FlexOS™
User's Guide

FLEXOS™

USER'S GUIDE

Second Edition: August 1987

Software Version: FlexOS 386 1.4

1073-2004-002

COPYRIGHT

DISCLAIMER

DIGITAL RESEARCH MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. Further, Digital Research Inc. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Digital Research Inc. to notify any person of such revision or changes.

NOTICE TO USER

From time to time changes are made in the filenames and in the files included on the distribution disk. This manual should not be construed as a representation or warranty that such files or materials and facilities exist on the distribution disk or as part of the materials and programs distributed. Most distribution disks include a "README.DOC" file. This file explains variations from the manual which do constitute modification of the manual and the items included therewith. Be sure to read this file before using the software.

TRADEMARKS

Second Edition: August 1987

Foreword

FlexOS™ is an operating system designed for microcomputer systems. FlexOS controls the flow of data between various application programs such as word processors or spreadsheets (software), and the individual components of your computer system such as video display terminals, disk drives, and printers (hardware).

FlexOS is both a multiuser and multitasking operating system. Multiuser means that more than one person at a time can use the system. Multitasking means that you can run several programs simultaneously, moving from one to another as you need to instead of starting and stopping each program. Multitasking also means that software like a print spooler can print files unattended while you continue to edit with a word processor or perform calculations with a spreadsheet. This helps you work faster and more efficiently.

What This Guide Contains

The FlexOS User's Guide (hereinafter called the User's Guide describes the FlexOS **user interface**, or how the user interacts with FlexOS. Topics include the following:

- how to enter commands, including some of the most commonly used FlexOS commands
- how FlexOS handles files and lets you organize them into directories and subdirectories
- how you can use logical names to substitute for long or complex names of files or physical devices
- how to make FlexOS direct its input and output to and from files and various physical devices
- how to direct FlexOS to perform a sequence of several commands with a single command (batch processing)
- how FlexOS reports errors

What You Need To Know

This book assumes you are already familiar with personal computers, including the general terminology used to describe hardware and software. You should read and be familiar with the documentation that accompanies your computer before proceeding.

The FlexOS Documentation Set

The User's Guide is one of several manuals in the FlexOS documentation set. Other user-oriented documentation includes:

- FlexOS User's Reference Guide: This book is a reference description of each FlexOS command.
- FlexOS Configuration Guide: This book describes system configuration and maintenance, including how to install new hardware devices and software, and how to format a hard disk.
- FlexOS Window Manager Guide: This book describes how to use the FlexOS Window Manager provided by Digital Research®.
- Running DOS Applications under FlexOS: This book describes how to install and run certified DOS application software under FlexOS.

The following books are written for application programmers and system developers. End users do not need them to use FlexOS.

- FlexOS Programmer's Guide: This book describes how to write applications and utilities to run under FlexOS.
- FlexOS System Guide: The book describes how to modify FlexOS to run on different computer systems. Information presented in this guide includes driver and supervisor interfaces, FlexOS's driver services, and how to construct a boot loader.
- FlexOS Supplements: Microprocessor-dependent supplements to the Programmer's Guide and the FlexOS System Guide that supply information about FlexOS for a specific microprocessor.
- FlexOS Programmer's Utilities Guides: The reference to FlexOS assembly language programming tools. There is a separate utilities guide for each microprocessor supported by FlexOS.

Terms and Conventions

This book uses the terms "diskette," "hard disk," and "disk" in the following manner:

diskette	The $5^{1/4}$-inch or $3^{1/2}$-inch magnetic disks that can be inserted into and removed from your microcomputer. Your system has at least one diskette drive.
hard disk	An internal or external fixed magnetic disk.
disk	The term used when information applies to both diskettes and hard disks.

User Input and FlexOS Output

In this book, user input is in **boldface** type and FlexOS system output, including the system prompt A>, is in regular type. The following example illustrates the convention:

```
A>DIR | FIND "ACCOUNTS PAY"

ACCOUNTS PAY          1429    8-18-86    1:42P
```

In this example, user input is all uppercase letters. However, you can enter commands in uppercase, lowercase, or a combination of the two.

Contents

1 Basic FlexOS Operation

2 I/O Redirection, Chaining, and Filtering

3 Batch Processing

Basic FlexOS Operation

1.1 System Prompt and Command Line

The most common user interaction with FlexOS involves listing directories, copying or deleting files, and starting application programs. To perform these tasks, you type a "command line" and then press the Enter key.

The FlexOS command line has the following components:

```
A>COMMAND FILESPEC -OPTION
```

A> is the **system prompt**, the place on the terminal screen where you enter commands. On most computers the system prompt is a letter (usually A, B, or C, designating the current, or "default," drive) followed by a greater-than symbol (>) and a blinking cursor that signifies the computer is ready to receive input.

command[1] can be either a built-in FlexOS command, an external command, or a batch file. **Built-in** commands are loaded into your computer's memory when you start FlexOS; they remain in memory until you turn off your computer. You can enter a built-in command at any time. DIR and TYPE are examples of built-in commands.

External commands are contained in disk files and not loaded into memory when you start FlexOS; they must be accessed from the disk. If the command is on a diskette, you must insert the diskette into your computer before entering the command. Executable files that start application programs such as word processors or spreadsheets are examples of external commands.

A **batch** file is a text file which contains one or more FlexOS commands. When you enter the name of the batch file, FlexOS executes each command in the file in sequence. Executing a batch file is equivalent to separately executing each command in the file. (Batch files are discussed in Chapter 3, "Batch Processing.")

filespec stands for "file specification", which is the name, and optionally, the location of a file on which you want to perform an operation (like COPY). Some commands require a filespec, and some commands accept more than one filespec. Section 1.2 describes FlexOS file specifications.

An **option** lets you modify what a command does. Multiple options are separated from each other by hyphens (-).

---

[1]The User's Reference Guide contains a complete description of each FlexOS command including its options.

1.1.1 Editing a Command Line

You can correct or cancel a command line before you press the Enter key by using various keys or control key combinations. The control key combinations involve pressing the Control key (Ctrl -- represented by a caret "^") while simultaneously pressing a letter key. For example, the notations **Ctrl-X** and **^X** both indicate that you press the Ctrl key while pressing the X on your keyboard.

Table 1-1 summarizes the command line editing keys and control key combinations.

Table 1-1. Command Line Editing Keys

Key	Description
left-arrow	Moves cursor one character to the left without affecting characters in the command line.
right-arrow	Moves cursor one character to the right without affecting characters in the command line.
Backspace	Moves cursor one character to the left, deleting the character at that position. Characters following the cursor move one position to the left.
Delete	Erases the character under the cursor. The cursor does not move, but characters following the cursor move one position to the left.
Ctrl-X	Deletes the entire command line. The cursor appears immediately after the system prompt.
Ctrl-R	Redisplays the most recently executed command line. You can then view, change, or reenter that command line.
Enter	Causes FlexOS to execute the command line.

1.1.2 Interrupting Command Execution

The Ctrl-S, Ctrl-Q, and Ctrl-C key combinations let you suspend, resume, and cancel commands that have started executing. These keys are summarized in Table 1-2.

