Tape Subsystem User’s Guide

SG–2051 10.0

Portions of this product may still be in development. The existence of those portions still in development is not a commitment of actual release or support by Cray Research, Inc. Cray Research, Inc. assumes no liability for any damages resulting from attempts to use any functionality or documentation not officially released and supported. If it is released, the final form and the time of official release and start of support is at the discretion of Cray Research, Inc.

New Features

Tape Subsystem User's Guide SG–2051 10.0

With the UNICOS 10.0 release, this manual documents the tape subsystem for both the UNICOS and UNICOS/mk operating systems. Its title, *UNICOS Tape Subsystem User's Guide*, has been changed to the *Tape Subsystem User's Guide*, Cray Research publication SG–2051. Any usage differences between the operating systems are highlighted in the text.

This revision documents GigaRing support; the new `MTIOCATTR` `ioctl`(2)flags; `MTIOCTOP` `ioctl`(2) code, `MTMSG`; new error code, `ETQRY` (90122); new and revised system messages; and man page updates. Miscellaneous editorial and technical changes were also made.

Tapelist I/O is no longer supported.

Record of Revision

Version	*Description*
	June 1988. Original Printing.
5.0	March 1989. Update and rewrite to support the UNICOS 5.0 release running on Cray Research computer systems.
6.0	December 1990. Reprint with revision to support the UNICOS 6.0 release running on Cray Research computer systems.
7.0	April 1992. Reprint with revision to support the UNICOS 7.0 release running on Cray Research computer systems.
7.0	November 1992. This restrock incorporates the corrected pages from the *UNICOS 7.0 Publications Errata*, publication ER-2124 7.0, which reflect the grave accent printing problem.
8.0	January 1994. Rewrite to support the UNICOS 8.0 release running on Cray Research computer systems.
8.1	May 1994. Updated online to remove CRAY X-MP, CRAY-2, IOS-B, and IOS-C support.
8.3	November 1994. Updated online to support the UNICOS 8.3 release.
9.0	August 1995. Rewrite to support the UNICOS 9.0 release running on Cray Research computer systems.
9.3	August 1997. Online documentation to support the UNICOS 9.3 release running on Cray Research computer systems.

10.0	November 1997. Documentation to support the UNICOS 10.0 release running on Cray Research computer systems and subsequent UNICOS/mk systems running on Cray T3E systems.

Contents

Page

Page

Page

Figures

Page

Page

Tables

Preface

This manual documents how to use the tape subsystem, the tape daemon-assisted interface (also called the Tape Management Facility) for the UNICOS 10.0 and subsequent UNICOS/mk releases. In addition, the manual contains a chapter that describes how to use the character-special tape interface.

Other chapters provides tape subsystem information on using tape formats, performing basic tape procedures, and writing Fortran and C tape applications.

Warning: Because of changes to available software, hardware, and system configurations since the UNICOS 8.0.2 system evaluation, the term "Trusted UNICOS" no longer implies an evaluated product. Any use of the term "Trusted UNICOS" in UNICOS 10.0 documentation refers to the currently available system configuration that closely resembles that of the evaluated Trusted UNICOS 8.0.2 system. Cray Research continues to offer a variety of specific MLS system configurations, including configurations that support functionality required by trusted systems.

Related Publications

The following administration manuals are available:

- *General UNICOS System Administration*, Cray Research publication SG–2301
- *Tape Subsystem Administration*, Cray Research publication SG–2307
- *UNICOS/mk General Administration*, Cray Research publication SG–2601

The following documents contain additional information that may be helpful:

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011
- *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080
- *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111
- *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165
- *Application Programmer's I/O Guide*, Cray Research publication SG–2168

- *UNICOS/mk User Commands Reference Manual*, Cray Research publication SR–2611
- *UNICOS/mk System Libraries Reference Manual*, Cray Research publication SR–2680

Ordering Cray Research publications

The *User Publications Catalog*, Cray Research publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907, or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: 1 User commands 1B User commands ported from BSD 2 System calls

	3	Library routines, macros, and opdefs
	4	Devices (special files)
	4P	Protocols
	5	File formats
	7	Miscellaneous topics
	7D	DWB-related information
	8	Administrator commands
	Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.	
variable	Italic typeface denotes variable entries and words or concepts being defined.	
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.	
[ ]	Brackets enclose optional portions of a command or directive line.	
...	Ellipses indicate that a preceding element can be repeated.	

The following machine naming conventions may be used throughout this document:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems.
Cray MPP systems	All configurations of the CRAY T3D series. The UNICOS operating system is not supported on CRAY T3E systems. CRAY T3E systems run the UNICOS/mk operating system.
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release.

The default shell in the UNICOS and UNICOS/mk operating systems, referred to in Cray Research documentation as the *standard shell*, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS and UNICOS/mk operating systems also support the optional use of the C shell.

Cray UNICOS Version 10.0 is an X/Open Base 95 branded product.

Reader comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail at the following address:

  `publications@cray.com`

- Contact your customer service representative and ask that an SPR or PV be filed. If filing an SPR, use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.
- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

We value your comments and will respond to them promptly.

Introduction [1]

This user's guide describes the characteristics and capabilities of the tape subsystem, which is available on the UNICOS and the UNICOS/mk operating systems. The tape subsystem is also called the tape daemon-assisted interface and the Tape Management Facility.

The guide explains the ways in which you may work with the tape subsystem and provides many examples of commonly used commands. It also describes the use of the character-special tape interface.

This publication is organized as follows:

Chapter	Description
1	Introduces the terminology associated with the tape subsystem, discusses tape-related hardware, documents tape interfaces, and describes the tape subsystem's features.
2	Describes the structure of tape formats.
3	Describes the commands that access tapes by using the tape subsystem, as well as tape status and information commands.
4	Describes the use of the tape subsystem from Fortran programs.
5	Describes the use of the tape subsystem from C programs.
6	Describes the use of the character-special tape interface.
Appendix A	Describes system messages.
Appendix B	Describes tape daemon return values.

Appendix C	Lists the user man pages associated with the tape subsystem.

1.1 Terminology

This section describes terminology used throughout this manual and briefly describes the Cray Research systems that run the tape subsystem. It also describes the features of the tape subsystem.

The following terms are associated with the tape subsystem and are used throughout this manual:

Term	Definition
block size	The block size specifies the size (in bytes) of a data block on a tape.
device group	Each tape device belongs to a device group. The device group name is the generic device name in the configuration file. Also referred to as a *resource*.
device name	Each tape device is identified by a device name, which is defined by a device name entry in the tape configuration file.
device type	Each device has a device type, which is specified by a number. The different tape devices available on Cray Research systems.
file identifier	The file identifier is the name of the file recorded in the `HDR1` label of a labeled tape. If specified in lowercase, it is converted to uppercase, per ANSI standard.
job ID	The job ID is the process identification number unique to the shell or batch job currently in use.
label type	The label type may be one of the following: nonlabeled, IBM standard, ANSI standard, or single tape mark format.
path name	Each tape file is defined by a path name. You can specify the path name of the tape file by using the `tpmnt`(1) command. The system creates an entry in the directory specified by the path name. The

	tape device assigned to the tape file may change during volume switching. While a tape device is assigned to a tape file, you may not remove the path name of that tape file; the path name is removed when the tape device is released.
record length	The record length specifies the maximum length of a logical record (in bytes).
volume ID	The volume identifier is a character string that consists of 1 to 6 alphanumeric characters identifying a tape. The volume ID may also be referred to as the *volume serial number (VSN)* or the *internal VSN*.
external VSN	The external VSN is the human readable label applied to the tape's container.
format ID	A format identifier (ID) is the unique identifier for ER90 devices that is recorded on a tape during the volume format. It is a character string that consists of 1 to 6 alphanumeric characters. It is recommended that this label be the same as the volume ID. If you do not specify a format ID, the volume identifier is recorded on the tape as the format ID.

1.2 Hardware

This section describes the hardware of the tape subsystem. It includes a brief discussion of Cray Research systems, tape devices, and loaders.

The tape subsystem runs on Cray Research systems that have either the I/O subsystem Model E (IOS-E or IOS-V) or GigaRing support.

A wide range of tape devices and autoloaders are available on Cray Research systems. For more information on these, see your Cray Research sales representative.

1.3 Tape interfaces

There are two methods for accessing tapes:

- Tape daemon-assisted interface, commonly referred to as the tape subsystem

- Character-special tape interface

The tape daemon-assisted interface, which is called the tape subsystem in this manual, uses a kernel device driver and the tape daemon. It is the standard method of accessing tape devices. This interface supports many functions including tape resource management, device management, volume mounts and dismounts through operator communication or autoloader requests, label processing, volume switching, and error recovery.

Note: Chapters 2, 3, 4, and 5, and appendixes A and B describe the tape daemon-assisted interface. Only Chapter 6 describes the character-special tape interface.

The character-special tape interface to the tape subsystem is similar to the traditional UNIX process of accessing tape devices. It gives you unstructured access to the tape devices so that you can use standard UNIX commands and `ioctl`(2) requests to manage your tapes.

1.4 Tape subsystem features

This section briefly highlights the following features of the tape subsystem:

- Tape subsystem architecture
- Tape label support
- Tape positioning
- Front-end servicing
- User end-of-volume (EOV) processing
- Multifile volume allocation
- Concatenated tape files

This section does not include features of the character-special tape interface. For information on this interface, see Chapter 6, page 121.

1.4.1 Tape subsystem architecture

The basic elements of the tape subsystem are the tape daemon and the tape device driver. The tape daemon is started by the system operator or the system administrator, or it is started automatically as part of the system startup. The tape daemon has super-user privileges. Therefore, it can communicate directly

with the tape device driver to process your requests. You can execute a tape-related command, which builds a request and sends it to the tape daemon, by way of a tape daemon request pipe. Figure 1 shows the architecture of the tape subsystem.

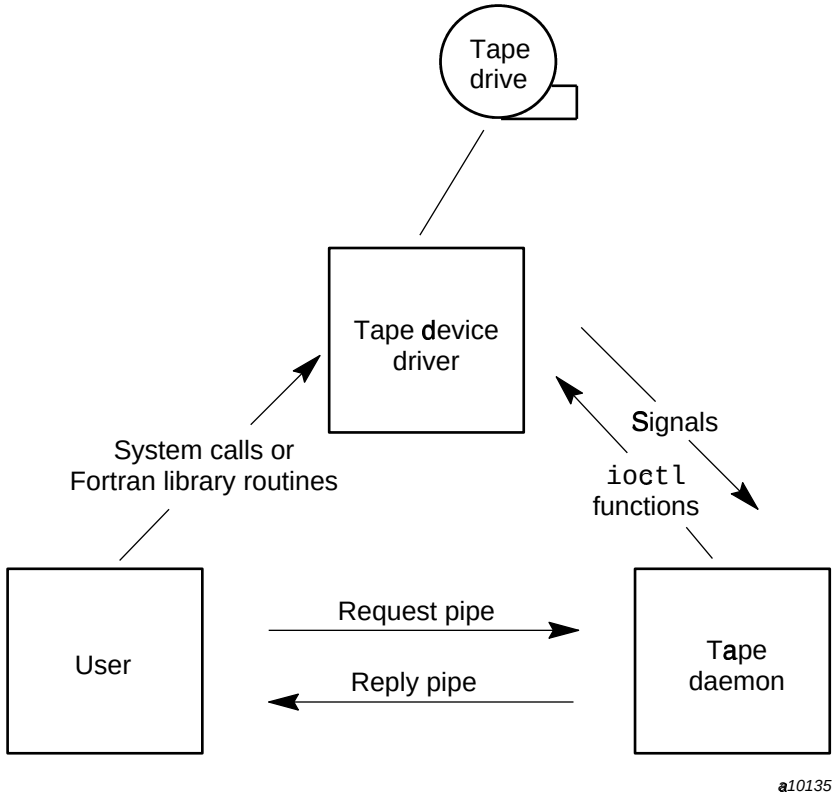

Figure 1. Tape subsystem architecture

The tape device driver signals the tape daemon if any of the following conditions occur:

- You issue an `open`(2) request to the tape path name.
- You issue a `close`(2) request to the tape path name.
- You issue the first I/O request to the tape path name.

- An I/O error occurs.
- A tape mark is read.
- An end-of-tape is detected during a `write`(2) operation.
- An end-of-file is detected, requiring tape mark processing.

If any of these conditions occur, your job is suspended until the tape daemon finishes processing. At this point, the tape daemon requests that the tape device driver either issue you an error message or allow you to continue. Figure 2 illustrates this process.

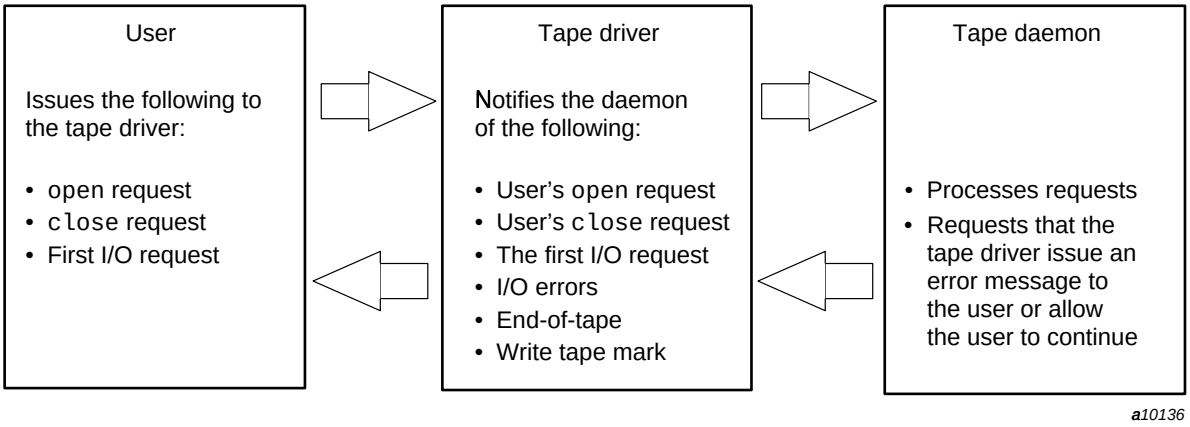

Figure 2. Communication between the user, tape driver, and tape daemon

1.4.2 Tape label support

The tape subsystem supports ANSI standard labels, IBM standard labels, single tape mark format tapes, or no labels. *Single tape mark format tapes* do not have labels and are terminated by a single tape mark at the end-of-volume, whereas a normal *nonlabeled tape* is terminated by two tape marks at the end-of-volume. Also, bypass-label processing is available to privileged users with proper user database (UDB) permission. *Bypass-label processing* lets privileged users read or write tape labels as regular files.

1.4.3 Tape positioning

Tape positioning lets you move to the beginning of a tape block. Tape movement may be forward or backward; however, tape positioning directives cannot be used to circumvent normal tape label processing or label checking unless you have tape-manager permission or bypass label permission and use an absolute track address positioning request (TR_PABS). You can position the tape file relative to a tape mark, tape block, or volume; or you can position the tape file to an absolute track address.

1.4.4 Front-end servicing

While processing tape labels, the tape subsystem requests permission and volume serial numbers from a specific front-end system. Front-end servicing is optional.

1.4.5 User end-of-volume processing

User end-of-volume (EOV) processing lets you gain control at the end of a tape volume. For EOV processing or positioning to a tape block, it is necessary to know that the file being processed is a tape file.

You may request to be notified when end-of-volume is reached. In addition, you can request special user EOV processing, which includes the reading, writing, and positioning of the volume before and after a volume switch. After special processing has completed, you must request that the tape subsystem resume normal processing.

1.4.6 Multifile volume allocation

Multifile volume allocation lets you process a multifile volume tape without the need for the system to unload and load tapes between files.

1.4.7 Concatenated tape files

The concatenated tape file feature lets you read multiple tape files as though they were one tape file. An end-of-volume status is returned for all concatenated files read, until the last file and its end-of-file is encountered.

1.5 Tape performance

This section briefly describes how to improve the performance of the tape subsystem. The tape transfer rate and system CPU time impact tape performance. The tape transfer rate is largely determined by the tape block size, but can be affected by other parameters such as buffer size and system buffer size.

System CPU time is largely a function of the number of system calls used for tape I/O; the primary determinant of system CPU time is the library buffer size.

You can improve tape performance by adjusting system buffering and kernel interfaces.

1.5.1 System buffering

System buffering is used to buffer user data when necessary. Buffering of tape data in the system buffer is optional and can be disabled with the `-U` option on the `tpmnt`(1) command. The defaults set by the system administrator are an important factor in the tape subsystem performance. For Fortran I/O, this means that data is transferred directly between the library buffer and the tape device. For C programmers using system calls directly for tape I/O, this option also imposes certain restrictions. See Section 5.2.

There are two advantages to using the `-U` option of the `tpmnt`(1) command:

- Because the system buffer is not used, the limit on the maximum tape block size imposed by the value of the system parameter `TAPE_MAX_PER_DEV` is removed. With the `-U` option, tape blocks are limited in size only by the size of the user's buffer, the device limit, and for blocked I/O, the `-b` option of the `tpmnt`(1) command.
- In some cases, performance is enhanced because the tape data is only copied twice, rather than three times on each transfer (once between the user's data area and the library, next between the library buffer and the system buffer, and then between the system buffer and the tape device).

In practice, the `-U` option is advantageous only with very large buffers or large tape blocks. For this discussion, large buffers are those over four times the maximum block size (MBS) of the tape, where the MBS is at least 128 Kbytes. In other cases, tape processing with the `-U` option is more expensive in system CPU time and slightly slower in terms of the tape data rate.

System buffering is an important factor in tape subsystem performance. The default is system buffering of all tape data. The `tpmnt -U` command disables

system buffering for blocked I/O. If the MBS (as specified by the `tpmnt -b` command) is less than half the size of `TAPE_MAX_PER_DEV`, the kernel driver divides this buffer into two equal parts and performs asynchronous read-ahead and write-behind operations (double buffering). If the MBS is larger than half the size of `TAPE_MAX_PER_DEV`, the system buffer is used as a single buffer and tape performance will suffer. Tape blocks larger than `TAPE_MAX_PER_DEV` will result in an error. For byte stream I/O, the ER90 driver divides the buffer into two equal parts and performs asynchronous read ahead and write behind operations (double buffering). (For information about the ER90 format, see Section 2.3, page 23.) The `TAPE_MAX_PER_DEV` is configurable by the system administrator. See the *Tape Subsystem Administration*, Cray Research publication SG–2307, for details.

1.5.2 Kernel interface

The kernel interface to tape I/O is through the standard `read`(2) and `write`(2) system calls. It is called transparent I/O.

Flexible file I/O (FFIO) library routines provide another way to perform tape I/O. Fortran I/O is based on FFIO, and the C library contains FFIO routines. See Chapter 4, page 41, for further information. Also see the *Application Programmer's I/O Guide*, Cray Research publication SG–2168, for more information on FFIO.

1.6 Tape multilevel security

When discussing the UNICOS tape subsystem as used on a UNICOS multilevel security (MLS) system in this manual, it is assumed that you have read and understood the following information:

- *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111
- *General UNICOS System Administration*, Cray Research publication SG–2301 MLS chapter of publication SG-2301

Warning: If your site is running a Trusted UNICOS system, you must thoroughly understand the information and follow all procedures discussed in the documents listed previously and noted throughout the rest of this manual to properly use the UNICOS tape subsystem on a Trusted UNICOS system.

UNICOS systems with the UNICOS MLS feature require a user's security label to be equal to the tape security label to write to that tape. The user's security label must dominate the tape security label to read to that tape.

All files on a tape must be at the same security label.

All tape activity can be audited. The tape security administrator can tune the tape subsystem to control what can be audited.

Tape Formats [2]

The tape subsystem supports the IBM compatible tape format and ER90 (D2 cassettes) tape format. The ER90 format is not available on systems that run the UNICOS/mk operating system or that have GigaRing support. This chapter describes and illustrates both formats and label fields.

2.1 IBM compatible tape format

This section briefly describes and illustrates the IBM tape format. Tape format is determined by the presence or absence of labels and the number of files on a tape volume or number of volumes for a tape file.

System labels and tape marks are accessible to a user process without privileges only through the use of the `tpmnt`(1) command.

In the following figures, the character b represents the beginning of the tape and the character * represents a tape mark (HDR2, EOV2, and EOF2 labels are optional for input). The UNICOS and UNICOS/mk operating systems always creates these labels for labeled tapes; other systems may not.

2.1.1 Nonlabeled tapes

Nonlabeled tapes are of two formats, determined by the number of tape marks that indicate end-of-volume.

2.1.1.1 Two tape mark tapes

Nonlabeled tapes with two tape marks, implemented by the `-l nl` option of the `tpmnt`(1) command, may consist of a single-volume file; a multivolume file; or multifile, multivolume file formats. Figure 3 illustrates these formats. For tapes with multiple files, a single tape mark separates files on the same volume. End-of-volume is reached when two consecutive tape marks are encountered and there is another tape to read.

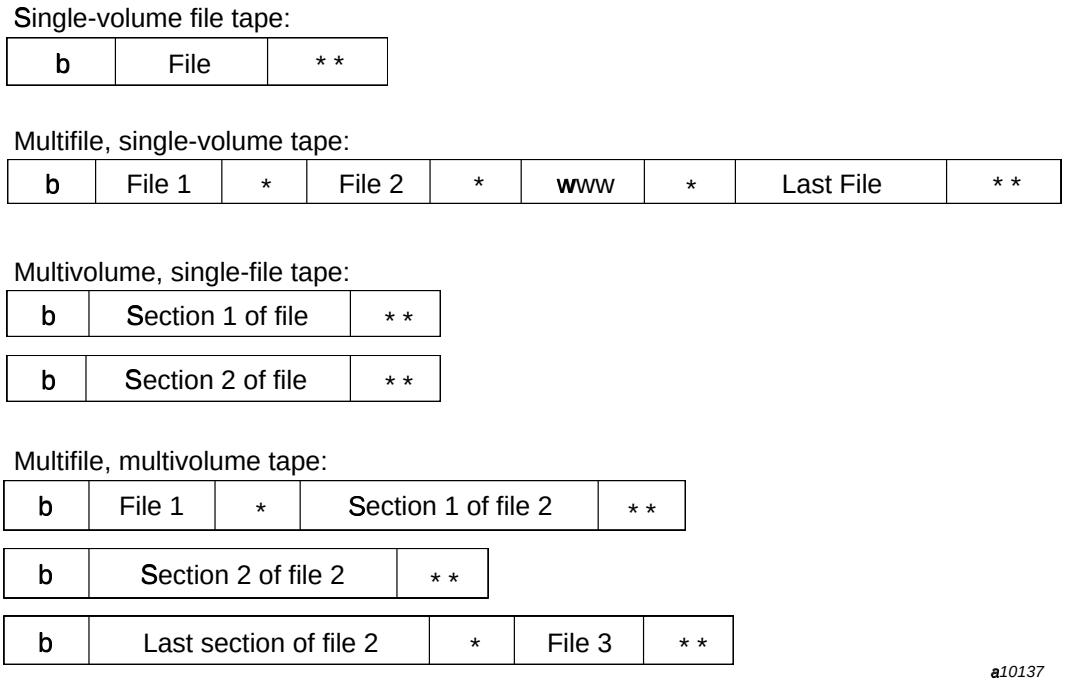

Figure 3. Nonlabeled, two tape mark formats

2.1.1.2 Single tape mark tapes

For nonlabeled, single tape mark format, implemented by the `-l st` option of the `tpmnt`(1)command, a single tape mark indicates end-of-volume. When using one tape mark tape as an input tape, the system reads only to the first tape mark encountered.

When using a single tape mark tape as an output tape, the system terminates the tape with three tape marks, allowing it to be read as a nonlabeled tape later on. Note that because the system processes only the data blocks and the first tape mark, you cannot have multifiles on a single tape mark tape. That is, you cannot use the `-l st` option with the `-q` option of the `tpmnt`(1) command.

Figure 4 illustrates nonlabeled, single tape mark formats.

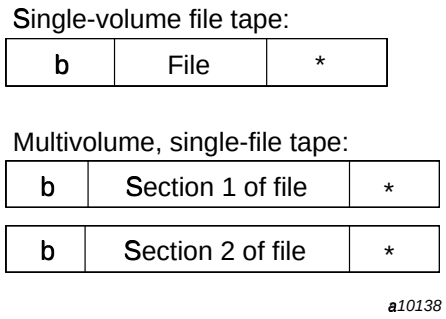

Figure 4. Nonlabeled, single tape mark formats

2.1.2 Labeled tapes

Labeled tapes are implemented by the `-l al` (ANSI standard labels) and the `-l sl` (IBM standard labels) options of the `tpmnt`(1) command. ANSI standard labels and IBM standard labels are similar, with the exception that in IBM standard labels the character fields are represented by EBCDIC characters while in ANSI standard labels the character fields use ASCII characters.

Labeled tapes have the following labels for the tape subsystem (see Section 2.2, page 16, for a description of these labels):

- Volume header label (VOL1)
- First file header (HDR1)
- First end-of-volume (EOV1)
- First end-of-file (EOF1)
- Second file header (HDR2)
- Second end-of-volume (EOV2)
- Second end-of-file (EOF2)

Figure 5 illustrates labeled tape formats.

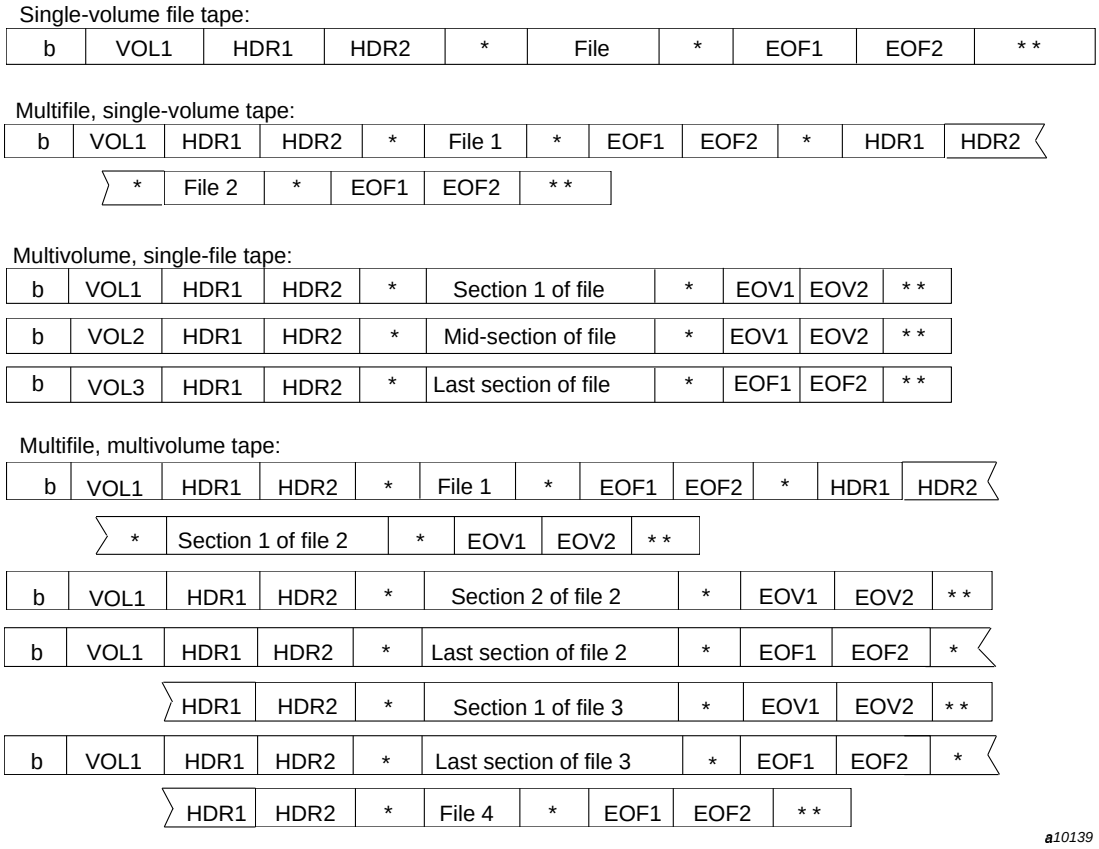

Figure 5. Labeled tape formats

2.1.3 IBM compatible tape format summary

The formats described previously are illustrated in Figure 6 through Figure 9 grouped by number of files and number of volumes. Figure 6 shows a single-volume file; Figure 7 shows a multifile, single-volume tape; Figure 8 shows a multivolume, single-file tape; and Figure 9 shows a multifile, multivolume tape. For each format type, the figures show both labeled (ANSI or IBM) and nonlabeled tapes.

