TOPS−20
User Utilities Guide

| Electronically Distributed

This manual describes utility programs available to both privileged and nonprivileged users of the TOPS−20 operating system.

Operating System:	TOPS−20 (KS/KL Model A) V4.1 TOPS−20 (KL Model B) V6.1

Software:

MAIL	Version	6
RDMAIL	Version	6
FILCOM	Version	22
CREF	Version	53B
MAKLIB	Version	2B
DUMPER	Version	5
PLEASE	Version	5

digital equipment corporation maynard, massachusetts

| TOPS−20 Update Tape No. 04, November 1990

First Printing, January 1980
Updated, January 1982
Updated, December 1982
Updated, September 1985
| Updated, November 1990

CONTENTS

FIGURES

TABLES

PREFACE

The TOPS−20 User Utilities Guide is intended for both the privileged and the nonprivileged user who needs information on utility programs that run on the TOPS−20 operating system. Before you use this manual, you should be familiar with the information contained in Getting Started with TOPS−20, the TOPS−20 User's Guide, and the TOPS−20 Commands Reference Manual.

This document provides detailed information on the following TOPS−20 utility programs: MAIL, RDMAIL, FILCOM, CREF, MAKLIB, DUMPER, and PLEASE. The manual contains tutorial and reference material in each chapter to accommodate both the novice and the experienced user.

The following conventions are used throughout the TOPS−20 User Utilities Guide:

<RET>	Indicates when you should press the RETURN key (on some terminals the key labeled CR)
<ESC>	Indicates when you should press the ESC key (on some terminals the key labeled ALT)
<DEL>	Indicates when you should press the DELETE key
<CTRL/x>	Indicates when you should hold down the CTRL key and at the same time type the letter x
file spec	Indicates a file specification
underlined text	Indicates anything you type or are expected to type on your terminal.

The current version of the following TOPS−20 documents are referenced in this manual:

Getting Started With TOPS−20

TOPS−20 User's Guide

TOPS−20 Commands Reference Manual

TOPS−20 Operator's Guide

TOPS−20 Monitor Calls Reference Manual

TOPS−20 LINK Reference Manual

TOPS−20 MACRO ASSEMBLER Reference Manual

TOPS−20 System Manager's Guide

TOPS−10/TOPS−20 Batch Reference Manual

CHAPTER 1

INTRODUCTION TO TOPS-20 USER UTILITIES

This manual describes utility programs available to any user of the TOPS-20 operating system.

The following utility programs are covered in this manual:

- The MAIL program, which allows you to send messages to other users of the system (Chapter 2)
- The RDMAIL program, which allows you to read messages sent to you via the MAIL program (Chapter 3)
- The FILCOM program, which allows you to compare two ASCII files or two binary files (Chapter 4)
- The CREF program, which produces cross-reference listings of symbols used in MACRO, FORTRAN, and ALGOL programs (Chapter 5)
- The MAKLIB program, which performs various functions on libraries of relocatable object modules (Chapter 6)
- The DUMPER program, which allows you to save files and directories on tape, and restore these files and directories to disk (Chapter 7)
- The PLEASE program, which allows you to communicate with the system operator (Chapter 8).

1.1 INVOKING THE UTILITIES

To invoke these utilities, you should be familiar with the TOPS-20 log-in procedure. Type the name of the program after the TOPS-20 prompt @ and press RETURN. The utility then prompts you for input. Thus, the general format is:


```
@Utility Name<RET>
Utility prompt
```

1.2 TYPING FILE SPECIFICATIONS

Many of the utilities accept file specifications as arguments. There are two forms of file specifications. The MAIL, RDMAIL, and DUMPER utilities accept file specifications in the following format:

```
dev:<dir>name.typ.gen;att...;att
```

where:

dev:	Indicates a device name, a file structure name, or a defined logical name
<dir>	Indicates a directory name
name	Indicates the filename of a particular file in the directory
.typ	Indicates a file type that helps identify the contents of the file
.gen	Indicates a generation number that shows the number of times a file has been changed
;att	Indicates a file attribute such as a file protection or an account string.

If you omit the dev: field of the file specification, the system assumes that you mean your connected structure. When you omit the <dir> field of the file specification, the system assumes that you mean your connected directory. When you omit the .gen field of the file specification, the system assumes that you mean the highest generation (largest generation number) for source files. For Destination files, the system assumes that you mean the highest generation plus one.

You can use recognition on file specifications in this format. You can use wildcards only in the DUMPER program. (Refer to Chapter 7.) For more information on file specifications, refer to the TOPS-20 User's Guide.

The FILCOM, CREF, and MAKLIB utilities accept file specifications in a slightly different format as follows:

```
dev:name.typ[PPN]
```

In this form of a file specification, filenames are restricted to six characters. File types are restricted to three characters. You cannot use recognition, and file generation numbers are not allowed. Therefore, the highest generation of a file is always used. You can use wildcards only in the MAKLIB program. (Refer to Chapter 6.)

The PPN is a project-programmer number, which you use instead of a directory name. To find out the PPN associated with a specific directory, give the TRANSLATE command. For example, if you wish to find out the PPN associated with the directory <ADLEY> on PS: you do the following:

```
@TRANSLATE <DIRECTORY> PS:<ADLEY><RET>
PS:<ADLEY> <IS> PS:[4,305]
@
```

You can avoid using PPN's in file specifications by defining a logical name that represents the directory you wish to access. Do the following procedure:

1. Give the DEFINE command to define a logical name as the directory.

2. Use the logical name in place of the device name and the PPN when you type the file specification.

The following is an example of defining a logical name for a directory and using it with the FILCOM program:

```
@DEFINE (LOGICAL NAME) ADL: (AS) <ADLEY><RET>
@FILCOM<RET>
*TTY:=ADL:TEST.MAC,ADL:TEST2.MAC<RET>
```

You can also define logical names to reference long filenames or particular file generation numbers. This is especially useful with the FILCOM program when you wish to compare two similar files with different generation numbers. For example:

```
@DEFINE (LOGICAL NAME) A: (AS) FOO.BAR.3<RET>
@DEFINE (LOGICAL NAME) B: (AS) FOO.BAR.4<RET>
@FILCOM<RET>
*TTY:=A:,B:<RET>
```

CHAPTER 2

THE MAIL PROGRAM

2.1 INTRODUCTION

You can use the MAIL program to send messages to other users of the system. You can send mail to a single user or to a group of users who are either logged in or not logged in.

2.2 RUNNING MAIL

To run MAIL, type MAIL after the TOPS-20 prompt @ and press the RETURN key. The program responds with the To: prompt as follows:

```
@MAIL<RET>
To:
```

Type the name of the user to whom you are sending the message, and press RETURN.

If you are sending a message to a group of users, type the names, separating them with commas, and press RETURN. For example:

```
To: Adley,Sartini,McElmoyle<RET>
```

The program then prompts:

```
CC:
```

Now list any secondary recipients of your message. Type the name or names (separated by commas) and press RETURN. If you do not want to send a copy to others, simply press RETURN after the CC: prompt.

If you type an invalid (nonexistent) user name, the program responds with:

```
?Invalid user name
```


MAIL returns with either the To: prompt or the CC: prompt. Type CTRL/H after either prompt. This retrieves only the names up to the error, and you can type any additional valid names.

You cannot send more than one copy of a mail message to a user. If you type a user name more than once after either the To: or the CC: prompt, the program prints a warning message. For example:

```
To: Adley<RET>
CC: Adley<RET>
%Duplicate name purged - ADLEY
```

The MAIL program continues after it prints the warning message; however, the program removes the duplicate name from the list of users.

The program then prompts with:

```
Subject:
```

Type a description of the message and press RETURN. For example:

```
Subject: Location of weekly writers meeting<RET>
```

If your description exceeds one line, you cannot continue the description on a second line; you must continue typing when you reach the end of a line. The system automatically continues your description on the second line by responding with a carriage return line feed sequence. When you have completed typing your description, press RETURN. For example:

```
Subject: Location of weekly meeting and change in software
release date.<RET>
```

NOTE

The system may interpret a character in the Subject: line (such as a question mark) as a special character. To avoid this, precede the character with a CTRL/V.

MAIL then prompts with:

```
Message (Terminate with ESC or CTRL/Z):
```

and waits for you to enter your message. Once you have terminated your message by typing ESC or CTRL/Z, the program informs you that it has processed your message:

```
Processing mail...


No errors.
```

```
-DONE-
```

and returns you to TOPS-20 command level.

If you send a message to a user who is logged in and accepting links and system messages, that user is informed immediately as follows:

```
[You have a message from SENDER]
```

If you send a message to a user who is not logged in, that user is informed the next time he logs in:

```
@LOGIN (USER) ADLEY (PASSWORD) (ACCOUNT) 341<RET>
 Job 54 on TTY33 23-Apr-79 09:46:05
 You have a message
@
```

If you make an error in sending mail to a user, you receive one of the following messages:

```
[USER NAME] not sent BECAUSE:
Invalid directory number
```

or

```
Invalid simultaneous access
```

or

```
No such file type (or some other reason related to why the
recipient's MAIL.TXT file could not be found)
```

or

```
[USER NAME] not sent BECAUSE:
Disk quota exceeded
```

NOTE

For additional information on these error messages, refer to Section 2.4, MAIL Messages.

You can use a recovery procedure to resend mail after receiving some of the MAIL error messages. This recovery is particularly helpful when your message is long and you do not want to retype it. The procedure is as follows:

1. Undelete the MAIL.CPY file in your logged-in directory. This file contains the message that could not be sent.

2. Rename the MAIL.CPY file; for example, ERROR.TXT. This prevents MAIL from deleting the file a second time during message processing.

3. After the TOPS-20 prompt @, type:

   ```
   @GET SYS:MAIL<RET>
   ```

4. The system gives the TOPS-20 prompt once again, and you type:

   ```
   @REENTER<RET>
   ```

5. After you press RETURN, the system prompts:

   ```
   File name of message file:
   ```

   Now type the new file spec of the renamed MAIL.CPY file, and press RETURN:

   ```
   File name of message file:  (file spec)<RET>
   ```

   The MAIL program now proceeds as though you had just typed ESC or CTRL/Z after the message.

2.3 OPTIONS TO THE MAIL PROCEDURE

The procedure in Section 2.2 describes the most common use of MAIL. Options to this basic procedure are as follows:

1. You can use the TALK command as an alternative to the MAIL program to communicate with a user who is logged in. (For more information on the TALK command, refer to the TOPS-20 Commands Reference Manual.)

2. You can use the INFORMATION MAIL command to check on the status of new mail, either your own or that of other users. (For more information on this command, see the TOPS-20 Commands Reference Manual.)

3. If you send mail often to a group of users, you can create a file containing these names. Then, instead of typing all the names each time you send a message, you can type the filename, preceded by an @, after the To: prompt or the CC: prompt. For example, if the file NAME.FIL.1 contains the user names ADLEY, CRUGNOLA, LYONS, type:

   ```
   To: @NAME.FIL.1<RET>
   ```

   The filename can also be combined with other user names following the prompt. However, the file must follow the list of additional user names. For example:

```
To: Sartini,McElmoyle,@NAME.FIL.1<RET>
```

4. You can use the contents of an indirect file as your message or Subject: line text. The indirect file you use in the Subject: line can contain only one line. You cannot type any additional text on this line with the indirect file. To send the contents of an indirect file as mail, type an @ followed by the name of the file, and press RETURN. You cannot type any additional text before or after the indirect file. For example:

```
@MAIL<RET>
To: Crugnola<RET>
CC:<RET>
Subject: Macro files<RET>
Message (Terminate with ESC or CTRL/Z):

@MACRO.CMD.1<RET>

Processing mail...

No errors.
-DONE-
@
```

In this case, you do not terminate the message with ESC or CTRL/Z, because you are using an indirect file as your message. However, you terminate the file spec by pressing RETURN. If you terminate your message with CTRL/Z, MAIL responds with:

```
@MACRO.CMD.1 (CTRL/Z)
?Not confirmed
```

To recover, type CTRL/H immediately to retrieve the indirect file spec. Then, press RETURN.

5. Normally, you send mail to other users. However, you can also send mail to any non-files-only directory on PS:. The most common non-files-only directories are PS:<REMARKS> and PS:<SYSTEM>.

PS:<REMARKS> can be used for recording information and problems that the system staff should be aware of. For example, use PS:<REMARKS> to record any system difficulties, hardware/software problems, or other related items. To send a message to this directory, type REMARKS after the To: prompt. For example:


```
@MAIL<RET>
To: REMARKS<RET>
CC:<RET>
Subject: Supplies<RET>
Message (Terminate with ESC or CTRL/Z):

Terminals in Room 216 need additional boxes of paper, size
9 7/8 x 11.  <ESC>

Processing mail...


No errors.
-DONE-
@
```

Generally, to send messages to all users of the system, you enable your WHEEL or OPERATOR capabilities and run MAIL. These messages are called Messages-of-the-Day. You can also send Messages-of-the-Day by connecting to the directory PS:<SYSTEM>. However, most systems are not set up to allow users to connect to this directory. The following example shows an enabled user running MAIL to send a Message-of-the-Day.

```
$MAIL<RET>
To: SYSTEM<RET>
CC:<RET>
Subject: System shut-down<RET>
Message (Terminate with ESC or CTRL/Z):

The system will be shut down tomorrow at 5 p.m. for
preventive maintenance.  <ESC>

Processing mail...


No errors.
-DONE-
$
```

If you attempt to send mail to PS:<SYSTEM> and do not enable WHEEL or OPERATOR capabilities, MAIL prints the following error message:

```
Processing mail...SYSTEM not sent BECAUSE:
WHEEL or OPERATOR capability required
```

To resend the mail, you can follow the recovery procedure described in Section 2.2 after you enable WHEEL or OPERATOR capabilities.

When you send a message to PS:<SYSTEM>, the following message appears on all terminals that are receiving system messages:

[New Message-of-the-Day available]

Users not logged in to the system at the time you send the message automatically receive new Messages-of-the-Day the next time they log in.

6. You can use MAIL to inform yourself that your batch job is completed. Place commands to MAIL in your control file as shown in the following example. Note that a period is used as the reply to the To: prompt. This character replaces a user name and informs MAIL that the message is to be sent to you. For example:

```
@CREATE (FILE) TEST.CTL<RET>
INPUT: TEST.CTL<RET>
00100  @FILCOM<RET>
00200  *TEST.FOR=DIFFER.FOR,ADDEM.FOR/A<RET>
00300  @PRINT TEST.FOR<RET>
00400  @MAIL<RET>
00500  *.<RET>
00600  *<RET>
00700  *BATCH JOB IS DONE<RET>
00800  *^Z
00900  <ESC>
*E
@
```

NOTE

Use of a period in place of your user name when you run MAIL is not a feature unique to batch. You can use it any time in the MAIL program when you wish to specify yourself as a recipient of mail.

2.4 MAIL MESSAGES

The most common MAIL messages, their descriptions, and suggested user responses follow. Fatal errors are preceded by a question mark (?). Warning messages are preceded by a percent sign (%).

%Duplicate name purged - [USER NAME]

Description: You attempted to send a user more than one copy of a mail message.

Suggested User Response: None. MAIL continues automatically, and eliminates the duplicate name.

?Invalid user name

Description: You typed an invalid (nonexistent) user name as a recipient of your message.

Suggested User Response: Type CTRL/H after either the To: prompt or the CC: prompt to retrieve only the names up to the error. Type any additional valid names.

?MAIL.CPY Failure
Entire file structure full

Description: The public structure (PS:) is full, and therefore MAIL cannot operate. You receive this message immediately after invoking the program.

Suggested User Response: Delete and expunge some files from your logged-in directory, or wait until some space is freed.

?MAILER is not running. Messages not sent.

Description: Your message was not sent because the MAILER program is not functioning.

Suggested User Response: Send a message to the operator with the PLEASE program (refer to Chapter 8) to report that MAILER is not functioning. Then, once MAILER is functioning, use the recovery procedure described in Section 2.2 to resend your message.

?Not confirmed

Description: You did not press RETURN immediately after typing an indirect file spec.

Suggested User Response: TYPE CTRL/H immediately after the error message to retrieve the indirect file spec. Then, press RETURN to confirm the indirect file spec.

?Processing errors occurred. No mail sent.

Description: There is a problem with you MAIL.CPY file; either MAILER cannot find it, or the file is not in correct format.

Suggested User Response: Check to see if there is another program updating MAIL.CPY. If not, contact your Software Specialist or send a Software Performance Report (SPR) to DIGITAL.

SYSTEM not sent BECAUSE:
WHEEL or OPERATOR capability required

Description: You attempted to send mail to PS:<SYSTEM> and did not enable WHEEL or OPERATOR capabilities.

Suggested User Response: You must enable WHEEL or OPERATOR capabilities before sending mail to PS:<SYSTEM>. To resend the mail, you can follow the recovery procedure described in Section 2.2 after you enable WHEEL or OPERATOR capabilities.

%Too many user names. 100 is maximum.

Description: You specified too many users as recipients of your message. MAIL only allows up to 100 user names as recipients of a message; it sends your message to the first 100 names but ignores all that exceed the first 100.

Suggested User Response: Send your message to the first 100 names. Next, edit your MAIL.CPY file to retrieve the message text. Then, run MAIL to send the message to the additional names, using "@" to send the edited MAIL.CPY file as the message text.

[USER NAME] not sent BECAUSE:
Invalid directory number

Description: You attempted to send mail to a nonexistent user directory.

Suggested User Response: You cannot send mail to a user who does not have a directory.

[USER NAME] not sent BECAUSE:
Invalid simultaneous access

Description: Another user has the receiver's MAIL.TXT file open for writing.

Suggested User Response: Follow the recovery procedure described in Section 2.2 to resend the message.

[USER NAME] not sent BECAUSE:
No such file type (or some other related reason)

Description: The intended receiver of your message has no MAIL.TXT file.

Suggested User Response: Ask the user to create a MAIL.TXT file to receive mail messages.

[USER NAME] not sent BECAUSE:
Disk quota exceeded

Description: The receiver's directory exceeds its working quota.

Suggested User Response: Some files must be deleted from the directory before mail can be received. You can also enable WHEEL or OPERATOR capabilities to ignore the user's quota. If the user is logged in, you can use the TALK command as an alternative.

2.5 TECHNICAL NOTES

MAIL works with another program called MAILER when it handles messages. When you type a mail message, MAIL creates a file, MAIL.CPY, in your logged-in directory. This file is closed when you complete your message input. At this point, MAIL sends an IPCF (Inter-Process Communication Facility) packet to the MAILER program to inform it that you want to send a message. (For more information on IPCF, refer to the TOPS-20 Monitor Calls Reference Manual.) MAILER processes the message by appending the contents of MAIL.CPY in your logged-in directory to the file MAIL.TXT in the recipient's logged-in directory. Then it sends an IPCF packet back to MAIL, which notifies you of the status of your message (sent or not sent). At this point, the MAIL.CPY file is deleted from your logged-in directory.

CHAPTER 3

THE RDMAIL PROGRAM

3.1 INTRODUCTION

The RDMAIL program prints messages that have been sent to you by other users of the system through the MAIL program. Your MAIL.TXT file in your logged-in directory on PS: contains these messages.

NOTE

If your MAIL.TXT file contains mail sent by Version 4 of the MAIL program, you must use Version 4 of RDMAIL to read it.

There are various ways that the system notifies you whenever there is mail that you have not read. If another user sends you mail while you are not logged in, you receive the following message the next time you log in:

```
@LOGIN (USER) DBELL (PASSWORD) (ACCOUNT) 341<RET>
 Job 35 on TTY42 29-Aug-79 16:14:12
 You have a message
@
```

Another user may send you a message while you are logged in. In this case, the system types

[You have a message from SENDER]

on your terminal.

Messages-of-the-Day sent to you when you are not logged in are printed on your terminal automatically after you log in. However, if your directory is set to REPEAT LOGIN-MESSAGES, you receive all Messages-of-the-Day every time you log in. For more information on the REPEAT LOGIN-MESSAGES subcommand, refer to the BUILD command description in the TOPS-20 Commands Reference Manual. If a system

message is sent while you are logged in and you are receiving system messages, you are notified immediately:

[New Message-of-the-Day available]

You can give the SET MAIL-WATCH command to keep informed of any new mail you receive, especially if you have given the REFUSE SYSTEM-MESSAGES command. (For more information on these two commands, refer to the TOPS-20 Commands Reference Manual.) You can add the SET MAIL-WATCH command to your COMND.CMD file, if you have one, or type it each time you log in. When you give this command, it tells the system to notify you when you have new mail. You receive this notification only when you are at TOPS-20 command level. At intervals of approximately five minutes, the TOPS-20 Command Processor informs you that you have new mail whenever it prompts you for a new command. This message appears on your terminal:

[You have new mail]

You may give the INFORMATION MAIL command, even if you are not logged in, to check on the status of new mail for yourself or other users. To do this, type the following:

@INFORMATION (ABOUT) MAIL (FOR USER) name<RET>

The system returns with one of the following responses:

New mail exists

or

No new mail exists

or

Mailbox protected

3.2 RUNNING RDMAIL

To start RDMAIL, type RDMAIL after the TOPS-20 prompt @ and press the RETURN key. The program responds with the date and time prompt as follows:

```
@RDMAIL<RET>
Date and time (/HELP for help)
```

If you have enabled WHEEL or OPERATOR capabilities, RDMAIL first asks you whether you want to read your own mail or that of another user. For example:

```
$RDMAIL<RET>
Special user (y or n)?
```

If you type y, you are indicating that you wish to read another user's mail. Type y and press RETURN. RDMAIL then prompts you to type the name of the user whose mail you wish to read. Type the user name and press RETURN. RDMAIL then prompts you for date and time input. For example:

```
$RDMAIL<RET>
Special user (y or n)?  y<RET>
User name:  DNeff<RET>
Date and time (/HELP for help)
```

If you type n, you are indicating that you wish to read your own mail. Type n and press RETURN. RDMAIL then prompts you for date and time input. For example:

```
$RDMAIL<RET>
Special user (y or n)?  n<RET>
Date and time (/HELP for help)
```

RDMAIL allows you to read your messages several ways:

- By giving a date and/or time
- By giving a program switch or combination of switches
- By giving a date and/or time combined with one or more program switches.

To read any new messages, simply press RETURN.

You can define a time period of mail you wish to read. To do this, type a date and/or a time. A common TOPS-20 format is:

```
MMM DD,YYYY HH:MM:SS
```

For example, a valid date and time is May 22,1979 17:00:00. If you type only a date, RDMAIL assumes the time 00:01:00. If you type only a time, the program assumes the present date. RDMAIL responds by displaying all messages you received on and after the date and/or time you typed.

If you type an invalid date or time, you receive an error message. Three of the most common ones are:

```
?Invalid date format

or

?Invalid time format
```



```
or

?Day of month too large
```

After the error message RDMAIL returns with the prompt for you to type a valid date and/or time.

Table 3-1 describes the RDMAIL switches you can use after the prompt, either alone or combined with date/time input.

Table 3-1: RDMAIL Switches

Switch	Function
/ALL	Types all messages, regardless of date.
/HELP	Types the program help text, outlining the time/date format and program switches and their functions.
/LIST	Outputs messages to the line printer, rather than to your terminal.
/MESSAGE-OF-THE-DAY	Types messages from the system Message-of-the-Day file, rather than from your own message file.
/PERUSE	Allows you to peruse messages, and gives only the following information for each message: Date: From: To: CC: and Subject.
/STOP	Instructs RDMAIL to stop after each message it types. At the end of each message, the system prompts you to type RETURN for more output.

3.2.1 Reading Mail Using RDMAIL Switches

You type RDMAIL switches immediately following the prompt, and may or may not combine them with a date and/or time. You have the options of combining switches and preceding each switch with a space. To use RDMAIL switches, type a slash (/) followed by the switch name.

/HELP − HELP Switch

Type /HELP to get information on running RDMAIL. For example:

```
@RDMAIL<RET>
Date and time (/HELP for help) /HELP<RET>
```

After the help text prints on your terminal, the system returns with the prompt for you to type date/time information and/or another switch.

NOTE

HELP overrides all other switches that you may combine with it. The system ignores all other specified switches in the combination, and prints the full RDMAIL help text.

/ALL − ALL Switch

Type /ALL when you wish to read all messages in the mail file, regardless of date.

/ALL may be combined with all other program switches except /HELP. If you type /ALL after the prompt,

```
@RDMAIL<RET>
Date and time   (/HELP for help) /ALLRET>
```

RDMAIL accesses all messages in your file.

/LIST − LIST Switch

Type /LIST when you want messages output to the line printer rather than to your terminal. /LIST can be combined with date/time input, and/or with /ALL. For example,

```
@RDMAIL<RET>
Date and time  (/HELP for help) May 13, 1979 12:00:00 /LIST<RET>
```

Prints all messages in your file on and after 12:00:00 of May 13, 1979 on the line printer.

If you type only /ALL /LIST after the prompt, RDMAIL prints all messages in your file on the line printer.

/MESSAGE−OF−THE−DAY − System Message Switch

Type /MESSAGE−OF−THE−DAY to print mail from the system Message−of−the−Day file (PS:<SYSTEM>MAIL.TXT), rather than from your own message file. Since new entries in the Message−of−the−Day file are typed on your terminal when you log in, you normally use this switch when a new Message−of−the−Day becomes available while you are logged in. /MESSAGE−OF−THE−DAY

may be combined with date/time input. It may also be combined with all other program switches except /HELP. For example:

```
[New Message−of−the−Day available]

@RDMAIL<RET>
Date and time  (/HELP for help) /MESSAGE−OF−THE−DAY<RET>

  ________
Date:  25 Jun 79 0853−EDT
From:  OPERATOR
To:  SYSTEM
Subject:  STAND−ALONE AT NOON

SYSTEM IS GOING DOWN AT NOON FOR NEW MONITOR TO BE LOADED.

  ========
@
```

In this case, the system prints the Messages−of−the−Day since June 25, 1979 on your terminal.

When you type /MESSAGE−OF−THE−DAY after the prompt, RDMAIL outputs all new Messages−of−the−Day since you last logged in, whether or not you have read them.

/PERUSE − PERUSE Switch

Type /PERUSE when you want to peruse messages in your file. Only the following lines for each message are printed: Date:, From:, To:, CC:, and Subject: /PERUSE can be combined with date/time input and all other program switches except /HELP. A sample output of /PERUSE is as follows:

```
@RDMAIL<RET>
Date and time   (/HELP for help) Apr 12,1979 /PERUSE<RET>
  ________
Date:  12 Apr 79 0900−EDT
From:  OSMAN
To:  ADLEY
  ________
Subject:  your files

  ________
Date:  12 Apr 79 1452−EDT
From:  HARAMUNDANIS
To:  PORADA
CC:  ADLEY,HARAMUNDANIS
  ________
Subject:  DUMPER
```

```
  --------
Date:  17 Apr 79 1451-EDT
From:  LYONS
To:  ADLEY
  --------
Subject:  Re:  TEST OF MAIL PROGRAM
```

The system continues to output messages from April 12, 1979 to the present date.

/STOP - STOP Switch

Type /STOP to cause RDMAIL to stop after each message that it types. Following each message, the program prompts you to press RETURN for more output. /STOP can be combined with date/time input and all other program switches except /HELP and /LIST. A sample output of /STOP and /PERUSE is as follows:

```
@RDMAIL<RET>
Date and time   (/HELP for help) May 1,1979 /STOP /PERUSE<RET>
  --------
Date:  1 May 79 1335-EDT
From:  OSMAN
To:  ADLEY
  --------
Subject:  MAILER

[Type <CR> for more]<RET>
  --------
Date:  1 May 79 1844-EDT
From:  LYONS
To:  ADLEY
  --------
Subject:  Your account on System 2116

[Type <CR> for more]<RET>
```

The system continues to output messages from May 1, 1979 to the present date, allows you to peruse them, and stops after each message.

If you type an invalid switch in RDMAIL, you receive the following message:

```
?Does not match switch or keyword
```

The system returns with the prompt for you to type a valid switch.

3.3 RDMAIL MESSAGES

The most common RDMAIL messages, their descriptions, and suggested user responses follow. Fatal errors are preceded by a question mark (?). A warning message is preceded by a percent sign (%).

?Day of month too large

Description: You typed an invalid day of the month after the program prompt, Date and time (/HELP for help).

Suggested User Response: Enter a valid day of month when the program returns with the prompt following the error message.

?Does not match switch or keyword

Description: You typed an invalid switch after the program prompt, Date and time (/HELP for help).

Suggested User Response: Type one or more valid RDMAIL switches.

%MAIL.TXT File contains updated entries or improper format

Description: Your MAIL.TXT file is in the wrong format, possibly from being edited with a text editor.

Suggested User Response: Copy the file so you have a record of any current messages. Then, delete the file.

?Invalid date format

Description: You made an error in the date format following the program prompt, Date and time (/HELP for help). This error differs from an invalid day of the month; the error might be, for example, misspelling of the month or an incorrect year.

Suggested User Response: Type a valid date format when the program returns with the prompt after the error message.

?Invalid time format

Description: You typed an invalid time after the program prompt, Date and time (/HELP for help). For example, a possible error is a nonexistent time such as 26:00:00.

Suggested User Response: Type a valid time when the program returns with the prompt after the error message.

?LPT: not available for output

Description: This message appears in response to /LIST. Either you defined a logical name for LPT: that points to an unavailable device. RDMAIL ignores /LIST, and prints the mail on your terminal.

Suggested User Response: Wait for the line printer to become available, or redefine the logical name to point to an available device (such as DSK:).

CHAPTER 4

THE FILCOM PROGRAM

4.1 INTRODUCTION

The FILCOM program compares two files and prints any differences between them. With FILCOM, you can compare either ASCII files (text files and source programs) or binary files (relocatable binary files and save files). The comparison is line by line for ASCII files, and word by word for binary files.

4.2 RUNNING FILCOM

To run FILCOM, type FILCOM, and press the RETURN key. The program prompts you for input with an asterisk:

```
@FILCOM<RET>

*
```

After the prompt, enter a FILCOM command string in the following format:

```
Destination file spec=Source file spec1,Source file spec2/Switches
```

where:

Destination file spec is the output file that contains the differences between the two Source files.

If you do not specify a Destination filename, FILCOM uses the name of the file in Source file spec2. If you omit the name in Source file spec2, the program uses the filename from Source file spec 1. If there is no filename in Source file spec 1, then the filename FILCOM is used. The default for the Destination file type is .SCM for ASCII files and .BCM for binary files. If you completely omit the Destination file spec, FILCOM writes the output to the device TTY:.

Source file spec1 is the first input file you wish to compare.

You must completely specify this file spec in the command string.

Source file spec2 is the second input file you wish to compare.

If you omit the filename in Source file spec2, FILCOM uses the filename in Source file spec1. If you omit the file type in Source file spec2, FILCOM uses the file type in Source file spec1. To indicate a null file type, simply type a period (.) at the end of the filename in either Source file spec1 or Source file spec2.

NOTE

FILCOM does not accept file generation numbers. You can compare two files with the same name and type but different generation numbers (for example, FOO.BAR.1 and FOO.BAR.2) by defining logical names for these files. For more information on defining logical names, refer to the TOPS-20 User's Guide.

FILCOM accepts six characters for a name and three characters for type. If more than nine characters are used, FILCOM truncates to nine characters.

You can enter switches after the two input file specs. These switches tell FILCOM how to compare the specified files. However, you don't always need to give switches, because FILCOM often determines the type of comparison by the file types. If either of the input files is of a type listed in Table 4-1, the files are compared in binary mode; otherwise they are compared in ASCII mode.

