation time for the LPT1 print device to 60 seconds. It does not actually access the printer, and therefore, does not reserve the printer.

Entering:

```
MOS HOLD LPT1
```

will then access the LPT1 print device and display the results of the access and reservation.

Entering:

```
MOS HOLD LPT1 OFF
```

will release any reservation of the LPT1 print device.

Also see the "MOSADM HOLD task LPTx nnnn" command for the system administrator. It is similar in function except it can set the reservation time or release a printer for another task.

.MOS TSR on | off

The MOS TSR on command can be used to improve the performance of some applications, particularly certain Terminate and Stay Resident (TSR) programs. If an application appears to pause/not process for periods of time, or a TSR program will not pop-up reliably, enter the following command at the system prompt:

```
MOS TSR on
```

## Multi-Tasking / Multi-User

To remove the utility, enter:

MOS TSR off

MOS ANSI on | off

This command is used to turn on or off the emulation of ANSI.SYS. With ANSI support turned off terminal screen handling is much improved. The default state of the ANSI emulation support is ON.

MOS DOSVER 3.2|3.3

This command is used to "tell" applications what version of DOS they are running under. Since you are running PC-MOS you are actually fooling the application into thinking it is running under a certain version of DOS. This is necessary for some application to run properly. For example, Microsoft Windows/286 v2.10 and 2.11 require DOS version 3.2 to run properly.

The form for the command is:

- |                |  |
|----------------|--|
| MOS DOSVER     | - Displays the current DOS version supported.    |
| MOS DOSVER 3.2 | - Sets DOS version supported to 3.2<br>(Default) |
| MOS DOSVER 3.3 | - Sets DOS version supported to 3.3              |



## The .MOSADM Utility Command

The .MOSADM command provides several additional utilities that you may not want to make available to all users. You may assign the .MOSADM.COM file a security class to allow only the system administrator access to these utilities.

You may display a list of the utility functions available in .MOSADM by entering .MOSADM at the system prompt. The display appears on your video screen similar to the following:

```
PC-MOS MOS ADMINISTRATOR UTILITY v#,##  
Copyright 1991 The Software Link Incorporated.  
All rights reserved worldwide.
```

### Available functions are:

.MOSADM SLICE n [task]	- set partition time slice to n
.MOSADM PRI n [task]	- set partition priority to n
.MOSADM CACHE on off	- turns disk cache on and off
.MOSADM SWITCH on off [task]	- turn ability to switch between partitions on off
.MOSADM TMFACTOR [n]	- set/show system time slice factor
.MOSADM HOLD task LPTx [nnnn]	- sets printer reserve time for task
.MOSADM HOLD task OFF	- releases printer reservation for task
.MOSADM VIRQ on off	- define IRQ task switching method
.MOSADM RESET on   off	- controls task restart logic
.MOSADM TIME on   off	- controls timer chip I/O protection
.MOSADM EMSLIMIT size [handles]	- controls task EMS size and handles
.MOSADM EMSLIMIT OFF	- turns off EMS emulation in task

These utility functions are invoked at the system prompt from within any partition by typing .MOSADM, followed by the function and any necessary operands. As in the .MOS command, you may invoke any of these functions by entering only their first three characters.

## Multi-Tasking / Multi-User

.MOSADM SLICE n {task}

The SLICE command statement in the CONFIG.SYS file instructs MOS to give each partition in turn an equal share of processing time. The default for SLICE is one time tick, or approximately 1/18 of a second of processing time for each partition in turn.

The .MOSADM SLICE utility lets you give a partition more of the processor's time by increasing the number of time ticks the partition receives during its turn. The system administrator may include the task ID number to change the number of time ticks for a specific partition. If you do not enter a task ID number, the number of time ticks will be changed for the current partition. For example, you may issue the following utility function while in a partition:

.MOSADM SLICE 3

MOS will now give this partition 3/18 of a second processing time during its turn. Because the partition has more of the processor's time, any processing that occurs in the partition will be faster, but the processing in other partitions will be slower.

You may also issue the following utility function to change the number of time ticks for a specific partition.

.MOSADM SLI 3 12

MOS will now give the partition assigned task ID number 12, 3/18 of a second processing time during its turn.

The maximum number of time ticks you may define with the SLICE utility is 255, however this is not a practical limit. A practical limit for assigning time ticks is 4. Note that if you enter 0 for the number of time ticks, it will disable the partition.



.MOSADM PRI n {task}

The .MOSADM PRI (priority) function also lets you control the amount of processing time for a partition, but is absolute where SLICE is relative. Each partition has a priority of 2 when you boot MOS. The PRI function lets you change the priority level for a partition to a number in the range of 0 to 7, with 7 being the highest priority.

The system administrator may include the task ID number to change the priority for a specific partition. If you do not enter a task ID number, the priority number will be changed for the current partition. For example:

.MOSADM PRI 7

This command will change the priority for the current partition. The following command will change the priority to 5 for the partition assigned task ID number 2:

.MOSADM PRI 5 2

The partition(s) with the highest priority will always get processing time before any partition(s) with a lower priority.

In real practice, a word processor running in a priority 7 partition will be given all the processing time until it no longer needs it. Processing time is then passed on to lower priorities.

If the partition with a priority of 7 is processing a program that requires continual access of the processor, no other partitions can do any processing. In effect, processing stops in other partitions. As soon as the partition with a priority of 7 releases the processor, MOS returns to giving the other partitions their time ticks for processing in turn.

## Multi-Tasking / Multi-User

.MOSADM CACHE on | off

The CACHE configuration command statement sets up a cache of memory for storing and retrieving data. A CACHE= statement must be in your CONFIG.SYS file for disk caching to be active on the system. You may then use the .MOSADM utility command to turn disk caching on or off for the entire system. For example, you may turn off disk caching with the following command:

.MOSADM CACHE OFF

Turning off disk caching will cause everything in the CACHE to be written to disk. This is a very good way to make sure that everything has been physically written to disk before turning the system off.

If you later want to turn disk caching back on, you may enter the following command:

.MOSADM CACHE ON

.MOSADM SWITCH on | off {task}

The .MOSADM SWITCH utility lets you turn on and off the capability to switch in and out of a partition (task). You may use this feature for security purposes, or for better handling of video cards that are not currently supported under PC-MOS. If you do not specify a task number, the command is invoked for the current task.

.MOSADM TMFACTOR {n}

The TMFACTOR utility lets you adjust MOS's time slicing rate by changing the value of one time tick to an amount less than the default of 1/18 of a second. The command may be entered when tasks are busy. This utility causes MOS's time slicing rate to be increased by a factor of "n".

For example, entering the command:

MOSADM TMFACTOR 2



will change the value of one time tick to approximately 1/36 of a second (1/18 divided by 2). This means that each task will in turn receive 1/36 of a second of processing time rather than 1/18 of a second. In other words, MOS will switch tasks twice as often.

The effect of increasing the time slicing rate is to improve the visual smoothness of scrolling displays and the response time between entering keystrokes and seeing the data appear on the screen.

**NOTE:** Use the lowest value for "n" that gives satisfactory results since using a larger multiplier than necessary will introduce extra system overhead and reduce the overall system throughput. Values for "n" in the range of 1 to 20 should work best in most cases. The maximum allowable value is 40.

Entering MOSADM TMFACTOR without any number will cause the current value to be displayed, for example:

Time slicing factor = 2

To reset the time slicing rate back to the default value, enter:

MOSADM TMFACTOR 1

This returns the value of one time tick to approximately 1/18 of a second, and each task will in turn receive 1/18 of a second of processing time.

.MOSADM HOLD task LPTx {nnnn} | OFF

The "MOS HOLD task LPTx nnnn" command can be used to automatically reserve a printer for use by a specific user for a specified length of time. "task" specifies the task number for which the reservation will be made. "x" specifies the printer, and "nnnn" specifies the reservation time in seconds. The maximum is 9999 seconds (about 166 minutes or 2 & 3/4 hours).

## Multi-Tasking / Multi-User

This command is similar in function to the MOS HOLD LPTn {nnnn} command except that it can set the reservation time or release a printer for any task. See the "MOS HOLD LPTx {nnnn}" command in this chapter for information on how to use this command.

### .MOSADM VIRQ on | off

The MOSADM VIRQ utility command changes the way asynchronous task switches are done in order to provide different types of mouse support and prevent a PAMswitch from interfering with communications.

When multiple mice are supported through the INT 14 method using \$MOUSE.SYS only the task RAM and context area are switched and there is no need for the PAMswitching process to disable interrupts. As a result, IRQ's are not lost. Therefore, if your system will involve the use of task-based IRQ handlers (e.g. telecommunications programs such as CrossTalk and ProComm) you should use the default condition (MOSADM VIRQ OFF) and interface any mice through the INT 14 method.

Patched mouse driver type of multiple mouse support (such as that under VNA using MOS USEIRQ and SETMOUSE.COM) requires the switching of video buffers during a task switch. If you require this type of multiple mouse support you should issue the command:

### MOSADM VIRQ ON

This will effect the changes to the task switching procedures required for patched mouse driver multiple mouse support.

### MOSADM RESET on | off

This command allows the system administrator to turn on and off the CTRL - ALT - DEL task restart option at the workstations. The command form is:

### MOSADM RESET on|off {task#}

where {task#} is the number of the task for which the option will be set.



## MOSADM TIME on | off

This command allows the system administrator to turn on and off I/O protection of the timer chip. The default condition is ON. It is recommended that you enter the MOSADM TIME OFF command when your application uses the timer chip to control the speaker and the sound is too slow or not right. One example is when using the FIRE PHASER command when logging onto a Novell server.

## MOSADM EMSLIMIT size {handles} | OFF

This command is used to add LIM EMS 3.2 or 4.0 memory to tasks that will run applications that need expanded memory. The \$EMS.SYS device driver must be loaded in your CONFIG.SYS file in order to use this command.

The command has two forms as follows:

**Form:**      MOSADM EMSLIMIT sizeK {handles}

                  MOSADM EMSLIMIT OFF

### Operands:

**size**      enter the amount of extended memory in kilobytes to allocate for the task. This may be from 64K to a maximum of 8192K (8MB) per task (limited by the amount of available memory in your computer).

**K**      enter a K to document that the memory size is in kilobytes.

**handles**      defaults to 32 handles. This is the number of blocks of memory required by an application. 32 is sufficient for most applications. If your application requires more handles, enter the number of handles required as specified in your application's user manual. Handles may be set to a value from 16 to 255.

**off**      If you enter OFF as the only operand with the command, the amount of EMS memory previously set for the task will be de-allocated and returned to available system extended memory.

## Multi-Tasking / Multi-User

The MOSADM EMSLIMIT command is used to reserve a specified amount of EMS expanded memory for an application. The command is task specific. That is, it must be entered in each task that will run an application that requires EMS memory. The amount of memory reserved with the command may be different for each task that it is entered in.

For example, if you were going to run Ventura Publisher 3.0 in task 1 on your system, you would first switch into task 1 by pressing ALT - 1 and then enter the following command at the system prompt:

```
MOSADM EMSLIMIT 256K
```

This would assign 256K of expanded memory to task 1 which can then be used by Ventura or any other application that is run in that task.

If you had another application that required 512K of EMS memory and 64 handles, that you intended to run in task 0, you would proceed as follows. First press ALT - 0 to switch back into task 0, and then enter the following command at the system prompt:

```
MOSADM EMSLIMIT 512K 64
```

The MOS MAP command display shows how much EMS expanded memory is assigned/available in each task. Remember that the amount of EMS memory assigned to each task can be different - so you must enter the MOS MAP command in each task to see how much EMS expanded memory has been assigned to the various tasks. (The amount of system extended memory will always display the same amount regardless of which task you enter the MOS MAP command in.)

For example, if you enter the MOS MAP command in task 1, the last two lines of the MOS MAP display might be:

2196K of 4096K Memory Available

256K of 256K Expanded Memory Available

58K of 80K SMP Allocated



In this example the total amount of memory available on the system is 4096K, or 4MB. Presently, 2196K of that is still available as system extended memory. Also, 256K of EMS expanded memory has been assigned to task 1, the task in which this MOS MAP command was just entered. All of the EMS memory is available since we had to exit the application to enter the MOS MAP command.

If you now enter the following command in task 1:

MOSADM EMSLIMIT OFF

the 256K of extended memory will be de-allocated and returned to available system extended memory. This can be verified by entering the MOS MAP command in task 1 again, which will now display:

2452K of 4096K Memory Available

58K of 80K SMP Allocated

Notice that the available system extended memory increased by 256K to 2452K and that the EMS expanded memory portion of the display no longer shows at all, indicating that no EMS expanded memory is assigned to task 1.

#### Task Reboot with EMS memory

Pressing CTRL - ALT - DEL in your task performs a warm reboot of JUST that task. Any EMS memory assigned to that task will be de-allocated. You must then re-enter the MOSADM EMSLIMIT command in the task to re-assign the amount of EMS expanded memory needed. Placing the MOSADM EMSLIMIT command in the startup batch file for the task will eliminate this inconvenience.

# Multi-Tasking / Multi-User

---

## Connecting Terminals and Workstations

In a multi-user environment, there are hardware considerations for workstation support. You must be sure you have the correct terminal or workstation device driver, serial port device driver, and the right cable connections for the host computer and workstation or terminal to communicate.

You must also be sure any workstation options are set up correctly for use under MOS. Some terminals may require the use of ESC key sequences for communication with the processor. The next sections explain the necessary details for the proper installation and set up of terminals and workstations.

### Terminals vs Workstations

Terminals are normally connected to the host computer through serial ports on the computer or on add-in serial port boards that you can install in your computer. The connection and use of terminals are explained in detail in the following sections.

Workstations are different than terminals in some important ways. Workstations normally consist of regular PC monitors and PC keyboards that are connected to the host computer through some special hardware designed specifically for that purpose. These hardware based workstation solutions offer increased speed over terminals and also the capability of supporting graphics applications at the workstations.

The special hardware usually consists of an add-in expansion board that installs in your computer plus some additional external junction boxes and necessary cabling to connect the workstations to the host computer.

Be sure to follow the installation instructions that come with your workstation hardware when adding workstations to a PC-MOS host computer.



## Serial Ports

Terminals are connected to a serial port on the host computer or on an add-in serial port board that you have installed in your computer. This connection may be directly to the host computer, or indirectly with a modem.

The serial ports are normally the standard type that uses the NSI 8250 UART or equivalent. MOS is also capable of supporting a variety of workstation communication links. You must use a driver that replaces \$SERIAL.SYS for the communication link. Special ports that have synchronous communications capability may not be compatible.

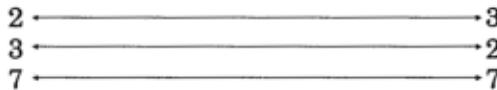
If your computer does not have enough serial ports to support the number of terminals you want to connect, you may need a serial adapter board that will support multiple serial ports. Such a board is available from your dealer.

The \$SERIAL.SYS device driver provided with MOS must be defined with a DEVICE= command statement in your CONFIG.SYS file. This driver identifies the address assignment of each port, and supplies MOS with specific information about the communications capability of the port.

## Cable Connections

To connect a terminal directly to the host computer, you need a "null modem" or "crossover" cable. You may use a standard cable if you attach a null modem adapter to one end of the cable. If you want to make your own cable, you need three wires. The pin connections for these wires are:

Host Computer Pin  
DB-25 Connector

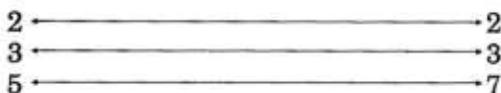


Terminal Pin  
DB-25 Connector

## Multi-Tasking / Multi-User

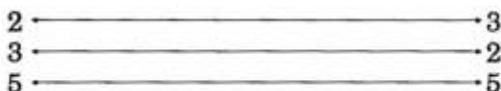
Host Computer Pin  
**DB-9 Connector**

Terminal Pin  
**DB-25 Connector**



Host Computer Pin  
**DB-9 Connector**

Terminal Pin  
**DB-9 Connector**

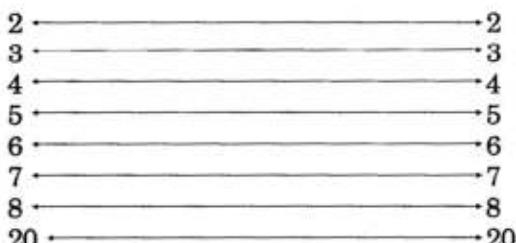


### Modem Cable Connections

You may also need to run a cable to a modem from either the host computer or a terminal. For either of these, use a standard modem cable including the following pin connections.

Main Console or  
Terminal Pin

Modem Pin





## Terminal and Workstation Device Drivers

MOS supplies the terminal device drivers for most standard terminals. All terminal device drivers that you use must be set up and defined in the CONFIG.SYS file with the DEVICE= command statement. The terminal device drivers included with MOS are:

PCTERM.SYS - for PC type terminals, Ampex 232, Falco 5500, Kimtron KT-7 PC and KT-70, Link Technologies MC1, MC3, MC5 and PCTerm, TeleVideo PCS1, Wyse Wy-60 and Wy-150.

VNA.SYS - for Video Network Adapter (VNA) workstations

SH.SYS - for MaXtation workstations

TTTERM.SYS - for Teletype type terminals

AVTERM.SYS - for ADDS Viewpoint type terminals

3ATERM.SYS - ADM 3A

T1TERM.SYS - Televideo 910

52TERM.SYS - DEC VT-52

31TERM.SYS - IBM 3101

19TERM.SYS - Zenith Z-19

TVTERM.SYS - Televideo 912C, 920C, 925

EXTERM.SYS - Excel 42/44

ANTERM.SYS - ANSI terminals

ATTERM.SYS - AT&T 605 Business Communications Terminal

ELTERM.SYS - for EmuLink Terminal Emulation Software.

SRTERM.SYS - SunRiver Fiber Optic Workstation

## Multi-Tasking / Multi-User

These terminal device drivers must be defined in the CONFIG.SYS file with the DEVICE= command statement before you can include them in an .ADDTASK command.

PCTERM.SYS supports almost all of the attributes of an IBM monochrome monitor. AVTERM.SYS supports inverse video only. Sound is not supported by any of the terminal device drivers.

There may be other terminals that are not included with the MOS standard terminal drivers that will also work with MOS.

The basic requirements for any workstation terminal are:

- 24 line by 80 column screen or larger
- ability to directly position the cursor
- ability to erase from cursor to end of screen
- scrolling capability
- control codes to move the cursor up, down, right and left

Most terminals have an array of switch settings, and some combination of hardware and keyboard selected options. You normally have a choice of option settings when you install a terminal workstation. You should review these options in the documentation that comes with the terminal to be sure it conforms to the following requirements.

- The total number of data, start, stop and parity bits must add up to 10.
- Configure the terminal for full duplex operation, using either 8 data bits and 1 stop bit, or 7 data bits and 2 stop bits.
- If given the option, select not to use parity checking. If a terminal requires a parity bit, then select either "space" or "mark" parity with 7 data bits and 1 stop bit.

Do not use any special features of a terminal, such as smooth scrolling or automatic generation of line feeds. If given the option, make sure that the terminal will automatically scroll up when the cursor moves down from the bottom line.



## Terminal Display Differences

The video screen on the main console has a 25 line display. The video screen on some terminals may have only a 24 line display. With applications that require the use of 25 lines, you may need to scroll the video screen on the terminal up or down to display the first or last line of a screen.

For example, some applications use the 25th line as a control line. While you are working at a terminal, only lines 1 through 24 appear on the video screen.

When you need to see the 25th line, you must scroll the screen up so that line 1 no longer displays, but line 25 does. You may then scroll the screen back down so that lines 1 through 24 again display.

Some keys on a terminal may not be immediately usable with MOS. Directly supported are the character keys (ASCII character set) ESC, SHIFT, CTRL, BKSP, ENTER, cursor movement (arrow) keys, and if present, the HOME key.

MOS provides escape sequences that emulate certain PC-style keys at non-PC style terminals. This allows the terminal to emulate all the keys on the main console.

# Multi-Tasking / Multi-User

## Escape Sequences for Non-PC Type Terminals

Escape sequences are invoked by pressing the ESC key followed by pressing one or more other keys. The following explains the ESC sequences you may use at a terminal to emulate scan codes for the key strokes from the main console. Following the detailed explanations is a quick reference chart listing only the ESC sequences and resulting keystrokes that are emulated.

### **ESC T**

Press ESC and then a T to refresh the first 24 lines of the display on the video screen of a terminal. The main console generally has a 25 line display, where some terminals have only a 24 line display. At this point, line 25 does not appear on the video screen of a terminal.

**NOTE:** If the cursor belongs on line 25 when you press **ESC T**, then it will be placed in the same column on line 24 of the terminal, and anything you type will move the cursor, but will not display until you press **ENTER**.

### **ESC B**

Press ESC and then a B to refresh the bottom 24 lines (lines 2 through 25) of the display on the video screen of a terminal. If the cursor belongs on line 1, the it will appear on line 2 of the current screen.

If you turn on the terminal after MOS has been booted, use **ESC T** or **ESC B** to refresh the display on the video screen.

### **ESC C**

Press **ESC** and **C** to turn on use of CTRL-key combinations sent to the processor. Otherwise, CTRL-key combinations can be confused with the same codes sent by the **HOME** and cursor movement keys. This is necessary for programs that use CTRL-key combination for special functions, such as many word processors. Turning on the CTRL-key combinations will disable the **HOME** and cursor movement keys.



## ESC N

Press **ESC** and **N** to turn off CTRL-key combinations and enable normal use of the HOME and cursor movement keys.

## ESC A char

Press **ESC** and **A** to send a scan code emulation for the ALT key, then type a character which may be a letter (A to Z), a digit (0 to 9), a hyphen, or an equal sign. For example, to emulate **ALT X** from a terminal, press **ESC A X**.

## ESC S H/E/U/D/I/X/P

Press **ESC** and **S** followed by one of the letters defined to send a scan code emulation for several special keys found on the numeric keypad on the main console's keyboard. The emulation sent for each defined letter is: **H** - HOME, **E** - END, **U** - PgUp, **D** - PgDn, **I** - INS, **X** - DEL, and **P** - SHIFT PrtSc. For example, to send a PgUp scan code emulation from a terminal, press **ESC S U**.

## ESC Z H/E/U/D

Press **ESC** and **Z** followed by one of the defined letters to send a scan code emulation for several CTRL-key combinations. The emulation sent for each defined letter is: **H** - CTRL HOME, **E** - CTRL END, **U** - CTRL PgUp, **D** - CTRL PgDn. For example, to send a CTRL PgUp scan code emulation from a terminal, press **ESC Z U**.

## ESC n

Press **ESC** and a number between 0 and 9 to emulate the scan code for the function keys F1 to F10 on the main console. **ESC 0** translates to F10. For example to send an F3 scan code emulation from a terminal, press **ESC 3**.

## Multi-Tasking / Multi-User

### **ESC F nn**

Press **ESC** and **F** followed by two numbers between 0 and 9 to emulate the scan code for the function keys F11 to F40 on the main console. At the main console, **SHIFT F1** sends a scan code for F11, **CTRL F1** sends a scan code for F21, and **ALT F1** sends a scan code for F31. To emulate **ALT F1** from a terminal, press **ESC F 3 1**.

### **ESC K**

Press **ESC** and **K** to send the scan code emulation from a terminal for the **CTRL BREAK** key function on the main console. Use of the break key may be disabled when running some application programs.

### **ESC M nn**

Press **ESC** and **M** followed by up to two numbers between 0 and 9, then press **ESC** and **M** again. This switches you into the new task identified by the numbers you entered.

### **ESC ESC**

Press **ESC** twice at a terminal to send the emulation scan code for the **ESC** key (hexadecimal code 1B) on the main console.

**NOTE:** If you are using the DEC VT-52, IBM 3101, or Zenith Z-19, do not use the **ESC** key. Instead, use the key combination that generates the hex code 1C. On the VT-52 or VT-100, this would be **CTRL** and backslash (**CTRL \**), on the 3101 (US keyboard) use **ALT** and equals (**ALT =**).



## Escape Sequence Chart for Non-PC Type Terminals

First Key	Second Key	Third Key	Fourth Key	<u>Key Emulated</u>
ESC	T			refresh top 24 lines on screen
ESC	B			refresh bottom 24 lines on screen
ESC	C			enable CTRL key combinations; disable use of HOME and cursor movement keys
ESC	N			disable CTRL key combinations; enable use of HOME and cursor movement keys
ESC	A	A-Z		ALT key combinations
ESC	A	0-9		ALT key combinations
ESC	A	-		ALT key combinations
ESC	A	=		ALT key combinations
ESC	S	H		HOME key
ESC	S	E		END key
ESC	S	U		PgUp key
ESC	S	D		PgDn key
ESC	S	I		INS key
ESC	S	N		Number Lock Key
ESC	S	X		DEL key
ESC	S	P		SHIFT-PrtSc
ESC	Z	H		CTRL-HOME
ESC	Z	E		CTRL-END
ESC	Z	U		CTRL-PgUp
ESC	Z	D		CTRL-PgDn
ESC	0-9			function keys F1-F10 (ESC 0 for F10)
ESC	F	1-4	0-9	function keys F11-F40
ESC	K			CTRL-BREAK
ESC	ESC			ESC
ESC	M	0-9	0-9	Toggle Task Switch

# Multi-Tasking / Multi-User

## .KEYMAP

.KEYMAP is an extrinsic command that lets you assign strings of data to selected keys. You may save the key assignments in a file so they may be used again at a later time. .KEYMAP lets you assign an escape sequence to a specific key, or create your own macros. For example, if a word processor uses CTRL K D to save a file, you could assign the string CTRL K D to the F1 key. The following is an example of a key assignment session that is invoked by entering .KEYMAP at the system prompt:

Enter input filename (Return if none): ENTER

You may enter the name of an existing key assignment file to use as initial input, or press ENTER to start a new file or add to the existing assignments that are in use but not saved in a file.

Translate from? \*

Enter up to four key strokes as the new key and press ENTER. If possible, use only one key stroke. If you enter multiple keys, enter them rapidly or they will not translate. Non-ASCII keys display on your video screen as an asterisk (\*), such as in this example for the F1 key. If the ENTER key is part of the string, enter \$ R, which will appear as a slash (/). You may enter \$ \$ to include a \$ as part of the string.

Translate to? CTRL K D

Enter the string, which has no practical limit, and press ENTER. If the key has been assigned to a different string, you will be asked if you wish to replace it, and you may enter either Y or N. A prompt "Is the above entry correct?" then appears. Enter Y if the assignment is correct. If not, enter N and you return to the MOS system prompt.

Translate from?

You may continue to enter new key assignments, or press ENTER without an entry to end the .KEYMAP session. A prompt, "All finished?" will appear, and you may enter either Y or N. A response of N returns you the "Translate from?" prompt.

Enter output file name (Return if none): SAMPLE.KEY



You may enter the name of a file to store your assignments, or press ENTER if you do not want to save the assignments in a file. You should use a file extension of .KEY to help you identify these files. If you save the assignments, you may recall the file to use the key assignments at a later time. For example, the command to use the assignments in the SAMPLE.KEY file is:

```
.KEYMAP SAMPLE.KEY
```

If you have more than one file of key assignments, always load the largest one first. If you load a small file and then load a larger file, the smaller file is overwritten. Depending on the number of key assignments you set up, the .KEYMAP function will use a small amount of memory in a partition, usually about 2K.

You may find it useful to create an empty key assignment file for removing key assignments in a partition. You may use the .KEYMAP command and enter only the output file name, such as NULL.KEY, to create the empty key assignment file. You may load the NULL.KEY file to remove the current key assignments in a partition.

# Multi-Tasking / Multi-User

## Modems

MOS will support a remote workstation via the use of modems. Since modems use the phone lines, you can have a workstation as far away as you can make a long distance phone call. The connection is the same as any other workstation, except that there are two modems and a telephone line as part of the connection.

The .ADDTASK command should be the same as for any other remote workstation. Include the terminal device driver, the port number where the modem cable is connected (based on the order it is listed in the DEVICE= \$SERIAL.SYS statement in your CONFIG.SYS file), and the baud rate.

## MODEM.COM - Modem Initialization Utility

This program allows you to initialize a modem for incoming remote terminal communications or remote LAN connections under PC-MOS. This must be done BEFORE you add the task that controls the modem. This could be done in your AUTOEXEC.BAT file by entering the appropriate command before the associated ADDTASK command.

To initialize a modem with MODEM.COM log onto the drive and directory which contains the MODEM.COM program and enter the following command at the system prompt:

MODEM x y

where "x" is the port number from the ADDTASK command for the partition that controls the modem, and "y" is the modem type from the following table:

- 0 - for Hayes 300 baud
- 1 - for Hayes 1200 baud
- 2 - for Hayes 2400 baud
- 3 - for Hayes 9600 baud



For example, the commands:

```
MODEM 1 2  
ADDTASK 384,1,,USER1,PCTERM,1,2400
```

would initialize the first port listed in your DEVICE=\$SERIAL.SYS statement to a Hayes 2400 baud modem. The modem would then be active in the partition activated by the ADDTASK command which specifies port 1 as the serial port for remote communication.