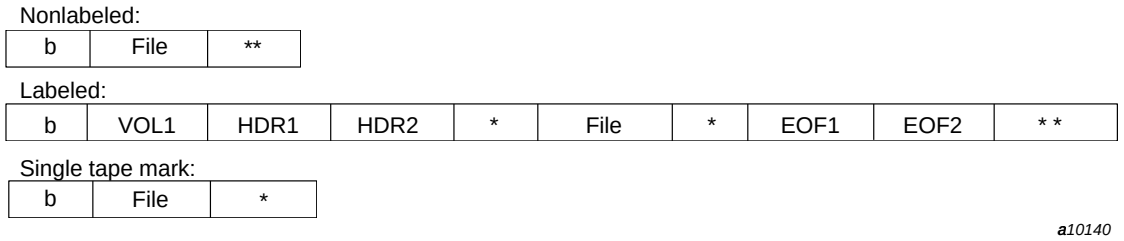

Figure 6. Single-volume file

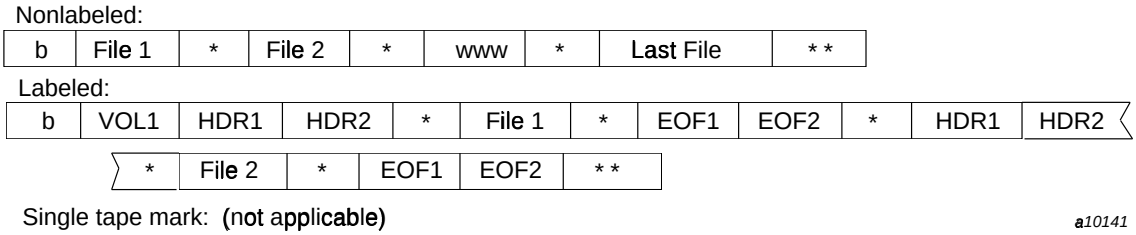

Figure 7. Multifile, single-volume tape

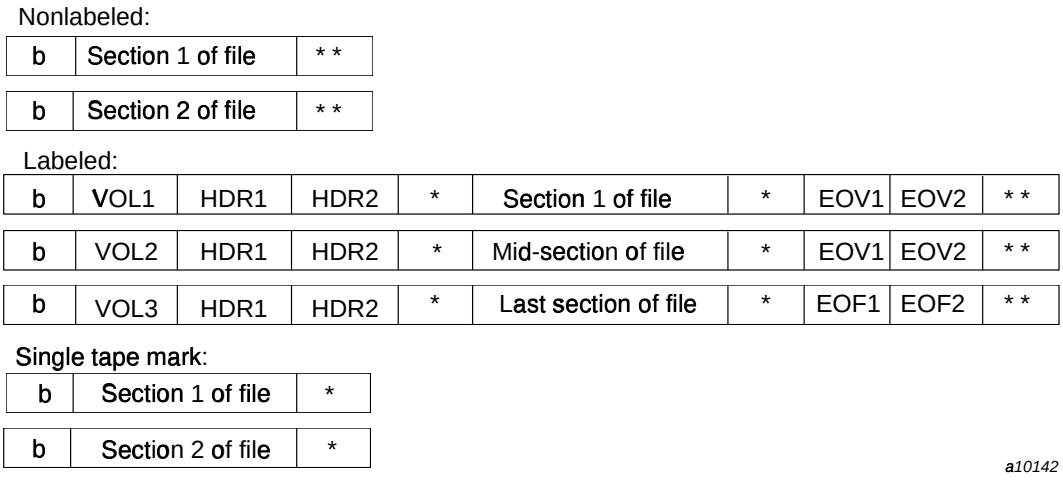

Figure 8. Multivolume, single-file tape

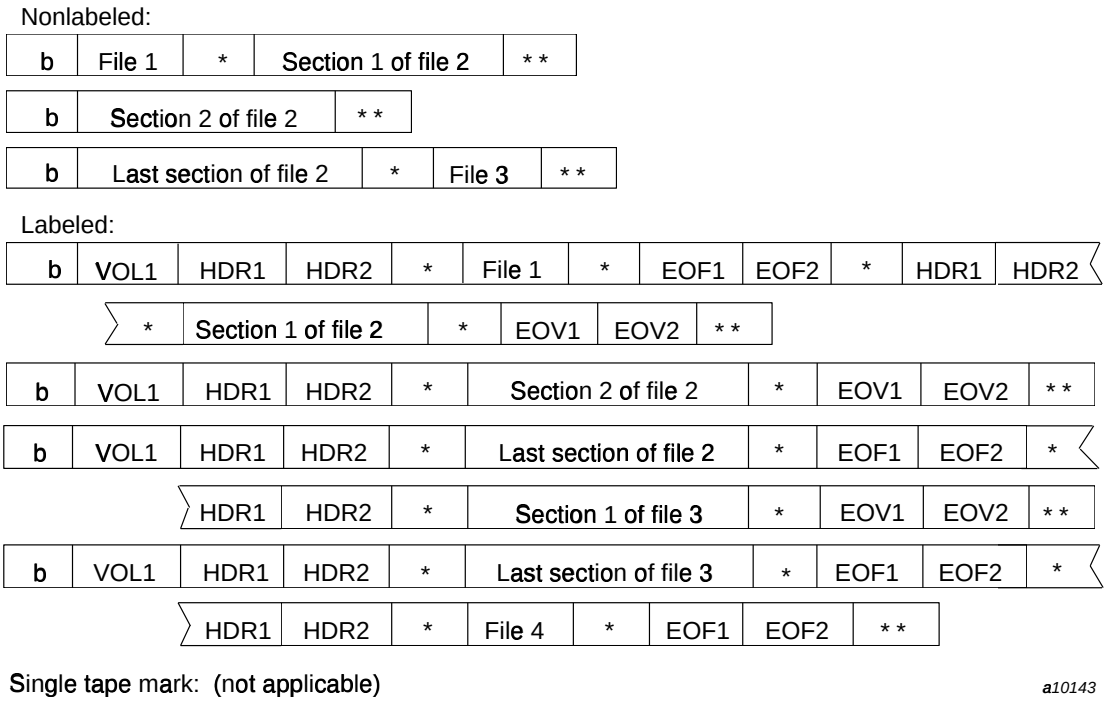

Figure 9. Multifile, multivolume tape

2.2 Tape label fields

This section describes the various tape label fields for ANSI standard and IBM standard labels. Specifically, it describes the fields in which label types are supported. These are checked by the system when reading or writing a tape and those that are filled in with parameter values when you use the `tpmnt`(1) command to create a labeled tape. The following tape labels are described for the tape subsystem:

- Volume header label (VOL1)
- First file header (HDR1)
- First end-of-volume (EOV1)
- First end-of-file (EOF1)
- Second file header (HDR2)

- Second end-of-volume (EOV2)
- Second end-of-file (EOF2)

In IBM standard-label character fields, the characters are represented by EBCDIC characters. ANSI standard labels use ASCII characters.

2.2.1 VOL1 label

The VOL1 label is the first block on a labeled tape. Table 1 describes the fields for an ANSI standard label. Figure 10 shows the format of the VOL1 label.

Table 1. VOL1 label values

Field	Starting byte	Length in bytes	Contents	Description
`label id`	1	4	VOL1	VOL1 label; required system-supplied character string.
`volume id`	5	6	*vi*	Volume identifier of the tape; it is specified with the `-v` option or contained in the file specified with the `-V` option of the `tpmnt`(1) command. It is checked on all labeled tapes and contains up to 6 alphanumeric characters.
`owner id`	38	14	*owner_id*	User ID of the tape owner.
`standard level`	80	1	*label standard version*	ANSI standard version number for label and data formats. For Cray Research systems, the version number is 4.

The fields of the ANSI standard VOL1 label are the same as the IBM standard VOL1 label, with the following exceptions:

- The `owner id` field of the IBM standard VOL1 label starts at byte 42 and has a length of 10 bytes.
- The standard level field is not used in the IBM standard VOL1 label.

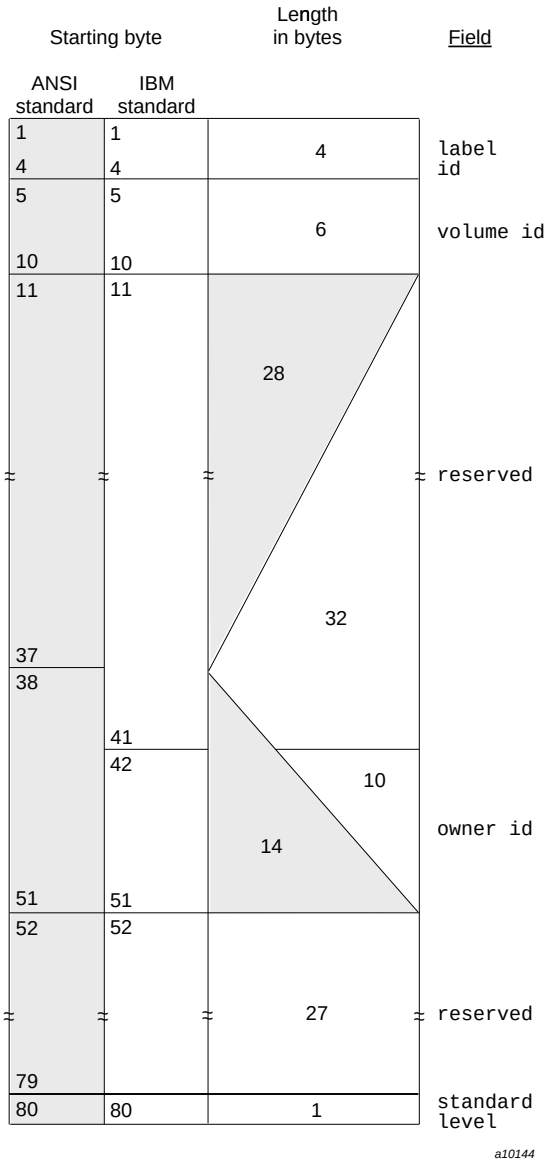

Figure 10. VOL1 label

2.2.2 HDR1, EOV1, and EOF1 labels

The HDR1 label is located before each file or section of a file on a tape volume. If a file is not completed on a tape volume and extends to the following tape volume, the data in the file is followed by an EOV1 label. If a file or file section is completed on a tape volume, the data in the file is followed by an EOF1 label.

The fields of the HDR1, EOV1, and EOF1 labels are the same in both the ANSI standard label format and the IBM standard label format. Table 2 describes the specified fields. Figure 11, page 21, shows the format of the HDR1/EOV1/EOF1 labels.

Table 2. HDR1/EOV1/EOF1 labels

Field	Starting byte	Length in bytes	Contents	Description
`label id`	1	4	HDR1 EOV1 EOF1	Label type; required system-supplied character string.
`file id`	5	17	*file_id*	File identifier; 1 through 17 alphanumeric character field specified by the `-f` option of the `tpmnt`(1) command. If the `-f` option is not specified, the file identifier is taken from the path name of the `-p` or `-P` option of `tpmnt`. The level of checking on `file id` is installation specified.
`sequence`	28	4	*number*	Order of this volume in a multivolume set; it is specified by a decimal number (1 through 9999) on the `-O` option of `tpmnt`.
`file sequence`	32	4	*number*	File order within a multifile tape; it is specified by a decimal number (1 through 9999) on the `-q` option of `tpmnt`. The system uses the specified value to position the tape volume to the proper file.
`creation date`	42	6	*cyyddd*	Creation date (pseudo-Julian format) of a new tape; *c* = century (blank = 19, 0 = 20, 1 = 21...), *yy* = year (00-99), and *ddd* = day (001-366).

Field	Starting byte	Length in bytes	Contents	Description
`expiration date`	48	6	*cyyddd*	Expiration date (pseudo-Julian format) at which time a tape may be scratched or overwritten. Normally, it is specified in the *cyyddd* format by using the `-x` option of `tpmnt`. Otherwise, you can specify the number of days on the `-t` option by using `tpmnt`(1). The specified number is added to the `creation date`, thus creating the `expiration date`.
`block count`	55	6	*number*	Number of data blocks in the preceding file section or file on the current tape volume for EOV1 and EOF1 labels. The block count in the HDR1 label contains a value of 000000. In EOV and EOF labels for standard labels (`sl`), if the block count is greater than 999,999, the `block count` field will represent the block count as mod 1,000,000. The overflow (`block count` / 1000000) will be stored in bytes 76 through 80. This is the `extended block count` field. For ANSI standard labels (`al`), if the block count is greater than 999,999, the `block count` field will represent the block count as mod 1,000,000.
`extended block count`	76	5	*number*	For standard labels (`sl`), if the block count is greater than 999,999, the `block count` field will represent the block count as mod 1,000,000. The `extended block count` field will contain the overflow (`block count` / 1000000).

Starting byte	Length in bytes	Field
1 4	4	label id
5 21	17	file id
22 27	6	reserved
28 31	4	sequence
32 35	4	file sequence
36 41	6	reserved
42 47	6	creation date
48 53	6	expiration date
54	1	reserved
55 60	6	block count
61 ≈ 75	≈ 14 ≈	reserved
76 80	6	extended block count

a10145

Figure 11. HDR1/EOV1/EOF1 labels

2.2.3 HDR2, EOV2, and EOF2 labels

An HDR2 label immediately follows an HDR1 label, and it is followed by a tape mark. An EOV2 label immediately follows an EOV1 label, and it is

followed by two tape marks. An EOF2 label immediately follows an EOF1 label, and if more files follow this file, it is followed by one tape mark. If the EOF2 label is the last file on the tape volume, it is followed by two tape marks.

ANSI standard does not specify a format for these labels, except for the first 4 bytes. IBM standard labels use the HDR2, EOV2, and EOF2 labels to store additional information concerning the file they bracket. The operating system automatically writes these labels when you use the `-l sl` or `-l al` options of `tpmnt`(1). These labels follow the IBM standard format. Table 3 describes the specified fields. Figure 12 shows the format of the HDR2/EOV2/EOF2 labels.

Table 3. HDR2/EOV2/EOF2 labels

Field	Starting byte	Length in bytes	Contents	Description
`label id`	1	4	HDR1 EOV1 EOF1	Label type; required system-supplied character string.
`record format`	5	1	*format*	Record format; 1-character filed specified by the `-F` option of `tpmnt`(1).
`block length`	6	5	*number*	Maximum block length, in bytes, for the associated file; specified by a decimal number (1 through 99999) on the `--b` option of `tpmnt`. If the block length is greater than 100000, the block length field will represent the block length as mod 100000.
`record length`	11	5	*number*	Record length, in bytes; specified by the `--L` option of `tpmnt`.
`density`	16	1	*number*	Tape density; specified by the `-d` option of `tpmnt`. The operating system supports 1600 bpi (this field contains the value 3) and 6250 bpi (this field contains the value 4).
`security level`	55	3	*number*	Security level.
`compartments`	59	17	*numbers*	Security compartments.

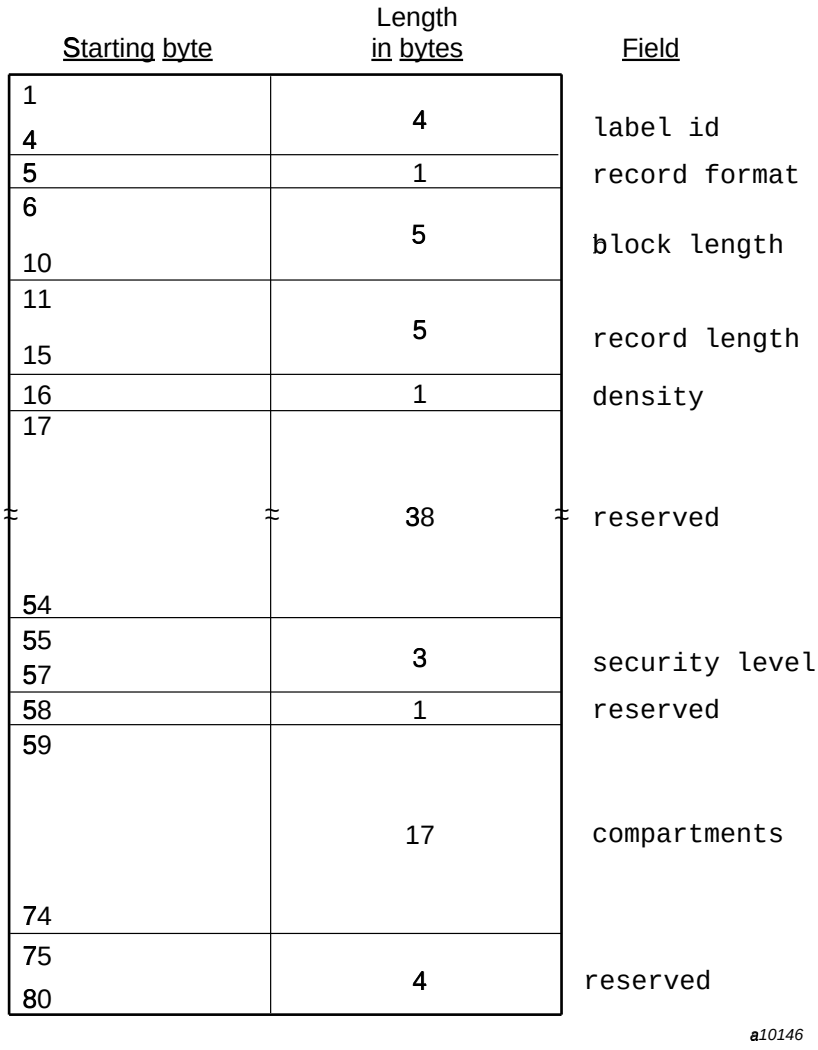

Figure 12. HDR2/EOV2/EOF2 labels

2.3 ER90 volumes

The ER90 supports D2 cassettes with 19-mm tapes in three cassette sizes: 25 Gbyte, 75 Gbyte, and 165 Gbyte.

Note: The ER90 format is not available on systems that run the UNICOS/mk operating system or that have GigaRing support.

You must format a volume before it can be used. To format a volume, you must create single or multiple partitions on the cassette, record a volume identifier, and, if requested, create system zones. A volume can be preformatted, or it can be formatted during write operations.

Partitions are logical volumes within a physical volume. A partition can span the length of the tape, or multiple partitions can be created. The tape subsystem treats partitions as individual volumes; they are accessed individually, and tape operations to one partition do not affect other partitions on the volume. Multiple partitions cannot be created during write operations.

The ER90 device records a format identifier (ID) as part of a volume format operation. The format identifier is an alphanumeric string consisting of up to 6 ASCII characters that uniquely identifies the cassette after tapes are mounted. The format identifier is recorded throughout the volume after each system zone and at the beginning-of-tape (BOT) and end-of-tape (EOT) markers.

You can format a volume with or without system zones. *System zones* are data-free areas on the tape that can be used to load and unload the cassette. With system zones, a cassette does not have to be positioned at the BOT or the EOT to be unloaded.

If a volume is formatted to have the default number of system zones, a tape unload takes approximately 16 seconds for small cassettes, 21 seconds for medium cassettes, and 24 seconds for large cassettes. Volumes that are formatted without system zones can take up to 185 seconds to be unloaded. The disadvantage of formatting with system zones is that if a tape is created during write operations, the ER90 device must suspend I/O operations to create the system zone. It takes approximately 18 seconds to create a system zone for small cassettes, 31.3 seconds for medium cassettes, and 55.8 seconds for large cassettes. It takes 3.2 seconds to skip over a system zone and continue writing for small cassettes, 3.6 seconds for medium cassettes, and 5.3 seconds for large cassettes.

By default, the tape daemon formats a blank tape as a single partition volume with system zones. The format ID specified on the tape mount is recorded on the volume, which is formatted during write operations.

The tape administrator can use the `tpformat`(8) command to preformat ER90 volumes. This command reserves an ER90 device, mounts the volume, issues the format request to the ER90 device, and then, after the format is completed, releases the reserved resource.

Tape Subsystem Tutorial [3]

This chapter describes the following tape subsystem procedures:

- Reserving, mounting, reading, writing, and releasing a tape
- Obtaining tape status and information
- Using the tape subsystem with standard operating system commands
- Mounting ER90 volumes
- Using MLS considerations

To see an online description of a particular command or routine, use the `man`(1) command.

Note: The ER90 format is not available on systems that run the UNICOS/mk operating system or that have GigaRing support.

3.1 Getting started

To use the tape subsystem, you follow four basic steps:

1. Reserve tape resources by using the `rsv`(1) command.
2. Request a tape mount for a tape file by using the `tpmnt`(1) command.
3. Process the tape file information by using whatever commands or programs you need to accomplish what you want to do.
4. Release the tape resources reserved by using the `rls`(1) command.

The identifier of the tape resource is site configurable by the system administrator in the tape configuration file. You can use the `tpstat`(1) command to display available tape resources. The resource name is displayed in the `dgn` column.

This following tape naming conventions are used in this manual:

Name	Description
TAPE	Half-inch round tape
CART	3480 type cartridge (square tape)

QIC	Quarter-inch cartridge tape
EXB	Helical scan recording on 8mm cartridge tape
DAT	Helical scan recording on 4mm cartridge tape
SILO	3480 tape located on a SILO/400 robotic loader
WOLF	3480 tape located on a WolfCreek robotic loader
3490	3490 type
3490E	3490E type
ER90	ER90 (D-2 format) helical scan tape device

Figure 13 shows a simple example that requests a tape to be mounted. This tape has an IBM standard label with a volume serial number (VSN) of `000001` and a file name of `tf`. The contents of the disk file named `data` is copied to `tf`. After the tape file is created, the tape is unloaded and the allocated tape drive is released.

```
$ rsv CART
$ tpmnt -l sl -v 000001 -P tf -n -g CART
$ cp data tf
$ rls -a
```

Figure 13. Creating a tape

Figure 14 shows an example that will mount the previous tape, indicating that it is an old tape, and read the data from the tape into a disk file named `old.data`.

```
$ rsv CART
$ tpmnt -l sl -v 000001 -P tf -o -g CART
$ cp tf old.data
$ rls -a
```

Figure 14. Reading an existing tape file

Figure 15 shows an example that will add a new tape file (file sequence 2) to the tape 000001 and will copy the disk file named new.data into the new tape file.

```
$ rsv CART
$ tpmnt -l sl -v 000001 -q 2 -P tf -n -g CART
$ cp new.data tf
$ rls -a
```

Figure 15. Adding a new file to an existing tape

The following example shows how to submit a job through a Network Queuing System (NQS). Before submitting the job, you need to know how the NQS maps the limit specification on the qsub(1) command to the available tape device groups. To display the available device groups in limit-enforced order, issue the tprst(1) command The following example shows a listing of available device groups:

```
dev grp  w     rsvd     used available
CART             0        0    2
TAPE             0        0    0
```

In this example, there are two NQS associations. Resource group a corresponds to the CART device group and resource group b corresponds to the TAPE device group. To submit a job using the TAPE resource group, you must specify the qsub option, -lUb 1.

Figure 16 shows an example of an NQS job that requests a specific tape (SCRSL) on to which a file named data will be copied.

```
$ cat > example.sh <$ cat > example.sh <EOF
rsv TAPE
tpmnt -l sl -v SCRSL -g TAPE -P tf -n
cp data tf
rls -a
EOF
$ qsub -lUb 1 example.sh
```

Figure 16. NQS tape job

3.2 Obtaining tape status

You can use the commands and files described in this section for obtaining tape status and for sending messages to the operator.

3.2.1 Tape status commands

To check the status of your tape reservations, use the `tprst`(1) command. For example, enter the `tprst`(1) command to display the reserved-tape status device group name, number of reserved devices, number of used devices, and number of devices available for use as shown in Figure 17. In this example, no `CART`, `TAPE`, or `SILO` devices have been used or reserved, but one `TAPE` device is available for reservation:

```
$ tprst
dev grp  w     rsvd     used available
CART             0        0    0
TAPE             0        0    1
SILO             0        0    0
```

Figure 17. `tprst`(1) status display

Note: The information display in the `dev grp` column is determined by the system administrator in the tape configuration file. See Section 3.1, page 25.

To check on the status of the tape subsystem, use the `tpstat`(1) command as shown in Figure 18. To display the user ID, device group name, device name,

device identifier, device type, status of the device, job ID of the user, and volume identifier of the mounted tape, enter `tpstat`(1).

In this example, two `CART` devices, `cart120` and `cart121`, and one `TAPE` device, `tape201`, are idle and available for use. One `CART` device, `cart122`, is assigned to user `jas`, job ID `170`, with a volume identifier of `ISCSL`. It is on tape block `101`. All other devices are down and unavailable.

```
tpstat

userid  jobid dgn   a stat dvn      bx i rl ivsn   evsn   blks     NQSid
              CART  + idle cart120  04 0
              CART  + idle cart121  02 0
jas     170   CART  + assn+cart122  03 0 is ISCSL  ISCSL  101
              CART  + down cart123  05 0
              TAPE  + down tape200  10 0
              TAPE  + idle tape201  11 0
              CART  - down 300      14 0
```

Figure 18. `tpstat`(1) status display

For ER90 devices, you can specify the `-l` option on the `tpstat`(1) command to display the format identifier. This option will also display a larger block count field.

If the administrator has given you bypass label permission, you can issue a `tplist`(1) command to display the contents of a tape volume. When using `tplist`(1), you do not have to issue separate `rsv`(1), `tpmnt`(1), or `rls`(1) command. For example, to display the contents of a cartridge tape with a volume ID of `000599` and a path name of `x`, enter the `tplist`(1) command as illustrated in Figure 19.

```
tplist -v 000599 -g CART x
EBCDIC Labels
VOL1:volser:000599 owner: wek
HDR1:file_id:x           file_section:0001 file_sequence:0001
   creation date: 93148 expiration date: 93148
HDR2:max_blocksize 32768
Recsize=80 Number=3
Total Records=3, Size=240
*****TAPEMARK*****
Recsize=4096 Number=10
Total Records=10, Size=40960
*****TAPEMARK*****
EOF1: Blockcount:000010
Recsize=80 Number=2
Total Records=2, Size=160
*****TAPEMARK*****
*****TAPEMARK*****
3 file(s)
```

Figure 19. `tplist`(1) display

A message is sent to the operator to mount the cartridge. After the cartridge has been mounted, `tplist`(1) reads the cartridge and sends the output to your screen.

3.2.2 Tape log file

When you issue a `rsv`(1) command, a log file called `tape.msg` is created in your current working directory. This log file keeps track of messages the tape subsystem issues concerning your tape job. All informative and error messages are appended to this file. Figure 20 shows an example of a tape message log file.

```
Jun 14 14:00:53 0000362.1113 TM000 - tape resource reserved for you
Jun 14 14:01:04 0000373.4520 TM122 - mount tape ABCDEF(sl) ring-in, on a TAPE device
for bob 23,  () or reply cancel / device name
Jun 14 14:02:25 0000454.6664 TM048 - /tmp/jtmp.000452a/tapefile : assigned to tape203
Jun 14 14:03:26 0000515.0516 TM049 - /tmp/jtmp.000452a/tapefile : ABCDEF(sl) : open :
blocks = 0
Jun 14 14:03:28 0000516.9823 TM049 - /tmp/jtmp.000452a/tapefile : ABCDEF(sl) : bov :
write : blocks = 0
Jun 14 14:03:28 0000517.0389 TM049 - /tmp/jtmp.000452a/tapefile : ABCDEF(sl) : bof :
write : blocks = 0
Jun 14 14:03:29 0000518.3960 TM049 - /tmp/jtmp.000452a/tapefile : ABCDEF(sl) : eot :
write : blocks = 12
Jun 14 14:03:29 0000518.6526 TM049 - /tmp/jtmp.000452a/tapefile : ABCDEF(sl) : close :
        blocks = 12
Jun 14 14:03:36 0000524.7945 TM050 - tape203 : released
Jun 14 14:03:36 0000524.8156 TM029 - all tape resources released
```

Figure 20. `tape.msg`

3.2.3 Messages to operator

The `msgi`(1) and `msgr`(1) commands let you send messages to the operator. For example, the following command line sends an informative message to the operator:

1. Reserve tape resources by using the `rsv`(1) command.
2. Request a tape mount for a tape file by using the `tpmnt`(1) command.
3. Process the tape file information by using whatever commands or programs you need to accomplish what you want to do.
4. Release the tape resources reserved by using the `rls`(1) command.

```
msgi  Please check device tape00
```

To send an interactive message to the operator, use `msgr`(1). The following is an example of a message you might send to the operator. The operator may then send a reply back to you.

```
msgr  "Is tape ABC on the system?"
```

3.3 Using standard commands

This section describes some of the ways in which you may work with the tape subsystem by using standard UNICOS and UNICOS/mk commands.

