MOTOROLA

MSET30DOS(D1)

EXORset 30
XDOS OPERATING SYSTEM

User's Guide

SYSTEMS

MICROSYSTEMS

MSET30XDOS(D1)

MARCH 1980

EXORset 30

DISK OPERATING SYSTEM

USER'S GUIDE

First Edition

MANUAL ORGANIZATION

The purpose of this guide is to provide the user with the necessary information required to generate an XDOS system, to use the XDOS command programs, and to produce user-written programs that are compatible with XDOS. In addition, a brief summary is presented of the XDOS-supported software products which are currently available.

The User's Guide has been divided into two parts. PART I is intended for the new user of XDOS who is just receiving his system. It is essentially a manual within a manual that can be read as an entity by itself. It provides the basic concepts that are necessary for the simplified operation of XDOS. PART I contains descriptions and examples of the basic forms of the most frequently used XDOS commands in the order in which they would most likely be used in a software development environment. The infrequently used commands are also summarized in order to direct the user to those chapters (command descriptions) as the need for their use arises.

PART II is intended as a detailed reference manual for those who need to know specific or extended information about the XDOS commands, the system structure, and the resident system functions.

TABLE OF CONTENTS Page

TABLE OF CONTENTS Page

PART II -- ADVANCED XDOS USER'S GUIDE

APPENDICES

CHAPTER 1

1. INTRODUCTION

The EXORset 30 is a single-sided, single-density mini-diskette drive system. Two diskette drives and their controller are included in the EXORset 30.

The EXORset 30 Diskette Operating System (XDOS) is a subset of the EXORciser Disk Operating System (MDOS). Used in conjunction with the hardware environment of the EXORset 30 (micromodule compatible), it provides a powerful and easy-to-use tool for software development. XDOS is an interactive operating system that obtains commands from the system console. These commands are used to move data on the diskette, to process data, or to activate user-written processes from diskette. All this can be accomplished with a minimum of effort; and since XDOS is a facilities oriented system, rather than a supervisory oriented one, a minimum of overhead is imposed.

In addition, an extensive set of resident system functions are provided for general development use. Such functions as dynamic space allocation, random access to data files, record I/O for supported and non-supported devices, as well as many register, string, and other diskette-oriented routines make XDOS a good basis for a user's application system.

1.1 Hardware Support Required

The minimum hardware configuration required to support XDOS consists of:

-- an EXORset 30 with EXORbug firmware
-- 16K RAM

The above minimum configuration will allow the user to run any of the XDOS commands that reside on the XDOS system diskette at the time of purchase. Other additional hardware may be required to run the XDOS-supported software products. Such information is described in Appendix H.

1.2 Additional Supported Hardware

XDOS also supports a line printer: The line printer interfaces to the EXORset 30 through the printer interface which consists of one PIA located on the EXORset main board.

1.3 Software Support Required

No additional software is required to run the operating system as it comes shipped on the system diskette.

1.4 Program Compatibility

All of the XDOS commands and system files that are shipped on the system diskette must be used with that particular version of XDOS. XDOS commands and system files from other versions should never be intermixed.

Most user-written assembly language programs that were developed independently of XDOS can be executed on an XDOS system without reassembly; however, such programs will have to be converted into the memory-image file format before they can be loaded from diskette into memory (see section 2.8.5). Programs need only be changed when transferred to XDOS if:

1. They make assumptions about the initialization of the stack pointer after they are loaded into memory,

2. They are origined to load (initialize memory while loading) below hexadecimal location $20,

3. They make assumptions about the physical structure of diskette tables or files,

4. They utilize the diskette for input/output,

5. They make assumptions about the contents of the SWI and IRQ interrupt vectors.

If a user has software that he has developed using the EXORciser and MDOS, then Appendix J should be consulted for a list of differences between XDOS 3.00 and MDOS that may affect programs running with XDOS 3.00.

1.5 Hardware Installation

The EXORset 30 should be inspected upon receipt for broken, damaged, or missing parts as well as for damage to the printed circuit boards. The packing materials should be saved in case reshipping is necessary.

1.6 Software Installation

There is no software installation that need be performed. All XDOS software is included on the diskette that is shipped with each EXORset 30. This diskette contains the operating system and a set of commands that comprise XDOS. It may or may not contain any of the XDOS-supported software products such as editors or assemblers. These products are dependent on the mode of system purchase.

CHAPTER 2

2. GENERAL SYSTEM OPERATIOI

This chapter provides the user with the basic concepts that are necessary for the simplified and typical operation of XDOS. It contains descriptions and examples of the initialization procedures and of the basic forms of the most frequently used commands. These examples clearly illustrate how XDOS is used to edit a program, to assemble it, to convert it into a loadable module, to load it and execute it, as well as some other useful operations. The commands are presented in a sequence that is commonly followed in a software development environment.

2.1 System Initialization

To initialize the operating system, power must first be applied to the EXORset 30. Once the power is on, the following steps must be followed:

1. The prompt "EXORBUG V.R" will be displayed by EXORbug indicating it is waiting for operator input. "V" indicates the version and "R" the revision number of the EXORbug monitor in the system. If the power was already on and XDOS must be reloaded, press the restart button located on the main board through the back panel.

2. An XDOS diskette (one shipped from Motorola or one that has been properly prepared by the user (see section 2.8.9)) must be placed in drive zero. The door on the drive unit must then be closed in order for the diskette to begin rotating. When shipped from Motorola, the left side drive is configured as drive zero, the right side drive as drive one, as seen from the front.