Table 4-1: Special File Types Recognized by FILCOM

File type	Extension		
Binary Files:	.APL	.DMP	.RIM
	.ATR	.MSB	.RMT
	.BAC	.OVL	.RTB
	.BIN	.OVR	.SCH
	.BUG	.QUC	.SFD
	.CAL	.QUD	.SYM
	.CHN	.QUE	.SYS
	.DAE	.QUF	.UFD
	.DBS	.REL	.UNV
	.DCR		.XPN
Sharable Save Files:	.EXE		

Nonsharable Save Files:	.LOW	.SAV	.SVE
Offset Address Files; word 0 of the file treated as if it was word 400000.	.HGH	.SHR	

NOTE

If FILCOM cannot determine the mode for comparison from the input file type or switches, it compares the files in ASCII mode.

For more information on sharable and nonsharable save files and the control words used in them, refer to the TOPS-20 Monitor Calls Reference Manual.

After you enter the command string specifying the mode for comparison, the two input files, and any necessary switches, press RETURN. When FILCOM has finished the comparison, it notifies you of the status:

```
     %files are different
or
     No differences encountered
```

The program then prints a second asterisk for you to enter another command string. For example:

```
@FILCOM<RET>

*COMPAR.FIL=EXFILE.1,EXFILE.2<RET>
%files are different


*
```

If you wish to stop the program, type CTRL/C to return to TOPS-20 command level.

4.2.1 Comparing ASCII Files

In ASCII mode, FILCOM compares the characters in each line of the two files, always ignoring nulls. Comments and spacing can be selectively ignored, based on the switches you type.

FILCOM contains the following switches that you use in the command string to compare ASCII files.

/A Instructs FILCOM to compare the two input files in ASCII mode. It treats both files as if they contain ASCII characters, searches the files for text differences, and ignores similar lines. /A is useful if the input files are ASCII files but have one of the file types listed in Table 4-1.

/B Considers blank lines in the comparison. If you do not specify /B in the command string, FILCOM normally ignores blank lines in the two files.

/C Instructs FILCOM to ignore comments and spacing in the files. Comments are defined as text on a line following a semicolon. Spacing is defined as spaces and tabs. FILCOM normally considers comments and spaces in the comparison. This switch is useful when you compare assembly language source files (MACRO Assembler source files).

/H Prints a FILCOM help text, which contains a description of the program and all switches, on your terminal. You can type /H by itself immediately after the FILCOM prompt, or in a command string.

NOTE

/H overrides all other switches that you may combine with it. The system ignores all other specified switches in the combination, and prints the full FILCOM help text.

/nL Defines the number of lines that end a match. When FILCOM finds that number of successive lines that are the same in both input files, it prints all differences found up to the time of the match. The FILCOM output includes the first line of the match for easy reference. FILCOM normally uses the value "3" for the number of lines (the value of n).

/O Instructs FILCOM to include a label and offset in the differences listing for ASCII files. There are three types of messages:

[;At top of file + nL] where nL is a decimal number representing the number of lines between the top of the file and the line where the difference occurs. If a difference occurs at the top of the file, nL is not listed.

[;At Label + nL] where Label is the MACRO-style label immediately preceding the difference and nL represents the decimal number of lines away from the label that the difference occurs. If the difference occurs at the label, nL is not listed.

[;At Label + nL following label name] for PDP-11 files, where label is the local label name in the form nn$, nL represents the decimal number of lines from the local label that the difference occurs and label name is the name of the last preceding block label. The block label name is listed as further help in locating the difference, since local label names are not always unique. If the difference occurs at the label, nL is not listed. If the difference occurs before FILCOM sees a label, the difference is listed as [;At label + nL] where label is the block label. The label name for all labels must be in the first ten characters of the line. Labels refer to file 1, not file 2.

/Q Instructs FILCOM to print only the status of the comparison (either ?files are different or No differences encountered). FILCOM does not enumerate the differences between the files. It stops reading the files after it discovers the first difference.

/S Ignores spaces and tabs in the comparison of two ASCII files. FILCOM normally considers spaces and tabs in the comparison.

/T Instructs FILCOM to generate an output file even if no differences are found. If the /T switch is omitted and there are no differences in the files, no output file is generated.

/U Compares your two input files in update mode. This means that FILCOM creates an output file, which is the second input file, with change bars in the left margin next to the lines that differ from those in the first input file. Deleted lines are indicated by a change bar on the next common line. /U is helpful when you are comparing two versions of text. To obtain a meaningful comparison, type the latest version of the file as the second input file in the command string. It is recommended that /nL be used with the /U switch.

The output file in ASCII mode comparison includes a header for each input file that contains the following information:

- the file number
- the file spec
- the date and time the input file was created.

The following is an example of a header that would appear in an ASCII output file:

```
File 1) DSK:FORLIB.TXT[4,244]   created: 1052 26-JUL-1979
```



```
File 2) DSK:FORTEX.TXT[4,244]   created: 1155 26-JUL-1979
```

NOTE

If you use /U in the FILCOM command string (compare in update mode), this header does not appear.

Each time FILCOM finds differences between two ASCII input files, it outputs a number corresponding to a page number in the first file, and the differences. At the end of the list of differences, the program prints a common line between the two files. The program then prints four asterisks, a number corresponding to the second input file, and the differences. Then FILCOM repeats the set of differences until all the differences between the two files are found. A row of 14 asterisks (*) marks the end of a difference.

For example, you have two text files named FIL1.TXT and FIL2.TXT. Use the TYPE command to examine the contents of both files:

```
@TYPE FIL1.TXT<RET>
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5
this is line 5.5
this is line 6
this is line 7
this is line 8
this is line 9  ;this is a comment
this is line 10
@TYPE FIL2.TXT<RET>
this is line 1
this is line 2
this is line 3
this is line 4
this is line 5.5
this is line 6
this is line 7


this is line 8
this is line 9
this is line 10
this is line 11
this is line 12
this is line 13
this is line 14, which is the end.
@
```

Run FILCOM to compare these files and name the output file DIFFER.SCM. Type the following:

```
@FILCOM<RET>

*DIFFER.SCM=FIL1.TXT,FIL2.TXT<RET>

%files are different

*
```

The program informs you that the files are different, and then gives you the asterisk prompt for more input. To see the differences that FILCOM found between the two files, return to TOPS-20 command level (type CTRL/C) and use the TYPE command.

```
%files are different
*CTRL/C
@TYPE DIFFER.SCM<RET>
FILE 1) DSK:FIL1.TXT[4,67]      created:  1212      25-Jul-1979
File 2) DSK:FIL2.TXT[4,67]      created:  1616      25-Jul-1979

1)1     this is line 5
1)      this is line 5.5
****
2)1     this is line 5.5
**************
1)1     this is line 9  ;this is a comment
1)      this is line 10
****
2)1     this is line 9
2)      this is line 10
2)      this is line 11
2)      this is line 12
2)      this is line 13
2)      this is line 14, which is the end.
**************
@
```

The output shows the differences between the two files. Line 5 was deleted, line 9 was changed, and lines 11-14 were inserted. The two blank lines in FIL2.TXT are ignored, because /B was not specified in the command string. The number "1" that appears beside 1) and 2) in the output is the page number of the file where the differences were found. Text pages are delimited by formfeeds.

Now, compare FIL1.TXT and FIL2.TXT using /C in the command string.

```
@FILCOM<RET>

*TTY:=FIL1.TXT,FIL2.TXT/C
File 1) DSK:FIL1.TXT[4,67]      created: 1212 25-Jul-1979
File 2) DSK:FIL2.TXT[4,67]      created: 1616 25-Jul-1979

1)1     this is line 5
```



```
1)      this is line 5.5
****
2)1     this is line 5.5
**************
2)1     this is line 11
2)1     this is line 12
2)1     this is line 13
2)1     this is line 14, which is the end.

%files are different
*
```

The output is different because /C causes FILCOM to ignore comments and spacing in the files.

Using the same two text files, now compare them in update mode, and write the output to the device TTY: (your terminal). Because FIL2.TXT is the latest version of the two text files, type it as the second input file in the command string. The command string is:

```
@FILCOM<RET>

*TTY:=FIL1.TXT,FIL2.TXT/U<RET>
        this is line 1
        this is line 2
        this is line 3
        this is line 4
|       this is line 5.5
        this is line 6
        this is line 7
|
|
|       this is line 8
|       this is line 9
|       this is line 10
|       this is line 11
|       this is line 12
|       this is line 13
|       this is line 14, which is the end.
%files are different

*
```

As mentioned previously, in update mode comparisons, the output file is the second input file (latest version) with change bars inserted next to the differences found between the two files. The example above shows such an output file. Note that the program also types the "%files are different" status on your terminal. Following this, FILCOM gives another prompt for another command string.

4.2.2 Comparing Binary Files

FILCOM automatically determines that a file is binary if it has one of the file types listed in Table 4–1.

Sharable and nonsharable save files represent the location of data in memory. In FILCOM, expanding the files before comparing them means to compare the data as it would appear if the files were loaded into memory. Comparing the files without expanding them means to compare each word in the files regardless of the usual meaning of the files' control words.

You can list a binary file with FILCOM. To do this, simply omit the comma and the second input file spec in the command string.

FILCOM contains the following switches that you use in the command string to compare binary files. Switches control what part of the binary file you see.

/E Forces FILCOM to consider both input files as sharable save files regardless of the file types given. Normally, FILCOM selects its comparison according to the file types of the files.

/H Prints the FILCOM help text. Refer to the description of /H in Section 4.2.1.

/nL Compares a binary file starting at word "n". The number "n" is an octal number. Refer to /nU, below.

/Q Instructs FILCOM to print only the status of the comparison. It does not list the actual differences, and causes FILCOM to stop reading the files after it discovers the first difference. Refer to the description of this switch in Section 4.2.1.

/nU Compares a binary file up through word "n". The value "n" is an octal number as in /nL. If you combine /nU with /nL in the command string, the input files are compared only within these limits.

/T Instructs FILCOM to generate an output file even if no differences are found. If the /T switch is not used, FILCOM produce no differences listing if there are no differences in the files.

/W Compares two binary files that have nonstandard file types. (Standard file types are listed in Table 4–1.) The files are not expanded before FILCOM compares them. /W compares the files' internal control words in addition to data, reading the files one word at a time.

/X Instructs FILCOM to expand nonsharable save files before comparing them in binary mode. The program ignores control words and compares only the code in the files.

All FILCOM binary switches apply when you dump a file. Expanding a file shows how it would appear in memory. Dumping a file without expanding it shows the file's internal format.

The output file of a binary mode comparison contains the same header as output files for ASCII comparisons. (Refer to Section 4.2.1.) However, the comparison differs because it is done word by word. If the left halves of the two words being compared are the same, FILCOM prints the absolute difference between the two words. Otherwise, FILCOM prints the logical exclusive OR (XOR).

For example, you want to compare two binary files. They are FIL3.BIN and FIL4.BIN. First you use FILCOM to dump them and examine their contents. The output is written to your terminal:

```
@FILCOM<RET>

*TTY:=FIL3.BIN<RET>
000000  201400 000000
000001  202400 000000
000002  202600 000000
000003  203400 000000
000004  203500 000000
000005  203600 000000
000006  203700 000000
000007  204400 000000
000010  204440 000000
000011  204500 000000

%files are different

*TTY:=FIL4.BIN<RET>
000000  201400 000000
000001  202400 000000
000002  202600 000000
000003  203400 000000
000004  203540 000000
000005  203600 000000
000006  203700 000000
000007  204420 000000
000010  204440 000000
000011  204500 000000

%files are different

*
```

Now compare the files. Note that since both file types are .BIN, the program automatically compares them in binary mode without you specifying any switches. Again, the output is written to your terminal:

```
@FILCOM<RET>

*TTY:=FIL3.BIN,FIL4.BIN<RET>
File 1) DSK:FIL3.BIN[4,67]      created: 1419 10-Sep-1979
File 2) DSK:FIL4.BIN[4,67]      created: 1419 10-Sep-1979

000004  203500 000000   203540 000000   000040 000000
000007  204400 000000   204420 000000   000020 000000

%files are different

*
```

Compare the files once more, but specify a quick comparison (/Q) in the command string. This causes FILCOM to merely report the status of the comparison. If there are differences, the status message prints with a question mark, ?, instead of a percent sign, %, for error detection in batch jobs. (Refer to Section 4.4, FILCOM Messages.) The command string for this comparison is as follows:

```
@FILCOM<RET>

*TTY:=FIL3.BIN,FIL4.BIN/Q<RET>
File 1) DSK:FIL3.BIN[4,67]      created: 1419 10-Sep-1979
File 2) DSK:FIL4.BIN[4,67]      created: 1419 10-Sep-1979

?files are different

*
```

Do a third comparison of these two files, but now restrict the range with /nU and /nL:

```
@FILCOM<RET>

*TTY:=FIL3.BIN,FIL4.BIN/5L/6U<RET>

No differences encountered
```

In the following example, you are comparing two executable files. FILCOM automatically expands them because of the .EXE file type:


```
@FILCOM<RET>

*TTY:=FIL1.EXE,FIL2.EXE<RET>
File 1) DSK:FIL1.EXE[4,67]      created: 1426 10-Sep-1979
File 2) DSK:FIL2.EXE[4,67]      created: 1427 10-Sep-1979

000141  600000 000000   255000 000000   455000 000000
002112  000004 015062   000004 015063          000001

%files are different

*
```

In this last example, you compare both .EXE files without expanding them first. Any differences that exist in the files' internal directory pages appear in the output. The command string for this comparison is as follows:

```
@FILCOM<RET>

*TTY:=FIL1.EXE,FIL2.EXE/W<RET>
File 1) DSK:FIL1.EXE[4,67]      created: 1426 10-Sep-1979
File 2) DSK:FIL2.EXE[4,67]      created: 1427 10-Sep-1979

000000  001776 000003   001776 000007          000004
000002  002000 000000   000000 000000   002000 000000
000003  001775 000003   000000 000002   001775 000001
000004  000000 254000   000000 000001          253777
000005  000000 000140   300000 000003   300000 000143
000006  001777 000001   000000 000002   001777 000003
000007  000000 000000   001775 000003   001775 000003
000010  000000 000000   000000 254000          254000
000011  000000 000000   000000 000140          000140
000012  000000 000000   001777 000001   001777 000001
001141  600000 000000   255000 000000   455000 000000
003112  000004 015062   000004 015063          000001

%files are different

*
```

4.3 FILCOM SWITCHES

Table 4-2 contains all the FILCOM switches in alphabetical order. It also lists the mode of comparison and description for each switch.

Table 4–2: FILCOM Switches

Switch	Comparison	Description
/A	ASCII	Compares files in ASCII mode
/B	ASCII	Considers blank lines in the comparison
/C	ASCII	Ignores comments and spacing in MACRO source files
/E	Binary	Considers both files as sharable save files
/H	ASCII& Binary	Prints the FILCOM help text
/nL	ASCII& Binary	Defines the number of lines that end a match in ASCII comparisons; determines the lower limit where a partial comparison begins in binary comparisons
/nU	Binary	Determines the upper limit where a partial binary comparison stops
/O	ASCII	Includes a label name and offset in the differences listing
/Q	ASCII& Binary	Prints only the status of the comparison; does not enumerate the differences
/S	ASCII	Ignores spacing and tabs in the comparison
/T	ASCII& Binary	Produces an output file even if no differences are found
/U	ASCII	Compares the files in update mode and inserts change bars next to the differences
/W	Binary	Compares binary files with nonstandard file types, ignoring control words (if any)
/X	Binary	Expands nonsharable save files before comparing them

4.4 FILCOM MESSAGES

Some of the messages printed by FILCOM contain information that is dependent on the exact command string, switch, or file you specified. The key to these message variables follows:

[device] A device name.

[file] A file spec.

[n] A designator for the first or second input file.

[reason] The reason for a file access failure. These are listed in Table 4–3 at the end of this section.

The most common FILCOM messages are listed below alphabetically. They are all fatal errors that are preceded by a question mark (?).

?Buffer capacity exceeded and no core available

Description: You attempted to compare two text files with a difference so large that FILCOM cannot obtain enough memory to store the differences.

Suggested User Response: Check that you are comparing two files that are reasonably similar; or, use /nL with n larger than the value 3.

?Command error

Description: You typed an invalid command. You may not have typed a second file specification. You may have included an invalid line terminator or a nonalphanumeric character in a file specification.

Suggested User Response: Retype the correct command syntax.

?Command error –– Unknown switch X

Description: You typed an invalid switch. The invalid switch is specified by the X character in the message.

Suggested User Response: Retype the correct switch and reissue the command.

?Command error –– Double filename illegal

Description: You typed a double filename in your command string.

Suggested User Response: Retype the correct command syntax and reissue the command.

?Command error -- Double file type illegal

Description: You typed a double file type in your command string.

Suggested User Response: Retype the correct command syntax and reissue the command.

?Device [device] Not available

Description: The device you specified is not available. In other words, someone may be using it, or there may be no such device on your system.

Suggested User Response: Specify a device available to you.

?FILE [n] NOT IN CORRECT EXE FORMAT

Description: The first or second input file in the command string is not a correctly formatted sharable save file (.EXE file type). You may have specified a file that is in nonsharable save file format (a result of the TOPS-20 CSAVE command, rather than the SAVE command).

Suggested User Response: Do not use nonsharable save files. They are less efficient than regular sharable save files. Bring the program into memory with the TOPS-20 GET command, save it again with the SAVE command, and then reissue the commands. You can also look at the files as nonsharable save files or pure binary files.

?File [n] not in SAV format

Description: Your first or second input file is not in proper nonsharable save file format.

Suggested User Response: Specify the correct file or look at the files as sharable save files or pure binary files.

?File [n] read error

Description: Your first or second input file contains an error.

Suggested User Response: Try the operation again. If it still fails, ask the operator to check the device for errors.

?Input error for input file [n]- [file] [reason]

Description: FILCOM could not access your first or second input file for the reason specified. (Refer to Table 4-3 for the exact reason.)

Suggested User Response: The particular reason in Table 4-3 gives corrective action.

?NOT ENOUGH CORE AVAILABLE TO HOLD DIRECTORY

Description: FILCOM is attempting to compare your sharable save files, but cannot get enough memory to hold the file's internal directory.

Suggested User Response: Re-create the .EXE file. If the error persists, contact your Software Specialist, or send a Software Performance Report (SPR) to DIGITAL.

? Output device error

Description: FILCOM received an error while writing your file.

Suggested User Response: The device may be faulty. If the problem persists, contact your operator to fix the problem.

?Output device error- [device] cannot do output

Description: You specified a device that is unable to do output, such as a card reader.

Suggested User Response: Specify a device that is capable of producing output.

?Output ENTER error for [file] [reason]

Description: FILCOM is unable to write the file for the reason specified.

Suggested User Response: Refer to Table 4-3 for corrective action that applies to the reason for the error.

Table 4-3 contains the various reasons for file access errors that can appear in FILCOM messages.

Table 4–3: Reasons for File Access Errors

Error	Reason
(0) File not found	You specified a file that does not exist. Specify an existing file.
(1) Nonexistent UFD	You specified a directory that does not exist. Specify an existing directory.
(2) Protection failure	You do not have sufficient privileges to access the named file. Negotiate the needed privileges with the system operator or the owner of the file.
(3) File being modified	Another job is currently modifying the named file. Try to access the file at another time or use a different filename.
(14) No room or quota exceeded	You exceeded the quota of the named directory or the entire capacity of the named file structure. Delete and expunge some of your files, or specify a directory or structure with sufficient space.
(15) Write–lock error	You requested an output file on a write–locked device, such as a magnetic tape. Either write–enable the device and try again, or specify another device.

CHAPTER 5

THE CREF PROGRAM

5.1 INTRODUCTION

The CREF program generates cross-reference listings from .CRF files. You produce .CRF files with LOAD-class commands. (Refer to the TOPS-20 Commands Reference Manual.) A cross-reference listing is one that contains your source program (from either the ALGOL or FORTRAN compilers, or the MACRO assembler) plus a list of all the symbols used and the line numbers on which they are used. As such, the CREF program is a helpful tool for debugging and modifying programs written in these languages.

NOTE

You do not need to run CREF for COBOL programs. The COBOL compiler automatically generates CREF listings.

5.2 RUNNING CREF

Because CREF reads only .CRF format files, you must first create these files before you run CREF. These files are created when your program is compiled. Section 5.2.1 describes how to create .CRF files.

Once you create .CRF files, you can run CREF to produce cross-reference listings. Section 5.2.2 describes how to run CREF.

5.2.1 Creating .CRF Files with COMPILE

To create a .CRF file, you must compile your program. To do this, type COMPILE after the TOPS-20 prompt @, followed by /CREF and the name of the program you want to compile; then press the RETURN key. After you press RETURN, the system compiles your program and then returns you to TOPS-20 command level. For example:


```
@COMPILE/CREF BE.MAC<RET>
MACRO:  BE

EXIT
@
```

Compiling a program does not automatically create a .CRF file. This is why you specify /CREF in the command string. For more information on the COMPILE command, refer to the TOPS-20 Commands Reference Manual.

You can create .CRF files of two programs in the same command string. For example:

```
@COMPILE/CREF BE.MAC,CREFA.MAC<RET>
MACRO:  BE
MACRO:  CREFA

EXIT
@
```

Note that this one command string is equivalent to creating .CRF files of both programs separately, as in the example below:

```
@COMPILE/CREF BE.MAC<RET>
MACRO:  BE

EXIT
@COMPILE/CREF CREFA.MAC<RET>
MACRO:  CREFA

EXIT
@
```

If your program has already been compiled, the above procedure will not work. You will immediately return to TOPS-20 command level after typing the command string. To create a .CRF file of an already compiled program, you must recompile the program. Type the COMPILE command followed by /COMPILE, /CREF, the program name, and press RETURN. For example:

```
@COMPILE/COMPILE/CREF  BE.MAC<RET>
MACRO:  BE

EXIT
@
```

5.2.2 Producing Cross-Reference Listings

Once you create a .CRF file, you can run CREF to generate the actual

cross−reference listing. There are four types of tables that you can get in these listings. The four types of tables are:

- Regular Symbols − This table contains user−defined symbols, labels, address tags, and assignments.
- OPDEF and Macro Names − This table contains user−defined operators such as macro calls and OPDEFs.
- Op Codes − This table contains hardware machine mnemonics and assembler pseudo−ops, such as MOVE and XALL.
- Procedure Nesting Levels − This table contains the names of procedures and numbers of blocks, indented to show their nesting in the program. Only the ALGOL compiler generates this table in a CREF listing.

You produce cross−reference listings by running CREF and giving a command string with switches to specify the type of listing you desire. The general command string format is:

Destination file spec=Source file spec1/Switches,Source file spec2/Switches...

where:

Destination file spec is the output file that contains the cross−reference listing.

If you omit the device and the filename in the Destination file spec, CREF uses the device LPT:. If you omit the device but do specify a filename, CREF uses the device DSK:. If you do not specify a filename in the Destination file spec, CREF uses the filename in Source file spec1. If you do not specify a file type in the Destination file spec, CREF uses the type .LST. If you omit the equal sign (=) in the command string, CREF uses all defaults for the Destination file spec (you only specify the Source file specs).

Source file spec1,Source file spec2...are the input files. These are the .CRF files you produced from compiling your programs.

If you omit the device in any of the Source file specs, CREF uses the device DSK:. If you omit a filename in any Source file spec, CREF uses the filename CREF. If you omit a file type in any Source file spec, CREF uses the file type .CRF, then .LST, .TMP, and null.

If you omit a PPN or a logical name for a PPN in either the Destination file spec or any of the Source file specs, CREF assumes that you mean your connected directory.

As mentioned, you can type CREF program switches in the command string after each input file spec. These switches and their functions are

described in Table 5−1.

Table 5−1: CREF Switch Options

Switch	Function
/A	Advances magnetic tape reel by one file. You can type this switch more than once in the command string.
/B	Backspaces magnetic tape reel by one file. You can type this switch more than once in the command string.
/C	Cancels the processing of any switches in your SWITCH.INI file.
/D	Restores the processing of any default switches in your SWITCH.INI file.
/H	Types the CREF help file. /H is illegal in a SWITCH.INI file.
/K	Suppresses Regular Symbol Table in the CREF listing.
/M	Suppresses OPDEF/Macro Table in the CREF listing.
/O	Includes the Op Code Table in the CREF listing.
/P	Preserves an input file with the file type .CRF or .LST. These types of input files are normally deleted.
/R	Requests the line number at which the CREF listing is to start. CREF types out "RESTART LISTING AT LINE:", after which you type the line number and press RETURN. If you use an indirect file, CREF looks for the number in the indirect file. /R is most useful for physical (non−spooled) line printers, and is illegal in a SWITCH.INI file.
/S	Suppresses the program listing and lists only the tables you select.
/T	Skips to the logical end of the magnetic tape.
/W	Rewinds the magnetic tape.
/Z	Zeros the DECtape directory. This is a historical switch, and is illegal.

If you always want to use specific switches when you run CREF, you can put these switches in a SWITCH.INI file. Then, each time you run CREF, it automatically reads these switches from the SWITCH.INI file. You can use all CREF switches in your SWITCH.INI file except /H and /R. For more information on creating a SWITCH.INI file, refer to the TOPS-20 User's Guide.

You can use an indirect file in CREF to reference a file within a command string. The indirect file can contain a complete or partial CREF command string (filenames and switches). For more information on using indirect files, refer to the TOPS-20 User's Guide.

You can produce cross-reference listings by typing CREF as a command or as a program name after the TOPS-20 prompt @. To run CREF as a command, type CREF after the TOPS-20 prompt @ and press RETURN. This causes CREF to produce a cross-reference listing of any .CRF files you created in that particular terminal session. For example, if you create .CRF files for BE.MAC and CREFA.MAC and then run CREF, type the following:

```
@CREF<RET>
CREF:   BE
CREF:   CREFA
@
```

Because you did not specify a switch, the default is for CREF to give you a cross-reference listing that contains a Regular Symbols Table and an OPDEF/Macro Table for each of the two .CRF files. If you only specify switches with this format, these switches apply to all files CREF processes. If one of your CRF files is an ALGOL program, the listing also includes the Procedure Nesting Level Table.

If you create .CRF files for several programs, but want a cross-reference listing of just one file, type CREF after the TOPS-20 prompt @ followed by a CREF command string. For example, you have .CRF files for BE.MAC and CREFA.MAC, but you only want a cross-reference listing for BE.MAC. Type the following:

```
@CREF BE.LST=BE.CRF/Switch<RET>
CREF:   BE
@
```

This command string causes CREF to produce a cross-reference listing for BE.MAC that includes any tables you specify with the switch.

To run CREF as a program, type R CREF after the TOPS-20 prompt @ and press RETURN. The program prompts you with an asterisk. Now type a CREF command string. For example, you create a .CRF file for BE.MAC and then run CREF as a program, specifying /O in the command string. Type the following:

```
@R CREF<RET>
```



```
*BE.LST=BE/O<RET>
[CRFXKC 5K CORE]
*
```

This command string causes CREF to produce an cross-reference listing for BE.MAC that contains a Regular Symbols Table, an OPDEF/Macro Table, and the Op Code Table (/O appears in the command string).

NOTE

For an explanation of the program message in square brackets that appears in the last example, refer to Section 5.5, CREF Messages.

You can also run other programs from CREF. To do this, type R CREF after the TOPS-20 prompt @ and press RETURN. After the asterisk prompt, type the filespec for the second program you wish to run, then type an exclamation point (!) and press RETURN.

5.3 CREF EXAMPLES

First, type your SWITCH.INI file to see that /O is included for CREF. Then define the logical name LPT: so that the CREF output goes to the disk, where you can type it. Without the logical name, the output is spooled to the line printer.

```
@TYPE SWITCH.INI<RET>
CREF/O
RUNOFF/CRETURN/MESSAGE:(NOPREFIX,FIRST)
MAKLIB/MESSAGE:(NOPREFIX,FIRST)
@DEFINE LPT: DSK:<RET>
@
```

Now, compile a MACRO assembler program and request a CREF listing. Process MACRO's listing file with CREF to get the final listing. Note that because of the /O on the CREF line in SWITCH.INI, CREF automatically includes the Opcode Table.

```
@COMPILE/COMPILE/CREF CREFA<RET>
 MACRO: CREFA

EXIT
@CREF<RET>
CREF: CREFA
@TYPE CREFA.LST<RET>
LCREFA CREF Example MACRO %53A(1152) 10:39 10-Sep-79 Page 1
CREFA MAC 11-Jul-79 11:54  Show What CREF Does
```

```
     1      TITLE CREFA CREF Example
     2      SUBTTL Show What CREF Does
     3
     4      SEARCH MACSYM, MONSYM
     5      SALL
     6      NOSYM
     7
     8   000000   T0=0
     9   000016   L=16
    10   000017   P=17
    11
    12   000017   CONST=17
    13      ENTRY TIMES.
    14
    15 000000' 201 00 0 00 000017  TIMES.: MOVX T0,CONST
    16 000001' 220 00 1 16 000000   IMUL T0,@(L)
    17 000002' 263 17 0 00 000000   RET
    18
    19      END

NO ERRORS DETECTED

PROGRAM BREAK IS 000003
CPU TIME USED 00:00.872

67P CORE USED

^LCONST     12#     15
L      9#     16
P     10#
T0      8#     15     16
TIMES.     13     15#
..MX1     15#     15     16
..MX2     15#     16
.PSECT     15     16

^LMOVX     15
RET     17

^LEND     19
ENTRY     13
IFDEF     15
IFE     15     16
IFNDEF     16
IMUL     16
MOVEI     15
NOSYM      6
PURGE     16
SALL      5
SEARCH      4
SUBTTL      2
TITLE      1
```



```
.IF     15
.IFN     15
.PSECT     15     16
@
```

Now, compile a FORTRAN program. Generate the CREF listing and type the result.

```
@COMPILE/COMPILE/CREF CREFB<RET>
 FORTRAN: CREFB
TIMES
@CREF<RET>
CREF: CREFB
@TYPE CREFB.LST<RET>
^LTIMES  CREFB.FOR FORTRAN V.5A(621) /KI/C 10-SEP-79 10:39 PAGE 1


00001       FUNCTION TIMES(ARG)
00002
00003       PARAMETER CONST = 17
00004       INTEGER TIMES, ARG
00005
00006 10    TIMES = CONST*ARG
00007       RETURN
00008
00009       END


SUBPROGRAMS CALLED



SCALARS AND ARRAYS [ "*" NO EXPLICIT DEFINITION - "%" NOT REFERENCED ]

 TIMES 1  ARG 2

TEMPORARIES



^LARG      1#      4#      6
CONST      3#      6
TIMES      1#      4#      6#
10P      6#

TIMES  [ NO ERRORS DETECTED ]
@
```

Finally, compile an ALGOL program. Generate the CREF listing as before and type the result. Note the Procedure Nesting Table.

```
@COMPILE/COMPILE/CREF CREFC<RET>
 ALGOL: CREFC

 EXIT
@CREF<RET>
CREF: CREFC
@TYPE CREFC.LST<RET>
DECsystem 10 ALGOL-60, Version 6A(654)   10-SEP-79    10:39:35
Command string: CREFC,CREFC/C=MISC:CREFC.ALG



     1 000005   B1           1 BEGIN REAL X, Y;
     2 000005                2
     3 000012                3     REAL PROCEDURE SQUAREROOT(X,L);
     4 000012                4
     5 000016                5         VALUE X;  REAL X;  LABEL L;
     6 000016                6
     7 000021   B2           7         BEGIN REAL Y, Z;
     8 000025                8             IF X < 0 THEN GOTO L;
     9 000027                9             Y := (1 + X)/2;
    10 000043               10         IT:     Z := (X/Y + Y)/2;
    11 000060               11                 IF ABS(Z - Y) < 1&-6 THEN GOTO OK;
    12 000063               12                 Y := Z; GOTO IT;
    13 000074               13         OK: SQUAREROOT := Z;
    14 000076        E2     14         END;
    15 000076               15
    16 000115               16 NG: WRITE("Number please: "); BREAK.OUTPUT;
    17 000117               17     READ(X);
    18 000122               18     Y := SQUAREROOT(X,NG);
    19 000127               19     WRITE("The square root of ");
    20 000132               20     PRINT(X);
    21 000135               21     WRITE(" is ");
    22 000140               22     PRINT(Y);
    23 000143               23     BREAK.OUTPUT;
    24 000145        E1     24 END;

 NO ERRORS



^LE----1 PROGRAM
    B----1          1
 B----2          7
  E----2        14
    E----1        24
^LABS    11
BREAKOUTPUT
    16     23
IT     10#     12
L      3#      5#      8
```



```
NG     16#     18
OK     11     13#
PRINT     20     22
READ     17
SQUAREROOT
      3#     13     18
WRITE     16     19     21
X      1#      3#      5#      8      9     10     17     18     20
Y      1#      7#      9     10     11     12     18     22
Z      7#     10     11     12     13
@
```

5.4 CREF MESSAGES

The messages printed by CREF fall into three general categories: informational messages, warning messages, and fatal errors. Informational messages are enclosed in square brackets [ ], and merely inform you of CREF's progress in processing your file. Warning messages are preceded by a percent sign (%) and indicate that something unexpected occurred, but that CREF was able to recover. In this case, verify that your output is correct. Fatal errors are preceded by a question mark (?), and indicate an occurrence that CREF could not handle. In this case, CREF aborts its operation. You must fix the problem before reissuing your command string. The % and the ? preceding the message are easily detected in Batch jobs.