Before you can issue requests to the tape subsystem for tape file processing, you must reserve the required number of tape drives for each device type needed. After you have reserved the tape drives, you may specify the tape volume in which the files to be processed are located.

After you have the volumes mounted, you can begin processing the tape files. When processing is complete, release the reserved tape drives.

3.3.1 Using the cp(1) command

The following example illustrates the process of copying a file from disk to tape using the cp(1) command:

1. Reserve a tape by using the rsv(1) command. In this example, the device group name is CART and the number of devices requested is 1:

   ```
   rsv CART 1
   ```

2. Request a tape mount by using the tpmnt(1) command. In this example, the tape has standard labels, a volume identifier of ISCSL, and a path name of /tmp/tapefile. During processing of the tpmnt(1) command, the tape subsystem creates a character-special file, /tmp/tapefile. Do not remove, rename, or move this file:

   ```
   tpmnt -v ISCSL -l sl -p /tmp/tapefile -g CART -b 32768 -n -r in
   ```

3. Copy file myfile by using the cp(1) command-line syntax, as follows:

   ```
   cp myfile /tmp/tapefile
   ```

   The cp(1) command copies bytes of data from the disk file to the tape file. It does not format any data, but blocks it into tape records of size 32768 bytes for IBM compatibles devices.

4. Release the reserved tape. The code in this example releases all resources. The tape device is allocated to you until you issue the rls(1) command with the -a, -d, or -p option or until you log out. When you issue the

`rls`(1) command or log out, the tape subsystem deletes the associated file, `/tmp/tapefile`.

```
rls -a
```

3.3.2 Using the `dd`(1) command

The following example uses the `dd`(1) command to copy a disk file to tape, converting it from ASCII to EBCDIC:

1. Reserve a tape:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has an IBM standard label, the volume ID is `SCRSL`, it is a new file, and a write ring is specified to be on the reel:

   ```
   tpmnt -b 4096 -l sl -v SCRSL -p /tmp/tapefile -n -r in -g TAPE
   ```

3. Use the `dd`(1) command to copy file `mydisk` to tape, specifying a block size of 4096 bytes, with a conversion from ASCII to EBCDIC for IBM compatible devices:

   ```
   dd if=mydisk of=/tmp/tapefile bs=4096 conv=ebcdic
   ```

4. You can also use the `dd`(1) command to read the tape file back into file `newfile`, and convert back to ASCII:

   ```
   dd if=/tmp/tapefile of=newfile bs=4096 conv=ascii
   ```

5. Release the tape resources:

   ```
   rls -a
   ```

3.3.3 Using the `tar`(1) command

The examples in this section show you how to read or write to tape by using the `tar`(1) command.

Procedure 1: Example 1

The following is an example of using the `tar`(1) command to read or write to tape. You must use the `-f` option of the `tar`(1) command and specify the device path name you used in the `tpmnt`(1) command.

1. Reserve a tape using the default values of the `rsv`(1) command:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount by using the `tpmnt`(1) command:

   ```
   tpmnt -l sl -p /tmp/tapefile -v SCRSL -n -r in -g TAPE
   ```

3. Copy the subtree to tape, starting at the current working directory:

   ```
   tar -cvfb /tmp/tapefile 8 *
   ```

4. Change to a new directory:

   ```
   cd /tmp/newdir
   ```

5. To read the tape back in, copy the tar subtree back from tape to your current working directory:

   ```
   tar -xvfb /tmp/tapefile 8
   ```

6. Release the reserved tape:

   ```
   rls -a
   ```

Procedure 2: Example 2

The following example shows you how to read a tape that was created as shown in example 1 or on another UNIX system. In this example, the contents of the tape are read into your current working directory.

1. Reserve a tape:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount by using the `tpmnt`(1) command:

   ```
   tpmnt -l sl -p /tmp/tapefile -v SCRSL -g TAPE -o
   ```

3. Read the tape, using the `tar`(1) command:

   ```
   tar -xvf /tmp/tapefile
   ```

4. Release the reserved tape:

   ```
   rls -a
   ```

3.3.4 Using the cpio(1) command

The following is an example of using the cpio(1) command to read and write to a tape. In this example, data is written to tape, then the cpio(1) command is used to read the data from tape:

1. Reserve a tape with a device group name of CART:

   ```
   rsv CART 1
   ```

2. Request a tape mount using the tpmnt(1) command:

   ```
   tpmnt -l sl -v ISCSL -p /tmp/tapefile -g CART -n -r in
   ```

3. Copy the subtree to tape, starting with the current working directory:

   ```
   find . -print | cpio -Bcov > /tmp/tapefile
   ```

4. Change to a new directory:

   ```
   cd /tmp/newdir
   ```

5. To read the tape back in, copy the cpio(1) subtree into your current working directory:

   ```
   cpio -civd < /tmp/tapefile
   ```

6. Release the reserved resources:

   ```
   rls -a
   ```

3.3.5 Using the tpmnt(1) command to read concatenated tape files

The -c option of the tpmnt(1) command allows you to read multiple tape files as though they were one tape file, with the following exceptions:

- Record size for all files to be concatenated in a job must be the same.
- Variable-length record size is not supported; it causes unpredictable results.

If front-end servicing is enabled and a tape management catalog is used, tape messages are sent to the front end.

The following example shows you how to concatenate three tape files that have the record size and block size in the label:

1. Reserve a tape by using the rsv(1) command, as follows:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount by using the `tpmnt`(1) command. The first occurrence of `tpmnt`(1) establishes tape file `target`(1), to which you concatenate your tape files. If a front-end catalog does not exist, as in this example, you must specify the volume identifier by using the `-v` option. If a front-end catalog does exist, it is not necessary to specify the volume identifier. The second and third occurrences of `tpmnt`(1), using the `-f` option, specify the tape files that you want concatenated to tape file `target`(1), specified with the `-c` option.

   ```
   tpmnt -p target -l sl -o -v t03600 -g TAPE
   tpmnt -c target -l sl -o -v t03700 -f myfile.1 -g TAPE
   tpmnt -c target -l sl -o -v t03800 -f myfile.2 -g TAPE
   ```

   The tape file read contains the contents of tape files target, `myfile.1`, and `myfile.2`.

3. Copy the three tape files to disk by using the `cp`(1) command:

   ```
   cp target diskfile
   ```

4. Release the tape unit, as follows:

   ```
   rls -a
   ```

3.3.6 Using the `tpmnt`(1) command to read or write multifile tapes

Multifile volume allocation lets you process a multifile volume tape without the need for the system to unload and load tapes between files. If you use the `tpmnt`(1) command to specify that the volume identifier for the multiple-volume tape is to be first in the list of volume identifiers, the tape daemon requests that the operator mount the multifile volume tape the first time it is requested. If you request that the same volume identifier be mounted again, no mount message is sent to the operator. However, if you do not specify the same volume identifier, the request is processed for separate volumes.

When using multifile volume allocation, you can use only one file on a tape at a time; that is, you must open a specified file, process the file, and then close it before you can open another file on the same multifile tape volume. You must also reserve a device for each multifile volume tape requested (for example, if the tape files reside on several volumes, you can have files on different volumes opened at the same time; however, you must have reserved enough devices to hold all of the volumes). The following examples show you how to use multifile volume allocation.

3.3.6.1 Example 1

The following example shows you how to use multifile volume allocation to read three files from the same tape without having the operator mount the tape three times:

1. Reserve a tape:

```
rsv TAPE 1
```

2. Request a tape mount using the `tpmnt`(1) command. In this example, the tape has an ANSI standard label, the volume identifier is `SCRAL`, and the path names are `one`, `two`, and `three`, with file sequence numbers of the files to be processed as `1`, `2`, and `3`:

```
tpmnt -l al -v SCRAL -p one -q 1 -r in -g TAPE
tpmnt -l al -v SCRAL -p two -q 2 -r in -g TAPE
tpmnt -l al -v SCRAL -p three -q 3 -r in -g TAPE
```

3. Read the accessed tape files into disk files:

```
cat one > firstfile
cat two > scndfile
cat three > thrdfile
```

4. Release the reserved tape:

```
rls -a
```

3.3.6.2 Example 2

The following example shows you how to use a multifile tape to write three files to the same tape:

1. Reserve a tape:

```
rsv TAPE 1
```

2. Request a tape mount using the `tpmnt`(1) command. In this example, the tape has an ANSI standard label, the volume identifier is `SCRAL`, and the path names are `one`, `two`, and `three`, with file sequence numbers of the files to be processed as `1`, `2`, and `3`. The `-u` option is used so that the tape will not be unloaded when the process terminates. This is useful when a tape is used repeatedly, and it minimizes operator time spent mounting tapes:

```
tpmnt -u -l al -v SCRAL -p one -q 1 -n -g TAPE
tpmnt -u -l al -v SCRAL -p two -q 2 -n -g TAPE
```

```
tpmnt -u -l al -v SCRAL -p three -q 3 -n -g TAPE
```

3. Write the disk files to the specified tape files:

```
cat file1 > one
cat file2 > two
cat file3 > three
```

4. Release the reserved tape:

```
rls -a
```

3.4 Mounting ER90 volumes

This section describes how to mount volumes on an ER90 device. Use the `tpmnt`(1) command to mount a volume. If you use the `-v` option, you can specify three identifiers and a partition number to uniquely identify the requested volume.

After the system determines that the correct physical volume is mounted, it positions the tape at the requested partition. For labeled tapes, the VOL1 label is read from the tape. The identifier in the label is compared to the requested internal ID to verify that the tape was positioned to the correct logical volume.

You can bypass format ID verification by specifying the `tpmnt`(1) `-I` option. Only users with bypass label or tape manager permission may use this option.

Normal mount processing positions the tape to the beginning of a partition. To request that the volume remain at its load position, specify the `-z` option (this is useful if processing should begin near the load position). The device requires that the logical position be established before issuing any tape movement requests; therefore, you must issue a position request to an absolute track address immediately after opening the tape file when `-z` is specified. Only users with bypass label or tape manager permission may use this option. Because label processing is bypassed, the specified label type must be a bypass label.

3.5 Using MLS

Special considerations should be taken when using the tape subsystem on a UNICOS multilevel security (MLS) system. This is especially true for authorized users who are allowed access to data that their active label does not dominate.

The mandatory access control (MAC) label associated with a tape is set to the active MAC label of the process writing to the tape.

Care should be taken when writing data to tape that will later be used to restore security attributes. The MAC label on the tape should sufficiently protect the tape so that only security administrators can access the tape. The label must prevent nonadministrative users from modifying the contents of the tape.

Tape headers contain a protection flag that indicates there are additional protections on the tape. On the UNICOS operating system, this flag implies the existence of a MAC label.

On UNICOS MLS systems before UNICOS release 8.0, a MAC label is not written to the tape when the user's MAC label is level 0 with no compartments. Any other MAC label forces the protection flag to be set and the MAC label is written on the tape. Tapes without the protection flag set are assumed to be at level 0 with no compartments. UNICOS systems that do not have security enabled do not set the protection flag; hence, tapes can be moved between secure and nonsecure systems.

UNICOS 8.0 MLS requires the ability to always set the protection flag and write the MAC label on the tape. Tapes without the protection flag set cannot be accessed.

The `allow_unprotected` parameter of the `OPTIONS` statement was added for compatibility considerations. When the option is set to `YES`, the previous behavior (before UNICOS 8.0) is enforced. When it is set to `NO` (required for Trusted UNICOS), the protection flag is always set and only tapes with the protection flag set can be accessed.

Tape device groups have associated MAC label ranges. To read a tape, the MAC label on the tape must be within the MAC label range of the device group . To write to a tape, the user's active MAC label must be within the MAC label range of the device group.

If you encounter any of the following, contact your system administrator for help in resolving the issue.

- The user fails the required MAC dominate or equal restrictions when reading or writing a tape.
- The `allow_unprotected` parameter of the `OPTIONS` statement is not set to `YES`, and consequently, the user cannot access tapes created on the following:
  - A UNICOS system with security disabled
  - A system before UNICOS 8.0 by users with the active MAC label of level 0 with no compartments

  - An UNICOS system with the `allow_unprotected` parameter of the `OPTIONS` statement set to `YES` by users with the active MAC label of level 0 with no compartments

- The device group does not support the MAC label of the data being accessed through the device. If MAC restrictions are not required on the device, the MAC label ranges of the device group should be `SYSLOW` with no compartments to `SYSHIGH` with all compartments.
- If the Data Migration Facility (DMF) is installed, all device groups used by DMF must support the `SYSHIGH` MAC label.
- The `tplabel`(1) command clears the protection flag in the tape label. Systems that have the `allow_unprotected` parameter of the `OPTIONS` statement set to `NO` are not able to access the tape after the tape has been relabeled.

Writing Fortran Applications Using Tapes [4]

This chapter describes how you can use the tape subsystem from Fortran programs. You can use Fortran programs while working with the tape subsystem and IBM compatible or ER90 devices.

For the examples in this chapter, it is assumed that you understand the `assign`(1) command. For more information, see the `assign`(1) command in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011 or in the *UNICOS/mk User Commands Reference Manual*, Cray Research publication SR–2611.

4.1 IBM compatible tape processing

This section briefly describes how to use the tape subsystem from Fortran programs using IBM compatible devices.

4.1.1 Reading and writing to tape

The following example illustrates the use of a Fortran program to read and write to tape:

1. Reserve a tape by using the `rsv`(1) command. In this example, the tape has a device group name of `TAPE` and the number of devices requested is `1`:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount with the `tpmnt`(1) command. In this example, it is a new, standard labeled tape with a volume identifier of `SCRSL`, a device group name of `TAPE`, a density of `6250`, a write ring specified to be on the reel, a block size of `32768` bytes, and the `-P` option overwriting the existing path name of `fort.20` with the newest version of `fort.20`.

   ```
   tpmnt -n -l sl -v SCRSL -g TAPE -d 6250 -r in -b 32768 -P fort.20
   ```

3. Use the `assign`(1) command to specify that file `fort.20` is a tape.

   ```
   assign -s tape f:fort.20
   ```

4. Compile and load the Fortran write program `tapewr.f`, directing the binary output to executable file `tapewr`:

   ```
   f90 -o tapewr tapewr.f
   ```

5. Execute `tapewr`, using the data from file `input` and appending the output to `tapewr.l`:

   ```
   tapewr < input >> tapewr.l
   ```

6. Compile, load, and execute the Fortran read program by repeating steps 4 and 5, using files `taperd.f`, `taperd`, and `taperd.l`:

   ```
   f90 -o taperd taperd.f
   taperd >> taperd.l
   ```

7. Release the resources:

   ```
   rls -a
   ```

Note the `write(20)` to `unit 20` in `tapewr` and the `read(20)` from `unit 20` in `taperd` reference steps 2 and 3, in which `fort.20` is used in the `tpmnt` and `assign` commands.

Figure 21 shows the Fortran write program, called `tapewr`. You must supply the `COMPUTE` routine:

```
PROGRAM TAPEWR
     INTEGER IBUF(10)
     REAL RNUM(5)
     CHARACTER*21 CDATA
     COMPLEX CNUM(3)
C
C     Write 5 records. Each record contains a mix of data types.
C
      DO 10 I=1,5
       CALL COMPUTE(I,IBUF,RNUM,CDATA,CNUM) ! Compute
       WRITE(20) IBUF,RNUM,CDATA,CNUM       ! Write them out.
  10 CONTINUE
     END
```

Figure 21. Writing an unlabeled tape

Figure 22 shows the Fortran read program, called `taperd`. You must supply the `ANALYZE` routine:

```
PROGRAM TAPERD
     INTEGER IBUF(10)
      REAL RNUM(5)
      CHARACTER*21 CDATA
      COMPLEX CNUM(3)
C
C     Read 5 records.
C
      DO 10 I=1,5
         READ(20) IBUF,RNUM,CDATA,CNUM        ! Read and convert data.
         CALL ANALYZE(I,IBUF,RNUM,CDATA,CNUM) ! analyze...
   10 CONTINUE
      END
```

Figure 22. Reading an unlabeled tape

4.1.2 Reading and writing tape marks

The following example illustrates the use of a Fortran program to read and write tape marks:

1. Reserve a tape by using the `rsv`(1) command:

   ```
   rsv TAPE 1
   ```

2. Compile and load Fortran program `tapemk.f`, directing the binary output to executable file `tapemk`:

   ```
   f90 -o tapemk tapemk.f
   ```

3. Request a tape mount by using the `tpmnt`(1)command. In this example, the tape has standard labels, a path name of `fort.1`, and a volume identifier of `SCRSL;` it uses the `-T` option to let you read or write tape marks:

   ```
   tpmnt -P fort.1 -l sl -v SCRSL -T -g TAPE -n
   ```

4. Use the `assign`(1) command to specify that file `fort.1` is a tape:

   ```
   assign -s tape f:fort.1
   ```

5. Execute `tapemk`:

   ```
   tapemk
   ```

6. Release the resources:

```
rls -a
```

Figure 23 shows the Fortran program that reads and writes a tape mark, called `tapemk`.

```
PROGRAM TAPEMK
      INTEGER BLOCK(1000)
C
C     Write 5 tape blocks, each followed by an end-of-file tape
C     mark (EOF).
C
      DO 10 I=1,5
         WRITE(1) BLOCK              ! Write out a tape block/record.
         ENDFILE(1)                  ! Write an EOF.
  10  CONTINUE

      REWIND 1

C
C     Read back the 5 tape blocks (records) and after each, read
C     the end-of-file tape mark (EOF).
C
      DO 20 I=1,5
        READ(1)         BLOCK           ! Read in a tape block/record.
        READ(1, END=20) TPMK           ! Read the EOF.
        PRINT *,' Error - no EOF.'     ! If no EOF, then error.
        STOP 'error'
      20 CONTINUE
      END
```

Figure 23. Reading and writing tape marks

4.1.3 Positioning a tape by blocks

The following example illustrates the use of a Fortran program to position a tape by blocks:

1. Reserve a tape by using the `rsv`(1) command:

```
rsv TAPE 1
```

2. Compile and load Fortran program `pos.f`, directing the binary output to the executable file `pos`:

   ```
   f90 -o pos pos.f
   ```

3. Request a tape mount with the `tpmnt`(1) command. In this example, the tape has standard labels, a path name of `fort.1`, and a volume identifier of `SCRSL:`

   ```
   tpmnt -p fort.1 -l sl -v SCRSL -g TAPE -n
   ```

4. Use the `assign`(1) command to specify that file `fort.1` is a tape:

5. Execute `pos`:

   ```
   assign -s tape f:fort.1

   pos
   ```

6. Release the resources:

   ```
   rls -a
   ```

Figure 24 shows the Fortran program that positions a tape by blocks, called `pos`.

```
PROGRAM POS
      INTEGER BLOCK(1000)
C
C     Write 5 records to the tape. (records 1-5)
C
      DO 10 I=1,5
        WRITE(1) BLOCK
   10 CONTINUE
C
C     Backspace the tape over the fifth record to position the
C     file after the fourth record on the tape.
C
      BACKSPACE 1
C
C     Rewrite record 5 with new data, and add records 6-9.
C
      DO 20 I=1,5
        WRITE(1) BLOCK
   20 CONTINUE

      END
```

Figure 24. Positioning by blocks

4.1.4 Positioning a tape by using the SETTP(3) library call

The following examples illustrate the use of the SETTP(3) library call in the positioning of a tape. For more information, see the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165.

4.1.4.1 Example 1

The following example illustrates the positioning of a tape on a multivolume file. The `tpos` program positions your tape to block number 50 on the second volume of a multivolume file.

1. Reserve a tape by using the `rsv`(1) command. In this example, the tape has a device group name of `TAPE` and the number of devices requested is `1`:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape is an old one with an IBM standard label, with three volume identifiers

of `VSN1:VSN2:VSN3`, a device group name of `TAPE`, a record format of fixed length, a block size of `64000` bytes, and a path name of `fort.1`:

```
tpmnt -o -l sl -v VSN1:VSN2:VSN3 -g TAPE -F F -b 64000 -p fort.1
```

3. Use the `assign`(1)) command to specify that file `fort.1` is a tape.

   ```
   assign -s tape f:fort.1
   ```

4. Compile and run the program shown in Figure 25:

```
     PROGRAM TPOS
IMPLICIT INTEGER (A - Z)
  DIMENSION BUFF(4096)
  RECLEN=4096
C
C  IN THE SETTP CALL BELOW, THE FIELDS CORRESPOND TO THE FOLLOWING:
C  FIELD 1: UNIT NUMBER.
C     2: BLOCK # REQUEST SIGN. '1H ' INDICATES THAT THE THIRD
C         FIELD (50) IS AN ABSOLUTE BLOCK # RELATIVE TO THE
C         BEGINNING OF THE VOLUME.
C     3: INTEGER BLOCK NUMBER
C     4: VOLUME # REQUEST SIGN. '1H ' INDICATES THAT THE FIFTH
C         FIELD (2) IS AN ABSOLUTE VOLUME # RELATIVE TO THE
C         BEGINNING OF THE VOLUME IDENTIFIER LIST SPECIFIED ON THE
C         TPMNT COMMAND.
C     5: INTEGER VOLUME NUMBER.
C     6: NAME OF VOLUME IDENTIFIER TO BE MOUNTED.
C          0 INDICATES THAT THIS PARAMETER IS IGNORED.
C     7: SPECIFIES WHETHER THE TAPE SHOULD BE SYNCHRONIZED
C         (0=NO).
C     8: INTEGER RETURN STATUS. ON EXIT, INDICATES WHETHER
C         POSITIONING WAS SUCCESSFUL OR NOT. 0 = SUCCESS;
C         NONZERO=ERROR OR WARNING.
C  POSITION TO THE 50TH BLOCK OF VOLUME 2:
  CALL SETTP(1,1H ,50,1H ,2,0,0,STAT)
C
  IF (STAT .NE. 0) THEN
    PRINT *,'SETTP ERROR: STAT = ',STAT
    CALL ABORT
  ENDIF
C
C  READ THE 50TH BLOCK ON VOLUME 2
C
  READ(1) (BUFF(N),N=1,RECLEN)
C
C  PROCESS THE DATA
C
  END
```

Figure 25. SETTP(3) positioning, example 1

4.1.4.2 Example 2

The following example shows you how to position forward nine blocks relative to the current position, and then read one block.

1. Reserve a tape by using the `rsv`(1) command. In this example, the tape has a device group name of `TAPE` and the number of devices requested is `1`:

   ```
   rsv TAPE 1
   ```

2. Request a tape mount by using the `tpmnt`(1)command. In this example, the tape is an old one with an IBM standard label, a volume identifier of `SCRNL`, a device group name of `TAPE`, a record format of fixed length, a block size of `64000` bytes, and a path name of `fort.1`:

   ```
   tpmnt -o -l sl -v SCRNL -g TAPE -F F -b 64000 -p fort.1
   ```

3. Use the `assign`(1) command to specify that file `fort.1` is a tape.

   ```
   assign -s tape f:fort.1
   ```

4. Compile and run the program shown in Figure 26:

```
PROGRAM TPOS
     IMPLICIT INTEGER (A - Z)
     DIMENSION BUFF(8000)
     RECLEN=8000
     NBLKS=200
C
     DO 500 I=1,NBLKS,10
C
C    SKIP 9 BLOCKS
C
     CALL SETTP(1,1H+,9,0,0,0,1,STAT)
     IF (STAT .NE. 0) THEN
       PRINT *,'SETTP ERROR: STAT = ',STAT
       CALL ABORT
     ENDIF
C
C    READ THE 10TH BLOCK
C
     READ(1) (BUFF(N),N=1,RECLEN)
C
C    PROCESS THE DATA
C
 500 CONTINUE
     END
```

Figure 26. SETTP(3) positioning, example 2

4.1.5 Reading and writing tapes containing foreign data

This section shows you how to convert foreign data to Cray Research data or convert Cray Research data to foreign data. Currently, the Fortran libraries support the translation and conversion of IBM, VAX/VMS, NOS/VE, IEEE, and CDC foreign data types. The two methods of foreign data conversion are the following:

- Explicit
- Implicit

4.1.5.1 Converting foreign data explicitly

This section shows you how to convert foreign data explicitly by using Fortran library data conversion routines to read tapes written on foreign computer

systems. For a complete list of Cray Research Fortran library data conversion routines, see the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165.

The following library conversion routines translate data in a foreign format into Cray Research format:

Routine	Description
IBM2CRAY(3)	Converts IBM data to Cray format
IEG2CRAY(3)	Converts IEEE or generic 32-bit data to Cray format
NVE2CRAY(3)	Converts NOS/VE data to Cray format
VAX2CRAY(3)	Converts VAX data to Cray format
CDC2CRAY(3)	Converts CDC 60-bit data to Cray format

The following library conversion routines translate data in Cray Research format into a foreign format:

Routine	Description
CRAY2IBM(3)	Converts Cray data to IBM format
CRAY2IEG(3)	Converts Cray format to IEEE or generic 32-bit data
CRAY2NVE(3)	Converts Cray data to NOS/VE format
CRAY2VAX(3)	Converts Cray data to VAX format
CRAY2CDC(3)	Converts Cray data to CDC 60-bit format

With explicit conversion you must specify the type and size of all data conversions before you can set up structures for the converted data; therefore, you must know the type and size of the data originally written on a tape.

Note: Be careful when specifying foreign record translation with the `-F` option on the `assign`(1) command. With most values of the `-F` options, you can only read, write, backspace, and rewind your tape. With the `-F ibm.u,tape`, `-F ibm.vbs,tape`, `-F ibm.vb,tape`, or `-F ibm.v,tape` option you can also use the `-d skipbad` option to request that bad data be automatically skipped. See the `assign`(1) man page for more details on the `-d` option. When you use the `-F bmx` or `-F tape` option, and do not specify any other FFIO layers, you can also use the Fortran tape positioning routines, process bad data, and do end-of-volume processing.