Table 1-2. Keys For Interrupting Commands

Key	Description
Ctrl-S	Suspends the execution and output of a running command. The command runs until it needs to display output, then FlexOS suspends it. The output remains suspended until you resume the command with Ctrl-Q, or cancel the command with Ctrl-C.
Ctrl-Q	Causes a command suspended with Ctrl-S to resume running where it was suspended. Any output in progress when you pressed Ctrl-S also resumes.
Ctrl-C	Deletes a command line before you press Enter, or stops a command from further execution. Generally, you should not use Ctrl-C to stop a running program or application, because results can be unpredictable.

1.2 Files and Directories

In FlexOS, a complete filespec has the following components:

DRIVE:/DIRECTORY_PATH/FILENAME.EXT

where **drive:** is the designation of a disk drive (either floppy disk or hard disk), and **filename** is the name of the file. **ext** is an optional file extension that usually indicates something about the file's contents.

A **directory** is a collection of files. The topmost directory on each disk is called the **root directory**, and each directory contained within the root is called a **subdirectory**. The **directory_path** is the sequence of directory names from the root directory to a specified directory. The directory_path is optional with most commands.

Figure 1-1 illustrates a typical directory structure.

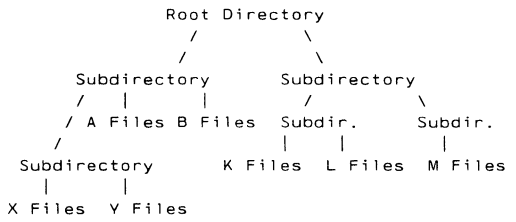


Figure 1-1. Directory Structure

When you format a disk, it initially contains only the root directory. To create subdirectories, use the MKDIR command. To change your current directory path, use the CHDIR command.

1.3 Accessing Subdirectories

When using a filespec with a command like DIR, you must separate directory names with a slash (/) or backslash (\). Be sure to specify all the subdirectories in the directory path in order, as well as the name of the file itself. Your path or filespec, from the root directory to the level you want to access, must be no more than 128 characters.

The following examples illustrate how to access subdirectories:

- If you are in the root directory, or you are in a subdirectory and want to access a subdirectory below your current level, begin the filespec with the name of the next subdirectory in the path, not preceded by a backslash, as in the following command:

```
C>DIR SUBDIRECTORY/SUBDIRECTORY/FILENAME.EXT
```

- If you are in a subdirectory and want to access a subdirectory above your current level, or you want to access a subdirectory in another path entirely, begin the filespec with a backslash (representing the root directory) and then give the complete path specification, as in the following command:

```
C>DIR /SUBDIRECTORY/SUBDIRECTORY/FILENAME.EXT
```

1.4 Basic File Operations

Table 1-3 describes four commands that perform basic FlexOS file operations.

Table 1-3. FlexOS File Operations

Command	Description
COPY	Copies one or more files from disk to disk, or from directory to directory, either on the same disk or on different disks. The COPY command has the form: COPY source_filespec destination_filespec destination_filespec can be simply a drive identifier, as in the command line "COPY MYFILE.TXT B:".

Table 1-3. (Continued)

Command	Description
TYPE	Displays the contents of a text (ASCII) file on the screen. The TYPE command has the form: TYPE filespec
RENAME	Gives one or more files new names, removing the original names from the directory. The RENAME command has the form: RENAME old_filespec new_filespec
ERASE	Deletes one or more files from the default or specified directory. The ERASE command has the form: ERASE filespec

1.5 Wildcards

FlexOS uses three **wildcard** characters to give you greater flexibility in your file specifications: the asterisk (*), the question mark (?), and the caret (^).

Wildcards can take the place of one or more characters in a directory name, filename, or file extension; they make it possible for a command to affect more than one file or directory at the same time. You can combine the asterisk and question mark wildcards in a single specification.

1.5.1 Asterisk (*) Wildcard

The asterisk wildcard takes the place of a variable number of characters in a filename, directory name, or extension. The number of characters the asterisk can represent depends on the specific case, as the following examples illustrate:

*.DAT — In this filespec, the asterisk represents from one to eight characters, so a command using this wildcard could specify any filename or directory name with the extension DAT. For example, S.DAT and MONDAYS.DAT both match this specification.

B*.*T — In this filespec, the asterisk before the dot (period) represents from zero to seven characters; the asterisk after the dot represents zero to two characters. A command using this filespec could specify any file or directory whose name begins with B and whose extension ends with T. For example, BIOLAB.TXT, B.DAT, and BARGAINS.T all match this specification.

M*N.TXT In this filespec, the asterisk represents from zero to six characters. A command using this filespec could specify any file or directory whose name begins with M and ends with N and whose extension is TXT. For example, MOON.TXT, MN.TXT, and MASON.TXT all match this specification.

1.5.2 Question Mark (?) Wildcard

The question mark wildcard takes the place of a single character at the wildcard's position.

The following examples illustrate the question mark wildcard.

TEST?.DAT This specifies any five-character file or directory name whose first four characters are TEST and whose extension is DAT. For example, TEST1.DAT and TESTG.DAT both match this specification; TESTNEW.DAT does not.

TE?TS.* This specifies any five-character file or directory name that begins with TE and ends with TS, and with any extension. For example, TEXTS.BAK and TESTS.TXT both match this specification.

DI??ONE.C This specifies any seven-character file or directory name that begins with DI and ends with ONE, with the extension C. For example, DIALONE.C and DISKONE.C both match this specification.

1.5.3 Caret (^) Wildcard

The caret wildcard excludes any files or directories that match the specification from the action on the command. The caret must be the first character in the specification. For example, the command:

```
A>RENAME ^*.DAT *.DAT
```

rename all the files or directories that do not have the extension DAT to have the extension DAT. The result of this command is that all files or directories in the current directory have the extension DAT.

1.5.4 Placing Wildcards

You can only use a wildcard in the rightmost (last) component of a filespec. The following command contains valid wildcards:

```
A>DIR B:/GRADES/MATH/*.*
```

The following command contains an invalid wildcard:

```
A>DIR B:/GRADES/?ATH/PERIOD1.DAT
```

1.6 File Attributes

A file can have one or more **attributes** that define how FlexOS handles the file. You can set file attributes with the FSET command. Table 1-4 lists the FlexOS file attributes.

Table 1-4. FlexOS File Attributes

Attribute	Description
-H	Hidden file. If -H=ON, the file becomes a **hidden** file, which means it does not appear in your directory unless you use the DIR command with the -H option. If -H=OFF, the file appears in your directory. If you do not specify a value for -H, -H=OFF is the default. However, if your file is also a system file (-S=ON), it does not appear in your directory even if -H=OFF.
-A	Archive file. If -A=OFF, it indicates the file has been archived since it was created or last modified. -A is automatically turned on whenever you modify your file. Programs that back up your files can turn off archiving so they can back up files that have changed since the last backup.
-S	System file. If -S=ON, the file becomes a **system** file. A system file may be accessed by any user on the system without the user needing a copy of the file in his or her directory. Applications that are used by many people should have the system attribute. A system file does not appear in directory listings unless you use the DIR command with the -H option. If -S=OFF, the file is not a system file and it appears in the directory listing. If you do not specify a value for -S, -S=OFF is the default. Note however, that if the file is also a hidden file (-H=ON), it does not appear in the directory listing even if -S=OFF.
-R	Read-Only file. If -R=ON, the file becomes a **Read-Only** file. You can only read the file, even if your User or Group ID (see Section 1.7 give you other privileges. -R=OFF turns off the Read-Only attribute. If you do not specify a value for -R, -R=OFF is the default.