#### Defining Modem Types

MODEM.COM can also be used to edit the modem initialization strings and/or define six additional modem types.

To edit or define modem types log onto the drive and directory which contains the MODEM.COM program and enter the MODEM command at the system prompt with no operands, as follows:

```
MODEM
```

A menu will appear that shows you the proper syntax for using the MODEM command to initialize a modem plus the existing modem types, followed by:

Enter 1 to edit modem initialization strings.  
Enter 2 to return to MOS.

Press 1 at that time to bring up the modem type editing menu.

Then use the UP or DOWN arrow keys to position the cursor on the modem type you wish to edit/create, and press ENTER. Key in the new modem speed and initialization string, pressing ENTER after each. Repeat this process for each modem type that you want to define.

When you finish your edits, press F3 to save the information and then press ESC to exit, or you may press ESC first to exit without saving the changes.

## Multi-Tasking / Multi-User

### SunRiver Terminal Driver

PC-MOS supports the SunRiver Fiber Optic Workstation. The terminal driver SRTERM.SYS is provided for this purpose. This driver can only be used in a memory-managed environment.

Follow the instructions provided with your SunRiver terminal to set up and install the associated hardware. It is recommended that you set the SunRiver host adaptor card to operate on interrupt level 5 or 7 when used with PC-MOS. If one of these two levels is not available, try interrupt level 12 or 15 next.

The terminal driver must be defined in your CONFIG.SYS file with a DEVICE= command statement, as follows:

```
DEVICE=SRTERM.SYS E4000
```

The only optional operand is the memory address for the driver's workspace. The default address is E4000. (The driver actually loads into the SMP, while the memory space that it requires out of the FREEMEM area is a workspace for the driver.) This workspace takes up 16K of memory, e.g. in this case the driver's workspace loads in memory from E4000 to E8000. (Remember to avoid using this address space when specifying your free memory with the FREEMEM= statement.)

Once the DEVICE= statement is set up in your CONFIG.SYS file, you may use SRTERM as the terminal ID in the ADDTASK command to establish a multiuser partition in which to run the SunRiver terminal. See the instructions on the ADDTASK command in this chapter.

The SunRiver terminals and the host monitor MUST all be installed as EGA monitors. The SunRiver host adaptor card can co-reside with most available EGA boards that you might use to run the host monitor. A list of those tested should be provided with your SunRiver hardware installation instructions.



---

Also, only one task at a time may view any particular display screen. For example, if the host computer is viewing its own display screen (task 0), no other user may press ALT-0 to switch tasks and view that screen at the same time. Multitasking partitions at the host do not present the same problem. As such, the system administrator may want to turn task switching off in multiuser partitions by using the MOSADM SWITCH OFF command.

A practical limit of one to four SunRiver workstations per host computer is recommended. Processor-intensive applications and the addition of mice at the workstations will additionally affect system performance. Your own limits will be determined by your specific application/hardware environment plus the level of system performance you feel is acceptable.

# Multi-Tasking / Multi-User

## MOS System Monitor

The MOS System Monitor is a terminate and stay resident (TSR) program that provides a dynamic status screen for viewing and changing your system's multitasking/multiuser environment. It can be thought of as an interactive MOS MAP display.

While the MOS MAP function only displays a map of statistical information about the partitions set up on your computer, the System Monitor program provides a menu interface with MOS which allows you to dynamically add and remove tasks without returning to MOS and using the ADDTASK or REMTASK commands.

The System Monitor program also interfaces with the MOSADM utility functions SLICE, PRI and CACHE. This allows you to change the time slice or priority values for tasks, or turn caching on and off, all from within the System Monitor program.

These capabilities provide an easy and convenient way to monitor and change the system environment to suit your processing needs.

To use the System Monitor first load the program into memory by entering the following command at the system prompt:

MONITOR

The following message will appear:

PC-MOS System Monitor v#.## Installed.  
(c) Copyright 1991 The Software Link, Incorporated  
To activate Monitor press CTRL-SPACE



The hotkey combination **CTRL SPACE** is used to activate the System Monitor once loaded. Pressing **CTRL SPACE** will bring up the System Monitor display, similar to the following:

PC-MOS System Monitor v#,##										
Task	Size	Video	User	Program	Port	Baud	Pri	Slice	Files	Status
0*	624K	EGA		MOS.COM		N/A	N/A	2	1	2 ACT DS
1	600K	MONO		COMMAND.COM		2	19200	2	1	7 ACT ND
2	600K	MONO		COMMAND.COM		3	19200	2	1	2 WAITDS
3	32K	MONO		LANSERVE.COM		N/A	N/A	2	1	10 WAITND

Highlight desired task and press ENTER for Command Window, or ESC to exit.

With the System Monitor display on the screen press **ENTER** to bring up the Command Window. The Command Window is a function submenu that is superimposed over the System Monitor display, as follows:

PC-MOS System Monitor v#,##										
Task	Size	Video	User	Program	Port	Baud	Pri	Slice	Files	Status
0*	624			F1 -Remove a Task		F6 -Change Keyboard Status				DS
1	600			F2 -Restart a Task		F7 -System Caching ON/OFF				ND
2	600			F3 -Create a New Task		F8 -Task Switching ON/OFF				DS
3	32			F4 -Change Priority		F9 -Change Hot-key				ND
				F5 -Change Slice		F10 -Remove Monitor				
				RET -Return to Monitor		ESC -Exit MOS System Monitor				

Use the function keys to select the desired function from the Command Window. When you select a specific function, messages or prompts will appear in the bottom half of the Command Window to assist you with the operation.

**NOTE:** Some of the functions in the Command Window are task specific, i.e. the operation will be performed on a specified task. To specify a task for modification you must use the cursor keys to highlight the desired task in the System Monitor display BEFORE pressing **ENTER** to call up the Command Window.

Any task specific functions you select will then be performed on the selected task. These include: remove a task, restart a task, change priority, change slice, change keyboard status, and task switching ON/OFF.

To perform any of the remaining non-task specific functions it does not matter which task is highlighted when you press the **ENTER** key to call up the Command Window.

If you are using a color monitor you may press **TAB** from the Command Window to change the color of the window.

Press **ENTER** to return to the System Monitor display from the Command Window. If you made any changes they will be immediately updated on the System Monitor display. You may optionally press **ESC** from the Command Window to exit the System Monitor or press **F10** to exit and remove the System Monitor from memory.

If the **CTRL SPACE** keystroke combination conflicts with other applications on your system, you may use **F9** to change the hotkey to another keystroke combination.

For more complete information on the various functions that can be performed with the System Monitor see the specific section of this User Guide that explains each function or command.

---

# CHAPTER 6: BATCH FILES

---

Introduction . . . . .	6 - 2
Batch File Features . . . . .	6 - 2
Creating Batch Files . . . . .	6 - 3
Batch File Commands . . . . .	6 - 3
.ABORT . . . . .	6 - 4
.AUTOCD . . . . .	6 - 5
.BATECHO . . . . .	6 - 6
.CALL/.RETURN . . . . .	6 - 7
.ECHO . . . . .	6 - 9
.FLUSH . . . . .	6 - 10
.FOR IN DO . . . . .	6 - 11
.GOTO . . . . .	6 - 13
.IF . . . . .	6 - 14
.INSERT . . . . .	6 - 17
.KEY . . . . .	6 - 18
.NEXT . . . . .	6 - 21
.PAUSE . . . . .	6 - 22
.REM . . . . .	6 - 23
.STOP . . . . .	6 - 24
.TEXT/.ENDTEXT . . . . .	6 - 25
Batch File Processing . . . . .	6 - 29
Automated Batch Files . . . . .	6 - 29
Nested Batch Files . . . . .	6 - 33
Replaceable Operand Files . . . . .	6 - 34
Compatibility . . . . .	6 - 36

# Batch Files

---

## Introduction

Batch files let you group commands in a file to create your own specialized processes. For example, you can create a batch file that will copy several files into one large file, .EXPORT the large file, and then erase the original files.

MOS recognizes batch files by their .BAT extensions. In the previous example, the name of the batch file could be EXP.BAT. To execute the batch file, type the file name, EXP, at the system prompt without its .BAT extension and press ENTER.

Any command may be used in a batch file. However, MOS contains several intrinsic commands that are designed just for use in batch files. They are:

```
.ABORT  
.AUTOCD  
.BATECHO  
.CALL/.RETURN  
.ECHO  
FLUSH  
.FOR IN DO  
.GOTO  
.IF  
.INSERT  
.KEY  
.NEXT  
.PAUSE  
.REM  
.STOP  
.TEXT/.ENDTEXT
```

## Batch File Features

Several features in MOS let you create very powerful batch files. You may nest batch files to permit one batch file to invoke another, process it, and then return to processing in the original batch file. With this feature, you can use one batch file to call another, which calls another, to create and execute complex command sequences.

You may also create a batch file with replaceable operands that let you use different data each time you execute it. These operands are placed in the file, and serve as variables. They will be replaced by the values you enter on the command line each time you execute the file.

Another feature in MOS lets you streamline the startup of your system with automated batch files. Every time MOS is booted, it checks for a batch file named AUTOEXEC.BAT, and executes the commands it contains.

For example, you may have several applications on your system, but always want to load the same application at startup. You can include the path and file name that load the application in your AUTOEXEC.BAT file. Each time the system is booted, it will then automatically arrive at that application. This can be useful in establishing partitions and signons. Automated batch files are discussed in this section.

## Creating Batch Files

You may use the MOS system editor, .ED, to create batch files in the same way you create other files. A batch file is an ASCII text file containing a group of commands as they would be entered from the keyboard. .ED lets you easily create a batch file and later add, delete, or insert lines. Refer to the Editor section of this manual for more information about creating a file.

## Batch File Commands

The following pages contain an explanation of each batch file command and how it is used. A section detailing the use of the special batch file features follows the command explanations.

# Batch Files

## .ABORT

.ABORT stops all processing within a batch file. This command is useful as a safety measure since batch file processing will be completely terminated regardless of any nested batch files.

**Type:** Intrinsic.

**Form:** .ABORT

### **Explanations:**

.ABORT is generally used with the .IF command. You may use the .IF command to test for various conditions, which must be true or false for processing to continue. In the example that follows, the conditional .IF statement is testing the value of an errorlevel, which is set by the processing of a command. Errorlevel 2 indicates that processing was unsuccessful. For more information about error levels, refer to the .IF command in this section.

```
.REM BACKUP BATCH FILE
.EXPORT C:\*.* A:/y
.IF ERRORLEVEL 2 .ABORT
.EXPORT C:\WPFILES\*.* A:/a /y
.IF ERRORLEVEL 2 .ABORT
```

If any step in the export process were to fail and generate an error-level 2, this batch file and any files that it may be nested from would be aborted.



## .AUTOCD

.AUTOCD recovers the current drive and directory that was saved with redirection to an output file from within a batch file. You may find this command useful with utility batch files or applications that call for a change of drive and directory.

**Type:** Intrinsic.

**Form:** .AUTOCD *filename*

### Explanation:

The example batch file which follows, FINDFILE.BAT, is a simple utility that streamlines the search for a file and changes directories several times.

```
.CD>C:\DIR1.SAV
.CD
.DIR %1
.CD \WORK
.DIR %1
.CD \WORK\TEMP
.DIR %1
.CD \WORK\TEMP\SAVE
.DIR %1
.AUTOCD C:\DIR1.SAV
```

To execute FINDFILE, type the name of the file you want to find at the prompt, and press ENTER. That file name will replace the %1 operand as explained later in this chapter. When FINDFILE executes, the current drive and directory will be output to the DIR1.SAV file. Then when the .AUTOCD command is encountered at the end of the file, that output will be retrieved, and drive and directory will be changed back to that which was current when FINDFILE was invoked.

Note that if you break out of FINDFILE by pressing CTRL BRK before the .AUTOCD command is encountered, the directory will not be changed back to the original directory.

# Batch Files

## .BATECHO

.BATECHO controls the initial state of .ECHO in batch files. .ECHO on is the default state of ECHO when MOS boots. This means that all commands in the batch file will be displayed on the video screen, as they are being executed. You may turn .ECHO off at the beginning of a batch file by entering .ECHO off as the first statement. However, if you use a .BATECHO off statement, for example, in your AUTOEXEC.BAT file, .ECHO will automatically be off when you start any batch file.

**Type:** Intrinsic.

**Form:** .BATECHO on | off

### **Operands:**

**on** leaves .ECHO on as the default for batch files.

**off** turns .ECHO off in batch files.

### **Explanations:**

Entering .BATECHO with no operand displays the current state of .ECHO. Unless .BATECHO off is issued, the default is .BATECHO on. An example from an AUTOEXEC.BAT file follows:

```
.BATECHO OFF  
.DATE  
.TIME  
.PROMPT $P$G  
.CLS
```

This file prompts you for the system date and time, and then initiates a prompt. It then clears the screen and only the prompt will display. Any batch files that may be executed after this file has executed will have .ECHO off.

If .BATECHO is off you may still display the commands within a batch file when it executes by using an ECHO on statement. If a nested batch file is then invoked, .BATECHO off will prevail for the nested batch file. (Also refer to the .ECHO command.)



## .CALL/.RETURN

.CALL statements let you redirect the order in which commands execute within a batch file. You may redirect processing to a specific line within the file if it is preceded by a label. When a .CALL command is encountered, execution proceeds with the first statement following the specified label. When a corresponding .RETURN command is found, processing is redirected back to the command following the .CALL statement.

**Type:** Intrinsic.

**Form:** .CALL *label*

:*label*

.RETURN

### Operand:

*label* a unique character string which may be up to eight characters long. You must use a colon before the label when it precedes a line to which processing is directed. However, do not use a colon with the label in the .CALL statement.

### Explanation:

.CALL/.RETURN lets you create subprocesses within a batch file. Subprocesses may be nested up to the capacity of your available memory. Each .CALL statement must have a .RETURN statement, or the file will not execute properly. In the following example, the .CALL/.RETURN statement redirects processing from the statement .ECHO PLACE DISK #1 IN DRIVE A: AND BACKUP DISK #1 IN DRIVE B:, to the .COPY statement. The .GOTO statement is explained later in this chapter.

## Batch Files

```
.GOTO START
:MOVE
:PAUSE
:COPY A:.* B:
:ERASE A:.*
:RETURN
:START
:ECHO PLACE DISK #1 IN DRIVE A:
:ECHO AND BACKUP DISK #1 IN DRIVE B:
:CALL MOVE
:ECHO PLACE DISK #2 IN DRIVE A:
:ECHO AND BACKUP DISK #2 IN DRIVE B:
:CALL MOVE
```

If no label is specified processing halts. Note that the label never displays during execution of a batch file.



## .ECHO

When .ECHO is on, the batch file statements are displayed on the video screen as they execute. .ECHO off prevents this display, but does not inhibit the display of any messages in the file.

**Type:** Intrinsic.

**Form:** .ECHO *on | off*

.ECHO *message*

### Operands:

*on* turns .ECHO on.

*off* turns .ECHO off.

*message* enter up to 121 characters to display on the video screen when the file is executed.

### Explanations:

MOS defaults to .ECHO on when it is booted. If you're uncertain about the state of .ECHO, type .ECHO with no operand and press ENTER. MOS will indicate whether it's on or off.

.ECHO is also useful in sending messages to the video screen. For example:

```
.ECHO OFF  
.CLS  
.ECHO THIS IS A LIST OF BATCH FILES  
.DIR *.BAT
```

Also refer to the .BATECHO command.

## Batch Files

---

### .FLUSH

.FLUSH mimics the action of pressing CTRL PgDn to clear the keyboard save buffer of entries. This prevents someone from viewing your previous keystrokes.

**Type:** Intrinsic.

**Form:** .FLUSH

#### **Explanation:**

MOS contains a command recall buffer that lets you retrieve your previous entries. You may use the up arrow to retrieve entries beginning with the newest entry in the buffer. Use the down arrow to retrieve beginning with the oldest entry.

SIGNOFF automatically clears this buffer. However, if you're not signed on under MOS Security, you may use .FLUSH to clear this save buffer. You may find it useful to include it in a batch file you invoke when you finish working in MOS. For example, look at the QUIT.BAT file which follows.

```
.FLUSH  
.CLS
```

To quit working in MOS, just type QUIT. Because you've cleared the buffer and cleared your screen, no one can retrieve your keystrokes to see what you were working on.



## .FOR IN DO

.FOR IN DO lets you repeat the execution of a command for each member of a user-defined group. The members of the group are processed sequentially. The group may contain commands, file names, replaceable operands or any constants you specify.

**Type:** Intrinsic.

**Form:** .FOR %*character* IN (*group*) DO *command*{%*character*}

### Operands:

%*character* replaced by a member of the group and must be preceded by %% when used in a batch file. A single character in the range A-Z should be used. If the command is entered at the system prompt instead of in a batch file, only one % is necessary.

(*group*) a set of parameters or a single parameter enclosed in parentheses.

*command* any MOS command that you want each member of the group to execute.

### Explanation:

Members of the group will be processed sequentially for the command specified. If the group is comprised of filenames with wildcards, the sequence of execution will occur in the order in which files matching the wildcard definition are encountered. An example follows.

```
.FOR %%A IN (*.asm *.doc *.lst) DO .COPY %%A B:
```

Note the use of a space as a delimiter between the items that comprise this group. In this example, the files that carry an .ASM, .DOC, and .LST extension make up the group. Each file with one of these extensions is a member of the group and each will be executed as an operand with the .COPY command. The position in %%A will be replaced by each of the files in the group. So if three files, MOSCOM, MOSATS, and MOSXOM, carrying the .ASM extension were encountered the execution would occur as follows:

## Batch Files

---

```
.COPY MOSCOM.ASM B:
```

**Files copied: 1**

```
.COPY MOSATS.ASM B:
```

**Files copied: 1**

```
.COPY MOSXCOM.ASM B:
```

**Files copied: 1**

The group members in this example could have been any type of files. Note that replaceable operands could be used as group members. To execute the file, the real operands would be entered on the command line.



## .GOTO

.GOTO redirects processing within a batch file to the line immediately following a specified label. Processing within the file continues from that point.

**Type:** Intrinsic.

**Form:** .GOTO *label*

### Operands:

*label* a unique character string which may be up to eight characters long. You must use a colon before the label in the file, when it precedes a line to which processing is directed. However, do not use a colon with the label in the .GOTO statement.

### Explanations:

Commands are executed beginning with the line after the .GOTO label. A label is never displayed during the batch file's execution. For example:

```
.EXPORT *.DOC A:  
.IF ERRORLEVEL 2 .GOTO ERROR  
.EXPORT *.WKS A:  
.IF ERRORLEVEL 2 .GOTO ERROR  
.ECHO DONE  
.ABORT  
:ERROR  
.ECHO AN ERROR WAS ENCOUNTERED
```

This batch file checks for errors during the .EXPORT and, if an error was encountered, reports it. Note the redirection of processing to the :ERROR label. If no label were specified, the batch file execution would stop when it encountered the .GOTO command and display the message, "Label not found."

# Batch Files

## .IF {NOT}

.IF permits conditional processing, where a condition must be met for processing to occur. The three types of conditions that .IF allows are: error levels, character strings, and EXIST statements.

**Type:** Intrinsic.

**Form:** .IF {NOT} *condition command*

### **Operands:**

{NOT} tests for the opposite condition.

*condition* must be an error level, a character string, or an EXIST statement.

*command* any command is permitted.

### **Explanations:**

You may use .IF to set up true/false processing. If a statement is true processing occurs; if it is false, it does not. This applies to all three types of conditions. You may also use the NOT operand to test for the opposite of the specified condition. In the following example, if the file MYFILE.DOC does not exist, processing skips past the next command line in the batch file.

```
.IF NOT EXIST MYFILE.DOC .GOTO STEP
.COPY MYFILE.DOC B:
.ABORT
:STEP
.ECHO FILE DOES NOT EXIST PLEASE CHOOSE ANOTHER
```



## Error Level Conditions

The value of the system variable ERRORLEVEL may be tested using an .IF statement within a batch file. The form is:

```
.IF ERRORLEVEL number command
```

This means that if an error level greater than or equal to that specified is encountered, the command will be executed. Note that some application programs may set ERRORLEVELS that are not defined as those in MOS. The MOS standard error levels are:

0 - normal execution

1 - execution of the command, but with a questionable condition.

2 - some error occurred during execution of the command.

You may also use an error level with a false statement. For example:

```
.EXPORT myfile.doc a:  
.IF NOT ERRORLEVEL 1 .GOTO OK  
.ECHO Export was unsuccessful  
.ABORT  
:OK  
.ERASE TEXTFILE.DOC
```

The statement IF NOT ERRORLEVEL 1 will only be true when ERRORLEVEL is 0.

# Batch Files

String == String

You may use string == string to create conditions which must match perfectly, before processing can continue. The command form is:

```
.IF STRING == STRING command
```

The strings must match perfectly, though the match is not case sensitive. Either string or both strings may consist of replaceable operands. In the following example, if no operand was specified when the batch file was started, the variable %1 will be null and the IF condition will appear as .IF !=!.GOTO NONAME.

```
.IF !%1 == ! GOTO NONAME.  
.COPY %1 B:  
.STOP  
:NONAME  
.ECHO Must use a filename
```

## Exist Statements

You may use a condition with an .EXIST statement. The form for an exist statement is:

```
.IF EXIST filename command
```

The negative form of .EXIST is also permitted. For example:

```
.IF NOT EXIST TEST1.DOC .ABORT
```



## .INSERT

MOS recognizes two modes of command line editing -- Typeover, which is the standard default, and Insert. Typeover mode is destructive and lets you typeover existing entries. Insert mode inserts characters into an existing command line entry. The .INSERT command, used in an AUTOEXEC.BAT file lets you specify that the Insert mode is active when MOS is first booted. You may also use .INSERT in other batch files.

**Type:** Intrinsic.

**Form:** .INSERT

### **Explanation:**

Specify Insert mode as follows in your AUTOEXEC.BAT:

.INSERT

If you don't include .INSERT in your AUTOEXEC.BAT file, MOS will boot with the Typeover mode active, which is recognized by its underlined cursor. You may then switch to Insert mode by pressing the INS key. This mode is identified by the block cursor.

# Batch Files

## .KEY

.KEY gives you the ability to prompt for a keystroke from within a batch file and take action dependent upon the key which was pressed.

**Type:** Intrinsic.

**Form:** .KEY %%a IN (set of keys) DO command %%a

### **Operands:**

%%a may be any character from A to Z.

(set of keys) enter the group of keys or range of keys which you want to invoke action. Keys must be separated by a delimiter such as a comma or a space. This group may include the letters A through Z, the numbers 0 through 9 and the function keys F1 through F10. You may specify a range of keys. If you do, they must be separated by a hyphen. Note that if a lower case letter is entered, MOS internally converts it to an uppercase letter. Therefore, if b is used you could press a lowercase b or an uppercase b and produce the same effect. You may also use SP for a space and CR for a carriage return.

command enter the command you want executed after the key has been pressed.

### **Explanation:**

The .KEY command is often used with either .GOTO, .CALL, or the name of a user-defined batch file. Each of these commands may be used to redirect processing within a batch file. The replaceable operand, %%a, specifies to which point processing is to be directed. For example:

```
.COPY %1 B:  
.ECHO Do you want to erase %1? (Y/N)  
.KEY %%A IN (Y,N) DO .GOTO .CHOICE%%A  
.CHOICEy  
.ERASE %1  
.CHOICEn
```



In this example if you were to press y, .GOTO choicey will be executed. The .GOTO command redirects processing to the :choicey branch within this batch file. Note that the character you press, either y or n will replace the %%a operands, thereby redirecting processing to a labelled point. In this example, there is no space between choice and %%a.

The .KEY command may also be used with the .CALL command. The .CALL command lets you redirect processing in a batch file to a labelled subroutine, execute it and return processing to the original point with a .RETURN command. Look at the use of .KEY in this example:

```
:START
.TEXT Press the letter of the drive you want to review (A,B)
Press F1 to exit
.ENDTEXT
.KEY %%A in (A-B,F1) DO .CALL SUB%%A
.GOTO START
:SUBA
.DIR A:/w /p
.RETURN
:SUBB
.DIR B:/w /p
.RETURN
:SUBF1
```

If a is pressed, processing will be redirected to SUBA2. After the .RETURN is encountered processing is directed back to the next batch file statement after the .KEY statement. When F1 is pressed, processing is redirected to the label SUBF1 which is at the end of the file. It is permissible to end a batch file with a .CALL pending. MOS will compensate for the lack of a corresponding .RETURN.

If you had pressed a key that was not in the group (A-B,F1), that keystroke would be ignored and processing halted until a valid key was pressed.

.KEY may also be used to execute other batch files. In this example, there are three batch files, BATR.BAT, BATN.BAT, and BATQ.BAT. Look at the following statement:

## Batch Files

```
.KEY %%a IN (r,n,q) DO bat%%a
```

If you pressed r, BATR.BAT would be executed. If you pressed n, BATN.BAT would be executed and so on.

You may also couple .KEY with the .TEXT/.ENDTEXT commands to create a menu that will be displayed when a batch file is executed. Note that you should use the .TEXT command before the .KEY command because .KEY will halt processing until a key is pressed. For example:

```
:MENU  
.CLS  
.TEXT
```

Press one of the following:

- 1) COPY
- 2) DISPLAY
- 3) DELETE
- 4) QUIT

```
.ENDTEXT  
.KEY %%a IN (1-4) DO .CALL route%%a  
.GOTO MENU  
:ROUTE1  
.COPY %1 B:  
.RETURN  
:ROUTE2  
.TYPE %1 ; .MORE  
.RETURN  
:ROUTE3  
.ERASE %1  
.RETURN  
:ROUTE4  
.ABORT
```

In this example, the menu will be displayed on the screen. Dependent upon the key that you press, processing would be directed to one of four points within the file.

## .NEXT

.NEXT makes it possible to use more than ten replaceable operands in a batch file by shifting the operands one position to the left. Replaceable operands are explained later in this chapter.

**Type:** Intrinsic.

**Form:** .NEXT

### Explanation:

Whenever you use .NEXT you will lose access to the current value assigned to the first operand (%0). Look at the example below and the values for the operands.

```
%0 %1 %2 %3 %4 %5 %6 %7 %8 %9  
a b c d e f g h i j
```

.NEXT

```
b c d e f g h i j k
```

Note that this permits you to add the k operand from the command line. Now look at the next example, a batch file called XYZ.BAT.

```
:LOOP  
.IF !%1==! GOTO DONE  
.COPY %1 B:  
.NEXT  
.GOTO LOOP  
.DONE
```

Note that the use of !%1==! lets you test for a null case, where the operand does not exist. Any character may be used in place of the !. To invoke this batch file, type:

```
XYZ File1 File2 File3...
```

The batch file will copy File1, File2, and File3 and so on until no more operands are encountered, permitting you to enter more than ten operands.

## Batch Files

---

### .PAUSE

.PAUSE temporarily halts processing and displays the message, "Press a key when ready." A user must then press a key to continue processing or break out of the batch file with a CTRL BRK or a CTRL C.

**Type:** Intrinsic.

**Form:** .PAUSE *{message}*

**Operand:**

*message* enter up to 120 characters for the message string.

**Explanation:**

.PAUSE allows you to temporarily halt processing within a batch file. You may elect to use the message operand, but the message will not be displayed unless .ECHO is on.

When processing is halted with .PAUSE, it may be restarted by pressing any keys except CTL BRK and CTL C. For example:

```
.ECHO Insert a new diskette  
.PAUSE
```

You may also use .PAUSE to display a message in addition to the "Press any key to continue" message. For example:

```
.PAUSE Insert a new diskette
```

## .REM

.REM inserts a remark into a batch file. If .ECHO is on, the remark displays on the video screen.

**Type:** Intrinsic.

**Form:** .REM {remark}

### Operand:

*remark* may be any character string up to 122 characters long.

### Explanation:

You may want to use .REM to insert remarks or comments into your batch file. If you want to use .REM to display remarks on the video screen during the execution of the batch file, .ECHO must be on.

.REM comments are useful to explain the logic within a batch file.  
For example:

```
.REM test for error-free .IMPORT before .RENAME
```

## Batch Files

.STOP

.STOP causes an immediate exit from a batch file.

**Type:**      Intrinsic.

**Form:**      .STOP

### **Explanations:**

You may use .STOP with the .IF command, to specify a condition which requires a .STOP. The .IF command also may be used to specify a condition that must exist for processing to occur. For example:

```
.IF NOT EXIST MOSTEST.DOC .STOP
```

If .STOP is used with nested files, it will return to the previous .BAT file. For example, if the file is only nested one layer deep, and is called by the original batch file, processing returns to the original file.

If the file is nested two layers deep -- called by a batch file that was called by the original file, processing returns to the batch file that called the .STOPped file. Execution of the file continues with the next command statement after the name of the batch file that was just exited.