4.1.5.2 Example 1

This example illustrates the handling of data by using library conversion routines:

1. Reserve a tape by using the `rsv`(1) command. In this example, the device group name is `CART` and the number of devices requested is `1`:

   ```
   rsv CART 1
   ```

2. Compile Fortran program `ibmcvt.f`:

   ```
   f90 ibmcvt.f
   ```

3. Load the `ibmcvt.o` file, directing the output to executable file `ibmcvt`:

   ```
   segldr -o ibmcvt imbcvt.o
   ```

4. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has an IBM standard label, a volume identifier of `ISCSL`, a device group name of `CART`, a block size of `32768` bytes, and a path name of `fort.29`:

   ```
   tpmnt -l sl -v ISCSL -g CART -b 32768 -p fort.29 -n
   ```

5. Use the `assign`(1) command to specify that file `fort.1` is a tape.

   ```
   assign -s tape f:fort.29
   ```

6. Execute `ibmcvt`:

   ```
   ibmcvt
   ```

7. Release the reserved resource:

   ```
   rls -a
   ```

Figure 27 shows the Fortran program, called `ibmcvt.f`:

```
PROGRAM IBMCVT
 INTEGER CRAY2IBM,IBM2CRAY
C
C  REALA and REALB are the arrays that hold the CRAY format numbers.
C  IBMA and IBMB hold the converted IBM data. Note that they are half as
C  large as the CRAY real numbers, because the IBM real format is only 32
C  bits long. The type of the IBM array is not important, because no
C  computations will be performed on them. Only the proper amount of
C  space is important.
C
   REAL REALA(10000)
   REAL REALB(10000)
C
   INTEGER IBMA(5000)
   INTEGER IBMB(5000)
C
   CALL GENERATE(REALA)  ! REAL data is generated and converted to IBM format
C
C  The data produced is converted to IBM internal format and placed in
C  array IBMA. See the man pages for the CRAY2IBM(3) and IBM2CRAY(3)
C  routines.
C
   ISTAT = CRAY2IBM(    2,   ! data type, 2=REAL
   +       10000,   ! number of items to convert
   +        IBMA,   ! 'foreign' array
   +         0,   ! bit offset in IBMB
   +       REALA,   ! CRAY data
   +         1)   ! stride
   IF (ISTAT.LT.0) STOP 'error 1'  ! Check for conversion error.
C
C  Write the converted data to unit 29. No 'foreign' assign options should
C  be present on this unit, or the data will be converted twice!
C
   WRITE(29) IBMA
   REWIND 29
C
   READ(29) IBMB         ! Read the IBM format data back from the file.
```

```
C
   ISTAT = IBM2CRAY(   2,   ! data type, 2=REAL
   +      10000,   ! number of items to convert
   +       IBMB,   ! 'foreign' array
   +        0,   ! bit offset in IBMB
   +      REALB,   ! CRAY data
   +        1)   ! stride
   IF (ISTAT.LT.0) STOP 'error 1' ! Check for convert error.
C
   CALL PROCESS(REALB)
   END
```

Figure 27. Converting data to an IBM format

The data generated on Cray Research systems, converted to IBM format by the data conversion routines, is altered to fit the storage capabilities of IBM computer systems because the system storage limits of precision have been exceeded. The following list describes the way data is handled when it exceeds the limits of precision for IBM computer systems:

- Cray Research positive numbers (if they exceed IBM computer systems' limits of precision) are assigned the largest positive values allowed for integer or floating-point real numbers that can be expressed on IBM systems.
- Cray Research negative numbers with absolute values that exceed IBM computer systems' limits of precision are assigned the most negative value that can be expressed on IBM systems.
- Cray Research positive and negative numbers approaching zero, which are more precise than the smallest positive or negative fractional value that can be expressed on IBM computer systems, are assigned a value of 0.

If the data is generated on IBM computer systems and read on Cray Research systems, you do not need to be concerned with loss of precision in the conversion process. This is true of any computer system that has both a smaller exponent and a smaller mantissa size than that of Cray Research systems.

4.1.5.3 Example 2

The Fortran program fragment shown in Figure 28 illustrates how you might read an unknown number of records that are all of the same length and contain the same data type:

```
  .
      .
      .
      ICT = 0
10    CONTINUE
      READ(29,ERR=20,END=30) (ARRAY(I),I=1,LENGTH)
      ICT = ICT + 1
      CALL IBM2CRAY(ITYPE,LENGTH,ARRAY,0,NARRAY,1)
      CALL PROCESS(NARRAY)
      GOTO 10

20    print *,' Error in read on record ',ICT+1
      STOP 'error'

30    print *' End of File encountered on record ',ICT+1
      .
      . (continue program)
      .
```

Figure 28. Reading an unknown number of records

4.1.5.4 Example 3

The partial Fortran program shown in Figure 29 reads data records consisting of floating-point, integer, or character data (or any combination of these):

```
C  Read in IBM data records.
C
     READ(29,END=20,ERR=15) (A(I),I=1,NUM)
     READ(29,END=30,ERR=25) (B(I),I=1,CNT)
     READ(29,END=40,ERR=35) (C(I),I=1,NUMBER)
     READ(29,END=50,ERR=45) (D(I),I=1,COUNT)
C
C Now convert IBM data to Cray format. The fields of IBM2CRAY are:
C    Field 1: Type code. Indicates format of IBM data.
C         2: Number of data items to convert.
C         3: IBM array from which you are converting.
C         4: Bit number to begin conversion.
C         5: Array to contain the converted data.
C
     CALL   IBM2CRAY(7,NUM,A,1,SPR)       ! 7 indicates short integer
     CALL   IBM2CRAY(1,CNT,B,1,INT)       ! 1 indicates long integer
     CALL   IBM2CRAY(2,NUMBER,C,1,DPR)    ! 2 indicates short real
     CALL   IBM2CRAY(6,COUNT,D,1,CHR)     ! 6 indicates char (EBCDIC)
```

Figure 29. Reading mixed data types

Note: To correctly convert the data to be read, you must know the data types of the contents of the tape.

4.1.5.5 Converting foreign data implicitly

This section shows you how to translate the blocking structures and convert foreign data implicitly to Cray Research data and vice versa by using the `assign`(1) command. Currently, implicit conversion supports the translation and conversion of IBM, VAX/VMS, CDC (60-bit), NOS/VE, ULTRIX, and IEEE foreign data types.

The following example shows you how to read foreign data with Fortran programs using the `assign`(1) command:

1. Reserve a tape by using the `rsv`(1) command. In this example, the device group name is `TAPE` and the number of devices requested is `1`:

   ```
   rsv TAPE 1
   ```

2. Compile and load Fortran program `impread.f`, directing the output to the executable file `impread`:

   ```
   f90 -o impread impread.f
   ```

3. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has an IBM standard label, a volume identifier of `SCRSL`, a device group name of `TAPE`, a block size of `32768` bytes, and a path name of `fort.29`:

   ```
   tpmnt -l sl -v SCRSL -g TAPE -b 32768 -p fort.29
   ```

4. Request implicit data conversion. In this example, specify (by using the `-F` option) that the blocking of the tape is in the IBM `VBS` format, specify (by using the `-N` option) that the numeric data to be converted is IBM data, and use the `assign`(1) command to specify that file `fort.29` is a tape:

   ```
   assign -F ibm.vbs,tape -N ibm f:fort.29
   ```

5. Execute `impread`:

   ```
   impread
   ```

6. Release the reserved resource:

   ```
   rls -a
   ```

Figure 30 shows the Fortran program, called `impread.f`:

```
PROGRAM IMPREAD
      INTEGER INT
      REAL RL(4)
      COMPLEX COM(3)
C
C     THIS PROGRAM READS DATA IN FROM THE DATA FILE AND PRINTS IT.
C     THE RECORD FORMAT OPTION (-F) AND DATA CONVERSION OPTIONS (-N)
C     MUST BE SPECIFIED IN THE ASSIGN COMMAND.
C
C     THE RUN-TIME LIBRARY WILL DEBLOCK THE FOREIGN RECORD(S) AND
C     CONVERT THE DATA ITEMS AS REQUESTED IN THE ASSIGN COMMAND.
C
      DO 10 I=1,10
        READ(29) INT,RL,COM
        PRINT *,'REC=',INT
        PRINT *,' RL=',RL
        PRINT *,'COM=',COM
 10   CONTINUE
      END
```

Figure 30. Converting foreign data

4.1.6 Using the bad data recovery routines

This section shows you how to use the bad data recovery routines, which allow you to skip or accept bad data. These routines check the status of the I/O function. If an error has occurred, these routines call a Fortran error-handling routine. For more information on the Fortran error handling routines, see the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165.

The Fortran error-handling routines available are as follows:

Routine	Description
SKIPBAD(3)	Skips bad data

ACPTBAD(3) Makes bad data available

4.1.6.1 Example 1

Figure 31 shows the following example, called `skipbdxmp`. This example illustrates how to use the Fortran error-handling routine `SKIPBAD`, which skips bad data on the read operation:

```
PROGRAM SKIPBDXMP
      IMPLICIT INTEGER (A-Z)
      PARAMETER (MAXSIZ=50000)
      DIMENSION BUFFER(MAXSIZ),UDA(512)

      NUMBLKS=1000000
      NWORDS=4096

      DO 5000 NBLK=1,NUMBLKS
      CALL READ(99,BUFFER,NWORDS,STATUS)
      IF(STATUS .EQ. 0) GO TO 5000

      IF(STATUS .EQ. 4) THEN
        PRINT*,'****PARITY ERROR ON READ AT RECORD',NBLK
        NPAR=NPAR+1
        GO TO 2500
      ENDIF

      IF(STATUS .EQ. 2) THEN
        PRINT*,'*****END OF FILE DETECTED,RECORDS=',NBLK
        STOP
      ENDIF

      IF (STATUS .NE. 0) THEN
        PRINT*,'UNEXPECTED RETURN CODE FROM READ=',STATUS
        CALL ABORT
      ENDIF

2500 CALL SKIPBAD (99,BLKS, TERMCND)
PRINT*,'SKIPBAD-BLOCKS SKIPPED',BLKS
      PRINT*,'STATUS EQUALS',TERMCND

      IF (TERMCND .EQ. 0) GO TO 5000
      IF (TERMCND .EQ. 1) THEN
        PRINT*,'****END OF FILE DETECTED, RECORDS READ=',NBLK
        PRINT*,'****NUMBER OF PARITY ERRORS=',NPAR
        STOP
      ENDIF
```

```
      IF (TERMCND .LT. 0) THEN
         PRINT*,'****NOT ON A RECORD BOUNDARY,ABORTING'
         CALL ABORT
       ENDIF
5000 CONTINUE
       STOP
       END
```

Figure 31. Using the SKIPBAD(3) routine

4.1.6.2 Example 2

Figure 32 shows you how to use the Fortran error-handling routine called ACPTBAD(3), which accepts bad data on the read operation):

```
PROGRAM ACPTBAD

      IMPLICIT INTEGER (a-z)
      PARAMETER (NUMBLKS=10000)
      PARAMETER (MAXSIZE=50000)
      PARAMETER (RECLEN=4096)
      DIMENSION BUFFER(MAXSIZE),UDA(MAXSIZE)

      NPAR = 0
      DO 5000 NBLK=1,NUMBLKS
         NWORDS=RECLEN
         CALL READ(1,BUFFER,NWORDS,STATUS)
         IF (STATUS .EQ. 0) GO TO 5000
IF (STATUS .EQ. 4) THEN
           PRINT *,'***PARITY ERROR ON READ AT RECORD ',NBLK
           NPAR = NPAR+1
           GO TO 2500
      ENDIF

      IF ((STATUS .EQ . 2) .OR. (STATUS .EQ. 3))THEN
         PRINT *,'END OF FILE/DATA DETECTED, RECORDS= ',NBLK
         STOP 'COMPLETE'
      ENDIF

      IF (STATUS .NE. 0)THEN
         PRINT *,'UNEXPECTED RETURN CODE FROM READ = ',STATUS,NBLK
         CALL ABORT
      ENDIF
```

```
2500 CALL ACPTBAD(1,UDA,CNT,TERMCND,UBC,MAXSIZE)

C
C     BUILD UP USER RECORD
C
      IX = 0
      DO 3500 I = (NWORDS+1),(NWORDS+CNT)
         IX=IX+1
         BUFFER(I)=UDA(IX)

3500  CONTINUE

      IF (TERMCND .LT. 0)THEN
         PRINT *, 'END OF RECORD NOT REACHED'
      ENDIF
      IF (TERMCND .EQ. 1)THEN
         PRINT *,'**** END OF FILE DETECTED,RECORDS = ',NBLK
         PRINT *,'**** NUMBER OF PARITY ERRORS = ',NPAR
      ENDIF
5000 CONTINUE
      END
```

Figure 32. Using the ACPTBAD(3) routine

4.1.6.3 Example 3

Figure 33 shows you how to reserve, mount, and release tapes from inside your Fortran program, using the ISHELL(3) routine. If you use this routine, you must make sure that the tape file is closed before the tape is released. Use the Fortran CLOSE statement to close a file.

```
PROGRAM TP1
 DIMENSION IBUF(500)
 INTEGER ISHELL

C RESERVE A CART DEVICE

 ISTAT = ISHELL('rsv CART 1')
 IF (ISTAT.NE.0) GOTO 100

C REQUEST MOUNT OF DESIRED CART

 ISTAT = ISHELL('tpmnt -l al -v ISCAL -p fort.10 -g CART -n')
 IF (ISTAT.NE.0) GOTO 200

C  WRITE TO THE TAPE. YOU MUST HAVE PREVIOUSLY USED THE ASSIGN
C COMMAND TO IDENTIFY UNIT 10 AS A TAPE.

 DO 10 I = 1,500
 WRITE(10)IBUF
 10 CONTINUE

C BEFORE RELEASING THE TAPE, THE FILE MUST BE CLOSED.
 CLOSE(10)

C RELEASE THE TAPE, BUT KEEP THE CART RESOURCE.

 ISTAT = ISHELL('rls -k -p fort.10')
 IF (ISTAT.NE.0) GOTO 300

C  REQUEST MOUNT OF ANOTHER TAPE

 ISTAT =ISHELL('tpmnt -l sl -v ISCSL -p fort.10 -g CART -n')
 IF (ISTAT.NE.0) GOTO 200
C WRITE TO TAPE

 DO 20 I = 1,500
 WRITE(10) IBUF
 20
```

```
C CLOSE THE FILE AND RELEASE ALL TAPE RESOURCES

 CLOSE(10)
 ISTAT = ISHELL('rls -a')
 IF (ISTAT.NE.0) GOTO 300
 STOP

100 PRINT *,'RSV FAILED'
 STOP

 200 PRINT *,'TPMNT FAILED'
 ISTAT = ISHELL('rls -a')
 STOP

 300 PRINT *,'RLS FAILED'
 PRINT *,'ISTAT = ',ISTAT
 STOP
 END
```

Figure 33. Using the `ISHELL`(3) routine

To execute the preceding program, `tp1`, type the following:

```
f90 -o tp1 tp1.f
assign -s tape f:fort.10./tp1
```

4.1.7 Using end-of-volume processing requests

This section describes user end-of-volume (EOV) processing from a Fortran program. For information about user EOV processing from a C program, see Chapter 5, page 81.

Normally, volume switching is handled by the tape subsystem and is transparent to you. However, when user EOV processing is requested, you gain control at the end-of-tape and your program may perform special processing. For more information on the Fortran interface routines used in EOV processing, see the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165. The library interface routines for EOV processing from a Fortran program are as follows:

Routine	Description
SETSP(3)	Enables or disables EOV processing
STARTSP(3)	Starts special tape processing
ENDSP(3)	Ends special tape processing
CHECKTP(3)	Checks tape position
CLOSEV(3)	Closes volume and mounts next volume specified in the volume identifier list

When using EOV processing for online tape files on the tape subsystem, make sure that data is flushed from the library and system buffers before calling certain routines as discussed in the following. Failure to flush the buffers and check for EOV can result in lost data at the end of the tape volume.

To instruct the system to perform EOV processing, call the SETSP routine with the appropriate parameter set to ON after a tape file is opened.

Check for EOV by calling the CHECKTP macro. To test whether a tape is at EOV, you must call CHECKTP after each WRITE, ENDFILE, or READ operation. In addition, for an output dataset, call CHECKTP after each GETTP or GETPOS call to see if EOV was encountered.

For output datasets, you should also ensure that the library and system have flushed their buffers, and then test whether the tape is at EOV, before issuing any of these statements:

CLOSE

REWIND

BACKSPACE

and before calling any of these routines:

SETTP(3)

SETPOS(3)

CLOSEV(3)

SETSP(3) (OFF)

To flush the buffers, call GETTP(3) with the SYNCH parameter set to ON. Then call CHECKTP(3) to see if EOV was reached.

4.1.7.1 Example 1

The following example shows you how to use EOV processing by using the library interface routines:

1. Reserve a tape by using the `rsv`(1) command:

   ```
   rsv CART 1
   ```

2. Compile and load Fortran program `teov.f`, directing the output to executable file `teov`:

   ```
   f90 -o teov teov.f
   ```

3. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has no label, a path name of `fort.9`, and volume identifiers of `x` and `y`:

   ```
   tpmnt -l nl -P fort.9 -v x:y -g CART -n
   ```

4. Use the `assign`(1) command to specify that file `fort.9` is a tape:

   ```
   assign -s tape f:fort.9
   ```

5. Execute `teov`:

   ```
   teov
   ```

6. Release the reserved resource:

   ```
   rls -a
   ```

Figure 34 shows the Fortran program, called `teov.f`:

```
 PROGRAM TEOV
      IMPLICIT INTEGER(A-Z)
      PARAMETER (BLKLEN = 512)

C     MAXREC = total number of records to be written
      PARAMETER (MAXREC = 10000)
      DIMENSION BLK(BLKLEN)
      INTEGER IPA(45)

C     Set up for special EOV processing
      CALL SETSP(9, 1, ISTAT)
      IF (ISTAT .NE. 0) GOTO 20

C     Write MAXREC records. Check for end-of-volume after each write.
C     If end-of-volume is detected, call the subroutine EOVPROC
      DO 10 I = 1, MAXREC
        WRITE (9)BLK
        CALL CHECKTP(9, ISTAT, ICBUF)
        IF (ISTAT. EQ. 0) THEN
C       At EOV
        CALL EOVPROC()
        ENDIF
10    CONTINUE
C     We have written all of the records. Call GETTP with the
C     sync parameter set to flush the library's and system's buffers.
      CALL GETTP(9, 40, IPA, 1, IREPLY)
      IF (IREPLY.NE. 0)GOTO 20
      CALL CHECKTP(9, ISTAT, ICBUF)
      IF (ISTAT. EQ. 0) THEN
C       At EOV
        CALL EOVPROC()
      ENDIF
C     Stop end-of-volume processing
      CALL ENDSP (9, ISTAT)
      IF (ISTAT .NE. 0)GOTO 20
C     Close the file
      CLOSE(9)
C     STOP
20    PRINT *,'ERROR'
      END
```

```
SUBROUTINE EOVPROC()
      DIMENSION TBLK(512)
      DIMENSION HBLK(512)

C     Start special processing at eov.
      CALL STARTSP(9, ISTAT)
      IF (ISTAT. NE. 0) GOTO 20

C     Write a special block at the end of the tape
C     and close the volume.
      WRITE (9) TBLK
      CALL CLOSEV(9, ISTAT)
      IF (ISTAT. NE. 0) GOTO 20

C     Write a special block at the beginning of the next tape.
      WRITE (9) HBLK

C     Stop special processing
      CALL ENDSP (9, ISTAT)
      IF (ISTAT. NE. 0) GOTO 20
      RETURN
20    PRINT *,'ERROR'
      STOP
      END
```

Figure 34. Using Fortran library routines for EOV processing

4.1.7.2 Example 2

The following example illustrates EOV processing when writing a tape file. The program writes until end-of-volume is reached. It then reads the last two blocks on the first volume and the blocks buffered in the IOS, and writes these on the second tape volume:

1. Reserve a tape by using the `rsv`(1) command:

   ```
   rsv CART 1
   ```

2. Compile and load Fortran program `teov2.f`, directing the output to executable file `teov2`:

   ```
   f90 -o teov2 teov2.f
   ```

3. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has no label, a path name of `fort.9`, and volume identifiers of `VOL1` and `VOL2`:

   ```
   tpmnt -l nl -v VOL1:VOL2 -p fort.9 -g CART -r in -n -T
   ```

4. Use the `assign`(1) command to specify that file `fort.9` is a tape.

   ```
   assign -s tape f:fort.9
   ```

5. Execute `teov2`:

   ```
   teov2
   ```

6. Release the reserved resources:

   ```
   rls -a
   ```

Figure 35 shows the Fortran program, called `teov2.f`:

```
 PROGRAM TEOV2

c    Example of EOV processing. Assumes that fort.9 is a tape file

     IMPLICIT INTEGER(A-Z)
     PARAMETER (BLKLEN = 512, PALEN = 30)
     DIMENSION BLK(BLKLEN), PA(PALEN)

c    Set up for special EOV processing.

     CALL SETSP(9,1,ISTAT)
     IF (ISTAT.NE.0) GOTO 100

c    Fill the first volume. CHECKTP returns a status that indicates
c    whether end-of-volume has been reached.

10   CONTINUE
     WRITE(9)BLK
     CALL CHECKTP(9,ISTAT,ICBUF)
     IF (ISTAT.LT.0) GOTO 10

c    Determine number of blocks buffered in the IOS.
C    Note that we do not request synch here.

     CALL GETTP(9, PALEN, PA, 0, ISTAT)
     IF (ISTAT.NE.0) GOTO 100

c    Start special processing
     CALL STARTSP(9,ISTAT)
     IF (ISTAT.NE.0) GOTO 100

c    Backspace 2 blocks. Read these 2 blocks from tape + blocks from
c    the IOS + blocks in the library buffer and store them in a
c    temporary file (fort.10)

     BACKSPACE(9)
     BACKSPACE(9)
     NBLK=PA(12)+2+PA(11)           ! blocks in IOS + 2 from tape
                                ! + blocks in library buffer
     DO 20 I = 1,NBLK
        READ(9)BLK
```

```
20   WRITE(10)BLK
c    Backspace 2 blocks before closing the volume, because these
c    2 blocks will be rewritten on the second volume.

     BACKSPACE(9)
     BACKSPACE(9)

c    The programmer wants to write a tape mark at EOV
     ENDFILE(9)

c    Close the volume and request mount of the next volume

     CALL CLOSEV(9,ISTAT)
     IF (ISTAT.NE.0) GOTO 100

     CALL ENDSP(9,ISTAT)            ! stop special processing
     IF (ISTAT.NE.0) GOTO 100

c    Disable special processing

     CALL SETSP(9, 0, ISTAT)
     IF (ISTAT.NE.0) GOTO 100

     REWIND(10)                    ! rewind temporary file

c    Write the blocks in fort.10 onto the second volume

     DO 30 I = 1,NBLK
        READ(10)BLK
30   WRITE(10)BLK

     CLOSE(9)                 ! close the file
     STOP
100  CALL ABORT()
     END
```

Figure 35. Using EOV processing when writing a file

4.1.7.3 Example 3

The following example shows how to use EOV processing to detect the end-of-volume when reading a multivolume file.

1. Reserve a tape by using the `rsv`(1) command:

   ```
   rsv
   ```

2. Compile and load the Fortran program `eovr.f` directing the output to executable file `eovr`:

   ```
   f90 -o eovr eovr.f
   ```

3. Request a tape mount by using the `tpmnt`(1) command. In this example, the file has a standard label, a path name of `fort.10`, and volume identifiers of `x` and `y`:

   ```
   tpmnt -l sl -p fort.10 -v x:y
   ```

4. Use the `rsv`(1) command to specify that file `fort.10` is a tape:

   ```
   assign -s tape f:fort.10
   ```

5. Execute `eovr`:

   ```
   eovr
   ```

6. Release the reserved resource:

   ```
   rls -p fort.10
   ```

Figure 36 shows the Fortran program, called `eovr.f`:

```
  PROGRAM EOVR

      IMPLICIT INTEGER (A-Z)
      PARAMETER (BLKLEN=4096)

      DIMENSION BLK(BLKLEN)

      CALL SETSP(10,1,ISTAT)
      IF (ISTAT.NE.0) THEN
        PRINT *,'BAD STATUS FROM SETSP ',ISTAT
        GOTO 100
      ENDIF

c     Read until we get to the end of a volume or the end of data

10    CONTINUE

      IC = BLKLEN
      CALL READ(10, BLK, IC, ISTAT)
      IF ((ISTAT.EQ.2).OR.(ISTAT.EQ.3)) THEN
        PRINT *,'END OF FILE/DATA'
        GOTO 100
      ELSEIF (ISTAT.NE.0) THEN
        PRINT *,'UNEXPECTED STATUS FROM READ ',ISTAT
        GOTO 100
      ENDIF

c     Check for end of volume

      CALL CHECKTP(10,REPLY,CB)
      IF (REPLY .LT. 0 ) GOTO 10
      IF (REPLY .GT. 0 ) THEN
        PRINT *,'UNEXPECTED STATUS FROM CHECKTP ',REPLY
        GOTO 100
      ENDIF

c     At EOV. Start special processing, and call CLOSEV to mount the
c     next tape
```

```
CALL STARTSP(10,ISTAT)
      IF (ISTAT.NE.0)THEN
        PRINT *,'BAD STATUS FROM STARTSP ',ISTAT
        GOTO 100
      ENDIF
      CALL CLOSEV(10,ISTAT)
      IF (ISTAT.NE.0)THEN
        PRINT *,'BAD STATUS FROM CLOSEV ',ISTAT
        GOTO 100
      ENDIF
      CALL ENDSP(10,1,ISTAT)
      IF (ISTAT.NE.0) THEN
        PRINT *,'BAD STATUS FROM ENDSP ',ISTAT
        GOTO 100
      ENDIF
      GOTO 10

c     Disable EOV processing
100   CALL SETSP(10,0,ISTAT)
      CLOSE(10)
      END
```

Figure 36. Using EOV processing when reading a multivolume file

4.2 ER90 tape processing

You can access ER90 devices through Fortran. Unformatted I/O is currently supported. You can select either the byte-stream mode or block mode of the device.

Note: The ER90 format is not available on systems that run the UNICOS/mk operating system or that have GigaRing support.

In byte-stream mode, two processing classes are available:

- Pure data mode
- COS blocking

To select the processing class, use the `assign`(1) command.

In block mode, select the FFIO tape layer by using the following command:

```
assign -F tape
```

With this processing class, each Fortran record corresponds to a block. Because ER90 devices require that each block be the size specified by the `-b` option of the `tpmnt`(1) command, all Fortran records must be the same size. An exception to this rule is the last record written before a tape mark or the end-of-file. This record may be smaller than the size specified by the `-b` option. When you choose this processing class, EOV processing routines, user tape marks, and the `SETTP`(3) routine are available.

4.2.1 Using pure data mode

In pure data mode, no record control words are written to the file. This indicates that the user must know the size of the records being read. Reading, writing, and rewinding are allowed in this mode.

Use the `assign`(1) command to select this mode, as follows:

```
assign -F er90 assign.object
```

Pure data mode does not support EOV processing, `SETTP`(3), `SKIPF`(3), and multiple `ENDFILES`. `GETTP` is allowed, but the meaning of some fields that are returned in the information array differs from that returned when using round or cartridge tapes and the following `assign`(1) command:

```
assign -[F,s] [tape,bmx]
```

`GETPOS`(3) and `SETPOS`(3) are supported when using this processing class. The `len` parameter for these routines must be at least 4. The values returned by `GETPOS`(3) in the `pa` array contain device specific information; it may be used in a subsequent call to `SETPOS`(3). For ER90 files, the information returned does not include the volume serial number (VSN) or partition information. Before using `SETPOS`(3), verify that the correct VSN and partition is in position.

When using round or cartridge tapes, each Fortran record corresponds to a physical tape block. When using the ER90 device in byte-stream mode, each byte is considered a block. Therefore, the `ipa(10)`, `ipa(11)`, and `ipa(12)` fields return byte counts.

When the file is assigned with `-F er90`, each Fortran read or write results in one or more system calls. The `bufa` layer may be used to provide asynchronous buffering, potentially reducing the number of system calls and improving performance. It may be combined with the `er90` layer as follows:

```
assign -F bufa,er90 assign.object
```

The following example illustrates the use of pure data mode with the ER90 device.

1. Reserve a device by using the`rsv`(1) command:

   ```
   rsv ER90
   ```

2. Compile the Fortran program `tpwr1.f`v, resulting in the relocatable file `ctpwr1.o`:

   ```
   f90 tpwr1.f
   ```

3. By default, the ER90 flexible file I/O (FFIO) layer is disabled. It can be enabled by the system administrator, or by specifying the `ff_er90` loader directives file.