1.7 User Classes

Under FlexOS, each user has both a User and Group Identification (ID) number that determines file access priviledges and ensures against accidental or unauthorized access.

Note: On a multi-user system, the system manager assigns you a User and Group ID when he sets up your account. If you are on a single-user system, you assign yourself a User and Group ID. See the Configuration Guide for more information on setting up user accounts.

There are three classes of user based on User and Group ID numbers:

owner	You are the owner of every file you create.
group	The group class consists of all users (including the owner) with the same Group ID number.
world	The world class consists of all users in the owner and group classes.

1.7.1 Determining File Access

When you try to access a file, FlexOS compares your User and Group IDs with the file's ownership information. If your User and Group IDs match the owner information, you are given the owner privileges of the file. If only your Group ID matches, you are given group privileges. If neither your User or Group IDs match the owner information, you are given world privileges.

1.7.2 Class Priviledges

Each class of user can have one or more of the following privileges:

R	Read the specified file.
W	Write to the specified file. To modify the contents of a file, you must have the write privilege.
E	Run the specified file (if the file is a program).
D	Delete part or all of the specified file. To change the security of a file, you must either have owner access or have the delete privilege.

1.7.3 Directory Versus File Access Privileges

The User and Group IDs that qualify users for access to individual files also apply to directory access. However, directory access privileges have a slightly different meaning than they do for files. Table 1-5 compares the two meanings.

Table 1-5. Privilege Definitions for Files and Directories

Privilege	For Files	For Directories
Read (R)	Allows you to read from a file	Allows you to list files in the directory
Write (W)	Allows you to write to a file (plus delete/set privileges)	Allows you to create and delete files in the directory
Execute (E)	Allows you to execute a file	Allows you to open files in the directory
Delete/Set (D)	Allows you to rename files, change file attributes, or delete files	Allows you to change attributes of files in directory

1.8 Logical Names

Logical names allow you to substitute a shorter name for all or part of a complex device name or filespec. You can also substitute one logical name for another.

For example, you could substitute the logical name CONIFER for the directory pathname A:/FLORA/TREES/CONIFER/. Whenever you need to invoke that path, you would simply enter CONIFER.

There are three types of logical names: logical devices, logical filenames, and logical information. Each is described later in this section.

1.8.1 Creating and Using Logical Names

To create a logical name, use the DEFINE command. For example, to create the logical name CONIFER:, you would issue the following command:

```
A>DEFINE CONIFER:=A:/FLORA/TREES/CONIFER/
```

Logical names have the following requirements:

- They can be up to nine characters. The original "long form" of your path or filespec cannot exceed 128 characters.
- They must be the first name following the command.
- There can only be one logical name on a command line.

When FlexOS finds a logical name on the command line, it replaces the logical name with the original name and then executes the specified command.

1.8.2 Logical Devices

A logical device name takes the place of a device or directory path name. Logical device names end in a colon (:).

The CONIFER: example given previously is a logical device name. If you have a text file in the CONIFER directory called PINES.DAT, use the following command line to display it:

```
A>TYPE CONIFER:PINES.DAT
```

FlexOS accesses drive A, the directory path /FLORA/TREES/CONIFER/, and the file PINES.DAT. Because the logical name includes a drive identifier, you can enter the command line from any disk drive.

You can use a logical device name to replace part of a directory path. For example, you could substitute FTREES: for A:/FLORA/TREES/. To access PINES.DAT, enter the following:

```
A>TYPE FTREES:CONIFER/PINES.DAT
```

This would allow you to access other subdirectories of TREES, perhaps one named FRUIT, with a file called APPLE.DAT:

```
A>TYPE FTREES:FRUIT/APPLE.DAT
```

1.8.3 Logical Filenames

Logical filenames can completely replace a filespec or just a filename and extension. When you replace only the filename and extension, FlexOS assumes the file is in the current directory of the current drive. Logical filenames do not have to include a colon.

One use of a logical filename could be to invoke an executable file from anywhere in the system. For example, if you have a text editing program called TXTEDIT.EXE in the root directory of drive B, you can give it the logical filename EDIT with the following command:

```
A>DEFINE EDIT=B:TXTEDIT.EXE
```

If you are on drive C and want to edit a file called REPORT.TXT in the current directory, type the following:

```
C>EDIT REPORT.DAT
```

Note: Remember you can only use one logical name per command line. You cannot combine a logical device name and a logical filename like this:

```
A>EDIT CONIFER:PINES.DAT
```

1.8.4 Logical Information

"Logical information" is a general term for the other kinds of logical names you can define. For example, you can change your system prompt by redefining the logical name PROMPT, as in the following example:

```
A>DEFINE PROMPT=Your wish is my command>

Your wish is my command>_
```

1.8.5 Reserved Logical Names

FlexOS reserves certain logical names for its own use. You must use them as intended or you may get unpredictable results. Table 1-6 lists the reserved logical names and how they should be used.

Table 1-6. Reserved Logical Names

Logical Name	Definition
a:, b:, ... p:	device
addmem	additional memory request
bgprn:	background printer
boot:	system boot directory path
bottom	FlexOS internal use only
con	console (keyboard and screen)
con:	physical console
default:	device
groupid	group identification number
helplvl	message help level
home:	current default home directory path
left	FlexOS internal use only
lptX:	parallel device (X = 0, 1, 2, etc)
mouse:	console pointer device
name	user LOGON name
nul	null device
null:	null device
order	command type search order
path	search path for commands
pi:	pipe device
prn	printer
prn:	printer
prompt	system prompt
protect	system protection flag

Table 1-6. (Continued)

Logical Name	Definition
right	FlexOS internal use only
ser:	serial device
shell	default shell
stdcmd	standard input/output (I/O) command stream path name
stderr	standard output error message stream path name
stdin	standard input stream path name
stdout	standard output stream path name
switchar	options separator
system:	default system directory path
tempdir:	temporary directory for print spooler
top	FlexOS internal use only
userid	user identification number
vc00X	window number (X = 0, 1, 2, etc)
vdiXX	virtual device drivers (X = 0, 1, 2, etc)
version	operating system and hardware identification
wmanager	default window manager program
wmessage	message pipe for window manager

1.9 Reserved File Extensions

Table 1-7 lists the file extensions (also called filetypes) reserved for use by FlexOS.