   The diskette must be oriented properly before being inserted into the drive. When the diskette is inserted properly, the label is facing right, and the edge of the diskette with the long narrow slot in the protective covering is inserted first. The labelled edge will be the last edge to be covered up as the diskette is inserted into the drive.

3. The EXORbug "XDOS" command must then be

enterd. It will give control to the diskette controller at address $E800. The controller will initialize the drive electronics and then proceed to read the Bootblock into memory. Once the Bootblock has been loaded, control is transferred to it. The Bootblock will then attempt to load into memory the remainder of the resident operating system.

2.2 Sign-on Message

If no errors occur during the initialization process, XDOS will display the message:

```
XDOS VV.RR
=
```

meaning that XDOS has been successfully loaded from disk and initialized. The "VV" and "RR" indicate the version and revision numbers of the operating system, respectively. In addition, an equal sign (=) is displayed as a prompt indicating that XDOS is ready to accept commands from the operator. The equal sign prompt is subsequently displayed each time the XDOS command interpreter gets control. The sign-on message showing the version and revision numbers is only displayed when XDOS is reloaded from the diskette.

2.3 Initialization Error Messages

If for some reason the drive electronics are not properly initialized, or if the diskette in drive zero cannot be read properly to load the Bootblock or the resident operating system, then a two-character error message will be displayed and control returned to the EXORbug monitor.

The following errors can be produced during initialization. All two-character messages begin with the letter "E".

Message	Probable Cause
E1	A cyclical redundancy check (CRC) error was detected while reading the resident operating system into memory.
E2	The diskette has the write protection tab punched out. During the initialization process, certain information is written onto the diskette.

The diskette is not damaged and can still be used for a system diskette; however, the write protection tab must first be covered with a piece of opaque tape to allow writing on the diskette.

E3 The drive is not ready. The door is open or the diskette is not yet turning at the proper speed. If the diskette has been inserted into the drive with the wrong orientation, the "not ready" error will be also generated.

Closing the door, waiting a little bit longer before entering the "XDOS" command, or turning the diskette around so it is properly oriented should eliminate this error.

E4 A deleted data mark was detected while reading the resident operating system into memory.

E5 This error status is returned when the track address has not been found after five attempts.

E6 The diskette controller has been presented with a track-sector address that is invalid.

This error indicates some type of a hardware problem. For example, the error can be caused by missing or overlapping memory, bad memory, or pending IRQs that cannot be serviced.

E7 A seek error occurred while trying to read the resident operating system into memory.

Like E6 errors, this one indicates some type of a hardware problem.

E8 A data mark error was detected while trying to read the resident operating system into memory.

E9 A CRC error was found while reading the address mark that identifies sector locations on the diskette.

The diskette controller errors E1, E4, E8, and E9 indicate that the diskette cannot be used to load the operating system; however, a new operating system can be generated on that diskette, making it useful again. Chapter 8, DOSGEN command, and chapter 10, FORMAT command, describe ways in which damaged diskettes can be regenerated. Depending on the extent of the errors, the diskette may be used in drive one to recover any files that may be on it (section 2.8.8).

The diskette controller error E5 can occur for a variety of reasons. The most common reason, and the most fatal, is the destruction of the addressing information on the diskette. If the addressing information has been destroyed (verified by using DUMP command to examine areas of diskette), the FORMAT command may be used to rewrite the addressing; however, information on the damaged diskette cannot be recovered. Occasionally, after a system has just been unpacked, the read/write head may have been positioned past its normal restore point on track zero. In this case, trying the event which caused the error three or more times may position the head to the proper place. If this fails, the head will have to be manually repositioned past track zero; however, this problem rarely occurs. The E7 errors can occur if a user-written program accesses drive 1 without using one of the system functions and without first restoring the read/write head on that drive.

Even after the resident operating system has been successfully read into memory, certain errors can occur in the subsequent initialization procedure. During initialization the resident operating system cannot access the error message processor since it has not been initialized. Messages similar in format to those generated by the diskette controller are displayed to indicate such errors. They differ from the diskette controller errors in that the second character of the two-character message is a non-numeric character. The following errors can occur during initialization, but only after the resident operating system has been read into memory.

Message	Probable cause
E?	This error indicates that the Retrieval Information Block (RIB) of the resident operating system file XDOS.SY is in error. The operating system cannot be loaded. The diskette probably is not an XDOS system diskette, or the system files have been moved from their original places.

EM This error indicates that there was insufficient memory to accommodate the resident portion of the operating system.

The memory requirements described in section 1.1 should be reviewed. If the minimum requirements are satisfied, then the existing memory should be carefully examined for bad locations.

EI The version and revision of XDOS already loaded into memory are not the same as those on diskette. This error usually occurs as the result of switching diskettes in drive zero without following the initialization procedure outlined in section 2.1. The error can also occur if the ID sector has been damaged.

The error can be avoided if the initialization procedure is followed correctly every time a new system diskette is inserted into drive zero.

ER The addresses of the Retrieval Information Blocks of the XDOS overlays are not the same as those at the time of the last initialization. This error may occur for the same reasons as the "EI" error.

EU An input/output system function returned an error during the initialization. Errors of this sort indicate a possible memory problem or the opening of the door to drive zero while the initialization is taking place.