Some of the messages contain variable information that is dependent on the exact command string, switch, or file you specified. These variables are as follows:

[access]	An octal code associated with a file access failure. For possible access failures, refer to Table 5-2 at the end of this section.
[device]	A device name.
[file]	A file spec.
[memory]	A memory size, such as 30K.
[status]	An octal code associated with a read or write error while processing a file. The meaning of this code is described in Table 5-3 at the end of this section.
[switch]	A switch name.

The most common CREF messages are listed below alphabetically.

?CRFCDN Can't get command file device [device]

Description: CREF cannot access the named device in your indirect file command.

Suggested User Response: Specify an existing or available device.

?CRFCEF Cannot ENTER file, [access] [file]

Description: CREF could not create the specified file for the reason associated with the printed access code. (Refer to Table 5–2.)

Suggested User Response: Refer to Table 5–2 for corrective action that applies to the reason for the error.

?CRFCFE Command file INPUT error, [status] [file]

Description: An input error occurred reading the named command file. For the meaning of the specified status code, refer to Table 5–3.

Suggested User Response: Refer to Table 5–3 for corrective action that applies to the reason for the error.

?CRFCFF Cannot find file, [access] [file]

Description: CREF cannot find the specified file for the reason associated with the printed access code. (Refer to Table 5–2.)

Suggested User Response: Refer to Table 5–2 for corrective action that applies to the reason for the error.

?CRFCFI Cannot find input file, [file]

Description: CREF cannot find the specified file. The reason can be any of those listed in Table 5–2.

Suggested User Response: Refer to Table 5–2 for corrective action that applies to the reason for the error.

?CRFCLC Can't LOOKUP command file [access] [file]

Description: CREF cannot find the specified command file for the reason associated with the printed access code. (Refer to Table 5–2.)

Suggested User Response: Refer to Table 5–2 for corrective action that applies to the reason for the error.

?CRFCME Command error – type /H for help

Description: You typed an invalid command.

Suggested User Response: Retype the correct command or type /H for help.

?CRFDNA Device not available, [file]

Description: You specified a device in the file specification that is not available to your job. It may be in use by another job.

Suggested User Response: Specify a device that is available to your job.

?CRFIBP Input buffer size phase error – 0 [file]

Description: The size of the buffer area set up by the system is larger than the size expected by CREF.

Suggested User Response: This is an internal error, and not expected to happen. If it does, contact your Software Specialist or send a Software Performance Report (SPR) to DIGITAL.

%CRFIDC Improper input data at line [decimal], continuing

Description: CREF detected incorrect data in your input file. This is most likely a problem with the compiler that generated the CREF input file.

Suggested User Response: First, be sure that you are processing a valid CREF input file. If so, this is an internal error, and not expected to happen. Try to re–create the .CRF file, since the data in the original one may have been corrupted. If the problem persists, contact your Software Specialist or send a Software Performance Report (SPR) to DIGITAL.

?CRFIMA Insufficient memory available

Description: Your input file is too large for CREF to handle.

Suggested User Response: Divide your program into smaller files and try again.

?CRFINE INPUT error, [status] [file]

Description: An input error occurred reading the named file. For the meaning of the specified status code, refer to Table 5–3.

Suggested User Response: Refer to Table 5–3 for corrective action that applies to the reason for the error.

?CRFOUE OUTPUT error, [status] [file]

Description: An output error occurred writing the named file. For the meaning of the specified status code, refer to Table 5–3.

Suggested User Response: Refer to Table 5–3 for corrective action that applies to the reason for the error.

%CRFPUE Please use "=" rather than "_"

Description: Older versions of CREF allowed only the use of an underbar ("_") to separate the input files from the output file. CREF currently allows either "=" or "_", but it is preferable for you to use "=". CREF acts as if you typed "=".

Suggested User Response: Use "=" in the future.

%CRFRLL Restart listing at line:

Description: You specified /R to continue a listing. CREF is requesting the line number of the line to resume printing.

Suggested User Response: Type the desired line number.

%CRFSIH "/H" or "/R" switch illegal in SWITCH.INI defaulting

Description: You specified /H or /R illegally in the CREF line in your SWITCH.INI file. CREF ignores these switches.

Suggested User Response: Remove /H or /R from your SWITCH.INI file.

%CRFSII Syntax error in SWITCH.INI defaults

Description: The CREF commands in your SWITCH.INI file are invalid. CREF ignores them.

Suggested User Response: Correct the invalid switches in your SWITCH.INI file.

?CRFSIO I/O error while reading SWITCH.INI

Description: An input error occurred reading SWITCH.INI switch defaults. The reason for the failure can be any of those listed in Table 5–3.

Suggested User Response: Refer to Table 5–3 for corrective action that applies to the reason for the error.

?CRFUKS Unknown switch "[switch]" {in DSK:SWITCH.INI}

Description: You specified a nonexistent switch.

Suggested User Response: Respecify the command string with a valid switch, or fix your SWITCH.INI file.

[CRFXKC [memory] [core]

Description: CREF finished its processing, and is informing you how much memory it used to process your CREF file.

The octal codes listed in Table 5–2 appear in various CREF error messages, wherever [access] is used. The explanation beside each octal code describes the reason for the failure. Appropriate corrective action for each failure follows the explanation.

Table 5–2: Reasons for File Access Errors

Octal Code	Explanation/Action
0	File not found. The named file was not found by CREF. Specify an existing file.
1	You specified a directory that does not exist. Specify an existing directory.
2	Protection failure. You do not have sufficient privileges to access the named file. Negotiate the needed privileges with the system operator or the owner of the file.
3	File being modified. Another job is currently modifying the named file. Try to access the file at another time or use a different filename.
14	No room or quota exceeded. You exceeded the quota of the named directory, or the entire capacity of the file structure. Delete and expunge some of your files, or specify a directory or structure with sufficient space.
15	Write–lock error. You requested an output file on a write–locked device, such as a magnetic tape. Either write–enable the device and try again, or specify another device.

The status code in various CREF error messages is an octal number that is best interpreted as bits. For example, if a CREF message prints the status code 40001, this means that you have exceeded your disk quota. The status code printed by CREF is the logical "OR" of all

applicable bit values. Table 5–3 contains these status codes, their meanings, and the corresponding remedial action.

Table 5–3: Error Status Codes

Status Code	Meaning/Action
400000	The device is write–locked. Either specify a write–enabled device or ask the system operator to write–enable your device.
200000	A hardware error occurred. The hardware is in error and should be fixed if the problem persists.
100000	A parity error occurred. The data on the device is incorrect. Correct the data and try again.
40000	Quota exceeded or structure full. You either exceeded your directory's quota or the entire capacity of the structure. Delete and expunge some files or specify a structure or directory with sufficient space.

NOTE

All other bit values (such as 20000 or 10) are simply status bits and do not indicate an error. Only those listed above indicate an error. Any other code simply indicates the status.

5.5 TECHNICAL NOTES

The information in this section is primarily for users who write their own compilers (such as MACRO, ALGOL, or FORTRAN) and users who create .CRF files. It describes the .CRF input file format and the various control characters that you use.

The control characters in Tables 5–4 and 5–5 appear in the CREF input file produced by MACRO, FORTRAN and ALGOL. At the beginning of each line of the listing, CREF input data is enclosed by DELETE B and

DELETE C. The symbol or instruction types and the number of their component characters are defined by control characters. The set of control characters defining symbols and instructions is the same as the set defining the number of symbol or instruction characters. A control character's position determines its function. For example, in the input CREF data <DELETE>B^C^CEND<DELETE>c, the <DELETE>B indicates the beginning of the data. The first CTRL/C (^C) defines the instruction END as a pseudo–op code. The second CTRL/C (^C) defines the number of characters in the instruction END as 3, and the <DELETE>C terminates the CREF data.

Symbols referenced in a block–structured language such as ALGOL should assign a unique number to each symbol name. This ensures that symbols with the same name but defined in different blocks have different numbers. Then, the number for each symbol should be referenced in CREF data rather than the name. After the last use of a symbol (usually at the end of its block), use CTRL/I to associate the symbol's name with the unique number assigned to it.

Control characters that require two symbols (such as CTRL/I) should have two adjacent length character and symbol name pairs.

The control characters that begin and end the CREF input data are described in

Table 5–4: Beginning and Ending Control Characters

Character	ASCII Codes	Meaning
<DEL> A (prints as A)	177, 101 (Octal)	Terminates the CREF data on each line and adds a horizontal tab to the line of the listing.
<DEL> B (prints as B)	177, 102	Signals beginning of the CREF data on each line.
<DEL> C (prints as C)	177, 103	Terminates the CREF data on each line. A line number is incremented and printed. This is the line number referenced in the CREF tables.
<DEL> D (prints as D)	177, 104	Identical to DELETE A, except that DELETE D does not increment or print line numbers.

<DEL> E (prints as E)	177, 105	Causes CREF to print its tables immediately, and reinitializes the tables for another program. FORTRAN uses this function between subroutines.
<DEL> F (prints as F)	177, 106	Identical to DELETE C, except that DELETE F does not increment or print line numbers.

Table 5–5 contains the control characters that define symbols, instruction types, and macros. Except for CTRL/B and CTRL/G, each of these characters precedes the symbol name being referenced.

Table 5–5: Symbol–Definition Control Characters

Character	ASCII Code	Meaning
CTRL/A (^A)	001	Declares a reference to a user–defined symbol (such as a regular MACRO symbol or a FORTRAN variable).
CTRL/B (^B)	002	Declares a defining occurrence of a user–defined symbol. This character immediately follows the symbol name.
CTRL/C (^C)	003	Declares a reference to a MACRO pseudo–op or a hardware–defined op code.
CTRL/D (^D)	004	Declares a defining occurrence of a MACRO pseudo–op or a hardware–defined op code.
CTRL/E (^E)	005	Declares a reference to a MACRO or OPDEF.
CTRL/F (^F)	006	Declares a defining occurrence of a MACRO or OPDEF.
CTRL/G (^G)	007	Causes CREF to delete the last symbol that it read.

CTRL/H (^H)	010	Combines two symbols that are defined at different block levels and are then found to be the same. The second symbol specified becomes the name for both. For example, in a one–pass block–structured compiler that allows symbol references before their definition, a symbol referenced in an inner block must be treated as a unique symbol until the end of the block. If no local definition is found, this symbol's references must then be combined with an outer block's references.
CTRL/I (^I)	011	Declares the user–defined symbol by giving it a name in place of the unique numeric. The numeric is the first argument, and its name is the second argument.
CTRL/J (^J)	012	Illegal in CREF as a symbol definition control character.
CTRL/K (^K)	013	Identical to CTRL/I, except that it operates on macros rather than on symbols.
CTRL/L (^L)	014	Illegal in CREF.
CTRL/M (^M)	015	Declares the beginning of a symbol block. The argument is the block name.
CTRL/N (^N)	016	Declares the end of a symbol block. The argument is the block name.
CTRL/O (^O)	017	Declares a line number to use in place of the line's actual position in the file. The line number is specified after the CTRL/O in the same format as other symbols.

The octal value of each character described in Table 5–6 is used by CREF to determine the number of characters in a symbol name. The same set of characters defines the symbol as well as its number of characters. The character's position determines its function. The character–count character immediately precedes the symbol with no intervening spaces or characters (^CEND, for example). Table 5–6 contains the characters and their meanings.

Table 5–6: Character–Count–Definition Characters

Character	ASCII Code	Meaning
CTRL/A (^A)	001 (octal)	The symbol contains 1 character.
CTRL/B (^B)	002	The symbol contains 2 characters.
CTRL/C (^C)	003	The symbol contains 3 characters.
:	:	:
:	:	:
:	:	:
?	077	The symbol contains 63 characters.

The following example illustrates the symbols and references recognized by CREF that you can make in an Assembly Language Program. The control characters show the references that are made to particular symbols.

Source File CREFD.MAC:

```
 TITLE CREFD CREF Example
 SUBTTL Show Internal CREF Information

 SEARCH MACSYM, MONSYM ;Pseudo–op reference

 CONST=17  ;Symbol definition

 DEFINE MACRO(X)< ;;Pseudo–op reference, macro definition (in–line)
 OPCODE X  ;OPDEF and symbol reference (when expanded)
>

 OPDEF OPCODE[MOVEI] ;Pseudo–op reference, OPDEF definition and
    ;  opcode reference

 START: MACRO CONST  ;Symbol and macro reference

  END START  ;Pseudo–op and symbol reference
 .CRF File CREFD.CRF:
```


NOTE

In this printout of a .CRF File, DELETE characters are designated as follows:

DELETE B = ^?B

```
   ^LCREFD CREF Example MACRO %53A(1152) 13:44 10–Sep–79 Page 1
   CREFD MAC 10–Sep–79 13:44  Show Internal CREF Information

   ^?B^C^ETITLE^?C      TITLE CREFD CREF Example
   ^?B^C^FSUBTTL^?C     SUBTTL Show Internal CREF Information
   ^?B^?C
   ^?B^C^FSEARCH^?C     SEARCH MACSYM, MONSYM ;Pseudo–op reference
   ^?B^?C
   ^?B^A^ECONST^B^?C   000017  CONST=17  ;Symbol definition
   ^?B^?C
   ^?B^C^FDEFINE^F^EMACRO^?C      DEFINE MACRO(X)< ;;Pseudo–op reference, macro definition
 (in–line)
   ^?B^?C      OPCODE X  ;OPDEF and symbol reference (when expanded)
   ^?B^?C     >
   ^?B^?C
   ^?B^C^EOPDEF^C^EMOVEI^F^FOPCODE^?C  201000 000000   OPDEF OPCODE[MOVEI] ;Pseudo–op refe
rence, OPDEF definition and
   ^?B^?C         ;  opcode reference
   ^?B^?C
   ^?B^A^ESTART^B^E^EMACRO^?C 000000'    START: MACRO CONST  ^;Symbol and macro reference
   ^?B^E^FOPCODE^A^ECONST^?C 000000' 201 00 0 00 000017   OPCODE CONST  ;OPDEF and symbol
reference (when expanded)
   ^?B^?C
   ^?B^C^CEND^A^ESTART^?C   000000'   END START  ;Pseudo–op and symbol reference

   NO ERRORS DETECTED

   PROGRAM BREAK IS 000001
   CPU TIME USED 00:00.904

   67P CORE USED
   ^LCREFD CREF Example MACRO %53A(1152) 13:44 10–Sep–79 Page S–1
   CREFD MAC 10–Sep–79 13:44  SYMBOL TABLE

   CONST  000017
   OPCODE 201000 000000
   START  000000'
   The following is an example of a listing you receive after running
   CREF with the preceding .CRF file:

   ^LCREFD CREF Example MACRO %53A(1152) 13:58 10–Sep–79 Page 1
   CREFD MAC 10–Sep–79 13:44  Show Internal CREF Information
```

```
       1      TITLE CREFD CREF Example
       2      SUBTTL Show Internal CREF Information
       3
       4      SEARCH MACSYM, MONSYM ;Pseudo-op reference
       5
       6   000017   CONST=17  ;Symbol definition
       7
       8      DEFINE MACRO(X)< ;;Pseudo-op reference, macro definition (in-line)
       9      OPCODE X  ;OPDEF and symbol reference (when expanded)
      10     >
      11
      12  201000 000000   OPDEF OPCODE[MOVEI] ;Pseudo-op reference, OPDEF definition and
      13         ;  opcode reference
      14
      15 000000'    START: MACRO CONST  ^;Symbol and macro reference
      16 000000' 201 00 0 00 000017   OPCODE CONST  ;OPDEF and symbol reference (when exp
anded)
      17
      18   000000'   END START  ;Pseudo-op and symbol reference

   NO ERRORS DETECTED

   PROGRAM BREAK IS 000001
   CPU TIME USED 00:00.873

   67P CORE USED
   ^LCREFD CREF Example MACRO %53A(1152) 13:58 10-Sep-79 Page S-1
   CREFD MAC 10-Sep-79 13:44  SYMBOL TABLE

   CONST  000017
   OPCODE 201000 000000
   START  000000'


   ^LCONST      6#     16
   START     15#     18

   ^LMACRO      8#     15
   OPCODE     12#     16

   ^LDEFINE      8
   END     18
   MOVEI     12
   OPDEF     12
   SEARCH      4
   SUBTTL      2
   TITLE       1
```

CHAPTER 6

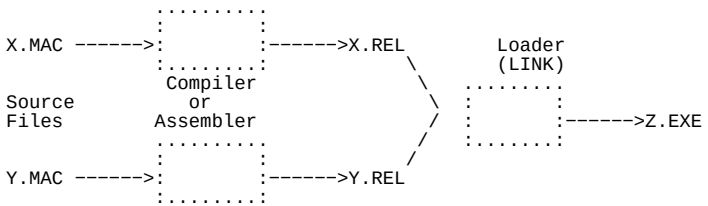
THE MAKLIB PROGRAM

6.1 INTRODUCTION

The MAKLIB program organizes and manipulates files of relocatable object (REL) modules. These REL modules are the output from a source language translator, such as the FORTRAN compiler or the MACRO assembler.

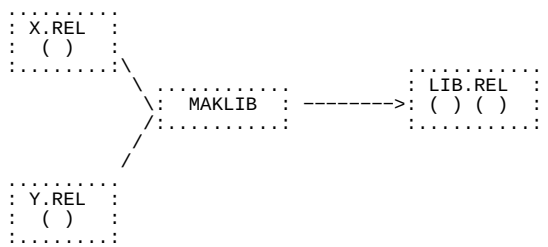
At load time, the modules are linked together to build an executable program. As shown in Figure 6–1, the MACRO assembler processes two source files, X.MAC and Y.MAC, and produces two corresponding .REL files, X.REL and Y.REL. The LINK program then loads the resulting object modules from these .REL files and produces an executable program, Z.EXE.

Figure 6–1: Figure Generation of an .EXE File



Multiple modules can be concatenated into a single file called a library. A file containing a single module can also be called a library. (See Figure 6–2.)

Figure 6–2: Generation of a Library



MAKLIB performs four functions on library files. Each function has switches that cause the MAKLIB program to do a specific task. The four functions are:

Obtaining Information about Libraries

These switches cause MAKLIB to give information about the status and contents of the library.

Manipulating Libraries

These switches cause MAKLIB to create new libraries by combining modules. Other switches cause MAKLIB to add, delete, extract, or replace modules.

Modifying Libraries

These switches cause MAKLIB to create new libraries from existing libraries, either by adding an index or removing local symbols. By modifying libraries you can reduce the amount of processing time required by the LINK program.

Editing Libraries

These switches cause MAKLIB to edit (or patch) modules within the library. You selectively change the object code in a module by supplying MAKLIB with the required MACRO assembly language code changes.

For more information on the contents of .REL files and REL Block types, refer to the TOPS–20 LINK Reference Manual. For more information on MACRO assembly language, refer to the TOPS–20 MACRO ASSEMBLER Reference Manual.

6.2 RUNNING MAKLIB

To run MAKLIB, type MAKLIB after the TOPS-20 prompt @ and press the RETURN key. The program prompts you with an asterisk:

```
@MAKLIB<RET>
*
```

After this prompt, enter a command string. MAKLIB takes commands in the following format:

Destination file spec=Source file spec1/Switches,Source file spec2/Switches... Source file specn/Switches

where:

Destination file spec is the output file that MAKLIB produces. It can be either a text file or a library, depending upon the function you perform.

If you do not specify a Destination filename, MAKLIB uses the name of the file in Source file spec1. If you omit the Destination file type, the default depends on the function you perform.

Source file spec1 is the master library. This file spec is always required in a MAKLIB command string.

You must specify a filename in Source file spec1. The default file type is always .REL.

Source file spec2 ...Source file specn are the transaction files. These are additional input files required to perform some MAKLIB functions. A function usually requires only one transaction file.

You include switches in the command string to instruct MAKLIB to perform a specific function. You specify switches in one of the following formats:

```
/Switch
/Switch:argument
/Switch:(arg1,arg2...argn)
```

You can perform only one action with a switch in a single command string, but MAKLIB allows up to 100 switch arguments for each command string.

You can use MAKLIB switches in abbreviated form as long as they remain unique. However, arguments to switches are usually module names and hence cannot be abbreviated. Parentheses are optional when you specify only one argument, but are required to enclose two or more switch arguments.

Table 6-1 is an alphabetical list of the MAKLIB switches; they are discussed in detail in Sections 6.2.1 through 6.2.4.

You can use an indirect file in a MAKLIB command string to reference another file. The indirect file can contain a complete or partial MAKLIB command string (filenames and switches). For more information on using indirect files, refer to the TOPS-20 User's Guide.

If you always want to use specific switches when you run MAKLIB, you can put these switches in a SWITCH.INI file. For more information on creating a SWITCH.INI file, refer to the TOPS-20 User's Guide.

MAKLIB allows you to type a string of commands on one or more lines. If the command string takes more than one line, type a hyphen (-) at the end of the first line, and press RETURN. Then, when MAKLIB prompts with a pound sign (#), continue to type the command string on the next line. You can also use multi-line commands in an indirect file.

To exit from MAKLIB and return to TOPS-20 command level, type either /EXIT or CTRL/Z after the asterisk prompt.

6.2.1 Running MAKLIB to Obtain Information About Libraries

MAKLIB contains four switches that allow you to obtain information on the status and contents of the master library (first input file). The four switches are: /LIST, /POINTS, /TRACE, and /LOAD.

Command String Requirements -

Files:	A master library and an output file are required in the command string for each of the four switches. None of the command strings require a transaction file.
Default file type:	Output file type - .LST Master library type - .REL
Arguments:	(Modules affected by the switch) - None

The output file (.LST) is a text file that can be written to any output device that supports text files, such as TTY: or LPT:. The discussion of /POINTS in this section illustrates the use of this option.

/LIST - LIST Switch

This switch lists the names of the modules that are contained in the master library. In addition to the names, MAKLIB also lists the two data values from the END block (REL Block type 5) of the

module. If the module is a two-segment program, the first value is the high-segment break, and the second value is the low-segment break. If the module is a one-segment program, the first value is the program break, and the second value is the absolute break. If the second value is zero, it is not printed.

For example, if you want to create a file, REAP.LST, showing the names of all modules in the library IAGO.REL, give the following command string:

```
@MAKLIB<RET>
*REAP.LST=IAGO.REL/LIST<RET>
*
```

When MAKLIB finishes the task you request, it prompts you with another asterisk. You can now enter another command string.

You get the following when you type the new file:

```
@TYPE (FILE) REAP.LST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B (  ) on 12-Dec-81 at 14:02:41

        *************************

DSK:IAGO.REL[4,244] created on 8-Dec-81 at 14:32:00

IAGO   401725   023746
@
```

The output file REAP.LST shows that the file IAGO.REL contains one module named IAGO. This module is a two-segment program. The first value listed, 401725, is the high-segment break. The second value listed, 023746, is the low-segment break.

/POINTS - POINTS Switch

This switch lists all entry points in the specified library. Entry points are usually subroutine starting addresses. They are used by the LINK program to determine if a global request can be satisfied by loading a module from a library.

For example, if you want to know all the entry points in the library NICE.REL and have the output written to the device TTY: (your terminal), do the following:

```
@MAKLIB<RET>
*TTY:=NICE.REL/POINTS<RET>
        Listing of Modules and Entry Points
Produced by MAKLIB Version 2B (  ) on 12-Dec-81 at 16:33:45

        *************************
```



```
DSK:NICE.REL[4,244] created on 12-Dec-81 at 16:11:00

NICE    BEGIN   LOOP    NICE
*
```

In this example, the output shows that the library NICE.REL contains one module, NICE, which has three entry points: BEGIN, LOOP, and NICE. After the output, MAKLIB returns with the asterisk prompt and waits for another command string.

/TRACE - TRACE Switch

This switch lists all the edits you have made to a library. This information is contained in the TRACE blocks in the specified library. MAKLIB creates these TRACE blocks (REL Block type 1060) when you use /FIX to edit a module in the library. The TRACE blocks contain information about the edits you insert and the changes you make to the library. For more information on TRACE blocks, refer to Section 6.5. /TRACE allows you to determine the exact binary patching status of the library.

For example, if you want to see all the edits in the library OLDLIB.REL, and have the output written to the device TTY:, do the following:

```
@MAKLIB<RET>

*TTY:=OLDLIB.REL/TRACE<RET>
        Listing of TRACE blocks
Produced by MAKLIB Version 2B(  ) on 10-Dec-81 at 9:33:59

        *************************

DSK:OLDLIB.REL[4,601] created on 1-Dec-81 at 9:31:00

Module: SORT  Edit: 341
  Status is Active
  Last affected by DRB
  Created by HAS On  2-Dec-81
  Installed by DRB On 10-Dec-81
  Program changes:
    Inserts 4 instruction(s) at location 001046
```

This example shows that the module SORT in the library OLDLIB.REL has one edit inserted. The status of the edit is active. You can change the status of a particular edit with the .REMOVE or .REINSERT pseudo-ops. The example also shows that the edit was last affected by DRB. This information comes from /WHO when you install or change an edit. The edit was created by HAS on 2-Jun-79. This information comes from the .NAME and .DATE pseudo-ops, respectively. The edit was installed by DRB on 10-Jul-79. This information comes from /WHO and the date that

the system supplies when you install an edit. For more information on these pseudo-ops and /WHO, refer to Section 6.2.4.

/LOAD - LOAD Switch

This switch shows additional loading instructions that are embedded within the library in either REQUEST (REL Block type 17), REQUIRE (REL Block type 16), or ASCII text blocks.

The library QUICK.REL was produced from a MACRO program containing the following statements:

```
TITLE     QUICK
.TEXT     "/VERSION:2(111)"
.REQUEST  SYS:FORLIB
.REQUIRE  SYS:MACREL
```

To see the additional loading instructions embedded within QUICK.REL in the blocks, do the following:

```
@MAKLIB<RET>
*TTY:=QUICK.REL/LOAD<RET>
        Listing of Internal loading instructions
Produced by MAKLIB Version 2B(  ) on 10-Dec-81 at 9:06:23

        *************************

DSK:QUICK.REL[4,601] created on 6-Dec-81 at 13:28:00

Module: QUICK
  Text string: /VERSION:2(111)
  Requests SYS:FORLIB
  Requires SYS:MACREL
```

6.2.2 Running MAKLIB to Manipulate Libraries

For handling and creating libraries, MAKLIB includes six switches that allow you to work with individual modules within libraries. The six switches are: /MASTER, /APPEND, /DELETE, /EXTRACT, /INSERT, and /REPLACE.

Command String Requirements -

Files: A master library (first input file) and an output file are required in the command string for each of the six switches. A transaction file is required with some switches.

Default file type: .REL for all files

Arguments: All switches accept arguments. /APPEND and /INSERT are two switches that do not always require arguments. For more information, refer to the discussions of these two switches in this section.

/MASTER - MASTER Switch

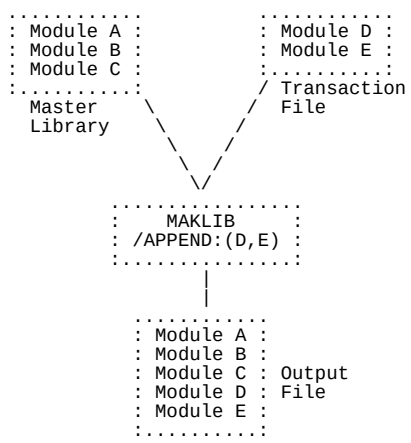
This switch identifies modules within the master library that correspond to those in the transaction file being used to effect the update. /MASTER takes at least one argument, and requires that another switch be given in the same command string. It is the only switch within this function that causes no real manipulation of a library. It is mentioned here because some of the switches used to manipulate libraries require /MASTER in their respective command strings.

You include /MASTER in the command strings for only two switches, /INSERT and /REPLACE. These switches are discussed later in this section.

/APPEND - APPEND Switch

This switch adds new modules to the end of an existing library. The output file is the master library plus the appended modules. MAKLIB reads these appended modules from the transaction file. You specify them as arguments to the switch. You must specify modules as arguments in the same physical order as they occur in the transaction file. Figure 6-3 illustrates the function of /APPEND. Note that, in the figure, modules D and E from the transaction file are appended to the master library.

Figure 6-3: Function of /APPEND



NOTE

When you do not specify an argument to /APPEND, the entire transaction file is appended to the master library.

For example, you want to append the module IAGO in the library IAGO.REL to the library GRAF.REL. You name the output file EXEN.REL. The command string is:

```
@MAKLIB<RET>
*EXEN.REL=GRAF.REL,IAGO.REL/APPEND:IAGO<RET>
*
```

MAKLIB returns with the asterisk prompt for you to enter another command string. To check the new file, use /LIST and have the output written to the device TTY:.

```
*TTY:=EXEN.REL/LIST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B(  ) on 14-Dec-81 at 10:45:13

        *************************

DSK:EXEN.REL[4,244] created on 14-Dec-81 at 10:43:00

CRIG    004582
```



```
PASZ    003217
TENP    036651
IAGO    402420   023746
*
```

The example shows that the new file, EXEN.REL, contains four modules: the three modules from the library GRAF.REL (CRIG, PASZ, TENP), and the appended module IAGO.

/APPEND also allows you to initially create a library. For example, you wish to combine the following four .REL files into a single library:

```
GLOBAL.REL
DTABLE.REL
INOUT.REL
IO.REL
```

Type the following MAKLIB command string to produce the file LIB.REL, which contains all modules from the four specified files:

```
@MAKLIB<RET>
*LIB=GLOBAL,DTABLE/APPEND,INOUT/APPEND,IO/APPEND<RET>
*
```

In this example, MAKLIB copies the file GLOBAL to a file named LIB, and then appends all modules from the files DTABLE, INOUT, and IO to LIB. This also illustrates the use of more than one transaction file.