## .TEXT/.ENDTEXT

.TEXT/.ENDTEXT lets you create a display within a batch file that appears on the video screen during its execution. Anything you enter within the .TEXT and .ENDTEXT commands echoes to the screen independent of the state of .ECHO. You cannot invoke commands from within the text block. Several operands lets you control the colors that will appear on the video screen, and the initial position of the cursor within the text block.

**Type:** Intrinsic.

**Form:** .TEXT {/Sbf} {/Kbf} {/Cbf} {/Pxxyy}  
| bf  
.ENDTEXT

### Operands:

/Sbf enter /S followed by the codes of the background and foreground colors you want to use for the entire display screen. Any existing text will be converted to this new color. This operand must be entered on the line with the .TEXT command. The colors you set carry forward to all .TEXT screens in the current batch file and all nested batch files until you change them with another /Sbf or with a /Kbf entry or return to the system prompt. See the following color table for the codes you may enter for the b (background) and f (foreground) colors.

/Kbf enter /K followed by the codes of the background and foreground colors you want to use for the entire display screen. Unlike the /Sbf option, the /Kbf option clears the display. The colors you set carry forward just as those in the /Sbf option.

## Batch Files

---

- /Cbf enter /C followed by the codes of the background and foreground colors to display on the first line and all subsequent line in this .TEXT screen only, or until you change them. This operand may be entered only on the line with the .TEXT command. The /C operand does not carry your selection forward to other .TEXT screens. You may find /C useful because it lets you enter an operand to determine the initial color for the text block. You cannot enter a replaceable operand within the text block. See the following color table for the codes you may enter for the b (background) and f (foreground) colors.
- :bf enter the broken vertical bar followed by the background and foreground colors anywhere in the text block to change the current color settings. This operand may be entered anywhere in the text display, but will not display on the video screen.
- b designates a code for a background color. With any of the operands, enter the single character code for the color to use for the background. Valid codes and corresponding colors are shown in the following Color Code Table.
- f designates a code for a foreground color. With any of the operands, enter the single character code for the color to use for the foreground. Valid codes and corresponding colors are shown in the following Color Code Table.
- /Pxxyy enter /P followed by the x and y coordinates to define the initial position of the cursor for the text block. The xx defines the column number and may be from 1 to 80, and the yy defines the line number and may be from 1 to 25. Four digits must appear after the /P, for example, /P0103.



### Color Code Table

Background Codes		Foreground Codes	
0	Black	0	Black
1	Blue	1	Blue
2	Green	2	Green
3	Cyan	3	Cyan
4	Red	4	Red
5	Magenta	5	Magenta
6	Brown or dark yellow	6	Brown or dark yellow
7	Light grey or white	7	Light grey or white
8	Blinking black	8	Dark grey or black
9	Blinking blue	9	Light blue
A	Blinking green	A	Light green
B	Blinking Cyan	B	Light cyan
C	Blinking red	C	Light red
D	Blinking magenta	D	Light magenta
E	Blinking brown	E	Yellow or light yellow
F	Blinking grey	F	White

Note that the value for a blinking background color is eight in hexadecimal added to the 0-7 values.

#### Explanation:

You may use .TEXT/.ENDTEXT to create screens for display during execution of a batch file. The image will be displayed on the video screen just as it exists in the batch file.

You may find it useful to use the clear screen command, .CLS, before the .TEXT statement to clear the screen for the display. For example:

```
.CLS  
.TEXT  
Exporting files  
.ENDTEXT
```

## Batch Files

You could specify the foreground and background colors for this display with the pipe operand, for example:

```
.CLS  
.TEXT  
!13 Exporting files  
.ENDTEXT
```

The !13 operand specifies that "Exporting files" will appear as cyan on a blue background and will be maintained until another background foreground operand entry is encountered within the display. Note the use of the broken vertical bar, !, it must precede each background, foreground entry. These entries will not appear as part of the display.

You may set default colors with the /Sbf, /Kbf, and /Cbf operands. You may also set the initial cursor position within the text with the /Pxxyy operand. For example:

```
.TEXT/S14 /P0105
```

The text will start on line 5. The contents of the screen will be converted to red on a blue background. This color setting becomes the new default for all subsequent uses of .TEXT because /S was used.  
.ENDTEXT



## Batch File Processing

MOS gives you great flexibility in creating batch files. You may group several commands in a batch file to create your own special utilities. In this section, we'll discuss three types of batch files -- automated batch files, nested batch files, and utility files that can be executed with replaceable operands.

### Automated Batch Files

When MOS is booted it looks for a startup file called AUTOEXEC.BAT to establish the conditions under which you begin processing. You may use it to set the .PATH to a specific directory, or to specify your .PROMPT. Since this file will be executed each time you boot your system, you will want to customize it to your needs. Some commands you may want to include in your AUTOEXEC.BAT are:

```
.SET  
.ENVSIZE  
.PATH  
.PROMPT  
.TIME  
.DATE  
.ADDTASK  
.SIGNON  
.CLS  
.INSERT
```

#### .SET

The .SET command is used to set frequently accessed strings of information into the environment, which is a reference area in memory. A copy of the environment is made available to all programs within a partition. Some items are automatically set into the environment when they are first invoked by command, such as .PATH and .PROMPT, which are discussed later.

# Batch Files

## **.ENVSIZE**

You may use the .ENVSIZE command to specify the minimum size of the environment. It must be followed by the size of the environment in bytes. For example, if you specify .ENVSIZE 256, this permanently reserves at least 256 bytes for the environment.

## **.PATH**

The .PATH command sets the directories to search and the order in which they will be searched for an executable file or a batch file, before displaying a file not found message and ending the search. If you wanted to bring up a specific application each time you booted, you would set the path for the directory that contains that application. For example if you wanted to bring your system up in a word processing application that resides in a directory that is removed by several levels from the root, you might enter:

```
.PATH=C:\dir1\dir2\dir3\wp
```

## **.PROMPT**

You may use the .PROMPT command to customize your system prompt. MOS defaults to its system prompt of the current drive and directory unless otherwise specified. You might find it helpful to redefine your prompt to reflect the date, the time, or some character string. For example, you may enter:

```
.PROMPT [JAN]
```

This would create the prompt:

```
[JAN]
```

You could use the .PROMPT command to automatically display the system time and date in your prompt. For more information about the prompt values, refer to the .PROMPT command in the General Commands section of this manual.

## .TIME

If your system does not have a perpetual clock, you may use .TIME to enter the system time. The .TIME is the internal time that is known to MOS. You must enter it in the format specified during system setup. For example:

14:24:03.42 is the current time  
New time is:

In this format, in which hours are expressed in military time, hundredths of seconds may be indicated in the last position.

When entering the time, you may omit the seconds and hundredths of seconds.

## .DATE

If your system does not have a perpetual calendar, you may use .DATE to enter the system date. The .DATE is the internal date that is known to MOS. You must enter it in the format specified during system setup. For example:

Wed 11-11-87 is the current date  
New date is (mm-dd-yy):

## .ADDTASK

If you know that you always want to have certain partitions set up in your system when it boots, include the .ADDTASK command in the AUTOEXEC.BAT. You may use multiple .ADDTASK commands to set up as many partitions as your system permits. In the example that follows, the .ADDTASK command statement adds one partition, called partition 1, to the system.

.ADDTASK 256,1,C,PART1

# Batch Files

This .ADDTASK statement gives the partition 256K of memory, sets its security class as C, its task ID as 1, and its startup file as PART1.BAT. More information about the .ADDTASK command is contained in the Multi-Tasking/User section of this manual. Note that the startup file contained in the .ADDTASK statement acts as an AUTOEXEC.BAT for the partition that it creates. You would use this startup file to set the conditions for processing within the partition.

## **.SIGNON**

If you've chosen to use security for your system, include the .SIGNON command in your AUTOEXEC.BAT. This will automatically display the MOS Signon screen when the system is booted:

User ID:

Password:

Note that SIGNON is only valid if a \$\$USER.SYS file exists.

## **.CLS**

.CLS clears the video screen. Include it at the end of your AUTOEXEC.BAT if you will be prompted to make entries during its execution. It will give you a clean screen once your startup has completed.

## **.INSERT**

The MOS command line entry operates in two modes -- insert and typeover. These modes function in the same manner as those in an editor. Insert mode allows you to add characters without typing over existing entries. Typeover, which is the MOS default, means that you will be overwriting characters when you add an entry to an existing line. If you want to invoke the Insert mode when your system is booted, include .INSERT in your AUTOEXEC.BAT.



## Nested Batch Files

You may create a batch file that in turn calls another, processes it and returns to the original file. However, if you've set up security for your system, remember that you can only call a batch file to which you have access. You can use this nesting feature to create subroutines within your batch files. You may couple this nesting feature with testing for conditions, to create branches within your files. The following batch file, ALLCOPY.BAT contains an example of three possible branches from a copy operation.

```
.COPY A:/*  
.IF ERRORLEVEL 2 SERIOUS  
.IF ERRORLEVEL 1 .GOTO ER1  
NORMAL  
.GOTO NEXT  
:ER1  
POSSIBLE  
:NEXT  
.PAUSE Insert next diskette for processing  
. .  
. .  
. .
```

NORMAL.BAT contains the following:

```
.CLS  
.TEXT
```

All files successfully copied

```
.ENDTEXT
```

If errorlevel 0 which indicates a successful operation occurred during execution of ALLCOPY, then NORMAL.BAT would be called and the message, "All files successfully copied" would be displayed on the video screen. If errorlevel 1 is returned, which indicates that a possible error occurred, POSSIBLE.BAT would be called. It contains:

## Batch Files

```
.CLS  
.TEXT
```

An error may have occurred. You may want to perform a VERIFY after this job has completed. You may halt processing by pressing CTL-C.

```
.ENDTEXT  
.PAUSE
```

If errorlevel 2 occurred, which indicates that the operation was not successful because of an error, SERIOUS.BAT will be called. It contains:

```
.CLS  
.TEXT
```

A serious error occurred, processing aborted...

```
.ENDTEXT  
.ABORT
```

In these examples, the called batch files were executed and processing was returned to the calling batch file. The called batch files themselves could have contained conditional statements that called additional batch files. So you can nest files as many levels deep as you need, up to the full extent of the memory in your system.

### Replaceable Operand Files

You can create a batch file that performs the same group of commands, but with different operands each time it is executed. This is accomplished using replaceable operands in the file. When you execute the file, MOS replaces the operands with those you enter on the command line. For example, look at the batch file, named MOVE.BAT that follows:

```
.ECHO OFF  
.CLS  
.COPY %1 %2  
.ERASE %1 /v
```



To execute MOVE.BAT, type the following at the system prompt:

```
MOVE C:\WP\*.DOC A:\
```

These values are assigned to the replaceable operands, %1 and %2. Additional operands, %0-%9 are also available for use. The initial value of %0 is the name of the batch file itself. More than 10 operands may be accessed from the command line using the .NEXT command explained earlier in this chapter. Note the use of the /v operand with the .ERASE statement. This causes MOS to display a verify prompt before deleting each file.

In addition to the %0 through %9 system variables, MOS supports a %A variable which can be used to determine the current task number from within a batch file. Upon finding the %A variable within a batch file statement, the command processor will determine the number of the current task and replace the %A with that number. This is very useful in AUTOEXEC or Startup batch files.

You may also use replaceable operands to pass data from one batch file to another. You may do so by setting strings into the environment with the .SET command. Look at the two example batch files that follow. The first is called A.BAT. It contains:

```
.SET DEST=A:  
.SET SOURCE=LIST1.DOC  
B  
.SET SOURCE=LIST2.DOC  
B  
.  
.  
.
```

The second batch file, B.BAT contains:

```
.COPY &SOURCE &DEST
```

Each time B.BAT is executed the current values for SOURCE and DEST are found in the environment and substituted in place of &SOURCE and &DEST. Note the use of the leading ampersand to identify these operands as environment reference variables. This allows one batch file to control the action of another.

## Compatibility

Generally, batch files developed under PC-DOS will run unmodified under MOS. There are, however, some differences. Two areas of batch file operations in which MOS's differences from PC-DOS can cause problems are:

1. MOS nests batch file to batch file calls, while DOS chains by calling the named batch file with no return to the calling batch file.
2. MOS sets an errorlevel on certain internal commands.

The environment variable, \$COMPAT\$, allows the user to modify MOS's behavior to allow for better compatibility with PC-DOS's batch file handling.

For example, issuing the command:

```
SET $COMPAT$=/N
```

will cause MOS to act like PC-DOS when one batch file is named from within another. If you have a batch file which counts on PC-DOS's non-nesting type of behavior, you can use this feature. If, however, you have batch files which are designed to nest -- do NOT use this option. (This option is primarily intended for compatibility with installation batch files that come with some applications.)

Issuing the command:

```
SET $COMPAT$=/I
```

will cause MOS to not set ERRORLEVEL on any internal commands. (This is also intended for compatibility with installation batch files.) Any part of this command may be entered in either upper or lower-case, and the switches may be combined. For example:

```
set $compat$/n/i
```

This feature control was implemented so that it would remain as set across command processor levels. For example, if a particular \$COMPAT\$ setting is in place and the installation batch file does a COMMAND/C, or some other type of shelling, the switch options will survive.

---

# CHAPTER 7:

## MOS EDITOR - ED

---

Introduction . . . . .	7-2
Invoking the Editor . . . . .	7-2
Editing in the Command Mode . . . . .	7-3
Command Mode Commands . . . . .	7-5
A - ASSIGN . . . . .	7-6
C - COPY . . . . .	7-7
D - DELETE . . . . .	7-8
E - END . . . . .	7-9
F - FILENAME . . . . .	7-10
H - HELP . . . . .	7-11
I - INSERT . . . . .	7-12
L - LIST . . . . .	7-13
M - MOVE . . . . .	7-14
P - PRINT . . . . .	7-15
Q - QUIT . . . . .	7-16
R - REPLACE . . . . .	7-17
S - SEARCH . . . . .	7-18
V - VISUAL MODE . . . . .	7-19
W - WRITE . . . . .	7-20
X - EXECUTE MACRO . . . . .	7-21
Editing in Visual Mode . . . . .	7-22
Visual Mode Editing Keys . . . . .	7-24
Using Macros . . . . .	7-26

# MOS Editor - ED

---

## Introduction

The MOS Editor provides an easy way for you to create new files or modify existing files. There are two modes of editing that you may select to use; the Command mode and the Visual mode.

The Command mode displays the file on your screen with line numbers. You can create or modify the file one line at a time by invoking commands associated with a specific line number. Any activity to the file does not appear on the screen until you re-display the file.

The Visual mode is similar to a word processor in that you may edit screens of text. The file displays without line numbers, and the cursor is positioned within the text. You can move anywhere in the file to add or delete text, copy a line, delete a line, page through screens, etc. In the Visual mode, all activity appears on the screen as it occurs.

From either mode, you can display a graphic character table, and assign graphic characters to the ten function keys. The function keys while in the Visual mode to place graphic characters into your text. You may find the graphic characters useful for designing special effects into your own menus and screen displays.

ED makes backup copies of all changed files. When an existing file is changed and saved, the old file is saved with a .BAK extension. The updated version retains the original filename and extension.

## Invoking the Editor

The MOS Editor is invoked with the extrinsic command, ED. To edit a file, enter the command at the system prompt, followed by the name of the file you want to create or modify. For example:

ED MENU.BAT

You may create or modify a file on a different drive or in a different directory by including the appropriate designations with the file name. For example:

ED D:\MYDIR\MYWORK\myfile.txt



A message displays as the Editor is loading, and then the edit screen appears similar to the following.

```
---FILE=MYFILE.TXT-----CL=1-----INSERT-
1 This is an example of the
2 MOS Editor, ED, as it will appear
3 on your screen
```

When you invoke the Editor in this manner, it will be in Command mode. The cursor is positioned at the top of your screen on the command line. The control line displays the file name, current line (CL=1) of control, and that the insert mode is on. If the file exists, the first 22 lines display on the text screen.

You may also invoke the editor directly in the Visual Mode by entering /v with the .ED command as follows:

```
.ED filename /v
```

The following explains how to edit a file in the Command mode, and the commands that are available. Following the commands is an explanation of how to edit a file in the Visual mode, and the editing keys that are available.

## Editing in Command Mode

In the Command mode, all commands and text entries are made on the command line. You can edit any line in the file by typing the line number on the command line and pressing ENTER. For example, if you type a 2 and press ENTER, the current text for line 2 appears on the command line as in the following:

```
the MOS Editor, ED, as it will appear
---FILE=MYTEXT.DOC-----CL=1-----INSERT-
1 This is an example of a file created with
2 the MOS Editor, ED, as it will appear
3 on your screen.
```

## MOS Editor - ED

You may change or edit the text on the command line. If you are creating a new file, you may type any text for the line. The text may not contain more than 80 characters. When you press ENTER, the text for the line is saved in the file. The edited line then becomes the current line.

Using the previous example, the control line would show CL=2 when you press ENTER. To display your changed line, you may enter the list (L) command. The file would re-display on the text screen, beginning with line 2, which is the current line. The current line is used as a default value for most commands when a line number is not included.

All commands in the Command mode consist of a single character. Because the commands are not case sensitive, you may enter them in either upper or lower case characters. Any operands always precede the command itself, and are separated by a space delimiter. For example, to copy (C) lines 1 through 3 to line 10, the command form is:

1 3 10 C

You may type as many commands on the command line as it will hold. You are not limited to issuing one command at a time. When you finish editing a file, there are two ways you may exit from the MOS Editor. The end (E) command saves the edited file on disk and returns you to the system prompt. The quit (Q) command displays a prompt that lets you select whether or not to save changes to the edited file. After entering your selection, MOS returns you to the system prompt.



## Command Mode Commands

Many commands are available in the Command Mode, and are explained in alphabetical order following these special notes about using the Command mode:

- These commands can be invoked only from the command line.
- You may press **CTRL Y** to delete the contents of an entire command line, whether it contains a command or a line of text.
- You may use the arrow keys and the backspace key to edit any entry on the command line.
- The **HOME** key moves the cursor to the first position on the command line, and the **END** key moves the cursor to the last position on the line.
- Press the **insert** key to toggle the insert mode on and off. When **insert** is on, you may insert text anywhere in the line. When **insert** is off, you will overwrite any current text.
- Most commands use the current line as a default when a line number is not entered with a command.
- Commands may be entered in either upper case or lower case.
- All operands must precede the command, and must be separated with a space delimiter.

## MOS Editor - ED

### A - ASSIGN

The A command displays a table of graphic characters that you may assign to function keys. When the assignment is complete, you may then use the function keys in the Visual mode to place the graphic characters in the text of a file. Press the ESC key to exit from the graphic character table and return to the Command mode.

**Form:**      A

**Explanation:**

When you invoke the A command, the graphic character table appears with any current function key assignments. If a function key does not have a current assignment, the message "undefined" appears. For example:

F1 = Undefined

To assign a character to a function key, use the arrow keys to move the cursor to the desired graphic character and press the function key. You may change the current character assigned to a function key by moving to a different graphic character and pressing the function key.

To remove a character assignment, place the cursor in the first position on the first line of the graphic character table. This position appears blank on the screen. Press the function key that has the assignment you want to remove. The display to the left of the graphic character table shows the "undefined" message for that function key.

When you assign values to the function keys, they are stored in a file named DEFKEYS.ED in the current directory. Each directory may have its own unique DEFKEYS.ED file. This file is loaded in a directory each time the editor is invoked in that directory.

If you exit the A command by pressing ESC without assigning any values to the function keys, a DEFKEYS.ED file is still created.

You will notice several blanks in the table of graphics characters. Character 20 is actually the "space" character. The blank characters 00, 07, 08 0A, 0D and FF are not recommended for use.



## C - COPY

The C command lets you copy one line or a block of lines to another location in the file.

**Form:**    *[f l] d C*

### Operands:

*f*                the number of the first line to be copied.

*l*                the number of the last line to be copied. To copy only one line, enter the same line number for both the first and last line number operands.

*d*                the number of the first destination line where the copied lines are to be located.

### Explanation:

You may copy the current line to a destination line by entering only the destination line number. The number that appears after CL= on the control line is the current line. For example, if the current line is 1 and the last line in the file is 28, you may copy line 1 to the end of the file with the following command:

29 C

To copy a block of lines, you must enter the first and last lines of the block in the command. For example to copy lines 1 through 5 to the end of the file, you would enter:

1 5 29 C

When copying lines, the destination line number must be outside the range of lines to copy. If the destination line is within the copy range, the copy will not take place. If the destination is past the current end of the file, the last line of the file becomes the destination. When you copy text to an existing line, the copy is inserted beginning at the destination line number. The existing line and all subsequent lines are moved down in the file. The copied and existing lines are renumbered in sequence.

## MOS Editor - ED

### D - DELETE

The D command lets you delete a line or a block of lines from the file.

**Form:**    {f} / D

#### **Operands:**

f        the number of the first or only line to be deleted.

/        the number of the last line of the block to be deleted.

#### **Explanation:**

You may delete only one line by entering the number of that line followed by the D command. For example:

42 D

Any other lines in the file after line 42 are moved up one line, and are renumbered in sequence. You may delete the current line by entering only D on the command line and pressing ENTER.

You may delete a block of lines by entering the first and last line numbers of the block to delete. For example, the following command will delete lines 3 through 10:

3 10 D

If you delete a block of lines and the last line number you specify is past the last line in the file, all lines through the last line of the file will be deleted.



## E - END

The E command ends the current editing session, and returns control to MOS.

**Form:**      E

### **Explanation:**

When you end an editing session with the E command, the edited version of the file replaces the original input version on the disk. The following message appears as the output is being saved:

Writing output file: filename

If no changes were made to the file, the original input file remains on the disk and is not replaced. If the F command was used during the editing session to assign a new file name to the output, then the output is written to the new file whether or not any changes were made. The original input file remains unchanged.

## **MOS Editor - ED**

---

### **F - FILENAME**

The F command lets you assign a new file name for output. When you issue a command that writes output, the output is saved to the new file name. The F command does not by itself cause anything to be written to the disk.

**Form:** Ffilename

#### **Operands:**

*filename* enter the new name you want to assign. There cannot be any spaces between the F and the filename, or within the filename.

#### **Explanation:**

You may assign a new file name at any time during the editing session. When you write the file to the disk, the output is written to the new file name. If a file with the new name does not exist, ED creates the file for you. If a file with the new name does exist, the output replaces any current information in the existing file.

For example, you may select to edit a file named MEMO.TXT. During the editing session you may assign a different file name for output by entering the following command on the command line:

Fmemo.sav

You may then continue editing the file. When you issue a W (write) or an E (end) command, the current text is saved on the disk in the MEMO.SAV file. If you enter the Q (quit) command and select not to save any changes, the original input file remains unchanged, the new file name is ignored, and no output is written to any file.



## H - HELP

The H command displays a list of Command mode commands.

**Form:**      H

### **Explanation:**

When you invoke the H command, a list summarizing the available Command Mode commands and their uses appears on your screen.

Once you have the information you need, you may press any key to return to the editor.

## I - INSERT

The I command lets you insert one line at a time between any two existing lines in the file. You may use the I command to insert a blank line or a line of text.

**Form:**      {*n*} |

**Operand:**

- n*      enter the number of the line where you want to insert a new line. You may omit the line number to insert a line at the current line.

**Explanation:**

You may use the I command to add a line of text or a blank line anywhere in a file. For example, to insert a line at line 5 you may enter:

5 |

When you press ENTER, the control line displays the prompt "Insert Line:" You may now enter text on the command line that will appear for the inserted line. If you want the inserted line to be blank, make no entry and press ENTER. Remember that the inserted line will not appear on the text screen until you re-display the file.

When you insert a line at an existing line, the existing line and all subsequent lines are moved down one line in the file, and are renumbered in sequence.



## L - LIST

The L command lets you display the current text in a file and specify the first line to appear on the text screen. You may use the L command to move the text screen display upward or downward from any point in the file.

**Form:** {n} L

**Operand:**

- n enter the number of the line where you want the display to begin. You may omit the line number to begin the display with the current line.

**Explanation:**

The L command lets you display the file on the text screen in 22-line increments. The display will include any changes you've made to the file. You may enter the first line to display. For example:

15 L

This command displays 22 lines of the file beginning with line 15. If you enter L without an operand, the text screen displays 22 lines of the file, beginning with the current line of control.

## MOS Editor - ED

---

### M - MOVE

The M command moves a line or a block of lines to a new location in the file. You can move any number of lines with a single command. All lines affected by the move are renumbered in sequence.

**Form:** {f l} d M

#### Operands:

- f enter the number of the first line to be moved.
- l enter the number of the last line to be moved. To move only one line, enter the same line number for both the first and last line number operands.
- d enter the number of the destination line for the move. To move only the current line, enter the destination line and no other operands.

#### Explanation:

If you want to move only one line, you must enter that line number as both the beginning and ending lines to be moved. If you do not, the move will not occur. For example:

18 18 2 M

If you want to move only the current line, you can omit the beginning and ending line numbers. In the previous example, if line 18 were the current line, you could enter:

2 M

When moving lines, the destination line number must be outside the range of lines to be moved or the move will not occur. If you specify a destination line number that is beyond the last line of the file, the moved lines will be placed immediately following the last line in the file. If the destination line exists in the file, the moved line(s) are inserted between the destination line and the line preceding the destination line. All line numbers affected are renumbered in sequence.



## P - PRINT

The P command lets you send the content of the file to the printer. The file prints in 60-line pages. You can print the content of the entire file, or specify a range of lines to print.

**Form:** { {f} / } P

### Operands:

- f enter the first line to print. If this is the only operand entered, this line and all lines to the end of the file will print.
- / enter the last line to print.

### Explanation:

If no operands are entered in the command line, the entire file is sent to the printer. When you print a file, the current printer device is used. If the MOS Print Spooler is active, output is intercepted and sent to a print file in the Spooler subdirectory. The print file may then be printed from the Spooler.

You may set an environment variable before loading the MOS Editor to direct output to any device MOS recognizes. The environment variable for the MOS Editor is LSTDEV. If this has been specified, output is directed to the device that is named. If you want to print the text to a disk file, then you must set LSTDEV to a filename. For example:

```
SET LSTDEV=prnfile.edt
```

Any text you print will be output to the file named prnfile.edt.

## MOS Editor - ED

---

### Q - QUIT

The Q command ends the current editing session and displays a prompt that lets you select whether or not to save any changes, or return to your document and continue.

**Form:**      Q

#### **Explanation:**

If no changes were made to the current file, invoking the Q command ends the editing session without writing any output. If changes were made to the file, invoking the Q command displays the following prompt:

Changes have been made! Save changes (Y/N/ESC)

You must respond with a Y if you want to save your changes. Responding with N ends the editing session without saving, and leaves the original input file unchanged.

Pressing ESC allows you to return to your document and continue editing.

If you used the F command during the editing session to change the file name for output, changes are saved in the new output file. If you select not to save changes, the original input file remains unchanged, and nothing is written to the disk.



## R - REPLACE

The R command lets you replace an existing string of text with a new string. You may replace every occurrence of the string in a file, or only replace the string within a specific range of lines. The ? operand displays a prompt with each match so you can selectively replace a string. You may press ESC to end this command at any time.

**Form:** {{ {f} / } ?} R

### Operands:

- f enter the number of the first line where you want the replace to begin.
- / enter the number of the last line to include in the replace. If no ending line number is specified, the replace occurs through the end of the file.
- ? enter a ? to display the prompt that lets you selectively replace the string.

### Explanation:

You may enter the R command without any operands to perform the search and replace on the entire file. A "Search for:" prompt appears in the control line. Enter the string to replace. A "Replace with:" prompt then appears in the control line. Type the replacement string. The string match is case sensitive, upper and lower case letters are not interchangeable.

You may want to include the ? operand to verify what is being replaced. A "Found: Change Y/N" prompt appears with each match. Any entry other than Y ignores the match. To perform the search and replace on a specific range of lines, enter the first and last number of the lines to include. For example:

10 35 R

Be careful how you enter replacement strings. If you search for "is" and replace it with "was", the string "this is" becomes "thwas was". You may want to enter spaces in the string, such as " is ".

# MOS Editor - ED

## S - SEARCH

The S command locates a string of text that matches a string you specify. You may search an entire file, or only a specific range of lines in a file. Unless you include the ? operand in the command, the search ends with the first match found.

**Form:** {{f} {l} ?} S

### **Operands:**

- f enter the number of the line where you want the search to begin.
- l enter the number of the last line to search. If no ending line number is specified, the search occurs through the end of the file.
- ? enter a ? to continue the search for all matches.

### **Explanation:**

You may enter the S command without any operands to perform the search on the entire file. When you invoke the S command, a "Search for:" prompt appears in the control line. Type the string to replace and press ENTER. The string match is case sensitive, upper and lower case letters are not interchangeable.

Unless you include the ? operand, the search ends with the first match found. When the ? is included, a "Found: Continue Y/N" prompt appears in the control line with each match. Any entry other than Y ends the search. To search only a specific range of lines, enter the first and last number of the lines to include. For example:

5 9 ? R

Note that if you enter "here" as the search string, a match is found with "where". You may want to enter spaces in the string, such as "here".