Table 1-7. Reserved File Extensions

Extension	Meaning
A	80386 assembly language source file
A86	8086 or 80286 assembly language source file
BAT	Batch file
CMD	CP/M-86® or Concurrent™ DOS native-mode command file
COM	DOS command file
EXE	DOS command file
INP	Input file of file names and options
LIB	Indexed library file of commonly used object modules
LIN	LINK 86 output file with line number symbols
LST	Output assembly language listing with error messages
L86	Indexed library file of commonly used object modules
MAP	Linker output file with segment information
O	Common object file format (COFF) object file
OBJ	Intel 8086/80286 relocatable format (IOMF) object file
SRL	Shared runtime library file
SYM	Symbol file with user-defined symbols
XRF	8086 assembly language symbol cross-reference file
286	FlexOS 286 command file
386	FlexOS 386 command file

End of Section 1

I/O Redirection, Chaining, and Filtering

2.1 I/O Redirection

The most common type of user interaction with FlexOS is typing input at the keyboard and seeing output displayed on the screen. FlexOS also provides you with the capability to redirect input/output (I/O) so you can use data from files or devices as input, and send output to a file, a device, or another command.

2.1.1 Redirecting Output to a File or Device

The > (greater than) symbol directs the output from a command to a specified file or device. If the file does not exist, FlexOS creates it.

Example:

```
A>DIR > PROGRAMS.LST
```

This command sends the output from the DIR command to the file PROGRAMS.LST.

2.1.2 Adding Output to a File or Device

The >> symbol takes the output from a command and adds it to the end of an existing file or device.

Example:

```
A>DIR B: >> PROGRAMS.LST
```

This command adds the directory of drive B to the end of the file PROGRAMS.LST on the default drive (drive A).

2.1.3 Getting Input from a File or Device

The < (less than) symbol uses a file or device as the input to a command.

Example:

```
A>MORE < QUOTAS.LST
```

```
SALES QUOTAS FOR FEBRUARY
-------------------------
Anderson, R.      $ 5,000.00
Blake, J.         $12,500.00
.
.
.
Whitfield, D.     $ 3,000.00
-- More --
```

In this example, the MORE command displays information from the file QUOTAS.LST. (MORE is one of the "filters" described later in this section. It allows you to display output one screen at a time.)

2.1.4 Piping Commands

The term **piping** means to take the output of one command and makes it the input to another command. To pipe between commands, place the vertical bar (|) symbol between the command names on the command line.

For example, suppose the DIR command gave this directory of files on drive B:

```
CLEANUP  BAT      25   6-06-1986   9:34a
MEMOS             25984   3-23-1986  11:03a
CATALOG  LST      17664   6-01-1986  10:13p
LETTERS           16256   4-16-1986   5:57p
```

Using the command:

```
A>DIR B: | SORT
```

you can sort the directory alphabetically, by piping the output from DIR to the SORT filter (described later in this section) to produce the following directory listing:

```
%P00001  $$$      0   6-08-1986  10:18a
%P00002  $$$      0   6-08-1986  10:18a
CATALOG  LST      17664   6-01-1986  10:13p
CLEANUP  BAT      25   6-06-1986   9:34a
LETTERS           16256   4-16-1986   5:57p
MEMOS             25984   3-23-1986  11:03a
```

The two new files with the $$$ extension are temporary files created by FlexOS.

2.2 Chaining Commands on a Command Line

Chaining allows you to enter more than one command in a single command line. The ! (exclamation mark) symbol separates the commands. FlexOS executes the commands in the order in which they appear in the command line.

The following command line displays the current directory of drive A, renames a file, and displays the contents of another file. Note that FlexOS displays each command line it runs.

```
A>DIR ! RENAME NOW.EVT THEN.OLD ! TYPE MSG.TXT
```

```
  Volume in Drive A: has no label
  Directory of  A:

JANUARY       <DIR>       1-06-1986   9:34a
APRIL         <DIR>       3-30-1986  11:03a
ACCOUNTS PAY     1429     1-18-1986   1:42p
ACCOUNTS REC    13421     1-03-1986   8:46a

           6 Files     324256 bytes free

A>RENAME NOW.EVT THEN.OLD

A>TYPE MSG.TXT

To All Employees:

    Please turn in your monthly status
reports by noon on Friday.

        Thank you.
```

Table 2-1 summarizes the FlexOS I/O redirection symbols.

Table 2-1. I/O Redirection Symbols

Symbol	Function
>	Redirects the output from a command to a file or device. COMMAND > FILESPEC
>>	Adds the output from a command to the end of a file or sends to a device. COMMAND >> FILESPEC
<	Uses a file or device as input to a command. COMMAND < FILESPEC
|	"Pipes" the output of the first command to become the input for the second command. COMMAND1 | COMMAND2
!	"Chains" a series of commands on a single command line. The commands are executed in order. COMMAND ! COMMAND ! COMMAND

Table 2-1. (Continued)

Symbol	Function
>*	Redirects the command's standard error file to a file or device. COMMAND >* FILESPEC
>>*	Adds the command's standard error file to the end of a file or device. COMMAND >>* FILESPEC
|*	Uses the standard error file of the first command as the input for the second command. COMMAND1 |* COMMAND2

Note: You can combine two or more ways of redirecting I/O on the same command line, but you cannot use I/O redirection symbols within a filespec.

2.3 Filtering Data

Filters are FlexOS commands that receive input from a file or command, perform a special operation, and send the resulting output to the screen, a file, device, or command. Filters alter the way information is displayed; they do not alter the contents of your files, or affect the output of your commands.

There are three FlexOS filters:

FIND — Searches one or more files for a specified text string; searches the output of a command for a specified string. FIND accepts options that display the number of matches found or the line number of the matches, as well as an option that excludes the string from the output.

MORE — Displays a file or the output of a command one screen at a time.

SORT — Sorts file input or command output alphabetically. SORT accepts options that sort in reverse alphabetical order and that sort on the characters in a specified column.

The User's Reference Guide contains detailed descriptions of the filters.

End of Section 2

Batch Processing

A batch file is a text file which contains one or more FlexOS commands. When you enter the name of a batch file on the command line, FlexOS executes each command in the file from top to bottom.

All batch files have the file extension BAT. When you run a batch file, you do not include the file extension.

Your batch file runs until all the commands within it have been run, or you cancel the batch file by pressing Ctrl-C. As long as you do not use any FlexOS commands in your batch file that change the current directory, FlexOS returns you to the directory from which you started your batch file.

FlexOS allows you to use I/O redirection with batch files. You can redirect I/O both from the command line and from within a batch file. Redirection from within a batch file overrides redirection invoked from the command line. Chapter 2 describes the FlexOS I/O redirection facilities.

To run a batch file, enter one of the following forms on your command line:

```
batch_filename
drive: batch_filename
batch_filename parameters
drive: batch_filename parameters
```

Parameters are values the batch file uses to accomplish the specified commands.

The following example illustrates batch files. Suppose you created a batch file called CLEANUP.BAT that consisted of the following commands:

```
CHDIR /
DIR *.BAK
ERASE *.BAK
```

If CLEANUP.BAT is in your current directory of drive A, you can run it by entering the command:

```
A>CLEANUP
```

After you press Enter, FlexOS runs the commands in order:

1. CHDIR makes the root directory the current directory.
2. DIR displays all backup (BAK) files in the root directory.
3. ERASE deletes all backup (BAK) files from the root directory.