/DELETE − DELETE Switch

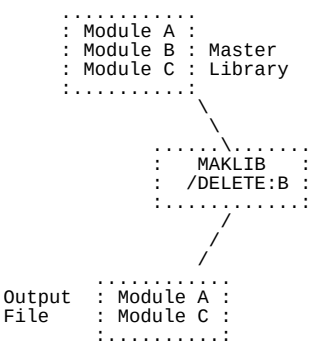
This switch removes one or more modules from an existing library. The output file is the master library minus the deleted module(s). All modules, except those specified as arguments to /DELETE, are read from the master library and copied to the output file. No transaction file is required.

NOTE

You must specify modules as arguments in the same physical order as they occur in the master library.

Figure 6−4 illustrates the function of /DELETE. Note that, in the figure, module B is deleted from the master library.

Figure 6−4: Function of /DELETE



Type the following command string to delete the module TENP from the library EXEN.REL. The output filename is DIPEP.REL.

```
@MAKLIB<RET>
*DIPEP.REL=EXEN.REL/DELETE:(TENP)<RET>
*
```

The program returns with the asterisk prompt for you to enter another command string. To check the new file, use /LIST and have the output written to the device TTY:.

```
*TTY:=DIPEP.REL/LIST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B(  ) on 14-Dec-81 at 14:10:41

        *************************

DSK:DIPEP.REL[4,244] created on 14-Dec-81 at 14:10:00

CRIG      004582
PASZ      003217
IAGO      402420   023746
*
```

/EXTRACT - EXTRACT Switch

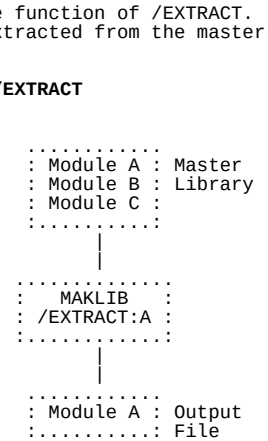
This switch produces an output file that is a subset of modules in the master library. You specify the modules as arguments to the switch. No transaction file is required.

NOTE

You must specify the modules as arguments in the same physical order as they occur in the master library.

Figure 6-5 illustrates the function of /EXTRACT. Note that, in the figure, module A is extracted from the master library.

Figure 6-5: Function of /EXTRACT



The following example illustrates /EXTRACT. The library FOO.REL contains four modules. To get a listing of the module names, use /LIST and have the output written to the device TTY:.

```
@MAKLIB<RET>
*TTY:=FOO.REL/LIST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B(  ) on 18-Dec-81 at 14:37:45

        *************************

DSK:FOO.REL[4,244] created on 18-Dec-81 at 13:54:00

SQUARE  000023
BOX     000014
MAIN    000433
DRAW    000505
*
```

You want to extract two of these modules, SQUARE and BOX, from the library FOO.REL, and put them in a new library, 2FOO.REL. After using /EXTRACT, check on the new file with /LIST, and have

the output written to the device TTY:. Type the MAKLIB command string after the asterisk prompt:

```
*2FOO.REL=FOO.REL/EXTRACT:(SQUARE,BOX)<RET>
*TTY:=2FOO.REL/LIST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B(  ) on 18-Dec-81 at 14:57:17

        *************************

DSK:2FOO.REL[4,244] created on 18-Dec-81 at 14:56:00

SQUARE  000023
BOX     000014
*
```

The example shows that the new library, 2FOO.REL, contains the two modules that you extracted from the master library, FOO.REL.

/INSERT - INSERT Switch

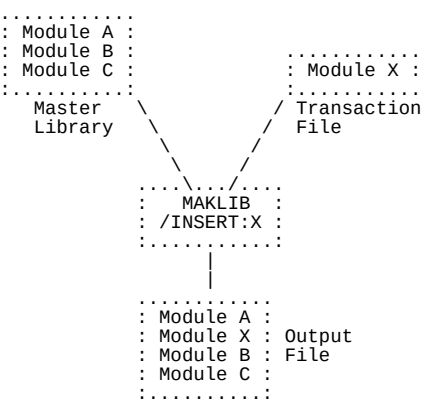
This switch inserts new modules into a master library. /MASTER is required in the command string. The output file is formed as follows: MAKLIB copies the master library to the output file up to but not including the module named as the first argument to /MASTER. Next, MAKLIB copies the module named as the first argument to /INSERT from the transaction file to the output file. The process repeats until the argument list specified to /MASTER and /INSERT is exhausted. At this point, MAKLIB copies the remaining modules in the master library to the output file. There must be one argument to /MASTER for every argument to /INSERT.

NOTE

You must specify the module names in the argument lists for /MASTER and /INSERT in the same physical order as they occur in the master library and the transaction file, respectively. When you do not specify an argument to /INSERT, the entire transaction file is inserted before the module you specify to /MASTER. You must always specify an argument to /MASTER.

Figure 6-6 illustrates this function of /INSERT. Note that, in the figure, module X in the transaction file is inserted before module B in the master library.

Figure 6-6: One Function of /INSERT



For example, the library FOO.REL contains four modules: SQUARE, BOX, MAIN, and DRAW. The library NICE.REL contains one module, NICE. You want to insert the module NICE before the module BOX in FOO.REL. The name of the output file containing the five modules is CLAR.REL. The command string to insert this module is as follows:

```
@MAKLIB<RET>
*CLAR.REL=FOO.REL/MASTER:BOX,NICE.REL/INSERT:NICE<RET>
*
```

MAKLIB returns with the asterisk prompt. You can now check on the new library, CLAR.REL, with /LIST. The output from /LIST is written to the device TTY:

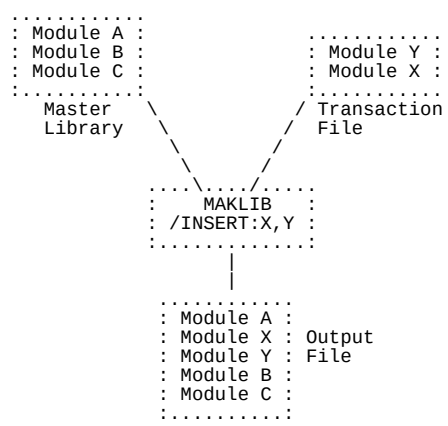
```
*TTY:=CLAR.REL/LIST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B(  ) on 27-Dec-81 at 15:40:28

        *************************

DSK:CLAR.REL[4,244] created on 27-Dec-81 at 15:40:00

SQUARE   000023
NICE     005557
```

```
BOX      000014
MAIN     000433
DRAW     000505
*
```

You may also insert more than one module in front of a module in a master library. However, the master library module name must appear repeatedly in the argument list to /MASTER. This produces a one-to-one correspondence between the module in the master library and the modules you wish to insert. In this case, you must list the argument names for both /MASTER and /INSERT in the same physical order that they appear as modules in the master library and transaction file, respectively.

Figure 6-7 illustrates this function of /INSERT. Note that, in the figure, modules X and Y in the transaction file are inserted before module B in the master library.

Figure 6-7: One Function of /INSERT



For example, the library SFJA.REL contains two modules: ILJA, and HLBET. The library FOO.REL contains four modules: SQUARE, BOX, MAIN, DRAW. You want to insert both modules in SFJA.REL in the library FOO.REL, before the module DRAW. The name of the new library is SFOO.REL. The command string is:


```
@MAKLIB<RET>
*SFOO.REL=FOO.REL/MASTER:(DRAW,DRAW),SFJA.REL/INSERT:(ILJA,HLBET)<RET>
*
```

After the asterisk prompt, check on the contents of the new library, SFOO.REL, with /LIST and have the output written to the device TTY:.

```
*TTY:=SFOO.REL/LIST<RET>
          Listing of Modules
Produced by MAKLIB Version 2B(  ) on 28-Dec-81 at 16:30:22

        *************************

DSK:SFOO.REL[4,244] created on 28-Dec-81 at 16:30:00

SQUARE   000023
NICE     005557
BOX      000014
MAIN     000433
ILJA     001047
HLBET    000433
DRAW     000505
*
```

/REPLACE - REPLACE Switch

This switch replaces modules in the master library with those specified in the transaction file. /MASTER is required in the command string so that the program can identify the modules in the master library that are to be replaced by those named as arguments to /REPLACE. There must be a one-to-one correspondence between the number of arguments to /MASTER and /REPLACE.

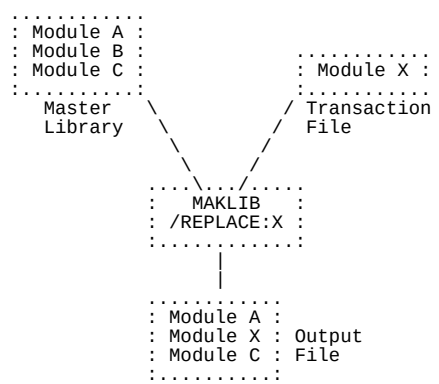
The output file is the entire master library, with modules replaced by those read from the transaction file (and named as arguments to the switch).

NOTE

You must specify the names in both argument lists (/MASTER and /REPLACE) in the same physical order as they appear as modules in the master library and transaction file, respectively.

Figure 6-8 illustrates the function of /REPLACE. Note that module X in the transaction file replaces module B in the master library.

Figure 6-8: Function of /REPLACE



For example, the library FOO.REL contains four modules: SQUARE, BOX, MAIN, and DRAW. The library NICE.REL contains one module, NICE. You want to replace the module MAIN in FOO.REL with the module NICE in NICE.REL. The output file name is WELW.REL. The command string is the following:

```
@MAKLIB<RET>
*WELW.REL=FOO.REL/MASTER:MAIN,NICE.REL/REPLACE:NICE<RET>
*
```

After MAKLIB completes the action you requested, it prompts you with an asterisk. Now check on the contents of the new library, WELW.REL, with /LIST and have the output written to the device TTY:.

```
*TTY:=WELW.REL/LIST<RET>
        Listing of Modules
Produced by MAKLIB Version 2B(  ) on 6-Dec-81 at 14:49:22

        *************************

DSK:WELW.REL[4,244] created on 6-Dec-81 at 14:48:00

SQUARE  000023
BOX     000014
NICE    005557
DRAW    000505
```



```
*
```

NICE replaced MAIN in the master library FOO.REL. The new library, WELW.REL, contains the four modules: SQUARE, BOX, NICE, and DRAW.

6.2.3 Running MAKLIB to Modify Libraries

The two switches within this function facilitate the processing of requests by the LINK program when it loads modules from libraries. The two switches are: /INDEX and /NOLOCALS.

Command String Requirements -

Files: A master library (first input file) and an output file are required in the command string for both switches. Transaction files are not allowed. Both switches appear with the master library in the command string.

Default file type: .REL for both the master library and the output file for both switch command strings.

Arguments: None

/INDEX - INDEX Switch

This switch produces an output file, which is identical to the master library, except with INDEX blocks (REL Block type 14) inserted in the file. Normally, programs make external requests to library subroutines they need, and LINK must search completely through the library to decide which modules to load to satisfy the requests. INDEX blocks list the entry point names and corresponding modules, allowing LINK to quickly determine which modules to load. LINK searches more efficiently, and loading time is shorter because the amount of I/O is reduced.

/NOLOCALS - NOLOCALS Switch

This switch produces an output file which is the master library with all local symbols deleted from the file SYMBOL blocks (REL Block type 2). Local symbols are useful for debugging purposes, and also when modules are edited with MAKLIB. (Refer to FIX files, Section 6.2.4.) In a production-mode library, local symbols are usually deleted, because they serve no purpose. This reduces the amount of mass storage space the library occupies. In addition, loading time is faster because the amount of I/O is reduced. Global symbols are not deleted because they are used in the linking of modules.

In the following example you create a new library, L2, from L1. The new library has an index but no local symbols. You can use /INDEX and /NOLOCALS together in the command string.

```
@MAKLIB<RET>
*L2=L1/NOLOCALS/INDEX<RET>
*
```

6.2.4 Running MAKLIB to Edit Libraries

MAKLIB provides a mechanism for you to patch (or edit) the code of a relocatable object module. This patching facility allows you to make program changes directly to a library. Although MAKLIB provides the facility for editing a program without having to change the source code, good programming practice requires that the same edits also be made at the source level.

To edit a library in this way, you must first create a text file called a .FIX file. This .FIX file contains one or more edits that you want to insert in the library. Each edit has a unique identifier and consists of a sequence of control pseudo-ops and MACRO assembly language code. The control pseudo-ops tell MAKLIB where and how to make the changes. Any new code is supplied as a sequence of MACRO assembly language statements. Each edit begins with an .EDIT pseudo-op and ends with an .ENDE pseudo-op. Figure 6-9 shows the order that pseudo-ops must appear within an edit.

When MAKLIB processes the .FIX file, it creates a new library with any new edits inserted. Although each edit is now a permanent part of the library, you can use MAKLIB to deactivate an edit. This operation effectively removes any code changes inserted by the edit. Therefore, edits in the library can be either active or inactive.

MAKLIB maintains edit history information on the library by generating TRACE blocks (REL Block type 1060) for each edit you insert. TRACE blocks are part of the module. Therefore, when you use /REPLACE, /EXTRACT, or /DELETE, the TRACE blocks move with the module. Section 6.2.1 describes how you can determine the edit status of the library using /TRACE.

NOTE

MAKLIB does not handle PSECTS.

Pseudo-ops for .FIX files

.EDIT xxxxxx - This pseudo-op is an identifying name for the edit you insert in the specified module. The edit name (xxxxxx) can

be up to six (6) SIXBIT characters and is stored in the TRACE block for any module affected by the edit. .EDIT is the first pseudo-op for each edit.

.DATE dd-mmm-yy - This pseudo-op gives the date that the edit was made. The day (dd) and year (yy) entries are numeric. The month entry (mmm) is alphabetic. .DATE is an optional pseudo-op. If you supply this information, it is stored in the edit TRACE block.

.NAME xxx - This pseudo-op gives the three initials (xxx) of the person who wrote the edit. .NAME is an optional pseudo-op. If you supply these initials, they are stored in the TRACE block for the edit.

.MODULE xxxxxx - This pseudo-op gives the name (xxxxxx) of the module that you wish to edit. It is the module name as it appears in the library, up to six Radix-50 characters. Once you give a name, that module is loaded. Editing continues with this module unless a new .MODULE pseudo-op is given. You do not have to edit modules in the same order that they reside in the master library (first input file). However, each module may be named only once within an edit.

.ASSOCIATED +edit1,-edit2,+edit3,+edit4..... - This pseudo-op gives information on which other related edits must be present in the specified module. The edit names here are the same as those designated under the .EDIT pseudo-op. The "+" indicates that the edit is required. The "-" indicates an edit that conflicts with the current edit; you cannot have both edits active simultaneously. You receive notification if the specified module does not contain the correct combination of associated edits. The default is "+" if no sign precedes the edit name. Information you supply with the .ASSOCIATED pseudo-op must precede any information supplied in the .FIX file by the .INSERT, .REMOVE, or .REINSERT pseudo-ops.

.VERSION nnnxx(nnnnnn)-g - This pseudo-op allows you to specify the version number of the edit. The version is in standard TOPS-20 version format. The nnn designates the major version number, which consists of three octal digits maximum. The xx designates the minor version, which consists of two letters maximum. The (nnnnnn) designates the edit number, and can be up to six octal digits. The g designates the code for the group that last edited the module, and it is one octal digit maximum. All fields are optional.

.ALTER location,<new value>,<original value> - This pseudo-op changes the contents of a specific location. All code is written in angle brackets, < >. The original value at the specified location is replaced by a new value. The first argument, location, is where you wish to place the new value. Once you

enter the new value, it is evaluated and placed into the specified location. You can specify a third argument, <original value>, to check whether the actual original value differs from the expected original value. If it does, MAKLIB gives you an error message and the location is not altered.

.INSERT location,keyword:n,<original value> – This pseudo–op allows you to add code to a module. You precede the new code with a .INSERT pseudo–op and terminate the sequence with a .ENDI pseudo–op. MAKLIB assembles the code and adds it to the module in the output file.

The .INSERT pseudo–op takes three arguments. The first gives the location at which you want the new code executed. This argument can be a numeric or symbolic expression. This location is assumed to be relocatable, and may be in either the low or the high segment. The second argument is a keyword that specifies how the code is to be located with respect to this location. This keyword is one of the following:

BEFORE – Insert the new instructions so that they are executed before the instruction at the specified location.

AFTER – Insert the new instructions so that they are executed after the instruction at the specified location.

REPLACE – Delete one instruction from the existing code for each one included in the edit, beginning with the instruction at the location of the edit. Then, insert the new instructions so that they are executed in place of the deleted code.

REPLACE:n – Delete n instructions from the existing code, starting with the instruction at the edit address. Then, insert the new instructions so that they are executed in place of the deleted code. This applies no matter how many instructions you insert. The argument may be an expression, and is evaluated in the current radix.

You can specify a third argument, <original value>, to check whether an actual original value differs from the expected original value. If it does, MAKLIB gives you an error message and the editing does not take place. This argument gives the line of code at the location specified by the first argument. The code is written in angle brackets, and is evaluated. It must exactly match the code at the specified location. If the code at the location is a literal, you must give only the first word.

MAKLIB always inserts the new code at the end of the current segment. It replaces the referenced instruction with an unconditional jump to the new code. The format of code insertion appears in Section 6.5, Technical Notes.

Because MAKLIB does not physically insert the code at the location you specify, you need to consider a restriction when you use this facility.

MAKLIB constructs a patch that has the effect of a skipping instruction, and assumes that it follows an instruction that can potentially skip, at most, one instruction. If the intent of the patch does not fit these assumptions, it may not work correctly. Further, for REPLACE functions, MAKLIB assumes that program control can only pass to the first instruction of the deleted code.

For example consider the following code segment,

```
        JRST    FOO
BAR:    JFCL
        JFCL
FOO:    JFCL
        JFCL
        JFCL
```

and a MAKLIB patch to it:

```
.INSERT BAR,REPLACE:4<JFCL>
        JFCL
.ENDI
```

Note that the JRST FOO instruction at BAR–1 still causes the old code to be executed in spite of the patch.

As another example, consider the following:

```
LABEL:  OPENF%          ;A JSYS MONITOR CALL
          ERJMP ERROR
        JFCL
```

A patch using .INSERT cannot specify any combination of BEFORE or AFTER with location LABEL or LABEL+1. To do so would separate the JSYS and the ERJMP instructions, which must be consecutive to operate properly.

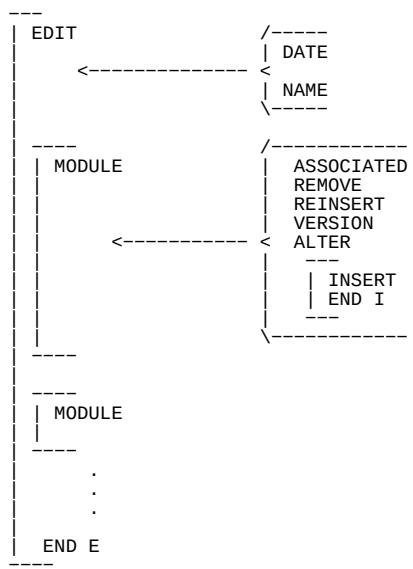
.REMOVE edit1,edit2,edit3.... – This pseudo–op deactivates the specified edits from the selected module. The original instructions displaced by the jumps to the edit area for each .INSERT are returned to that location. No changes are made to the symbol table. The arguments are the edits you wish to remove.

.REINSERT edit1,edit2,edit3... – This pseudo–op activates any edits you previously removed with the .REMOVE pseudo–op.

.ENDI – This pseudo–op marks the ending point of the code following the last .INSERT pseudo–op.

.ENDE – This pseudo–op marks the ending point of the complete edit. It also instructs MAKLIB to check for undefined labels or other invalid entries within the edit. Figure 6–9 illustrates the order that pseudo–ops appear in a .FIX file:

Figure 6–9: Order of Pseudo–ops in a .FIX File



The MAKLIB program has a one–pass assembler. Because of this, forward references to labels and expressions are restricted to simple addition and subtraction on the halfword boundary. References to undefined labels or symbols are valid where references to external symbols would

be valid in MACRO (with no polish fix–ups). Literals are treated as forward references, because the actual location of the literal is not known until the .INSERT pseudo–op ends. Defining a label inside of a literal is not valid. Finally, the value you place in the right–hand side of an assignment must not be forward or external.

It is not required that assignments be inside .INSERT in the .FIX file. It is required, however, that the .EDIT and .MODULE pseudo–ops precede any assignments, because these define new symbols in the symbol table. MAKLIB does not allow redefinitions of existing symbols, because it is impossible to backtrack references to a symbol in the relocatable binary file. So, any label or symbol you create with /FIX must be new to the program.

To simplify editing and to keep the appearance of binary edits as close as possible to the source level, the following pseudo–ops are implemented in the MAKLIB .FIX file assembler and operate as they do in MACRO:

ASCII	ASCIZ	BLOCK
BYTE	COMMENT	DEC
EXP	IOWD	OCT
POINT	PURGE	RADIX
RADIX50	REMARK	SIXBIT
SQUOZE	SUBTTL	TITLE
XWD		

NOTE

The pseudo–ops BYTE, DEC, OCT, and EXP are limited to a maximum generation of one word of data.

All MACRO operators and qualifiers are available except ^F. MAKLIB also supports the following MACRO pseudo–ops for writing conditional code:

IFN	IFG	IFDEF
IFE	IFLE	IFNDEF
IFL	IFGE	

You may follow symbols with ## (double pound sign) to indicate that they are EXTERNAL quantities. However, if the symbol is defined as EXTERNAL (already in the symbol table), you do not have to use ##. It is not necessary to follow undefined symbol names with # (single pound sign), since it is assumed that any undefined symbol is a forward reference. If a symbol name is already assigned and followed by the #, you receive an error message (see Section 6.5, MAKLIB Messages). You may define labels as internal (available to other programs) if they are followed by :: (double colon). Entry points may not be defined. The full facilities available in MACRO for combinations of DDT suppression and internal declaration are available for both labels and assignments.

Command String Requirements −

Files:	A master library (first input file) and an output file are required in the command string. The .FIX file is the required transaction file.
Default file type:	.REL for both the output file and the master library. The default for the transaction file is .FIX.
Arguments:	None.

The editing command string accepts two switches. They are /FIX and /WHO.

/FIX − FIX Switch

This switch makes changes to the actual code and symbol table of a module. It appears with the transaction (.FIX) file spec.

/WHO:xxx − WHO Switch

This switch is optional and you use it only with /FIX. You can enter it in either the master library or the transaction (.FIX) file spec. The argument to /WHO can be up to three characters (xxx). These are usually the initials of the person using MAKLIB at the time the edit is installed. If you include this information, it appears in the TRACE block of all new edits (in the last affected field and in the last installed field). If any of these edits change the status of an existing edit (such as .REMOVE or .REINSERT), this information is entered in the last affected field of the TRACE block of the affected edit. If you use /WHO without /FIX, MAKLIB ignores it in the command string.

In order to edit a library with a .FIX file, you can use the following command string. In this example, you use the .FIX file FIX1.FIX to edit the library OLDLIB.REL, and create an updated library NEWLIB.REL.

```
@MAKLIB<RET>
*NEWLIB=OLDLIB,FIX1/FIX/WHO:SFA<RET>
*
```

The following are sample .FIX files. This first example illustrates the use of the .INSERT and .VERSION pseudo−ops. One module in the library is modified.

```
.EDIT 341
.NAME HAS
.MODULE GLOB..
.VERSION 4C(341)
```



```
.INSERT START,AFTER,<RESET>
MOVE       P,[IOWD PDLEN,PDLIST]
.ENDI
.INSERT    LOOP+3,BEFORE,<JRST LOOP>
        PUSHJ      P,NEXTCR
.ENDI
.ENDE
```

The following edit illustrates the use of .ALTER pseudo−op to change the value of a table entry. Location RAD50+46 is changed from a "." to a "$". In addition, this edit uses the .ASSOCIATED pseudo−op to specify that this edit also requires edit 343 to the module TABLES.

```
.EDIT 344
.NAME HAS
.MODULE TABLES
.ASSOCIATED 343
.ALTER RAD50+46,<"$">,<".">
.ENDE
```

This edit uses the .REMOVE pseudo−op to deactivate edit 345 in the module FSORT. As a result of this operation, any code that was changed by edit 345 will be restored to its previous state.

```
.EDIT 346
.NAME HAS
.MODULE FSORT
.REMOVE 345
.ENDE
```

6.3 MAKLIB SWITCH OPTIONS

Table 6−1 is a reference table of all MAKLIB program switches. These switches are listed alphabetically. The function that each switch performs appears beside it in the table.

Table 6−1: MAKLIB Switches

Switch	Function
/APPEND	Adds new modules to the end of an existing library.

/DELETE	Removes one or modules from an existing library.
/EXIT	Terminates MAKLIB and returns you to TOPS-20 command level.
/EXTRACT	Produces an output file that is a subset of modules in the master library.
/FIX	Makes changes to the actual code and symbol table of a module.
/INDEX	Produces an output file identical to the master library except with INDEX blocks inserted in the file.
/INSERT	Inserts new modules into the master library.
/LIST	Lists the names of the modules that are contained in the master library.
/LOAD	Shows additional loading instructions that are embedded within the library in either REQUEST, REQUIRE, or ASCII text blocks.
/MASTER	Identifies files within the master library that correspond to those in the transaction file being used to effect the update.
/NOLOCALS	Produces an output file which is the master library with all local symbols deleted from the file symbol blocks.
/POINTS	Lists all entry points in the specified library.
/REPLACE	Replaces modules in the master library with those specified in the transaction file.
/TRACE	Lists all the edits made to a library.
/WHO	Specifies the initials of the person using MAKLIB when an edit is installed.

6.4 MAKLIB MESSAGES

The MAKLIB program issues two types of messages: fatal errors and warning messages. Fatal errors are preceded by a question mark (?) and cause the current command to be aborted. Warning messages are preceded by a percent sign (%) and indicate that the command will be completed, but the operation may not have been performed as you

intended.

All messages are typed on your terminal. They begin with a six character code that identifies the error. This is followed by a short description of the problem.

MAKLIB uses the command scanner routines SCAN and WILD. The following list of messages contains the most common messages that these routines produce. These messages begin with SCN or WLD.

Some of the messages contain information that is dependent on the exact command string, switch, or file you wish to process. The key to these message variables follows:

[edit]	The name of a specific edit.
[file]	A file spec.
[label]	The name of the label which caused the error.
[location]	The location where the error was detected. This is expressed as either a symbolic address or as a line number in the .FIX file.
[module]	The name of a specific module.
[pseudo-op]	A specific pseudo-op.
[statement]	A specific statement related to or causing the error.
[status]	A specific numeric file status code.
[switch]	A specific MAKLIB command switch.
[symbol]	A symbol. (Refer to the TOPS-20 MACRO ASSEMBLER Reference Manual for an exact definition.)
[type]	A REL Block type.
[value]	A specific value.

All MAKLIB messages are listed here alphabetically by the six-character code. A suggested user response is provided for each fatal error and those warning messages that require correction.

?MKLAAC .ASSOCIATED seen after .INSERT,.REMOVE or .REINSERT in edit [edit]

Description: In the indicated .EDIT, the .ASSOCIATED pseudo-op

is out of order.

Suggested User Response: Move the .ASSOCIATED pseudo-op so that it appears before any .INSERTs, .REMOVEs, or .REINSERTs.

?MKLAAL .ALTER address is not in current module in edit [edit]

Description: In your edit, you used a .ALTER pseudo-op to change the contents of an address. However, you gave an address that does not exist in the specified module.

Suggested User Response: Change the .ALTER pseudo-op so that it specifies a legal address.

%MKLAFI Arguments to /FIX switch are ignored

Description: In the command string, you gave arguments on /FIX. There are no defined arguments for /FIX, so MAKLIB ignores any that you specify.

%MKLAMI Assignment to [symbol] with no module selected was ignored:
[statement]

Description: In a .FIX file you assigned a value to a symbol, but you did not specify a module.

Suggested User Response: Change the .FIX file so that the assignment statement occurs after the .MODULE pseudo-op. This specifies which symbol belongs with a particular module.

?MKLANA Asterisk not allowed as output file spec

Description: In the command string you gave an output file name that included a wildcard character.

Suggested User Response: Since wildcard characters are illegal for the output file name, reissue the command with an explicit output file name.

?MKLASG FORWARD/EXTERNAL assignment to [symbol] at [location] (Edit [edit])
[statement]

Description: In the specified edit, you assigned a forward or external reference.

Suggested User Response: MAKLIB does not support forward or external references. Change the assignment statement.

?MKLBAM BEFORE, AFTER or REPLACE missing from .INSERT in edit [edit]

Description: You typed an incomplete .INSERT pseudo-op in a .FIX

file.

Suggested User Response: Include the required second argument for .INSERT indicating how the code should be inserted.

?MKLBDA Bad .DATE argument for edit [edit]

Description: In a .FIX file you specified a date that was not in a recognizable format for the .DATE pseudo-op.

Suggested User Response: Specify the date in the form dd-mon-yy, such as 7-JUL-77.

?MKLBNI Binary patching tool not included in MAKLIB

Description: You attempted to use a .FIX file with a version of MAKLIB that does not support .FIX files.

Suggested User Response: Rebuild MAKLIB from the MACRO source files and set feature switch FTBPT non-zero.

?MKLCDM Existing code does not match original code

Description: The original value you specified with either the .ALTER or .INSERT pseudo-op does not match the actual code at the location you were attempting to change.

Suggested User Response: First, determine if this is an error in the original value field of the pseudo-op. If this is actually the value you expected at that location, this error could indicate that you are trying to edit a different version of the library file.

%MKLCII Code generated outside of range of .INSERT was ignored:
[statement]

Description: You entered a .FIX file with a sequence of code that was not included between .INSERT and .ENDI pseudo-ops.

Suggested User Response: Change the .FIX file so that the instructions are within the range of an .INSERT. They can then be inserted in the module.

%MKLCNF Insertion of edit [edit1] by edit [edit2] conflicts with edit [edit3]

Description: Edit2 contains a .REINSERT pseudo-op for edit1. This conflicts with the .ASSOCIATED list in edit3 which is currently an active edit for this module. This is only a warning message, and the actual .REINSERT does take place. You can verify the current status of any edits with /TRACE.

%MKLCNF Removal of edit [edit1] by edit [edit2] conflicts with edit [edit3]

Description: Edit2 contains a .REMOVE pseudo-op for edit1. This conflicts with the .ASSOCIATED list of edit3 which is currently an active edit for this module. This is only a warning message, and the actual .REMOVE does take place. To verify the status of any edits, use /TRACE.

?MKLCSR Command switch is required

Description: You typed an incomplete command string. Supply additional information with file switches.

Suggested User Response: Reissue the command string including the necessary switches on either the master library or transaction files.

?MKLEEI .ENDE seen before .ENDI in edit

Description: Your .FIX file is missing an .ENDI pseudo-op, or the .ENDI pseudo-op is out of order.

Suggested User Response: Make sure that each .INSERT pseudo-op is matched with an .ENDI pseudo-op to identify the code to be inserted. The .ENDE pseudo-op indicates the end of an edit. Therefore, it is always the last statement of an edit.

?MKLEFF End of file found before END block in module

Description: The input master file is bad. Although part of the file is readable, the END block (REL Block type 5) is missing for the particular module. This indicates that the file is damaged.

Suggested User Response: Re-create the file or restore it from a backup copy.

%MKLEMA Entire MASTER file will be appended

Description: For the current command string, the entire MASTER file is being appended to the output file even though you specified an individual module.