## V - VISUAL MODE

The V command allows you to switch out of the Command mode and into the Visual Editing Mode.

**Form:**      V

### Explanation:

To switch to the Visual mode from the Command mode, enter the V command on the command line.

When you switch to the Visual mode, the current file displays on the text screen without line numbers, and the cursor is positioned at the beginning of the first line of the text screen display. See [Editing in Visual Mode](#) for information on using this editing mode.

Press ESC to return to the Command mode. You must be in Command mode to save or exit the file.

## MOS Editor - ED

---

### W - WRITE

The W command writes the current text to a file, and leaves the file open so you can continue editing.

**Form:**      W

#### **Explanation:**

The W command writes the current text back to the original input file. If you specified a different output file name with the F command, then the current text is written to the new file, and the original input file is not changed.

You may issue a W command at any time during your editing session. The output is permanently written to the designated file. If you end the editing session with the Q command and select not to save changes, the output file contains the text last written with the W command.



## X - EXECUTE MACRO

The X command executes a defined macro.

**Form:**      X

### **Explanation:**

Entering X on the command line will execute the macro that you defined in the Visual mode. See Using Macros for information on the macro feature of ED.

After macro execution the cursor will be positioned at the right end of the control line. Use the cursor keys to position the cursor at the desired location in your file to continue editing.

## MOS Editor - ED

### Editing in Visual Mode

To switch to the Visual mode from the Command mode, enter the **V** command on the command line and press **ENTER**. From the Visual mode, you may press the **ESC** key to return to the Command mode.

When you switch to the Visual mode, the current file displays on the text screen without line numbers, and the cursor is positioned at the beginning of the first line of the text screen display. For example:

```
---File: MYFILE.TXT-----ROW-1 COL-1---INSERT-
This is an example of a file created with
the MOS Editor, ED, as it will appear
on your screen.
```

The control line now displays the row and column numbers of the cursor position. If you are creating a new file, the cursor is positioned at row one and column one and the screen will be blank. You are ready to begin entering text in the file.

If the file exists, you may use the Visual mode editing keys to move the cursor to any position within the file. The **ESC** key switches control back to the Command mode. You must be in the Command mode to exit the file.

Many editing keys are available in the Visual mode, and are explained following these special notes about using the Visual mode:

- Each line may contain a maximum of 80 characters.
- If you type beyond the last position of a line, the text automatically wraps to the next line.
- You may press **ENTER** to end the current line and move to the next line.



- While in Visual mode, you may press **CTRL** and **Q** to display a help screen of the available Visual mode editing keys.
- **ESC** switches control to the Command mode.
- Text on the screen automatically scrolls if you continue entering text beyond the 22nd line on the screen.
- You may press the **INS** key to toggle the insert mode on and off. When the insert mode is on, the word "INSERT" displays in the control line.

# MOS Editor - ED

## Visual Mode Editing Keys

The MOS editing keys in Visual mode and their functions are:

↑	Moves the cursor up one line at a time. If you move beyond the first line on the text screen, the file scrolls down one line. However, you cannot scroll beyond the first line of the file.
↓	Moves the cursor down one line at a time. If you move beyond the last line of the text screen, the file scrolls up one line. However, you cannot scroll beyond the last line of the file.
←	Moves the cursor to the left, one character at a time. The move is non-destructive and will not erase any text on the line.
→	Moves the cursor to the right, one character at a time. The move is non-destructive and will not erase any text on the line.
BkSp	Deletes the character to the left of the cursor, moving all characters to the right of the deleted character left one space.
DEL	Deletes the character at the cursor, moving all characters to the right of the cursor left one space.
END	Moves the cursor to the last character of the line.
ESC	Exits the Visual mode and returns to the Command mode.
HOME	Moves the cursor to the first character of the line.
INS	Toggles the insert mode on and off. When insert is on, you may insert text anywhere in the file. When insert is off, you will overwrite any current text.
PgDn	Scrolls the display towards the end of the file in 22 line increments.



PgUp	Scrolls the display towards the beginning of the file in 22 line increments.
CTRL PgDn	Displays the last 22 lines of the file.
CTRL PgUp	Displays the first 22 lines of the file.
CTRL A	Lets you assign graphic characters to function keys for inserting in text. See the Command mode explanation of the A command for more detail.
CTRL D	Duplicates the current line on the next line.
CTRL L	Moves control to a specified line in the file.
CTRL N	Splits the current line at the cursor position, and moves characters to the right of the cursor down to the next line.
CTRL O	Turns macro definition keystroke saving ON and OFF.
CTRL P	Sends the text in the current file to the printer, or to an alternate device. See the Command mode explanation of the P command for more detail.
CTRL Q	Displays the Visual mode help screen.
CTRL R	Searches and replaces text strings starting at the current line through the end of the file. A prompt appears with each match so you can selectively replace strings.
CTRL S	Searches for text starting at the current line through the end of file. A prompt appears with each match so you can selectively continue the search.
CTRL W	Writes the current text to the designated output file.
CTRL X	Executes previously defined macro.
CTRL Y	Deletes the current line no matter where the cursor is positioned in the line, and moves subsequent lines in the file up one line.

## Using Macros

A macro feature is available with the Editor. Macros must always be defined in the Visual Mode. A macro can be defined from within any file you are currently editing. If you are not already editing a file and want to define a macro, enter the editor using any file name you desire since the file does not have to be saved to define a macro.

The Visual Mode editing keys CTRL-O tell the editor to start/stop saving keystrokes for the macro you want to define. All keystrokes are saved to the file ED.KEY in the current directory. Entering CTRL-O once will toggle keystroke saving ON. (When on, the word CAPTURE appears at the right side of the command line. When off, the word CAPTURE is not shown.) With CAPTURE on, enter the text/command combination that you want for your macro. When finished, enter CTRL-O again to toggle keystroke saving OFF, ending your macro definition.

To execute a macro in the Visual Mode, position the cursor at the point in the file where you want the macro and enter CTRL-X. After macro execution the cursor will be positioned at the right end of the control line. Use the cursor keys to position the cursor at the desired location in your file to continue editing.

To execute a macro in the Command Mode use the "X - Execute Macro" command. Entering X on the command line will execute the macro that you defined in the Visual mode. The macro will be inserted into your file at the "current line" as defined by CL= in the control line.

---

# Chapter 8: DEBUG

---

Introduction . . . . .	8-3
.DEBUG Break Points . . . . .	8-4
Entering .DEBUG Commands . . . . .	8-4
.DEBUG Commands . . . . .	8-6
A - ASSEMBLE . . . . .	8-7
AU - ASSEMBLE - UNASSEMBLE . . . . .	8-9
BC - CLEAR BREAKPOINT . . . . .	8-10
BD - BREAKPOINT DISABLE . . . . .	8-11
BE - BREAKPOINT ENABLE . . . . .	8-12
BL - BREAKPOINT LIST . . . . .	8-13
BS - BREAKPOINT SET . . . . .	8-14
C - COMPARE . . . . .	8-15
C? - DEBUG CONFIGURATION . . . . .	8-16
CO - CHANGE CONSOLE . . . . .	8-17
D - DUMP . . . . .	8-18
E - ENTER . . . . .	8-19
F - FILL . . . . .	8-21
G - GO . . . . .	8-23
H - HEX . . . . .	8-24
I - INPUT . . . . .	8-25
L - LIST . . . . .	8-26
M - MOVE . . . . .	8-27
N - NAME . . . . .	8-28
O - OUTPUT . . . . .	8-29
P - PROCEED . . . . .	8-30
Q - QUIT . . . . .	8-31

DEBUG

# DEBUG

---

R - REGISTER . . . . .	8 - 32
S - SEARCH . . . . .	8 - 33
T - TRACE . . . . .	8 - 35
U - UNASSEMBLE . . . . .	8 - 36
V - VERIFY . . . . .	8 - 37
W - WRITE . . . . .	8 - 38
7 - COPROCESSOR . . . . .	8 - 40
\ - SWAP SCREEN . . . . .	8 - 41
! - SHELL . . . . .	8 - 42
" - PAUSE . . . . .	8 - 43
:: - DELAY . . . . .	8 - 44
;* - REMARK . . . . .	8 - 45
? - HELP . . . . .	8 - 46
Using DEBUG Commands . . . . .	8 - 47



## Introduction

.DEBUG is an advanced tool that lets you examine the execution of a program in memory. If you write new programs, or modify existing programs, .DEBUG lets you examine values in memory as the program executes, and search for any problems or "bugs".

.DEBUG acts as a "window" into the contents of your computer's memory. It may also be used to create a controlled setting in which to execute a program, and examine the execution, one instruction at a time.

For example, you may instruct .DEBUG to execute an assembled or compiled program, and halt the execution at specific locations in the program. When the program halts, you are then switched to the .DEBUG editing mode. The editing mode lets you issue .DEBUG commands to display the contents of memory, load information into specific addresses and unassemble program instructions. There are many commands in .DEBUG that let you edit a program. This gives you the ability to make changes to a program, and then test the results before making the changes permanent.

For example, .DEBUG lets you unassemble instructions at a specific location in a program to make changes or insert a patch. You may then test the change without reassembling the entire program. This gives you the ability to test your programs before the changes become permanent.

DEBUG

# DEBUG

## .DEBUG Break Points

The locations where you may halt processing are called "break points". These are instruction locations in the program. .DEBUG lets you set both hard and soft break points. Hard break points are retained in memory for the length of the .DEBUG session or until they are specifically cleared. Soft break points are executed only one time during a specified command. You may set a maximum of 10 hard break points and 10 soft break points at any given time in a .DEBUG session.

When you run .DEBUG, you are actually executing two programs; the program you are debugging, and the .DEBUG program itself. Using the \ command (SWAP), you may swap the display between the input/output of .DEBUG and the application being .DEBUGged.

If you have the multi-user versions of MOS, you may invoke .DEBUG at the main console to view the output of the program you are executing. At the same time, you may send the output of the program you are debugging to a workstation's terminal. See the CO command in .DEBUG for information on how to set this up.

## Entering .DEBUG Commands

.DEBUG commands consist of one or two characters, which may be followed by operands. The command entry itself is not case sensitive. You may enter a .DEBUG command at the prompt, which is ]. .DEBUG will not act on it until you press ENTER.

Operands, as they are presented in the command form, may appear inside braces { }. This means that they are optional. The following terms are used for operands in presenting the .DEBUG command forms:

*addr* the full address including the segment and offset or just the offset portion of the address. If the segment is not given, the following defaults will be used. All instructions default to the CS register. All data defaults to the DS register. BP and SP default to the SS register. ES must be specified if it is to be used.



All commands that use *addr* as an operand will default to the segment previously specified in the last .DEBUG command statement, if the segment is not included as part of the address.

<i>begin</i>	the beginning address of a range of memory.
<i>count</i>	the number of logical disk sectors.
<i>end</i>	the ending address of a range of memory.
<i>filename</i>	the name of the file, including its extension.
<i>hex4</i>	a four-digit hexadecimal value.
<i>length</i>	the size of the block of data in hexadecimal.
<i>list</i>	a series of two-digit hexadecimal numbers or character strings, or a combination of both.
<i>n</i>	breakpoint, may be either hard or soft.
<i>portaddr</i>	the hexadecimal address of an I/O port.
<i>register</i>	the two-character name of a valid word register.
<i>sector</i>	a beginning logical disk sector number.
<i>string</i>	a group of characters or hexadecimal numbers.
<i>data</i>	a character string or a byte string.

If you enter a command form that is wrong, .DEBUG will prompt you with:

\*Above line contains a syntax error

# DEBUG

---

## .DEBUG

Used to invoke .DEBUG.

**Type:**      Extrinsic.

**Form:**      .DEBUG *{filename operands}*

### **Operands:**

*filename*      if the file resides on a different drive and directory than that which is current, you must include the drive and directory as part of its name.

*operands*      those operands that are required to run the program such as drive specifications or data files which are required for program execution.

### **Explanation:**

You may automatically load a specific program into .DEBUG at the CS:100 address by including the program name with the .DEBUG command. For example:

```
.DEBUG test.com
```

Note that if this were an .EXE file, it would automatically be loaded at the address specified in the header of the file.

You may load a program at a specific address if you enter .DEBUG without a file name, use the name command (N) to name the input file, and then use the load command (L) with a specific address at which to load the named input file. If you do not specify an address with the L command, .DEBUG loads the program at the default address of CS:100. Files with an .EXE extension do not load at address CS:100; they are loaded as specified in the header of the file.

Once a program is loaded in .DEBUG, there are many commands that let you examine and edit execution of the program. The commands in .DEBUG are explained on the following pages. The sample .DEBUG session presented after the commands may help you better understand how to use .DEBUG.



## A - ASSEMBLE

The A command lets you create instructions in memory and assemble them on the fly without having to reassemble your entire program. Use A to create patches or to test different values. Note that these modifications will be retained in memory for the current .DEBUG session. If you want them to become a permanent part of the program, you must write the program to disk.

**Form:**      A {addr}

**Operand:**

*addr*      specify the address at which you want to start assembling instructions. Note that the address must be separated from the command by a delimiter such as a space or comma.

**Explanation:**

If you enter A with no parameters, .DEBUG defaults to the address of the current CS:IP unless A has been used previously, then it will start where it last left off. You may then add instructions. Each time you press ENTER, the new instruction is accepted. Once .DEBUG assembles an instruction into memory, it moves to the next position where you may add instructions. Press ENTER again to escape from "assemble" mode.

For example, to enter instructions type:

a 123

When you press ENTER, MOS will display the current address:

EF05:0123

## DEBUG

EF05 is the value of the CS register. CS may vary from session to session. You may then enter the instruction:

EF05:0123 MOV AX, 0201

Press **ENTER** again and MOS displays the address for the next instruction in the program.

EF05:0126



## AU - ASSEMBLE - UNASSEMBLE

The AU command lets you assemble an instruction into a program, and immediately unassemble it. You may use this command to clarify instructions you've added.

**Form:** AU {addr}

**Operand:**

addr enter the address at which you want to add and assemble the instruction.

**Explanation:**

If you do not enter an address, .DEBUG defaults to the address at which any previously entered commands finished executing. For example, if you wanted to assemble instructions and unassemble at address 123, you could enter:

```
au 123
```

.DEBUG displays the value of the register and echoes the address. For example:

```
EF05:123
```

You may then enter your instruction set, such as:

```
EF05:123 MOV AX, 0201
```

When ENTER is pressed again, .DEBUG assembles the instruction and then unassembles it, echoing what you entered originally. As in:

```
EF05:123 MOV AX, 0201
EF05:123 B8 01 02      MOV AX, 0201
EF05:126
```

# DEBUG

## BC - CLEAR BREAKPOINT

The BC command lets you clear hard break points that have been set previously with the breakpoint set (BS) command.

**Form:** BC *n*

**Operand:**

- n* enter the number of the breakpoint. If you do not know the number of the break point you want to clear, use the BL command to list the break points currently contained in the break point table. Break points are numbered from 0 to 9.

**Explanation:**

BC is used to clear only one break point at a time. You may only clear the break point by its number, for example:

```
BC 1
```

DEBUG will respond:

```
Break point 1 cleared
```



## BD - BREAKPOINT DISABLE

The BD command lets you disable hard break points that were previously set during the current .DEBUG session. Use BD for continued processing without hard break point interruptions.

**Form:** BD

### Explanation:

BD lets you selectively disable hard break points without removing them. When your program returns to processing, hard break points are then ignored until the break point enable command is encountered.

You may use the configuration command (C?) to determine if break points are enabled or disabled. It displays:

Microprocessor Chip on System is an 80386  
Math Coprocessor found on this System.  
Debug Console Port is COMx.  
Breakpoints are Enabled.

DEBUG

# DEBUG

## BE - BREAKPOINT ENABLE

The BE command lets you enable break points.

**Form:** BE

### **Explanation:**

BE lets you selectively turn back on previously disabled breakpoints. Enter the BE command at the point in the program at which you want breakpoints to be recognized.



## BL - BREAKPOINT LIST

The BL command is used to list hard break points. Hard break points are retained in memory for the length of the DEBUG session.

**Form:** BL

### Explanation:

When hard break points are created, they are assigned the next free sequential number. You may have up to 10 hard break points, numbered 0 - 9, in one program. Use the BL command to list the break points that have been set. For example:

BL

If hard break points have been set, a display similar to the following appears.

```
Breakpoint 0 set to 3189:0123  
Breakpoint 1 set to 3189:0126  
Breakpoint 2 set to 3288:1111
```

DEBUG

# DEBUG

## BS - BREAKPOINT SET

The BS command lets you set up to 10 hard break points in a program file. Break points default to being enabled, unless you invoke the break point disable command (BD), which causes .DEBUG to ignore them.

**Form:**      BS {addr}

**Operand:**

*addr*      enter the address at which you want the first break point to fall. The address could also be a two-character register name.

**Explanation:**

.DEBUG permits up to 10 hard break points in one program, numbered 0 - 9. They are maintained in a breakpoint table for the current .DEBUG session. Break points are expressed as an address within a code segment. You may specify the full address, but if you only enter the offset portion, .DEBUG will default to the code segment currently addressed by the CS register.

The break points are numbered consecutively as they are added. If you remove several break points and add new break points, they will be assigned the next available consecutive numbers. You may display the break point table with the break point list (BL) command.

All of the following examples set break points:

```
BS 123:0191  
BS CS:123  
BS DS:DX
```

When ENTER is pressed and the break point accepted, .DEBUG displays:

Breakpoint 1 set to 3F05:123



## C - COMPARE

The C command lets you compare the contents of two ranges of memory. These ranges may overlap, or may be entirely separate.

**Form:**    *C begin end addr*

*C begin L length addr*

### Operands:

*begin*      enter the beginning address of the first memory range.

*end*      enter the ending address of the first memory range.

*addr*      enter the beginning address of the memory range you want to compare. Note that both operands must be separated from the command and from one another by a space. Segment defaults to DS register.

*length*      must be preceded by L; enter the size of the range to compare, in hexadecimal form.

### Explanation:

The C command performs a byte-by-byte comparison. If the contents of memory are identical, they are not displayed. However, if a discrepancy is found, C displays the values that are different. Look at the example that follows:

C 100 L 200 555

DEBUG responds with:

3F05:0400 53 73 3F05:0500

The hexadecimal value of the bytes is displayed after the first address.

# DEBUG

## C? - DEBUG CONFIGURATION

The C? command displays the current configuration of .DEBUG. This display also appears automatically when .DEBUG is invoked. You may use C? to view this display from any point in the .DEBUG session. The configuration display indicates the microprocessor on the system, the math coprocessor, whether or not hard breakpoints have been disabled or enabled and the .DEBUG port address.

**Form:** C?

**Explanation:**

You may use the C? command during a .DEBUG session to tell you if break points have been enabled. When C? is invoked, MOS displays:

Micropocessor Chip on System is a 80386  
Math Coprocessor found on this System.  
Debug Console Port is COMx.  
Breakpoints are Enabled.



## CO - CHANGE CONSOLE

The CO command lets you use a terminal for .DEBUG's input/output from the main console. However, you must first set up the communications port by defining a DEVICE=\$SERIAL.SYS command statement in the CONFIG.SYS file. You must then initialize the port for use with the .MOS SERINIT command. You may then debug a program on the main console, and view any output that the program writes to the video display screen on the terminal.

**Form:** CO *port*

**Operand:**

*port* enter the port number.

**Explanation:**

You may find CO more useful than the swap command (\) because you can actually debug the program and simultaneously see its output.

DEBUG

# DEBUG

## D - DUMP

The D command lets you "dump" the contents of a specific range of memory, and display it on the screen, 16 bytes to a line.

**Form:**      D {begin} {end}

                D begin L length

### **Operand:**

*begin*      enter the beginning address of the range you want to dump.

*end*      enter the ending address of the range. If not entered, .DEBUG defaults to 128 bytes, or about eight lines.

*length*      must be preceded by L; enter the size of the range to dump, in hexadecimal form.

### **Explanation:**

When a range of memory is displayed on the screen it appears in the following form:

```
0040:0000 F8 03 F8 02 00 00 00 00 00-BC 03 00 00 00 00 00 00 .....  
0040:0010 23 44 00 80 02 00 00 00-00 00 32 00 32 00 75 15 #D.....2.2.u.  
0040:0020 67 22 20 39 3E 34 6F 18-75 16 74 14 0D 1C 78 2D g " 9)4o.u.t.x-
```

The first positions contain the address. The positions that follow contain the offset. Note the column to the right. It contains the ASCII characters for the hexadecimal codes that are displayed. Non-ASCII characters are displayed as periods.

If you do not specify a range for the dump, .DEBUG will default to the current memory address and display its contents in eight paragraphs, totalling 128 bytes.



## E - ENTER

The E command has two forms that let you change the contents of memory. One form will let you enter information directly into memory. The other form displays what is already in memory so you can choose whether or not to change it.

**Form:**      E {addr}

                  E {addr} {list}

### Operands:

*addr*        enter the address which contains the information you want to change.

*list*        enter the two-digit hexadecimal value you want to change.

### Explanation:

You may change the contents of an address by entering:

E 400 x

Press ENTER and .DEBUG will display the first byte found at that address.

3F05:0400 CD

If you press the space bar, .DEBUG will display the next byte position. If you enter a new value, .DEBUG will replace the existing value. When an eight-byte boundary is reached, a new line will begin. You may press ENTER to end E at any time. For example, if you entered:

E 100

## DEBUG

.DEBUG would display:

3F08:0100	CD.12	20.33	00.45	00.	00. 00.	00.3	00.4
3F08:0108	00.5	00.	00.6	00.7	00. 00.8	00.	



## F - FILL

The F command automatically fills a range of memory with bytes of data.

**Form:**      *F begin end data*

*F begin L length data*

### **Operands:**

*begin*        enter the beginning address of the range you want to fill with data.

*end*        enter the ending address.

*length*        must be preceded by L; enter the length of the range you want to fill, in hexadecimal form.

*data*        enter the data with which you want to fill the range. Data may be a text string or a byte list.

### **Explanation:**

The F command will automatically fill a specified range of memory with one single command entry. For example, if you wanted to fill the address range from 400 to 430 with a text string, you could enter:

*F 400 430 "data"*

Press ENTER. When the .DEBUG prompt returns, the range has been filled with data. You may view this fill with the dump command (D) by entering:

*D 400 430*

## DEBUG

.DEBUG displays:

```
3F01:0400 64 61 74 61 64 61 74 61-64 61 74 61 64 61 74 61  dataadatadatadata  
3F01:0410 64 61 74 61 64 61 74 61-64 61 74 61 64 61 74 61  dataadatadatadata  
3F01:0420 64 61 74 61 64 61 74 61-64 61 74 61 64 61 74 61  dataadatadatadata  
3F01:0430 64 61 74 61 64 61 74 61-64 61 74 61 64 61 74 61  dataadatadatadata
```

Had you wanted to enter the range with a byte string, you could have typed:

```
F 400 430 00 02 07
```



## G - GO

Each G command lets you set a soft break point or a series of break points. You may set more than one breakpoint on the same command line, up to a maximum of 10. However, the break points will execute one at a time. Soft break points are not retained in memory and disappear after processing.

**Form:**      G {=addr} {addr}...

### Operand:

=addr      enter the address at which you want execution to begin.

addr      any subsequent addresses entered become soft break-points.

### Explanation:

If you type G, with no address, execution will begin at the current CS:IP and continue through the end of the program. You may set more than one soft breakpoint on the same command line by entering:

G=123 126

.DEBUG will begin execution at CS:0123, then stop and execute it again until the 126 address is encountered. Although you may enter more than one soft breakpoint on a line, they will only be executed one at a time.

DEBUG

# DEBUG

## H - HEX

The H command lets you add and subtract hexadecimal values. You may want to use this command to gauge how far away one location is from another.

**Form:**     H *hex4 hex4*

### **Operands:**

*hex4*       enter the four-digit hexadecimal value you want to add to or subtract from.

*hex4*       enter the four-digit hexadecimal value that is to be added or subtracted.

### **Explanation:**

The H command is useful for calculating addresses. Say, for example, you know the beginning address of a table in a program and how long the table is. You could then use H to calculate the ending address by adding the length of the table to the beginning address. So you would enter:

H 29CE 300

MOS would display:

2CCE 26CE

The first value is the result of adding the length of the table to the beginning address. The second value is the result of subtracting the length of the table from the beginning address.



## I - INPUT

The I command retrieves and displays the one-byte value from the specified port.

**Form:**      I *addr*

**Operand:**

*addr*      enter the address of the port.

**Explanation:**

You may find the I command useful in testing input from ports. For example, if you wanted to read the value in COM1, you could enter:

I 03F8

The port address of COM1 is 03F8. .DEBUG will then display, the hexadecimal value from the port.

DEBUG

# DEBUG

## L - LOAD

The L command loads a program into memory. You may also use L to load the contents of a disk sector or a specific record in a file.

**Form:**     L {addr} {drive sector count}

### **Operands:**

**addr**       enter the address at which you want to load the program or record.

**drive**      used in loading logical disk sectors. Enter the value for the drive; 0 for A:, 1 for B:, etc.

**sector**     enter the starting logical disk sector number. The number may range from 0 to FFFFFF. If the number is larger than FFFF, it must be entered as a double word quantity with a colon separating the high and low words.

**count**      enter the number of logical disk sectors you want to load. Can't be larger than 80 Hex.

### **Explanation:**

You may use L to load a program into .DEBUG after naming the input file with the name (N) command. You may also reload a program that has completed execution and was previously loaded for a .DEBUG session.

You may also use L to load a logical disk sector, such as the boot sector on a disk. For example:

```
L 100 0 0 1
```

This command entry loads the boot sector from the disk in drive A. If you don't specify a segment address, .DEBUG defaults to CS:100. You may display this sector with the dump (D) command.

To load a sector number larger than FFFF, say 123456 (Hex), you would enter:

```
L 100 2 12:3456 1
```



## M - MOVE

The M command moves a string of bytes in memory.

**Form:** M *begin end addr*

M *begin L length addr*

### Operands:

*begin* enter the beginning address of the range of values in memory you want to move.

*end* enter the ending address of the memory range.

*addr* the starting address of the destination to which you want to move the values.

*length* preceded by L; enter the length of the range you want to move, in hexadecimal form.

### Explanation:

You may move values from a specific range of memory into a new range of memory. For example:

M 100 200 400

In this example, the contents of the range from 100 to 200 are moved to the 400 address. You could then use the dump (D) command to display the contents of the new address.

# DEBUG

## N - NAME

The N command lets you specify the filename of the program you want to .DEBUG. You may include an optional second filename to be passed to the program being debugged.

**Form:**      N *filename* {*filename*}

### **Operands:**

*filename*    enter the name of the input file to be loaded into memory for a DEBUG session. N can also be used to change the name of the output file during a debug session.

*filename*    enter the name of the file on which the executable file (the program being debugged) is to perform.

### **Explanation:**

When invoking .DEBUG, you may use N to name the file to debug, and the files to be executed as operands of the program to .DEBUG. For example:

```
N TEST.COM DATATEST.DAT
```

In this example TEST.COM is the program to be debugged. DATATEST.DAT is the file to be used by the executable file TEST.COM while it is being debugged.

Although you may use N to name an output program file, nothing will be written to it unless you use the write command (W).



## O - OUTPUT

The O command is used to send a value to an I/O port.

**Form:**       *addr*

**Operand:**

*addr*      enter the address of the port to which you want to send a value.

**Explanation:**

The O command will send specified data to a port. For example, you may type:

03F8 a4

When ENTER is pressed, .DEBUG returns to the prompt.

DEBUG

# DEBUG

## P - PROCEED

The P command executes the program until the next instruction or a hard breakpoint is encountered. This lets you "skip" over calls, interrupts, repeated strings or string moves.

**Form:**      P {=addr} nnnn

### **Operands:**

=addr      enter the first address at which you want to start executing the program. The program will execute the instruction found at that address and then stop.

nnnn      enter the number of times the proceed will occur, in hexadecimal form.

### **Explanation:**

You may use P to execute a program to a specific point. For example:

P 128



---

**Q - QUIT**

The Q command may be invoked at any point to exit the .DEBUG session. Changes made to the program being debugged will NOT be saved to disk unless they are written back to the input file with the write command (W).

**Form:**      Q

**Explanation:**

Use Q to end a DEBUG session at any point.

# DEBUG

## R - REGISTER

The R command displays the contents of a register.

**Form:**      R *{register}*

R F

### Operands:

*{register}*    enter the name of the register whose contents you want to display.

F                displays the flags register. Mnemonics are:

- O - overflow
- D - direction flag
- I - interrupt enabled
- S - sign
- Z - zero
- A - auxiliary carry
- P - parity
- C - carry

### Explanation:

Flags are either on or off. When F is invoked, a display similar to the following appears:

```
ODISZAPC  
0 0 0 0 0 0 0
```

In this example, no flags are set. An entry of 0 in the column under the single-character flag mnemonic turns the flag off. An entry of 1 turns the flag on.

Pressing ENTER will set the flags as you've specified.