When the processing stops, you are left in the root directory of drive A and FlexOS displays the system prompt:

```
A>
```

If you are on drive B and want to run CLEANUP.BAT on drive A, enter the command:

```
B>A:CLEANUP
```

CLEANUP.BAT runs its commands against the files on drive A. When it is finished, it returns to drive B in the directory where you started.

3.1 Using Parameters in Batch Files

Batch files are very useful; however, it is impractical to create a specific batch file for every situation you may encounter. The solution is to create general purpose batch files that can be used in many situations.

To create a general purpose batch file, replace specific information in the batch file, such as drive letters and file names, with "placeholders." Then, when you run the batch file, supply the missing values on your command line. These command line values are called "parameters."

You can have up to 10 different placeholders in a batch file. A placeholder is a percent sign followed by a digit 0-9.

FlexOS takes the first parameter on your command line and substitutes it for each %1 in your batch file. FlexOS substitutes the second parameter for each %2, and so on.

The placeholder %0 is a special case. When FlexOS sees %0 in a batch file, it replaces %0 with the drive name (if you specify one) and the batch filename you are currently running.

The following example shows CLEANUP.BAT with placeholders.

```
CHDIR %1 \
DIR %1*.BAK
ERASE %1*.BAK
```

If CLEANUP.BAT is on drive A and you enter the command:

```
A>CLEANUP C:
```

CLEANUP.BAT substitutes C: for each %1 it finds. If your system has a drive C, FlexOS runs each command, displaying it and any output it produces on the screen. The following example runs CLEANUP.BAT with a parameter of C:.

```
B>CLEANUP C:

B>CHDIR C:

B: \

B>DIR C:*.BAK

 Volume in drive C: is HARDDISK
 Directory of C:

ONE       BAK        17    5-08-1985 14:04p
TWO       BAK        17    5-08-1985 14:04p

            2 Files        12345 bytes free

B>ERASE C:*.BAK

B>_
```

Notice that the ERASE command does not have output.

You can put several placeholders in your batch file. For example, suppose ERASEOLD.BAT contains these commands:

```
DEL %1
DIR %1
DEL %2
DIR %2
```

When you enter the command:

```
A>ERASEOLD B:*.BAK C:*.BAK
```

FlexOS substitutes B:*.BAK for %1 and C:*.BAK for %2.

Note: If you want to use the percent sign as part of a filename within a batch file, you must specify two percent signs. For example, to specify the file TAX%.EXE, you must enter it as TAX%%.EXE in the batch file.

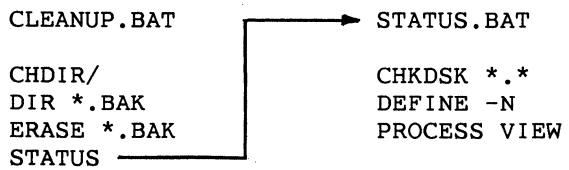
You can put placeholders in any order. You can invert the order of the commands in ERASEOLD.BAT and the batch file still works:

```
DEL %2
DIR %2
DEL %1
DIR %1
```

3.2 Chaining Batch Files

One of the commands you can include in a batch file is the command to run another batch file. This enables you to run two or more batch files consecutively by entering only one command line. To "chain" two batch files together, make the last command in the first batch file the command to run a second batch file.

The following example illustrates the principle of chaining batch files. Suppose there are two batch files that contain the commands as shown:



When you enter the command:

```
A>CLEANUP
```

FlexOS runs the commands in CLEANUP.BAT from top to bottom. When it reaches the command STATUS, control passes to STATUS.BAT. FlexOS then runs this file as if you had entered STATUS on the command line. When the commands in STATUS.BAT have finished, FlexOS returns you to your current directory.

If you place the command STATUS ahead of other commands in CLEANUP.BAT, control passes to STATUS.BAT, and the remaining commands in CLEANUP.BAT are ignored.

Note: If you include the command CLEANUP as the last command in STATUS.BAT, you create an "infinite loop." After CLEANUP.BAT runs its commands, control passes to STATUS.BAT which runs its commands, and then CLEANUP.BAT starts running again. The loop continues until you break out of it with Ctrl-C.

If you press Ctrl-C while running a batch file, FlexOS asks you:

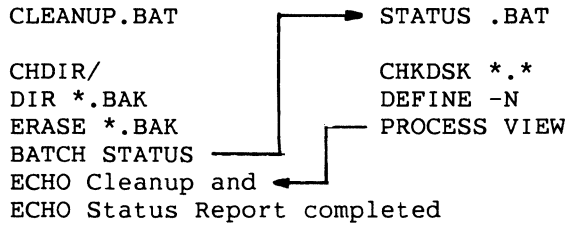
```
Terminate batch job (Y/N/A)?_
```

If you press Y, FlexOS ignores the remaining commands in the batch file. If you press N, only the currently running command ends; batch processing continues with the next command in the batch file. If you press A (for "All"), FlexOS stops the current batch file as well as any nested batch files.

3.3 Nesting Batch Files

Another way to run two or more batch files using one command line is to use a **nested** batch file. A nested batch file contains the command BATCH followed on the same line by the name of another batch file. Unlike a chained batch file, when the second batch file is finished running, control returns to the first batch file.

The following example illustrates the principle of nested batch files. Suppose there are two batch files that contain the commands as shown:



When you enter the command:

```
A>CLEANUP
```

FlexOS runs the commands in CLEANUP.BAT from top to bottom. When FlexOS reaches the command BATCH STATUS, control passes to STATUS.BAT. FlexOS runs STATUS.BAT as if you had entered STATUS on the command line. When STATUS.BAT is finished, FlexOS returns to CLEANUP.BAT and continues to run the commands there.

A batch file can contain more than one nested operation. The number of batch files you can nest is limited by the amount of memory available on your computer. If you run out of memory while running a batch file, processing stops and FlexOS displays an error message.

3.4 Commands for Batch Files

FlexOS has nine commands for batch processing:

BATCH	CLS	ECHO
FOR	GOTO	IF
PAUSE	REM	SHIFT

BATCH, GOTO, and SHIFT can only be used in a batch file. The other commands can also be entered on the command line, although they are more suited for batch use. A batch file can contain any other FlexOS command described in the User's Reference Guide. The following pages explain these commands in alphabetical order.

3.5 BATCH

Form BATCH filespec

Explanation

BATCH directs FlexOS to perform a nested operation. Program control flows from the batch file containing the BATCH filespec command to the named filespec.

filespec is the name of a batch file. If the batch file is not in the current directory, you must specify a directory path with filespec.

When the commands in the second filespec are finished, control returns to the first batch file and any additional commands are then run.

For examples of nested batch files, see "Nesting Batch Files" above.

3.6 CLS

Form CLS

Explanation

CLS clears the screen. Output returns to the screen when you run your next command, application, or batch file.