   Load the relocatable file `tpwr1.o`, directing the binary output to the executable file `tpwr1` using the following `segldr`(1) command:

   ```
   segldr -o tpwr1 tpwr1.o -j ff_er90
   ```

4. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has no label, a path name of `fort.1`, and a volume identifier of `00011`:

   ```
   tpmnt -p fort.1 -l nl -v 00011 -g ER90
   ```

5. Use the `assign`(1) command to specify that file `fort.1` is an ER90 file with asynchronous buffering:

   ```
   assign -F bufa,er90 fort.1
   ```

6. Execute `tpwr`:

   ```
   ./tpwr1
   ```

7. Release the reserved resources:

   ```
   rls -a
   ```

Figure 37 shows the Fortran program, called `tpwr1.f`:

```
program tpwr1
      integer buf(2000)
      integer ipa(4)
c     write 100 records
      do 10 i = 1,100
        do 5 j = 1,2000
          buf(j) = i
5     continue
      write(1)buf
      if (i.eq.50)then
        call getpos(1, 4, ipa, istat)
        if (istat.ne.0)then
          print *,'bad stat ', istat
          stop 'error'
        endif
10    endif
      rewind(1)

c     read the records and verify
      do 20 i = 1,100
        read(1) buff
        do 15 j = 1,2000
          if (buf(j).ne.i)then
            print *,'bad data ', buf(j),i
            stop 'error'
          endif
15    continue
20    continue
c     now position back to the point where we did the getpos
      call setpos(1, 4, ipa, istat)
      if (istat.ne.0)then
        print *,'bad stat ',istat
         stop 'error'
      endif
      read(1)buf
      if (buf(1).ne.51)then
        print *,'bad data after setpos ', buf(1)
      endif
      end
```

Figure 37. Using pure data mode

4.2.2 Using COS blocking mode

In COS blocking mode, users can read, write, rewind, and backspace.

Use the `assign`(1) command to select COS blocking for the ER90, as follows:

`assign -F cos,er90` *assign.object*

The COS blocking mode does not support EOV processing, `SKIPF`(3), `CLOSEV`(3), `SETTP`(3), and concatenated tape files. `GETTP`(3) is allowed, but some of the fields that are returned in the information array differ from those returned when using round or cartridge tapes. The COS blocking layer buffers data, and it is not included in the value returned by `GETTP` in `ipa(11).`

`GETPOS`(3) and `SETPOS`(3) are supported when using this processing class. The `len` parameter for these routines must be 6 or higher. The values returned by `GETPOS` in the `pa` array contain device specific information.

The following example illustrates the use of COS blocking mode with the ER90 device.

1. Reserve a device by using the `rsv`(1) command:

   ```
   rsv ER90
   ```

2. Compile the Fortran program `tpwr2.f`, resulting in the relocatable file `tpwr2.o`:

   ```
   f90 tpwr2.f
   ```

3. By default, the ER90 flexible file I/O (FFIO) layer is disabled. It can be enabled by the system administrator, or by specifying the `ff_er90` loader directives file.

   Load the relocatable file `tpwr2.o`, directing the binary output to the executable file `tpwr1` using the following `segldr`(1) command:

   ```
   segldr -o tpwr2 tpwr2.o -j ff_er90
   ```

4. Request a tape mount by using the `tpmnt`(1) command. In this example, the tape has no label, a path name of `fort.1`, and a volume identifier of `00011`:

   ```
   tpmnt -p fort.1 -l nl -v 00011 -g ER90
   ```

5. Use the `assign`(1) command to specify that file `fort.1` is an ER90 file:

   ```
   assign -F er90 fort.1
   ```

6. Execute `tpwr`:

   ```
   ./tpwr2
   ```

7. Release the reserved resources:

   ```
   rls -a
   ```

Figure 38 shows the Fortran program `tpwr2.f`:

```
 program tpwr2
      real rbuf(2000)
c     write 100 records
      do 10 i = 1,100
        do 5 j = 1,2000
          rbuf(j)=i
5     continue
      write(1) rbuf
10    continue
      rewind(1)
c     read the records and verify
      do 20 i = 1,100
         read(1) rbuf
        do 15 j=1,2000
          if (rbuf(j).ne.i)then
            print *,' bad data ', rbuf(j),i
            stop
        endif
15    continue
20    continue
      end
```

Figure 38. Using COS blocking mode

Writing C Applications Using Tapes [5]

This chapter describes the ways in which you may work with the tape subsystem with C programs.

Before you can access the tape subsystem for file processing, you must reserve the required number of tape drives for each device type needed. After you have reserved the tape drives, you may specify the tape volume in which the files to be processed are located.

After you have the volumes mounted and positioned, you can begin processing the tape files. When processing is complete, release the reserved tape drives.

There are two levels of access to the tape subsystem. The recommended and easiest to use is the C library level, using flexible file routines. The second level is to use system calls, which requires much greater detail than the C library level.

This chapter discusses accessing the tape subsystem with the following approaches:

- C flexible file I/O library routines
- System call I/O
- Tape information requests
- Tape positioning requests
- End-of-volume requests
- Tape control requests

Note: The ER90 format is not available on systems that run the UNICOS/mk operating system or that have GigaRing support.

5.1 C flexible file I/O library routines

The flexible file I/O (FFIO) routines provide another way to perform tape I/O with the ease of use of system calls. The FFIO routines automatically recognize tape devices and use the appropriate buffering.

The C library routines `ffopen`(3), `ffread`(3), `ffwrite`(3), `ffseek`(3), `ffbksp`(3), `ffclose`(3), and `ffwrite`(3) provide the capability to read and

write records to tape, rewind the tape and backspace records, and read and write tape marks.

For IBM compatible devices, the `ffread`(3) and `ffwrite`(3) routines provide an interface that is sensitive to block boundaries and that returns information on tape block boundaries on request. For ER90 devices, `ffread`(3) and `ffwrite`(3) provide a way to perform I/O by using either the byte-stream mode or block mode of the device. With the FFIO layer, a rewind operation can be performed simply with a call to `ffseek`(3). Tape marks can be written with `ffweof`(3) (`ffweof`(3) is not supported for ER90 devices in byte-stream mode), and tape marks can be read with `ffread`(3). A call to `ffwrite`(3) can write a tape block of a designated number of bytes on a tape. A call to `ffread`(3) can read up to one tape block from a tape. Explicit information about tape block boundaries and the ability to read and write partial tape blocks is available through the use of optional parameters on `ffread`(3) and `ffwrite`(3).

The `ffpos`(3) and `fffcntl`(3) routines provide the same complete set of capabilities as available from Fortran including additional positioning, access to information about the current tape, and end-of-volume processing. The `ffpos`(3) and `fffcntl`(3) routines are available on all systems. Some of the functionality available with `ffpos`(3) and `fffcntl`(3) on IBM compatible devices are not available on ER90 devices.

The FFIO tape layer may be used with either byte-stream mode or block mode of the ER90 devices. When you use byte-stream mode, EOV processing, user tape marks, and some positioning functionality are not available with the FFIO tape layer. When you use block mode and the FFIO tape layer, each record written must be the same size as specified on the `-b` option of the `tpmnt`(3) command. An exception to this rule is the last record written before a tape mark or the end-of-file.

Figure 39 shows a program, called `cexam.c`. This program demonstrates how these routines can be used. For more information on the C library routines, see the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080 or the *UNICOS/mk System Libraries Reference Manual*, Cray Research publication SR–2680. For detailed information about I/O, see the *Application Programmer's I/O Guide*, Cray Research publication SG–2168.

```
#include <fcntl.h>
#include <sys/types.h>
#include <foreign.h>
#include <errno.h>
main()
{
   int ffd;
   int i,j;
   int buf[2000];
   int ret;
   ffd = ffopen("mytape", O_RDWR);
   if (ffd<0){
      printf("open failed, error = %d\n",errno)
      exit(1);
   }

    /************** Write 10 records, a tape mark, and 10 more records to tape */
    for (j = 0; j < 2; j++){
      for (i = 0; i < 10; i++){
         ret = ffwrite(ffd, buf, 800);
         if (ret < 800){
         printf("ffwrite returned %d\n",ret);
         printf("error = %d\n",errno);
         }
      }

    /************** Write a tapemark */
      ret = ffweof(ffd);
      if (ret < 0)
         printf("ffweof failed, error = %d\n",errno);
   }
    /************** Rewind the tape */
    ret = ffseek(ffd,0,0);
    if (ret != 0)
      printf("ffseek failed, error = %d\n",errno);
```

```
   /************** Read the tape until the first tape mark is reached. */
    for (;;){
      ret = ffread(ffd, buf, 16000);
      if (ret < 0) {
         printf("ffread failed, error = %d\n",errno);
         break;
      }
      else if (ret == 0)
         break;
   /* Just read a tape mark */
      else
         printf("We read %d bytes\n",ret);
   }

    /************** Close the file */
    ffclose(ffd);
}
```

Figure 39. C library routine usage

Figure 40 shows how to execute `cexam.c`:

```
cc cexam.c
rsv CART 1
tpmnt -v ISCSL -l sl -p mytape -g CART -r in -n -T
assign -F tape mytape
./a.out
rls -a
```

Figure 40. Executing `cexam.c`

The `fffcntl`(3) routine provides the capability to detect tape end-of-volume, and to do special end-of-volume processing. An example of special end-of-volume processing using the FFIO routines follows.

For more information about end-of-volume processing, see Section 4.1.7, page 65. As described in this section, you must check for EOV after each `ffwrite`(3), `ffweof`(3), or `ffread`(3) when EOV processing is requested. For output data sets, check for EOV after each `fffcntl`(3) using `cmd FC_GETTP`

or `cmd FP_GETPOS`. For output data sets, you should also ensure that the library and system have flushed their buffers, and then test whether the tape is at EOV, before calling any of the following routines:

- `ffseek`(3)
- `ffpos`(3) (with `cmd FP_SETPOS`, `FP_SETTP`, `FP_SKIPF`, or `FP_BSEEK`)
- `ffclose`(3)
- `fffcntl`(3) (with `cmd FP_CLOSEV` or `FP_SETSP(off)`)

To flush the buffers, call `fffcntl`(3) using `cmd FC_GETTP` and with the structure field `ffc_synch` set to 1.

To execute `cexam2.c`, shown in Figure 41, enter:

```
cc cexam2.c
rsv TAPE 1
tpmnt -v VOL1:VOL2 -g TAPE -p mytape -r in -n -T
./a.out
rls -a
```

Figure 41. Executing `cexam2.c`

The `fffcntl`(3) and `ffpos`(3) routines, shown in Figure 42, are on all systems. EOV processing with `fffcntl`(3) is not available for ER90 devices.

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/iosw.h>
#include <foreign.h>
#include <errno.h>

#define BUFSIZ 4000
#define ERREXIT(a, b) {printf("%s error = %d\n",a,b); exit(1); }

main()
{
        int ffd, fftmp;
        int i;
        long bufcnt;
        int buf[BUFSIZ];
        int ret, eov = 0;
        struct ffc_chktp_s checktp;
        struct ffc_gettp_s gettp;
        struct ffp_settp_s settp;
        long pa[40];
        struct ffsw stat;

        ffd = ffopens("mytape",O_RDWR,0,0,&stat,"tape");
        if (ffd < 0)
                ERREXIT("open failed ",stat.sw_error);
        /*
         * Initiate special end-of-volume processing
         */
        if (ffsetsp(ffd, &stat) < 0)
                ERREXIT("ffsetsp failed ",errno);
        /*
         * Write until we reach EOV
         */
        do {
                if (ffwrite(ffd,buf,BUFSIZ) != BUFSIZ)
                        ERREXIT("ffwrite failed ",errno);
```

```
        /*
         * We must check for EOV status after each write
         */
        if (fffcntl(ffd, FC_CHECKTP, &checktp, &stat) < 0)
                ERREXIT("CHECKTP failed ",stat.sw_error);
        if (checktp.stat == 0)
                eov = 1;        /* Have reached eov */
} while(!eov);

/* Determine how many blocks are buffered */
gettp.ffc_glen = 40;
gettp.ffc_synch = 0;
gettp.ffc_pa = pa;
if (fffcntl(ffd, FC_GETTP, &gettp, &stat) < 0)
        ERREXIT("GETTP failed ",stat.sw_error);
bufcnt = pa[10] + pa[11];        /* blocks in library buffer + system */
/*
 * Start special end-of-volume processing
 */
if (fffcntl(ffd, FC_STARTSP, 0, &stat) < 0)
        ERREXIT("STARTSP failed ",stat.sw_error);
/*
 * We will write the last 2 blocks on this volume and the
 * blocks that are buffered on the next volume.
 * Position backward 2 blocks.
 */
settp.ffp_nbs_p = FP_TPOS_BACK;
settp.ffp_nb = 2;
settp.ffp_nvs_p = 0;
settp.ffp_nv = 0;
settp.ffp_vi = 0;
if (ffpos(ffd, FP_SETTP, &settp, 0, &stat) < 0)
        ERREXIT("GETTP failed ",stat.sw_error);
/*
 * Read 2 blocks from tape + buffered blocks and store them
 * in a temporary file that is memory resident.
 */
if ((fftmp = ffopens("tmpfile",O_RDWR | O_CREAT,0,0,&stat,
                     "mr.scr.novfl")) < 0)
        ERREXIT("Error opening temporary file ",
                     stat.sw_error);
```

```
       for (i = 0; i < bufcnt+2; i++) {
                if (ffread(ffd,buf,BUFSIZ) != BUFSIZ)
                        ERREXIT("ffread failed ",errno);
                if (ffwrite(fftmp,buf,BUFSIZ) != BUFSIZ)
                        ERREXIT("ffwrite failed ",errno);
}
        /*
         * Position back 2 blocks.
         */
        settp.ffp_nbs_p = FP_TPOS_BACK;
        settp.ffp_nb = 2;
        settp.ffp_nvs_p = 0;
        settp.ffp_nv = 0;
        settp.ffp_vi = 0;

if (ffpos(ffd, FP_SETTP, &settp, 0, &stat) < 0)
                ERREXIT("SETTP failed ",stat.sw_error);
        for ( i = 0; i < 2; i ++) {     /* write 2 tape marks */
                if (ffweof(ffd)  < 0)
                        ERREXIT("ffweof failed ",errno);
        }
        /*
         * Close this volume and mount the next one in volume  identifier list
         */
        if (fffcntl(ffd, FC_CLOSEV, 0, 0, &stat) < 0)
                ERREXIT("Closev failed ",stat.sw_error);
        /*
         * End special processing.
         */
        if (fffcntl(ffd, FC_ENDSP, 0, 0, &stat) < 0)
                ERREXIT("Endsp failed ",stat.sw_error);
        /*
         * Disable special processing.
         */
        if (fffcntl(ffd, FC_SETSP, 0, 0, &stat) < 0)
                ERREXIT("Setsp failed ",stat.sw_error);
        /*
         * Write the data saved at eov. First rewind the temporary
         * file.
         */
```

```
       if (ffseek(fftmp, 0, 0) < 0)
                ERREXIT("Rewind of temporary file failed ",errno);
        for (i = 0; i < bufcnt+2; i++) {
                if (ffread(fftmp,buf,BUFSIZ) != BUFSIZ)
                        ERREXIT("Ffread failed ",errno);
                if (ffwrite(ffd,buf,BUFSIZ) != BUFSIZ)
                        ERREXIT("Ffwrite failed ",errno);
        }
        /*
         * Write 5 more blocks of data.
         */
        for (i = 0; i < 5; i++) {
                if (ffwrite(ffd,buf,BUFSIZ) != BUFSIZ)
                        ERREXIT("Ffwrite failed ",errno);

}
        /*
         * Close the tape file.
         */
        ffclose(ffd);
}
```

Figure 42. Using C library routines for EOV processing

5.2 System call I/O

Tape I/O at the system call level requires you to work with many details. You have a choice of synchronous or asynchronous I/O, and buffered or unbuffered I/O. You need to be concerned with buffer addresses, block size, number of bytes, and exception conditions. You need to know about specific hardware requirements of different Cray Research systems.

5.2.1 Cray Research systems

This section briefly describes system call level I/O concerns, and then describes in detail transparent I/O.

IBM compatible tape devices support blocked I/O. ER90 tape devices support blocked I/O and byte stream I/O.

For synchronous read and write requests, you must specify the buffer address and the number of bytes to read or write.

The block size for read and write operations restriction is based on a field size of 48 bits for IBM compatible devices. The CRAY J90 series have a maximum block size of 128 Kbytes except for the Small Computer System Interface (SCSI) I/O processor (IOP), which has maximum block size of 64 Kbytes minus 1 byte.

When a tape is read, the block size must be larger than or equal to the largest block size on the tape. The block size is specified with the `-b` option on the `tpmnt`(1) command or from the header label.

For ER90 devices, a *blocked file section* type consists of blocks of the size specified by the `-b` option of the `tpmnt`(1)command. Blocks within the file section, excluding the last block, must be the same length. The block size must be in the range of 80 through 1,199,832 bytes in 8-byte increments.

ER90 file sections within a tape can have different block lengths. You can change the block length for a file section from the value specified with the `tpmnt`(1) command by using the `TPC_SDBSZ ioctl`(2) request. The argument to the `ioctl`(2) request is the new block length, which cannot exceed the value specified with the `-b` option on the `tpmnt`(1) command. The block length can be changed only when the tape is positioned at the beginning of a file section.

You can use transparent I/O requests for reading and writing tape files. When you use transparent I/O, you do not need to be concerned with block size. Your program treats the data as a stream of bytes. In addition, transparent I/O allows you to specify either buffered or unbuffered I/O.

For ER90 devices, a *byte stream file* type is composed of blocks that are 1 byte in length. The ER90 device cannot access data that begins at an odd-byte memory address, therefore, byte stream data must be input and output to the device in even increments.

When using asynchronous I/O for transparent I/O or multilist I/O, you must acknowledge any exceptional conditions returned by the `reada`(2) and `writea`(2) system calls. When an exceptional condition occurs, the tape driver removes your I/O requests from the queue. When the driver receives additional I/O requests, it cannot determine if the requests were issued before or after an exceptional condition was returned; erroneous results may be generated. For example, an error status is returned in the `sw_error` field of the `iosw` structure for the `reada`(2) or `writea`(2) system call.

You may receive one of these exceptional conditions if you perform one of the following actions:

- You use asynchronous multilist I/O while processing tape marks.

  You must send an acknowledgment after each user tape mark is read.

- You use asynchronous multilist I/O while processing user end-of-volume.

  You must send an acknowledgment after receiving the ENOSPC status from the reada(2) or writea(2) system calls.

If you receive one of these exceptional conditions, but are able to continue processing, you must acknowledge receiving the condition by issuing a TPC_ACKERR ioctl(2) call as shown in the following example:

```
ioctl(fd, TPC_ACKERR, 0);
```

The fd option specifies the file descriptor.

All I/O requests received by the driver in the time between returning the exceptional condition and receiving the ioctl(2) acknowledgment are terminated with the error code ETPDACKERR. After the tape driver receives the acknowledgment, all I/O requests are processed normally.

5.2.2 Transparent I/O

If you are using transparent I/O, data is treated as a stream of bytes. To specify transparent I/O, open the tape file and issue read(2) or write(2) requests. If you issue a read(2) or write(2) request without specifying transparent I/O, the I/O is transparent by default. Transparent I/O can be either buffered or unbuffered.

5.2.2.1 Transparent buffered I/O

If transparent buffered I/O is requested, user data is temporarily stored in a system buffer. Transparent buffered I/O is the default I/O request type (do not include the -U option on the tpmnt(1) command).

To read a tape file with transparent buffered I/O, use the read(2) system call. The tape driver reads data blocks into a system buffer before copying data into a user buffer. The user may read any number of bytes. The tape driver copies the same number of bytes from the system buffer to the user buffer.

The following example shows you how to read 100 bytes, followed by another request to read the next 3 bytes from a tape file that has a maximum block size of 10,000 bytes, using transparent buffered I/O. This example can be used on all IOS systems.

1. Specify the block size as 10,000 bytes in the tpmnt(1) command:

   ```
   tpmnt -b 10000 -v SCRSL -f FILE
   ```

2. Specify the `open`(2) and `read`(2) statements in your C program:

```
filedes = open("file",O_RDONLY);
i = read(filedes, buf, 100);  /* read 100 bytes */
i = read(filedes, buf, 3);    /* read 3 bytes */
    . . .
```

To write a tape file with transparent buffered I/O, use the `write`(2) system call. The number of bytes requested to be written are copied into the system buffer. For IBM compatible devices, when the number of bytes of data accumulated in the system buffer is equal to the block size specified by the `-b` option of the `tpmnt`(1) command, the block of data is written to tape. For ER90 devices, when the buffer becomes full, the buffer is written to tape.

The following example shows you how to write a tape file that has a maximum block size of 10,000 bytes, using transparent buffered I/O. This example can be used on all IOS systems.

1. Specify the block size as 10,000 bytes in the `tpmnt`(1) command:

```
tpmnt -b 10000 -f FILE -v SCRSL -n
```

2. Specify the `write` statement in your C program:

```
filedes = open("FILE", O_RDWR);
write(filedes, buf, 20000);/*write 20000 bytes*/
        /* 2 blocks will be written to tape */
write(filedes, buf, 20);  /* write 20 bytes */
```

5.2.2.2 Transparent unbuffered I/O

To request unbuffered I/O, specify the `-U` option on the `tpmnt`(1) command. No system buffer will be used for user I/O. All I/O operations are done to and from your I/O buffer.

For ER90 byte stream requests, the byte count must be specified in even increments (excluding the last I/O) and be less than or equal to the device request limit, `CE_MAX_BLOCKS`.

For ER90 blocked requests, the byte count for reads must be greater than the maximum block size. In addition, each read transfers one block. For writes, the byte count must be a multiple of the block size, excluding the last I/O request.

To read a tape file with transparent unbuffered I/O, use the `read`(2) system call. A `read`(2) request transfers a tape block into your I/O buffer. For IBM compatible devices and ER90 blocked I/O requests, the number of bytes

specified in the `read`(2) request must be larger than or equal to the maximum block size specified by the `-b` option on the `tpmnt`(1) command, and it must be a multiple of 4096 bytes. When a `read`(2) completes with no error, a tape block is transferred into your I/O buffer, and the specified number of bytes is returned.

Figure 43 shows you how to read a tape file to an IBM compatible device using transparent unbuffered I/O:

1. Specify the block size as 10,000 bytes in the `tpmnt`(1) command:

```
rsv
tpmnt -v 123456 -l sl -P x -b 10000 -U -g CART
```

2. Specify the `read`(2) statement in your C program:

```
#include <fcntl.h>
main()
{
   char   buf[4096*3]; /* need 3 x 4096 bytes to hold 10000 bytes */
   int    fd;
   int    bytes;

   fd = open("x", O_RDONLY);
   bytes = read(fd, buf, 4096*3);
                            /* must request multiple of 4096 bytes */
}
```

Figure 43. Reading from an IBM compatible device (unbuffered I/O)

Figure 44 shows you how to read a tape file from an ER90 device using transparent unbuffered blocked I/O:

1. Specify the block size as 10,000 bytes in the `tpmnt`(1) command:

```
rsv CART
tpmnt -v 123456 -l sl -P x -b 10000 -B -U -g CART
```

2. Specify the `read`(2) statement in your C program:

```
#include <fcntl.h>
main()
{
   char   buf[4096*3];     /* 3 x 4096 bytes needed to hole 10000 bytes */
   int    fd;
   int    bytes;

   fd = open("x", O_RDONLY);
   bytes = read(fd, buf, 4096*3);}
```

Figure 44. Reading from an ER90 device (unbuffered blocked I/O)

Note: If a tape is accessed as blocked I/O as in the previous example, but is actually a byte stream file, `4096*3` bytes will be returned. An error will not be returned on the I/O request, even though the actual file type differs from the requested type.

Figure 45 shows you how to read a tape file from an ER90 device using transparent unbuffered byte stream I/O:

1. No block size is to be specified in the `tpmnt`(1) command:

   ```
   rsv ER90
   tpmnt -v 123456 -l sl -P x -U -g ER90
   ```

2. Specify the `read`(2) statement in your C program:

```
#include <fcntl.h>
 main()
 {
   char   buf[10000];
   int    fd;
   int    bytes;

   fd = open("x", O_RDONLY);
   bytes = read(fd, buf, 10000);}
```

Figure 45. Reading from an ER90 device (unbuffered byte stream I/O)

Note: If a tape is accessed as byte stream as in the previous example, but is actually a blocked tape, an error will be returned on the I/O request as the byte count is not a multiple of 4096 bytes.

To write a tape file with transparent unbuffered I/O, use the `write`(2) system call. For IBM compatible devices and ER90 blocked I/O, each `write`(2) request results in a block written from your user buffer to tape. When the `write`(2) returns with no error, the data in the user buffer is written to tape as a block. For ER90, blocked I/O requests must match the size specified with the `-b` option on the `tpmnt`(1) command. Each ER90 byte stream request writes the number of bytes requested.

Figure 46 shows you how to write a tape file to an IBM compatible device using transparent unbuffered I/O:

1. Specify the block size as 10,000 bytes in the `tpmnt`(1) command:

```
rsv CART 1
tpmnt -v ISCSL -l sl -P x -b 10000 -U -n -g CART
```

2. Specify the `write`(2) statement in your C program:

```
#include <fcntl.h>
main()
{
     char    buf[10000];  /* write buffer */
     int     fd;
     int     bytes;

     fd = open("x", O_WRONLY);
     bytes = write(fd, buf, 10000);  /* 10000-byte block */
     bytes = write(fd, buf, 500);    /* 500-byte block */
}
```

Figure 46. Writing to an IBM compatible device (unbuffered I/O)

Figure 47 shows you how to write a tape file to an ER90 device using transparent unbuffered byte stream I/O:

1. Specify the block size as 10,000 bytes in the `tpmnt`(1) command:

```
rsv ER90
tpmnt -v 123456 -l sl -P x -B -U -n -g ER90
```

2. Specify the `write`(2) statement in your C program:

```
#include <fcntl.h>
main()
{
     char    buf[10000];  /* write buffer */
     int     fd;
     int     bytes;

     fd = open("x", O_WRONLY);
     bytes = write(fd, buf, 10000);  /* 10000-byte block */
}
```

Figure 47. Writing to an ER90 device (unbuffered byte stream I/O)

5.3 Tape information requests

A C program can obtain tape subsystem information using system calls or tape daemon requests. This section discusses obtaining tape subsystem information from the tape information table, a tape daemon request, and several `ioctl`(2) requests.

5.3.1 Tape information table

The tape information table holds information about the tape system and is available to tape users. It is initialized by the tape driver when the system is started. When the tape daemon starts, it updates the table with information from its startup file.

The tape information table is defined in the `tapetab.h` file and included in a program by using the following preprocessor statement:

```
#include        <sys/tapetab.h>
```

The tape information table is defined so that it is not necessary to recompile if new fields are added to it in the future. It consists of a header with fixed length fields, followed by a variable length section. Figure 48 shows the format of the header:

```
typedef struct tapetab_struct {
     word  tape_tabsize;         /* size of table in bytes */
     word  tape_hdrsize;         /* size of tapetab header in bytes */
     word  tape_maxsize;         /* max size allocated to hold tapetab */
     word  tape_ios_model;       /* model E ios */
     word  tape_flag;            /* flags indicating various status */
     word  tape_dev_major;       /* major device number of tape devices */
     word  tape_dev_driver;      /* tape device driver name */
     word  tape_file_major;      /* user tape files major device number */
     word  tape_file_driver;     /* tape file driver name */
     word  tape_max_dev;         /* maximum number of tape devices */
     word  tape_conf_up;         /* maximum number of devices configured up */
     word  tape_max_per_dev;     /* max bytes for buffers per device */
     word  tape_max_bufs;        /* max buffers per device */
     word  tape_bmx_max_cmdlist; /* max cmds in a bmx cmdlist request */
}    tapetab;
```

Figure 48. Tape information table header

New fixed length fields may be added at the end of the header section. Offsets of variable fields are included in the fixed length fields.

Variable length fields follow the header with offsets defined in the header. Offsets are measured in words from the beginning of the table. These fields contain data, such as names. Fields of character strings must be null terminated. Variable length fields always start on word boundaries.

There are two types of variable length fields: single-item fields and a list of fields. Single-item fields, such as the daemon request pipe name, require a word in the header to hold its offset. A list of fields consists of a variable length list of offsets pointing to the corresponding field and requires two words in the header to hold the number of items in the list and the offset of the list.

The example, shown in Figure 49, accesses the tape information table, extracting the maximum number of tape drives:

```
#include   <sys/table.h>
#include   <stdio.h>
#include   <sys/types.h>
#include   <sys/tapetab.h>

main()
{
     word    tabsize;          /* size of table */
     tapetab_struct  *tblp;    /* pointer to tape information table */