## S - SEARCH

The S command lets you search a range of memory for specific bytes of data or a specific string of text.

**Form:**      S {begin} {end} 'data'

                  S addr L length

### Operands:

*begin*        enter the beginning address of the range of memory to be searched.

*end*        enter the ending range of memory.

'*data*'        enter the data or text string for which you want to search. Text strings should be enclosed in single quotes.

*addr*        enter the beginning address of the range of memory to be searched.

*length*        must be preceded by L; enter the size of the range to search, in hexadecimal form.

### Explanation:

You may enter a range of memory to be searched for a specific string. If you only enter a beginning address at which the search is to start, .DEBUG will search through the end of the file. When the string has been found, .DEBUG will display its position. For example if you entered:

S 300 500 'COM'

DEBUG

## DEBUG

---

.DEBUG would display:

```
3F05:0401  
3F05:0501
```

These are the two positions that are occupied by the string, 'COM'. Had you wanted to search for a non-text string, such as a byte list, you could have entered:

```
S 300 500 04 1E 92
```



## T - TRACE

The T command executes the specified number of instructions, with a display of each, and then stops. If the instruction set includes a call, it will jump to that call. Break points are ignored.

**Form:**      T [=addr] nnnn

**Operands:**

=addr      enter the starting address at which you want the program to execute.

nnnn      enter the number of instructions to be executed, in hexadecimal form.

**Explanation:**

When you enter a T command such as:

T 102

.DEBUG will execute 258 instructions and then stop.

DEBUG

# DEBUG

---

## U - UNASSEMBLE

The U command lets you unassemble instructions in memory.

**Form:**      U *{begin} {end}*

                U *begin L length*

### Operand:

*{begin}*      enter the beginning address of the range of the instruction set you want to unassemble.

*{end}*      enter the ending address. If no ending address is entered, .DEBUG defaults to 32 bytes.

*length*      must be preceded by L; enter the size of the range to unassemble, in hexadecimal form.

### Explanation:

When U is invoked for a specific address, MOS unassembles the code and displays any ASCII characters associated with it. For example:

34FE: 05A7	B800F0	MOV	AX,F000
34FE: 05AA	8EC0	MOV	ES,AX
34FE: 05AC	26	ES:	
34FE: 05AD	A0FEFF	MOV	AL,FFFF



## V - VERIFY

The V command lets you verify that a character string or a byte string occupies a specific address. This command is used with patch files to verify data before allowing the patch to occur.

**Form:**      *V addr data*

### Operands:

*addr*      enter the address of the data you want to verify.

*data*      enter the data you want to verify.

### Explanation:

If the specified data is not found at the address, .DEBUG returns an ERRORLEVEL 1, indicating that the verification was unsuccessful. For example, a patch file may contain the following command statements:

```
V CS:1250 '1.00'  
E CS:1400 'OD 14 15'  
W  
Q
```

The V command statement verifies that 1.00 is at address :1250. If the data is found, a new string, OD 14 15, is entered with the E command. The file is then written back to disk.

If the verification is unsuccessful, returning an ERRORLEVEL 1, .DEBUG will automatically exit the patch, just as if a Q command was encountered.

# DEBUG

## W - WRITE

The W command may be invoked at any point during the .DEBUG session. It is used to write any changes you've made back to the input file. You may also use W to change the contents of a disk sector or specific record in a file.

**Form:**      *W {addr} {drive sector count}*

### **Operands:**

- addr*      enter the address in memory from which you want to write the output file.
- drive*      used in loading logical disk sectors. Enter the value for the drive; 0 for A:, 1 for B:, etc.
- sector*      enter the starting logical disk sector number. The number may range from 0 to FFFFFFF. If the number is larger than FFFF, enter it as a double word quantity with a colon separating the high and low words.
- count*      enter the number of logical disk sectors to load. Can't be larger than 80 Hex.

### **Explanation:**

W lets you write any changes made during the .DEBUG session to the output file. When W is invoked, .DEBUG displays:

Writing 0000001AH bytes of program TEST.COM



When a file has completed execution, DEBUG assumes the output file is the original input file. Unless its name has been changed, or a new file specified with the name (N) command, .DEBUG will overwrite its contents with the W command. When W is invoked, .DEBUG reads the CX register for the length of the file and uses that value to write the file to disk.

For example, to change the contents of a boot sector of a disk (sector 0), you would enter:

```
W 100 0 0 1
```

To write to sector number 123456 (Hex), you would enter:

```
W 100 2 12:3456 1
```

# DEBUG

## 7 - COPROCESSOR

You may invoke the 7 command from any point in a .DEBUG session to display the contents of the registers of a math coprocessor you may have on your system.

**Form:** 7

**Explanation:**

When 7 is invoked MOS displays an eight-register display if a math coprocessor is found on the system. The display is similar to the following:

Control=037F Status=4100 Tag=FFFF Instruction=0000:0000 Data=0000:00

Reg	Sign	Exponent	Significant
R0	1	2A15	875C A160 CD85 DECF ADB4 ESA4 01AD
R1	0	0C00	AA45 FFB5 9581 9795 9000 A2FS A597
.			
.			
.			

This command is useful because it allows you to view the registers of the math coprocessor during execution of a program.



## \ - SWAP SCREEN

You may use the \ command to switch to the screen output of a program as it executes under .DEBUG. The \ command is only recognized when issued from the .DEBUG console, it is not valid when invoked at a workstation.

**Form:**

**Explanation:**

When a program writes output to the screen, you may use the \ command to view that output from within .DEBUG. Pressing ENTER from the screen containing the output returns you to .DEBUG. For example, if your program wrote output to the screen you could enter:

Which would display:

Now processing

The screen output will appear just as it does during a normal execution of the program. Pressing ENTER will then return you to .DEBUG.

DEBUG

## DEBUG

---

! - SHELL

You may invoke an additional MOS shell during a .DEBUG session by entering the ! command and pressing ENTER.

**Form:**      !

**Explanation:**

The ! command lets you invoke another copy of the Command Processor if the program you are .DEBUGging has finished execution, or if no program is currently loaded in .DEBUG. You may return to .DEBUG by entering .EXIT.



## " - PAUSE FOR A KEY

The " command is used in patch or program modification files to insert a pause into the file when it is invoked. It will halt processing until a key is pressed.

**Form:**      " {text}

**Operand:**

{text}      the " command may be followed by a message to be displayed when the patch file is invoked.

**Explanation:**

You may use the " command to create a pause in the processing of a patch file and display a message.

## DEBUG

---

**: - DELAY**

The : command is used in patch files to halt processing for half a second before executing the next line in the file.

**Form:** :

**Explanation:**

The : command will halt processing for half a second. You may want to use it to let the user of a patch file view any output from processing.



---

; \* - REMARK

Either an \* or ; may be used to indicate remarks in patch files.

**Form:**      ; {text}

\* {text}

**Operand:**

text        enter any text as a message in the file.

**Explanation:**

Use of the \* or ; prevents .DEBUG from trying to interpret the text in a patch file as an instruction set and execute it.

DEBUG

# DEBUG

? HELP

The ? command invokes the .DEBUG HELP screen from any point during a .DEBUG session.

**Form:** ?

**Explanation:**

Use ? to view the following screen:

A Assembler Command	P Proceed Command	BL Break Point List
C Compare Command	Q Quit Command	BS Break Point
D Dump Command	R Register Command	7 Coprocessor Register
E Enter Command	S Search Command	\ Swap Screen Output
F Fill Command	T Trace Command	I Shell to MOS
G Go Command	U Unassemble Command	? Online Help Command
H Hex Command	V Verify Command	C? Debug Configuration
I Input Command	W Write Command	CO External Console
L Load Command	AU Assemble -Unassemble	" Pause for key
M Move Command	BC Break Point Clear	: Delay
N Name Command	BD Break Point Disable	* Comment
O Output Command	BE Break Point Enable	; Comment



## Using the .DEBUG commands

You may find that one of the most common uses you'll have for .DEBUG is to make modifications to existing programs. Review the sample session that follows. It creates a small program that will write characters to the screen using BIOS interrupt 10h. For more information about the interrupts on your computer, refer to its technical documentation.

This sample session is presented to help you familiarize yourself with .DEBUG and with its commands. Each of these commands is explained in greater detail earlier in this chapter.

Invoke .DEBUG by typing:

```
.DEBUG
```

The screen will clear and display the following:

```
PC-MOS Debugger Version #.##
Copyright 1987 The Software Link, Incorporated
All Rights Reserved Worldwide

Microprocessor Chip on System is an 80386.
Math Coprocessor found on this System.
Debug Console Port is COM1
Break points are Enabled

AX BX CX DX SI DI BP DS ES SS SP O DIS Z A PC
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0 0 0 0 0 0 0 0
2183:0100 CD20      INT     20
]
```

DEBUG

## DEBUG

Note the information that is presented in this display. The two-character names of the registers appear across the screen, followed by the single-character flag mnemonics.

The .DEBUG prompt (!) appears on the bottom most line. This prompt will be used throughout this sample .DEBUG session to give you a point of reference. It is not used elsewhere in this chapter.

Type the following instructions. If you choose, you may include the comments.

```
]A 100
2183:0100 MOV AH,0E      ;Use Function 0E
2183:0102 MOV AL,4D      ;Write Character 4DH 'M'
2183:0104 MOV BH,0       ;Set Screen Page to 0
2183:0106 INT 10        ;Issue BIOS interrupt 10
2183:0108 MOV AH,0E
2183:010A MOV AL,4F      ;Write Character 4FH 'O'
2183:010C MOV BH,0
2183:010E INT 10
2183:0110 MOV AH,0E
2183:0112 MOV AL,4E      ;Write Character 4EH 'N'
2183:0114 MOV BH,0
2183:0116 INT 10
2183:0118 INT 20        ;Terminate program
2183:011A               ;Press return to stop assembling
```

Note that the Assembly language instructions appear in the positions after the address. The segment value is dependent on your system environment and may not be the same as that shown in the example.

You may then insert comments, be certain to mark them with the remark command, (:). This program will use the MOS BIOS interrupt 10 to write the characters "MON" to the video display screen.



Look at the ending address of the program you've written -- 2183:011A. You must now set the size of the program in the CX register. This tells .DEBUG how big the file is and where it ends. Set the size to 1A bytes, which is the difference between the beginning address and the ending address, by typing the following:

```
]RCX  
CX 0000  
:1A
```

You must now assign a name to the program with the name (N) command, if you want to save it.

```
]N TEST.COM
```

You may now write the program to disk with the write (W) command. This stores the program for future use.

```
]W  
Writing 0000001AH bytes of program TEST.COM
```

Now execute the program with the go (G) command.

```
]g  
Program Completed Execution
```

To see the output of the program, type the swap (\) command.

```
\
```

DEBUG

## DEBUG

The screen should now show the output that the program writes to the video display screen.

MON

To return to .DEBUG, press RETURN.

Now, set hard break points to examine how the program executes.

```
JBS CS:0108  
Breakpoint 0 set to 2183:0108  
JBS CS:0110  
Breakpoint 1 set to 2183:0110  
JBS CS:0118  
Breakpoint 2 set to 2183:0118
```

Now, run the program again.

```
]g=100
```

As the program reaches each breakpoint, processing will stop and DEBUG will display the registers.

```
AX  BX  CX  DX  SI  DI  BP  DS  ES  SS  SP  ODISZAPC  
0E4D 0000 001A 0000 0000 0000 218A 218A 218A FFFE 00000000  
2183:0108 CD20      INT    20  
2183:0108 B402      MOV AH,0E
```



---

Now enter a swap (\) command to view the results, swapping the screen to see the output being written to the video display screen.

\

The screen should display the following:

M

To continue running the program, type the go command (G).

]g

This process will be repeated at each break point.

Now, exit .DEBUG.

]q

To see the results of your .DEBUG session, type TEST at the MOS prompt.

[C:\]TEST  
MON  
[C:\]

To change the message printed by TEST.COM, restart .DEBUG.

[C:\].DEBUG TEST.COM

DEBUG

# DEBUG

Now, unassemble the program to ensure that it is correct.

```
]u 100 118
2183:0100 B40E      MOV AH,0E
2183:0102 B04D      MOV AL,4D
2183:0104 B700      MOV BH,00
2183:0106 CD10      INT 10
2183:0108 B40E      MOV AH,0E
2183:010A B04F      MOV AL,4F
2183:010C B700      MOV BH,0D
2183:010E CD10      INT 10
2183:0110 B40E      MOV AH,0E
2183:0112 B04E      MOV AL,4E
2183:0114 B700      MOV BH,00
2183:0116 CD10      INT 10
2183:0118 CD20      INT 20
```

At the .DEBUG prompt type in the following:

```
]e 113
```

.DEBUG will respond with the current value at that location. In this example, the value is 4Eh, N. You will change 'N' to an 'S' by entering 53, the hexadecimal value for 'S'.

```
]e 113
218A:0113 4E.53
```

Now save the modified program. Note that the program does not need to be named, since it was loaded into memory when .DEBUG was started.

```
]w
Writing 0000001AH bytes of program TEST.COM
```



Now run the program to see the results of the modification.

```
]g  
Program Completed Execution  
]\  
MOS
```

Now exit from .DEBUG.

```
]q
```

Sometimes patches or modifications for programs are presented as text files, often with the extension .PAT. In order to demonstrate this, a patch file for TEST.COM will be created. It contains the keystrokes to invoke the .DEBUG commands to make the changes to the file.

```
[C:]COPY CON TEST.PAT  
;THIS PATCH FILE CHANGES TEST.COM TO ORIGINAL VALUES  
E 113 4E  
W  
" Press any key to return to system  
Q
```

Press CTRL-Z to end the file. Now apply the patch file to TEST.COM by entering:

```
[C:]DEBUG TEST.COM<TEST.PAT
```

The program will run automatically, taking its input from the TEST.PAT file. It will pause and wait for a key stroke when the " command is encountered. Now run TEST.COM.

```
[C:]TEST  
MON
```

DEBUG

Your display should be similar to the following:

PC-MOS Debugger Version #.##  
Copyright 1987 The Software Link, Incorporated.  
All Rights Reserved Worldwide

Microprocessor Chip on System is an 80286  
Math Coprocessor found on this System.  
Debug Console Port is COM1.  
Breakpoints are Enabled.

AX BX CX DX SI DI BP DS ES SS SP ODISZAPC  
0000 0000 0020 0000 0000 0000 0000 361B 361B 361B FFFE 00000000  
2183:0100 MOV AH,0E

]this patch file changes test.com to original values  
]E 113 4E  
]W  
Writing 00000020H bytes of program TEST.COM  
]"Press any key to return to system  
]Q

---

# **CHAPTER 9:**

## **SECURITY**

---

Introduction .....	9 - 2
How Security Works .....	9 - 2
Initiating User Security .....	9 - 3
Entering User Security Records .....	9 - 4
Security Commands .....	9 - 6
.CLASS .....	9 - 7
.SIGNOFF .....	9 - 9
.SIGNON .....	9 - 10
Assigning Security to Your Items .....	9 - 11
Partition Level Security .....	9 - 12
Directory Level Security .....	9 - 15
File Level Security .....	9 - 17
MOS File Level Security and VERIFY Command .....	9 - 18
Precautions for Security Users .....	9 - 19
Advanced Security .....	9 - 19
The MOS Encryption Key .....	9 - 20
The Master Password Encryption Key .....	9 - 20
Changing the Master Password .....	9 - 22

# Security

---

## Introduction

MOS provides a method for securing the files, directories and partitions on your computer. When you receive MOS, security is not active. If you want to use security, it must be set up for your specific needs. For security to be effective, only the system administrator should set up and maintain security.

MOS security is very flexible, and can be very powerful. You may find that security causes some undesired effects if you don't understand how it works. Before you begin setting up security, be sure you understand how access to files, directories and partitions will be handled. You may want to experiment with security before you set it up for actual use on your computer.

It's a good idea to read through this entire chapter before you begin setting up security. Security can be set up or changed at any time. If you don't think you will need security, then don't set it up on your computer.

## How Security Works

You can implement security for all items on your computer, and for all users. If there are only a few directories or files that need to be secured, you can implement security for only those items. You may also implement security only for a few users who need access to the secured items.

Security is based on 26 security classes that correspond to the letters of the alphabet, A to Z. You may define these classes to be anything you want to use for your computer. A blank or space is also a valid security class to designate that the item is not secured.

You must first define how many classes you want to use, and what each class will represent. One class may then be assigned to each file, directory and partition you create. You must assign a blank or space class to any items you do not want to secure.



A user security file \$\$USER.SYS is set up to control who can access secured items, and the level of access each user has to each class. This file contains a unique record for every user, or only for the users who need access to secured items. In each user record, you may assign one of four levels of access to each class; no access (0), execute only (1), read and execute only (2), or unrestricted access (3).

When you first boot MOS, any user may access any file, directory or partition that is not secured. These unsecured items have a blank class. All secured items are invisible to the unauthorized user as if they did not exist. Such a user cannot see or access any secured items.

In order for a user to access a secured item, the user must invoke a .SIGNON command, and enter a valid user ID and password that exists in the user security file. After signing on, the user is further restricted from accessing files, directories and partitions based on the access level the user has to the class assigned to the secured items.

The access levels for secured items are handled differently at the partition, directory and file level. These differences are explained later in this chapter. First you need to understand how the user security file is set up, and how the MOS security commands are used.

If you already have partitions, directories or files on your computer, the .CLASS command is available to assign or change the class of an existing partition or directory. You may use the .COPY command with the /c operand to assign or change the class of an existing file.

## Initiating User Security

User security is set up and maintained in a file named \$\$USER.SYS. MOS looks for this file at the root directory when user security is active. You may place the \$\$USER.SYS file in a different directory if you include the .USERFILE command in the CONFIG.SYS file so that MOS will know where to find the file.

It is a good idea to set up a record for the system administrator with unrestricted access to all security classes. This will give the system administrator complete control of all secured items.

# Security

You may also want to reserve one security class exclusively for the system administrator. Assign all other users the "no access" level to that class. When the \$\$USER.SYS file is complete, assign it a reserved class with the .COPY command. The .COPY command is explained in the General Commands chapter of this manual. This will restrict any other users from being able to read or change the information in the \$\$USER.SYS file.

You may use the MOS Editor to create the \$\$USER.SYS file and enter the user records. The Editor is explained in a separate chapter of this manual.

## Entering User Security Records

To simplify setting up user records, enter the following comment line as the first line in the file. You will use this line as a format for entering user records. The asterisk in the first position of the line marks it as a comment line.

\* ID, PASSWD, ABCDEFGHIJKLMNOPQRSTUVWXYZ, c

Each subsequent line in the file is a user record. You may want to use the second line of the file for the system administrator's record. As many user records may then be entered in the file as necessary. The following is the information to enter for each record.

- \* ID      the user ID may be up to four alphanumeric characters (A to Z and 0 to 9), and must be followed by a comma as a delimiter. The user ID is not case sensitive during sign on and may contain any combination of upper and lower case characters. The user ID must be unique for each record you set up. You may want to assign a user's name or initials as the user ID. For example:



If the user ID is less than four characters, you may enter spaces. MOS will ignore spaces entered at the beginning or end of the user ID. The comma delimiter may be placed immediately following the user ID, or may be aligned with the comma in the format line. For example:

```
* ID,  
DAVE,  
BOB ,
```

PASSWD the user password may be up to six alphanumeric characters (A to Z and 0 to 9), and must be followed by a comma as a delimiter. The user password may also be entered in either upper or lower case characters as it is not case sensitive during sign on. The password should be unique for each user. For example:

```
* ID, PASSWD,  
DAVE, SYSADM,
```

If the user password is less than six characters, you may enter spaces. MOS will ignore spaces entered at the beginning or end of the password. The comma delimiter may be placed immediately following the password, or aligned with the comma in the format line. For example:

```
* ID, PASSWD,  
DAVE, SYSADM,  
BOB , ADM,
```

ABCDEFGHIJKLM NOPQRSTUVWXYZ

SECURITY

The next 26 positions correspond to the 26 security classes. Each user may have only one access level assigned to each class. A comma delimiter must follow the 26 positions. Enter one of the following access level codes in place of the position for each class:

## Security

- 0 No access
  - 1 Execute only
  - 2 Read and execute only
  - 3 Unrestricted access

For example:

- C Although a user may access several classes, any new files or directories the user creates can only be assigned one class. "c" is the default output class assigned when the user creates files and directories. It follows the 26 classes, and may be enter in upper or lower case. In this example Dave's default output class is A and Bob's is F:

If the default class is left blank, any files or directories the user creates will have a blank or space class. A space class designates no security and the item can subsequently be accessed by anyone. Continue adding user records until the user security file is complete. Remember to assign a security class to this file to keep other users from reading or changing any information in the file.

## Security Commands

MOS contains three commands that are used when user security is active. The .CLASS command lets you assign or change the class of existing directories and partitions. The .CLASS also lets you change the default output class for a user. The .SIGNON and .SIGNOFF commands let you sign on and off of a secured system. The following pages explain these commands in detail. Note that the .COPY command may be used to change the class of a file, and is explained in the General Command chapter of this manual.



## .CLASS

.CLASS lets you assign or change the security class of a directory. You may also use .CLASS to display the current partition class and user default output class. You may only change the partition class and user default output class of the partition you are currently accessing. You must use the .COPY command to assign or change the security class of a file.

**Type:** Intrinsic.

**Form:** .CLASS *dirname c*

.CLASS *pd*

### Operands:

*dirname* enter the name of the directory whose class you want to assign or change.

*c* enter the letter of the new class you want to assign, in either upper or lower case.

*pd* enter the letters of the security class to assign to the current partition (p) and the current user's default output class (d) in either upper or lower case characters. You may enter an asterisk (\*) in either or both positions to display the current partition class and/or default output class. The partition class is originally assigned in the .ADDTASK command. The default output class is originally assigned as the default class of the user who signed on in the partition.

### Explanation:

You may use .CLASS to assign or change the class of a directory, even if you do not have access to that class. However, once you assign a class other than one to which you have access, you cannot access the directory. The following command will change the current class of the MYDIR directory to class B:

# Security

```
.CLASS MYDIR B
```

You may only change the class of the partition you are currently accessing. You may also change the default output class for the current user who signed on in the partition you are accessing. To display the current class assignments, you may enter asterisks for these two operands in the CLASS command. For example:

```
.CLASS **
```

The first asterisk represent the partition class and the second asterisk represents the user default class. This command displays a message similar to the following:

```
Partition Class - S  
Default Output Class - X
```

To change either class, type the new class to assign in place of the asterisk. For example, the following command changes the default user class for this partition to Z.

```
.CLASS *z
```

You may change only the partition class, or you may change both classes at the same time. Simply enter .CLASS followed by the new classes you want to assign as the partition class and the user default output class. For example:

```
.CLASS ab
```

You may delete one or both security classes by entering the underline character (\_) in place of the corresponding asterisk for that operand. For example the following command deletes both the partition security class and the user default output class:

```
.CLASS __
```



## .SIGNOFF

.SIGNOFF lets you exit from the secured mode and return to the system prompt. If you are sharing a workstation with other users, you should always sign off before allowing anyone to use the workstation. If you don't, the new user will be able to access all of your secured files, directories and partitions.

**Type:** Intrinsic.

**Form:** .SIGNOFF

### Explanation:

Entering .SIGNOFF causes the current user to exit the secured mode and return to the system prompt in an unsecured mode. A sign off message appears similar to the following:

ROD signed off.

Once you have signed off, you may still use MOS, but you may only access a file, directory or partition that has a blank class.

Because the keyboard buffer retains key strokes, MOS will clear the buffer when you invoke the .SIGNOFF command. This will prevent any other users from recalling your key strokes and viewing them on the screen.

# Security

## .SIGNON

Before you can access any secured items, you must enter the .SIGNON command at the system prompt to activate security for your partition. This will display the MOS sign on screen where you may enter your user ID and password.

**Type:** Intrinsic.

**Form:** .SIGNON *{user ID}*

### **Operands:**

*user ID* optionally, you may enter your user ID as part of the .SIGNON command.

### **Explanation:**

If you entered your user ID with the .SIGNON command, it appears on the sign on screen as shown in the following example:

```
User Name : ROD  
Password :
```

You will only have to enter your password to complete the sign on. To keep your password confidential, it will not appear on the screen as you enter it on the keyboard. MOS checks the \$\$USER.SYS file to be sure that the user ID and password are correct.

When sign on is successful, a screen similar to the following appears:

```
User ROD has Signed on.  
Partition Class - S  
Default Output Class - S
```



If the sign on is not successful, MOS will prompt "Try again." This will appear if either the user ID or password is incorrect. If the sign on has not been successful after a random number of attempts (between three and seven), MOS will "lock" the workstation and display the following message:

Access denied. Please wait.

The workstation will remain in a "locked" state for a random length of time (between four and 15 minutes). When the workstation is freed, you may then attempt the sign on again.

## Assigning Security to Your Items

There are several ways that you may assign security to a partition, directory or file. A security class may be assigned to a partition as part of the .ADDTASK command. Files and directories may be assigned a user's default class at the time a user creates them. If security is active when a .DISKID command is invoked, the name assigned to the disk is also assigned a security class.

If an existing file does not have security, its class assignment is blank. You may assign a security class to the file with the .COPY command. You may also use the .COPY command to change the current security class of a file. See .COPY command for details.

The .CLASS command lets you assign or change the security class of an existing directory. The .CLASS command also lets you assign or change the security class of the partition that you are currently working in and the default output class of the current user.

Once user security is initiated, it is important to understand how security works at the partition level, directory level and file level. Be sure you understand the following example and explanations before you set up security for your computer.

# Security

---

## Partition Level Security

If your system does not have security, any user may access any partition from any workstation. When a user is working in an unsecured partition, any other user may access the same partition with the partition access keys at another workstation. Both users are sharing the same resources.

If security is active, and a partition has been assigned a security class as part of the .ADDTASK command, a user must first sign on in order to access the partition. The user may then access only the partitions that have a class that the user is allowed to access. If a user attempts to access a partition that has a class the user is not allowed to access, the partition access keys will appear not to work.

- 0 - no access to the partition. If the user has an access level of 0 to the class assigned to a partition, the user cannot access the partition. A user may change the class from within a partition to a class set up for the user with an access level of 0.

The user may stay in the partition, but cannot read or write information. The only thing the user can do is use the partition access keys to move out of the partition. Once out of the partition, the user will not be able to access the partition again.

- 1 - execute only. The user can access the partition, but cannot type information. The user may change the class from within a partition to a class set up for the user with an access level of 1. The user may then only use the partition access keys to move in and out of that partition.
- 2 - read and execute only. Same as access level 1.
- 3 - unrestricted access. The user may access the partition, and may read or change any information to which the user has access.



If a user is working in a secured partition that has a class other users can access, another user may move into the partition. Both users are sharing the resources. The user who moved into the partition can now access all the files and directories that are accessible to the user who signed on in the partition. The second user's original sign on is not effective while he is in this new partition.

You may want to keep users from accessing other users' partitions. You could set up all partitions with a blank class. Each time you boot MOS, all partitions are available to all users. Each user may access a partition in which they will be working. Once in the partition, a user may use the .CLASS command and change the class of the partition to a class that only that user may access. That user may access or create additional partitions if multi-tasking capability is necessary.

You may also control partition access by setting up dedicated workstations for each user. For example, you may want to secure only one or two workstations where accounting functions are run. You must first set up a security class with unrestricted access for each user who will run in the secured mode. Make sure that no other user may access that class.

Then define a partition for each secured user with the .ADDTASK command in the AUTOEXEC.BAT file. In this example, the user, Sara, has a security record that defines unrestricted access to class A. All other users are assigned no access to class A, or are not set up to run in the secured mode. The .ADDTASK command that creates the partition to run at Sara's workstation is set up in the AUTOEXEC.BAT file as follows:

```
.ADDTASK 128,2,A,SARA,PCTERM,1,19200
```

This partition is set up to use 128 bytes of memory, is assigned task number 2, security class A, and a startup file named SARA.BAT. This partition will run on a PC type terminal, connected to port 1, at a baud rate of 19200.

## Security

You must include the .SIGNON command in the startup file (SARA.BAT) so that the signon screen appears on SARA's workstation each time MOS is booted. If you don't include the .SIGNON command, you could never access MOS to sign on to the secured workstation. For example:

```
.SIGNON SARA
```

This will invoke the .SIGNON command and enter SARA for the user ID. SARA will need to enter the password to sign on to the partition at the workstation. Once signed on, no other users will be able to access Sara's partition.

While in partition 2, SARA may invoke additional .ADDTASK commands to create a multi-tasking environment within the partition. SARA may use the partition access keys to switch to a different partition as long as it has a security class that SARA may access.

Because no other users can access class A, and class A is assigned to the partition running on Sara's workstation, no other users can access this partition. The partition access keys for partition 2 will not work at any other workstation.



## Directory Level Security

If your system does not have security, any user may access any directory on your computer. You may want to secure all directories or only a few directories that contain confidential information.

A user's access to a directory is controlled by the level of access the user has to the class assigned to the directory.