3.7 ECHO

Forms
ECHO
ECHO ON
ECHO OFF
ECHO message

Explanation

A batch file is a collection of command lines. The ECHO command determines whether these commands lines are displayed as the batch file runs. Any output and messages produced by a command are always displayed, whether ECHO is on or off.

Examples

```
A>ECHO

ECHO is ON
```

Displays whether ECHO is on or off.

The command ECHO ON displays the command line of each command as it runs.

When you use ECHO OFF, the command lines in the batch file are not displayed as the file runs. In addition, any remarks in the file made with the REM command are not displayed. Instead, the cursor sits suspended below the command line and no system prompt appears. FlexOS waits for you to enter another command.

The last form consists of the ECHO command followed by a message. This message is always displayed when you run your batch file, whether ECHO is on or off. The message is a string of ASCII characters up to 127 characters long. The following batch file, saved as ECHOTEST.BAT, illustrates this form:

```
ECHO OFF
TIME -D
ECHO Although ECHO is OFF, this message
ECHO is displayed.
ECHO ON
REM  ECHO is ON.  Now remarks are displayed
REM  as well.
```

When you run ECHOTEST, the following output is displayed:

```
A>ECHOTEST

A>ECHO OFF

Current time is 09:41:22.15

Although ECHO is OFF, this message
is displayed.

REM  ECHO is ON.  Now remarks are displayed
REM  as well.
```

3.8 FOR

Form FOR %c IN (item_1 ... item_n) DO command %c

Explanation

The FOR command lets you repeat another command a number of times for different files, commands, or applications. FOR can be used on the command line or in a batch file, but it is usually used in a batch file. Only one FOR command is allowed on a command line or on one line in a batch file.

The words FOR, IN, and DO are required. The **c** in the "%c" component can be any single character. You can specify two percent signs with the %c component if you wish.

item_1 ... **item_n** can be the names of files (including other batch files), commands, applications, or text. Separate items with one or more blank spaces. There is no limit to the length of your item list; however, the entire FOR command line must be 128 characters or fewer. You can use wildcards with the FOR command, but not path names.

The **command** at the end of the FOR command can be the name of a FlexOS command, an application, or a batch file. This command is run against each of the items in your list in left-to-right order.

Examples

```
FOR %v IN (FILE1 FILE2 FILE3) DO TYPE %v
```

Runs the TYPE command against FILE1, FILE2, and FILE3, in that order. This is the same as entering TYPE FILE1, TYPE FILE2, and TYPE FILE3 on successive command lines or on successive lines in your batch file.

Items in a FOR command line must be filenames.

3.9 GOTO

Form GOTO label . . .

:label

Explanation

The GOTO command transfers control to the line which has the label matching the one used on the GOTO line. GOTO lets you override the normal top-to-bottom running of commands. You can only use GOTO in a batch file.

Labels in a GOTO command appear in at least two places: after the word GOTO and again on a separate line anywhere else in your batch file. The label name next to GOTO is separated from GOTO by a blank space.

Either label name can be in uppercase or lowercase, or a combination of the two cases.

FlexOS treats the first eight characters in a label name as significant. FlexOS ignores any subsequent characters.

The second label name (the point to which control is passed) must meet three requirements:

- It must be on a line by itself. Any information after the label is treated as a comment.
- It must begin with a colon (:).
- The colon must be in column 1 of the new line.

If the label referenced by GOTO does not meet these conditions, the batch file ends with the message "Label not found."

If you use a label without a corresponding GOTO, then FlexOS treats the labeled line as a comment line.

Example

In the following batch file, GOTO2.BAT, the control passes from GOTO THREE to :THREE. FlexOS ignores the statements between the GOTO statement and the label.

```
REM  The name of this file is GOTO2.BAT.
REM  GOTO2.BAT exits normally but skips
REM  label :TWO.
REM
:ONE
REM  I'm in label ONE.
DIR A:
GOTO THREE
:TWO
REM  I'm in label TWO.
DIR B:
REM
A:CHKDSK
CLS
:THREE
REM  I'm in label THREE.
VER
```

3.10 IF

Forms
IF EXIST filespec command
IF NOT EXIST filespec command
IF string1==string2 command
IF NOT string1==string2 command
IF ERRORLEVEL decimal_number command
IF NOT ERRORLEVEL decimal_number command

Explanation

The IF command lets you test a condition to see if it is true or false. If the condition is true, FlexOS runs the command specified at the end of the IF statement. If the condition is false, FlexOS does not run the command.

The command on the IF/IF NOT line must be a FlexOS command or batch command.

A practical application of this command is to find out if a file is present or absent on disk before you take a certain action.

Examples

The form IF EXIST tests for the presence of a file in the current directory of the current or specified drive. In this case, the filespec is limited to a drive name, a filename, and a file extension; directory paths are prohibited. You can use wildcards in the filespec.

Here is how an IF EXIST statement might be used in a batch file:

```
IF EXIST %1.bak GOTO exit
REM  Else filename.BAK does not exist.
REM  We need to copy %1 to drive B
REM  as filename.BAK.
REM
DIR B:*.BAK
REM
REM  Observe the lack of filename.BAK
REM  on drive B.
REM
PAUSE
COPY %1 B:%1.BAK
DIR B:*.BAK
REM
REM  Confirm filename.BAK has been copied
REM  to drive B.
REM
PAUSE
GOTO BYPASS
:EXIT
ECHO filename.BAK already exists
DIR %1.BAK
REM  See?  I told you so.
:BYPASS
```

The form IF NOT EXIST inverts the logic of IF EXIST. Here is how the same backup task 1 might look using IF NOT EXIST:

```
IF NOT EXIST %1.BAK GOTO BAKUP
REM
:SHOWME
REM
REM filename.BAK exists; prove it
REM
DIR A:*.BAK
DIR B:*.BAK
PAUSE
CLS
GOTO EXIT
REM
:BAKUP
REM Copy %1 to drive B as
REM filename.BAK
REM
COPY %1 B:%1.BAK
GOTO SHOWME
REM
:EXIT
ECHO filename.BAK already exists
```

The preceding examples show that using IF or IF NOT is often a matter of style.

Forms 3 and 4 make sure two strings are identical before running the command. The strings can be any alphanumeric characters. If the two strings are identical, even to the matter of uppercase and lowercase letters, FlexOS runs the command at the end of the line.

For example, if a batch file called DISRAELI.BAT contains the following lines:

```
ECHO
IF %1==Gladstone ECHO %1 is a self-made man.
VOL
```

and if your command line looks like this:

```
DISRAELI Gladstone
```

FlexOS substitutes "Gladstone" for %1 in both places and tests to see if the parameter "Gladstone" matches the string on the right side of the double equal signs. If the test string in the batch file does match the string on the command line, FlexOS runs the ECHO command.

```
A>DISRAELI Gladstone

A>ECHO
ECHO is ON

A>ECHO Gladstone is a self-made man.
Gladstone is a self-made man.

A>VOL
Volume in drive A: is BATCHDISK

A>_
```

If the two strings are different, FlexOS does not run the ECHO command. FlexOS runs the VOL command whether the strings match or not.