EV One of the system files is missing or cannot be loaded into memory. If a system file is missing, the diskette has been improperly generated or the file was intentionally deleted. If a file cannot be loaded, then the diskette should be regenerated. The diskette may be used in drive one to save any files that may be on it (section 2.8.8). This error may also occur if the door to drive zero is opened while initialization is in

progress.

EN — A NMI has occured and the XDOS NMI vector (NMI$VC) was not initialized. This error may also occur after completion of the initialization.

EQ — An IRQ has occured and the XDOS IRQ vector (IRQ$VC) was not initialized. This error may also occur after completion of the initialization.

EF — A FIRQ has occured and the XDOS FIRQ vector (FIR$VC) was not initialized. This error may also occur after completion of the initialization.

ES — A SWI2 has occured and the XDOS SWI2 vector (SW2$VC) was not initialized. This error may also occur after completion of the initialization.

EW — A SWI3 has occured and the XDOS SWI3 vector (SW3$VC) was not initialized. This error may also occur after completion of the initialization.

2.4 Operator Command Format

After the sign-on message is displayed, XDOS is ready to accept commands from the operator. The equal sign prompt (=) indicates that the command interpreter is awaiting input via the console. Generally, the equal sign prompt will be redisplayed after each command has finished its function. The operator-entered command line must always indicate which command is to be executed. In addition, the file names that may be required by the command must be specified. Some commands also allow various options that can alter the way in which their functions are performed. These options are also entered on the command line. Each command line must be terminated with a carriage return. The command line has the following format:

<name 1> <name 2>,<name 3>,....,<name n>;<options>

where each <name i> (i=1 to n) has the form of a complete XDOS file name (see section 2.7.1). The name of the command to be executed is always <name 1>. The remaining names and the options may not be required, depending on the individual command. The following lines:

```
DIR EDIT.CM:1;E
FREE
MERGE FILE1:1,FILE2:0,FILE3:1,FILE1:1
```

are valid examples of XDOS command lines. Section 2.8 describes in a simplified form the basic format (i.e., the command's name, what file names must be specified, and what options are available) of the most frequently used commands. PART II gives a complete and detailed description of all XDOS commands. In addition, Appendix H contains a summary of the command line formats of all XDOS-Supported software products.

Most frequently a "space" is used to separate <name 1>, the command name, from the other names which are typically separated by "commas". The "semicolon" always separates the options from the rest of the command line. The "space" and "comma" are the recommended separators since they make the command line the most readable; however, any character that will not be mistaken for an XDOS file name character, a suffix delimiter, a logical unit number delimiter, or a device name delimiter (see section 2.7.1) can be used as a separator. The use of special characters, although permitted, is not recommended because the command line becomes very unreadable.

2.5 System Console

The system console is used as the communications device between the operator and the operating system. XDOS messages are displayed on the EXORset screen. XDOS commands, as well as operator inputs prompted by the commands, are entered via the keyboard. All command line input and most input to the various commands requires upper case, alphabetic characters. Numeric and special characters, of course, are case independent. To allow corrections to be made to any typed line before the terminating carriage return is entered, several special keys on the keyboard can be used. In addition, two other special keys serve to prematurely abort a command in progress or to "freeze" the display of messages on the console.

2.5.1 Carriage return key

The CARRIAGE RETURN key is used to terminate any operator response to an XDOS input prompt. This is true for the command line as well as all other input that may be required from the operator by the various commands. The CARRIAGE RETURN will automatically perform both carriage return and line feed functions.

2.5.2 Control-P

Control-P is actually a combination of two keys being depressed simultaneously: the CONTROL or CTL key and the P key. Depressing control-P is recognized as a special controlled abort function key. Most XDOS commands that take a long time to complete their function periodically check to see if the control-P has been depressed. If it has, the command will come to a premature, but controlled, termination point.

The control-P should be used, whenever possible, as an alternative to using the EXORset RESTART pushbutton or the keyboard BREAK key (The BREAK key is the equivalent of the EXORciser ABORT pushbutton). The controlled abort that is achieved with the contol-P ensures that all system tables are intact. Since termination is delayed until all critical diskette accesses have been completed, no file space is lost nor is any system table destroyed. Such precautions cannot be guaranteed if the BREAK key or RESTART pushbutton are used, since the operator has no way of knowing whether or not diskette data transfers are in progress.

2.5.3 Control-W

Control-W is actually a combination of two keys being depressed simultaneously: the CONTROL or CTL key and the W key. This combination is used to halt the display of information on the system console or printer. All commands that respond to the Control-P abort function will also be "haltable" with the CTL-W key. Most XDOS commands that display more than a few lines of information on the console will occasionally check to see if the CTL-W key has been depressed. If a CTL-W is detected, the command will suspend processing until any other key on the console keyboard is depressed (except, of course, another CTL-W). This feature is particularly useful to hold the display if the output rate is too high to read the messages. In addition, if output is being directed to the printer, the CTL-W can be used to suspend printing until the paper is realigned.

2.5.4 Control-X

Control-X is actually a combination of two keys being depressed simultaneously: the CONTROL or CTL key and the X key. This combination is used to cancel the input line that was just entered by the operator (before a carriage return is depressed). All system inputs from the console support CTL-X. Any characters entered on the current input line thus far will be deleted and input can be resumed from the beginning of the line. A carriage return and line feed will be sent to

the console, so that the operator has a positive feedback that the line was cancelled.