- 0 - no access to the directory. If the user has an access level of 0 to the class assigned to a directory, the user cannot access the directory. If the user displays a directory, any directory with a class to which the user has an access level of 0 will not appear on the directory listing.
- 1 - execute only. The user can see directories that are assigned a class to which the user has level 1 access. The user can execute any files in this directory, but cannot read or change the content of any files.
- 2 - read and execute only. The user can access the directory and can read, load and execute files to which the user has access. The user cannot create new files, or execute programs that create new files.

Note that if a user has an access level 2 to the class of a directory and runs an application that for any reason creates a file, the application may report an error such as "directory full." The directory is not really full. MOS cannot create the file because the security access level prohibits writing to the directory.

- 3 - unrestricted access. The user has unrestricted access to the directory, and may access any files in the directory to which the user is allowed access.

## Security

You may assign security at the directory level to prevent unauthorized users from accessing that directory. For example, if you are running an accounting package, you may want to assign class P to the directory where payroll records are kept. Then assign the no access level to class P for all users except the person who runs payroll.

If a directory has a class that is restricted from a user, the user cannot access any files in that directory, even if the files have a class that the user can access. In the same way, a user must be able to access all levels of directories from the root that precede the directory to be accessed. For example:

<b>Directory:</b>	\dirname1	\dirname2	\dirname3	\myfile
<b>Security Class:</b>	A	B	C	D
<b>User's Level:</b>	3	2	3	3

In this example, the directory name to access MYFILE is three levels from the root directory. A user must have access to classes A, B, C and D to be able to access MYFILE. The level of access a user has to each class also determines the level of access the user has to a file.

In our example, the user has unrestricted access to classes A, C and D, but has only read and execute access to class B. The user may read the content of MYFILE, but cannot access MYFILE to make changes even though MYFILE has a class to which the user has unrestricted access. The lowest access level of a preceding directory determines the user's access level to the files.



## File Level Security

If your system does not have security, any user may access any file on your computer. You may want to secure all files or only a few files that contain confidential information.

A user's access to a file is controlled by the level of access the user has to the class assigned to the file. Access may also be restricted by the class assigned to the directory where the file resides.

- 0 - no access to the file. If the user has an access level of 0 to the class assigned to a file, the user cannot access the file. If the user displays a directory, any files with a class to which the user has an access level of 0 will not appear on the directory listing.
- 1 - execute only. All files with a class to which the user has an access level of 1 will appear on a directory listing, and can be run if they are executable programs. However, the user cannot type or load the file with another program.
- 2 - read and execute only. The user can read or execute the file. A user must have an access level of 2 or 3 to a class assigned to a batch file in order to execute the batch file. A user may display the content of a file with the .TYPE command, but cannot edit the file.
- 3 - unrestricted access. The user has unrestricted access to the file.

If you have users who share the same directory, you may want to set security only at the file level. For example, a directory may have a class of A, but the files in the directory have classes of B and C. One user may be set up with unrestricted access to classes A and B, but may have no access to class C. Another user may have unrestricted access to classes A and C, but may have no access to class B.

# Security

**NOTE:** If you are signed on to security and create a file, the file is assigned your user default output class. The file appears in the directory with your user ID, default class, and the time and date of creation and last update.

If you create a file that replaces an existing file, the user ID, creation date and time, and class remain the same. The last update time and date will contain the time and date of the replacement.

When running some applications the replacement may appear not to work. Some applications may rename or delete a file while you are using it, and then create a new file when you save the output. MOS will handle this situation as if the file is a new file and not a replacement or update of the original file.

## MOS File Level Security and VERIFY Command

When using MOS SECURITY for file level security, care must be exercised when using the VERIFY command to check a disk for file allocation errors! ONLY users with "level 3 - unrestricted access" assigned for ALL file classes should be allowed to run VERIFY! (This level of access is usually reserved for someone at the system administrator level.)

**CAUTION:** If the user running VERIFY does not have level 3 access for all file classes, VERIFY will not be able to properly read the disk.

\* WARNING \* If the user runs VERIFY with the /f operand to fix lost allocation clusters, there could be some disk corruption if they do not have level 3 access assigned for all file classes!



## Precautions for Security Users

When you invoke security, every file having a non-blank class is stored in an encrypted state. It is possible for an unauthorized person to create his own \$\$USER.SYS file on a MOS diskette, and then boot your computer from the diskette. If the \$\$USER.SYS security file on diskette contains a record that allows unrestricted access to all classes, the user may sign on from the diskette and access all of your secured files on the hard disk.

If security is important, you may want to keep the main console in a secured location. If your computer has a lock, you may want to keep it locked. Never leave the MOS diskette lying around where anyone may use or copy it.

## Advanced Security

MOS contains an advanced security feature that lets you change the key used to encrypt files. You may use any unique encryption key for your computer. A user could still boot MOS from a diskette and access your secured files on the hard disk. But, because the user would not know your master password to unencrypt the files, secured files would display garbage.

The master password is not to be taken lightly. You may initiate user security without a master password. If you initiate the master password, it is NOT SAVED IN ANY FILES WHERE IT CAN BE LOCATED OR SEEN. The master password becomes part of the encryption key for all secured files created on your computer.

If you forget the master password, it's like crashing your entire computer. You can do nothing, possibly not even boot MOS. And because the master password key is not saved in any files and cannot be unencrypted, WE CAN DO NOTHING TO HELP YOU. Therefore, consider it carefully before choosing to use it.

# Security

## The MOS Encryption Key

Initiating the master password causes MOS to prompt you for the master password encryption key each time you boot MOS. Whatever you enter becomes the encryption key while the computer is active. Because the master password is not saved in any files or anywhere it can be seen, MOS cannot tell if the master password you enter is correct or not.

If you make a typing error or enter an incorrect master password when you boot MOS, any secured files you create will have an encryption key that is different from the encryption key of any existing files. Any secured files previously created with a different key will appear as garbage.

If you decide to use the master password to change the MOS encryption key, initiate it very carefully. Each time you boot MOS and enter the master password key, signon and check a secured file to be sure it does not appear in its encrypted state.

We suggest that you set up a test file with a non-blank security class and use it only for checking the master password when booting MOS. If the test file appears to be encrypted and displays garbage, the master password was incorrectly entered. You should reboot MOS and enter the master password again.

## The Master Password Encryption Key

If you decide to use a master password, it should be initiated before you initiate user security. Be sure that all files have a blank security class so that no files are encrypted prior to the time you initiate the master password key. If you attempt to use previously secured files after initiating a master password, they will appear as garbage and be unusable.

The master password is initiated by creating a file named **\$\$MASTER.SYS** at the root directory. MOS checks for the existence of a **\$\$MASTER.SYS** file at the root directory before checking for the **CONFIG.SYS** file. If MOS does not find a **\$\$MASTER.SYS** file at the root directory, MOS assumes a master password is not used.



You may create the \$\$MASTER.SYS file with the MOS Editor. You may leave the file blank, or enter anything in the file you like. MOS checks only for the existence of a \$\$MASTER.SYS file, and any entries in the file are ignored by MOS.

After creating the \$\$MASTER.SYS file, reboot your system. MOS will now prompt you to enter the master password. Whatever you enter for the master password is used as the encryption key for your files. The display on your screen will be:

ENTER MASTER PASSWORD

Carefully enter up to six alphanumeric characters, A to Z and 0 to 9, for the master password key. The master password is not case sensitive, so your entry may be in either upper or lower case characters. To keep the master password confidential, it does not appear on the screen as you type it.

If you make a mistake while entering your master password, you may backspace over it and re-enter it, or press ESC to completely remove it and then re-enter it.

Be certain the password is entered correctly before pressing the ENTER key. When you press ENTER, the master password becomes part of the encryption key. You must enter the master password exactly as it is entered now each time you boot the computer. It's a good idea to write down your master password and keep it in a safe place.

## Changing the Master Password

It is possible to change the master password for your computer. Before you change a master password, be sure all files have a blank security class so they are not encrypted with the master password key that you are going to change.

You can use the .COPY command with the /c operand to change the security class assigned to a file. When you assign the new copy of the file a blank security class, it is saved in an unencrypted state. When you are sure all files have a blank security class, reboot your computer.

When you reboot your computer, enter the new master password to use as a part of the encryption key. Then use the .COPY command with the /c operand to assign security classes to files. When you assign the new copy of the file a security class, it is saved in an encrypted state with the new master password as part of the encryption key.

---

# CHAPTER 10: PRINT SPOOLER

---

Introduction . . . . .	10 - 2
Print File Names . . . . .	10 - 2
Print File Extensions . . . . .	10 - 4
Disposition . . . . .	10 - 4
Priority . . . . .	10 - 5
Print Class . . . . .	10 - 5
MOS HOLD LPTn Commands . . . . .	10 - 6
PRINT.CTL File . . . . .	10 - 6
Print Spooler Commands . . . . .	10 - 7
.PRINT . . . . .	10 - 8
.SPOOL . . . . .	10 - 11
Initiating the Print Spooler . . . . .	10 - 15
Automating the Print Spooler . . . . .	10 - 16
The Print Spooler Menus . . . . .	10 - 17
Print Processor Menu . . . . .	10 - 17
Spooler Menu . . . . .	10 - 19
Multiple Printer Environments . . . . .	10 - 22
Print Spooler Example Batch Files . . . . .	10 - 22
Batch Files for Multiple Users . . . . .	10 - 22
Batch Files for Multiple Printers . . . . .	10 - 23

# Print Spooler

## Introduction

Printing may be done directly using the COPY command. However, a Print Spooler is available in MOS to provide flexible printing operations. The spooler intercepts reports (or any output) sent to the printer and saves the reports in print (disk) files. These files are then printed from the spooler to let you share the printer with other users, and to improve throughput. Printer errors (reserved, out of paper, off-line, etc.) display in an error window on the requesting task's screen.

There are two programs used in the spooling operation; a spooler and a print processor. The spooler is invoked in all partitions where spooled output is desired. The print processor is invoked in a partition dedicated to the spooling operation, and controls printing of the spooled files.

The two programs used for spooling printer output require the use of a dedicated directory. When the spooler intercepts printer output from a partition, the resulting print files are sent to the dedicated directory. The print processor controls printing of the spooled files by continually checking the dedicated directory for spooled files that are ready to print. The dedicated directory may be on any logical disk, i.e. C:, D:, etc.

The Print Spooler also gives you more control of the reports or files you need to print. You may prioritize or classify reports depending on your printing requirements. You may rename, delete and save reports. A report may be flushed from the spooler, or may be restarted at a specific page number. Letterhead and other noncontinuous forms can easily be handled.

## Print File Names

When the spooler is invoked, it sends printer output to the dedicated directory and assigns the resulting print file a name. These print file names are comprised of the following:

SPLxxxx.dpc



Where:

- SP - designates a spooled print file.
- L - is the system name (0-9 or A-Z) of the computer that originated the file. Defaults to L if not specified with the SPOOL command.
- xx - is the task ID of the partition where the file originated
- nnn - is a sequential number the spooler assigns to each file
- d - is the disposition of the file to be printed
- p - is the priority assigned to the file
- c - is the print class of the file

All print file names begin with SP to designate that they are spooled print files. The next position is a number (0 to 9) or a letter (A-Z) designating the computer that originated the file. For multiuser systems spooling to a server each computer must have different spool file names. These system names are assigned with the SPOOL command. (Defaults to "L".) The task ID of the partition where the printer output originated appears in the next two position of the print file name. For example, 00 appears for partition 0, 01 appears for partition 1, 12 appears for partition 12, etc.

A sequential number is assigned to each file as it is created to be sure each file has a unique name. The files from each partition are numbered individually, beginning with the number 001. For example, if two spool files are created from partition 1, then a spool files is created from partition 2, and then another is created from partition 1, the file names would be:

SPL01001.dpc  
SPL01002.dpc  
SPL02001.dpc  
SPL01003.dpc

Any spool files that are in the dedicated directory when you reboot MOS or power down the computer are saved in the dedicated directory. However, a file in the process of being written may not be usable. When the .SPOOL command is invoked in a partition, new spool files are assigned the next sequential number after any existing files. If there are no existing spool files for a partition, then numbering begins with 001.

# Print Spooler

---

## Print File Extensions

The extension of each print file name is comprised of values that indicate the file's disposition, priority and print class (dpc). These three values determine the status of the print file, and how the print processor treats the file.

Each time you boot MOS and invoke the spooler, default values are assigned to the three file extension operands. The default values assigned are D - disposition, 2 - priority, and A - print class.

You may change these default values with the .SPOOL command, or with an option from the spooler menu that is available in each partition where the spooler is active. You may change these values after a spool file has been created with an option from the print processor menu. These three features are explained later in this chapter. The values you may enter for these three operands are:

### Disposition

There are five possible choices for a print file's disposition:

- D - Print the file and then delete it. (The default).
- H - Hold the file, do not print it.
- S - Save the file after printing. When printing is complete, the file disposition is changed to H.
- N - No spooling; the spooler is bypassed and the output is sent directly to the printer.
- I - Ignore output, spool file will not be generated.



## Priority

Priority is one digit in the range of 0 to 9, with 0 being the lowest priority and 9 the highest. The priority determines when a file will print in relation to any other files that are ready to print. For example, if several files are ready to print, a file with a priority 9 prints before a file with a priority 8. The default is 2.

## Print Class

Print class is one alpha character in the range A to Z that indicates any special printing requirements. The print class has no relation to security classes. Print classes A to T are user definable. Print classes U to Z are reserved for single sheet forms where a separate piece of paper must be placed in the printer at the end of each page. The default is A.

The print processor will print only the files that are assigned the class that is currently the active class. If you include the /P=\* operand with the .PRINT command, all classes may be active at the same time.

You may use the class for any special printing requirements. For example, you may group reports that require special forms into a separate class. Then place the special forms for that class in the printer and make that class the active class. Only the reports that require the special forms will print. When they are complete, remove the special forms from the printer and change the active class to another class.

# Print Spooler

---

## MOS HOLD LPTn Commands

The "MOS HOLD LPTx nnnn" command can be used to automatically reserve a printer for use by a specific user for a specified length of time. "x" specifies the printer, and "nnnn" specifies the time in seconds. The maximum is 9999 seconds (about 166 minutes or 2 & 3/4 hours).

This automatic reservation prevents other users from using the printer for the specified time after the reserved user has accessed it. This feature allows both direct printing and spooled printing on the system.

The "MOSADM HOLD task LPTx nnnn" command for the system administrator is similar in function except it can set the reservation time or release a printer for another task.

See Chapter 5 for complete information on using these MOS utility commands.

## PRINT.CTL File

The PRINT.CTL file contains initialization and termination strings for each print class. Before each spool file is printed the class's initialization string is sent to the printer. After each file is printed the class's termination string is sent to the printer.

The initialization strings are used to set up printer fonts or forms control. The termination messages usually contains a formfeed to separate print files and other characters to reset the printer.



The PRINT.CTL file has the following format:

Line 1 contains the initialization string for Class A spool files

Line 2 contains the termination string for Class A files.

Lines 3/4 contains the init/term strings for Class B spool files.

Lines 5/6 for Class C

Lines 7/8 for Class D

.

.

Lines 51/52 for Class Z

**IMPORTANT:** The PRINT.CTL file must exist and reside in the designated spool directory.

Control characters may be generated by preceding a character with a caret (^); ^D will generate a CTRL-D. ^^ is used for a single caret (^).

Consider the following sample PRINT.CTL file:

```
^D@font(COUR12)^M
```

```
^L
```

```
^L
```

Class A's initialization string is ^D (CTRL-D) followed by @font(COUR12) followed by ^M (Carriage return). The termination string is ^L (formfeed). Class B's initialization string is null; nothing will be sent to the printer before the spool file is printed. Class B will output a ^L formfeed after each file is printed. If /NOFF is included on the command line, for formfeed suppression, the formfeed may be ignored if the printer is already on a new page. In this example, all other initialization and termination strings are null.

## Print Spooler Commands

There are two commands in MOS that you will use to set up the spooling operation; .PRINT and .SPOOL. These two commands are explained in detail on the following pages.

# Print Spooler

## .PRINT

The .PRINT command lets you invoke the print processor and specify the drive and path of the directory to check for print files.

**Type:** Extrinsic.

**Form:** .PRINT [*path*] {/{FF | NOFF} /*class*[/*pt*/*lines*]] /*T=nn* /*P=x*

### **Operands:**

*path* defines the drive and directory to search for spool files and the printer control file PRINT.CTL. Defaults to C:\SPOOL.

*/FF* all formfeeds will be printed for the following classes. (Doesn't change anything that goes to the printer, i.e. formfeeds in file will be acknowledged.) This is the default condition.

*/NOFF* redundant formfeeds will be suppressed for the following classes. A formfeed when the printer is already at the top of a new page, or a formfeed following a formfeed is redundant

*/class* defines printable classes and optionally changes the default printer and lines per page. /\* will assign the parameters to all the classes. Printer 0 disables any printing for that class.

*/T=nn* defines the directory scan interval. The print program will check the directory for a file to print every "nn" seconds. Defaults to 15 seconds. Range is 1 to 99 seconds.

*/P=x* defines the active class(es). Defaults to class "A". A single class may be enabled with this switch or you may enter /P=\* to print all printable classes, i.e. all classed defined with the /class operand.



## Explanation:

The .PRINT command should be invoked in a partition that is dedicated to the printing of spooled reports. A print processing menu displays in that partition. Files are printed based on their status, which you may control from the menu.

The drive and directory entered for the .PRINT command must be the same as the drive and directory entered with the .SPOOL command. For example:

```
.PRINT C:\SPOOL\
```

This command invokes the print processor and instructs it to check the C:\SPOOL directory and print any files found based on their status. The .SPOOL command you enter in a user's partition must be set up to send the spooled output files to this directory.

If you have two printers, you may create more than one directory for spooling printer output. You must invoke the .PRINT command in a dedicated partition for each of these directories. You may then use the .SPOOL command and define the directory to which each partition will send printer output. You may also direct output to different printers from one directory by assigning a different class for each printer.

The following are some examples of PRINT commands:

```
PRINT /A
```

Prints Class A jobs on default printer (LPT1). Spool directory defaults to C:\SPOOL. Form length defaults to 66 lines per page. Directory scan time defaults to 15 seconds.

```
PRINT D:\SPOOLER /A1 /C3 /NOFF /B284 /P=*
```

## Print Spooler

---

Prints spool files from \SPOOLER on the D: drive. Prints Class A jobs on LPT1, Class B jobs on LPT2, and Class C jobs on LPT3. LPT2 has 84 lines per page and all redundant formfeeds will be suppressed to it. /P=\* means print all defines classes, in this case A, B, and C. Any other print classes will not be printed.

```
PRINT /* /B2 /C /U3 /V /P=*
```

Sends all print classes to LPT1 except Classes B and C which are sent to LPT2 and classes U and V which are sent to LPT3. Spooled files are found in the default directory C:\SPOOL.

```
PRINT /* /D2 /E3 /NOFF /F /P=A
```

Sends all print classes to LPT1 except Classes D, E and F. Class D is sent to LPT2. Classes E and F are sent to LPT3. Class F will suppress all redundant formfeeds in the print file. Initially only Class A will be printed, all other classes will be held until the operator specifically enables them.



## .SPOOL

The .SPOOL command lets you specify the drive and directory where the print files for a partition are to be sent. .SPOOL is also used to change the default print status and security class that is assigned to print files for that partition.

**Type:** Extrinsic.

### Forms:

.SPOOL {/[Nm] [path] [/Sc] [/Kxxxx] {[d[p[c]]] [/Tnnnn] [/LPTx]}

.SPOOL OFF

### Operands:

/Nm	assigns a character to include in the spool file names. Multiuser systems spooling to a server must have different spool file names. "m" represents the system name or number (0-9 or A-Z) of the computer that originated the file. Defaults to L.
path	write spool files to the given path. Defaults to C:\SPOOL. A complete path must be specified, including drive and directory.
/Sc	"c" reassigns the security class of the generated spool file. Used when security is active to declassify output so it can be printed by an unsecured print task. This allows the print task to have a low security class (to protect higher class files from unauthorized access) and still be able to print out files of a higher class when desired. Use an underscore (_) or a space to assign a blank security class.
/Kxxxx	changes the spooler popup hotkey to the specified new keycode. The value for "xxxx" for the desired hotkey can be obtained by first using the spooler menu to temporarily change the hotkey assignment.

These first four parameters are only specified once.

# Print Spooler

---

The following three parameters must be specified for EACH printer specified. (If any of these three parameters are specified, they become the defaults for the following specified printers, unless changed.)

- dpc* enter the disposition (D, S, H, N, I), priority (0-9), and class (A-Z) of the printer output.
- /Tnnnn* Defines a timeout value. Enter /T and the number of seconds that elapse without any output before a print file is automatically closed and printed. Timeout defaults to 18 seconds. Maximum is 3600 seconds (60 minutes). Minimum is 1 second.
- /LPTx* Parameters before LPTx will be used for that local printer. Unspecified printers will be merged with the current parameters.

## Explanation:

When you invoke the .SPOOL command in a partition, a TSR (terminate and stay resident) program is loaded that intercepts the printer output for that partition and directs it to the dedicated directory as a spooled print file. The drive and directory entered for the .SPOOL command must be the same as the drive and directory entered with the .PRINT command.

To remove the spooler TSR from memory, enter the following command:

SPOOL OFF

The spool program is then unloaded and all subsequent print output from that task is not spooled, but sent directly to the local printer device. This is not a good idea if other tasks on the system could also generate print requests.



The following are some examples of SPOOL commands:

```
SPOOL D:\SPOOLER /SA D5A
```

Spooled files go to the D:\SPOOLER directory. Spooled files are assigned a disposition of "D" and a security class of "A". Priority is changed to 5.

```
SPOOL D2B /T100 /LPT2
```

LPT2 output is spooled to Class B with a 100 second timeout. Output to unspecified printers (LPT1 and LPT3) is merged and spooled with the same class and timeout value.

```
SPOOL /N0 F:\SPOOLER D4A /LPT1 D6B /LPT2 D2C /T100 /LPT3
```

Spools all output to network drive F:\SPOOLER with assigned file names SP0ttnnn.dpc.. Output to LPT1 is given class A, priority 4. Output to LPT2 is given class B, priority 6. Output to LPT3 is given class C, priority 2 and has a 100 second timeout. Files can be printing on all three printers at the same time. One printer might be a dot matrix printer, one a laser printer, etc.

```
SPOOL /N1 C:\SPOOL1 /LPT1 N /LPT2 C:\SPOOL3 D6V /LPT3
```

Spools LPT1 output to C:\SPOOL1\SP1xxnnn.D2A. Output to LPT2 is not spooled, but printed directly to LPT1). Output to LPT3 is spooled to C:\SPOOL3\SP1xxmn.D6V.

After invoking the .SPOOL command to establish the directory, the .SPOOL command may be used for other functions. MOS restricts any subsequent uses of the .SPOOL command in a partition to changing the default status of spooled disk files, or changing the time that elapses without any output to a print file before the print file is automatically closed.

## Print Spooler

The default print status is determined by the extension of the print file name. When .SPOOL is first invoked, the extension of the print file defaults to D2A, where D is the disposition, 2 is the priority, and A is the print class. An explanation of print file names is given in a separate section of this chapter, and describes the possible print status selections.

You may change any or all of the print status defaults. For example, to assign the disposition H (hold) to all print files for the current partition, you would enter:

```
.SPOOL h
```

To also change the priority for this partition to 6, and the class to b, you would enter:

```
.SPOOL h6b
```

To change the disposition and class, but not the priority, use an underscore in the position for the priority. You would enter:

```
.SPOOL h_b
```

You may change how quickly a spool file closes when there is no output to the file with the /T operand. The /T operand may be included with the command that invokes the spooler directory. For example:

```
.SPOOL C:\SPOOL\ /T10
```

Or, you may enter the .SPOOL command at the system prompt with only the /T operand. For example:

```
.SPOOL /T10
```

Both of these commands instruct the spooler to close a file if 10 seconds elapse without any output to the file. This command changes the time only for the partition where the command is invoked.



You may have programs that print one line, process information, print another line, etc. If the processing time is longer than the default of 18 seconds, the spool file will automatically close and the report will not be complete. You may need to increase the time span in a partition where 18 seconds is not enough time for a report to be completed.

If you are sending information to the printer where a program is not involved (such as with the PrtSc key), you may want the print files to close more quickly to speed up processing. You may want to set the number of seconds that elapse to a much smaller number.

If you assign a different hot key to bring up the spooler menu in a partition, you may enter the .SPOOL command without any operands to reset the hot key in that partition back to its original value of CTRL HOME.

**NOTE:** The .PRINT and .SPOOL commands should never be invoked in the same partition. .SPOOL intercepts printer output and writes it to the print files while .PRINT reads the print files and creates printer output. Placing both commands in the same partition will create an endless loop.

## Initiating the Print Spooler

Before initiating the Print Spooler, create a directory that will be dedicated to the spooling operation. You should give this directory an identifying name, such as SPOOLER. For example:

```
.MD SPOOLER
```

Then use an editor to create the file PRINT.CTL, and copy it into this directory. See the section on the PRINT.CTL file in this chapter.

Create a dedicated partition for the print processor with the .ADDTASK command. This partition can be set up to use the minimum memory size of 32K. For example:

```
.ADDTASK 32
```

# Print Spooler

---

Now, install the print processor in the dedicated partition you created. Enter the .PRINT command in this task followed by the drive and path where the directory to check is located. If the directory is several levels from the root, you must include all levels in the directory name. For example:

```
.PRINT C:\WORDPROC\MEMOS\SPOOLER
```

The .SPOOL command is invoked in each partition that you want controlled by the print processor. Enter the .SPOOL command followed by the same drive and path entered for the .PRINT command. For example:

```
.SPOOL C:\WORDPROC\MEMOS\SPOOLER
```

Spooled output files for the partition are now sent to the directory SPOOLER. The print processor checks the SPOOLER directory and prints the files based on their print status.

## Automating the Print Spooler

If you plan to use the Print Spooler, you may want to automate the .PRINT and .SPOOL commands by placing them in the appropriate batch files. You may place .ADDTASK commands in the AUTOEXEC.BAT file to create user partitions and the dedicated partition for the print processor each time you boot MOS.

The startup batch file for the print processor partition may include the .PRINT command to automatically invoke the print processor. The .SPOOL command may be placed in the startup batch file for each user partition that is to be controlled by the print processor.

Make sure each .SPOOL command is invoked with the same drive and path of the directory for spooling as entered with the .PRINT command. For more detail, see the example batch files at the end of this chapter.



## The Print Spooler Menus

There are two menus available with the Print Spooler to let you control the printing operation. When you invoke the .PRINT command, the print processor menu appears in the partition where the .PRINT command is invoked. This should be a dedicated partition from which the printing operation is controlled.

In each partition where the .SPOOL command is invoked, a hot key is available to pop-up a spooler menu that lets you control the print files for that partition. The following explains each of these menus and the options that are available.

### Print Processor Menu

The print processor menu is available in the partition where you invoke the .PRINT command. You have full control of the print files that appear on the menu. The menu appears similar to the following:

PC-MOS Print Processor v#.##  
Copyright 1989 The Software Link, Incorporated

Active Class: B Now Printing -nothing- Page 0 Size 0 0% Done

Commands-----	Files-----
C - Change active class	SP00101.D2A 32198
F - Flush current listing	SP00401.D2C 3640
L - Change LPT/Class assignment	SP00402.D2C 1425
S - Stop/Start printer	SP00201.H2A 24281
R - Restart with new page nbr	SP00102.D2A 15573
M - Modify disp/pri/class	
D - Delete file	
X - Exit to MOS	

Command?

The print processor continually looks for files to print. A file is printable if it has a disposition of D or S, its class matches the active class displayed on the menu, and its length is not zero. The active class and file currently printing are displayed at the top of the menu.

# Print Spooler

---

Each command on the menu is invoked by a single character code:

- C - Change active class. Lets you change the class that is currently active. Only files with a class that matches the active class will print. (A-Z and \*. See PRINT command.)
- F - Flush current listing. The file currently printing is flushed then deleted or renamed, depending on its disposition.
- L - Change LPT/Class assignment. When you invoke .PRINT, the device default is LPT1 and the active class A. This option lets you direct a class of print files to any attached LPT device. For example, you can direct all class C files to LPT2 or all class U files to LPT3. You may even direct a class of print files to printers that are not attached. However, when you activate that class, printing will be sent to the default device, LPT1.
- S - Stop/start printer - This option lets you stop or halt printing of the current file, and then restart printing at the position in the file where it was stopped.
- R - Restart with new page nbr - You may stop printing from the current file, and then restart printing beginning with a page number you assign. This is very helpful if you want to print only part of a report, or if a paper jam or printer problem occurs.
- M - Modify disp/pri/class - This option lets you assign a different extension to a print file to change the manner in which it is processed.
- D - Delete file - You may use this option to delete a print file from the directory where the print processor is running.
- X - Exit to MOS - This option terminates the .PRINT command and returns control of the partition to MOS. Spooled print files will still be sent to the directory, but will no longer be printed by the print processor.