Forms 5 and 6 test exit codes in a program that you create and run against a batch file. If the exit code decimal number in your program is equal to or greater than the decimal number in your batch file, FlexOS runs the specified command. Otherwise, that command is not run and control in your batch file passes to the line following IF ERRORLEVEL.

The ECHO command is often used with the IF and IF NOT statements to inform you of the exit conditions FlexOS encounters while running a program. Consider the following batch file, TESTPROG.BAT:

```
TESTHELP
REM
REM  TESTHELP is a program you are running
REM
:CHKERRS
        IF ERRORLEVEL 2 GOTO WRITERR
        IF ERRORLEVEL 1 GOTO READERR
        IF ERRORLEVEL 0 GOTO NORMAL
:WRITERR
        ECHO TESTHELP had a WRITE failure.
        GOTO REALEXIT
:READERR
        ECHO TESTHELP had a READ  failure.
        GOTO REALEXIT
:NORMAL
        ECHO TESTHELP finished normally.
        REM  It wasn't condition 2 or 1.
        GOTO REALEXIT
:REALEXIT
        REM  Th-Th-That's all, folks!
```

TESTPROG.BAT checks each IF ERRORLEVEL number value for a matching value in your running program. Of course, the messages you use with each ERRORLEVEL statement should reflect what is happening in the program.

3.11 PAUSE

Forms PAUSE
PAUSE comment

Explanation

PAUSE suspends batch processing so that you can evaluate the current situation and possibly take an appropriate action. For example, you might want to change diskettes between commands.

Insert PAUSE commands in your batch file where you want to end or continue batch processing. Each PAUSE command stops FlexOS and gives you time to decide whether to end processing. If you decide to end processing, press Ctrl-C. To continue processing, press any other key.

Examples

Form 1 of the PAUSE command suspends processing and displays the message, "Strike a key when ready...." If you press any key (except Ctrl-C), the batch file continues to run with the line following PAUSE. If you press Ctrl-C, batch processing ends. For example, suppose the file MAYBE1.BAT contains the following commands:

```
VER
DIR A:*.BAK
PAUSE
DIR A:*.TXT
PAUSE
COPY A:*.* B:
```

The two PAUSE commands let you examine your screen's contents before you proceed or stop.

Form 2 includes a comment with the PAUSE command. The comment is displayed when PAUSE suspends processing. The comment is a string of characters up to 121 characters long. For example, suppose MAYBE2.BAT contains the following commands:

```
VER
DIR A:*.BAK
PAUSE  The next command erases *.BAK on A!
DEL A:*.BAK
DIR A:*.TXT
PAUSE  The next command erases *.TXT on A!
DEL A:*.TXT
```

When you run MAYBE2.BAT, the comments on the PAUSE lines inform you of the consequences of continuing with the batch session. If you want all files with a BAK file extension erased when the first PAUSE occurs, press any key except Ctrl-C. If you decide to continue batch processing, you reach another decision point at the second PAUSE command. Press Ctrl-C to stop or any other key to continue, depending on whether you want to delete all the files on A with a file extension of TXT.

3.12 REM

Forms REM
REM remark

Explanation

The REM command lets you put remarks in your batch file. These remarks are displayed on the screen when FlexOS encounters them in running batch files. Processing is not interrupted by a REM command; the remark is merely displayed and processing continues on the line following the REM command.

REM lines are not displayed when ECHO is OFF.

Examples

Form 1 of the REM command inserts a blank line in your batch file. Careful use of blank lines makes batch files easy to read because of the visual (and hopefully logical) separation of one block of text from another. The following example uses the REM command to separate the COPY commands from the ERASE commands.

```
REM
COPY A:*.BAK C:
COPY B:*.BAK C:
REM
REM
ERASE A:*.BAK
ERASE B:*.BAK
REM
DIR C:*.BAK
```

Form 2 lets you put remarks in your batch file. A remark is a string up to 123 characters. Adding text to your REM line helps keep you informed of actions your batch file took or is about to take. For example, suppose ENDOFDAY.BAT contains the following commands:

```
REM     copy *.BAK files to drive C
COPY A:*.BAK C:
COPY B:*.BAK C:
REM
REM     Erase *.BAK files on A and B
REM
ERASE A:*.BAK
ERASE B:*.BAK
REM     Show current set of *.BAK files
REM     on drive C
DIR C:*.BAK
```

The meaning of each portion of text becomes clear. Using remarks in your batch files makes the files easier to read and to work with.

3.13 SHIFT

Form SHIFT

Explanation

The SHIFT command lets you use more than 10 parameters on one command line. SHIFT does this by shifting all the words on a command line one word to the left. SHIFT can only be used with batch files.

Remember that you can only put 10 placeholders in a batch file (%0 through %9). However, SHIFT lets you use different parameters for the same placeholders in a given batch file.

Because placeholders in a batch file depend on a parameter's position on a command line for placeholder substitution, SHIFT changes the relationship of the parameters to the batch file placeholders in the following way:

- The parameter leftmost on the line before SHIFT was run is no longer available to the running batch file after a SHIFT is run.
- The remaining command line parameters decrement their ordinal values by one, so that the 11th parameter becomes the 10th, the 10th becomes the 9th, and so on.

Compare the relationship of the command line parameters to the batch file placeholders in the following example:

```
A>SHIFT1 FIRST SECOND THIRD

SHIFT1.BAT:

ECHO %0 %1 %2
SHIFT
ECHO %0 %1 %2
SHIFT
ECHO %0 %1 %2
```

FIRST, SECOND, and THIRD are the first, second, and third parameters to be used with SHIFT1.BAT.

The batch file has three placeholders: %0, %1, and %2. Consequently, SHIFT1 substitutes for %0, FIRST for %1, SECOND for %2, and THIRD for %3 before any shifting occurs. When you run SHIFT1 using the following command line, this output is displayed on the screen:

```
A>SHIFT1 FIRST SECOND THIRD

A>ECHO SHIFT1 FIRST SECOND
SHIFT1 FIRST SECOND

A>ECHO FIRST SECOND THIRD
FIRST SECOND THIRD

A>ECHO SECOND THIRD
SECOND THIRD

A>ECHO THIRD
THIRD
```

Each leftmost parameter from the original command line disappears after each SHIFT execution. Notice also how the values for %0, %1, and %2 change after each SHIFT: FIRST substitutes for %0, SECOND for %1, and THIRD for %2 after the first SHIFT. Finally, notice that the parameter THIRD was not available to the running batch file until the parameter list had shrunk by one word -- that is, after the first SHIFT.

By structuring your parameters and synchronizing them with the placeholders in your batch file, SHIFT lets you use as many parameters as you can fit on a command line.

3.14 VERIFY

Forms VERIFY
VERIFY ON
VERIFY OFF

Explanation

The VERIFY command makes sure that data is correctly written to the disk and that the data can be read without error.

VERIFY does this whether it is on or off.

End of Section 3

USING THE PRINT SPOOLER WITH FLEXNET™

This chapter uses the following terminology: The **server** refers to the remote node with the printer you want to access. The **requester** is the node that originates the print jobs. A node can be both server and requester for different purposes depending on the system configuration.