2.5.5 DEL or RUBOUT

The DEL or RUBOUT key serves as a backspace key during console input. If the operator detects an error in the current input line (before a carriage return is depressed), the DEL key will cause the preceding character to be removed from the input line. The character that is removed will be echoed back to the console so that the operator has a positive feedback that a character was backed out of the line.

2.5.6 Control-D

Control-D is actually a combination of two keys being depressed simultaneously: the CONTROL or CTL key and the D key. This combination allows the operator to re-display the current input line (before a terminating carriage return is depressed). If the input line has had several characters backed out (see DEL key above), the line is very unreadable. The CTL-D key can, therefore, be used to show a "clean" copy of the line for operator inspection. The newly displayed line will be shown on the line following the current input line. Operator input is not terminated with the CTL-D key. Any remaining input must still be supplied, as well as the terminating carriage return.

2.6 Common Error Messages

Many error messages are common to the XDOS commands. In order to be aware of the most common errors, their descriptions are included here. These common error messages will be recognizable to the operator since they are prefaced with a pair of asterisks (**) and a two-digit reference number. Each command may, in addition, have a set of specific error messages that will not be displayed by other commands. These specific error messages will not have the asterisks or two-digit reference number. Such messages are explained along with each command's detailed description in PART II. A summary of the standard error messages can be found in Chapter 21. The messages are listed there in order of their two-digit reference numbers.

WHAT?

> The first name entered on the command line was not the name of a file in the diskette's directory. Most often this error occurs as the result of a mistyped command name.

** 01 COMMAND SYNTAX ERROR

The syntax of the command line parameters could not be interpreted. Most often this error refers to undefined characters appearing in the options field.

** 02 NAME REQUIRED

The file name required by the command as a parameter was omitted from the command line.

** 03 <name> DOES NOT EXIST

The displayed file name was not found in the diskette's directory. The file name must exist prior to using the command. The <name> is displayed to show which name of the multiple names specified as parameters caused the error.

** 04 FILE NAME NOT FOUND

The file name entered on the command line as a parameter does not exist in the diskette's directory. The file name must exist prior to using the command. No file name is displayed, since only one parameter is required by the command.

** 05 <name> DUPLICATE FILE NAME

The displayed file name already exists in the diskette's directory. The file name must not exist prior to using the command. The <name> is displayed to show which name of the multiple names specified as parameters caused the error.

** 06 DUPLICATE FILE NAME

The file name entered on the command line as a parameter already exists in the diskette's directory. The file name must not exist prior to using the command. No file name is displayed, since only one parameter is required by the command.

** 07 OPTION CONFLICT

The specified options were not valid for the type of function that was to be performed by the command. Several of the options are mutually exclusive and cannot be specified at the same time.

** 11 DEVICE NOT READY

Most frequently this indicates that a command is trying to output to the printer while the printer is not ready.

** 12 INVALID TYPE OF OBJECT FILE

Most frequently this indicates that an attempt was made to load a program into memory whose file does not have the "loadable" memory-image format, e.g., a source file.

** 13 INVALID LOAD ADDRESS

An attempt was made to load a program into memory that: 1) loads outside of the range of contiguous memory established at initialization; 2) loads over the resident operating system; 3) loads below hexadecimal location $20; or 4) loads beyond hexadecimal location $FFFF.

** 25 INVALID FILE NAME

A file name was specified that contained a family indicator (*), that began with a device name indicator (#), or that did not begin with an alphabetic character, or that contains a non-alphanumeric character.

** 41 INSUFFICIENT DISK SPACE

A command is trying to create a file or to write into a file. Upon trying to allocate more file space, insufficient room remains on the diskette to accommodate the space requirements.

**PROM I/O ERROR--STATUS=nn AT h DRIVE i-PSN j

An unrecoverable error occurred while trying to access the diskette. The error status "nn" is a value returned by the diskette controller. The errors are of the same type that cause the initialization process to give control to EXORbug; however, instead of beginning with the letter "E", the status (nn) begins with the digit "3". The second digit of the status corresponds directly to the diskette controller error number discussed in section 2.3. The "E" has been replaced by the "3". Thus, status

31 is the same as E1
32 is the same as E2
.
.
.
39 is the same as E9.

A memory address (only meaningful for system diagnostics) is substituted for the letter "h"; the drive number is substituted for the letter "i"; and the physical sector number (PSN) at which the error occurred is substituted for the letter "j".

2.7 Diskette File Concepts

In XDOS, a diskette file is a set of related information that is recorded more or less contiguously on the diskette. The information can be actual machine instructions that comprise a command or user program. The information can also be textual data, object program data, or any of the forms described in Chapter 17. The following section describes how files are named, created, deleted, and protected.

2.7.1 File name specifications

An XDOS file name specification consists of three parts: a "file name", a "suffix", and a "logical unit number". File names can be from one to eight alphanumeric characters in length, the first of which must be alphabetic. The alphabetic characters must be upper case letters. Valid file names could look like the following:

```
DIR
DATA115
BACKUP
S0
XDOSNEWS
Z
```

In most cases, all that need be specified when a file name specification is called for is the file name. The suffix and logical unit number are usually given appropriate default values by the various commands.