While printer output is being spooled, the print file name appears on the menu with a file length of zero. The print file will not be closed by the spooler until all output is complete. If output is sent by an application program, output is complete when the program ends. Or, you may pop-up a menu in the spooler partition and select the option to manually close a file. When the print file is closed, the actual file length appears on the menu.

The print processor cannot print an open file; it must wait until the file is closed. The spooler automatically closes a file if there is a span of time during which there is no output to the print file.

For example, if you use the print screen key, a program is not actually running that will terminate the print file. You may terminate the file manually with an option from the pop-up spooler menu in the originating partition, or wait for the spooler to close the print file for you. The span of time to close a file defaults to 18 seconds, but can be changed with the .SPOOL command.

#### Spooler Menu

You may use the spooler hot key to bring up the spooler menu in any partition where the spooler is active. The hot key is set to CTRL HOME each time you invoke the .SPOOL command. Press the hot key and the spooler menu pops-up over any current application. You may select any option from the menu and then return to your application.

Press the CTRL and HOME keys at the same time and the spooler menu appears on your screen similar to the following:

**SUPER SPOOLER**

---

C:\SPOOL\SPL00???	-	LPT1
<hr/>		
[1] Current Spool Parameters	-	D2A
[2] Spool file status	-	CLOSED
[3] Spool file close time	-	18
[4] Spooler popup Hotkey	-	1143
[5] Keyboard Polling mode	-	NODIS
<hr/>		
Enter 1-5	UP/DN for printer	<RETURN> exits
<hr/>		

# Print Spooler

---

Type the number of the option you want to select. The options available are:

1. Current Spool Parameters. The default disposition, priority, and class for this partition appears for option 1.

To change the disposition, type a 1 and the message "Please enter new Disposition (D,S,H,I,N)" appears. Type the single character disposition to assign for this partition and it appears on the menu. (You may press the space bar to skip the disposition.)

The message "Please enter new Priority (0 - 9)" then appears. Type the number of the priority to assign for this partition and it appears on the menu. (You may press the space bar to skip the priority.)

The message "Please enter new Class (A - Z)" then appears. Type the single character class to assign for this partition and it appears on the menu. (You may press the space bar to skip the Class.)

You may then select another option or press ENTER to return to the application.

2. Spool File Status. The status of the current file appears for option 2, either OPEN or CLOSED. If there is no file open, option 2 has no function.

If a file is open, you may type a 2 to close the file so it will print more quickly. For example, if you are at the MOS level and use the PrtSc key, there is no program running to close the print file that is created. MOS will automatically close the print file if there is no output after the specified file close time (default is 18 seconds). You may close the file more quickly by bringing up the spooler menu and typing a 2.



3. Spool File Close Time. This option sets the number of seconds that elapse without any output before a print file is automatically closed and printed. To change, enter a 3 and the message "Enter new spooler close time" appears. Then type the new value to assign, in seconds, and press ENTER. (Defaults to 18 seconds, maximum is 3600 seconds - 60 minutes.)
4. Spooler Popup HotKey. If the use of CTRL HOME is in conflict with an application program, or is inconvenient, you may assign a different hot key to bring up the spooler menu. The new assignment must be a two-key combination using ALT, CTRL or SHIFT, i.e. ALT B, CTRL END, etc. Type a 4 and the message "Press the key you want for the new HotKey" appears. Type the key to use for the new hot key. A second message appears asking you to press the new hot key again to verify your choice. Press the same keys again and the message "Hotkey verified. Press any key to continue" appears. (If you change your mind, press any other key and the program will not change the hot key.)
5. Keyboard Polling Mode. The default keyboard mode is NODIS, for no disabling of programs that repeatedly test the keyboard. You may type a 5 to toggle the keyboard mode between NODIS and DIS. The mode displayed on the menu when you press ENTER to exit is the mode that is set for this partition.

# Print Spooler

## Multiple Printer Environments

There may be times when two or three printers are attached to your computer. You may direct printer output to any attached printer by associating print classes (A to Z) with a specific printer (LPT1, LPT2, or LPT3).

The association of print classes to printers is made from the print processing menu with the L - Change LPT/Class assignment option. These assignments may be anything you want to use. For example, you may assign classes A and B to the printer identified as LPT1. You may then assign classes C and D to the printer identified as LPT2.

By assigning classes to spooled print files, a user may direct output to any available printer. A user may invoke the .SPOOL command to change the print class, or may change the print class with option 1 from the spooler menu.

## Print Spooler Example Batch Files

The following are examples of how you could set up batch files for invoking the print spooler. These are only example. Your batch files should include all commands necessary for your particular needs.

### Batch Files for Multiple Users

This example is for one printer and three users. There are four partitions, one for the print processor, and three for the users. Note that the STARTUP.BAT file is nested in the AUTOEXEC.BAT file. This will cause MOS to first invoke the spooler for partition 0, and then begin executing the .ADDTASK commands.



## AUTOEXEC.BAT File

```
.BATECHO OFF  
STARTUP  
.ADDTASK 256,1,,startup,pcterm,1,19200  
.ADDTASK 256,2,,startup,pcterm,2,19200  
.ADDTASK 32,3,,print1  
.CLS  
.TEXT
```

"The spooler is now installed"

```
.ENDTEXT
```

## STARTUP.BAT File

```
.PATH=C:\;C:\PCMOS  
.SPOOL C:\SPOOL
```

## PRINT1.BAT File

```
.PATH=C:\;C:\PCMOS  
.PRINT C:\SPOOL
```

## Batch Files for Multiple Printers

This example is for two active printers and four users. Two users send all output to a letter quality printer (LPT1), and the other two users send all output to a dot matrix printer (LPT2). Note that the SPOOL1.BAT file is nested in the AUTOEXEC.BAT file. This will cause MOS to first invoke the spooler for partition 0, and then begin executing the .ADDTASK commands.

When the print processor is first invoked, the L option is selected from the print processor menu to assign LPT1 a print class of Q. Only print files with a class of Q will go to LPT1. The L option must also be selected to set LPT2 a class of D, so that only files with a class of D go to LPT2. However, only one printer will be active at a time.

Note that the .SPOOL command is included in the startup batch files to assign the print class of files for the user partitions.

### **AUTOEXEC.BAT File**

```
.BATECHO OFF  
SPOOL1  
.ADDTASK 256,1,,spool1,pcterm,1,19200  
.ADDTASK 256,2,,spool2,pcterm,2,19200  
.ADDTASK 256,3,,spool2.pcterm,3,19200  
.ADDTASK 32,4,,printer  
.CLS  
.TEXT  
  
"The spooler is now installed"  
  
.ENDTEXT
```

### **SPOOL1.BAT File**

```
.PATH=C:\;C:\PCMOS  
.SPOOL C:\SPOOL d2q
```

### **SPOOL2.BAT File**

```
.PATH=C:\;C:\PCMOS  
.SPOOL C:\SPOOL d2d
```

### **PRINTER.BAT File**

```
.PATH=C:\;C:\PCMOS  
.PRINT C:\SPOOL
```

---

## APPENDIX A: WARNING AND ERROR MESSAGES

---

<b>Introduction</b>	<b>A - 2</b>
<b>Self-Explanatory Messages</b>	<b>A - 2</b>
<b>General Error Messages</b>	<b>A - 2</b>
<b>Critical Error Messages</b>	<b>A - 3</b>

# Warning and Error Messages

## Introduction

Many of the warning and error messages in MOS are self-explanatory, and indicate what action you need to take. For example, the message, "Invalid operand" tells you that MOS does not recognize an operand you entered with a command. If you made a typing error, you may simply enter the command again. If you don't know what caused the error, use the .HELP command to display the valid operands for the command.

Normally, an application will be running when an error occurs. Depending on how the application traps and displays errors, you may never encounter an error at the MOS system level.

## Self-Explanatory Messages

The messages that are not self-explanatory are presented in this appendix in two groups - general error messages and critical error messages. Each message is explained, along with any action necessary to correct the error.

## General Errors

During processing, MOS will display general error messages if an error condition is encountered that is not critical. Most general errors will cause processing to halt, and MOS will display a self-explanatory message. Self-explanatory messages, such as "Cannot find file", "Invalid directory name", etc., are not explained here. The following are general error messages that may not be self-explanatory, along with an explanation of why the error occurred.

### File lock conflict

A file lock conflict error typically occurs if you use an application that is not designed for a multi-user environment.



## Record lock conflict

A record lock conflict error typically occurs if you use an application that is not designed for a multi-user environment.

## Wrong disk

The wrong disk error occurs if you are writing output to a file on disk and you change the disk before the write operation has finished.

## Critical Errors

Critical Errors are the messages that MOS displays when a situation occurs that prevents further processing. The Critical Error messages are displayed to give you an opportunity to correct the problem so that processing can continue. When you encounter a Critical Error, you must use your best judgement to deal with the situation.

The MOS Critical Error messages are presented in this appendix, with an explanation of the situation in which they may occur and the action that may be required to correct the situation.

Critical Errors are comprised of three variables and take the following form:

ERROR variable1 while variable2 variable3

Variable1 will give a description of the kind of error that occurred, variable2 identifies the type of processing, and variable3 indicates what was being accessed. For example, a Critical Error may appear as follows:

ERROR Not ready while writing A:

## Warning and Error Messages

The possibilities for each of the variables that may appear in a critical error are as follows:

Variable1	Variable2	Variable3
Write protect	reading	device name
Unit not known	writing	drive name
Not ready		
Data		
Request header		
Search		
Unknown disk		
Cannot find sector		
Printer needs paper		
Write		
Read		
General Failure		

### Write Protect

A write protect error can occur when MOS is attempting to write to a protected disk.

Action: Check the write-protect notch on the diskette.

### Invalid Command

An invalid command message may appear when an application issues a command MOS does not recognize.

Action: Contact your dealer or the developer of the application program you were using when the error occurred.

### Unit Not Known

A unit not known error typically occurs if MOS is looking for a device or drive that is not installed.

Action: Check the application that you are running to see if a device driver is required that has not been installed.



## Not Ready

A not ready error may occur if you attempt to access a drive and the drive door is open, or if you attempt to send output to a printer and the printer is not turned on.

- Action: If accessing a drive, check to be sure the disk is in the drive and the drive door is closed. If attempting to send output to a printer, check to be sure the printer is turned on.

## Data

A data error usually occurs when MOS cannot read or write data on a disk.

- Action: The disk may be damaged or not suitably formatted for your computer. If attempting to write to a disk, try formatting the disk first. If attempting to read from a disk, try again. If the data error continues, the disk is probably damaged beyond use.

## Request Header

A request header error occurs when MOS cannot find the necessary information for a device driver.

- Action: Contact your dealer or the developer of the application program you were using when the error occurred.

## Search

A Search error typically occurs when your disk is damaged.

- Action: Check your disk and try again. If the search error continues, the disk is probably damaged beyond use.

### **Unknown Disk**

An unknown disk error occurs when MOS is attempting to access a disk that is not formatted for your computer.

Action: Your disk may need formatting before use with MOS.

### **Cannot Find Sector**

A cannot find sector error means that your disk may be damaged.

Action: Check to be sure you have the correct disk and try again. If the error continues, the disk is probably damaged beyond use.

### **Write**

A Write error indicates that your disk may be damaged and MOS cannot write data to it.

Action: Check to be sure you have the correct disk and try again. If the error continues, the disk is probably damaged beyond use.

### **Read**

A read error indicates that your disk may be damaged or not in a format that MOS can read.

Action: Check to be sure you have the correct disk and try again.

### **General Failure**

A general failure indicates that an error occurred, but the error doesn't fit any of the defined MOS errors as to how or why it happened.

Action: Make note of what was processing when the error occurred and contact your dealer for assistance.

---

## APPENDIX B: NETBIOS EMULATION

---

Introduction . . . . .	B - 2
Setting Up NETBIOS Communication . . . . .	B - 2
Signing On to the Network . . . . .	B - 3
Network Emulation Error Messages . . . . .	B - 3

# NETBIOS Emulation

## Introduction

Some application software programs support the Network Basic Input/Output System (NETBIOS) protocol for interfacing to network hardware, such as the IBM® Network. The network offers peer to peer communication between individual computers. Peer to peer communication allows all computers connected in the network to communicate with each other. This is typically accomplished by placing a "network interface card" in each computer that will tie into the network, and connecting the individual computers with cables. Each computer in the network is identified as a user.

MOS supports applications written for NETBIOS by offering a software emulation of the network interface cards. Each MOS partition can emulate an interface card for running network applications on a single computer. The emulation treats each MOS task (partition) as a separate computer. A user may communicate with other partitions using the emulation software in the same way a user may communicate with other computers under NETBIOS.

## Setting Up NETBIOS Communication

The MOS network emulation must first be defined with a DEVICE= command statement in your CONFIG.SYS file. This device driver makes the emulated interface cards available to the partitions. The command statement for this device driver is:

```
DEVICE=$NETBIOS.SYS tasks=nn
```

The tasks=nn operand sets the number of tasks (partitions) that will emulate the network interface card. This number may range from 2 to 25. Each interface card you define will use approximately 1600 bytes of memory allocated from the System Memory Pool (SMP).

For example, if five workstations and the main console will use the network application, you need to define six tasks for emulating six interface cards. This will require approximately 9600 bytes of memory from the SMP. Some applications require that one task (partition) act as a disk file server. If that is the case, you need to define one additional task.



Be sure you have enough memory reserved with the **SMPSIZE=** command statement for the number of partitions that will use the interface card emulation, and all other device drivers. The **SMPSIZE=** command statement is also defined in your CONFIG.SYS file.

## Signing On to the Network

Before running an application that requires the network, a user in a partition must first sign on to the network. If a dedicated partition is required to act as a disk file server, a user must also be signed on in the dedicated task. MOS provides the **.NETNAME** command as the method of signing on to the network.

Each user must sign on from within a partition and enter a unique user name of up to 15 characters without any spaces. The user name identifies each task (partition) and allows the network to control and track each user's access to other user's resources. For example:

```
.NETNAME GARY
```

If the sign on is successful, or if you enter the **.NETNAME** command without an operand to display the current user sign on, MOS displays the following message:

User name is GARY

Note that only one user can sign on to the network from within a partition, and only the number of users defined with the \$NETBIOS.SYS device driver can access the network.

## Network Emulation Error Messages

There are several messages that may appear when you attempt to sign on to the network emulation. The following explains why these messages occur.

**Not enough MOS memory for NETNAME**

This message will appear if there is not sufficient memory to run the .NETNAME command.

**NETBIOS driver must be loaded before NETNAME is run.**

This message appears if the \$NETBIOS.SYS device driver has not been defined. You should define this driver in your CONFIG.SYS file so that it is always available when you boot MOS. You may use the .ADDDEV command to define the driver without rebooting.

**GARY is being used.**

This message appears if you enter a username that has already been used to sign on in another partition.

**Too many network commands are active**

The interface card is busy and cannot process the sign on command. You may want to wait a few seconds and try again.

---

## APPENDIX C: CUSTOMIZING THE HELP FACILITY

---

Introduction .....	C - 2
File Descriptions .....	C - 2
Editing the HELP.SRC File .....	C - 3
Rules for Editing HELP.SRC .....	C - 4
Building the .HELP Data Files .....	C - 4

# Customizing the HELP Facility

## Introduction

The .HELP command displays a menu listing all the commands and utilities that are available in MOS. From the menu, you may display on-line screens of help text for individual commands. A detailed explanation of how to use the .HELP command is given in the General Commands chapter of this manual.

There may be times when you want to change the information in the help displays to customize it for your own use. You may add your own utilities or commands to the menu list and create the associated help text screens you want to display. If necessary, you may also modify the existing help text for translation to other languages. The files necessary to customize the help text and menu are provided with MOS.

## File Descriptions

The files provided with MOS that allow you to customize the .HELP facility are:

- |          |  |
|----------|--|
| HELP.EXE | This program contains the machine language instructions to display the menu and help text screens. There are two data files that are necessary for this command, both of which must reside in the same directory as this file. |
| HELP.TXT | This is the first data file HELP.EXE uses. This data file is built from the source file HELP.SRC. DO NOT EDIT this file, or you may find the .HELP command no longer functions properly.                                       |
| HELP.NDX | This is the second data file HELP.EXE uses, and is also built from the source file HELP.SRC. This file contains the index for building the menu and displaying the associated text blocks as help text screens.                |
| HELP.SRC | This file contains the raw text used to build the HELP.TXT and HELP.NDX data files. This is the current data used with the .HELP command that you may customize for your own use.  |

This file does not have to be in the same directory as the HELP.EXE file. The rules for customizing this file are given following these file descriptions.

- HELPGEN.EXE** This program is a conversion utility that builds the data files, HELP.TXT and HELP.NDX, from the raw text in the HELP.SRC file. Any text that is in the HELP.SRC file when you run this utility will overwrite the current contents of these two data files. Note that the HELP.SRC file and the HELPGEN file must reside in the same directory when you run this conversion utility.

### Editing the HELP.SRC File

You may edit the contents of the HELP.SRC file with any editor or word processor capable of operating on ASCII files. When you edit this file, you will notice that it contains blocks of text similar to the following:

```
;HELP  
The help command lets you display ...  
;IMPORT  
The .IMPORT command restores files ...
```

Each block begins with a "keyword" that is associated with the block. Keywords are identified by a preceding semicolon, which must be positioned in column one of the file. The keyword that follows the semicolon cannot contain more than 14 characters.

In the previous example, .HELP and .IMPORT are keywords because they are preceded by a semicolon. The HELPGEN utility places the keywords in the HELP.NDX file for displaying on the menu list. Any text associated with the keyword is placed in the HELP.TXT file.

The order in which the keywords and their associated text blocks are entered in the HELP.TXT file determines the order in which the keywords appear on the menu. If you want to add help text for your own group of utility commands, you may preserve the current alphabetical order by placing each block in the appropriate position of the existing HELP.SRC file. Or, you may want to place all of your own entries at the end of the HELP.SRC file as a group.

### Rules for Editing HELP.SRC

When entering your own text in the HELP.SRC file, be sure to abide by the following rules:

- a text line may have a maximum of 80 characters.
- a block of text may contain a maximum of 4096 characters.
- a block of text may contain a maximum of 200 lines.
- the help text screen displays 18 lines of the text block at a time.
- the HELP.SRC file will facilitate a maximum of 90 keywords, including the existing MOS entries.
- be sure the only semicolons that appear in column one of the file are associated with keyword entries.

### Building the .HELP Data Files

After modifying the HELP.SRC file, run the HELPGEN conversion utility by entering HELPGEN at the command line. The HELP.SRC file must reside in the same directory as the HELPGEN.EXE file when the conversion is run.

The HELPGEN utility will read HELP.SRC and create the data files HELP.TXT and HELP.NDX in the same directory. If the data files already exist in the same directory as HELP.SRC and HELPGEN, the conversion utility will overwrite their current contents. If you create new data files, you may then copy the new data files over the existing data files in the directory where the HELP.EXE program file resides.

---

**Attribute** - An attribute in MOS is a characteristic inherent to a file. MOS files have four attributes that display with file names in a directory; R - read-only, \* - archive, S - secured, and H - hidden.

**Assembled Program** - A program that has been converted to machine readable language.

**Automated Batch File** - A special file that contains a group of commands that specify the processing environment. An example of an automated batch file is the AUTOEXEC.BAT.

**ASCII** - American Standard Code for Information Interchange. A method of translating computer machine language to a user-identifiable character set.

**Backup** - To make copies of files for safekeeping. MOS provides the .EXPORT command to backup files.

**Base Memory** - The memory below 1 megabyte in RAM (Random Access Memory). The first 640 kilobytes of this memory is typically reserved for processing applications.

**Batch File** - A file that contains a command or group of commands that are to be executed one after the other. MOS recognizes a batch file by its .BAT extension. You may use batch files to create your own utilities or to automate a string of commands.

**Bit** - A binary digit, with possible values of 0 or 1. This unit is recognizable by computers since the two possible values can be represented by the presence (1) or absence (0) of voltage.

## Glossary

**Boot** - To supply power to your computer and cause the computer to load the operating system. To reboot simply means to cause your computer to reset memory and reload the operating system.

**Boot Sector** - A special disk sector reserved for information the computer needs to load the operating system.

**Byte** - A group of eight bits used to represent one character in the PC character set, such as the letter "A".

**Class** - MOS identifies two types of class; a security class and a print class. A security class is assigned to files, directories and partitions, and user access is then defined on a class-by-class basis. A print class is used with the MOS Print Spooler to control printing of spooled print files.

**Cluster** - The smallest unit of disk space allocated for a file. The size of the cluster is computed when the disk is formatted.

**Command** - An instruction to MOS to perform a specific task.

**Command Line** - The line on which the MOS prompt appears and where you may enter commands.

**Command Processor** - The interface between the user and the operating system kernel, through which commands are interpreted and executed. Also called the shell. The MOS shell is \$\$SHELL.SYS. The MOS kernel is \$\$MOS.SYS.

**Configuration** - The grouping of hardware and software that makes up your computer system. This grouping includes the main console, any printers, workstations, the operating system, plus any other software applications or hardware.

**Current Directory** - The directory in which you are presently processing. If no other directory is specified, MOS defaults to the current directory.



**Data Compression** - A means of packing data to lessen the space it occupies for storage.

**Default Drive** - The present drive on which you are working. If you make a command line entry and do not specify otherwise, MOS defaults to the current drive.

**Delimiter** - A reserved character, such as a forward slash (/), a comma (,) or a space that signifies the end of one command or operand and the beginning of another.

**Device Driver** - A piece of software that contains the instructions for running a particular device in your computer. When invoked it activates and controls communication with the device.

**Directory** - A group of filenames, with associated pointers and statistical information.

**Disk** - A medium on which you may read, write, or store data. The medium could a floppy disk, a hard disk, or a removable disk.

**Diskette** - The removable medium in a drive on which data may be read, written, or stored.

**Drive** - A device containing disk media on which data may be read, written, or stored. The current drive is the drive which contains the files on which you are presently working.

**Encryption** - A means of scrambling the contents of a file so it cannot be read or written to by unauthorized personnel. An encryption key is needed to decipher the encrypted files.

**Environment** - A reserved area of memory where frequently accessed strings of data are stored.

## Glossary

**Error Level** - A code that is set by MOS when a command is executed. This code indicates whether or not the execution was successful. MOS returns one of three codes when a command is executed -- 0 for a completely successful execution; 1 for an execution which may have contained errors; and 2 for an execution that was unsuccessful.

**Executable File** - A file which contains machine recognizable language that can be executed without being assembled or interpreted.

**Extension** - Preceded by a period, the character string that follows a file name. It may be up to three characters long and can serve as an internal identifier of the file type. MOS recognizes the .BAT, .COM, .EXE, and .SYS extensions, and treats files with these extensions in a specific manner.

**Extended Memory** - That memory which is above one megabyte on your computer.

**Extrinsic** - A type of MOS command that resides on disk in an individual program file.

**Filter** - A method by which input is redefined and then output.

**Hard disk** - A non-removable disk with hard platters.

**Intrinsic** - A type of MOS command that is loaded into memory and is available for execution at any time.

**Kernel** - The core of the operating system that does all the work. The MOS kernel is \$\$MOS.SYS. When a user makes a request of the operating system through the shell and command processor, it is the kernel that actually does the work.

**Keystroke** - The act of pressing a key on the keyboard.



**Logical Disk** - A disk device defined in software that exists regardless of the number of physical disks present on the system. For example, one physical hard disk may be divided into four logical disks - C:, D:, E: and F:.

**Main Console** - The computer that forms the base of your system, including its video screen and keyboard.

**Nesting** - The act of calling a second batch file to execute from within one batch file and then returning to the original batch file.

**Operand** - A single character or character string that further defines how a command is to be executed.

**Parent** - The directory that immediately precedes the name of a subdirectory in a path.

**Partition** - A defined area of memory that is associated with a specific task.

**Patch File** - A file that contains the changes that are to be made to an existing program file with MOS's .DEBUG utility.

**Path** - The directories that precede a specified directory or filename.

**Pipe** - A character device to which output may be redirected and from which it may be retrieved.

**Port** - The physical device which serves as a channel for peripheral devices to communicate with the Central Processing Unit.

**Prompt** - A system-supplied or user-defined character string or group of symbols displayed by the operating system, signifying that it is ready to accept a command.

**Random Access Memory - (RAM)** Memory reserved for storage and retrieval of data during processing. The application's workspace.

**Replaceable Operands** - False operands in a batch file that are replaced by real operands entered from the command line when the batch file is invoked.

**Root** - The base level of file storage on a disk. The root directory is the first directory, under which all subsequent directories are made.

**Shell** - The shell is the interface between the user and the operating system kernel. The MOS shell is \$\$SHELL.SYS. The MOS kernel is \$\$MOS.SYS.

**SMP** - System Memory Pool. A small amount of memory assigned at system setup to supervise all tasks and track all activity on the system.

**String** - A group of characters which may represent specific values. MOS may refer to strings of data placed into the environment for easy retrieval.

**Time Slicing** - The method by which MOS divides processing time and allocates it to each partition in a multi-tasking/multi-user environment.