The following files must be installed on each server node:

- SPLDRV.DRV
- SPOOL.286
- DESPOOL.286

Note: FlexNET must be installed on the system as described in the FlexNET User's Guide.

There are three ways to use the Print Spooler:

1. with the PRINT command
2. with the COPY or TYPE command
3. with your keyboard's PrtSc (Print Screen) key

The implications of printing jobs within a network using each of these methods are described below.

4.1 The PRINT Command

To use PRINT without entering a device name each time, you must define the background (BGPRN) printer to be the printer most often used. This is also the case if the printer is on a remote server node. If you replace the line:

```
define -s prn: = spldrv:
```

in CONGIG.BAT with

```
define -s prn: = server::spldrv:
```

you need not define a node or device when you enter a job for this printer. Replace the word "server" with the desired node name.

From a server node you can use the PRINT command to send a print job to another node by entering:

```
A>PRINT server::dev:filespec
```

where **server** is the destination node name and **dev** is the device name on that node.

4.2 The COPY or TYPE Command

If you use COPY or TYPE for printing, define PRN: in CONFIG.BAT as follows:

```
define -s prn: = server::spldrv:
```

Print jobs entered with COPY or TYPE are then queued on the default printer on the node specified as **server**. If no remote server is defined, you can still direct print jobs to remote nodes by entering the node designation with each command line. For example:

```
A>COPY filespec server::prn:

A>TYPE filespec > server::prn:
```

where the actual node name replaces **server**. Print jobs entered this way are printed on the default printer of the node specified.

4.3 The PrtSc Key

To use the PrtSc (Print Screen) key, PRN: must be defined in CONFIG.BAT as it is for COPY and TYPE. You must also have the Window Manager installed.

End of Section 4

Error Handling

FlexOS displays a variety of on-line error messages. Using the DEFINE command (described in the User's Reference Guide) you can set your own "help level," the amount of information displayed in the error message.

If you are a beginning user, you should set your error message help level to 4. If an error occurs, the level 4 message provides you with the most information about the problem. As you become more familiar with FlexOS, you will not need such detailed information to correct your errors and you can set your help level to a lower level.

This chapter discusses several classes of errors, their causes, and some recovery procedures. It has the following sections:

- Troubleshooting Checklist
- Disk Handling
- Corrupted Files
- Disk Error Messages
- Insufficient Memory
- Unresolvable Errors

5.1 Troubleshooting Checklist

This checklist covers common error conditions whose origin might be unknown to you:

- Are all electrical connections to and from all the peripheral devices connected properly to your computer and to electrical outlets?
- Is your hard-disk unit turned on? If you have a separate hard-disk unit, it must be turned on before you can access it.
- Have you used FORMAT and FDISK to prepare a FlexOS partition? If you have a hard disk as part of your system, you cannot access your hard-disk drive until you have prepared the hard disk with these commands. FORMAT is described in the User's Reference Guide; FDISK is described in the Configuration Guide.
- If a file is set to Read-Only, you can read the file but you cannot write to it. If you are the owner of the file, you can use the FSET command to give yourself the write privilege. If you are not the owner of the file, ask the owner to give you a copy of the file or have him modify the protection on the file so you can write to it.
- If the drive is set to Read-Only, you cannot write to files on that drive. This can occur when you exchange disks when files are still open on the disk you removed.

- If you press Ctrl-S or Scroll Lock, your keyboard is locked for that window until you press Ctrl-Q to unlock it.
- Did you specify a drive when entering the command line? If you did not specify a drive, FlexOS looks for a file only in the current directory of the current drive.
- Did you specify a valid drive? FlexOS drives can be identified by the letters A-P.
- Do you have sufficient privileges to work with the file or disk? FlexOS has three classes of user: owner, group, and world. Each of these classes can have different privileges. For example, the owner of the file may be able to delete the file, but group and world users may not. If you are unfamiliar with the FlexOS file and disk security, see the DISKSET and FSET commands in the User's Reference Guide.
- Are you running a program under the correct operating system version? Most FlexOS commands run only under the operating system with which they are shipped.

5.2 Disk Handling

Many errors arise from the handling of disks. Either the disk is incorrectly inserted or the disk itself is damaged. Use the following checklist to locate the error:

- Is the disk correctly inserted?
  - If there is no disk in the drive, you may receive an error. Make sure that you have a disk in the drive before you continue.
  - If the disk is not securely in place, or if the drive door is open, you may also receive an error. Make sure that the disk is correctly inserted and the drive door securely closed before you continue.
  - This class of errors can also occur when the disk is not in the correct drive. The FlexOS system prompt usually identifies the current drive.
- Is the disk the correct number of sides for the drive? Double-sided drives can read single- and double-sided disks. Single-sided drives can access only single-sided disks.
- Is the disk physically write-protected? 5 1/4-inch disks are write-protected in the absence of a notch on the upper-right disk edge. Check the documentation provided by the disk manufacturer for instructions on removing the write-protection on the disk.

- If the error is not caused by one of the above problems, the disk might be damaged. Use the COPY command to copy as much of the information on the disk as possible to a fresh disk. Replace any files you cannot copy with the files from your backup disk. If you have not maintained a backup disk, you must recreate the missing files.

You can use the CHKDSK command to verify the integrity of your disks. CHKDSK is described in the User's Reference Guide.

If these directions fail to correct an error condition, you probably are faced with a hardware error. As far as you are able, reconstruct the exact situation that produced the error message, then consult the dealer from whom you purchased your system.

5.3 Corrupted Files

Files can be corrupted by unforeseen events. Power outages are the most common cause. Replace the damaged file from your backup copy. If you do not have a backup copy, you will have to recreate the file.

If you are writing your own programs, it is possible to create one that writes over other files on the disk, or which corrupts the directory or other FlexOS system tracks. If this class of error occurs after you have run a program of your own, debug your program carefully before attempting to run it again.

5.4 Disk Error Messages

If you receive an error related to disks, such as "Disk not ready", go through "Troubleshooting Checklist" and "Disk Handling," above. Check for the obvious: open drive doors, hard-disk units not turned on, upside-down disks, write-protect tabs on disks. If, after running through the two checklists, you still cannot discover the error condition, you probably have a problem with the disk medium. Copy as many files as you can to a formatted and verified diskette.

5.5 Insufficient Memory

You might receive a message indicating that you have insufficient memory to load a program. Such an error might also occur when there is space in memory, but the available memory is fragmented.

If you encounter an insufficient memory message, you can stop programs running in different windows to create more free space in memory. The CANCEL command lets you terminate programs running in any window.

5.6 Unresolvable Errors

If you encounter an error condition that you are unable to correct, contact the dealer from whom you purchased your personal computer. Be prepared to provide the following information:

- The serial number of the operating system you are using. This number appears in the sign-on message when you start FlexOS.
- Your personal computer's configuration: the number of drives, quantity of memory, and additional equipment such as printers, modems, and so forth.
- How the error occurred. Indicate which programs were running at the time the error occurred. If possible, provide a disk with a copy of the program.
- Provide the text of the error message you received.

End of Section 5

Index

1073-2004-002