The suffix can be either one or two characters in length. Like file names, suffixes must begin with an upper case alphabetic character. The rest of the suffix must be alphanumeric. A suffix is used to explicitly refer to a particular entry in the directory. That is, there may be several entries with the same file name but with different suffixes. In such cases, a file name reference alone would be

ambiguous. Thus, the suffix is used to differentiate between entries with the same file name. Usually, suffixes designate a particular format of the file. Thus, a source file could have the suffix "SA". Its assembled object version could have the same file name but with the suffix "LX", and its executable version could have the same file name but with the suffix "LO". XDOS commands usually supply an appropriate default suffix when dealing with specific files.

If both file name and suffix are specified, they must be separated by a period (.). The following are examples of valid file name specifications using both file name and suffix:

```
XDOSNEWS.SA
BACKUP.CM
Z.SA
PROC1.CF
DOCUMENT.Y
```

Since each diskette is a complete file system in itself, with complete directory and system files, it is possible to have directory entries with the same file names and suffixes on separate diskettes. Thus, the logical unit number is required to uniquely specify a directory entry on a given drive. Logical unit numbers consist of a single decimal digit (0 or 1). In most cases, XDOS commands supply a default value for the logical unit number. If a particular drive must be identified, it must be entered by the operator as a part of the file name specification. Logical unit numbers follow either the file name or the suffix depending on whether one or both are specified. The logical unit number must be separated from the file name or from the suffix by a colon (:). The following are examples of valid file name specifications using logical unit numbers.

```
BACKUP.CM:0
TEST.X:1
DIR:1
Z456.D3:0
ASM:1
```

2.7.1.1 Family names

Some commands allow the operator to specify a family of file names. Family indicators can occur in either the file name or the suffix. An asterisk (*) is used as a family indicator. The family indicator represents all or part of a file name or suffix. For example,

```
FILE.*
```

would be a file name specification that includes all

directory entries with the file name "FILE " but with any suffix on the default drive. Similarly,

PROG*.SA

is a file name specification that includes all directory entries with "PROG" as the first four characters of their file names, regardless of what the remaining characters are, and with suffix "SA" on the default drive. The asterisk cannot have characters following it. Thus, the following file name specifications are invalid:

*PROG.SA
PROGRAM.*B

Not all commands allow file name specifications to contain the family indicator. The individual command descriptions should be consulted to see where family indicators are acceptable.

2.7.1.2 Device specifications

Some commands allow the operator to enter a device specification in the command line instead of a file name specification. Device specifications consist of two parts: a "device name" and an optional "logical unit number". Device names are two characters long, both of which must be alphabetic. A pound sign (#) is used as a leading character to indicate that the subsequent two-character sequence is a device name. For example,

#LP
#CN

are valid device names used for the line printer and the console, respectively. A device specification may be entered with a logical unit number. Logical unit numbers must follow the device name and must be separated from it by a colon (:). The individual command descriptions should be consulted to see where device specifications are allowed.

2.7.2 File creation

XDOS files are never explicitly created by the operator. All commands that write to output files will create them automatically if they do not exist. Files will be created according to the file name specification given on the command line. That is, if explicit suffixes and logical unit numbers are specified, the file will be created on the indicated drive. Otherwise, the appropriate default values supplied by the command will be used to create the file. Existing files are unaffected by the creation of a new file.

2.7.3 File deletion

Unlike file creation, file deletion is controlled explicitly by the operator via the DEL command which is described later. No other command program will delete existing files on the diskette. Exceptions to this are commands that automatically create an intermediate work file to perform the command's function. These intermediate files are deleted by the command as an automatic clean-up process.

2.7.4 File protection

All XDOS files can be configured with delete protection, with write protection, or with no protection. Delete protection will prevent the operator from inadvertently deleting the file (the protection can be changed by the operator so that the file can be deleted). Write protection will prevent any command from writing to that file as well as preventing deletion of the file. Normally, files are unprotected, allowing both writing to or deletion of the file. The NAME command, described later, can be used to set or to change a file's protection.

2.8 Typical Command Usage Examples

The following sections give simple, but meaningful, descriptions and examples of the most frequently used XDOS commands in a typical software development environment. No attempt is made in these sections to cover all capabilities and options of the described commands. The detailed command descriptions in PART II serve that purpose. After reading this section, the operator should be able to go "on-line" with XDOS and be able to display the directory of a diskette, create a source program file, assemble it, and load it into memory for testing. The commands to delete a file, to change its name or protection, to copy it between diskettes are also described. New XDOS diskette generation is discussed in the last part of this section.

It is assumed in the subsequent discussion that the system has been properly installed and initialized. Thus, a system diskette with the XDOS commands resides in drive zero. Command program files have a suffix of "CM" which is supplied as a default to the first file name that is entered on the command line. The default logical unit number that is supplied is ":0". In the command examples that follow, it will be seen that both suffix and logical unit number are not specified for the command name.

The following notation will be used in the description of the command line formats as well as throughout the

remainder of the manual:

Notation	Meaning
$nnnn	Hexadecimal number "nnnn".
<>	Syntactic elements are printed in lower case and are contained in angle brackets, e.g., <options>, <name>.
[]	Optional elements are contained in square brackets. If one of a series of elements may be selected, the available list of elements will be separated by the word "or", e.g., [<tag1> or <tag2>].
{}	A required element that must be selected from the set of elements will be contained in curly brackets. The elements will be separated by the word "or".