     /*
      * get the size of tapetab.
      */

     if (tabread(TAPETAB, (char *)&tabsize, sizeof(word), 0)) {
       perror("can't read TAPETAB size");
       exit(1);
     }

     tblp = (tapetab_struct *)calloc(tabsize, 1);

     if (tabread(TAPETAB, (char *)tblp, tabsize)) {
       perror("can't read TAPETAB table");
       exit(1);
     }

     printf("max buffer size = %d bytes\n", tblp->tape_max_per_dev);
}
```

Figure 49. Using the tape information table

The `tabinfo`(2) and `tabread`(2) (see `tabinfo`(2)) system calls let you read a system table without reading `/dev/kmem`. The `tabinfo`(2) call describes table characteristics: location, header length, number of entries, and size of entry. Using the information returned by `tabinfo`(2), you can create a user buffer into which `tabread`(2) will read all or part of a table.

5.3.2 Tape daemon requests

The tape daemon request, called `TR_INFO`, lets you perform a tape status inquiry from within a C program. You must perform the following steps to send a tape daemon request and receive a reply.

1. Determine the request pipe name.

   The request pipe is automatically created for you when you issue the `rsv`(1)) command. Also, it is automatically deleted when you release all of your reservations by using the `rls`(1) command. The request pipe name must be the absolute path name, not just the file name portion. The directory of the request pipe is determined through the `#define USER_DIR` directive in file `tapedef.h`, which is set up at installation time to be the environment variable. The default is your environment variable `$TMPDIR`. For the file name portion, the `#define U_REQPIPE` directive in file `/usr/include/tapedef.h` defines the first part of the file name, which is appended by the job ID of your job. The default is `TAPE_REQ_`.

   The `#define MAXPATH` directive in file `tapedef.h` defines the longest path name minus one that the requests may use. If any path name is larger than `MAXPATH-1`, you must have the value of `MAXPATH` increased by your system administrator.

2. Build a reply pipe by using the `mknod`(2) system call. Open a pipe with an `open`(2) system call, keeping the pipe open until a reply returns.

   You can either build a reply pipe for each request and delete it after a reply has returned, or build a single reply pipe, using it for all of your requests. Regardless of the option you use, it is important to keep the reply pipe open until all replies have returned.

3. Place the reply pipe name in the request header. You must supply the absolute path name of the reply pipe.

4. Write the request into the pipe.

   Use the `write`(2) system call to write your request into the request pipe.

5. Read the reply header from the reply pipe.

   For each request submitted, the tape daemon sends a reply. Depending on the request you send, the reply may be just the reply header, or the reply header along with its data. To determine whether data has been returned, read the reply header from the reply pipe; if the size of the reply is larger than the reply pipe header, read in the rest of the reply.

You may use the echo field in the request and reply headers to help keep track of requests. The system copies what you input to the echo field of the request and reply headers.

Figure 50 shows a `TR_INFO` tape daemon request:

```
#include        <fcntl.h>
#include        <stdio.h>
#include        <tapedef.h>
#include        <tapereq.h>
#include        <sys/types.h>
#include        <sys/stat.h>
#include        <sys/jtab.h>

extern  char    *calloc();
extern  char    *getenv();

main()

{
    char    *dirptr;
      char    *req_pipe_name;             /* request pipe file name */
      char    *rep_pipe_name;             /* reply pipe file name */

      struct  jtab    jobtab;             /* structure for job table info */
      struct  stat    status;             /* structure to stat tape file */

      int     req_fd;                     /* request pipe file descriptor */
      int     rep_fd;                     /* reply pipe file descriptor */
      int     tape_fd;                    /* tape file file descriptor */

       struct  trinfo  info_req;           /* tape info structure */
       struct  trinfor info_rep;           /* tape info reply structure */
       struct  rephdr  *rh;                /* reply header */

       int     c;
       int     size;                       /* total size of reply */

/*
 *      Check the status of the tape file (this assumes you have
 *      performed a tpmnt with -P or -p to a file "tapefile")
 *      and open the tape file
 */
  c = stat("tapefile",&status);
    if (c < 0) {
        perror("Stat failed for tapefile");
        exit(1);
 }
```

```
   tape_fd = open("tapefile", O_RDWR);

   if (tape_fd < 0) {
        perror("Unable to open tapefile");
        exit(1);
   }

/*
 *      Make the named reply pipe and open it
 *      Use tempnam() to get a unique temporary file name
 */

    rep_pipe_name = tempnam(NULL, NULL);
    c = mknod(rep_pipe_name, 010700);
    if (c < 0) {
        perror("Unable to mknod reply pipe");
        close(req_fd);
        close(tape_fd);
        exit(1);
   }

    rep_fd = open(rep_pipe_name, O_RDWR);
    if (rep_fd < 0) {
        perror("Unable to open reply pipe");
        close(req_fd);
        close(tape_fd);
        exit(1);
   }
/*
 *      Construct request pipe file name and open it
 */
  dirptr = calloc(1, MAXPATH);
    req_pipe_name = calloc(1, MAXPATH);

    dirptr = getenv(USER_DIR);
    c = getjtab(&jobtab);

    sprintf(req_pipe_name, "%s/%s%d",dirptr, U_REQPIPE, jobtab.j_jid);
    req_fd = open(req_pipe_name, O_WRONLY);
```

```
if (req_fd < 0) {
         perror("Unable to open request pipe");
         close(tape_fd);
         exit(1);
   }

  info_req.rh.size = sizeof(struct trinfo);
  info_req.rh.code = TR_INFO;
  info_req.rh.jid = jobtab.j_jid;
  info_req.st_dev = status.st_dev;
  info_req.st_ino = status.st_ino;
  strcpy(&(info_req.rh.rpn),rep_pipe_name);

  c = write(req_fd, &info_req, info_req.rh.size);
  if (c < 0) {
        perror("Unable to write to daemon's request pipe");
        close(req_fd);
        close(rep_fd);
        unlink(rep_pipe_name);
        close(tape_fd);
        exit(1);
  }

  close(req_fd);
  req_fd = 0;
/*
 *      Now read the reply back from the tape daemon from
 *      the reply pipe
 */
  rh = (struct rephdr *)calloc(1, sizeof(struct rephdr));

  c = read(rep_fd, (char *)rh, sizeof(struct rephdr));
  if (c < 0) {
        perror("Read of reply pipe failed");
        close(rep_fd);
        unlink(rep_pipe_name);
        close(tape_fd);
        exit(1);
  }
```

```
 size = rh->size;
  c = read(rep_fd, &info_rep, size);

  if (c < 0) {
        perror("Read of trinfor failed");
        close(rep_fd);
        unlink(rep_pipe_name);
        close(tape_fd);
        exit(1);
  }

/*
 *      Program can go on to print out selected fields of the tsdata
 *      structure returned, or use them for another purpose.
 */

   printf("ts_fcn (last function) = %o\n", info_rep.tsdata.ts_fcn);
   printf("ts_dst (device status) = %o\n", info_rep.tsdata.ts_dst);

/*
 *      Close remaining open files and clean up.
 */

   close(rep_fd);
   unlink(rep_pipe_name);
   close(tape_fd);
   exit(0);}
```

Figure 50. Using the TR_INFO request

Figure 51 shows the information that is returned from the #define TR_INFO directive:

```
struct tsdata {
  /*
   * Device status information
   */
int  ts_ord;                        /* Device ordinal */
int  ts_fcn;                        /* Last device function */
int  ts_dst;                        /* Last device status */
int  ts_dtr;                        /* Data transfer count */
int  ts_bmblk;                      /* Buffer memory block count */
int  ts_bmsec;                      /* Buffer memory sector count */
int  ts_pbmcnt;                     /* Partial block bytes in buffer memory */
int  ts_orsc;                       /* Outstanding sector count */
int  ts_orbc;                       /* Outstanding block count */
int  ts_bnum;                       /* Block number: Block number */
                                    /* relative to tape mark */
int  ts_utmnum;                     /* User Tape Mark number: */
                                    /* This only includes */
                                    /* tape marks embedded */
                                    /* in the user's data */
int  ts_tmdir;                      /* Direction from tape mark */
                                    /*    0 :   after tape mark */
                                    /*    1 : before tape mark */
/*
 * Tape file information
 */

char  ts_path[MAXPATH];             /* Path name */
char  ts_dgn[16];                   /* Device group name */
char  ts_dvn[16];                   /* Device name */
int   ts_year;                      /* Today's year */
int   ts_day;                       /* Today's day */
char  ts_fid[48];                   /* File id */
char  ts_rf[8];                     /* Record format */
int   ts_den;                       /* Density: */
                                    /*    1: 1600 bpi */
                                    /*    2: 6250 bpi */
int   ts_mbs;                       /* Max block size */
int   ts_rl;                        /* Record length */
int   ts_fst;                       /* File status: */
                                    /*    1  : new   */
                                    /*    2  : old   */
                                    /*    3  : append */
```

```
int   ts_lb;                    /* Label type: */
                                /* 1 : no label */
                                /* 2 : ANSII label */
                                /* 3 : IBM label  */
                                /* 4 : bypass label */
                                /* 5 : single tape mark label */
int  ts_fsec;                   /* File section number */
int  ts_fseq;                   /* File sequence number */
int  ts_ffseq;                  /* Fseq of 1st file on tape */
int  ts_ring;                   /* Write ring status:*/
                                /*    0 : ring out */
                                /*    1: ring in   */
int  ts_xyear;                  /* Expiration year */
int   ts_xday;                  /* Expiration day */
int  ts_first;                  /* First vsn of file */
char  ts_v1[80];                /* Vol1 label */
char  ts_h1[80];                /* Hdr1 label */
char  ts_h2[80];                /* Hdr2 label */
int   ts_numvsn;                /* Number of vsn */
int  ts_vsnoff;                 /* Offset to vsn list */
                                /* from beginning of */
                                /* struct tsdata */
int   ts_cvsn;                  /* Current vsn index */
int  ts_eov;                    /* User eov selected: */
                                /*    0 : eov processing off */
                                /*    1 : eov processing on  */
int  ts_eovproc;                /* If user is currently in */
                                /* special user EOV processing,*/
                                /* field is set to 1. Otherwise*/
                                /* field is 0. ts_eov would be 1*/
int  ts_urwtm;                  /* User read/write tape mark */
                                /*    0 : not requested   */
                                /*    1 : requested by -T */
                                /*    option of tpmnt command */
char ts_ba[8];                  /* block attribute */
int  ts_blank4;                 /* Unused */
int  ts_blank5;                 /* Unused */
```

```
/* Following the tsdata structure is the vsn list. It is
 * of variable length. The tsdata.ts_numvsn field is the number
 * of vsns in the list. The tsdata.ts_vsnoff field is the offset
 * (in bytes) to the beginning of the vsn list from the beginning of
 * the tsdata structure. The vsns are of the form char[8]. */
}
```

Figure 51. TR_INFO information

5.3.3 ioctl(2) requests

The ioctl(2) system call requests of TPC_EXTSTS and TPC_RDLOG ioctl(2) let you request information about the tape subsystem. The TPC_EXTSTS request lets you obtain information on ER90 devices and the TPC_RDLOG request can be used to obtain information on ER90 and IBM compatible devices.

5.3.3.1 ER90 TPC_EXTSTS request

To obtain the extended status of an ER90 device, use the TPC_EXTSTS ioctl request. The extended status consists of the following responses to device commands: report addressee status, attribute, operating mode, and report position.

The report addressee status response gives the state of the ER90 device (ready/not ready or online/offline), a description of the mounted volume, and the ER90 detailed status.

The attribute response returns the operational characteristics of the ER90 device (for example, the data block size, burst size, early end-of-media warning (EEW), location, and so on).

The operating mode response describes those attributes that were temporarily defined for the time the tape was positioned within the current partition.

The report position response contains the current absolute track address, the remaining partition capacity, and other tape location information (for example, at beginning-of-tape, past the EEW location, at a system zone, and so on).

Refer to the *ER90 Interface Control Document* provided by E-Systems, Inc., for a complete description of the command responses.

The extended status is obtained by issuing an ioctl(2) system call with a request code of TPC_EXTSTS, to either the tape path or to file TPDDEM_REQ. The tape path is the path specified on the tpmnt(1) command. TPDDEM_REQ is

a pseudo device used to issue requests to a device without users having to have the device assigned to them. If the request is issued to the pseudo device, the device name must be specified in the request. (`TPDDEM_REQ` is defined in the `tapedef.h` file.)

The argument of the `ioctl`(2) call must be a pointer to structure `ctl_extsts`. This structure is defined in Figure 52:

```
struct ctl_extsts   {
     int     device;
     char    *rep_addr;
     int     len_rep_addr;
     char    *attributes;
     int     len_attributes;
     char    *oper_mode;
     int     len_oper_mode;
     char    *report_pos;
     int     len_report_pos;
}
```

Figure 52. `ctl_extsts` structure

Set `rep_addr`, `attributes`, `oper_mode`, and `report_pos` to pointers to memory in which the response packets will be copied to receive responses to all of the commands. Set to `NULL` the memory pointers of the response packets that are not to receive only selected portions of the extended device status. Set the amount of memory allocated for the command in the `len_rep_addr`, `len_attributes`, `len_oper_mode`, or `len_report_pos` for each command requested. If the request is made to `TPDDEM_REQ`, `device` must be set to the device name. The length of each response packet is returned in the variables `len_rep_addr`, `len_attributes`, `len_oper_mode`, and `len_report_pos`.

The following restrictions apply to the ER90 `TPC_EXTSTS` request:

- The format or asynchronous I/O requests cannot be outstanding.
- Only the super user can issue this request through a pseudo device.
- The device must be configured up.

Note: If the operating mode response is requested and a cassette is not loaded, the cassette is blank, or the logical position has not been established, an operating mode response will not be returned.

Issuing requests to a device through the pseudo device suspends the current device activity until the extended status has been obtained.

Figure 53 shows how to obtain the extended status of an ER90 device by issuing a TPC_EXTSTS request using the tape path:

```
/*    Get the extended device status.
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include <errno.h>
#include <sys/sysmacros.h>
#include <sys/tpdctl.h>
#include <sys/epack.h>
#include <sys/epacki.h>
main()
{
       struct  ctl_extsts  ctl;
       char    rep_addr [MAX_IPI3_RESP_B];
       char    attributes [MAX_IPI3_RESP_B];
       char    oper_mode [MAX_IPI3_RESP_B];
       char    report_pos [MAX_IPI3_RESP_B];
       extern  int  errno;
       int         fd;
       int         c;
       /*
        *   Open the tape device path
        */
       fd = open( "tape_path", O_RDWR );
       if ( fd < 0 ) {
          perror( "Unable to open the device path" );
          exit(errno);
       }
       ctl.rep_addr = rep_addr;
       ctl.len_rep_addr = MAX_IPI3_RESP_B;
       ctl.attributes = attributes;
       ctl.len_attributes = MAX_IPI3_RESP_B;
       ctl.oper_mode = oper_mode;
       ctl.len_oper_mode = MAX_IPI3_RESP_B;
       ctl.report_pos = report_pos;
       ctl.len_report_pos = MAX_IPI3_RESP_B;
       /*
```

```
*   Issue the request for the extended device status.
        */
       c = ioctl( fd, TPC_EXTSTS, &ctl );
       if ( c < 0 ) {
          perror( "ioctl TPC_EXTSTS" );
          exit(errno);
       }
}
```

Figure 53. Using the ER90 TPC_EXTSTS request (tape path)

Figure 54 shows how to obtain the extended status on an ER90 device by issuing a TPC_EXTSTS request using a pseudo device:

```
/*
 *    Get the current position and remaining partition capacity of the
 *    mounted volume.
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include <errno.h>
#include <sys/sysmacros.h>
#include <sys/tpdctl.h>
#include <sys/epack.h>
#include <sys/epacki.h>
main()
{
       struct  ctl_extsts  ctl;
       char   report_pos[MAX_IPI3_RESP_B];
       extern  int  errno;
       int         fd;
       int         c;
       /*
        *   Open the pseudo device
        */
       ctl.device = 0;
       strncpy((char *)&ctl.device, "devname",strlen("devname"));
       fd = open( TPDDEM_REQ, O_RDWR );
       if ( fd < 0 ) {
          perror( "Unable to open the device path" );
          exit(errno);
       }
       bzero( (char *)&ctl, sizeof(struct ctl_abspos));
       ctl.len_report_pos = MAX_IPI3_RESP_B;
       ctl.report_pos = report_pos;
       /*
        *   Issue the request for the extended device status.
        */
       c = ioctl( fd, TPC_EXTSTS, &ctl );
       if ( c < 0 ) {
          perror( "ioctl TPC_EXTSTS" );
          exit(errno);
       }
}
```

Figure 54. Using the ER90 TPC_EXTSTS request (pseudo device)

5.3.3.2 ER90 read of the buffer log using TPC_RDLOG

The ER90 error log can be obtained by issuing an `ioctl`(2) system call, with a request code of `TPC_RDLOG`, to either the tape path or to file `TPDDEM_REQ`. The tape path is the path specified on the `tpmnt`(1) command. `TPDDEM_REQ` is a pseudo device used to issue requests to a device without the users having to have the device assigned to them. If the request is issued to the pseudo device, the device name must be specified in the request. (`TPDDEM_REQ` is defined in the `tapedef.h` file.)

The argument of the `ioctl`(2) call must be a pointer to structure `ctl_rdlog`. This structure is defined in Figure 55:

```
struct ctl_rdlog   {
     int     device;
     char    *device_log;
     int     length;
}
```

Figure 55. `ctl_rdlog` structure

The `device_log` field must be set to a pointer to the memory in which the ER90 error log will be copied. `length` must be set to the amount of memory allocated for the device log. If the request is made to `TPDDEM_REQ`, `device` must be set to the device name. The length of the device log will be returned in `length`.

The following restrictions apply to the `TPC_RDLOG` request:

- The format or asynchronous I/O requests cannot be outstanding.
- Only the super user can issue this request through a pseudo device.
- The device must be configured up.

  Note: Issuing requests to a device through the pseudo device suspends the current device activity until the extended status has been obtained.

Figure 56 shows how to read the ER90 error log by issuing a `TPC_RDLOG` request:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include <sys/sysmacros.h>
#include <sys/tpdctl.h>
#include <sys/epack.h>
#include <sys/epacki.h>
#include <sys/er90_cmdpkt.h>

main()
{
       struct  ctl_rdlog  ctl;
       char    device_log[MAX_IPI3_RESP_B];
       int         fd;
       int         c;
       /*
        *   Open the tape device path
        */
       fd = open( "tape_path", O_RDWR );
/*
 *  or   ctl.device =0;
 *      strncpy((char *)&ctl.device, "devname", strlen("devname"));
 *      fd = open( TPDDEM_REQ, O_RDWR );
 */
       if ( fd < 0 ) {
         perror( "Unable to open the device path" );
         exit(1);
       }
       /*
        *   Issue the request for the ER90 Error Log.
        */
       ctl.length = MAX_IPI3_RESP_B;
       ctl.device_log = device_log;
       c = ioctl( fd, TPC_RDLOG, &ctl );
       if ( c < 0 ) {
         perror( "ioctl TPC_RDLOG" );
         exit(1);
       }
}
```

Figure 56. Using the ER90 TPC_RDLOG request

5.3.3.3 IBM compatible read of the buffer log using TPC_RDLOG

Note: The TPC_RDLOG request returns zeros on SCSI devices. It does not return an error code.

The IBM compatible buffer log can be obtained by issuing an `ioctl`(2) system call using the TPC_RDLOG request, as shown in Figure 57:

```
int   buflog[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };
int   i;

if ( ioctl( fd, TPC_RDLOG, buflog ) < 0) {
        perror( "Error reading buf log" );
        exit( 1 );
}

for ( i = 0 ; i < 8 ; i++ ) {
        printf( "0x%16.16x\n", buflog[i] );
}
```

Figure 57. Using the TPC_RDLOG request (IBM compatible)

The TPC_RDLOG request may be made after any read, write, or position request. It may be used to calculate compression ratio on the tape.

It is necessary to examine the sense information returned by the read buffer log request to determine compression ratios and distance from the end of the tape at any point while using the tape. The *IBM Hardware Reference Manual*, publication GA32-0127, provides detailed information on sense bytes and their formats.

Format 30 of sense bytes 32 through 43, provide counts for bytes processed by the channel and device. Channel counts reflect the number of bytes requested for the I/O operation, while the device counts reflect the number of bytes actually read or written by the device. The difference between the counts is the compression ratio achieved for the I/O operation.

Format 30 of sense byte 31 gives information about the length of the tape. It is possible to use this information in combination with the compression ratio information to determine approximately how much tape is used or remaining.

5.4 Tape positioning requests

C programmers use the `ffpos`(3) and `ffseek`(3) routines to implement tape positioning. For more information on positioning, see Section 5.1, page 81. Fortran programmers can position by using blocks and volumes as shown on Section 4.1.3, page 44 and Section 4.1.4, page 46.

5.5 End-of-volume requests

For information about user EOV processing from a Fortran program, see Section 4.1.7, page 65. Usually, volume switching is handled by the tape subsystem and is transparent to you. However, when user EOV processing is requested, you gain control at the end-of-tape and your program may perform special processing.

5.6 Tape control requests

You can use `ioctl`(2) system calls to control some characteristics of tapes. For ER90 devices, you can control the data block size and synchronize your program with the tape.

5.6.1 ER90 set data block size request

The data block size of a file section can be set by issuing an `ioctl`(2) system call `TPC_SDBSZ` request to a tape path. The tape path is the path specified on the `tpmnt`(1) command.

The argument of the `ioctl`(2) call is the data block size. The data block size must be in the range of 80 to 1,199,832 byes, and must be a multiple of eight.

The following restrictions apply to the `TPC_SDBSZ` request:

- The tape mount, format, or asynchronous request cannot be outstanding.
- The tape must be positioned at the beginning of a file section.
- The data block size cannot exceed the maximum block size specified by the `-b` option of the `tpmnt`(1) command.

Figure 58 shows how to set the data block size on an ER90 device:

```
#include <sys/types.h>
#include <sys/fcntl.h>
#include <errno.h>
#include <sys/tpdctl.h>

main()
{
       extern  int  errno;
       int     fd;
       int     c;
       /*
        *   Open the tape device path
        */
       fd = open( "tape_path", O_RDWR );
       if ( fd < 0 ) {
          perror( "Unable to open the tape path" );
          exit(errno);
       }
       /*
        *   Issue the request to set the DataBlock size.
        */
       c = ioctl( fd, TPC_SDBSZ, 32768 );
       if ( c < 0 ) {
          perror( "TPC_SDBSZ error" );
          exit(errno);
       }
}
```

Figure 58. Setting data block size

5.6.2 ER90 synchronize request

Synchronizing your program with the tape is accomplished by issuing an `ioctl`(2) system call `TPC_DMN_REQ` request to a tape path. The tape path is the path specified on the `tpmnt`(1) command.

The argument of the `ioctl`(2) call must be a pointer to structure `dmn_comm`. This structure is defined in Figure 59:

```
struct  dmn_comm {
     int     POS_REQ;
     int     POS_ABSADDR;
     int     POS_COUNT;
     int     POS_REP;
}
```

Figure 59. `dmn_comm` structure (synchronizing request)

There cannot be any outstanding asynchronous I/O requests for the `TPC_DMN_REQ` synchronize request to complete.

Note: If the previous request was a write request, data in the driver's buffer will be flushed to the tape. A synchronize request is then issued to the device, flushing the contents of the device's buffer to the tape. If the data in the system buffer is not a multiple of the data block size, a short block is output to the tape.

If the previous request was a read request and data is in the driver's buffer, the driver will backspace over the read ahead blocks. If there is a partial block in the buffer, the tape position is left after this block but the remainder of the block is deleted from the buffer.

Figure 60 shows how to synchronize your program with a tape on an ER90 device:

```
#include <sys/types.h>
#include <sys/fcntl.h>
#include <errno.h>
#include <tapereq>
#include <sys/tpdctl.h>

main()
{
       struct  dmn_comm  pos;
       extern  int  errno;
       int     fd;
       int     c;
       /*
        *   Open the tape device path
        */
       fd = open( "tape_path", O_RDWR );
       if ( fd < 0 ) {
          perror( "Unable to open the tape path" );
          exit(errno);
       }
       pos.POS_REQ  = TR_SYNC;
       /*
        *   Issue the sync request
        */
       c = ioctl( fd, TPC_DMN_REQ, &pos );
       if ( c < 0 ) {
          perror( "TPC_DMN_REQ error" );
          exit(errno);
       }
       /*
        *   Get the reply
        */
       c = ioctl( fd, TPC_DMN_REP, &pos );
       if ( c < 0 ) {
          perror( "TPC_DMN_REP failed" );
          exit(errno);
```

```
        }
        if ( pos.POS_REP ) {
           printf( "SYNC  error = %d", pos.POS_REP );
           exit(1);
        }
}
```

Figure 60. Synchronizing your program with a tape

Using the Character-Special Tape Interface [6]

The character-special tape interface provides unstructured access to the tape hardware, similar to the traditional UNIX method of accessing tape devices. This interface is useful in performing specific tasks:

- System administrators can use the interface for routine tape manipulations such as copying. They can use standard UNIX commands and `ioctl`(2) requests to manage their tapes. The first section briefly describes this usage.
- Programmers can use the interface to develop file management applications. Section 6.2, page 122, on writing C applications, describes opening and closing files, managing I/O, and using the `ioctl`(2) requests.

6.1 Using character-special tapes

Character-special tape files are created by executing the `tpdaemon`(8) command. This command creates a file for each device defined in the tape configuration file (`/etc/config/text_tapeconfig`). These files reside in the `/dev/tape` directory.

Before terminating, the `tpdaemon`(8) command creates a detached process that is used to assist the tape driver. If tape devices will be accessed using only the character-special tape interface, this process may be terminated using the `tpdstop`(8) command. The tape daemon may be restarted as long as all character-special device files are closed.

The character-special tape interface and the tape daemon-assisted interface may operate concurrently. Devices for both interfaces are defined in the same configuration file and are defined identically; that is, the interface is not identified in the configuration file.

The system identifies the type of interface being used when the device is opened. The character-special tape interface is used if a device file residing in the `/dev/tape` directory is opened. Once opened, the device cannot be accessed by the tape daemon until it is closed.

If a device will be accessed by using the tape daemon-assisted interface, the device must be configured up by using the `tpconfig`(8) command. A device is not accessible to the character-special tape interface while configured up.

6.2 Writing C applications

This section provides information programmers need to write C applications using the character-special tape interface:

- Opening files
- Closing files
- Using I/O
- Using `ioctl`(2) requests

6.2.1 Opening files

A tape device file to be opened must reside in the `/dev/tape` directory, but it cannot be a diagnostic device file. The device file cannot be available to the tape daemon (that is, the device must be configured down or the tape daemon must be down) and cannot be open already.

Open processing assigns the device to the host from which the open request was issued. Opening an ER90 device file resets the device attributes to their default values, excluding the burst size, which is set to a value appropriate for the physical interface used. The first open of an ER90 device file, following a tape daemon start-up, also clears the device log and executes a diagnostic check.

Note: The ER90 format is not available on systems that run the UNICOS/mk operating system or that have GigaRing support.

6.2.2 Closing files

If data is being output before a tape device file is closed, the tape is terminated with two tape marks, and the tape is left-positioned between the tape marks. The tape marks are not output if the last user request is a tape mark write request.

If a rewind operation is requested with the `MTIOCATTR` `ioctl`(2) system call, the tape is rewound. If an unload operation is requested with the `MTIOCATTR` `ioctl`(2) system call, the tape is unloaded.

6.2.3 Using I/O

The character-special tape interface supports only unbuffered, transparent input and output (I/O).

ER90 devices support both byte stream and blocked file types. By default, byte stream files are created. The size of the I/O request is limited, by the device, to `CE_MAX_BLOCKS`.

ER90 blocked I/O can be performed by modifying the file type through the `MTIOCATTR` `ioctl`(2) system call. Blocked read requests transfer one block; write requests can transfer multiple blocks. For optimal performance, output requests should be a multiple of the data block size.

Although the block multiplexer I/O requests can be any size and ER90 requests are limited only by the device maximum, data is transferred to and from the IOP in words. The user's buffer must be a multiple of the Cray word size (64 bits).