**Utility** - A system-related or general-purpose program that performs a specific function or functions.

---

# INDEX

---

**\$\$MASTER.SYS**  
    changing 9-22  
    initiating 9-20 to 9-21  
    purpose 1-9, 9-20  
    precautions for use 9-20, 9-21

**\$\$MOS.SYS**  
    making a diskette bootable 4-56, 4-67  
    purpose 1-8

**\$\$SHELL.SYS**  
    making a disk bootable 4-56, 4-67  
    purpose 1-8, 4-5

**\$\$USER.SYS**  
    creating 9-3 to 9-6  
    initiating 9-3 to 9-6  
    purpose 1-9, 4-3  
    securing from users 9-3, 9-6

**\$COMPAT\$**  
    explanation 6-36  
    purpose 6-36

**\$EMS.SYS**  
    default value 2-38  
    emulation of EMS 2-38  
    explanation 2-38  
    operands 2-38  
    purpose 1-8

**\$MOUSE.SYS**  
    explanation 2-40  
    purpose 2-40

**\$NETBIOS.SYS**  
    errors during network sign on B-3 to B-4  
    explanation B-2 to B-4  
    operands B-2  
    purpose 1-8  
    setting up network emulation B-2 to B-4  
    signing on to the network B-3

# Index

---

## \$PIPE.SYS

- default value 2-42
- explanation 2-43
- operands 2-42
- purpose 1-8

## \$RAMDISK.SYS

- default value 2-44
- explanation 2-45
- operands 2-44
- purpose 1-9

## \$SERIAL.SYS

- .ADDTASK serial port assignments 5-13
- default value 2-46, 2-47
- explanation 2-47 to 2-50
- initializing a port 5-30
- interrupt vectors, reserving 5-23 to 5-27
- modem connections 2-49
- operands 2-46, 2-47
- purpose 1-9

## 8087 command statement

- default value 2-4
- explanation 2-4
- form 2-4
- operands 2-4
- type 2-4, 4-5

# A

## .ABORT

- command form 6-4
- explanation 6-4
- nested batch file usage 6-4
- type 4-5, 6-4

access keys, partitions 5-19

access levels, security 9-2 to 9-22

- directory level access 9-15, 9-16
- file level access 9-17, 9-18
- partition level access 9-12 to 9-14



- .ADDDEV
  - explanation 4-11
  - form 4-11
  - operands 4-11
  - type 4-4, 4-11
- adding tasks 5-11 to 5-17
- .ADDTASK
  - batch file usage 6-31
  - command form 5-11
  - explanation 5-14 to 5-17
  - modem connection at remote location 5-64
  - operands 5-11 to 5-13
  - port assignments 5-13, 5-16
  - security class assignment 5-12, 5-15
  - startup batch file 5-13, 5-14
    - security example 9-13
  - terminal baud rates 5-13
  - terminal ID 5-12, 5-16
  - type 4-4, 5-11
- .ALIAS
  - explanation 4-13, 4-14
  - form 4-13
  - operands 4-13
  - type 4-4, 4-13
- archive attribute 3-4, 4-53
- asterisk
  - attributes of files 3-4, 4-53, G-1
  - wildcard character 3-5, 3-6
- .AUTOCD
  - command form 6-5
  - explanation 6-5
  - type 4-5, 6-5
- AUTOEXEC.BAT file
  - explanation 6-3, 6-29
  - purpose 1-9, 4-3
  - setting environment 6-29

# Index

## B

backup  
    EXPORT command 4-49 to 4-52  
bank switching  
    \$EMS.SYS 2-38  
    \$RAMDISK.SYS 2-44  
base memory  
    definition G-1  
    multi-tasking 5-4 to 5-7  
    partition number 5-4, 5-6, 5-11  
.BAT file extension 3-3, 4-6, 6-2  
batch file  
    automated batch files 6-29, G-1  
    color screen displays 6-25 to 6-28  
    commands 6-3 to 6-28  
    compatibility 6-36  
    creating 6-3  
    examples 6-29 to 6-35  
    explanation 6-2, 6-3, G-1  
    nested 6-2, 6-33  
    replaceable operands in 6-2, 6-21  
    startup batch file for a partition 5-11, 5-15  
.BATECHO  
    command form 6-6  
    explanation 6-6  
    type 4-5, 6-6  
baud rate for terminal workstations 5-13, 5-64  
boot sector  
    copying to a disk 4-55, 4-67  
    definition G-1  
    replacing current on a disk 4-67  
braces { }  
    command form conventions 4-9  
.BREAK  
    explanation 4-16  
    form 4-16  
    operands 4-16  
    type 4-5, 4-16  
break points in .DEBUG 8-4



cable connections  
    modems 5-54  
    workstations 5-53  
**CACHE** command statement  
    default values 2-5, 2-6  
    explanation 2-6  
    form 2-5  
    operands 2-5  
    turning on and off (.MOSADM CACHE) 5-46  
    type 2-5, 4-5  
cache of memory 2-5, 5-46  
**.CALL/.RETURN**  
    command form 6-7  
    explanation 6-7  
    type 4-5, 6-7  
**.CD**  
    explanation 4-17  
    form 4-17  
    operands 4-17  
    type 4-5, 4-17  
**CGA** video cards supported 1-3  
    mode settings 5-29  
    snow-like effect corrected 2-16  
characteristics of files 3-4 to 3-5  
**.CLASS**  
    explanation 9-7, 9-8  
    form 9-7  
    operands 9-7  
    type 4-4, 9-7  
**class**  
    definition 1-10, G-1  
    displaying security assignments 3-5  
    print spooler file 10-4 to 10-7, 10-11 to 10-22  
    security 9-2 to 9-22  
**.CLS**  
    batch file usage 6-32  
    explanation 4-19  
    form 4-19  
    type 4-5, 4-19

# Index

cluster  
    definition G-2  
    file allocation check 4-90 to 4-92  
color video screen displays in batch files 6-25 to 6-28  
.COM file extensions 3-3, 4-6  
.COMMAND  
    explanation 4-20 to 4-21  
    form 4-20  
    operands 4-20  
    type 4-5, 4-20  
command  
    batch file commands 6-1 to 6-36  
    configuration command statements 2-2 to 2-36  
    conventions in this manual 4-7 to 4-9  
.DEBUG commands 8-7 to 8-46  
    definition 1-10, G-2  
    editor commands 7-6 to 7-21  
    extrinsic 1-8, 4-3, 4-4  
    general commands 4-11 to 4-94  
    intrinsic 1-8, 4-3, 4-5  
    multi-tasking/multi-user commands 5-11 to 5-18, 5-21 to 5-51  
    print spooler commands 10-8 to 10-15  
    security commands 9-7 to 9-11  
command line  
    definition G-2  
    MOS editing modes 1-13  
    MOS editing control keys 1-14 to 1-15  
Command mode editing with .ED 7-2 to 7-26  
command processor  
    changing to a new layer 4-20  
    definition 4-3, G-2  
    loading 5-4  
    sharing among partitions 5-6  
command recall buffer 1-13  
    clearing 1-13, 6-10, 9-9  
    retrieving lines from 1-13  
COMMAND.COM  
    making a disk bootable 4-55, 4-67  
    purpose 1-8, 4-3



.COMFILE  
explanation 4-22 to 4-24  
form 4-22  
operands 4-22  
type 4-4, 4-22  
computer  
cable connections 5-53  
definition 1-10  
host 5-7  
workstations connections 5-7, 5-53  
COMSPEC= 4-20, 4-84  
CONFIG.SYS file  
command statements 2-2 to 2-36  
defining an operating environment 2-2 to 2-36  
explanation 2-2, 2-3  
purpose 1-9  
control keys for editing  
at MOS system prompt 1-14, 1-15  
in MOS Editor 7-24, 7-25  
conventions  
command form 4-7 to 4-9  
naming files 3-2 to 3-4  
.COPY  
ASCII and binary files 4-30  
changing file dates and times 4-29  
changing a security class 4-31, 9-3, 9-6  
device input/output 4-27  
explanation 4-25 to 4-31  
file concatenation 4-28  
form 4-25  
operands 4-25, 4-26  
secured files 4-31  
type 4-5, 4-25  
COUNTRY command statement  
available codes 2-15  
default value 2-15  
explanation 2-15  
form 2-15  
operands 2-15  
type 2-15, 4-5  
creating files 3-2, 7-2 to 7-4  
critical error messages in MOS A-3 to A-6  
cursor  
insert mode display (block) 1-13  
typeover mode display (underscore) 1-13

# Index

## D

### .DATE

- batch file usage 6-31
- explanation 4-32
- form 4-32
- operands 4-32
- type 4-5, 4-32

### .DEBUG

- break points 8-3
- commands 8-7 to 8-46
- explanation 8-3 to 8-5
- form 8-6
- operands 8-4 to 8-6
- sample session 8-47 to 8-54
- type 4-5, 8-6

default drive 4-7, 4-8, G-3

delimiter 4-7, G-3

### DESNOW command statement

- default value 2-16
- explanation 2-16
- form 2-16
- operands 2-16
- type 2-16, 4-5

### DEVICE command statement

- default value 2-17
- explanation 2-17
- form 2-17
- operands 2-17
- optional device drivers 2-17, 2-38 to 2-50
- standard device drivers 2-17, 2-38
- terminal drivers supplied with MOS 5-12, 5-13, 5-55, 5-56
- type 2-17, 4-5

### device drivers

- adding with .ADDDEV 4-11
- adding with DEVICE= 2-17
- optional 2-38 to 2-50
- provided with MOS 1-8, 1-9, 2-37 to 2-50
- removing with .REMDEV 4-78
- SMPSIZE required for loading 2-29
- standard 2-37
- terminal driver requirements 5-55, 5-56
- terminal drivers supplied with MOS 5-12, 5-13, 5-55, 5-56



device names 2-37 to 2-50

.DIR

    explanation 4-34, 4-35

    form 4-33

    operands 4-33

    type 4-5, 4-33

directory

    access levels under security 9-12 to 9-17

    backslash usage 3-9 to 3-12

    changing (.CD) 4-17

    class 9-7

    current directory 3-9 to 3-14, 4-17

    dedicated for print spooling 10-2

    diagram 3-8

    displaying 3-13, 4-33 to 4-37

    explanation 3-8 to 3-14, G-3

    extension 3-10

    levels 3-9 to 3-14

    making (.MD) 4-61 to 4-62

    maintaining 3-11

    names 3-10

    organizing 3-11

    parent 3-9 to 3-12

    path 3-8 to 3-10

    removing (.RD) 4-76

    root 3-8

    securing 9-2, 9-3, 9-7, 9-12 to 9-17

    signing on to security 9-10

    structure 3-8 to 3-14

    substituting access with .ALIAS 4-13

.DIRMAP

    explanation 4-36

    form 4-36

    operands 4-36

    type 4-4, 4-36

dirname 4-7

disk

    copy entire contents 4-38, 4-39

    definition 1-10, G-3

    formatting 4-55 to 4-57

    making bootable 4-55, 4-67

    naming 4-40

# Index

.DISKCOPY  
    explanation 4-38  
    form 4-38  
    operands 4-38  
    type 4-4, 4-38  
.DISKID  
    explanation 4-40  
    form 4-40  
    operands 4-40  
    type 4-4, 4-40  
displaying files 3-3  
disposition of print spooler file  
    default 10-3 to 10-4  
    changing 10-14, 10-17 to 10-21  
.DOT command 4-6  
drivers (see device drivers)

## E

.ECHO  
    command form 6-9  
    explanation 6-9  
    type 4-5, 6-9  
echoing to the video screen 6-7, 6-9  
    turn off modem echoing 5-64  
.ED (MOS Editor)  
    Command mode explanation 7-2 to 7-5  
    Command mode commands 7-6 to 7-21  
    editing modes 7-2, 7-3  
    graphic character assignments 7-2, 7-6  
        DEFKEYS.ED file 7-6  
    type 4-4, 7-2  
    using macros 7-26  
    Visual mode editing keys 7-24, 7-25  
    Visual mode explanation 7-2, 7-19  
editing  
    command line editing keys 1-14, 1-15  
    MOS Editor - .ED 7-1 to 7-26  
EGA video card  
    conflict with FREEMEM 2-19  
    supported 1-3



**EMS**  
conflict with FREEMEM 2-20  
emulation 1-8  
**encryption for security** 9-20 to 9-22, G-3  
**environment**  
    AUTOEXEC.BAT file 6-29  
    changing  
        .COMMAND 4-20  
        .PATH 4-71  
        .SET 4-84, 4-85  
    definition G-3  
    reserving size for 4-42, 6-30  
**.ENVSIZE**  
batch file usage 6-30  
explanation 4-42  
form 4-42  
operands 4-42  
type 4-5, 4-42  
**.ERASE**  
explanation 4-43, 4-44  
form 4-43  
operands 4-43  
type 4-5, 4-43  
**error**  
    errorlevel conditions in batch file 6-15, G-4  
    messages in MOS A-2 to A-6  
**escape key sequences** 5-58 to 5-61  
**escape sequence chart** 5-61  
**.EXCEPT**  
    CANCEL 4-45 to 4-48  
    DO 4-45 to 4-48  
    explanation 4-45 to 4-48  
    form 4-45  
    operands 4-45  
    type 4-5, 4-45  
**.EXE file extensions** 3-3, 4-3  
**.EXPORT** (also see .IMPORT)  
explanation 4-49 to 4-52  
form 4-49  
operands 4-49  
type 4-4, 4-49

# Index

## extensions

- file names, in 3-2 to 3-3, 4-3, 4-6
- directory names, in 3-10
- print spooler files 10-2 to 10-4
- special file types 3-3

extrinsic commands 1-8, 4-3, 4-4, G-4

## F

### .FILEMODE

- explanation 4-53, 4-54
- form 4-53
- operands 4-53
- type 4-4, 4-53

### files

- accessing in a directory 3-8 to 3-10, 9-17
- attributes 3-4 to 3-5
- backup (.EXPORT) 4-49 to 4-52
- characteristics 3-3 to 3-4
- command form conventions 4-7 to 4-9
- comparing (.COMPPFILE) 4-22 to 4-24
- concatenation 4-28, 4-29
- copying 4-25 to 4-31
- creating 3-2
  - date and time 3-5
  - secured 9-17, 9-18
  - with MOS Editor 7-2 to 7-25
- debugging (see .DEBUG)
- directory groupings 3-8 to 3-14
- displaying 3-3, 4-33 to 4-35, 4-88
- disposition in print file names 10-3 to 10-4
  - changing 10-14, 10-17 to 10-21
- erasing 4-43
- explanation 3-2 to 3-7
- extension
  - .BAT purpose 6-2, G-4
  - .COM purpose 1-8, 4-3
  - .EXE purpose 1-8, 4-3
  - print spooler assignments 10-2 to 10-5
  - .SYS purpose 1-8
  - usage by MOS 3-2, 3-3, 4-3



limiting number of open 5-40  
maintaining 3-7  
naming conventions 3-2, 3-3  
    print spooler files 10-2 to 10-5  
print spooler files 10-2 to 11-5  
provided to customize .HELP facility C-2 to C-4  
provided with MOS 1-8, 1-9  
README file ii  
renaming 4-79 to 4-81  
restoring (.IMPORT) 4-59, 4-60  
securing 9-2, 9-3, 9-11, 9-17, 9-18  
security assignment display 3-4  
sharing 3-5  
signing on to security 9-10  
sizes 3-4  
special processing, used for 1-9, 3-3  
updating 3-4  
    time and date 3-4  
    with MOS Editor 7-2 to 7-26  
verification 4-90 to 4-94  
wildcard characters 3-5

.FLUSH  
    command form 6-10  
    explanation 6-10  
    type 4-5, 6-10

.FOR IN DO  
    command form 6-11  
    explanation 6-11  
    type 4-5, 6-11

foreign keyboard drivers 5-38, 5-39

.FORMAT  
    explanation 4-55 to 4-57  
    form 4-55  
    operands 4-55  
    type 4-4, 4-55

FREEMEM command statement  
    explanation 2-19 to 2-21  
    form 2-19  
    operands 2-19  
    type 2-19, 4-5

# Index

## G

### .GOTO

- command form 6-13
  - explanation 6-13
  - type 4-5, 6-13
- graphic character assignments in the Editor 7-6

## H

hard disk, making bootable 4-55, 4-67

### .HELP

- customizing C-2 to C-4
- explanation 1-3, 4-58
- form 4-58
- operands 4-58
- type 4-4, 4-58

host computer (see computer)

hot key in print spooler 10-15, 10-21

## I

### .IF

- command form 6-14
- explanation 6-14
- type 4-5, 6-14

### .IMPORT

- explanation 4-60
- form 4-59
- operands 4-59
- type 4-4, 4-59

input/output redirection 4-9 to 4-10

- sorting files 4-64 to 4-66
- symbols for 4-9, 4-10

### .INSERT

- command form 6-17
- explanation 6-17
- set command line mode 1-13
- type 4-5, 6-17

insert mode 1-13, 6-17



---

interrupt vector  
    displaying reserved vectors 5-27  
    freeing a reserved vector 5-26  
    reserving 5-23 to 5-27  
    task switching method 5-48  
intrinsic commands 1-4, 4-3, 4-5, G-4  
IRQ (see interrupt vector)

## K

kernel (or core) of MOS 1-3

.KEY  
    command form 6-18  
    explanation 6-18 to 6-20  
    type 4-5, 6-18

key  
    hot key in print spooler 10-15, 10-19, 10-21  
    master password encryption 9-20 to 9-22  
    partition access 5-19

keyboard  
    accessing partitions 5-19  
    command line editing keys 1-14, 1-15  
    command recall buffer 1-13, 6-10  
    disable keyboard checking 5-23  
    emulation of host computer at workstation 5-55 to 5-61  
    escape sequence chart 5-61  
    escape sequences for workstations 5-58 to 5-61  
    foreign keyboard drivers 5-38, 5-38  
    rebooting from 5-9  
    scan code emulation 5-58 to 5-61  
    turning on and off partition access 5-13  
    workstation keyboard emulation 5-58 to 5-61

.KEYMAP Command 5-62

## L

LDEVICE command statement 2-17

levels of security access 9-6  
    directories 9-15, 9-16  
    files 9-17, 9-18  
    partitions 9-12 to 9-14  
levels of directories from the root 3-9 to 3-14  
logical disk G-5

# Index

## M

- main console
  - definition 1-10, G-5
  - securing with master password 9-20 to 9-22
- master password 1-9, 9-20 to 9-22
- .MD
  - explanation 4-61 to 4-62
  - form 4-61
  - operands 4-61
  - type 4-5, 4-61
- MEMDEV command statement
  - explanation 2-22 to 2-25
  - form 2-22
  - operands 2-22, 2-23
  - type 2-22, 4-5
- memory
  - addressable 5-4 to 5-6
  - allocation for partitions 5-5
  - managed environments 1-4, 2-22 to 2-25
  - management 1-4, 2-22 to 2-25
  - management drivers 2-24, 2-25
  - partition (see partition)
- memory paging
  - \$EMS.SYS 2-38
  - \$RAMDISK.SYS 2-44
- mode
  - changing a workstation video mode 5-29
  - setting for color monitor 5-29
  - video mode settings 5-29
- modem
  - baud rate settings 5-64
  - defining types 5-65
  - initialization utility 5-64
  - remote workstation connections 5-64
  - partition set up 5-7
- monochrome video cards supported 1-3
- MORE
  - explanation 4-63
  - form 4-63
  - type 4-4, 4-63



.MOS  
    type 4-4  
    utility command 5-21  
    utility functions 5-9, 5-21 to 5-42  
    utility function display 5-21  
.MOS ANSI 5-42  
.MOS DIS 5-23  
.MOS DOSVER 5-42  
.MOS DSPORT 5-35  
.MOS FILES 5-40  
.MOS FREEIRQ 5-26  
.MOS HOLD LPTn 5-40, 5-41, 10-6  
.MOS INFO 5-33  
.MOS IRQ 5-27  
.MOS KEYB 5-38  
.MOS MAP 5-22  
.MOS MOUSE 2-40, 5-36, 5-37  
.MOS NODIS 5-23  
.MOS RESIZE 5-32  
.MOS ROUTE LPTn to COMn 5-31  
.MOS ROUTE LPTn to LPTn 5-31  
.MOS ROUTE LPTn to NOTERM 5-32  
.MOS ROUTE LPTn to TERM 5-32  
.MOS SERINIT 5-30  
.MOS TSR 5-41  
.MOS USEIRQ 5-23  
.MOS utility command 5-21  
.MOS VMODE 5-29  
.MOS WAIT 5-28  
.MOSADM  
    change system time tick default 2-27  
    help display 5-43  
    type 4-4  
    utility functions 5-9, 5-43 to 5-51  
.MOSADM CACHE 5-46  
.MOSADM EMSLIMIT 5-50, 5-51  
.MOSADM HOLD 5-47, 10-6  
.MOSADM PRI 5-45  
.MOSADM RESET 5-48  
.MOSADM SLICE 5-44  
.MOSADM SWITCH 5-46  
.MOSADM TIME 5-49  
.MOSADM TMFACTOR 5-46, 5-47  
.MOSADM VIRQ 5-48

# Index

mouse  
    device driver 2-40  
    initializing 5-36, 5-37  
.MSORT  
    explanation 4-64 to 4-66  
    form 4-64  
    operands 4-64  
    type 4-4, 4-64  
.MSYS  
    explanation 4-67  
    form 4-67  
    making a disk bootable 4-67  
    operands 4-67  
    type 4-4, 4-67  
multi-tasking  
    commands 5-10 to 5-19  
    command processor, loading 5-4 to 5-7  
    concepts 5-4 to 5-6  
    definition 1-10  
    environment 5-4 to 5-6  
    memory organization 5-4 to 5-6, 2-22  
    processing 5-7  
multi-user  
    advantages 5-8  
    commands 5-10 to 5-19  
    concepts 5-7, 5-8  
    definition 1-10  
    processing 5-8  
    workstations 5-7, 5-8

## N

nested batch files 6-2, 6-33, G-5  
NETBIOS emulation with MOS 1-8, B-2 to B-4  
.NETNAME  
    explanation B-3 to B-4  
    form B-3  
    operands B-3  
.NEXT  
    command form 6-21  
    explanation 6-21  
    type 4-5, 6-21



## O

### .ONLY

CANCEL 4-68 to 4-70  
DO 4-68 to 4-70  
explanation 4-68 to 4-70  
form 4-68  
operands 4-68  
type 4-5, 4-68  
operand  
definition 1-10, 4-8, G-5  
replaceable (in batch files) 6-2, 6-21, 6-34, 6-35  
operating system  
definition 1-2  
output redirection 4-9 to 4-10

## P

parent directory 3-9 to 3-12, G-5  
partition  
access 5-7, 5-19  
with security 9-12 to 9-14  
access keys 5-19  
adding 5-11 to 5-17  
adjusting memory size 5-32  
class, security  
assigning 5-12, 5-15, 9-7  
changing 9-7, 9-8  
security example with .ADDTASK 9-13, 9-14  
communication with pipe devices 2-42  
dedicated for print spooler 10-2, 10-8 to 10-16  
definition 1-11, G-5  
explanation 5-4 to 5-7  
information display, statistical 5-22  
memory size 5-11, 5-32  
numbers 5-4 to 5-6, 5-18, 5-19  
priority, changing 5-45  
processing time ticks 2-27  
rebooting 5-9  
removing 5-18  
resizing 5-32

# Index

- securing 9-2 to 9-4, 9-7, 9-8, 9-11 to 9-14
- sharing 5-7
- signing on to security 9-10
- statistical information display 5-22
- time ticks, changing 5-44
- turning access keys on and off 5-20
- patch file 8-3, G-5
- .PATH
  - batch file usage 6-30
  - explanation 4-71
  - form 4-71
  - in AUTOEXEC.BAT file 4-3
  - operands 4-71
  - type 4-5, 4-71
- path
  - command form conventions 4-8
  - in directory structure 3-9 to 3-14
- .PAUSE
  - command form 6-22
  - explanation 6-22
  - type 4-5, 6-22
- PC-MOS
  - customizing the .HELP facility C-2 to C-4
  - files provided 1-8 to 1-9
  - introduction 1-2 to 1-15
  - modular expansion 1-2
  - release number 4-77
  - single-user, multi-tasking 1-2
  - special features 1-2
  - system monitor 5-68 to 5-70
  - system prompt 1-13
  - terms used in 1-10 to 1-11
  - upgrading to multi-user versions 1-2
  - version number 4-77
- pipe
  - device driver 1-8, 2-42
  - example of adding with .ADDDEV 4-11
  - command form conventions 4-9
  - symbol for redirection 4-9, 4-10



.PRINT  
explanation 10-8 to 10-10  
dedicated directory 10-2, 10-8  
form 10-8  
operands 10-8  
type 4-4, 10-8  
print class 10-3 to 10-5  
    changing 10-14, 10-17, 10-18  
    default value 10-3 to 10-5  
print processor menu 10-17, 10-18  
print spooler menu 10-19 to 10-21  
ports  
    address assignments 2-46  
.ADDTASK port number assignments 2-49, 5-13  
definition G-5  
initializing from a workstation 5-30  
reserved for modem connections 2-49  
\$.SERIAL.SYS device driver 2-46 to 2-50  
standard buffered interface 2-46 to 2-50  
terminal (workstation) connections 2-46, 5-7, 5-8, 5-13  
Print Spooler 1-3, 5-8, 10-1 to 10-24  
    automating 10-16, 10-23, 10-24

# Index

priority  
partition 5-45  
print spooler file  
    changing 10-12, 10-18 to 10-21  
    default value 10-3, 10-4  
program files (see files)  
prompt  
    definition G-5  
    MOS system default 1-13  
.PROMPT  
    batch file usage 6-30  
    explanation 4-73 to 4-75  
    form 4-73  
    operands 4-73, 4-74  
    type 4-5, 4-73

## Q

question mark wildcard character 3-6

## R

RAM (Random Access Memory) 5-4 to 5-6, G-6  
RAM disk  
    emulation 1-9  
    conflict with FREEMEM 2-20  
    \$RAMDISK.SYS 2-44, 2-45  
.RD  
    explanation 4-76  
    form 4-76  
    operands 4-76  
    type 4-5, 4-76  
read-only attribute 3-4  
reboot from keyboard 5-9  
rebooting tasks 5-9  
record sharing 5-8  
redirect processing  
    .GOTO batch file command 6-13  
    sorting files 4-64 to 4-66



.REL  
  explanation 4-77  
  form 4-77  
  type 4-5, 4-77  
.REM  
  explanation 6-23  
  form 6-23  
  type 4-5, 3-26  
.REMDEV  
  explanation 4-78  
  form 4-78  
  operands 4-78  
  type 4-4, 4-78  
removing tasks 5-18 to 5-20  
.REMTASK  
  explanation 5-18 to 5-20  
  form 5-18  
  operands 5-18  
  task ID numbers 5-18  
  type 4-4, 5-18  
.RENAME  
  explanation 4-79 to 4-81  
  form 4-79  
  operands 4-79  
  type 4-5, 4-79  
replaceable operands 6-2, 6-21, 6-34, 6-35, G-6  
resource sharing 1-2, 5-7, 5-8  
root directory 3-9 to 3-14, G-6

## S

.SEARCH  
  explanation 4-82, 4-83  
  form 4-82  
  operands 4-82  
  type 4-4, 4-82

# Index

security 1-3, 1-9, 5-8  
    access levels 9-2 to 9-5  
        directory level access 9-15, 9-16  
        file level access 9-17, 9-18  
        partition level access 9-12 to 9-14  
    advanced 9-19 to 9-22  
    assigning to a partition 5-12, 5-15, 5-16  
.CLASS command 9-7 to 9-8  
classes 9-2  
commands 9-7 to 9-10  
default output class 9-6  
encryption key 9-20 to 9-22  
explanation 9-2 to 9-3  
master password 9-20 to 9-22  
precautions for using 9-19  
.SIGNOFF command 9-9  
.SIGNON command 9-10  
system administrator's record 9-4  
user ID 9-4  
user password 9-5  
user records 9-4 to 9-6  
serial adapter boards 5-53  
serial port  
    .ADDTASK command port assignments 5-13  
    connecting workstations 5-7, 5-13, 5-52  
    initializing from a workstation 5-30  
    modem connections 5-54  
    workstation connections 5-7, 5-13, 5-34  
.SET  
    batch file usage 6-29  
    explanation 4-20, 4-84, 4-85  
    form 4-84  
    operands 4-84  
    type 4-5, 4-84  
SHELL command statement  
    changing system default 2-26  
    default value 2-26  
    explanation 2-26  
    form 2-26  
    operands 2-26  
    type 2-26, 4-5  
    user defined command processor 2-26



shifting operands 6-21  
.SIGNOFF  
    explanation 9-9  
    form 9-9  
    type 4-5, 9-9  
.SIGNON  
    batch file usage 6-32, 9-14  
    explanation 9-10, 9-14  
    form 9-10  
    operands 9-10  
    type 4-5, 9-10  
sign on to network emulation (NETBIOS) B-3  
single-user  
    definition 1-11  
SLICE command statement  
    changing for a partition 5-44  
    default value 2-27  
    explanation 2-27, G-6  
    form 2-27  
    operands 2-27  
    time sharing explanation 2-27, 5-8, 5-9  
    type 4-5  
SMPSIZE command statement  
    default value 2-28  
    explanation 2-28  
    form 2-28  
    operands 2-28  
    type 2-28, 4-5  
.SPOOL  
    dedicated directory 10-11  
    explanation 10-11 to 10-15  
    operands 10-11, 10-12  
    type 4-4, 10-11  
spooler hot key 10-15, 10-18  
spooler menu 10-19 to 10-21  
standard device drivers 2-37  
startup batch file  
    .ADDTASK command 5-12, 5-14  
    print spooler example 10-23, 10-24  
    security example 9-13, 9-14

# Index

## **.STOP**

- command form 6-24
  - explanation 6-24
  - nested in batch files 6-24
  - type 4-5, 6-24
- SunRiver terminal driver 5-35, 5-66
- .SWITCH** command 5-10, 5-20
- .SYS** file extensions 2-38 to 2-50, 3-3
- system administrator
  - definition 1-12
  - invoking security 9-2 to 9-22
  - responsibilities 1-12
  - security record 9-4
- system memory pool (SMP) 2-28
- system monitor 5-68 to 5-70
- system prompt 1-13, 4-73 to 4-75

## T

- task (see partition)
- task ID 5-11, 5-15, 5-18
- terminal (also see workstation)
  - baud rates for communications 5-13
  - device drivers supplied with MOS 5-12, 5-13, 5-55
  - requirements for workstations 5-7, 5-8, 5-55 to 5-57
- .TEXT/.ENDTEXT**
  - background screen colors 6-25
  - color code table 6-27
  - command form 6-25
  - explanation 6-25 to 6-28
  - foreground screen colors 6-25
  - type 4-5, 6-25
- .TIME**
  - batch file usage 6-31
  - explanation 4-86
  - form 4-86
  - operands 4-86
  - type 4-5, 4-86
- time sharing 2-27
  - adjusting rate 5-46, 5-47
  - explanation 5-8, 5-9
- time ticks 2-27, 5-8, 5-9



## .TYPE

explanation 4-88  
form 4-88  
operands 4-88  
type 4-5, 4-88  
typeover mode 1-13, 6-17

## U

### user

default output class 9-6 to 9-8  
security records 9-4 to 9-6

### user ID

assigning 9-4  
displaying file assignments 3-4  
.SIGNON command 9-10

### USERFILE command statement

default value 2-30  
explanation 2-30  
form 2-30  
operands 2-30  
type 2-30, 4-5

### utility functions

multi-user/multi-tasking 5-9, 5-20, 5-51

## V

### .VERIFY

explanation 4-90 to 4-92  
form 4-90  
operands 4-90  
type 4-4, 4-90

### VGA video card

conflict with FREEMEM 2-19  
supported 1-3

### video

cards supported 1-3  
clearing display (.CLS) 4-19, 6-32  
display on terminals 5-56  
handling 1-6  
screen colors in batch files 6-25 to 6-28  
type, for memory allocation 2-31 to 2-36

**VIDPATCH.COM** 1-6  
virtual disk emulation  
  **\$RAMDISK.SYS** 1-9, 2-44  
Visual mode editing 7-22 to 7-25  
Visual mode editing keys 7-24, 7-25  
**VTYPE** command statement 2-31 to 2-36

## W

warning messages in MOS A-2 to A-6  
wildcard characters  
  in command form conventions 4-8  
  in file names 3-5, 3-6  
workstation  
  connecting to host computer 5-52, 5-53  
  definition 1-4, 5-7, 5-8  
  keyboard differences 5-57  
  requirements for 5-56  
  terminal drivers supplied with MOS 5-12, 5-13, 5-55  
  video display differences 5-57  
.WVER  
  explanation 4-93  
  form 4-93  
  operands 4-93  
  type 4-5, 4-93