All elements that appear outside of angle brackets (<>) must be entered as is. Such elements are printed in capital letters (if words) or printed as the actual characters (if special characters). For example, the syntactical element [;<options>] requires the semicolon (;) to be typed whenever the <options> field is used.

2.8.1 DIR -- Directory display

The DIR command is used to display the contents of a diskette's directory. Either the entire directory or selective parts of it can be displayed. The format of the command line for the DIR command is:

DIR [<name>] [;<options>]

The file name specification <name> indicates what to display. The <options> specification indicates how to display it. If DIR is entered by itself on the command line, it will display on the system console the file names of all user-generated files on drive zero. If no user-generated files exist on drive zero, a message will be displayed indicating that no directory entries were found. This is normally the case when DIR is used without any options on the system diskettes that are shipped with the new system. To display the system and the user-generated files, the "S" option can be placed into the options field:

```
DIR ;S
```

If drive one's directory is to be displayed, then a ":1" must be typed in place of the file name specification:

```
DIR :1;S
```

To direct the output of the DIR command to the printer, only one other option letter need be specified -- "L". Thus,

```
DIR :1;LS
```

will produce a listing of drive one's complete directory on the printer. The "S" and "L" can be in any order, as long as they follow the semicolon.

The DIR command can also be used to see if a specific file name exists on a given drive. This is accomplished by entering a complete file name specification (i.e., name, suffix, and logical unit number). Thus,

```
DIR EDIT.CM:1
```

will perform a directory search for the indicated file name specification on drive one. If the directory entry exists, its file name and suffix will be displayed. Otherwise, a message indicating that no entries were found will be displayed. Directory searches for specific file names do not require the "S" option to distinguish between system files and user files. Chapter 7 contains a complete description of the DIR command's use.

2.8.2 EDIT -- Program editing

The EDIT command is used to create and/or to change user-written source program and data files on diskette. If the EDIT command resides on the system diskette, it is invoked with the following XDOS command line:

```
EDIT
```

If the EDIT command is not copied to the system diskette, it can be invoked from the diskette in drive one with the following command line:

```
EDIT:1
```

No parameter may be supplied in the command line. A complete description of the EDIT command's format and usage is found in the relevant manual.

2.8.3 ASM -- Program assembling

The ASM command (hereafter called the assembler) is used to assemble the source program files created with the EDIT command. The assembler translates these source programs into object programs. If the assembler resides on the system diskette, it is invoked with the following XDOS command line:

ASM <name> [;<options>]

If the assembler is not copied to the system diskette in drive zero, it can be invoked from the diskette in drive one by using the following command line:

ASM:1 <name> [;<options>]

The only required parameter is the name of the file that is to be assembled. Normally, this would be the name of the file specified in the previous description of the EDIT command. The assembler will automatically supply the default suffix for both the source file that is read (SA) and for the memory image file that is created (LO). If its name is not specified, the memory image file will have the same file name as <name>, but a different suffix will be assigned to it to differentiate it from the source file.

Normally, a listing of the assembled program is desired. The assembler will not produce a source listing unless the option to do so is specified in the <options> field. Thus, the command line to assemble a source program file named TESTPROG with source listing output would appear as:

ASM TESTPROG;L

As with the DIR command, the "L" option directs the printed output to the printer. If a printer is not available, or if the program is short, the source listing can be produced on the system console by using the following option:

ASM TESTPROG;L=#CN

If errors are detected during the assembly process, they will be included on the source listing. If no source listing is being produced, errors will automatically be displayed on the console. Typically, the software development process involves several iterations of the editing and assembly processes before an error-free object file is produced. The assembler, however, requires that the object file does not exist prior to the assembly process. Therefore, if a duplicate file name error message is displayed, the object file already exists. It must first be deleted before the assembly process can continue. The next section describes the process of file deletion.

During the iterative process of editing/assembling to obtain an error-free program, the object file created by the assembler can be suppressed by specifying the option "-O" in the options field. The command line

```
ASM TESTPROG;L-O
```

for example, will assemble the source program as in the above examples creating the listing on the line printer; however, the object file will not be created. Thus, the deletion of the object file between repetitive assemblies is not required since it is never created.

The relevant manual should be consulted for a complete description of the assembler's function, usage, and command format.

2.8.4 DEL -- File deletion

The DEL command is used to delete file names from the directory. The removal of a file's name from the directory makes the file unaccessible to any other process. The file itself is effectively deleted. Thus, in the subsequent descriptions, the phrases "delete a file name" and "delete a file" are equivalent. The format of the command line for the DEL command is:

```
DEL <name>
```

which will cause the specified file to be deleted. If the object file from the assembly process example above is to be deleted, for instance, the following command line would be entered:

```
DEL TESTPROG.LX
```

It should be noted that the suffix is specified. Since the DEL command is a general purpose command, like the DIR command, no default value for the suffix is supplied. Only those commands that can validly make an assumption about the type of file they will be dealing with (e.g., EDIT, ASM) will supply default suffixes.

The DEL command will display a message indicating that the file name was deleted or that the file name was not found. Chapter 6 contains a complete description of the DEL command's other capabilities.

2.8.5 LOAD -- Program loading/execution

The LOAD command is used to load programs from a memory-image file on the diskette into memory. After the

program has been loaded, the debug monitor can be given control (for testing the program), or the program can be given control directly (for execution). The format of the command line for program loading is:

LOAD <name> [;<options>]

The name of the file whose contents are to be loaded is given as <name>. The default suffix "LO" is automatically supplied by the LOAD command. Thus, in normal software development, only a file's original source program name is required to take a user through the three processes of editing, assembling and program loading.