?MKLEPM .EDIT pseudo-op is missing from FIX file

Description: The .FIX file is incorrect.

Suggested User Response: Change the .FIX file so that each edit begins with the .EDIT pseudo-op and ends with an .ENDE pseudo-op.

?MKLERI Edit [edit] tried to .REMOVE or .REINSERT itself

Description: The .FIX file contains an edit that has a .REMOVE or .REINSERT pseudo-op referencing itself. These pseudo-ops must reference edits that have previously been applied to the module.

Suggested User Response: Change the .FIX file so that the .REMOVE or .REINSERT pseudo-op does not reference the current edit.

?MKLETC MACRO code expression too complex at [location] (Edit [edit])

Description: The expression at the indicated location is too complex for MAKLIB to evaluate.

Suggested User Response: Try to break the expression into several simpler expressions.

?MKLETL ENTRY block is too large to read in for module [module]

Description: MAKLIB does not have enough space allocated to process all the entry points for the specified module.

Suggested User Response: Try to reduce the number of entry points in the module.

?MKLFF4 Cannot apply FIX to F40 produced REL file

Description: You attempted to apply a .FIX file to a .REL file produced by the F40 FORTRAN compiler. This operation is not supported by MAKLIB. F40 generates .REL files that MAKLIB cannot edit from .FIX files.

?MKLFNI Qualifier ^F not implemented

Description: MAKLIB does not support ^F for entering a fixed-point decimal number.

Suggested User Response: Enter the value in an alternate form.

?MKLFSI File status error on input [status] for file [file]

Description: An error occurred while MAKLIB was reading the specified file. The status value that appears is described in this manual in Table 5-3.

Suggested User Response: This could indicate a more global problem with the system. Refer to Table 5-3 in this manual to determine the specific problem with the named file.

?MKLFSO File status error on output [status] for file [file]

Description: An error occurred while MAKLIB was writing the specified file. The status value that appears is described in

this manual in Table 5–3.

Suggested User Response: This could indicate a more global problem with the system. Refer to Table 5–3 in this manual to determine the specific problem with the named file.

?MKLIAA Illegal address in .ALTER in edit [edit]

Description: In the specified edit, you used a .ALTER pseudo–op to change the contents of an address. However, you gave an illegal value for an address.

Suggested User Response: Change the .ALTER pseudo–op so that is specifies a legal address.

?MKLIAI Illegal address in .INSERT in edit [edit]

Description: You specified an illegal address in the location field for a .INSERT pseudo–op.

Suggested User Response: See if the address you specified is the address of a literal, external, or undefined address. These are not allowed.

?MKLIAL .INSERT address is not in current module in edit [edit]

Description: You specified an address in the location field for a .INSERT pseudo–op that is not in the specified module.

Suggested User Response: First determine if the address you specified is the one you intended. Then check the .FIX file to verify that you placed the .INSERT sequences after the correct .MODULE pseudo–ops.

?MKLIBT Illegal block type ([type]) was seen in file [file]

Description: MAKLIB encountered an illegal REL Block while reading the indicated file.

Suggested User Response: The .REL file being read may be damaged. Re–create the .REL file and try again.

?MKLIED Internal error detected at [location] in MAKLIB

Description: This indicates that MAKLIB cannot perform the operation you were attempting.

Suggested User Response: Contact your Software Specialist or send a Software Performance Report (SPR) to DIGITAL.

?MKLIIA .INSERT pseudo–op illegal inside range of .INSERT [edit]

Description: In the specified edit, you nested .INSERT psuedo–ops.

Suggested User Response: These pseudo–ops cannot be nested. Use the .ENDI pseudo–op to end the first .INSERT sequence before you begin another.

?MKLIII Illegal pseudo–op in range of .INSERT: [value] at [location] (edit[edit])

Description: In your .FIX file you used a MACRO pseudo–op which is illegal in a .INSERT sequence.

Suggested User Response: Verify that this is the correct MACRO pseudo–op for the operation you wish to perform.

?MKLILS Illegal use of long string or BLOCK in .ALTER at [location] (edit[edit])

Description: You tried to use a multi–word value with the .ALTER pseudo–op.

Suggested User Response: The .ALTER pseudo–op is for changing single word values only. This applies to the original value as well as the new value. Use a separate .ALTER pseudo–op for each word to be altered.

?MKLIPM .ENDI seen without .INSERT in edit [edit]

Description: The .FIX file is incorrect.

Suggested User Response: Change the .FIX file so that each set of instructions you wish to insert begins with an .INSERT pseudo–op and ends with a .ENDI pseudo–op.

?MKLIRF Illegal relocation in FORWARD reference to [symbol] in edit [edit]

Description: MAKLIB could not process the reference to the specified relocatable symbol.

?MKLIRM /INSERT requires at least one /MASTER specification

Description: The command string is incomplete.

Suggested User Response: When you specify /INSERT on a transaction file, you must include /MASTER with the master file. You must supply a specification on /MASTER for every module you wish to insert.

?MKLISM /INSERT,/REPLACE and /FIX are illegal switches on MASTER

Description: The command string is incorrect.

Suggested User Response: Reissue the command string putting /INSERT, /REPLACE, or /FIX on the transaction file. When you use /INSERT or /REPLACE, you must include /MASTER on the master file.

?MKLIST interim symbol table overflowed, Code too complex in edit [edit]

Description: Your specified edit contains code that has too many symbols for MAKLIB to process.

Suggested User Response: Try to break the single edit into several edits, each having a fewer number of symbols.

?MKLITS Insufficient TRACE block storage in edit [edit]

Description: The specified edit exhausted all allocated storage for TRACE block information.

Suggested User Response: Try to break the edit into several edits, thus reducing the amount of storage MAKLIB needs for TRACE blocks at any one time. For each edit, MAKLIB creates a TRACE block for each module that is changed.

?MKLIUN Illegal to have null address in .INSERT in edit [edit]

Description: In the specified edit, the location field of a .INSERT pseudo-op is null.

Suggested User Response: The location field cannot be null. It specifies the location where you want to insert the patch code.

%MKLLII Label outside of .INSERT was ignored: [label]

Description: In a .FIX file you specified a label outside the range of a .INSERT.

Suggested User Response: Change the .FIX file so that the label occurs within the range of a .INSERT-.ENDI pair. This specifies where you insert the new label in the program.

?MKLLTL MACRO code line is too long at [location] (Edit[edit])

Description: In the specified edit, a line of code in the .FIX file is too long for MAKLIB to process.

Suggested User Response: Try to reduce the length of the line by breaking it into several shorter lines.

?MKLMCA Pseudo-operator argument error at [location] (Edit[edit])

Description: You gave an illegal pseudo-op argument. The values you can use depend on the particular pseudo-op.

Suggested User Response: Supply a legal value for that pseudo-op.

?MKLMCB MASTER device must be capable of binary IO

Description: The input master file in your command string is not on a device that is capable of binary input/output. MAKLIB performs binary input/output on the master file.

Suggested User Response: Keep the files you plan to use as MAKLIB master files on devices capable of binary input/output.

?MKLMCE Command error

Description: MAKLIB was not able to recognize a valid command from the command string that you typed. This error could occur if you improperly formatted the command string or if you used a non-unique abbreviation to specify a switch.

Suggested User Response: Retype the command string in the correct format: Destination File Spec=Source File Spec1/Switches,Source File Spec2/Switches,...Source File Specn/Switches

?MKLMCF Illegal forward or external reference at [location] (Edit [edit])

Description: At the specified location you made a reference to an undefined symbol in this module. It could be a forward reference to a new symbol or it could be an external reference. MAKLIB cannot process forward references to new symbols. This error could also occur if you attempted to reference an existing symbol and supplied the wrong symbol name.

Suggested User Response: Change your forward or external references to legal ones.

?MKLMCM Attempt to redefine value of symbol [symbol] at [location] (Edit [edit])

Description: You tried to redefine the value of an existing symbol. MAKLIB does not support this.

Suggested User Response: You cannot use MAKLIB to change the value of an existing symbol. If this is not what you intended to do, you may be using a symbol that was already defined. Try another symbol.

?MKLMCN MACRO code numeric error at [location] (Edit [edit])

Description: You supplied a numeric argument at the specified location that is illegal in that context.

Suggested User Response: Supply a legal numeric value.

?MKLMCQ MACRO code is questionable at [location] (Edit [edit])

Description: MAKLIB cannot process the code at the specified location.

Suggested User Response: Check the code for legal MACRO syntax.

?MKLMCR MACRO code relocation error at [location] (Edit [edit])

Description: The value at the specified location contains a half-word that is neither absolute nor simply relocatable. Expressions that would require polish, such as A + A where A is relocatable, cannot be handled in MAKLIB.

Suggested User Response: Give absolute or simple relocatable values only.

?MKLMCU Undefined symbol: [symbol] at [location] (Edit [edit])

Description: The symbol shown is not defined.

Suggested User Response: Define the symbol. If you are trying to use a symbol that you think is already defined, check for the correct symbol name.

?MKLMCW BYTE,EXP,DEC,or OCT more than one word at [location] (Edit [edit])

Description: You used an expression that generates a multi-word value. In this context, MAKLIB supports only a single word value.

Suggested User Response: Try to break the expression into simpler expressions, each generating a single word value.

?MKLMEP Missing .ENDE for edit [edit]

Description: The specified edit in your .FIX file is incorrect.

Suggested User Response: Change the .FIX file so that each edit begins with the .EDIT pseudo-op and ends with an .ENDE pseudo-op.

?MKLMFR Master file rejected by condition

Description: The master file did not meet the condition you specified; hence processing cannot continue.

?MKLMHE Module already has an edit [edit]

Description: You attempted to apply an edit to a module that already has an edit with a similar identifier.

Suggested User Response: Each edit to a module must have a unique identifier. You may have already applied this edit to the module.

?MKLMKM [Pseudo-op] pseudo-op in edit [edit] without preceding .MODULE

Description: In the specified edit, you gave a pseudo-op out of order. It applies to a specific module that you must specify with a preceding .MODULE pseudo-op.

Suggested User Response: Make certain that a .MODULE pseudo-op precedes those pseudo-ops that refer to a specific module.

?MKLMNF Module [module] was not found in file [file]

Description: MAKLIB cannot find the specified module in this file. If you are trying to apply a .FIX to a master library, you may have specified a non-existent module name on a /MODULE pseudo-op. You can also receive this error when you perform an operation that does not involve .FIX files. You may have specified several module names and supplied them out of order. The module in question may be in the file. But since MAKLIB processes files in a sequential manner, it will not find all modules.

Suggested User Response: Use /LIST to see if the specified module is really in the library. If not, you cannot perform this function.

%MKLMNI /MASTER module names are ignored when patching

Description: For editing, the correct way to specify module names is with the .MODULE pseudo-op in the .FIX file. /MASTER is not required for this type of command. MAKLIB ignores it.

?MKLMTF /MASTER switch cannot be used on transaction file

Description: The command string is incorrect.

Suggested User Response: Retype the command string with the switch in the correct place. Include only /MASTER on the master file.

?MKLNEA Not enough arguments specified

Description: The command string is incomplete. This error usually means that you omitted a required file spec.

Suggested User Response: Retype the command string in the correct format: Destination File Spec=Source File Spec1/Switches,Source File Spec2/Switches...Source File Specn/Switches

?MKLNEC Not enough core is available

Description: MAKLIB cannot obtain enough core to process this command.

Suggested User Response: Break the .REL files into smaller numbers of modules, or break large modules into smaller modules.

?MKLNEI Null argument to .EDIT is illegal

Description: You did not specify an edit identifier on the .EDIT pseudo-op.

Suggested User Response: Each .EDIT pseudo-op requires an argument (up to 6 characters) that identifies the edit.

%MKLNIO Output file [file] will not be indexed

Description: The output file will not include an index. To add an index to the file, issue a separate MAKLIB command with /INDEX.

?MKLNMS Null specification to .MODULE [edit]

Description: In your edit, you included a .MODULE pseudo-op without the module name.

Suggested User Response: Correct the .FIX file by supplying a module name on each .MODULE pseudo-op.

?MKLNPC No program code was found for module in edit [edit]

Description: You have specified a non-existent module in the master file.

?MKLNPS No program names were specified for file [file]

Description: You tried to manipulate some of the modules in the specified file, but you did not supply any module names.

Suggested User Response: In order to /EXTRACT or /DELETE modules from a file, supply the module name on the file switch.

?MKLNRP Not a recognized position switch: [value]

Description: You gave an illegal position indicator on an .INSERT pseudo-op.

Suggested User Response: Use one of the three legal position indicators: BEFORE, AFTER, or REPLACE.

?MKLNTM Not enough TRANSACTION modules were specified

Description: You did not supply enough replacement module names to perform the /REPLACE operation.

Suggested User Response: When you give the MAKLIB command string, make certain that there is a corresponding replacement module for each module you intend to replace in the master library.

?MKLODD Output device must be DISK or DECTAPE

Description: In your MAKLIB command, you specified an illegal output library device.

Suggested User Response: Retype the command string and use disk for the output device of the library file.

?MKLPEF Premature end-of-file during edit [edit] in file [file]

Description: While processing the indicated edit, MAKLIB encountered an unexpected end of file in the .FIX file. This error usually occurs when you omit the .ENDE pseudo-op.

Suggested User Response: Check the .FIX file for errors, and look especially for .ENDI and .ENDE pseudo-ops.

%MKLPEP Precluded edit [edit] is present in module

Description: The module you are editing contains an active edit that your current edit precludes with the .ASSOCIATED pseudo-op. MAKLIB still applies your edit.

%MKLPES Purging EXTERNAL symbol [symbol] may give bad REL file

Description: One of the symbols that this edit PURGEd from the symbol table was an EXTERNAL symbol. With this symbol removed, it may not be possible to LINK the file correctly.

?MKLRBF REQUEST or REQUIRE block is badly formatted

Description: In the master library being processed, MAKLIB encountered a REQUEST block (REL Block type 17) or REQUIRE block (REL Block type 16) that was not in the expected format.

Suggested User Response: The library file may be damaged. Try to rebuild it.

%MKLREM Required edit [edit] is missing from module

Description: The module you are editing is missing an active edit that your current edit requires with the .ASSOCIATED pseudo-op. MAKLIB still applies your current edit.

%MKLRER Required edit [edit] is inactive in module

Description: The module you are editing contains an inactive edit that your current edit requires with the .ASSOCIATED pseudo-op. MAKLIB still applies your current edit.

%MKLRIA Edit [edit] tried to .REINSERT already active edit

Description: Your current edit contains a .REINSERT pseudo-op that attempts to activate an edit that is already active.

%MKLRIE Edit [edit] tried to .REMOVE already inactive edit

Description: Your current edit contains a .REMOVE pseudo-op that attempts to deactivate an edit that was previously deactivated.

%MKLRIN Edit [edit] tried to .REINSERT non-existent edit

Description: Your current edit contains a .REINSERT pseudo-op that attempts to activate a non-existent edit.

%MKLRNE Edit [edit] tried to .REMOVE non-existent edit

Description: Your current edit contains a .REMOVE pseudo-op that attempts to deactivate a non-existent edit.

?MKLRTL .INSERT's REPLACE argument of [value] too large for module [module]

Description: On a .INSERT pseudo-op you specified a number of instructions to be replaced. This number exceeds the number of instructions in the module beyond the starting replacement address.

Suggested User Response: Correct the argument to the REPLACE keyword on the .INSERT pseudo-op.

?MKLSCE Storage for patch code was exhausted in edit [edit]

Description: MAKLIB allocates a fixed amount of space for processing the new code inserted from a .FIX file. Your .FIX file contains more code than MAKLIB can process in the allocated space.

Suggested User Response: Try to break your .FIX file into several .FIX files with fewer lines of new code.

?MKLSIO Switches are illegal on output

Description: The command string is incorrect. Switches are not recognized on the output file.

Suggested User Response: Retype the command string including any necessary switches with the appropriate input files.

%MKLSNF Symbols not found for module

Description: There are no symbols in the master file for the module that you wish to edit.

?MKLSSE Storage for patch symbols was exhausted during edit [edit]

Description: MAKLIB allocates a fixed amount of storage for processing the new symbols from .FIX files. Your .FIX file contains more symbols than MAKLIB can process in the allocated space.

Suggested User Response: Try to break your .FIX file into several .FIX files with fewer symbols.

%MKLTBF TRACE block is badly formatted in module

Description: One of the TRACE blocks (REL Block type 1060) for this module is not in the expected format. The .REL file may be damaged. Since the loader ignores trace blocks when reading a file, you may still be able to load from this .REL file.

%MKLTFI Transaction file ignored

Description: You included a transaction file in the command string to create an indexed library.

Suggested User Response: You must create the indexed library as a single operation. If there are several operations you must perform on the library, the command to index the library should be the last command that you issue.

?MKLTFR All transaction files rejected by condition.

Description: Processing cannot continue because the transaction files did not meet the condition you specified (as size, creation date, etc.).

?MKLTMN Too many module names . . . stopped at [module]

Description: The command string is too long.

Suggested User Response: Retype the command string as several shorter commands. It is unlikely that this message will appear, since MAKLIB now allows up to 100 switch arguments for each command string.

?MKLTMS Too many switches

Description: You included too many switches on a file spec. You specified some of the switches in an illegal combination.

Suggested User Response: Retype the command string in a correct format.

?MKLUDF Module [module] in edit [edit] contains undefined symbol(s)

Description: Your current edit contains undefined symbols in the indicated module.

Suggested User Response: Make certain that all new symbols introduced in your .FIX file have values associated with them.

?MKLWIO Wild cards illegal for output file specification

Description: You gave an output file name in the command string that included wildcard characters (either * or ?).

Suggested User Response: Since wildcard characters are illegal for the output file name, retype the command string with an explicit output file name.

?SCNSVR Switch value required on [switch]

Description: The specified switch requires a value.

Suggested User Response: Retype the command string and supply a value for the switch. The format for providing a value is /switch:value or /switch:(value1,value2).

?WLDLKE Protection failure file [file]

Description: The specified file is protected. You do not have the privileges to access it.

Suggested User Response: If this is the output file, you need privileges to create a file in the directory you specified. If it is an input file, you need privileges to read that file from that particular directory.

?WLDLKE Non-existent file [file]

Description: The specified file does not exist.

Suggested User Response: This error usually occurs when the name of an input file is incorrect. Retype the command string with the correct input file name.

6.5 TECHNICAL NOTES

The following is supplementary information related to editing libraries with MAKLIB. Section 6.5.1 describes TRACE blocks (REL Block type 1060). Section 6.5.2 contains the format for code insertion in .FIX files.

6.5.1 Format of TRACE Block Data (REL Block Type 1060)

MAKLIB uses the TRACE block to include, in the .REL file, information for verifying and changing the patch status of a program. The format of the TRACE block follows.

The first part of the TRACE block is the static area. This area appears in each module affected by the particular edit. The static areas give information common to all modules affected by an edit and the variable gives the changing data on the particular edit as it goes from module to module.

```
TB$HED                      REL Block Type       Length of Block

TB$EDT                       SIXBIT Edit Name   (Up to 6 chars.)

TB$STA                        -1 If Active      Who  Last Affected

TB$MAK       Who Created          Date (15 BIT)

TB$INS                        Who Installed       Date (15 BIT)

TB$FUT            Reserved for Future Use

TB$LEN     # of Assoc. Edits   # of PCO Groups
```

The static area, which repeats in each module, is followed by a variable area. The variable area consists of two parts. The first gives data on the associated edit status for this module, and the second gives the actual program change orders (PCO's). The length of each of these areas appears in the static area of the TRACE block.

For each associated edit, the following group appears:

```
TB$AEN        SIXBIT Edit  Name of Assoc. Edit

TB$AES     X     Reserved for Future Use       0B0 If can't be present
                                                1B0 If must be present
```

After the associated edit groups appear (if there are any), the PCO

groups for that module appear. There are currently three types of program change groups: INSERT, REMOVE, and REINSERT. They can appear in any order and the total number is variable.

```
                    INSERT PCO:

TB$PCO     PCO Type Code (1)            Length of Group

TB$DAT      Instrs. INSERTed            Addr. of INSERT

TB$PAT     New Addr. of Original Code   Addr. of Patch Code


                    REMOVE PCO:

TB$PCO     PCO Type Code (2)   Length of Group

TB$REN              SIXBIT Edit Name


                    RE-INSERT PCO:

TB$PCO     PCO Type Code (3)   Length of Group

TB$RIN              SIXBIT Edit Name


                    ALTER PCO:

TB$PCO     PCO Type Code (4)            Length of Group

TB$DAT          Unused                  Addr. of ALTER

TB$PAT      New Addr. of Original Code      Unused
```

6.5.2 Format of Code Insertion

The four formats of code insertion in a .FIX file are shown here. Notice that, in all cases, the patch ends with exactly two JUMPA instructions. Thus, the last instruction of the patch can at most skip a single instruction and still return control to the original code.

To INSERT any instruction or series of instructions (code) BEFORE a location, use this format:

```
     .INSERT location, BEFORE, <original instruction>

LOCATION:       JUMPA   %PATCH

%PATCH:                 First Patch Instruction

                       Second Patch Instruction

                                 .
                                 .
                                 .

                        Last Patch Instruction

                      Original Patch Instruction

                          JUMPA 1, LOCATION+1

                          JUMPA 2, LOCATION +2

                            Any "Literals"
```


The actual label created at the location of the patched-in code is of the form:

```
        "%"<edit-name><edit-part>
```

where the edit-part is from "A" to "Z", incremented for each .INSERT in the edit.

To INSERT any instruction or series of instructions (code) AFTER a location, use this format:

```
     .INSERT location, AFTER, <original instruction>

LOCATION:       JUMPA   %PATCH

%PATCH:                  Original Instruction

                        First Patch Instruction

                       Second Patch Instruction

                                 .
                                 .
                                 .

                        Last Patch Instruction

                         JUMPA 1, LOCATION+1

                         JUMPA 2, LOCATION+2

                           Any "Literals"
```

To REPLACE a single instruction, use the format:

```
     .INSERT location, REPLACE, <original instruction>

LOCATION:       JUMPA   %PATCH

                       Original Instruction

%PATCH:              First Patch Instruction

                    Second Patch Instruction

                              .
                              .
                              .

                  nTH (Last) Patch Instruction

                      JUMPA 1, LOCATION+n

                     JUMPA 2, LOCATION+n+1

                        Any "Literals"
```

If you do not insert instructions (n=0), then the return is to LOCATION+1 and LOCATION+2.

To REPLACE more than one instruction at a location, use the format:

```
     .INSERT location REPLACE:m <original instruction>

LOCATION:       JUMPA   %PATCH

                       Original Instruction

%PATCH:              First Patch Instruction

                    Second Patch Instruction

                              .
                              .
                              .

                    Last Instruction of Patch

                      JUMPA 1, LOCATION+m

                     JUMPA 2, LOCATION+m+1

                         Any "Literals"
```

If you do not specify m, or it is zero, the effect is the same as the

REPLACE keyword without an argument. In other words, one word is skipped over on return for every one inserted.

CHAPTER 7

THE DUMPER PROGRAM

7.1 INTRODUCTION

DUMPER is a TOPS−20 utility program used to save files on magnetic tape, and later to restore any or all of these files to a specified directory on disk. DUMPER is available to both nonprivileged and privileged users. (Throughout this chapter, the term "operator" is occasionally used instead of privileged user.) As a nonprivileged user, you can use DUMPER to save your own files or any files to which you have access, by transferring them from the disk to a 9−track magnetic tape (DUMPER does not function on a 7−track magnetic tape drive), and later restoring them to disk. As a privileged user, you can use DUMPER to:

- o save other users' files and directory information on tape
- o back up the system files (copy all files onto tape for an indefinite period of time)
- o archive users' files (copy files marked for storage onto tape and delete them from disk)
- o migrate users' files (copy files onto tape and delete them from the disk) to create added disk space
- o restore other users' files
- o retrieve previously archived or migrated files.

If your installation is using a Version 6−based TOPS−20 monitor, DUMPER supports encrypted passwords and project−programmer numbers (PPN). Earlier versions of DUMPER, however, do not support the password encryption and PPN features. Therefore, use extreme caution when changing versions of DUMPER and the TOPS−20 monitor. See Section 7.6 for information on these features.

The length of time files remain on tape is determined by each installation. In general, however, archived files are kept longer

than backup files. Archiving is voluntary on the part of the nonprivileged user; migrating is not.

Sections 7.2 through 7.4 describe the functions of DUMPER, the use of tapes, and the procedure to run DUMPER. Sections 7.5 and 7.6 describe the commands available to both the nonprivileged and privileged user. Examples are provided to illustrate the use of each command. Section 7.7 contains an alphabetical list of all commands. Section 7.8 contains an alphabetical list of all error messages. Before continuing, you should know the following terms:

JFN	Indicates a job file number, an octal number that represents a particular file.
saveset	Indicates a group of files on tape stored as the result of one SAVE command to DUMPER.
saveset name	Indicates a string of up to 200 alphanumeric characters used as the name for a saveset. The specified name is written in the saveset header on the tape.
tape set	Indicates a set of one or more volumes (reels) of tapes grouped under a single name. Each tape is distinguished by a unique identification of up to six alphanumeric characters.

NOTE

As mentioned at the beginning of this manual, it is assumed that you are familiar with TOPS−20 log−in procedures and the basic commands. To understand DUMPER, you should be particularly familiar with the following commands: MOUNT TAPE, DISMOUNT TAPE, DEFINE, ASSIGN, and INFORMATION. (Refer to the TOPS−20 Commands Reference Manual.)

7.2 FEATURES

The following is a list of DUMPER's most useful features.

- o With DUMPER, you can specify particular files to be transferred between disk and tape. For example, you can specify files using the standard TOPS−20 file specification format of dev:<dir>name.typ.gen and/or you can select files

based on the dates and times that the files were created, modified, or accessed. Other conditions can also be set; if the file meets all conditions, it is transferred. (Refer to Section 7.5.1.)

- You can save a set of files that exceeds one reel of magnetic tape. If all files specified cannot fit on one tape, DUMPER continues the operation on subsequent tapes (except in INTERCHANGE mode, which allows only one tape).

- As a privileged user, you can transfer files marked for archival by another user. This ensures that those files are saved for a period of time (determined by each installation) for future use or reference.

- As a privileged user, you can migrate files from disk to tape that have not been referenced within a specified period of time (as defined by each installation).

- As a means of verification, you can request that file names, directory names, and saveset names be printed during save and restore operations.

- DUMPER can read and write tapes written under older versions of DUMPER. Section 7.3 describes the use of tapes with previous versions of TOPS-20 DUMPER.

7.3 USING TAPES WITH AND WITHOUT TAPE DRIVE ALLOCATION

Before running DUMPER, you must have a tape mounted. Tape drive allocation allows you to request a tape to be mounted; TOPS-20 fulfills that request with any free drive. The absence of tape drive allocation requires that you assign a specific tape drive yourself. If you are using unlabeled tapes, tape drive allocation has no effect on the content of the tapes that DUMPER writes. If you are using labeled tapes, tape drive allocation is required. The label information on the tape then helps to identify the tape without operator intervention. Tape drive allocation also allows you to read tapes written under versions of DUMPER previous to Version 4 of TOPS-20. The difference between the presence or absence of tape drive allocation is in the method of mounting and dismounting tapes, and the fact that you cannot read or write labeled tapes without tape drive allocation. A description of each method follows.

NOTE

You can determine whether your installation is using tape drive allocation by typing the TOPS-20 command INFORMATION SYSTEM-STATUS.


```
@INFORMATION (ABOUT) SYSTEM-STATUS<RET>
 Operator is in attendance
 Remote logins allowed
 Local logins allowed
 Pseudo-terminal logins allowed
 ARPANET terminal logins are not allowed
 Console terminal login allowed
 Accounting is being done
 Account validation is enabled
 Tape-drive allocation is enabled
 Automatic file-retrieval-waits allowed
 Scheduler bias-control setting is 11
 Class scheduling by accounts enabled,
  windfall allocated, batch class 1
@
```

If your installation does not have tape drive allocation, use a tape as follows:

- Select a tape drive (for example MTA1:) and type the TOPS-20 command ASSIGN MTA1:.
- The system responds with MTA1: ASSIGNED.
- Mount the tape on the assigned drive.
- Run the DUMPER program to perform your tape operations. First identify the assigned tape by typing the DUMPER command TAPE MTA1: in response to the prompt, DUMPER>. If you omit the command, DUMPER looks for a device that has been defined as MTA-DUMPER:. To define the logical name MTA-DUMPER: as your assigned tape drive (and thereby save a step after entering DUMPER), perform the following steps:
  - Assign a tape drive.

    @ASSIGN MTA1:<RET>

  - Define that drive as MTA-DUMPER:

    @DEFINE MTA-DUMPER: MTA1:<RET>

  - Upon completion of tape operations, exit from DUMPER.
  - Type the TOPS-20 commands UNLOAD MTA1: and then DEASSIGN MTA1:. Remove the tape from the drive.

If your installation is using tape drive allocation, mount a tape as follows:

- Type the TOPS-20 command MOUNT TAPE name:. The name (for example, TAPE1:) is a logical name. (In the simplest case,

the logical name is identical to the volume name.) The system responds with a message indicating that your request is in queue. For example:

```
@MOUNT TAPE TAPE1:  /WRITE-ENABLED<RET>
[Mount Request TAPE1 Queued, Request-ID 174]
```

NOTE

By typing /WRITE-ENABLED as part of the MOUNT TAPE command, you can write onto the tape. If you do not specify this switch, you can do only read operations (i.e., print a list of file names or restore files to disk).

If your installation requires that the tape be mounted by an operator, the tape reel must have an external identification label with the specified name clearly visible to the operator.

NOTE

If your tape has never been initialized, be sure to tell the operator. The tape must be initialized before you can use it.

- Either you or the operator (depending upon the procedure at your installation) mounts the tape on any available drive. If the tape is an unlabeled tape, it must be identified to the system. (Refer to the TOPS-20 Operator's Guide.)
- The tape is then known to the system, and the system assigns a tape device number. You receive a message such as:

```
[Mount Request TAPE1 Queued, Request-ID 114]
[Tape set TAPE1, volume TAPE1 mounted]
[TAPE1: defined as MT0:]
```

- Run DUMPER to perform your tape operations.
- Upon completion of tape operations, exit from DUMPER.
- Type the TOPS-20 command DISMOUNT TAPE name: where the name is the same logical name used with the MOUNT TAPE command.
- The system responds with:

```
[TAPE dismounted, logical name TAPE1: deleted]
```

To assign the logical name MTA-DUMPER: before entering DUMPER, define the logical name using your tape name or the tape drive number you

were assigned:

```
@DEFINE MTA-DUMPER:  TAPE1:<RET>
```

or

```
@DEFINE MTA-DUMPER:  MTn:<RET>
```

DUMPER supports both unlabeled and TOPS-20 and ANSI labeled tapes. It is not necessary to specify which you are mounting. However, in case there is a duplication of set name or identification between labeled and unlabeled tapes, it is wise to be specific. This avoids confusion for the operator mounting your tapes. For example, you can append a label-type switch to the MOUNT TAPE command:

```
@MOUNT TAPE TAPE1:  /LABEL-TYPE:UNLABELED /WRITE-ENABLED<RET>
```

or

```
@MOUNT TAPE TAPE1:  /LABEL-TYPE:TOPS-20 /WRITE-ENABLED<RET>
```

If you think you need more than one tape, you can identify all the tapes in advance by adding another switch to the MOUNT TAPE command:

```
/VOLIDS:volid1,volid2,...,volidn
```

If you use /VOLIDS:, the logical name in the MOUNT TAPE command now refers to the entire tape set. The specifications volid1 through volidn (for example, DH33, DH44, DH55) identify each volume of tape within the set. Each volume identifier (volid) can be one to six alphanumeric characters.