If the I/O completes successfully, the number of bytes read or written is returned. If a tape mark is read, a byte count of 0 is returned and the tape is left-positioned after the tape mark.

If an error occurs on the I/O request, -1 is returned and `errno` is set to indicate the error. The number of bytes that did not get read or written can be obtained by using the `MTIOCGET` `ioctl`(2) system call.

If the I/O request is unsuccessful, `errno` is set to one of the following:

Error code	Description
`EFAULT`	The buffer argument points outside the allocated address space.
`EINTR`	The system call was interrupted.
`ENOSPC`	The end-of-tape (EOT) was detected.
`ETPDACKERR`	An error has not been acknowledged.
`ETPDBUFZ`	The byte count is less than the data block size.
`ETPD_MAX_IOREQT`	The byte count exceeds the device limit.

If an error occurs on an asynchronous I/O request, all queued I/O requests are terminated with `ETPDACKERR`. All subsequent I/O requests are also terminated with `ETPDACKERR` until the error is acknowledged with the `MTIOCACKERR` `ioctl`(2) system call.

6.2.4 Using `ioctl`(2) requests

The character-special tape interface supports four `ioctl`(2) requests:

Request	Description
MTIOCACKERR	Acknowledges an asynchronous I/O error
MTIOCATTR	Modifies the tape attributes
MTIOCGET	Returns the tape status
MTIOCTOP	Executes a tape operation

All `ioctl` requests require that there be no outstanding asynchronous I/O.

6.2.4.1 MTIOCACKERR call

The `MTIOCACKERR` `ioctl`(2) system call acknowledges an error condition. The argument to `ioctl` is `NULL`.

After an error condition is detected, all queued I/O requests and I/O requests received before an acknowledgment are terminated with `ETPDACKERR`. After `MTIOCACKERR` is received, I/O requests are processed normally.

6.2.4.2 MTIOCATTR call

The `MTIOCATTR` `ioctl`(2) system call modifies the attributes of the tape device file. The argument to this call is a pointer to the `mtattr` structure:

```
struct mtattr {
     uint  mt_attribute;
     uint  mt_blksiz;
}
```

`mt_attribute` is a flag constructed from the following list. The flags specify the attributes to modify. When the device is closed, the attributes are reset to the default values.

Flag	Description
MT_REPORT	Reports the current attribute settings.
MT_BYTESTREAM	Modifies the file type to byte stream. This flag is only valid for ER90 device files. It is a default.
MT_BLOCKED	Modifies the file type to blocked. The data block size is specified in `mt_blksiz`. This flag is only valid for ER90 device files.
MT_IGNORE_EOT	Ignores the EOT status.
MT_OBSERVE_EOT	Returns the EOT status. This is a default.

MT_CLOSE_UNLOAD	Unloads the tape when the device file is closed.
MT_NO_CLOSE_UNLOAD	Does not unload the tape when the device file is closed. This is a default.
MT_CLOSE_REWIND	Rewinds the tape when the device file is closed.
MT_NO_CLOSE_REWIND	Does not rewind the tape when the device file is closed. This is a default.
MT_READ_RAW	Transfers all data regardless of data errors. This flag is only valid for ER90 device files.
MT_READ_NORMAL	Transfers only valid data. This flag is only valid for ER90 device files. It is a default.
MT_COMPRESSION	Enables device data compression.
MT_NO_COMPRESSION	Disables device data compression.

If a blocked file is requested with the MT_BLOCKED flag, mt_blksiz specifies the size of the data blocks. For optimal performance, all blocks within the file section should be of size mt_blksiz. mt_blksiz must be a multiple of 8 bytes and must be in the range 80 to 1,119,832 bytes.

Flags MT_BLOCKED, MT_BYTESTREAM, MT_READ_RAW, and MT_READ_NORMAL are only valid for ER90 device files.

Flags MT_COMPRESSION and MT_NO_COMPRESSION are only valid for 3480, 3490, and 3490E devices. If neither attribute MT_COMPRESSION or MT_NO_COMPRESSION is specified, the devices default to the device default compaction mode. Data compression will also return to the device default after a tape unload.

6.2.4.3 MTIOCGET call

The MTIOCGET ioctl(2) system call returns the device status. The argument to this call is a pointer to the mtget structure:

```
struct mtget{
    short     mt_type;
    int       mt_dsreg;
    caddr_t   mt_erreg;
    int       mt_resid;
    int       mt_fileno;
    int       mt_blkno;
    short     mt_flags;
}
```

`mt_type` specifies one of the following tape device types:

Device type	Description
`MT_3803`	IBM 3803 type tape device
`MT_3480`	IBM 3480 cartridge device
`MT_3490`	IBM 3490 cartridge device
`MT_3490E`	IBM 3490E cartridge device
`MT_ER90`	ER90 tape device

`mt_dsreg` contains the device status. It is one of the following flags:

Flag	Description
`MT_ONL`	The device is online.
`MT_RDY`	The device is ready.
`MT_WPT`	The cassette loaded in the device is write protected.
`MT_EOT`	An end-of-tape (EOT) status was received on last device request (BMX); the tape is positioned past the early-end-of-media warning (EEW). (This flag is only for ER90 devices.)

`mt_resid` contains a residual count. If the last system call was an I/O request, it is the number of bytes that did not get read or written. If the last system call was an `ioctl`(2) system call performing a tape operation, it represents the number of tape operations that did not complete. If a request is interrupted, the accuracy of the residual count cannot be guaranteed.

`mt_erreg` is a pointer to a structure describing the response status of the last user request issued to the device. For block multiplexer devices, it is a pointer to the `bmxerec` structure, defined in `bmxerec.h`. For ER90 devices, it is a pointer to the `er90_erecord` structure, defined in the `er90_erec.h` file. If `mt_erreg` is `NULL`, the status is not returned.

`mt_flags` specifies one or more of the following response flags:

Flag	Description
`MT_VALID_FILENO`	Specifies that `mt_fileno` is valid

`MT_VALID_BLKNO` Specifies that `mt_blkno` is valid

If `mt_flags` is set to `MT_VALID_FILENO`, `mt_fileno` specifies the current file number. If `mt_flags` is set to `MT_VALID_BLKNO`, `mt_blkno` specifies the current block number. These fields are never valid for block multiplexer device files. They are valid for ER90 device files only if the logical position has been established.

6.2.4.4 `MTIOCTOP` call

The `MTIOCTOP` `ioctl`(2) system call performs a tape operation. The argument to the `MTIOCTOP` `ioctl`(2) system call is a pointer to the `mtop` structure:

```
struct mtop     {
     short      mt_op;
     int        mt_count;
     caddr_t    mt_arg;
     int        mt_size;
}
```

`mt_op` specifies the type of tape operation to execute. Valid `mt_op` codes are:

Operation code	Description
MTWEOF	Writes a tape mark
MTFSF	Spaces file forward
MTBSF	Spaces file backward
MTFSR	Spaces record forward
MTBSR	Spaces record backward
MTREW	Rewinds tape
MTOFFL	Unloads the tape volume
MTSYNC	Synchronizes the user and the tape device
MTGABS	Returns the absolute track address
MTPABS	Positions to an absolute track address
MTGPOS	Returns the current position
MTSEEK	Positions to a specific tape area
MTEXTS	Returns the extended status
MTFMT	Formats a tape volume
MTGFMT	Reports the cassette and volume format
MTRDLOG	Reads the device log
MTCLRLOG	Clears the device log
MTVERIFY	Verifies recorded tape data
MTTRACE	Verifies recorded tape data
MTMSG	Displays a message on a tape device

`mt_count` specifies the number of tape operations to execute. This variable is only valid for the `MTWEOF`, `MTFSF`, `MTBSF`, `MTFSR`, and `MTBSR` operations. For all other tape operations, the number of tape operations to execute defaults to 1.

`mt_arg` is a pointer to a buffer that provides information needed to complete the tape operation, or it is a pointer to a buffer into which the response is returned.

`mt_size` specifies the size of the buffer available for the response. The size of a tape response is returned in `mt_size`.

If the `ioctl`(2) request does not complete successfully, the number of tape operations that did not complete can be obtained by using `MTIOCGET`.

6.2.4.4.1 MTWEOF

MTWEOF records tape marks at the current position. mt_count specifies the number of tape marks to record.

6.2.4.4.2 MTFSF and MTBSF

MTFSF positions forward by tape marks. The tape position is left on the EOT side of the last tape mark positioned over. MTBSF positions backward by tape marks. The tape position is left on the BOT side of the last tape mark positioned over. The number of tape marks to position is specified in mt_count.

6.2.4.4.3 MTFSR and MTBSR

MTFSR positions forward by tape blocks or bytes. The tape position is left on the EOT side of the last block or byte positioned over. MTBSR positions backward by tape blocks or bytes. The tape position is left on the BOT side of the last tape block or byte positioned over. The number of blocks or bytes to position is specified in mt_count.

6.2.4.4.4 MTREW

For block multiplexer devices, MTREW rewinds the tape to the beginning-of-tape (BOT). For ER90 devices, MTREW positions the tape to the beginning of the current partition.

6.2.4.4.5 MTOFFL

MTOFFL ejects the tape volume from the tape device. If this request is issued to a 3480, 3490, or 3490E device, the device log is automatically cleared.

6.2.4.4.6 MTSYNC

MTSYNC synchronizes the user with the tape device. All data in the device buffer is flushed to tape.

6.2.4.4.7 MTRDLOG and MTCLRLOG

MTRDLOG reads the device log. mt_arg is a pointer to the buffer into which the device log is read or copied. mt_size specifies the size of the buffer. The buffer size must be at least 64 bytes for requests issued to block multiplexer

device files and 808 bytes for requests to ER90 device files. The size of the ER90 device log is returned in `mt_size`. This operation leaves the ER90 device log intact; it clears the block multiplexer device log.

`MTCLRLOG` clears the device log. This operation is only valid for ER90 device files. The `MTRDLOG` request must be used to clear a block multiplexer device log.

6.2.4.4.8 MTGABS and MTPABS

`MTGABS` returns the absolute track address. `MTPABS` positions to an absolute track address.

For block multiplexer device files, `MTGABS` returns the absolute address in the integer pointed to by `mt_arg`. `MTPABS` positions to the absolute address in the integer pointed to by `mt_arg`. `mt_size` must be at least 8 bytes for `MTGABS` requests and 4 bytes for position requests.

The absolute address is comprised of two 4-byte block identifiers as shown in Figure 61.

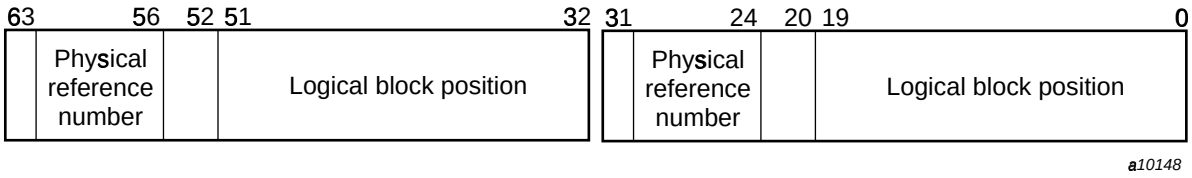

Figure 61. Block identifiers

Bits 32 through 63 identify the next block to be transferred between the host and the device. Bits 0 through 31 identify the next block to be transferred between the control unit buffer and the tape. The difference between the logical block position portion (bits 0 through 19 and 32 through 51) of the block identifiers is the amount of data in the device buffer. Only the first block ID (bits 32 through 63) is used on the `MTPABS` request.

For ER90 devices, `MTGABS` returns the absolute track address in the structure pointed to by `mt_arg`. `MTPABS` positions to the address in the structure pointed to by `mt_arg`. The structure is defined as follows:

```
struct tpc_abspos     {
     uint       tpc_valid_logdb       : 1,
                tpc_valid_absaddr     : 1,
```

```
                tpc_valid_partition   : 1,
                tpc_valid_filesec     : 1,
                tpc_valid_timecode    : 1,
                tpc_unused            : 11,
                tpc_logical_datablock : 48;
     uint       tpc_absolute_address  : 32,
                tpc_file_section      : 32;
     uint       tpc_partition_number  : 16,
                tpc_time_code         : 48;
}
```

`tpc_valid_logdb` is set to 1 if the `tpc_logical_datablock` variable is valid. `tpc_valid_absaddr` is set to 1 if the `tpc_absolute_address` variable is valid. `tpc_valid_partition` is set to 1 if the `tpc_partition_number` variable is valid. `tpc_valid_filesec` is set to 1 if the `tpc_file_section` variable is valid. `tpc_valid_timecode` is set to 1 if the `tpc_time_code` variable is valid.

`tpc_logical_datablock` specifies the data block number of the next block to be transferred between the host and the device. The block numbering begins with 0 at the beginning of a file section.

Absolute addresses are recorded on the longitudinal track of a tape volume when the volume is formatted. Each address corresponds to a physical block. `tpc_absolute_address` is the address identifying the physical block of the next data block to be transferred between the device buffer and the tape.

A file on an ER90 volume is a sequence of blocks terminated by a file mark. `tpc_file_section` specifies the file section number of the current block. The file section numbering begins with 1 at the beginning of a partition.

Partitions are logical volumes created on the tape when the tape is formatted. `tpc_partition_number` specifies the current partition number. If the tape has one partition spanning the length of the tape, the partition number will be 0. If the tape is multipartitioned, the partition numbers are offset by 0x100 and range from 0x100 to 0x4FF.

`tpc_time_code` specifies the time code. This field does not apply to the files created with the character-special tape interface, because this interface does not output data with time codes.

6.2.4.4.9 MTGPOS

MTGPOS returns the current tape position for ER90 device files. The current position is returned in the structure pointed to by mt_arg. The structure is defined as follows:

```
struct tpc_er90_pos   {
     uint       tpc_valid_logdb        : 1,
                tpc_valid_physblock    : 1,
                tpc_valid_absaddr      : 1,
                tpc_valid_index        : 1,
                tpc_valid_partition    : 1,
                tpc_valid_filesec      : 1,
                tpc_valid_phydb        : 1,
                tpc_valid_timecode     : 1,
                tpc_unused_0           : 8,
                tpc_logical_datablock  : 48;
    uint        tpc_physical_block     : 32,
                tpc_absolute_address   : 32;
     uint       tpc_index              : 16,
                tpc_partition_number   : 16,
                tpc_file_section       : 32;
     uint       tpc_physical_datablock : 48,
                tpc_time_code_a        : 16;
     uint       tpc_time_code_b        : 32,
                tpc_unused_1           : 32;
    uint        tpc_pos_bom            : 1,
                tpc_pos_emw            : 1,
                tpc_pos_rsvd_0         : 1,
                tpc_pos_eew            : 1,
                tpc_pos_rsvd_1         : 4,
                tpc_pos_bot            : 1,
                tpc_pos_eor            : 1,
                tpc_pos_eot            : 1,
                tpc_pos_sysz           : 1,
                tpc_pos_eom            : 1,
                tpc_pos_rsvd_2         : 3,
                tpc_sysz_number        : 8,
                tpc_reserved_0         : 6,
                tpc_valid_rem_part     : 1,
                tpc_valid_rem_dbframes : 1,
                tpc_rem_partition      : 32;
     uint       tpc_rem_doubleframes   : 32,
                tpc_reserved_1         : 32;
```

`tpc_valid_logdb` is set to 1 if the `tpc_logical_datablock` variable is valid. `tpc_valid_physblock` is set to 1 if the `tpc_physblock` variable is valid. `tpc_valid_absaddr` is set to 1 if the `tpc_absolute_address` variable is valid. `tpc_valid_index` is set to 1 if the `tpc_index` variable is valid. `tpc_valid_partition` is set to 1 if the `tpc_partition_number` variable is valid. `tpc_valid_filesec` is set to 1 if the `tpc_file_section` variable is valid. `tpc_valid_phydb` is set to 1 if the `tpc_physical_datablock` variable is valid. `tpc_valid_timecode` is set to 1 if the `tpc_time_code_a` and `tpc_time_code_b` are valid.

`tpc_logical_datablock` specifies the data block number of the next block to be transferred between the host and the device. The block numbering begins with 0 at the beginning of a file section.

A physical block is the smallest unit in which data can be recorded on tape. `tpc_physical_block` specifies the block number of the next physical block to be transferred from the ER90 device buffer to tape.

`tpc_physical_datablock` specifies the data block number of the next block to be transferred between the device buffer and the tape.

Absolute addresses are recorded on the longitudinal track of a tape volume when the volume is formatted. Each address corresponds to a physical block. `tpc_absolute_address` is the address identifying the physical block in which the current physical data block is located.

`tpc_index` specifies the index. ER90 devices do not support an index; `tpc_index` will not, therefore, contain a valid value for these devices.

Partitions are logical volumes created on the tape when the tape is formatted. `tpc_partition_number` specifies the partition number of the current position. If the tape has one partition spanning the length of the tape, the partition number is 0. If the tape is multipartitioned, the partition numbers are offset by 0x100 and range from 0x100 to 0x4FF.

A file section on an ER90 volume is a sequence of blocks terminated by a file mark. `tpc_file_section` specifies the file section number of the current block. The file section numbering begins with 1 at the beginning of a partition.

`tpc_time_code_a` and `tpc_time_code_b` specify the time code. The character-special tape interface does not support time-stamping. These fields do not contain valid values if they are created with the character-special tape interface.

A beginning-of-media (BOM) zone is created at the beginning of each partition when the tape is formatted. It consists of special physical blocks identifying the

logical beginning of a partition. `tpc_pos_bom` is set to 1 if the logical position is at the BOM. After a position to the BOM, a ER90 device is ready to process the first block of the first file section of the current partition.

The end-of-media warning (EMW) is the tenth physical block from the end of the partition. It provides a warning that the tape is positioned near the end of the partition. `tpc_pos_emw` is set to 1, for write operations, if the tape is positioned at or beyond the EMW of the current partition. It is set for read operations if the logical position is at or beyond the EMW of the current partition.

The early-end-of-media warning (EEW) is a tape location defined by the host. It provides a warning when the end of the partition approaches. `tpc_pos_eew` is set to 1, for write operations, if the tape is positioned at or beyond the EEW of the current partition. It is set for read operations if the logical position is at or beyond the EEW of the current partition.

The beginning-of-tape (BOT) is an area, located at the physical beginning of tape, used for tape loads and unloads. `tpc_pos_bot` is set to 1 if the tape is positioned at the BOT. There is no address associated with this area. The logical data block number, physical data block number, file section number, partition number, and absolute address fields are not valid when positioned at the BOT.

The end-of-recording (EOR) is recorded by the ER90 device after the last user data of the partition. `tpc_pos_eor` is set to 1 if the tape is positioned at the EOR.

The end-of-tape (EOT) is an area, located at the physical end of tape, used for tape loads and unloads. `tpc_pos_eot` is set to 1 if the tape is positioned at the EOT. There is no address associated with this area. The logical datablock number, physical datablock number, file section number, partition number, and absolute address fields are not valid when positioned at the EOT.

System zones are created on a tape volume when the volume is formatted. They provide an area of tape, other than the BOT and EOT zones, for loading and unloading a cassette. `tpc_pos_sysz` is set to 1 if the tape is positioned within a system zone. The system zone number is specified in `tpc_sysz_number`.

The end-of-media (EOM) is the end of the recording for a partition. `tpc_pos_eom` is set to 1 if the tape is positioned at the EOM of the current partition.

`tpc_valid_rem_part` is set to 1 if the `tpc_rem_partition` variable is valid. `tpc_rem_partition` specifies, in millions of bytes, the amount of data that can be recorded between the current position and the EOM.

`tpc_valid_rem_doubleframes` is set to 1 if the `tpc_rem_doubleframes` variable is valid. `tpc_rem_doubleframes` specifies the approximate number of double-frames (physical blocks) between the current position and the EOT.

6.2.4.4.10 MTSEEK

`MTSEEK` positions to a tape area specified in the `tpc_er90_seek` structure. This request is only valid for ER90 device files. `mt_arg` is a pointer to this structure. It is defined as follows:

```
struct tpc_er90_seek  {
  int  tpc_pos_flag;
  int  tpc_sysz_number;
}
```

`tpc_pos_flag` is a flag specifying the tape entity or area to position to. It is constructed from one of the following flags:

Flag	Description
`TPC_POS_EMW`	Positions to the EMW of the current partition
`TPC_POS_BOM`	Positions to the BOM of the current position
`TPC_LOAD`	Positions to a volume format information (VFI) zone
`TPC_POS_BOT`	Positions to the BOT
`TPC_INIT_POS`	Positions the tape to the BOM and initializes the volume
`TPC_POS_SYSZONE`	Positions to the system zone specified in `tpc_sysz_number`
`TPC_POS_EOT`	Positions the tape to the EOT
`TPC_POS_EOR`	Positions the tape to the EOR of the current partition
`TPC_PARK`	Positions the tape to the nearest system zone, in the BOT direction, and unthreads the tape

The `TPC_LOAD` request involves searching for and then reading the volume format information (VFI). This information is recorded when the cassette is formatted and consists of the format ID plus system zone and partition information. The operation is performed automatically when a cassette is loaded and should not have to be requested.

The TPC_INIT_POS request positions to BOM and then initializes the tape so that the tape is formatted during write operations. The tape is formatted with a NULL format ID, system zones, and one partition spanning the length of the tape. This request cannot be used on a cassette with an existing format that has a nonzero format ID.

The TPC_PARK request is used to minimize head wear. It positions to a system zone and then unthreads the tape from the helical scanner. Tape processing can resume at the current position without losing any buffered data and without issuing any additional requests.

For information on positioning with MTGPOS, see section Section 6.2.4.4.9, page 132.

6.2.4.4.11 MTEXTS

MTEXTS returns the extended status of a device for ER90 and block multiplexer device files.

For ER90 device files, it consists of the responses to commands: Report Addressee Status, Attribute, Operating Mode, and Report Position.

The Report Addressee Status Response describes the state of the ER90 device (ready/not ready or on-line/off-line), a description of the mounted volume, and the ER90 detailed status. The Attribute Response returns the operational characteristics of the ER90, for example, the data block size, burst size, early-end-of-media warning (EEW) location, and so on. The Operating Mode Response describes those attributes that have been defined only for as long as the tape is positioned within the current partition. The Report Position Response contains the current absolute track address, the remaining partition capacity, and other tape location information.

mt_arg is a pointer to the ctl_extsts structure. This structure is defined as follows:

```
struct ctl_extsts     {
      int     device;
      int     len_rep_addr;
      char     *rep_addr;
      int      len_attributes;
      char     *attributes;
      int      len_oper_mode;
      char     *oper_mode;
      int      len_report_pos;
      char     *report_pos;
```

```
}
```

To receive responses to all commands, `rep_addr`, `attributes`, `oper_mode`, and `report_pos` must be set to pointers to the memory into which the response packets are copied. To receive only select portions of the extended device status, the memory pointers of the response packets that are not desired must be set to `NULL`. For each command requested, the amount of memory allocated for the command must be set in the `len_rep_addr`, `len_attributes`, `len_oper_mode`, or `len_report_pos`. The length of each response packet is returned in these variables. Field device is not used for the character-special tape interface.

If the operating mode response is requested and a cassette is not loaded, the cassette is blank, or the logical position has not been established, an operating mode response is not returned.

`MTEXTS` returns the sense information of a device. This information contains the device status, tape position, recoverable error counters, and other information. `mt_arg` is a pointer to the buffer into which sense information is read. `mt_size` specifies the size of the buffer receiving the sense information. The buffer size must be at least 64 bytes.

6.2.4.4.12 MTFMT

`MTFMT` formats a cassette for ER90 device files. Formatting records a volume identifier, creates partitions (logical volumes), and, if requested, creates system zones. `mt_arg` is a pointer to a structure defining this format. The structure is defined as follows:

```
struct tpc_format {
     uint  tpc_preformat     : 1,
           tpc_syszone       : 1,
           tpc_pack          : 1,
           tpc_extend        : 1,
           tpc_waste         : 1,
           tpc_verify_volume : 1,
           tpc_unused_0      : 10,
           tpc_fmtid         : 48;
     uint  tpc_count_a       : 16,
           tpc_count_b       : 16,
           tpc_sysz_spacing  : 32;
     uint  tpc_size_a        : 32,
           tpc_size_b        : 32;
     uint  tpc_old_fmtid     : 48,
```

```
            tpc_unused_1      : 16;
}
```

`tpc_preformat` specifies whether the tape should be preformatted. If set to 1, the volume is preformatted with the information provided in the `tpc_format` structure. If `tpc_preformat` is set to 0, the tape is formatted during write operations. Multiple partitions cannot be requested if the tape is formatted during write operations.

`tpc_syszone` specifies whether system zones are created on the tape. System zones are data-free areas on the tape that can be used to load and unload the cassette. If `tpc_syszone` is set to 1, the volume is formatted with system zones. Otherwise, no system zones are created. If a volume is formatted without system zones, the volume is positioned to the beginning-of-tape (BOT) or the end-of-tape (EOT) when it is unloaded. It could take up to 185 seconds to complete the unload. If the default system zone spacing is used, the unload time can be reduced to approximately 16 seconds for small cassettes, 21 seconds for medium cassettes, and 24 seconds for large cassettes.

`tpc_pack` is set to 1 to allow partitions to span system zones. This option must be specified if a single partition is requested or if no system zones are requested. `tpc_pack`, `tpc_extend`, and `tpc_waste` are mutually exclusive.

`tpc_extend` is set to 1 to request that the ER90 attempt to minimize the amount of system zone discontinuities in a partition. If the ER90 device determines that a partition should be created after a system zone, the previous partition is extended to the system zone dividing the two partitions. This option cannot be specified if a single partition is requested or if no system zones are requested. `tpc_pack`, `tpc_extend`, and `tpc_waste` are mutually exclusive.

`tpc_waste` is set to 1 to request that the ER90 attempt to minimize the number of system zone discontinuities within a partition. If the ER90 device determines that a partition should be created after a system zone, the previous partition is not extended to the system zone dividing the two partitions. Instead, the area between the previous partition and the system zone is wasted. This option cannot be specified if a single partition is requested or if no system zones are requested. `tpc_pack`, `tpc_extend`, and `tpc_waste` are mutually exclusive.

`tpc_verify_volume` is used to request volume verification. If set to 1, the value specified in `tpc_old_fmtid` is compared with the ID recorded on the volume to be formatted. If the volume IDs do not match, the request is terminated with the `ETPD_BAD_REQT` error code.

`tpc_fmtid` specifies the identifier to be recorded on the tape during the volume format. The format identifier must not be longer than 6 alphanumeric characters.

`tpc_count_a` and `tpc_count_b` specify the number of A partitions and the number of B partitions that should be formatted. The number of A partitions specified must be in the range 1 through 255; the size is specified with `size_a` field.

The A partitions are formatted on the volume until all partitions have been created or the end of the tape is detected. If tape remains after formatting the A partitions and no B partitions are requested, the tape is formatted with A partitions until the EOT is detected.

The number of B partitions specified must be in the range 0 through 255. B partitions are created following the last A partition. If one B partition is requested with a size of 0, the volume is formatted with one B partition spanning the remainder of the volume. If you specify more than one B partition, the volume is formatted with B partitions until all partitions are formatted or until the EOT is detected.

If the end of the volume is not detected after creating the B partitions, formatting continues, beginning again with A partitions.

`tpc_size_a` and `tpc_size_b` specify the size of the partitions. The size of the partition is specified in millions of bytes and must be in the range 0, 0xF0 through 0x1312D00 (240 through 20,000,000).

If the A partition size is 0, one partition is created spanning the length of the volume. Any size specified for the B partition is then not valid. If the A partition size is 0, one B partition is created spanning the length of the tape remaining after the A partitions.

Nonstandard system zone spacing can be requested with field `tpc_sysz_spacing`. `tpc_sysz_spacing` specifies the length, in double frames, between system zones. The length specified must be in the range 0x842 through 0xFFFFFF. If this field is set to 0, the default system zone spacing is used.