The <options> field of the LOAD command line is used to specify whether the debug monitor or the loaded program is to be given control, and whether or not the program overlays the resident operating system. If the file TESTPROG from the previous examples was origined to the hexadecimal memory address $100, the following command line:

LOAD TESTPROG;V

would be used to load the program. The "V" option is used to specify that the program to be loaded will overlay the resident operating system. If the "V" option were left off the command line, an error message would be displayed. The absence of the "G" option letter means that the debug monitor will be given control after the program is loaded. So, the above example would be used to load TESTPROG into memory for testing.

If, on the other hand, the program TESTPROG has already been tested and works, the command line:

LOAD TESTPROG;VG

would be used to load and execute the program. No operator intervention is required to specify the starting execution address. This is only true if the starting execution address has been specified on the END statement of the source program during the assembly process.

Typically, most user-written programs that have been developed prior to receiving the XDOS system would be loaded and tested in this fashion. Programs that are developed with XDOS as a basis (i.e., programs that use the resident system functions) are loaded without the "V" option. Chapter 13 describes the details of the LOAD command and should be consulted if more information is required.

2.8.6 NAME -- File name changing

The NAME command allows file names and/or suffixes to be changed from their originally assigned values. Often, as a program is developed, its author decides that a file name other than the original one would be more appropriate and descriptive. The format of the command line for changing a file's name is:

NAME <name 1>,<name 2>

This command line requires the operator to enter two names. The first name, <name 1>, specifies the current or original name of the file. The default suffix "SA" is supplied automatically if none is given by the operator. The second name, <name 2>, indicates the new name that is to be assigned to the file now known by <name 1>. Thus, if the file from the above examples, TESTPROG, were to be given a more descriptive name, such as BLAKJACK, the following command would be used:

NAME TESTPROG,BLAKJACK

In this case, only the file name of the source file would be changed. Other files with the name TESTPROG but with suffixes other than "SA" would remain unaffected. The contents of the file that has its name changed are also unaffected -- only the name in the directory is changed.

2.8.7 NAME -- File protection changing

The NAME command is also used to change the protection attributes of a file. The command line format for changing a file's protection is:

NAME <name>;<options>

The <name> entry is required to identify the file whose attributes are to be changed. The <options> field contains the letters D, W, or X to indicate how the protection attributes are to be changed. The letters take on the following meanings:

D -- Set delete protection
W -- Set write protection
X -- Set no protection (remove existing protection)

Thus, if the file TESTPROG (source file) is to be protected against deletion, the following command line would be used:

NAME TESTPROG;D

If the memory-image file that was produced from the source of TESTPROG were to be write protected and delete protected, the following command line would be used:

```
NAME TESTPROG.LO;DW
```

The protection on this file could later be removed with the command line:

```
NAME TESTPROG.LO;X
```

Chapter 15 describes in more detail the other features of the NAME command.

2.8.8 COPY -- File copying

The COPY command is used to make a duplicate copy of a file on a single diskette, to move a file between two different diskettes, or to move a file between a peripheral device and a diskette.

To make a duplicate copy of a file on the same diskette, the following command line is used:

```
COPY <name 1>,<name 2>
```

where <name 1> specifies the current name of an existing file, and <name 2> specifies the name of the duplicate copy. The default suffix "SA" and the default logical unit number zero are supplied for <name 1> if those parts of the file name specification are omitted. Normally, the destination file, <name 2>, does not exist. The COPY command, however, will alert the operator if <name 2> does exist, and ask him if that file should be overwritten. If <name 2> has a different logical unit number than the original file, the file will be duplicated on the specified drive. If the TESTPROG source file from the above examples is to be saved in a file called TEMP, the following command line would be used:

```
COPY TESTPROG,TEMP
```

The file TEMP will be created on the same drive as TESTPROG, namely, drive zero. To copy TESTPROG to drive one, one need only specify the logical unit number (:1) after the second name.

```
COPY EDIT.CM:1,:0
COPY ASM.CM:1,:0
```

would be the commands that are entered if the diskette in drive one contained these files. The suffixes "CM" are

explicitly specified since neither the EDIT or ASM commands are source programs.

A similar procedure would be followed to copy any files from a diskette in any drive to the system diskette in drive zero. If a diskette has been damaged or cannot be used to initialize XDOS, it may be placed in another drive in attempt to save any files that may be on it. The COPY command should be used to save files in this manner. If diskette controller errors occur during such a save process, the files cannot be recovered.

If a user wants to transfer external data to a disk file, he has to write his own driver, then invoke the COPY command as follows:

COPY #UD,<name 2>;D=<name 3>

where <name 2> is the name of the diskette file into which the data are to be written. The first parameter, #UD, specifies the user driver as source device, and the "D=<name 3>" option specifies the memory image file name in which the driver may be found (see 5.2 and 19).

The above process can be changed slightly so that a file on diskette can be written to a user defined peripheral. For example,

COPY <name 1>,#UD;D=<name 3>

will transfer the file named by <name 1> to the user device through the output driver found in <name 3>. Chapter 5 describes in more detail the other features of the COPY command.