When you type the MOUNT TAPE command, the system requests that the first volume of the set be mounted. When the operator has done this, you receive a message, such as:

```
[Tape set, TAPE1, volume DH33 mounted]
[TAPE1:  defined as MT0:]
```

If you wish to read the tape set, you must specify the volumes in the order in which they were written. You can begin with any volume in the set, but you cannot then specify a tape with a volid previous to the one with which you began. For example, if you wrote the tapes in the order DH33, DH44, DH55, you can request:

```
/VOLIDS:DH44,DH55
```

You cannot, however, request volumes DH44, DH33 nor omit a volid by requesting volumes DH33, DH55.

When the first volume is mounted, you are ready to run DUMPER.

NOTE

If your write operations require more than the specified number of tapes, DUMPER automatically

requests the operator to mount an additional tape. When the operation is complete, you can verify the additional tape in your set by typing the TOPS-20 command INFORMATION (ABOUT) VOLUMES name: before you type the DISMOUNT command (where name: is the name of the tape set). The system responds with the volid of each volume.

```
@INFORMATION (ABOUT) VOLUMES (OF TAPE) TAPE1:<RET>
Volumes of tape set TAPE1: DH33,DH44,DH55,DH56
@
```

7.4 RUNNING DUMPER

To run DUMPER, type DUMPER and press RETURN in response to the TOPS-20 prompt @. DUMPER prompts with its name and a right angle bracket (>):

```
@DUMPER<RET>
DUMPER>
```

NOTE

The examples in this chapter assume use of unlabeled tapes and tape drive allocation enabled.

As with TOPS-20 commands, you can type DUMPER commands in full, abbreviated, or recognition mode. Also, you must terminate all DUMPER commands by pressing RETURN.

To leave DUMPER and return to TOPS-20 command level, type either QUIT or EXIT:

```
DUMPER>EXIT<RET>
@
```

7.5 THE NONPRIVILEGED USER

As a nonprivileged user, you can use DUMPER to save and restore your own files, or any other files to which you have access.

The DUMPER commands to which you have access can be classified into three categories:

1. Status-setting commands, which set parameters affecting future operation of action commands.

2. Tape-positioning commands, which control the position of the tape.

3. Action commands, which start, stop, interrupt, or continue a file transfer or file check.

Sections 7.5.1 through 7.5.3 describe the commands in each category. Each section includes command formats and examples. Where applicable, command complements are included in the description. The complement of a command negates the command, and is formed by preceding the command with NO.

7.5.1 Setting the Status of Operation

The Status-setting commands set parameters that affect the operation of the action commands CHECK, RESTORE, RETRIEVE, and SAVE (refer to Section 7.5.3). Once you have set a parameter, it remains in effect until you change it or restart DUMPER. If you specify multiple conditions, the file you wish to transfer must satisfy all conditions to be transferred.

Two parameters generally associated with these commands are date and time. The date can be specified in any standard TOPS-20 format such as: 16-May-79, 5/16/79, etc. The time can be specified either in 24-hour format or in AM/PM format. For example 7:23 in the evening can be specified as 19:23 or as 7:23pm.

For a printed explanation of all DUMPER commands, run DUMPER and type HELP, followed by RETURN. This lists and briefly describes all DUMPER commands, and lists required arguments.

Table 7-1 lists all Status-setting commands according to their functions. A detailed description of each command follows the table.

Table 7-1: Status-Setting Commands

Function	Commands
Specifying the selection of specific files	BEFORE, ABEFORE MBEFORE, SINCE, ASINCE, MSINCE, NO DATES, SUPERSEDE, INITIAL
Specifying the printing (or suppressing) or storing of file or directory names	DIRECTORIES, FILES LIST, SILENCE
Specifying disk file	ACCOUNT, PROTECTION

characteristics	
Specifying the tape characteristics	DENSITY, FORMAT, INTERCHANGE, PARITY, TAPE, SET BLOCKING-FACTOR, SSNAME, SET TAPE-NUMBER
Specifying checksums to be included when a list of saved files is printed	CHECKSUM

The descriptions and examples below are divided into sets according to the functions described in Table 7-1. The first set describes the selection of files to be transferred on the basis of date and time, and according to name, type and generation.

NOTE

The parameters that transfer files on the basis of date and time cannot be used with the CHECKSUM command.

The format is the same for all BEFORE and SINCE commands. Below is a description of each, followed by examples of the commands.

BEFORE (DATE AND TIME) date time

The BEFORE command specifies that only files whose last user written date is before the date and time specified are to be transferred. The last user written date is the most recent time the user changed the actual data of the file. This date is recorded in the .FBWRT entry of the FDB, and is preserved when the file is copied.

NOTE

If time is omitted, the default is 00:00:01.

ABEFORE (DATE AND TIME) date time

The ABEFORE command specifies that only files whose last user access is before the date and time specified are to be transferred. The last access date is the most recent time the file was typed, printed, or read, but not modified. This date is recorded in the .FBREF entry of the FDB.

MBEFORE (DATE AND TIME) date time

The MBEFORE command specifies that only files whose last system write date is before the date and time specified are to be

transferred. The last system write date is the most recent time the file was physically changed on disk. This change includes copying the file. The date is recorded in the .FBCRE entry of the FDB, and is not settable by a nonprivileged user.

NOTE

Refer to Table 7-4 for a description of FDB entries that affect DUMPER.

SINCE (DATE AND TIME) date time

The SINCE command specifies that only files whose last user written date is later than the date and time specified are to be transferred. The last user written date is the most recent time the user changed the actual data of the file. This date is recorded in the .FBWRT entry of the FDB, and is preserved when the file is copied.

NOTE

If time is omitted, the default is 00:00:01.

ASINCE (DATE AND TIME) date time

The ASINCE command specifies that only files whose last user access is later than the date and time specified are to be transferred. The last access date is the most recent time the file was typed, printed, or read, but not modified. This date is recorded in the .FBREF entry of the FDB.

MSINCE (DATE AND TIME) date time

The MSINCE command specifies that only files whose last system write date is later than the date and time specified are transferred. The last system write date is the most recent time the file was physically changed on disk. This change includes copying the file. The date is recorded in the .FBCRE entry of the FDB, and is not settable by a nonprivileged user.

NO DATES

The NO DATES command disables all date and time commands at once. There is no DATES command.

Examples:

1. DUMPER>BEFORE (DATE AND TIME) 5/18/84 8:00AM<RET>

   only files created, or whose contents were modified, before 8:00 A.M. on May 18, 1984 are transferred.

2. DUMPER>ABEFORE (DATE AND TIME) 18-MAY-84 17:00<RET>

   only files accessed by a non-write operation (i.e., those that were typed, printed, or read) before 5:00 P.M. on May 18, 1984 are transferred.

3. DUMPER>MSINCE (DATE AND TIME) MAY-1-84 8:30<RET>

   only files that have been system modified (for example, copied) since 8:30 A.M. on May 1, 1984 are transferred.

4. DUMPER>SINCE (DATE AND TIME) 4-29-84<RET>

   only files that have been created or changed since the first minute of April 29, 1984 are transferred. (Because no time was specified, the default is midnight.)

```
DUMPER>REWIND<RET>
DUMPER>FILES<RET>
DUMPER>SINCE 1-JAN-84<RET>
DUMPER>SAVE (DISK FILES) PS:<MATO><RET>

 DUMPER tape #1, Fri 27-Jul-84 1326.  , volid TAPE2

 PS:<MATO>
   PS:<MATO>DUMPER.EXAMPLE.1
   PS:<MATO>DUMPER.EXE.6
   PS:<MATO>INIT.CMD.4
   PS:<MATO>INIT.MAC.1
   PS:<MATO>LOGIN.CMD.139
   PS:<MATO>MATO.LST.1
   PS:<MATO>MS.INIT.41
   PS:<MATO>NFT.INIT.10
   PS:<MATO>QE5.TEC.5
   PS:<MATO>TV.EXE.2


 Total files dumped:    10
 Total pages dumped:    77
 CPU time, seconds:     1.03
DUMPER>
```

The SUPERSEDE command sets the condition under which DUMPER overwrites a disk file with a magnetic tape file of the same name and type. You must specify one of the following conditions:

1. ALWAYS - when you want the tape file to overwrite the disk file of the same name and type regardless of the date or generation number of the disk file. If the disk file has a higher generation number, DUMPER restores the file from tape to disk deleting existing disk files with higher generation numbers. If the disk file has a lower generation number, DUMPER overwrites the disk file using the generation number of the file on tape. (Refer to the example below.)

2. NEVER - when you do not want the tape file to overwrite the disk file under any circumstances. In this case, a file is transferred to disk only if there is no file on disk with the same name and type.

3. OLDER - when the date that the tape file was last modified (created, read, written) is more recent than the date of the existing disk file with the same name and type. In that case, the tape file replaces the disk file. If the SUPERSEDE command is not specified, DUMPER assumes SUPERSEDE OLDER.

Example:

```
@VDIRECTORY (OF FILES) TV.INI<RET>

  PS:<MATO>
TV.INI.5;P777700        1 45(7)         7-Apr-83 14:12:56
@DUMPER<RET>
DUMPER>REWIND<RET>
DUMPER>SUPERSEDE ALWAYS<RET>
DUMPER>RESTORE (TAPE FILES) PS:<MATO>TV.INI<RET>
Saveset "Save of PS:<MATO>"
 Loading files into PS:<MATO>

End of Tape.

Total files restored:     1
Total pages restored:     1
```

The commands to activate or deactivate printing are similar. The defaults, however, are different, as described below.

DIRECTORIES

Normally, DUMPER prints directory names on your terminal as it saves or restores them. For example,

```
@DUMPER<RET>
[Using MTA-DUMPER:]
DUMPER>REWIND<RET>
```

```
DUMPER>SAVE (DISK FILES) LINK:<*>*.*.* (AS) LINK:<*>*.*.*<RET>

 DUMPER tape #1, Fri 27-Jul-84 1300.  , volid MTA-DU

 LINK:<HADY>
 LINK:<HADY.LINK6>
 LINK:<HDAVI>
 LINK:<HDAVI.TESTS.TST>
 LINK:<LINK.ALU>
 LINK:<LINK.ALU.V5M>
   .
   .
   .
```

To suppress printing, type NO DIRECTORIES before the SAVE or RESTORE command. This is useful if you are a privileged user performing backup of files on many directories. For example,

```
DUMPER>NO DIRECTORIES<RET>
DUMPER>SAVE (DISK FILES) LINK:<*>*.*.* (AS) LINK:<*>*.*.*<RET>

 DUMPER tape #1,  Fri  27-Jul-84  1300.  ,  volid MTA-DU


Total files dumped: 1413
Total pages dumped: 4923
CPU time, seconds:  15.04
DUMPER>
```

In the example above all files in all directories on the structure LINK: are saved on tape but DUMPER does not print the directory names on the terminal.

To reactivate printing before the next transfer operation, type DIRECTORIES in response to the DUMPER prompt.

FILES

Normally, DUMPER does not print a list of file specs as it is saving or restoring them. You see only the name of the structure and directory, and the number of total files and pages transferred. For the file specs to print on your terminal, type FILES before the SAVE or RESTORE command.

Example:

```
@DUMPER<RET>
[Using MTA-DUMPER:]
DUMPER>REWIND<RET>
DUMPER>DIRECTORIES<RET>
```



```
DUMPER>FILES<RET>
DUMPER>SAVE (DISK FILES) PS:<MATO>*.*.*<RET>
 DUMPER tape #2,  Fri  27-Jul-84 1309.  ,  volid TAPE2

 PS:<MATO>
   PS:<MATO>COMAND.CMD.32
   PS:<MATO>DUMPER.EXAMPLE.1
   PS:<MATO>DUMPER.EXE.6
   PS:<MATO>FTS.INIT.2
   PS:<MATO>INIT.CMD.4
   PS:<MATO>INT.MAC.1
   PS:<MATO>LOGIN.CMD.139
   PS:<MATO>MS.INIT.41
   PS:<MATO>NFT.INIT.10
   PS:<MATO>QE5.LIB.1
   PS:<MATO>QE5.TEC.5
   PS:<MATO>SED.INIT.10
   PS:<MATO>TV.EXE.2
   PS:<MATO>TV.INI.4
   PS:<MATO>TV2.INI.4
   PS:<MATO>TVSM.INI.1


Total files dumped:     16
Total pages dumped:     76
CPU time, seconds:      2.04
```

To deactivate printing before the next transfer operation, type NO FILES in response to the DUMPER prompt.

LIST (LOG INFORMATION ON FILE) file spec

Normally, DUMPER does not produce a file containing a list of all files as it saves them on tape. If you wish to have such a list but know it is too long to be printed at your terminal, use the LIST command, including the name of the file in which you want the list printed. For example,

```
DUMPER>LIST (LOG INFORMATION ON FILE) MATO.LST<RET>
```

When the save operation is complete, you can return to TOPS-20 command level and print the file MATO.LST or type it on your terminal. You cannot use LIST when doing a restore.

Example:

```
@DUMPER<RET>
DUMPER>REWIND<RET>
DUMPER>SSNAME Save of PS:<MATO><RET>
DUMPER>LIST (LOG INFORMATION ON FILE) MATO.LST<RET>
DUMPER>SAVE PS:<MATO><RET>
```

```
 DUMPER tape #2,  Fri  27-Jul-84  1310.  Saveset "Save of PS:<MATO>", volid TAPE2

Total files dumped:     16
pages dumped:           76
CPU time, seconds:      2.04
```

If you omit a file spec with the LIST command, DUMPER automatically creates a file called LPT:DUMPER.LOG, which is printed on the line printer. If an existing file is specified, the new list of file specs is appended to the existing file.

To deactivate the LIST command before the next transfer operation, type the command NO LIST.

SILENCE

As described previously, directory names are normally printed; file specs are not. The SILENCE command suppresses the printing of directories and file specs as the files are saved or restored. The SILENCE command is therefore equivalent to NO DIRECTORIES if the FILES command has not been specified. Typing NO SILENCE is equivalent to FILES and DIRECTORIES.

The commands ACCOUNT and PROTECTION define the file characteristics for the file being restored from magnetic tape.

ACCOUNT (OF RESTORED FILES FROM) keyword

The ACCOUNT command specifies that any file being restored to disk assumes either your current account (specified as SYSTEM-DEFAULT) or the account of the file when it is saved on tape (specified as TAPE). Without the command, DUMPER defaults to TAPE.

PROTECTION (OF RESTORED FILES FROM) keyword

The PROTECTION command specifies that any file being restored takes either the system default protection code or the protection code of the file on tape. The arguments are SYSTEM-DEFAULT (usually 777700) or TAPE. Without the command, DUMPER defaults to TAPE.

Of the commands that set tape characteristics, the TAPE command is used most often. The others, described below, either change the basic attributes of the tape, prepare it for reading or writing with another format, or supply an identifying name for a saveset.

TAPE (DEVICE) name:

The TAPE command identifies the mounted magnetic tape to DUMPER. The name can be either the physical device assigned to you (such as MTA3: or MT0:) or logical name (such as TAPE1:). If you omit the name:, DUMPER prints the reminder:

```
Tape specification needed:
```

and waits for you to supply the device name.

As mentioned in Section 7.3, if you define your tape with the logical name MTA-DUMPER:, you need not type the TAPE command for DUMPER to access the tape.

DENSITY (OF MAGTAPE) n

The DENSITY command sets the density of the tape to the specified number of bits per inch (bits/in). The density can be 200, 556, 800, 1600, 6250 or JOB-DEFAULT (set by the TOPS-20 command SET TAPE DENSITY). If you do not specify the DENSITY command, DUMPER uses the density listed when you type the TOPS-20 command INFORMATION (ABOUT) TAPE-PARAMETERS. If the tape is labeled, the density is predetermined by the label information on the tape.

PARITY (OF MAGTAPE) type

The PARITY command sets the parity of the mounted magnetic tape to EVEN or ODD. DUMPER normally uses the parity listed when you type the TOPS-20 command INFORMATION (ABOUT) TAPE-PARAMETERS (the system default is ODD).

SET BLOCKING-FACTOR (TO) n (RECORDS)

The SET BLOCKING-FACTOR command sets the number of logical records per physical record that DUMPER writes on tape. The default value is 1. The number must be in the range of 1 to 15. The maximum depends on the density of the tape as follows:

Density	Maximum Blocking Factor
200	1
556	3
800	4
1600	10
6250	15

NOTE

INTERCHANGE mode uses a blocking factor of 1. If the blocking factor is changed and you specify INTERCHANGE

mode, DUMPER resets the blocking factor to 1. If you specify NO INTERCHANGE, DUMPER uses whatever blocking factor you previously used.

Blocking is defined as writing physical records that contain more than one logical record. (Versions of DUMPER previous to Release 4 write only one logical record per physical record.) The term "blocking factor" refers to the number of logical records per physical record. The advantage of blocking is to permit DUMPER to fit more information on a tape than it could without blocking, because DUMPER reduces the number of gaps between logical records. This reduces the number of tapes and the amount of time it takes to save and restore files. However, a tape written with a blocking factor other than 1, and any labeled tape, cannot be restored by DUMPER previous to Version 4 of TOPS-20. It is not necessary to give this command when restoring files from tape; DUMPER automatically determines the blocking factor by reading the tape, and then processes the tape accordingly. There can be only one blocking factor for each tape.

FORMAT (VERSION NUMBER IS) n

The FORMAT command tells DUMPER the tape format that was used to write the tape. DUMPER accepts versions 4, 5, and 6. Version 6 is the current tape format version. If you position a tape so that you are not reading from the beginning, use the FORMAT command to tell DUMPER what version to expect.

INTERCHANGE (FORMAT)

The INTERCHANGE command allows DUMPER to read and write tapes written or to be read by the TOPS-10 BACKUP program. (Normally DUMPER uses its own format.) You should never use INTERCHANGE when writing tapes to be read by another TOPS-20 system. DUMPER uses NO INTERCHANGE for its default. When restoring from an INTERCHANGE tape, <CTRL/E> and the date commands are ignored.

The following example shows a tape written by the TOPS-10 BACKUP program followed by an example of reading a tape using the TOPS-20 DUMPER program in INTERCHANGE mode.

```
.MOUNT BACKUP:/WRITE:YES/REELID:MACS<RET>
[Mount Request BACKUP Queued, Request-ID 109]
[Volume MACS Mounted on MTB263 as Logical Name BACKUP]
BACKUP mounted, MTB263 used

.R BACKUP<RET>

/TAPE BACKUP<RET>
```



```
/FILES<RET>
/INTERCHANGE<RET>
/REWIND<RET>
/SAVE *.MAC[30,5153]<RET>
!30,5153                DSKC
 DECPNT MAC
 LOCAL  MAC
 READ   MAC
 TRMTYP MAC
 NDBDEF MAC
 NETDPY MAC
 PRVCK  MAC
 DET    MAC

"Done

/EXIT

.DISMOUNT BACKUP:<RET>
MTB263 Dismounted

.

@
@MOUNT TAPE (NAME) MACS: /READ-ONLY /LABEL-TYPE:UNLABELED<RET>
[Mount Request MACS Queued, Request-ID 127]
[Tape set MACS, volume MACS mounted]
[MACS: defined as MT2:]
@DUMPER<RET>
DUMPER>TAPE (DEVICE) MACS:<RET>
DUMPER>INTERCHANGE (FORMAT)<RET>
DUMPER>REWIND<RET>
DUMPER>FILES<RET>
DUMPER>RESTORE (TAPE FILES) *.*.* (TO) DSK*:<*>*.*.*<RET>

DUMPER tape # 1  , Tuesday, 18-Sep-79 1131   Volid MACS.
 Loading file(s) into MISC:<HUTCHINS>
DECPNT.MAC (TO) DECPNT.MAC.1    [OK]
LOCAL.MAC (TO) LOCAL.MAC.1      [OK]
READ.MAC (TO) READ.MAC.1        [OK]
TRMTYP.MAC (TO) TRMTYP.MAC.1    [OK]
NDBDEF.MAC (TO) NDBDEF.MAC.1    [OK]
NETDPY.MAC (TO) NETDPY.MAC.1    [OK]
PRVCK.MAC (TO) PRVCK.MAC.1      [OK]
DET.MAC (TO) DET.MAC.1 [OK]

End of saveset
DUMPER>EXIT<RET>
@
```

SSNAME name

The SSNAME command specifies the name to be written in the saveset header on the tape or to appear when you type a RESTORE, SAVE, SKIP, EOT, or PRINT command. The name may contain up to 200 alphanumeric characters, including spaces. Whenever you save or restore files, the saveset name prints on your terminal.

When you do not specify a saveset name (SSNAME), DUMPER prints a period and a comma. The saveset name normally appears between the period and the comma.

The CHECKSUM command is used in conjunction with the DUMPER SAVE and PRINT commands. (Refer to Section 7.5.3.)

CHECKSUM (FILES) type

While files are being saved, a checksum is computed. When you type PRINT, each filename is listed with a six-digit octal checksum number.

One of two types can be specified: SEQUENTIAL or BY-PAGES. If you specify SEQUENTIAL, DUMPER computes a checksum for the entire file. (In INTERCHANGE mode, you should use only sequential checksums.) If you specify BY-PAGES, the checksum includes each word of existing pages up to the End-Of-File (EOF) pointer. When you type PRINT after requesting a checksum by pages, the checksum number is followed by the letter P, as shown in the example below.

```
@DUMPER<RET>
[Using MTA-DUMPER:]
DUMPER>REWIND<RET>
DUMPER>CHECKSUM (FILES) BY-PAGES<RET>
DUMPER>PRINT (DIRECTORY OF TAPE ONTO FILE)<RET>

 DUMPER tape #2  Fri 27-Jul-84 1311.  saveset "Save of PS:<MATO>",  volid TAPE2

     Filename                  Last write date        Pages        Checksum
Page #1

     PS:<MATO>COMAND.CMD.32    8-Sep-83 1041           1           536370 P
     PS:<MATO>DUMPER.EXAMPLE.1 27-Jul-84 1307          1           223135 P
     PS:<MATO>DUMPER.EXE.6     27-Jul-84 1245          31          144736 P
     PS:<MATO>FTS.INIT.2       30-Jun-83 1842          1           664252 P
     PS:<MATO>INIT.CMD.4       25-Jul-84 1848          1           332727 P
     PS:<MATO>INT.MAC.1        9-Jul-84 2143           3           632501 P
     PS:<MATO>LOGIN.CMD.139    17-Apr-84 2035          1           172204 P
     PS:<MATO>MATO.LST.1       27-Jul-84 1311          1           642240 P
     PS:<MATO>MS.INIT.41       25-Jan-84 1155          1           610300 P
     PS:<MATO>NFT.INIT.10      25-Jul-84 1851          1           247324 P
     PS:<MATO>QE5.LIB.1        27-Jan-83 1351          1           042624 P
     PS:<MATO>QE5.TEC.5        9-Jul-84 2214           3           136673 P
```



```
     PS:<MATO>SED.INIT.10      18-Oct-82 0913          1           332725 P
     PS:<MATO>TV.EXE.2         20-Jan-84 1303          27          713471 P
     PS:<MATO>TV.INI.4         7-Apr-83 1412           1           632303 P
     PS:<MATO>TV2.INI.4        30-Jun-82 1512          1           103641 P
     PS:<MATO>TVSM.INI.1       6-Jan-82 1240           1           756311 P

 End of Tape.
```

When the checksum of the disk file disagrees with the checksum of the tape file, there is an error in the file transfer.

Two other Status-Setting commands allow you to begin a save at a specific file, or to continue a save onto a second reel of tape.

INITIAL (FILESPEC) file spec

The INITIAL command allows you to save files starting with a specific file specification. If you specify a file, the save operation begins when DUMPER encounters a matching file. DUMPER ignores all files until a match is seen. For example, if you want to begin a save starting with FILE3, type

```
DUMPER>INITIAL (FILESPEC) FILE3<RET>
```

SET TAPE-NUMBER (DECIMAL NUMBER) n

The SET TAPE-NUMBER command is used for multi-reel savesets in either of two cases: when continuing a restore after a system crash, or when restoring files nonsequentially from a multi-reel saveset.

7.5.2 Positioning the Tape

Tape-Positioning commands control the position of the tape without transferring information between the tape and disk. These commands affect the tape you specified with your last TAPE command to DUMPER. If you did not use a TAPE command, DUMPER issues a request for one, or uses a tape assigned to the logical name MTA-DUMPER:. If you are working on a system without tape drive allocation, before entering DUMPER, you must assign the tape drive to your job, using the TOPS-20 command ASSIGN.

Table 7-2 lists the four Tape-Positioning commands and their functions. Following the table is a detailed description, including examples, of each command.

Table 7–2: Tape–Positioning Commands

Command	Function
EOT	Positioning to the end of the last saveset on the tape
REWIND	Positioning to the beginning of the currently mounted tape
SKIP n	Positioning to the end of the nth from the current saveset
UNLOAD	Rewinding the tape entirely onto the source reel

The three commands below position the tape at specific points.

EOT

The EOT (End–Of–Tape) command positions the tape at the end of the last saveset on the tape and DUMPER prints the message:

```
End of Tape.
```

As DUMPER encounters each saveset between its current position and EOT, it prints the saveset name (but not its contents).

Example:

```
DUMPER>EOT<RET>
Saveset "Save of PS:<MATO>"
Saveset "Saveset #2, PS:<MATO>"
 End of Tape.
```

REWIND

The REWIND command rewinds the currently mounted volume to the beginning of the tape.

If you are using a multi–reel tape set mounted by typing the TOPS–20 command MOUNT TAPE, you can switch to a different reel by typing:

```
REWIND SWITCHING (TO VOLUME NUMBER) n<RET>
```


With this command, n specifies the number of the reel within the set, not the volume id. When you type the REWIND SWITCHING command, DUMPER releases the currently mounted tape and requests the nth volume to be mounted. (If you are using labeled tapes, DUMPER allows you to switch to volume 1 only.) The following example shows a switch to the volume identified as TWO from volume ONE:

```
@MOUNT TAPE TAPE1: /WRITE–ENABLED /VOLIDS:ONE,TWO,THREE<RET>
[Mount Request TAPE1 Queued, Request–ID 174]
[Tape set TAPE1, volume ONE mounted]
[TAPE1: defined as MT0:]
@DUMPER<RET>
DUMPER>TAPE (DEVICE) TAPE1:<RET>
DUMPER>REWIND SWITCHING (TO VOLUME NUMBER) 2<RET>
DUMPER>
```

NOTE

If you plan to switch back and forth between volumes of a set, specify the /NOUNLOAD switch with the TOPS–20 command MOUNT TAPE. This is a request to the operator not to unload any of the tapes within the set, but to use additional tape drives so that more than one tape can be mounted at a time.

SKIP (NUMBER OF SAVESETS) n

The SKIP command moves the tape over the specified number of savesets. This ensures that those savesets are not deleted by another save. DUMPER prints the name of each saveset encountered, and positions the tape at the end of the nth saveset.

If n is 0, the tape is positioned at the beginning of the current saveset. If n is a negative number, the tape backspaces by n savesets and is positioned at the beginning of that saveset. You cannot use zero or negative numbers with labeled tapes.

Examples:

```
@DUMPER<RET>
[Using MTA–DUMPER:]
DUMPER>REWIND<RET>
DUMPER>SKIP (NUMBER OF SAVESETS) 0<RET>
Saveset "Saveset #2, PS:<MATO>"
DUMPER>
```

```
DUMPER>SKIP (NUMBER OF SAVESETS) -4<RET>
Saveset "Save of PS:<MATO>"
 Beginning of tape.
DUMPER>
```

The following command releases your mounted tape from the drive.

UNLOAD

If your installation is not using tape drive allocation, you can remove your tape with UNLOAD to DUMPER.

Example:

```
DUMPER>UNLOAD<RET>
```

This command, however, does not deassign the tape drive.

7.5.3 Interacting with Tape Files

The action commands affect the files on the tape specified by the last TAPE command. These commands allow you to start, stop, interrupt, or continue a file transfer or file check.

The transfer commands take, as an optional argument, one or more file specifications. The file specifications can contain wildcard characters. With transfer commands (SAVE and RESTORE), you can indicate both input and output (source and destination) file specifications for each file specified. This allows the files to be renamed as they are saved or restored. If no destination specification is given, the specified files are transferred without being renamed. If no argument is given with the transfer commands, all files on your connected directory are saved onto tape, or all files that were saved from your connected directory, and exist in the current saveset, are restored to your connected directory.

Both the SAVE and RESTORE commands can take the source and destination arguments. In the case of SAVE, the source is the file specification identifying the name and location of the file to be saved; the destination identifies the file name under which you want the file to be stored on tape. In the case of RESTORE, the source is a file specification identifying the file you want copied from the tape; the destination is a file specification identifying the name of the file into which you want the file copied on disk.

With both commands, if you omit the source, DUMPER restores all files in the saveset corresponding to your connected directory, or saves all files as your connected directory. If you omit the destination with SAVE, DUMPER transfers the file(s) with the same file specification(s) as the one(s) on disk. If you omit the destination with RESTORE,

DUMPER transfers the file(s) with the same file specification(s) as the one(s) on tape.

Table 7-3 lists all action commands according to their functions. A detailed description of each command follows the table.

Table 7-3: Action Commands

Command	Function
ABORT	Cancelling an interrupted command and starting a new action command
RESTORE, SAVE, CHECK, TRANSFER	Transferring and comparing disk and tape files
PRINT	Printing a list of file names on tape
<CTRL/A>	Printing status information about the command
<CTRL/E>, CONTINUE	Halting and continuing command operation
TAKE	Executing commands from a command file
EXIT, QUIT	Exiting to TOPS-20
EXACT	Saving or restoring files using system-wide logical names

SAVE (DISK FILES) source (AS) destination, source (AS) destination

The SAVE command creates a saveset on tape, containing all the source files specified. To rename the files as you save them, you must specify source files followed by a space, followed by destination files. With either source or destination files you can use * and % wildcard characters. To transfer additional source and destination file specifications, you must separate each pair with a comma. If you do not specify destination, DUMPER saves all source file specifications under their original specifications. If you specify neither source nor destination, DUMPER saves all files on your connected directory into the saveset on tape (i.e., DUMPER assumes the file specification *.*.*).

Examples:

Note that in these examples, three savesets are created, one for each SAVE command.

```
@DUMPER<RET>
[Using MTA-DUMPER:]
DUMPER>REWIND<RET>
DUMPER>FILES<RET>
DUMPER>SAVE (DISK FILES) PS:<MATO>TV.INI.* (AS)
PS:<MATO>TV.OLD.*<RET>

 DUMPER tape #2, Fri 27-Jul-84 1319.  Saveset "Saveset #2, PS:<MATO>", Volid
TAPE2

 PS:<MATO>
   PS:<MATO>TV.INI.5 (as) PS:<MATO>TV.OLD.5


 Total files dumped:      1
 Total pages dumped:      1
 CPU time, seconds:       0.03
DUMPER>SAVE (DISK FILES) <MOSE>*.*.* (AS) *.*.*<RET>

 DUMPER tape #2, Fri 27-Jul-84 1319.  Saveset "Saveset #2, PS:<MATO>", Volid
TAPE2

 PS:<MOSE>
   PS:<MOSE>2946-CDRSRV-MAC.RED.1
   PS:<MOSE>2946-PROLOG-MAC.RED.1
   PS:<MOSE>ACCESS.CMD.11
   PS:<MOSE>ALERT.CMD.16
   PS:<MOSE>BACCESS.CMD.5
   PS:<MOSE>BATCH.CMD.7
   PS:<MOSE>BMOUNT.CMD.9
   PS:<MOSE>CF.CTL.1
   PS:<MOSE>CFSINT.DOC.1
   PS:<MOSE>CH2EX.MAC.3
          .
          .
          .

Total files dumped:     55
Total pages dumped:     172
CPU time, seconds:      7

DUMPER>NO FILES<RET>
DUMPER>SAVE (DISK FILES) PS:<PERLMA>*.*.* (AS) *.*.*<RET>

 DUMPER tape #2, Fri 27-Jul-84 1321.  , volid TAPE2

 PS:<PERLMA>

Total files dumped:      37
Total pages dumped:      208
```



```
CPU time, seconds:       5.04
```

NOTE

Remember that if the NO FILES command is typed, as in the last SAVE example above, the file specifications are not listed as they are saved.