2.8.9 BACKUP -- XDOS diskette creation

New diskettes, or diskettes never before used on an XDOS system, must first be prepared for use with XDOS. The quickest way to generate a new XDOS diskette is to use the BACKUP command. Usually, a copy is retained of the original system diskette that was shipped with the EXORset 30. This diskette should be used to generate subsequent XDOS diskettes. It is recommended that the original diskette not be used for development purposes. It should serve only as the master copy from which all other diskettes are generated.

A formatted blank or scratch diskette should be placed into drive one. The master system diskette should be resident in drive zero. The following command line will then cause a complete copy of the master diskette to be created:

BACKUP ;U

The "U" option specifies that the entire surface of the diskette in drive zero is to be read and copied to the diskette in drive one. This process ensures that all sectors on the new diskette can be written to. Once the BACKUP command has been invoked in this way, it will display the following message:

BACKUP FROM DRIVE 0 TO 1?

to which the operator should respond with a "Y". Any other response will terminate the BACKUP process, leaving the diskette in drive one intact. The "Y" response will cause the diskette copy to take place.

As an added precaution, the two diskettes should be compared against each other after the BACKUP command has completed. This diskette verification is invoked with the following command line:

BACKUP ;UV

If any messages are displayed during the verification process, the diskette in drive one should not be used as a system diskette.

Chapter 3 describes the BACKUP command in detail. Chapter 8 describes an alternative method of generating new system diskettes.

2.9 Other Available Commands

Several other powerful commands are included with each XDOS diskette. These commands are not needed initially in becoming familiar with the system; however, they do provide helpful and necessary tools for the advanced software developer. A brief description of these commands is given here to shed some light on their utility.

2.9.1 BACKUP -- Diskette copying

The BACKUP command allows making copies of entire XDOS diskettes. Options exist for making complete copies, for file reorganization to consolidate fragmented files and available diskette space, for appending families of files from one diskette to another, and for diskette comparisons. Chapter 3 contains the complete description of the BACKUP command.

2.9.2 MERGE -- File concatenation

The MERGE command allows one or more files to be concatenated into a new file. This command is useful in

combining several smaller program modules. several smaller program modules. Chapter 14 contains the complete description of the MERGE command.

2.9.3 FREE -- Available file space display

The FREE command displays how many unallocated sectors and how many empty directory entries are on a diskette. Chapter 11 contains the complete description of the FREE command.

2.9.4 CHAIN -- XDOS command chaining

The CHAIN command allows predefined procedures to be automatically executed. A procedure consists of any sequence of XDOS command lines that have been put into a diskette file. Instead of obtaining successive command lines from the console, CHAIN will fetch commands from a file. This feature allows complicated and lengthy operations to be defined once, and then invoked any number of times, requiring no operator intervention. The additional capability of conditional directives to the CHAIN command at execution time permits an almost unlimited number of applications to be handled by a CHAIN file. Chapter 4 contains the complete description of the CHAIN command.

2.9.5 DUMP -- Diskette sector display

The DUMP command allows the user to examine the entire contents of any physical sector on the diskette. The sector can be displayed on either the system console or the printer. The display contains both the hexadecimal and the ASCII equivalent of every byte in the sector. The DUMP command allows opening of files so that they can be examined using logical sector numbers. Sectors can also be moved into a temporary buffer where changes can be applied before they are written back to diskette. Chapter 9 contains the complete description of the DUMP command.

2.9.6 FORMAT -- Diskette reformatting

The FORMAT command attempts to rewrite the sector addressing information on damaged diskettes. Upon receipt from supplier, diskettes may not be formatted. They then need be formatted before they can be used. The FORMAT command must be used to initialize them. Chapter 10 contains the complete description of the FORMAT command.

2.9.7 DOSGEN -- XDOS diskette generation

The DOSGEN command allows specialized XDOS diskettes to be prepared. Diskettes that have bad sectors can have those sectors locked out so that the diskette can be used in an XDOS environment. DOSGEN will also create all system tables and files on the generated diskette. The DOSGEN command can be used to generate system diskettes on either single-sided or on appropriately formatted double-sided diskettes. Chapter 8 contains the complete description of the DOSGEN command.

2.9.8 ROLLOUT -- Memory rollout to diskette

The ROLLOUT command is used for writing the contents of memory to diskette. The ROLLOUT command does support the alternate map feature of the EXORset 30. Options exist for writing memory directly into a diskette file or for writing to a scratch diskette. Chapter 16 contains the complete description of the ROLLOUT command.

2.10 XDOS-Supported Software Products

Although the preceding list of commands provides the user with many powerful tools for software development, there are some other Motorola products which are capable of running in an XDOS environment, even though they were developed independently. • These products are called XDOS-Supported software products. No attempt will be made in this User's Guide to comprehensively describe any XDOS-Supported software product. Appendix H contains a list (complete at time of publication) of all products that can be invoked from an XDOS diskette as a command. Each description will contain the additional hardware requirements, if any, the command line formats, and a brief discussion of the product's capabilities. XDOS-supported software products may be received on separate diskettes. Section 2.8.8 describes how such products can be copied onto the system diskette.

2.11 Paper Alignment

All XDOS commands that output to the line printer will return the paper to its original position upon termination. Thus, if the paper is correctly aligned at the time XDOS is initialized, then the paper will never have to be aligned again. The paper should be placed so that the print line is positioned three lines before a perforation (assuming fan-fold forms). XDOS commands use the standard format of 66 lines/page.