If all files cannot fit on one tape, DUMPER requests that you mount a second tape. This tape is the next volume of the tape set if you type /VOLIDS: with the MOUNT TAPE command. If you do not specify the switch, the operator mounts a tape and it becomes the second volume of the set. When the additional tape is mounted, DUMPER resumes the save operation.

It is important to note that savesets are stored on tape according to the position of the tape when the SAVE command is typed. If you position the tape at the beginning (refer to the REWIND command), the specified files are written there regardless of any previous files saved. If more than one saveset exists on the tape, they are all deleted when a new one is saved at the beginning of the tape. To avoid deleting existing savesets, position the tape at the end of the last saveset (EOT) or skip the number of savesets (SKIP n) you want to be sure to preserve.

RESTORE (TAPE FILES) source (TO) destination, source (TO) destination

The RESTORE command transfers the specified source files from magnetic tape to disk. To rename the files as you restore them, you must specify destination file specifications. With source files you may use the * and % wildcard characters; with destination files, you can use only the * wildcard.

If you do not specify destination, all specified files are restored with the same name, type, and generation. If you do not specify either source or destination specifications, all files saved from your connected structure and connected directory are restored to your connected structure and directory with the same name, type and generation number.

When DUMPER begins restoring files to a directory, it prints the message:

Loading File(s) into <directory>

NOTE

If you do not see this message, DUMPER is not transferring the files.

In that case, the files may already exist on disk, they may have

been saved from a structure or directory other than your currently connected directory, they do not exist in the current saveset, or you used a wildcard expression that does not match the files on tape.

If you are restoring files that were saved from a directory other than your own, you must specify that directory in the source file specification. If files exist on that directory with the same name and type as those on tape, DUMPER overwrites the disk files according to the condition specified with the SUPERSEDE command: ALWAYS, NEVER, or OLDER. If the SUPERSEDE command has not been typed, DUMPER assumes SUPERSEDE OLDER and overwrites the disk file only if the tape file is newer. All date commands and time commands are ignored when you restore from an INTERCHANGE tape.

As is true with the SAVE command, files are restored from the current position on tape. If you want to omit one or more savesets, position the tape to the end of the last saveset to be omitted (SKIP n).

Example:

```
@DUMPER<RET>
[Using MTA-DUMPER:]
DUMPER>REWIND<RET>
DUMPER>SKIP (NUMBER OF SAVESETS) 2<RET>

Saveset "Saveset #2, PS:<MATO>"
Saveset "Saveset #3, PS:<MATO>"
Saveset, unnamed
DUMPER>FILES<RET>
Saveset, unnamed
 Loading files into EXODUS:<PERLMA>
  PS:<MATO>DUMPER.EXAMPLE.1 to EXODUS:<PERLMA>DUMPER.EXAMPLE.1;P777777;AMONITOR [O
K]
  PS:<MATO>DUMPER.EXE.6 to EXODUS:<PERLMA>DUMPER.EXE.6;P777777;AMONITOR [OK]
  PS:<MATO>INIT.CMD.4 to EXODUS:<PERLMA>INIT.CMD.4;P777777;AMONITOR [OK]
  PS:<MATO>INT.MAC.1 to EXODUS:<PERLMA>INT.MAC.1;P777777;AMONITOR [OK
  PS:<MATO>LOGIN.CMD.139 to EXODUS:<PERLMA>LOGIN.CMD.139;P777700;AMONITOR [OK]
  PS:<MATO>MATO.LST.1 to EXODUS:<PERLMA>MATO.LST.1;P777777;AMONITOR [OK]
  PS:<MATO>MS.INIT.41 to EXODUS:<PERLMA>MS.INIT.41;P777700;AMONITOR [OK]
  PS:<MATO>NFT.INIT.10 to EXODUS:<PERLMA>NFT.INIT.10;P777700;AMONITOR [OK]
  PS:<MATO>QE5.TEC.5 to EXODUS:<PERLMA>QE5.TEC.5;P777777;AMONITOR [OK]
  PS:<MATO>TV.EXE.2 to EXODUS:<PERLMA>TV.EXE.2;P777777;AMONITOR [OK]
End of Tape.

 Total files restored:     10
 Total pages restored:     77
DUMPER>
```

Once the files have been saved or restored, you can check the disk and tape versions to verify that the transfer was accurate.

TRANSFER (TAPE FILES)

The TRANSFER command is another way to restore files to disk. If you type,

```
DUMPER>TRANSFER<RET>
```

all files from all structures and directories on the tape are restored to your connected directory with the same name and type. The TRANSFER command always defaults to DSK*:<*>*.*.* for an input file specification and to your connected directory for a destination specification. If you specify both input and output specifications, the TRANSFER command acts as a RESTORE command with different default actions.

EXACT (MODE FOR SAVE COMMAND)

The EXACT command saves or restores files without translating logical names into actual structure names. The files are saved or restored using the logical name as it is specified.

For example, if the system uses logical name PS: to refer to the structure SYSDSK: and you use the EXACT command before a SAVE command, the files are saved as PS:<dir>file spec. If you want to restore files that are saved using the EXACT command, type EXACT before typing RESTORE and the files are restored as PS:<dir>file spec.

Files that are saved using the EXACT command cannot be restored using the NO EXACT command since DUMPER cannot match the files.

To use the EXACT command, type

```
DUMPER>EXACT<RET>
DUMPER>SAVE (DISK FILES) LOGICAL NAME:<DIR> FILE SPEC<RET>
```

If you want to have logical names translated, use the NO EXACT command. NO EXACT is the default.

CHECK (ALL TAPE FILES)

The CHECK command checks the File Descriptor Blocks (FDB) in the current saveset to make sure they agree with the FDB of the files on disk. If any files do not agree, DUMPER prints one or more error messages. Table 7-4 is a list of the possible differences in the FDBs that DUMPER checks.

NOTE

Be sure to position the tape at the beginning of the saveset before typing CHECK.

Table 7–4: File Descriptor Block (FDB) Entries Checked by DUMPER

A difference in location	Means the files do not have the same:
.FBCTL	–Temporary, permanent, not–to–be–saved–by DUMPER, or file–class status.
.FBPRT	–File access code.
.FBCRE	–Date and time of last write to the file. It is modified when any program writes to the file. This word is changeable by a user with privileged capabilities.
.FBAUT	–Pointer to the string containing the name of the author.
.FBGEN	–Generation and directory numbers of the file.
.FBACT	–Account information.
.FBBYV	–Number of generations to retain, byte size, mode of the last write, or the number of pages.
.FBSIZ	–Number of bytes in the file. This word is changeable by a user with write access.
.FBCRV	–Date and time of creation of the file. This word is changeable by a user with write access.
.FBWRT	–Date and time of the last user write to the file. This word is changeable by a user with write access.
.FBREF	–Date and time of the last non–write access to the file. This word is changeable by a user with write access.
.FBCNT	–Count of writes or references.
.FBUSW	–Contents of the user–settable data area.
.FBLWR	–Pointer to the string containing the name of the user who last wrote to the file.

Example:

```
DUMPER>REWIND<RET>
DUMPER>CHECK (ALL TAPES FILES)<RET>
```



```
Saveset "The check–files–for–changes"
%Difference in .FBCRE of file SNARK:<SANTI>ET.HLP.1
%Difference in .FBCRV of file SNARK:<SANTI>ET.HLP.1
%Difference in .FBWRT of file SNARK:<SANTI>ET.HLP.1
%Difference in .FBCRE of file SNARK:<SANTI>EXEDMP.HLP.1
%Difference in .FBSIZ of file SNARK:<SANTI>EXEDMP.HLP.1
%Difference in .FBCRV of file SNARK:<SANTI>EXEDMP.HLP.1
%Difference in .FBWRT of file SNARK:<SANTI>EXEDMP.HLP.1
 End of Saveset.
DUMPER>
```

If a disk file is renamed before you do a check, the tape and disk file will not agree in either name, type, or generation. In this case, DUMPER prints an error message (see Section 7.8).

You cannot use the CHECK command in combination with the date and time commands.

If you want a printed list of all files saved on tape, position the tape to the beginning and type the PRINT command. This command differs from LIST in that it creates a file of file names already on tape. LIST creates a file of file names as DUMPER is saving the files.

<CTRL/A>

The <CTRL/A> command prints one or more lines of information. With all DUMPER commands, <CTRL/A> prints the name of the command DUMPER is processing. With SAVE commands, in addition to the name of the command in process, <CTRL/A> prints the name of the file and the number of the disk page DUMPER is currently processing. The <CTRL/A> will not echo on your terminal.

```
@DUMPER<RET>
[Using MTA–DUMPER:]
DUMPER>REWIND<RET>
DUMPER>SAVE (DISK FILES) LINK:<*>*.*.* (AS) LINK:<*>*.*.*<RET>
 DUMPER tape #1, Fri 27–Jul–84 1300.  , volid MTA–DU

 LINK:<HARY>
 LINK:<HARY.LINK6>
 LINK:<HDAVI>
 LINK:<HDAVI.TESTS.TST>
 LINK:<LINK.ALU>
 LINK:<LINK.ALU.V5M>
 <CTRL/A>
 SAVE in progress. File: LINK:<LINK.ALU.V5M>LNKHST–41.RED.1 (1)
  .
  .
```

.

PRINT (DIRECTORY OF TAPE ONTO FILE) file spec

The PRINT command records the file specifications of the entire tape (beginning with the current saveset) in the specified output file. You can subsequently use the TOPS-20 TYPE or PRINT command to examine this file.

```
@DUMPER<RET>
[Using MTA-DUMPER:]
DUMPER>REWIND<RET>
DUMPER>PRINT (DIRECTORY OF TAPE ONTO FILE) FILES.LST.1<RET>
 DUMPER tape #2, Fri 27-Jul-84 1311.  Saveset "Save of PS:<MATO>", volid TAPE2
End of tape.
DUMPER>EXIT<RET>
@TYPE (FILE) FILES.LST<RET>
```

If you omit the destination file specification, the contents of the current saveset are printed on your terminal. Typing PRINT/FAST also defaults to printing on your terminal. PRINT/FAST tries to limit output to 80 columns.

TAKE (COMMANDS FROM FILE) file specification

The TAKE command causes DUMPER to execute commands in the specified command file (file specification). Commands from the command file are executed until either the end of the command file or until DUMPER encounters an error. DUMPER does not echo commands as they are processed unless more information is needed or DUMPER encounters an error.

```
@DUMPER<RET>
DUMPER>TAKE (COMMANDS FROM FILE) D.CMD<RET>

[End work:<MATO>D.CMD.1]
DUMPER>EXIT<RET>
@
```

If you end a command file with a TAKE command that is not followed by a file specification, you do not receive the [End] message shown in the example above.

If necessary, you may interrupt the operation of an action or tape-positioning command. You may then continue the operation, or cancel it by typing another command.

<CTRL/E>

You type the <CTRL/E> command by pressing the CTRL and E keys simultaneously. The <CTRL/E> is not echoed on your terminal. This command halts the action of a SAVE, RESTORE, CHECK, RETRIEVE or any tape-positioning command. When DUMPER processes the <CTRL/E> command, it interrupts the current operation and prints:

```
Interrupting...
DUMPER>>
```

You can type any status-setting command except INTERCHANGE in response to the prompt, and then type CONTINUE to continue the operation at the point at which it was interrupted.

To start a new command, type ABORT to discard the interrupted command. Then type the new action command.

CONTINUE

The CONTINUE command allows you to continue the operation of an interrupted action or tape-positioning command.

You cannot use the CONTINUE command if you have not interrupted DUMPER with <CTRL/E>.

Example:

```
DUMPER>REWIND<RET>
DUMPER>SAVE (DISK FILES) LINK:<*>*.*.*<RET>
 DUMPER tape #1, Fri 27-Jul-84 1300.  , volid MTA-DU

 LINK:<HARD>
 LINK:<HARD.LINK.6>
<CTRL/E>
Interrupting...
DUMPER>>NO DIRECTORIES<RET>
DUMPER>>CONTINUE (SAVE)<RET>
 Continuing SAVE command...
<CTRL/E>
Interrupting...
DUMPER>>ABORT<RET>
 Aborting SAVE command...
```

There are two commands to exit from the DUMPER program and return to TOPS-20 command level. The EXIT and QUIT commands have identical functions.

Example:

```
DUMPER>EXIT<RET>
@
```

or

```
DUMPER>QUIT<RET>
@
```

7.5.4 Marking Files to be Archived

If you wish to save any of your files on the installation's archive tape, you can mark the files for archival. To do this, use the TOPS-20 ARCHIVE command. Once the operator has processed your request, only the File Descriptor Blocks (FDB's) remain. The file names become invisible (i.e., if you take a directory of your disk area, the files appear to be gone). This is to assure you that your files have been archived and that you can no longer access or modify them.

NOTE

To see your files listed in the archive request queue, type the TOPS-20 command INFORMATION ARCHIVE-STATUS.

7.6 THE PRIVILEGED USER

As a privileged user, you can perform operations beyond the capabilities of a nonprivileged user. These operations include backup and restoration of all system files, archiving, and migrating. Below is a list of switches, commands, and defaults to which only privileged users may have access.

- As part of the SAVE command, you may append the following switches:

  /FULL-INCREMENTAL to request that DUMPER save all specified files and reset the file save count to one. All information is preserved regarding the number of times the file has been saved. This is the command for a weekly system backup of all files.

  /INCREMENTAL:n to request that DUMPER save all files that have either not been saved at least n times, or have been modified since the last INCREMENTAL or FULL-INCREMENTAL run. This is the command for a daily backup of all files.

  NOTE

  If a save has not previously been done with the FULL-INCREMENTAL switch, specifying the INCREMENTAL

  switch will have the same results as specifying /FULL-INCREMENTAL (i.e., DUMPER saves all files).

  /NOINCREMENTAL to request that DUMPER not perform an incremental save. In this case, DUMPER saves all files, but does not preserve any information regarding the number of times the file has been saved. (/NOINCREMENTAL is the default.)

  NOTE

  In using any of these switches, you do not have the option of renaming files from disk to tape. A file transferred to tape has the same name as the file on disk. Unless one of these switches or the CREATE command is used, directory information is not saved on the tape

  /UNLOAD to request that DUMPER unload a mounted tape after a save.

- You can RESTORE or RETRIEVE files from an archive or migration tape.
- You can use the CREATE command to recreate an entire directory that has been deleted.
- When you type SAVE or RESTORE with no specified directory, the default is <*> on the connected structure. (For a nonprivileged user, the default is the connected directory and structure.)
- When you restore files, the last writer string of the file on disk is set to the name of the last writer at the time of the save. (For a nonprivileged user, the last writer string is set to the name of the user doing the RESTORE.)

If your installation is running a TOPS-20 Version 6 based monitor, DUMPER supports encrypted passwords and project-programmer numbers (PPN). Earlier versions of DUMPER, however, do not support the password and PPN features. Therefore, use extreme caution when changing versions of DUMPER and the TOPS-20 monitor.

At user level, password encryption and PPNs are not visible in DUMPER. These two features affect the process of creating, saving and restoring directories. They do not affect files.

If, for example, you save a directory using a 4.1 version of DUMPER and then restore that tape using TOPS-20 Version 6, passwords are not handled correctly. They are not useable and you have to respecify the passwords using the BUILD command. See the TOPS-20 Commands Reference Manual for an explanation of the BUILD command. PPNs are not restored

at all. For a detailed explanation of the compatibility between DUMPER versions and TOPS-20 versions, see the TOPS-20 System Manager's Guide.

Sections 7.6.1 through 7.6.6 describe the use of these privileged commands and switches. This manual does not include step-by-step procedures for performing a system backup, restore, archival run, or migration run. For details, refer to the TOPS-20 Operator's Guide.

7.6.1 Backing Up System Files and/or Other Users' Files

To minimize loss of disk files, you should put backup copies of all files on magnetic tape. In many installations, this should be done on a daily basis. (For details, refer to the TOPS-20 System Manager's Guide.)

Backup is done with the CREATE and SAVE commands. As a privileged user you have the option of doing one of three types of saves by using SAVE plus either /FULL-INCREMENTAL, /INCREMENTAL, or /NOINCREMENTAL.

NOTE

It is recommended that you specify the CREATE command before you perform a SAVE that is intended as a backup copy. With the SAVE /FULL command, the CREATE command writes directory information on the tape. This is useful if you have to restore a damaged or deleted directory.

A sample backup routine for an installation might be:

Friday night: Run DUMPER to save all system files and directories using /FULL-INCREMENTAL.

All other nights: Using /INCREMENTAL:2, save all files with changes made since the save of the previous night. This allows you a margin of error in recovering files that have been inadvertently deleted and expunged.

The incremental runs can be made either more or less frequently at the discretion of the installation. In the event of a total file system loss, the installation can restore the disk to the state of the most recent incremental by first restoring the full-incremental tapes to a fresh set of packs, and then restoring any incremental tapes that were made since the full-incremental.

If the installation desires two sets of backup tapes (to gain an extra measure of safety), the above procedure can be modified as follows:

Friday night: Run a /FULL-INCREMENTAL, followed immediately by

another /FULL-INCREMENTAL. This creates two sets of full-save tapes.

All other nights: Make two consecutive runs with /INCREMENTAL:2. This creates two sets of incremental save tapes, unless any file has been changed between saves.

If your installation has structures other than the public structure, it is important to back up files on all structures. It is advisable to use a different tape for each structure. Furthermore, it is important to use a different tape for every function: backup, migrate, and archive.

Example:

The following example shows a FULL INCREMENTAL SAVE of one directory.

```
DUMPER><RET>
[Using MTA-DUMPER:]
DUMPER>SSNAME Full Incremental of one directory<RET>
DUMPER>SAVE (DISK FILES) /FULL-INCREMENTAL PS:<MATO>*.*.*<RET>

 DUMPER tape #1, Fri 27-Jul-84 1339.  Saveset "Full Incremental of one directory",
volid TAPE2


 PS:<MATO>

 Pass 2, for Incremental Save, Starting.
 End of Pass 2.

 Total files dumped:            18
 Total pages dumped:            88
 Total directories dumped:      1

 Approx. CPU time, seconds:     2.34
DUMPER>
```

To notify users that you have SAVED, ARCHIVED, MIGRATED, or RETRIEVED files, use the /MAIL option to the LIST command and the MAIL command. Before you type your SAVE or RETRIEVE command, type

```
DUMPER>LIST /MAIL filespec<RET>
```

If you do not specify a filespec, DUMPER creates the file DUMPER-MAIL.TXT.

After you have completed your SAVE or RETRIEVE, type

```
DUMPER>MAIL (from list file) filespec<RET>
```

This sends mail to the users specified in the DUMPER-MAIL.TXT file.

If you do not specify a filespec with the MAIL command, DUMPER defaults to the file specified in the last LIST command.

You cannot interrupt a MAIL command with <CTRL/A> or <CTRL/E>.

7.6.2 Restoring Files and Directories from System Backup Tapes

If a user accidentally deletes files from the disk, he will want to restore them from backup tapes. In many installations, you as a privileged user or operator must restore these files.

To fill his request, mount the full save tape (of files saved on a weekly basis) and restore the requested files. Then, if necessary, mount the incremental save tape (of new or changed files saved on a daily basis) and restore the remainder of files in the user's request. The RESTORE command is the same as for a nonprivileged user. You must specify the source in order to transfer the files of the user who made the request.

Example:

```
     @DUMPER<RET>
     [Using MTA-DUMPER:]
     DUMPER>REWIND<RET>
     DUMPER>SKIP (NUMBER OF SAVESETS) 2<RET>

     Saveset "Saveset #2, PS:<MATO>"
     Saveset "Saveset #3, PS:<MATO>"
     Saveset, unnamed
     DUMPER>FILES<RET>
     DUMPER>RESTORE (TAPE FILES) PS:<MATO>*.*.* (TO)
     EXODUS:<PERLMA>*.*.*<RET>
     Saveset, unnamed
      Loading files into EXODUS:<PERLMA>
       PS:<MATO>DUMPER.EXAMPLE.1 to EXODUS:<PERLMA>DUMPER.EXAMPLE.1;P777777;AMONITOR [O
K]
       PS:<MATO>DUMPER.EXE.6 to EXODUS:<PERLMA>DUMPER.EXE.6;P777777;AMONITOR [OK]
       PS:<MATO>INIT.CMD.4 1 to EXODUS:<PERLMA>INIT.CMD.4 1;P777777;AMONITOR [OK]
       PS:<MATO>INT.MAC.1 to EXODUS:<PERLMA>INT.MAC.1;P777777;AMONITOR [OK
       PS:<MATO>LOGIN.CMD.139 to EXODUS:<PERLMA>LOGIN.CMD.139;P777700;AMONITOR [OK]
       PS:<MATO>MATO.LST.1 to EXODUS:<PERLMA>MATO.LST.1;P777777;AMONITOR [OK]
       PS:<MATO>MS.INIT.41 to EXODUS:<PERLMA>MS.INIT.41;P777700;AMONITOR [OK]
       PS:<MATO>NFT.INIT.10 to EXODUS:<PERLMA>NFT.INIT.10;P777700;AMONITOR [OK]
       PS:<MATO>QE5.TEC.5 to EXODUS:<PERLMA>QE5.TEC.5;P777777;AMONITOR [OK]
       PS:<MATO>TV.EXE.2 to EXODUS:<PERLMA>TV.EXE.2;P777777;AMONITOR [OK]
     End of Tape.

      Total files restored:     10
      Total pages restored:     77
     DUMPER>
```

If a user's entire directory is accidentally deleted and you saved the

directory with the CREATE option, you can restore it with the DUMPER command CREATE. Having entered DUMPER, type CREATE before RESTORE. As DUMPER restores all files and directories, it restores (creates) the deleted directory exactly as the directory was saved.

The TRANSFER command is another way to restore files to disk. If you type:

```
     DUMPER>TRANSFER<RET>
```

all files from all structures and directories on the tape are restored to your connected directory with the same name and type. The TRANSFER command always defaults to DSK*:<*>*.*.* for an input file specification and your connected directory for a destination specification.

7.6.3 Archiving Marked Files

If your installation is using the archive/virtual disk system for off-line storage of files, the installation establishes a schedule under which it runs the DUMPER program to copy files marked for archiving onto tape.

Mount a tape used only for archiving and run DUMPER using the ARCHIVE command.

```
     DUMPER>ARCHIVE STR:<*>*.*.*<RET>
```

If no file specification is given with the command, all files marked for archiving on the connected structure are transferred to tape. You can use wildcards to specify entire fields, as in A.*, but not portions of a field, as in A*.mem.

At the beginning of the archive run, DUMPER asks:

```
     Is this a new tape?
```

If you answer YES, DUMPER asks:

```
     Are you sure?
```

If you answer YES again, DUMPER writes at the beginning of the tape, erasing anything that is currently written on it. If you answer that this is not a new tape, DUMPER positions the tape after the last saveset and appends new files to the existing files.

When the marked files have been transferred, or the end-of-tape is reached, DUMPER indicates that the operation is complete and notifies you that Pass 2 has begun. Pass 2 is a check to determine if DUMPER should delete the file contents and update the archive status. Remove the first tape and mount a second tape. Type another ARCHIVE command. The purpose of this tape is to write the files again, verify them as on the first tape, and then delete the contents of the files from disk, unless the user has requested that the contents be retained. The files are then marked ;OFFLINE and set invisible in the user's directory.

To notify users that you have ARCHIVED files, use the /MAIL switch with the LIST command and the MAIL command. Refer to Section 7.6.1 for information on these commands.

Example:

```
$DUMPER<RET>
DUMPER>TAPE (DEVICE) T2:<RET>
DUMPER>REWIND<RET>
DUMPER>FILES<RET>
DUMPER>ARCHIVE (DISK FILES) PS:<TODAY>*.*.*<RET>
 Is this a new tape? YES<RET>
 Are you sure? YES<RET>

 DUMPER tape #1, Fri 2-Jul-84 1354.  ARCHIVE , volid T2

 PS:<TODAY>
   PS:<TODAY>QE5.LIB.1
   PS:<TODAY>QE5.TEC.1

 Pass 2 started.
 Pass 2 completed.


Total files dumped:       2
Total pages dumped:       4
CPU time, seconds:        0.35
DUMPER>EXIT
@
```

7.6.4 Migrating Files

According to the procedures of your installation, you can periodically migrate files (i.e., copy them onto tape and delete them from disk). This is not done at the user's request. It is a means of clearing disk space of files that have not been referenced within a specified period of time. It is also a technique for returning directory disk usage to within its quota.

You must specify a time period with the MIGRATE command to the REAPER program. When REAPER is run, it marks, for involuntary migration, all files that have not been used during that time period.

After you run REAPER, run DUMPER and use the MIGRATE command. This dumps all files marked for migration onto the mounted tape.

As with archiving, you use two tapes when migrating files. This provides a backup system in case one tape is bad or misplaced. DUMPER then follows the same routine as with archiving. It first asks if this is a new tape. When the files have been successfully migrated to the second tape, their contents are deleted from disk. The files then have an ;OFFLINE status in the user's directory.

To notify users that you have MIGRATED files, use the /MAIL switch with the LIST command and the MAIL command. Refer to Section 7.6.1 for information on these commands.

7.6.5 Retrieving or Restoring Archived and Migrated Files

Once a file has been archived or migrated, the user cannot access the file because its contents have been deleted from disk (unless he specified the subcommand RETAIN with the archive request). Only the FDB remains. The user can, however, request a retrieval of a file by using the TOPS-20 RETRIEVE command. This creates an entry in the system retrieval queue. (The queue can be displayed by typing the INFORMATION RETRIEVAL-REQUESTS command.) All retrieval requests are kept in order according to the archived or migrated tape information. This information consists of the volume identification, the tape saveset number, and the tape file number. When you are ready to process the retrieval request queue, run DUMPER and use DUMPER's RETRIEVE command:

```
DUMPER>RETRIEVE (FILES) file spec<RET>
```

If the file specification is omitted, DUMPER processes all requests in the queue. If a file specification is included, only those files that match the file specification are processed. (The file specification may include wildcards.) For example:

```
DUMPER>RETRIEVE (FILES) PS:<TODAY>QE5.LIB<RET>
```

When DUMPER selects a file or files for retrieval, it submits a mount request for the tape containing the file(s). After you or the operator have mounted that tape, DUMPER begins the retrieval process.

NOTE

When processing retrieval requests, you do not need to type the TOPS-20 command MOUNT TAPE or the DUMPER command TAPE. DUMPER automatically requests the tapes it needs.

If you cannot find the appropriate tape, type the OPR command CANCEL MOUNT-REQUEST. If the operator has to refuse the mount request, you receive the following message:

```
Mount refused by Operator
[Additional information-opr reason optional]
Try again?
```

If you answer YES, the same tape request is tried again, even if it was not available the first time.

If you answer NO, the following question appears:

```
Should I ask about this tape anymore during the run?
```

Answering NO means that any other requests for the tape are ignored.

Answering YES means that additional requests for the same tape are allowed.

Assuming all files can be found, only one RETRIEVE command is necessary to process the retrieval queue. If the files to be retrieved are on different tapes, DUMPER automatically unloads the mounted tape and submits a mount request for the next tape it needs.

NOTE

If you type RETRIEVE when there are no retrieval requests in the queue, DUMPER waits for approximately 5 minutes. DUMPER then stops the retrieval and sends the message

```
?Assuming no requests in the retrieval queue.
```

Whenever possible, check the queue (with the TOPS-20 command INFORMATION RETRIEVAL-REQUESTS) before typing RETRIEVE.

Example:

```
$RETRIEVE (FILES) PS:<TODAY>QE5.LIB<RET>
 QE5.LIB.1 [OK]
$DUMPER<RET>
DUMPER>RETRIEVE (FILES)<RET>
[Mounting tape volume T2]
```



```
[Volume T2 mounted]
  PS:<TODAY>QE5.LIB.1

Total files restored:     1
Total pages restored:     1
```

If a user deletes the file name from his directory, and therefore the file descriptor block (FDB), and then wishes to retrieve the file, you must use the RESTORE command. When you locate the file, you can restore it with or without the tape information. This information indicates whether the file is archived or migrated, and if so, onto which tapes. If you do not wish to restore this information with the file, type the command RESTORE/NOTAPE-INFORMATION. To reinstate the transfer of this information with the file, type the command RESTORE/TAPE-INFORMATION. RESTORE/TAPE-INFORMATION is the DUMPER default.

To notify users that you have RETRIEVED files, use the /MAIL switch with the LIST command and the MAIL command. Refer to Section 7.6.1 for information on the commands.

7.7 DUMPER COMMANDS

This section contains an alphabetical list of all DUMPER commands. Each listed command includes a brief description, command type (status-setting, action, or tape-positioning), and description of optional arguments. Where applicable, there is an indication that the command is for privileged users only.

[NO] ABEFORE (DATE AND TIME) date time — Status-Setting

Saves or restores only files that were typed, printed, or read (as maintained by .FBREF) before the specified date and time. Default: Date - none; Time - 00:00:01

ABORT — Action

Cancels an interrupted CHECK, RESTORE, SAVE, RETRIEVE or tape-positioning command and allows you to issue a new action command. ABORT can be used only after you have typed the <CTRL/E> command. The ABORT command does not reposition the tape.

ACCOUNT (OF RESTORED FILES FROM) argument — Status-Setting

Restores files with either the system (SYSTEM-DEFAULT) account or

the account stored with the file (TAPE).
Default: TAPE

ARCHIVE (DISK FILES) Action

Saves files that have been marked for offline storage. Archiving is voluntary on the part of the user.

(Privileged User Only)

[NO] ASINCE (DATE AND TIME) date time Status–Setting

Saves or restores only files that were typed, printed, or read (as maintained by .FBREF) since the specified date and time.
Default: Date – none; Time – 00:00:01

[NO] BEFORE (DATE AND TIME) date time Status–Setting

Saves or restores only files that were created or modified (as maintained by .FBCRV and .FBWRT) before the specified date and time.
Default: Date – none; Time – 00:00:01

CHECK (ALL TAPE FILES) Action

Checks every File Descriptor Block (FDB) in the current saveset against the FDBs of the corresponding files on disk. Corresponding files must have the same name, structure, directory, type, and generation number for the check to be made. The CHECK command cannot be used with the date and time commands. (Refer to Table 7–3 for a list of possible differences.)

[NO] CHECKSUM (FILES) type Status–Setting

Activates or suppresses checksumming during the PRINT command. Two types may be specified: SEQUENTIAL (for INTERCHANGE mode) or BY–PAGES (that checks every word of every page).
Default: NO CHECKSUM

CONTINUE Action

Continues a CHECK, RESTORE, RETRIEVE, or SAVE, after you have interrupted the command with <CTRL/E>.

[NO] CREATE (DIRECTORIES FROM TAPE DATA) Status–Setting

Creates or does not create user directories from directory information on the tape.
Default: NO CREATE.

(Privileged User Only)

<CTRL/A> Action

Prints one line of status information. The information includes the command in process, and, for certain operations, the file and disk pages DUMPER is currently processing.

<CTRL/E> Action

Halts the action of a CHECK, RESTORE, RETRIEVE, SAVE, or tape–positioning command. DUMPER responds with INTERRUPTING... and its prompt. You then issue the ABORT, CONTINUE, or any status–setting command.

DENSITY (OF MAGTAPE) n Status–Setting

Sets the tape density to the given number of bits per inch (bits/in): 200, 556, 800, 1600, 6250, or JOB–DEFAULT (set by the system command SET TAPE DENSITY). If no DENSITY command is given, DUMPER uses the job default density on the first tape. This command has no effect on Labeled tapes.
Default: The density listed in the TOPS–20 command INFORMATION (ABOUT) TAPE–PARAMETERS.

[NO] DIRECTORIES Status–Setting

Reactivates or suppresses printing, on your terminal, directory names as DUMPER saves or restores each directory.
Default: DIRECTORIES

EOT Tape–positioning

Positions the mounted tape at the end of the last saveset written on the tape. DUMPER prints all existing saveset names and the message:

End of Tape

[NO] EXACT Action

Saves or restores files without translating logical names into

actual structure names.
Default: NO EXACT

EXIT — Action

Exits to TOPS-20 command level. (Same as the QUIT command.)

[NO] FILES — Status-Setting

Reactivates or suppresses printing file specs on your terminal, as DUMPER saves or restores each file.
Default: NO FILES

FORMAT (VERSION NUMBER IS) n — Status-Setting

Allows DUMPER to read tapes written with previous versions of DUMPER.
Default: Version 6

HELP — Action

Prints a list of all valid DUMPER commands on your terminal.

INITIAL (FILESPEC) file spec — Status-Setting

Begins a SAVE with the specified file.

[NO] INTERCHANGE (FORMAT) — Status-Setting

Allows or does not allow DUMPER to read tapes written with the TOPS-10 BACKUP program or to write tapes to be read by the TOPS-10 BACKUP program. (INTERCHANGE should not be used when writing tapes to be read by another TOPS-20 system.)
Default: DUMPER format

[NO] LIST (LOG INFORMATION ON FILE) file spec — Status-Setting

Prints or does not print a list, in the specified file, of all files as DUMPER saves them.
Default: NO LIST
Default file spec: LPT:DUMPER.LOG

The LIST/MAIL command creates a DUMPER-MAIL.TXT file that is used with the MAIL command to notify users that you have completed the SAVE or RESTORE operation.

(Privileged User Only)

Default: /NOMAIL

MAIL file spec — Action

Sends mail notifying a user that a SAVE, ARCHIVE, MIGRATE or RETRIEVE has been completed.

Default: Last file name used in the LIST/MAIL command.

(Privileged User Only)

[NO] MBEFORE (DATE AND TIME) date time — Status-Setting

Transfers only files modified (changed, created, appended, or renamed as maintained by .FBCRE) before the specified date and time.
Default: Time=00:00:01

MIGRATE — Action

Saves files that have been marked for involuntary offline storage by the REAPER program.

(Privileged User Only)

[NO] MSINCE (DATE AND TIME) date time — Status-Setting

Transfers only files modified (as maintained by .FBCRE) since the specified date and time.
Default: Time=00:00:01

NO DATES — Status-Setting

Disables all the date and time commands at once. The date and time commands are ABEFORE, ASINCE, BEFORE, MBEFORE, MSINCE, and SINCE.

PARITY (OF MAGNETIC TAPE) parity — Status-Setting

Sets the parity of the mounted tape to EVEN or ODD.
Default: The parity listed in the TOPS-20 command INFORMATION (ABOUT) TAPE-PARAMETERS

PRINT (DIRECTORY OF TAPE ONTO FILE) destination — Action

Records a printed list of file names in the specified file. The list contains file specifications of all existing files on the entire tape beginning at the current saveset.
Default: Prints on your terminal

PROTECTION (OF RESTORED FILES FROM) argument — Status-Setting

Restores files with SYSTEM-DEFAULT protection or protection taken from TAPE.
Default: TAPE (if the PROTECTION command is omitted).
Argument Default: None; you must specify one or the other.

QUIT — Action

Exits to TOPS-20 command level. (Same as the EXIT command.)

RESTORE (TAPE FILES) source (TO) destination — Action

Restores the magnetic tape source file(s) to disk with the given destination file specifications. If you are restoring files from a directory other than your connected directory, you must specify that directory in the source file specification. If files exist in the destination directory with the same names and types as the files on tape, DUMPER restores the files according to the specification of the SUPERSEDE command: ALWAYS, NEVER, OLDER
Default: Source file spec = all files in the current saveset that were saved under the connected structure and directory; if you are privileged, all files in all directories on your connected structure.
Destination file spec = same names and types as the files on tape.

(Privileged User Only):

The FDB of each file contains information that reflects whether a file is archived or migrated, and, if so, the volume identifiers of the DUMPER tapes that contain the file. The /TAPE-INFORMATION switch instructs DUMPER to restore this information to the FDBs of the files that are being restored. /NOTAPE-INFORMATION requests that this information not be restored.
Default: /TAPE-INFORMATION

RETRIEVE (FILES) file spec — Action

Instructs DUMPER to process the requests in the system's file retrieval queue.
Default: Process all files in the queue

(Privileged User Only)

REWIND argument — Tape-Positioning

Positions the specified tape at the beginning of the tape. If the argument is CURRENT-VOLUME, DUMPER rewinds the currently mounted tape to the beginning. If the argument is SWITCHING (TO VOLUME NUMBER) n, DUMPER releases the currently mounted tape, requests volume n of the multi-reel tape set, and rewinds it to the beginning.
Default: CURRENT-VOLUME

SAVE (DISK FILES) source (AS) destination — Action

Saves the disk source file(s) onto magnetic tape. If all files cannot fit on one tape, DUMPER requests that the next volume in the set be mounted.
Default: Source file spec = all files in your connected directory; if you are privileged, all files in all directories on your connected structure.
Destination file spec = same names and types as the files on disk. (Privileged User Only):

The following switches can be appended to the SAVE command:

/FULL-INCREMENTAL for a system backup of all files. (This also marks all files as having been saved.)

/INCREMENTAL:n for a daily backup of files that either have not been saved on at least n tapes, or have been modified since the last INCREMENTAL or FULL-INCREMENTAL run.
Default: /INCREMENTAL:1

/NOINCREMENTAL for a backup of all files. /NOINCREMENTAL overrides the CREATE command.

/UNLOAD to request that DUMPER unload the tape after the SAVE is complete.

Default: /NOINCREMENTAL

NOTE

Unless one of these switches is used, directory information is not saved on the tape.

SET BLOCKING-FACTOR (TO) n (RECORDS) — Status-Setting

Sets the number of logical records per physical record written on tape by DUMPER. The range must be between 1 and 15.
Default: 1 record

SET TAPE-NUMBER (DECIMAL NUMBER) n Status-Setting

Assigns a number to a new tape when you continue a restore after a crash, or when you restore a file nonsequentially from a multi-reel saveset.

[NO] SILENCE Status-Setting

Activates or suppresses the printing of directory names and file specifications on your terminal, as files are saved or restored. NO SILENCE is equivalent to FILES and DIRECTORIES.
Default: NO FILES, DIRECTORIES

[NO] SINCE (DATE AND TIME) date time Status-Setting

Transfers only files created or whose contents were changed since the specified date and time.
Default: Time = 00:00:01

SKIP (NUMBER OF SAVESETS) n Tape-Positioning

Skips over n savesets. The skip can be directed forward (n), backwards (-n), or to the beginning of the current saveset (0). Labeled tapes do not support zero or backward (-n) skips.

SSNAME name Status-Setting

Specifies the name to be written in the saveset header on the tape. The name can be any string of up to 200 characters. It is printed on your terminal whenever you save or restore files.
Default: Existing saveset name, if any.

SUPERSEDE condition Status-Setting

Sets the condition under which DUMPER rewrites a disk file with a magnetic tape file of the same name and file type. You must specify one of three conditions:

1. ALWAYS - to supersede the file on disk regardless of the last write date or generation number of that file.

2. NEVER - to never supersede the disk file; the specified file is written to disk only if there is no existing disk file with the same name and file type.

3. OLDER - to supersede the disk file only when the file on tape is newer (i.e., has a later write date and/or a higher generation number than the file on disk).

Default: SUPERSEDE OLDER (if the SUPERSEDE command is omitted)

TAKE (COMMANDS FROM FILE) file spec Action

Instructs DUMPER to execute commands from the specified command file. Commands are executed until the end of the command file is reached or until DUMPER encounters an error. Commands are not echoed on the terminal as they are processed.

TAPE (DEVICE) name: Status-Setting

Specifies the tape device to be used for file transfers. The name can be either a physical device (MT1:) or a logical name (TAPE1:).

NOTE

If you define the mounted tape as MTA-DUMPER:, you can omit the DUMPER command TAPE. If you use the RETRIEVE command you do not need to use the TAPE command.

TRANSFER Action

Restores files from tape to disk. The TRANSFER command defaults to DSK*:<*>*.*.* for an input file specification and your connected directory for a destination specification.

UNLOAD

Used to rewind a tape onto the source reel if your installation does not have tape drive allocation enabled.

7.8 DUMPER MESSAGES

This section contains an alphabetical listing of all messages generated by DUMPER. Included with most messages are a description and a suggested user response. The messages printed by DUMPER fall into three general categories: warning messages, fatal errors, and

those requiring some action. Warning messages are preceded by a percent sign (%) and indicate that something unexpected occurred, but that DUMPER was able to recover. In this case, verify that the process in progress at the time of the warning is correct. Fatal errors are preceded by a question mark (?), and indicate an occurrence that DUMPER could not handle. In this case, DUMPER aborts the operation, and you must fix the problem before reissuing your command string.

In cases where an internal or system problem results in an error message, the best (and usually only) way to deal with the internal problem is to contact your Software Specialist or submit a Software Performance Report (SPR) to DIGITAL.

Some of the messages contain information that is dependent on the exact command string or file you specified. These message variables are as follows:

<action>	A suggested course of action.
<cmd>	A DUMPER command.
<dev>	A device name.
<dir>	The name of a directory.
<file>	A file specification.
<n> or <m>	The number of a page or record, or other integer.
<reason>	The reason for the error.

[Additional information − <reason>]

Description: The Operator gave a reason for a tape not being available.

Suggested User Response: Read the reason given and follow the action suggested.

?Assuming no requests in the retrieval queue

Description: A RETRIEVE request was started but QUASAR, a GALAXY component, never sent and retrieval requests. DUMPER sends this message after waiting approximately 5 seconds.

Suggested User Response: Use the INFORMATION (ABOUT) RETRIEVAL−REQUESTS at TOPS−20 level to see if there are any requests to be processed before issuing a DUMPER RETRIEVE command.

[At end of tape]

Description: DUMPER wrote or read to the end of the tape. The next message DUMPER types indicates what should be done.

Suggested User Response: Do what the DUMPER message indicates. Most often, you will have to mount a tape.

%Bad checksum, record <n>

Description: A record was read with a bad checksum.

Suggested User Response: Make a note of the record number and file being restored. If the file is not restored properly, try restoring again, possibly from another tape or on a cleaner drive.

%Bad definition for MTA−DUMPER:, ignored

Description: DUMPER found MTA−DUMPER defined, but the definition did not refer to a tape drive.

Suggested User Response: Exit DUMPER and either redefine MTA−DUMPER or undefine it.

%Bad physical record length, record <n>

Description: The record DUMPER just read does not appear to be a DUMPER record.

Suggested User Response: You may be encountering one of the following conditions: reading past the end of an incomplete saveset; you have a damaged tape; you have a dirty or misaligned tape drive or, the tape was not written by DUMPER or BACKUP. Check these conditions before attempting the command again.

[Before and since commands are still in effect]

Description: This is an informational message telling you that date and time commands you used during other DUMPER operations are still in effect.

Suggested User Response: If you want to keep the commands, ignore this message. Otherwise, disable or change the commands.

?Can’t change tape settings mid−tape, please rewind first

Description: You tried to change the tape density or parity after you started to read or write the tape.

Suggested User Response: REWIND the tape before you change tape density or parity.

?Can't open magtape − <reason>

Description: DUMPER is unable to access the drive in the mode it needs to process your command.

Suggested User Response: The reason for this message is usually one of the following: the drive is offline, the drive is write−protected, or the device is assigned to another job. Fix the condition and try again.

%Can't read <file> − <reason>

Description: DUMPER tried to dump a file to tape but found it could not open it for read access. DUMPER has tried both read and read unrestricted.

Suggested User Response: You may not have read access for the file or the file may be corrupted. It is also possible that you have not enabled your privileges. If this is the case, enable your privileges and try again.

?Can't step to next file − <reason>

Description: DUMPER cannot save any more files because there is damage to the disk. The command is aborted.

Suggested User Response: Contact Digital Field Service.

?Can't switch to next tape volume − <reason>

Description: The next tape in the tape set is not available.

Suggested User Response: The command is aborted when you receive this message. If you were performing a write operation, the saveset will have to be rewritten after the problem is fixed.

%Data write error, record <n>

Description: DUMPER wasn't able to write a record properly.

Suggested User Response: You do not have to take any action when you receive this error. DUMPER leaves the bad record on tape and tries to write a duplicate record as the next record.

%Deleting <file> while superseding

Description: DUMPER is warning you that files had to be deleted to perform a SUPERSEDE ALWAYS during a RESTORE.

Suggested User Response: This is an informational message. You do not have to take any action. If you want to recover the deleted file, try the UNDELETE command after the restore is

finished and see if the file can be undeleted. If it cannot be undeleted, the file is lost.

?Device must be DISK

Description: For a RESTORE or SAVE command, you specified a device other than DISK.

Suggested User Response: Specify a device as the device and try again. DUMPER only saves and restores disk files.

?Directory <dir> not created − <reason>

Description: DUMPER could not create the directory specified in <dir>.

Suggested User Response: Read the reason to determine what to do to correct the problem.

%Directory specifications differ − not saving directory info on: <dir>

Description: You are using the CREATE command with the SAVE command and your destination directory file specification does not include your source directory.

Suggested User Response: If you use the CREATE command with the SAVE command, make sure that your destination directory file specification includes your source directory. You may use a wildcard for your destination directory. The CREATE command is used for tapes that will be used to recreate directories.

?DUMPER doesn't support that tape format

Description: You set a tape format to a value less than 4

Suggested User Response: If you have a tape with a format of less than 4, the tape was not written by a Digital supported version of DUMPER. Either do not use the FORMAT command or else use the FORMAT 4 command.

[Ending <file>]

Description: DUMPER is informing you that it has finished reading a TAKE command file.

Suggested User Response: You do not have to take any action. If you do not want to see this message, end your command file with TAKE followed by a carriage return <RET>.

?EOT on first record, try a rewind

Description: DUMPER encountered an end of tape record or mark when it first started reading.

Suggested User Response: First try REWIND. If the REWIND doesn't succeed, it is likely that the tape you are using has just been initialized and has never been written to.

%<error> – <file>

Description: DUMPER is not saving a file. The error is given before the name of the file.

Suggested User Response: Correct your file command according to the reason given and reissue the command.

?Error writing LIST file, list file ended

Description: DUMPER encountered problems while writing the LIST file and has aborted the file. The command continues to process but the list file ends.

Suggested User Response: Try another list file or check your directory quotas to make sure you have enough space for the LIST file.

%Excessive retries in writing record, continuing...

Description: DUMPER encountered problems in writing a record. DUMPER wrote the record multiple times and is now moving to the next record.

Suggested User Response: Note the record that did not get written properly. Previous error messages tell you which record it is. Since the problem may be due to a bad spot on the tape, you may want to perform another save on a new tape.

%Failed to create <dir> – <reason> – RETRYING

Description: DUMPER encountered the problem described in <reason> while trying to create a directory. DUMPER tries to correct the error and create the directory again.

Suggested User Response: You do not have to take any action. DUMPER will try to correct the problem. The directory created may be missing some information. The <reason> portion of the message indicates what did not work.

%Failed to restore <file> because: <reason>

Description: A RETRIEVE command did not work properly.

Suggested User Response: Read the reason and correct the

problem. Then try the command again.

%File <file> needed to be opened unrestricted.

Description: DUMPER tried to dump a file to tape but could not open the file for READ. UNRESTRICTED READ, however, worked.

Suggested User Response: Try to avoid dumping files to tape that might be opened for write. The error message is caused by someone writing to the file while you were doing a SAVE. This message has information only. It indicates that a saved file may be different.

%File <file>not found

Description: DUMPER, while performing a CHECK, found a file on tape but could not find the corresponding file on disk.

Suggested User Response: Check to see if the file is on disk. It may have been deleted.

?Illegal data mode or density for this controller

Description: The tape drive does not work with the specified density or mode.

Suggested User Response: Use the system default for density or mode if you encounter this problem.

?Illegal file specification

Description: You typed an illegal file specification.

Suggested User Response: Retype the command with the correct file name.

?Illegal LIST file choice

Description: DUMPER encountered a problem with your LIST file

Suggested User Response: Be sure you are listing files to a disk file or terminal and not to the tape you are saving.

?Illegal PRINT file choice

Description: DUMPER encountered a problem with the PRINT file you specified.

Suggested User Response: Be sure you have specified a PRINT file that will be written to disk or terminal and not to the tape you are reading.

?Illegal tape type

Description: TOPS−20 only supports TOPS−20 and ANSI labeled or unlabeled tapes.

Suggested User Response: Check to make sure you are using an unlabeled tape, a TOPS−20 or an ANSI tape and try the operation again.

?Illegal to read a labeled tape this way

Description: Tape labels are being sent directly to DUMPER

Suggested User Response: Either do not read the tape on an assigned drive, or do not specify /LABEL:BYPASS when you mount the tape.

%Illegal value for format, assuming 4

Description: DUMPER does not support the value written on the tape as the format value. The tape may not have been written by DUMPER or the tape may be damaged.

Suggested User Response: This message needs no action. If you see many error messages following this message, you may have a damaged tape or a misaligned tape drive.

?In command <cmd>

Description: DUMPER is reading a command from a TAKE file and cannot process the next command. The information in brackets is the command that DUMPER could not process.

Suggested User Response: Correct the command indicated in the <cmd> field of the error message and try the command again.

?In file <file>

Description: DUMPER is reading a TAKE file and cannot process the next command. The information in brackets is the file containing the command that DUMPER could not process.

Suggested User Response: Correct the command indicated in the <file> field of the error message and try the command again.

?INTERCHANGE tapes of BLOCKING−FACTOR other than 1 are illegal
Description: DUMPER cannot read the tape. The tape has not been written properly for an interchange tape. Suggested User Response: Rewrite the tape using BACKUP with a BLOCKING−FACTOR of 1.

[Interrupt ignored]

Description: You typed a <CTRL/A> or a <CTRL/E> as DUMPER was completing a command.

Suggested User Response: You do not need to take any action. DUMPER ignored the <CTRL/A> or <CTRL/E>.

?JFN LIST overflow
?<error> − No more JFNs available

Description: You have specified too many filenames in one command. DUMPER allows approximately 30 filenames per command. A wildcard specification, however, is counted as one filename.

Suggested User Response: Correct your command file by either removing some of the filenames or using a wildcard and try the command again.

?May not change INTERCHANGE state mid−tape, please REWIND first

Description: DUMPER cannot read or write tapes that have both INTERCHANGE and NO INTERCHANGE savesets.

Suggested User Response: Correct your command and try the command again.

?May not read from this saveset without WHEEL or OPR

Description: Privileges are required to read this saveset.

Suggested User Response: Enable your privileges, reposition the tape, and try the command again.

[Mounting next tape volume]

Description: DUMPER is ready to read the next tape in the set.

Suggested User Response: Mount or have the operator mount the next tape volume in the saveset.

[Need to mount next retrieval tape]
Provide the volid of the next retrieval tape in the set

Description: DUMPER is doing a RETRIEVE and needs the rest of the file, which is on another tape.

Suggested User Response: Type the volid (volume id) of the next tape in the set. DUMPER then mounts this tape and finishes the RETRIEVE.

%No files dumped

Description: The SAVE command did not dump any files to tape.

Suggested User Response: Check your SAVE command to make sure you have typed it correctly.

?No freespace

Description: This message indicates a problem internal to DUMPER.

Suggested User Response: Contact your Digital Software Service Representative.

?No such <cmd> option

Description: You typed an option to DUMPER command that DUMPER doesn't understand.

Suggested User Response: Check our command line for the error. You can type HELP to see the options that DUMPER accepts. Then try the command again.

?Not a defined command

Description: You typed a command incorrectly or you typed a command that is not a DUMPER command.

Suggested User Response: Correct your command and try it again.

%Not loading <file> – <reason>

Description: This message means that DUMPER is not restoring a file that you specified in a RESTORE command.

Suggested User Response: Check to make sure that this file has not been ARCHIVED. If the file has been ARCHIVED, the RESTORE command will not work.

?Only one file specification is allowed

Description: Some DUMPER commands accept only one file specification. You have typed a multiple file specification and DUMPER cannot process it.

Suggested User Response: Correct the problem and issue the command again.

%Requeuing <file>

Description: DUMPER was not able to finish a RETRIEVE from a specific tape.

Suggested User Response: You do not need to take any action. DUMPER requeues the request and tries to get the file from

another tape.

[Restoring BLOCKING–FACTOR to <n>]

Description: You enabled INTERCHANGE mode at some point and DUMPER set the BLOCKING–FACTOR to 1. Your previous BLOCKING–FACTOR was saved. You have since disabled INTERCHANGE mode and DUMPER is reverting to your previous BLOCKING–FACTOR.

Suggested User Response: You do not have to take any action unless you wish to change the BLOCKING–FACTOR to another value.

%Retrieve aborted

Description: QUASAR, a GALAXY component, aborted a RETRIEVE action.

Suggested User Response: A user probably canceled his request. No action is necessary.

%Sequence error, <n> after <m>

Description: DUMPER encountered an error reading the tape. This could be caused by a damaged tape. Data may be missing from the file being restored. It is also possible that you are reading from an incomplete saveset.

Suggested User Response: If you are sure that the tape was written properly, and DUMPER does not report any other tape errors, submit a Software Performance Report (SPR).

[Starting from <file>]

Description: This message is generated by the SAVE command after an INITIAL command was given. The name specified in <file> will be the first file saved on the tape.

Suggested User Response: There is no action you need to take. This is confirmation of your INITIAL command.

%Structure not mounted, skipping file <file>

Description: DUMPER tried to retrieve a file that was on an unmounted structure.

Suggested User Response: Mount the structure and try the command again.

?TAKES nested too deeply, aborting

Description: You had TAKE commands nested too deeply.

Suggested User Response: Remove the last TAKE in the series of TAKES in your command file and try your command file again, or you can continue manually from the point that your command file aborted.

?Tape blocking-factor is already set to <n>, please rewind first

Description: You tried to change the tape blocking-factor after you began to read or write a tape.

Suggested User Response: Set the blocking-factor before you write a tape. Do not try to change it in the middle of a tape.

?Tape is write-protected. <action>

Description: The tape is write locked and you typed a SAVE command.

Suggested User Response: Place a write ring in the tape and mount the tape /WRITE-ENABLED.

?Tape number incorrect (wrong tape mounted)

Description: DUMPER is telling you that the tape just mounted is not the next tape in the series.

Suggested User Response: Put the tapes in the proper order and try the command again.

?Tape number must be positive

Description: You have typed an invalid number, such as 0, for a tape number.

Suggested User Response: Specify a tape number that is a positive number.

%Tape went offline, <action>

Description: The tape drive went offline while DUMPER was using it.

Suggested User Response: Do what is indicated in the action portion of the message. You may have to remount the tape or put the drive back online. If a question mark (?) precedes this message, you have to start your command over.

?That BLOCKING-FACTOR is illegal

Description: You specified a value that was not between 1 and 15. DUMPER accepts values only between 1 and 15.

Suggested User Response: Specify a value between 1 and 15 and try your command again.

?That requires WHEEL or OPR privs

Description: You tried to perform a privileged command without enabling your privileges.

Suggested User Response: Enable your privileges or do not attempt to issue the command.

?The <cmd> command will not be legal until ABORT <cmd> is typed.
a command is interrupted by CTRL/E.

Description: You typed a command that is either not available until after a <CTRL/E> is issued, or the command is not available at the interrupt prompt.

Suggested User Response: Before you type a new action command to an interrupted command, be sure to type the ABORT command.

%The date and time given have not yet occurred

Description: You entered a date and time in one of the date and time commands that hasn't occurred yet.

Suggested User Response: Set the time to a time that has passed. If you leave the command as you have typed it, no files are transferred.

%This appears to be a BACKUP tape, turning on INTERCHANGE mode

Description: To read this tape, INTERCHANGE mode must be turned on.

Suggested User Response: You do not need to take any action. DUMPER turns on INTERCHANGE mode for you.

%This appears to be a DUMPER tape, turning off INTERCHANGE mode

Description: To read this tape, INTERCHANGE mode must be turned off.

Suggested User Response: You do not need to take any action. DUMPER turns off INTERCHANGE mode for you.

?This doesn't appear to be a DUMPER or BACKUP tape, <action>

Description: DUMPER could not read the tape.

Suggested User Response: Rewind the tape and try to read it again. If this does not work, check to make sure you are using a

DUMPER or BACKUP tape.

%This is a labeled tape with labels passed

Description: Tape labels are being passed directly to DUMPER. This is allowed only when privileges are enabled.

Suggested User Response: Only do this is GALAXY is unavailable and files must be restored. This is not a Digital-supported procedure.

?This tape is full. Please mark it.

Description: DUMPER found there was no more room on the tape.

Suggested User Response: Mark the tape full, and continue with a new tape.

%Unrecovered data error, record <n>

Description: DUMPER encountered an error reading the tape.

Suggested User Response: Check the file being restored when the RESTORE is complete. If there are errors, try to restore it again from another tape or on a cleaner drive.

%User directory is no longer valid, <file>

Description: DUMPER tried to retrieve a file and did not find the directory. The file is not restored.

Suggested User Response: If you need the file, do a RESTORE.

CHAPTER 8

PLEASE

8.1 INTRODUCTION

The PLEASE program allows you to communicate either with your system operator or with a remote-node operator. For instance, you may ask the operator to perform a task or ask for information about your job or the system, or give the operator information about your job.

NOTE

The PLEASE command that you use in a batch job does not function like the timesharing PLEASE program. For information on using the PLEASE command in a batch job, refer to the TOPS-10/TOPS-20 Batch Reference Manual.

8.2 SWITCHES USED WITH PLEASE

The PLEASE program has two switches:

/HELP	prints information about this program on your terminal
/NODE:node-name::	specifies the node name of the operator, other than your system operator, that you wish to communicate with. You must terminate the node name with two colons (::).

8.3 MESSAGE TERMINATORS USED WITH PLEASE

<RET>	the only terminator for a message on the PLEASE command line. (The message must be limited to a single line.)

<ESC>	a valid terminator only for a message being sent in dialogue mode. It indicates that you do not expect a reply, and returns you to system level.
<CTRL/Z>	a valid terminator for a message being sent in dialogue mode. It indicates that you want to wait for a reply from the operator.

8.4 RUNNING PLEASE

To send a message to your own system operator, you can use PLEASE in either the DIALOGUE or MESSAGE mode.

The simplest way to enter DIALOGUE Mode is to type PLEASE, followed by a one-line message, and then press RETURN. The PLEASE program acknowledges the message and notes the time your operator received it. You then see your operator's reply. That is followed by a prompt from PLEASE, which indicates that you are now in DIALOGUE mode and can respond with one or more additional messages, as in the following example:

```
@PLEASE When is the system scheduled to go down?<RET>
[PLSOPN Operator at KL2102 has been notified at 10:04:42]
10:05:58  From Operator at terminal 3
        => At noon
Enter text, terminate with CRTL/Z to wait for response
Or ESC to send message and Exit
Thank you <ESC>
@
```

To terminate DIALOGUE mode, you press ESC after your final reply to the operator in order to return to the system level. In this case, the user has chosen to end the interchange by thanking the operator and typing <ESC> to return to the system level.

The second way to communicate with your own operator in DIALOGUE Mode is to type PLEASE and then press RETURN. You enter DIALOGUE mode immediately and receive the PLEASE prompt for your message, as

```
@PLEASE<RET>
Enter text, terminate with CTRL/Z to wait for response
Or ESC to send message and exit<RET>

When is the system scheduled to go down?<CTRL/Z>
```

To communicate with a remote-node operator, type PLEASE, the /NODE switch, and the node name of the remote operator, and press RETURN to enter dialogue mode and receive the PLEASE prompt for your message, as

```
@PLEASE/NODE:node-name::<RET>
Enter text, terminate with CTRL/Z to wait for response
Or ESC to send message and exit<RET>
```

From the point where you enter DIALOGUE mode and receive the message prompt, messages to both your own operator and to a remote-node operator follow the same format.

If you do not need a reply from the operator, use MESSAGE mode. After you have accessed PLEASE and typed your message, type <ESC> to end the message. PLEASE acknowledges that the message has been sent and records the time the operator received it, before returning you to the system level, as

```
@PLEASE I am leaving here at 4 today<ESC>
[PLSOPN Operator at KL2102 has been notified at 11:00:03]
```

In this example, you have sent a message to the operator and have been immediately returned to the system level. You are not requesting an immediate reply.

8.5 PLEASE MESSAGES

Following is an alphabetized list of the PLEASE messages. Informational messages are enclosed in brackets ([]). Warning messages are preceded by a percent sign (%); for these, processing will continue but perhaps not in the way you intended. Fatal error messages are preceded by a question mark (?); such messages may terminate the program.

Each message is followed by a brief explanation of the problem you may encounter, which may in itself tell you what you need to do to correct it. In most cases, simply trying the procedure again is sufficient to correct the problem. In some cases, though, you may need to call your Software Specialist.

?PLSBHR - Bad help request, use PLEASE/HELP

Reminder: Use no other text after PLEASE/HELP.

?PLSCME - Command error

Description: You typed an invalid command.

?PLSEFO - Error from ORION

Description: ORION, which is the message dispatcher, has received a

message from PLEASE which it cannot send on to the operator. Possibly you typed a message containing format errors (size, text, etc.)

?PLSEPM - Error in parsing message

Description: You terminated a PLEASE command line improperly. The only legal command-line terminator is carriage return <RET>.

%PLSNHA - No help available

Description: The program cannot find the HLP:PLEASE.HLP file. It may not be in the right place and hence not in the system search list.

Suggested user response: Check whether the file was properly taken off the distribution tape, or whether it is incorrectly protected from you, or whether there may be a physical device error.

?PLSNIN - Node name user typed not in network, or specified without trailing double colon

Description: You either specified a node name that does not exist or you improperly terminated a node name.

%PLSNOP - No operator in attendance

Description: The system is unattended. However, your message will be sent to the operator, and PLEASE will notify you at what time the operator received it.

[PLSOPN - Operator at [node:name] has been notified at [time]]

Description: PLEASE simply notifies you that the operator has received your message.

?PLSSSE - Switch syntax error

Description: you gave a command that contains an illegal switch, or a switch delimiter (a slash) with no switch name following it.

?PLSSUT - Switch used twice

Reminder: You can use a switch only once in a given command.

?PLSUMO - Unrecognized message from ORION

PLEASE

Description: ORION has responded with an unknown message instead of the response you expected from the operator. Perhaps you are running an old version of PLEASE. (Check the title page of this manual for the current version number.)