6.2.4.4.13 `MTGFMT`

`MTGFMT` returns a description of the cassette and volume format of the currently loaded tape for ER90 device files. The format is described in the `tpc_fmtdesc` structure, which is copied into the buffer pointed to by `mt_arg`. The structure is defined as follows:

```
struct tpc_fmtdesc   {
     int        tpc_fmtid;
     uint       tpc_cas_not_supported : 1,
                tpc_cas_loaded         : 1.
                tpc_cas_size           : 2,
                tpc_tape_thickness     : 2,
                tpc_tape_coercivity    : 2,
                tpc_not_wrt_protected  : 1,
                tpc_not_pre_striped    : 1,
                tpc_volume_loaded      : 1,
                tpc_preformat          : 1,
                tpc_syszone            : 1,
                tpc_pack               : 1,
                tpc_extend             : 1,
                tpc_waste              : 1,
                tpc_partition_table    : 1,
                tpc_non_std_sysz_spc  : 1,
                tpc_physical_blktype  : 1,
                tpc_count_a            : 16,
                tpc_count_b            : 16,
                tpc_unused             : 9;
      uint      tpc_size_a             : 32,
                tpc_size_b             : 32;
      uint      tpc_sysz_spacing       : 32,
                tpc_sysz_size          : 32;
      uint      tpc_last_part_number   : 32,
                tpc_last_part_size     : 32;
}
```

`tpc_fmtid` specifies the identifier recorded on the tape during the volume format.

`tpc_cas_not_supported` specifies whether the cassette configuration is supported. The tape thickness, tape coercivity, the write protection mechanism, and prestripe state are evaluated to determine if the cassette is supported. This field is set to 1 if the cassette is not supported.

`tpc_cas_loaded` is set to 1 if the cassette is loaded. A cassette is loaded when it is inserted into the ER90 device, the tape cassette hubs and servo capstan hubs are interlocked, and the tape is positioned over the longitudinal heads. If this bit is set to 0, all other fields in the response are invalid.

`tpc_cas_size` specifies one of the following for the cassette size:

Setting	Description
0	Small cassette
1	Medium cassette
2	Large cassette

`tpc_tape_thickness` specifies one of the following for the tape thickness:

Setting	Description
0	16 micrometer tape
1	13 micrometer tape
3	Cleaning cassette

`tpc_tape_coercivity` specifies one of the following for the tape coercivity:

Setting	Description
0	850 oersted tape (D1)
1	1500 oersted tape (D2)
3	Cleaning cassette

`tpc_not_wrt_protected` is set to 1 if the tape is not write protected.

`tpc_not_pre_striped` is set to 1 if the tape has not been prestriped. Prestriping prerecords the longitudinal servo track.

`tpc_volume_loaded` is set to 1 if the volume in the device has been loaded. A volume reaches the loaded state after the volume format information has been read. If this bit is set to 0, the remainder of the fields in structure `tpc_fmtdesc` are invalid.

`tpc_preformat` is set to 1 if the volume has been preformatted.

`tpc_syszone` is set to 1 if the volume was formatted with system zones.

On UNICOS systems, `tpc_pack` is set to 1 if the volume was formatted with the `-z` option of the `tpformat`(8) command. `tpc_extend` specifies is set to 1 if the volume was formatted with the `-e` option of the `tpformat`(8) command. `tpc_waste` specifies is set to 1 if the volume was formatted with the `-w` option of the `tpformat`(8) command.

`tpc_partition_table` is set to 1 if the partition table has been recorded on the volume.

`tpc_non_std_sysz_spc` is set to 1 if the volume was formatted with nonstandard system zone spacing.

`tpc_physical_blktype` specifies one of the following physical block types. A physical block is the smallest unit in which data can be recorded on tape.

Type	Description
0	Type 0 physical blocks
1	Type 1 physical blocks

`tpc_count_a` specifies the number of A partitions formatted on the cassette. `tpc_count_b` specifies the number of B partitions formatted on the cassette.

`tpc_size_a` specifies the size of the A partitions, in millions of bytes. A value of 0, indicates that the partition spans the length of the tape. `tpc_size_b` specifies the size of the B partitions, in millions of bytes. A value of 0 indicates that the B partition spans the length of the tape remaining after the A partitions.

`tpc_sysz_spacing` specifies the distance between the system zones. The distance is specified in double frames.

`tpc_sysz_size` specifies the size of the system zones, in double frames. The size is fixed per cassette size. If no system zones have been formatted, the size is 0.

`tpc_last_part_number` specifies the number of the last partition formatted on the volume.

`tpc_last_part_size` specifies the size, in million of bytes, of the last partition formatted on the volume. If the volume was not preformatted, this field will be 0.

6.2.4.4.14 MTVERIFY

`MTVERIFY` verifies the integrity of the data recorded on tape for ER90 device files. `mt_arg` is a pointer to a structure defining the extent to which the tape should be verified and where the verification should begin. The structure is defined as follows:

```
struct tpc_verify {
     uint       tpc_extent          : 4,
                tpc_position        : 1,
                tpc_unused_0        : 59;
     uint       tpc_valid_logdb     : 1,
```

```
                tpc_valid_absaddr     : 1,
                tpc_valid_partition   : 1,
                tpc_valid_filesec      : 1,
                tpc_valid_timecode     : 1,
                tpc_unused_1           : 11,
                tpc_logical_datablock : 48;
     uint       tpc_absolute_address  : 32,
                tpc_file_section       : 32;
     uint       tpc_partition_number   : 16,
                tpc_time_code          : 48;
}
```

`tpc_extent` specifies the extent to which the tape should be verified. It is one of the following flags:

Flag	Description
`TPC_VERIFY_FILESEC`	Verifies the integrity of the data within the specified file section
`TPC_VERIFY_PARTITION`	Verifies the integrity of the data within the specified partition

File verification leaves the tape positioned after the last data block of the file section. Partition verification leaves the tape positioned after the last data block of the last file section of the partition.

`tpc_position` is set to 1 to request that the tape be positioned to the absolute address specified before verifying the integrity of the recorded data.

For a description of the absolute address fields, see Section 6.2.4.4.8, page 130.

6.2.4.4.15 MTTRACE

`MTTRACE` reads the device trace for ER90 device files. `mt_arg` is a pointer to the buffer into which the device trace is read. The trace information is always 2,399,680 bytes in length.

The ER90 data buffer is used to transfer the trace information. This request will, therefore, destroy all user data in the device buffer.

6.2.4.4.16 MTMSG

MTMSG displays a message on a tape device. mt_arg is a pointer to a buffer containing the string to be displayed. mt_size specifies the length of the message. For ER90 devices, the length of the message is limited to 8 characters. For BMX devices, the length is limited to 16 characters.

For BMX devices, mt_count specifies the type of message display. This field must be set to one of the following flags:

Flag	Description
FMsgAcl	Specifies that a load request be sent to an automatic cartridge loader.
FMsgHigh	Specifies that the characters in bytes 8 through 15 of the message buffer be displayed. By default, the message in bytes 0 through 7 will be displayed.
FMsgBlink	Specifies that the message be displayed intermittently. The message will be displayed for 2 seconds at intervals of 0.5 seconds.
FMsgAlt	Specifies that the device alternate between displaying the characters in bytes 0 through 7 and the characters in bytes 8 through 15. Each message will be displayed for 2 seconds at intervals of 0.5 seconds.
FMsgUnload	Specifies that the message in bytes 0 through 7 be displayed until a cartridge is unloaded from the tape device. If no cartridge is loaded, the message will be displayed only briefly.
FMsgLoad	Specifies that the message in bytes 0 through 7 be displayed until the tape device is next loaded.
FMsgNone	Specifies that no message be displayed.
FMsgHighUntilLoad	Specifies that the message in bytes 8 through 15 be displayed until the device is next loaded.

6.3 Hardware error codes

When a request cannot complete because of an IOP or device-detected error, one of the following error codes is returned.

Error code	Description
ETPD_BAD_REQT	The contents or format of a request are incorrect, or the sequence of requests issued is incorrect.
ETPD_BLANK_TAPE	The command was terminated because it cannot be issued to a device with a blank tape loaded.
ETPD_BOT	The beginning of tape or beginning of partition was detected.
ETPD_DATA_ERROR	An unrecoverable data error occurred.
ETPD_DEV_HUNG	A response was not received from the tape device.
ETPD_DEVBUSY	The device is busy.
ETPD_DEVICE	A device error occurred.
ETPD_EOM	The end of media was detected.
ETPD_EOR	The end of recording was detected.
ETPD_FORMAT	The volume format is not supported.
ETPD_HPCONN	A HIPPI connection error occurred.
ETPD_HPDATA	A HIPPI parity or checksum error occurred.
ETPD_HPREQ	A HIPPI request error occurred.
ETPD_HPTRNS	A HIPPI transmission error occurred.
ETPD_IOPERR	An IOP error occurred.
ETPD_IPCONN	An IPI connection error occurred.
ETPD_LGPS	A logical position has not been established.
ETPD_MAX_IOREQT	I/O request exceeded maximum size allowed.
ETPD_MEDIA	The media is not supported.
ETPD_NO_CASSETTE	The cassette is not loaded.
ETPD_NOT_BOF	The tape is not positioned at the beginning-of-file.
EPTD_NOT_OPER	A hardware error occurred.
ETPD_NOT_READY	Device is not ready.
ETPD_POSACC_ERR	The position cannot be accessed.
ETPD_SHPI	A HIPPI controller error occurred.
ETPD_SYSTEM	A tape driver error occurred.
ETPD_TAPE_ADDR	An invalid tape address occurred.

ETPD_TAPE_ERROR	A problem with the tape media occurred.

Interpreting System Messages [A]

This appendix lists and describes the system messages, error or informative, that you may encounter while you are working with the tape subsystem. These system messages are found in either the `tape.msg` file or your standard output file. For system messages indicating that a tape daemon error has occurred, contact your system administrator. For internal tape subsystem problems), contact your system support staff.

Each message description is followed by a label (USER, OPERATOR, or USER/OPERATOR) signifying the recipient of the message.

`TM000 - tape resource reserved for you`
Tape resources have been reserved for you by means of the `rsv`(1) command. - USER

`TM001 -` *file name* `: request pipe open error :` *errno*
The request pipe *file name* could not be opened; the error returned is *errno*. This message may indicate that the tape daemon is not running. - USER

`TM002 - incorrect value :` *string*
The system does not recognize *string* as a correct parameter. - USER

`TM003 -` *device group name* `: daemon does not have it`
The tape daemon does not have devices belonging to *device group name*. This indicates an error in the tape daemon. - USER

`TM004 -` *device group name* `: resource count invalid :` *count*
The number *count* of resource *device group name* is incorrect. This indicates an error in the tape daemon. - USER

`TM005 - too many devices : max =` *maximum number*
The number of devices specified in the `rls`(1) or `tpu`(8) command exceeds the maximum of *maximum number* allowed. - USER

`TM006 - invalid request to tape daemon`
An invalid request was made to the tape daemon. This indicates either an error in the tape daemon or an invalid request issued from a C program. - USER

`TM007 - not used`

`TM008 - should volume` *vsn* `on device` *device* `switch from` *label type1* `to` *label type2* `for user` *user id*`? reply y/n`

The operator must specify whether the tape named *vsn* on *device* may be switched from *label type1* to *label type2* for user *user id*. The operator replies `y` for yes or `n` for no. - USER/OPERATOR

`TM009 -` *pipe name* `: unable to read requests :` *errno*

The tape daemon tried to read from pipe *pipe name* and got an error return of *errno*. This indicates an error in the tape daemon. - USER

`TM010 - cannot initialize table :` *table name*

The tape daemon is unable to initialize table *table name*. This indicates an error in the tape daemon. - USER

`TM011 - enter vsn for tape on device` *device name*

You must specify the volume identifier of the volume on device *device name*. - OPERATOR

`TM012 - Unable to obtain memory for` *variable*

The tape daemon or tape command could not acquire memory for the *variable* variable. This indicates an error in the tape subsystem. When this message is issued, the tape daemon will exit. The operator or administrator should collect the trace files for examination by software product support. - OPERATOR

`TM013 - tape` *vsn* `on device` *device name* `not expired; reply y/n for user` *user id* `to write on tape`

You must specify whether user *user id* may write on unexpired tape *vsn*. Reply `y` for yes or `n` for no. - OPERATOR

`TM014 - chown error`

The tape daemon issued a `chown`(2) command, and an error was received. - OPERATOR

`TM015 -` *pipe name* `: unable to get reply`

The tape daemon or a command is unable to read a reply from pipe *pipe name*. This indicates an error in the tape daemon. - USER

`TM016 - cannot create` *pathname*

The tape daemon is unable to create *pathname*. Check to see whether you have write permission to the directory or to path name *pathname*. - USER

`TM017 - `*string*` : value of `*option*` exceeds `*count*` character`
The number of characters in *string* is larger than *count* and is the value of option *option*. - USER

`TM018 - duplicated option : `*option*
The *option* option is duplicated on your command line. - USER

`TM019 - no device available`
No device is available for reservation (see `tprst`(1)). - USER

`TM020 - not used`

`TM021 - Exceeded the maximum number of vsn's allowed, `*maxvsn*
The number of volume identifiers in the volume identifier list is greater than *maxvsn*. Either use fewer volume identifiers or see your system administrator to change *maxvsn*. - USER

`TM022 - Options `*option1*` and `*option2*` are mutually exclusive`
Options *option1* and *option2* are mutually exclusive. You may use only one of them. - USER

`TM023 - A path name must be specified`
Specify the path name by using the `-p` or `-P` option on the `tpmnt`(1) command line. - USER

`TM024 - Unable to create [file|directory] `*file*`|`*directory*` (errno = `*errno*`)`
The tape subsystem was unable to create a file or directory; the error returned is *errno*. Check to see whether you have the correct permissions for creating the file or directory. -USER/OPERATOR

`TM025 - release previous reservation before issuing reserve`
You have issued a `rsv`(1) command, but you must release all previously reserved resources by using the `rls`(1) command. - USER

`TM026 - cannot communicate with tape daemon`
A child of the tape daemon cannot communicate with the tape daemon. This indicates an error in the tape daemon. - OPERATOR

`TM027 - not used`

`TM028 -` *pathname* `: ` *vsn* `: mount canceled by operator`
The operator canceled your mount request for file *pathname* and volume *vsn*. - USER

`TM029 - all tape resources released`
The tape daemon has released all tape reservations. - USER

`TM030 - not used`

`TM031 -` *pathname* `: cannot write with no ring`
You requested the *no-ring* option on the `tpmnt`(1) command and issued a write operation to the *pathname* file, but the volume mounted has no ring. - USER

`TM032 - not used`

`TM033 - could not execute program for device`
The tape daemon cannot execute a program for the device. This indicates an error in the tape daemon. - OPERATOR

`TM034 -` *pipe name* `: can't read less than size of rephdr`
The tape daemon cannot read a request which has a size less than that of the reply header. This indicates an error in the tape daemon. - OPERATOR

`TM035 - not used`

`TM036 - volume offset > number of vsn's`
The value specified on the offset option is larger than the number of volume identifiers in the volume identifier list. - USER

`TM037 - no such user`
The tape daemon cannot find the user specified in the command. - OPERATOR

`TM038 -` *pipe name* `: unable to send request`
The tape subsystem is unable to send the request by using *pipe name*. This may indicate that the tape daemon is not running. - USER/OPERATOR

`TM039 -` *pathname* `: path name being used for another tape file`
Another tape file called *pathname* is being used by either you or another user. - USER

`TM040 -` *pathname* `: please recreate entry before using`
The tape daemon cannot read the *pathname* entry. Delete it and re-create the entry before using it again. - USER

`TM041 - can't send action message; check message daemon`
The tape daemon cannot communicate with the message daemon. - OPERATOR

`TM042 - cannot find tdt with pid` *pid*
The tape daemon cannot find a tape device table with a process ID of *pid*. This indicates an error in the tape daemon. - OPERATOR

`TM043 - value` *value* `of` *option* `is invalid`
The value of *value* is invalid for the *option* option. - USER

`TM044 - cannot open tape device`
The tape daemon cannot open the tape device required by a command. - USER/OPERATOR

`TM045 -` *device* `: invalid device name`
Device name *device* was specified in a command, but it is invalid. - OPERATOR

`TM046 -` *mount* `or` *remount* `tape` *vsn* `(`*label type*`)` *ring option* `on device` *device name* `for` *userid pid*`, (`*reason*`) or reply cancel / device name`
The operator must mount the tape with volume identifier *vsn*, a label of *label-type*, write ring in or out, on device *device name*, for user *userid* with process ID of *pid*. An optional *reason* may be given. The operator may mount the tape on the specified drive, reply with a different device name, or reply `cancel`. If the operator replies `cancel`, the tape mount is canceled and the user cannot continue with tape processing. This message is displayed when automatic volume recognition (AVR) is turned off and is analogous to TM122. - USER/OPERATOR

`TM047 -` *pathname* `:` *device* `:` *function* `:` *code* `: errno =` *errno*
An error occurred when the *function* executed with *code* for file *pathname* on device name *device*. The error number is *errno*. - USER/OPERATOR

`TM048 -` *pathname* `:` *assigned* `or` *reassigned* `to` *device name*
File *pathname* is assigned or reassigned to *device name*. - USER/OPERATOR

`TM049 -` *pathname* `:` *vsn*`(`*label type*`) :` *function* `: blocks =` *number*

number blocks were read or written to file *pathname* with *vsn* and *label type*. - USER

`TM050 -` *pathname* `: released`
File *pathname* is released. - USER

`TM051 - not used`

`TM052 -` *pathname* `: block count error : system =` *number1*`, label =` *number2*
A block count error was issued to the tape mounted on *pathname*. The tape subsystem has *number1* blocks, and the label on the tape has *number2* blocks. - USER

`TM053 -` *function* `: unexpected signal received : signal =` *signo*
The *function* received an unexpected signal (signal number *signo*). This indicates an error in the tape subsystem. - OPERATOR

`TM054 - invalid device name`
An invalid device name was specified on the `tpmnt`(1) command. - USER

`TM055 - invalid device group name`
An invalid device group name was specified on the `rsv`(1) or `tpmnt`(1) command. - USER

`TM056 - device group not reserved`
Either the device group name on the `tpmnt`(1) command does not match the device group name you used on the `rsv`(1) command or you have not issued an `rsv` command. - USER

`TM057 -` *pathname* `: file not mounted`
The *pathname* used in the `rls`(1) command was not mounted with a `tpmnt`(1) command. - USER

`TM058 -` *command* `: interrupted by signal` *signo*
The *command* (`rsv`(1) or `tpmnt`(1)) has been interrupted by signal *signo*. - USER

`TM059 -` *pathname1* `and` *pathname2* `have the same device id.`
Files *pathname1* and *pathname2* have the same device ID. This indicates a configuration error. - OPERATOR

`TM060 -` *pathname* `: waiting for device` *dgn*
Path name *pathname* is waiting for device group name *dgn* during `tpmnt`(1) command processing. - USER

`TM061 -` *pathname* `: can't update directory`
You cannot update the directory for *pathname*. This usually means that you do not have write permission in the directory. - USER

`TM062 -` *pathname* `: volume protected :` *vsn*
Volume identifier *vsn* mounted on *pathname* is volume protected. See the system administrator. - USER

`TM063 -` *pathname* `: incorrect label type :` *vsn*
Volume identifier *vsn* mounted on *pathname* has an incorrect label type. Check your tape. - USER

`TM064 -` *pathname* `: file not on volume :` *vsn*
Volume identifier *vsn* mounted on *pathname* does not contain the specified file. Check your tape. - USER

`TM065 -` *pathname* `: file not expired :` *vsn*
Volume identifier *vsn* mounted on *pathname* does not contain the specified file in an expired state. - USER

`TM066 - not used`

`TM067 -` *pathname* `: no vsn for file`
There is no VSN list for file *pathname*. - USER

`TM068 - not used`

`TM069 - invalid function from driver :` *function*
An invalid function was received from the driver. This indicates an error in the tape daemon. - OPERATOR

`TM070 -` *device pathname* `IOP Status:` *status* `Function:` *iop function*
File *pathname* has an error when performing an IOP function on device named *device*. The function is *iop function* and the IOP status is *status*. - USER

`TM071 -` *device pathname* `Invalid IOP Response:` *status flag* `Function:` *iop function*

File *pathname* received an invalid response to an IOP or device request. The function is *iop function* and the IOP status is *status flag*. - USER

`TM072 - too many device types, max =` *max number*
Too many device groups are specified in the configuration file. The maximum is *max number*. - USER

`TM073 - incomplete reply from tape daemon`
The incomplete reply you received from the tape daemon indicates an error in the tape daemon. - USER

`TM074 - no response from tape daemon`
The tape daemon has not responded. This indicates an error in the tape daemon. - USER/OPERATOR

`TM075 - not used`

`TM076 -` *pathname* `: invalid label structure :` *vsn*
Volume *vsn* containing file *pathname* has an invalid label structure. - USER

`TM077 - Cannot find` *tusr* `structure for` *jid %d*
A valid user identification structure could not be found during job exit processor. - OPERATOR

`TM078 - tape daemon stopped`
The tape daemon is stopped. Either a `tpdstop`(8) command has been issued or an error has occurred. - OPERATOR

`TM079 - Invalid %s`
Validation failed for either a *fit* or *tusr* structure. - OPERATOR

`TM080 -` *pathname* `: no matching fit :` *file id*
The file with *pathname* and *file id* has no matching File Information table. This indicates an error in the tape daemon. - USER

`TM081 -` *pathname* `: bad file sequence number :` *fseq*
The file indicated by file sequence number *fseq* is not on the tape. - USER

`TM082 - not used`

`TM083 - invalid dolist function :` *code*

Function code *code* is invalid. This indicates an error in the tape daemon. - OPERATOR

`TM084 - tape system error`
The tape system returned an error indicating an internal tape subsystem error. - USER

`TM085 - no volume serial number`
You must specify a VSN. - USER

`TM086 - tape daemon error code :` *error code*
The tape daemon returned error *error code*. - USER

`TM087 - incorrect range`
The range of devices is incorrect in the `tpconfig`(8) or `tplabel`(8) command. - OPERATOR

`TM088 -` *pathname* `: file exists`
You specified *pathname* on the `tpmnt`(1) command by using the `-p` option, and *pathname* exists. The `-p` option of the `tpmnt`(1) command does not delete file *pathname* if it exists. You can either delete file *pathname* or use the `-P` option. - USER

`TM089 -` *pathname* `: is a directory`
You specified *pathname* on the `tpmnt`(1) command by using the `-p` or `-P` option; *pathname* is a directory. - USER

`TM090 - environment variable USER_DIR not set up`
Environment variable `USER_DIR` was used by the `rsv`(1) command, but it is not set up correctly. - USER

`TM091 -` *pathname* `: pathname >` *number* `characters`
The *pathname* specified is larger than the maximum of *number* characters accepted by the tape subsystem. See the system administrator. - USER

`TM092 - Unable to get the current working directory (errno =` *number*`)`
The tape subsystem cannot get your current working directory. The `errno` is *number*. - USER

`TM093 - can't open user's request pipe`

The tape subsystem cannot open your request pipe. Check permissions in the directory and on the pipe file. - USER

`TM094 - mount request postponed because of unfinished request`
The `tpmnt`(1) command was postponed to finish another request. - USER

`TM095 - rsv failed, maximum tape user limit reached`
The maximum number of tape users was exceeded. - USER

`TM096 - not used`

`TM097 - the following tape users are deadlocked`
The following users are deadlocked during device allocation. - USER

`TM098 -` *pathname* `: possible deadlock, allocation delayed`
pathname has a possible system deadlock, and allocation is delayed. - USER

`TM099 - open failed, file not known to tape daemon`
Path name was not known to the tape daemon when it processed the request code sent by the user. - USER

`TM100 - invalid pid for kill: pid =` *pid*
Process identifier *pid* was not a valid process identifier when the tape daemon tried to kill a process. - USER/OPERATOR

`TM101 - device release pending`
When the tape daemon was processing a release request, the device could not be released immediately. It will be released as soon as possible. - USER

`TM102 - waiting for previous release to complete`
The tape daemon received your reserve request and is waiting for the release of devices from a previous release request. The tape daemon delays the processing of your reserve request until all pending releases are completed. - USER

`TM103 - system requires ring out with blp`
The tape daemon is installed with the option that `ring out` must be used when label type `blp` is used. - USER

`TM104 - operator replied :` *reply-string*

The operator replied *reply-string* to an operator message about your tape. - USER

`TM105 -` *program* `: not part of a job`
The *program* is not part of a job. This indicates a system error. - USER

`TM106 -` *program* `: can't get job table : errno =` *errno*
The tape subsystem cannot get the job table to *program*; `errno` is *errno*. This indicates a system error. - USER

`TM107 -` *user* `: not a super user`
user is not a super user. This message is displayed when you attempt to bring up the tape daemon. - OPERATOR

`TM108 -` *program* `: job table full`
The operating system returned job table full status when you attempted to bring up the tape daemon. - OPERATOR

`TM109 - request exceeds job limit`
You issued a `rsv`(1) command, exceeding your current job limit for tape resources. - USER

`TM110 - not used`

`TM111 -` *pathname* `: read/write tape mark not allowed`
You attempted to read or write a tape mark to tape file *pathname* without using the `-T` option of the `tpmnt`(1) command. - USER

`TM112 -` *option1* `option must have` *option2* `option also`
You must specify *option2* along with *option1* on the `rls`(1) command. - USER

`TM113 -` *pathname* `: missing label :` *vsn*
Your *pathname* tape file either does not have a valid label or is missing a label for *vsn*. - USER

`TM114 - invalid EOV select for` *pathname* `:` *number*
number is not correct on a user end-of-volume (EOV) `select`(2) request for *pathname*. Valid values for the `TR_EOV` request are `EOV_ON` and `EOV_OFF`. - USER

`TM115 - user EOV processing set to` *value* `for` *pathname*

This is an informational message sent to your `tape.msg` file. *value* can be set to `on` or `off` for *pathname* during a `TR_EOV` request. - USER

`TM116 - user EOV processing not selected for` *pathname*
pathname has requested a function that is not available unless user EOV processing is turned on. - USER

`TM117 - active user EOV processing started for` *pathname*
This is an informational message sent to your `tape.msg` file during a `TR_BEOV` request, indicating that user EOV processing has begun for *pathname*. - USER

`TM118 - active user EOV processing ended for` *pathname*
This is an informational message sent to your `tape.msg` file during a `TR_EEOV` request, indicating that user EOV processing has ended for *pathname*. - USER

`TM119 - sync requested to flush buffers for` *pathname*`: blocks =` *number*
This is an informational message sent to your `tape.msg` file during a `TR_SYNC` request, indicating that *pathname* has requested that the buffers be flushed. - USER

`TM120 - sync requested for` *pathname* `: not output tape`
You have requested a `TR_SYNC` for *pathname*, but you are not writing to the tape. - USER

`TM121 - not used`

`TM122 -` *mount* `tape` *vsn* `(`*label type*`)` *ring option* `on a` *dgn* `device for` *userid jobid*`,` *NQSid* `(`*reason*`) or reply cancel / device name`
The operator must *mount* the tape with volume identifier *vsn*, a label of *label type*, write ring in or out, on a device of device group name *dgn* for user *userid*, with job ID of *jobid*, and optional NQS ID of *NQSid*. An optional reason may be given. This message is displayed on an AVR system and is analogous to TM046. - USER/OPERATOR

`TM123 -` *device name* `: device assigned`
You tried to unload *device name* with the `tpu`(8) command, but the device is already assigned. - OPERATOR

`TM124 - AVR not active`
You tried to use the `tpu`(8) command, but AVR is not active. - OPERATOR

`TM125 -` *device name* `: device down`
You tried to unload *device name* with the `tpu`(8) command, but the device is configured down. - OPERATOR

`TM126 - tape subsystem busy, unable to set` *type* `option`
You used the `tpset`(8) command to change a tape daemon option, but the tape subsystem is busy. The options available for *type* are `avr`, `front-end servicing`, `Cray/REELlibrarian`, `tape operator id`, and `tracing`. - OPERATOR

`TM127 - not used`

`TM128 -` *pathname* `: opened file on volume`
You have tried to issue a second open to *pathname*. - USER

`TM129 -` *pathname* `: invalid position request :` *number*
The *number* you specified for *pathname* is an invalid tape positioning request. - USER

`TM130 -` *pathname* `: cannot read a new file`
You specified the `-n` option on `tpmnt`(1) and attempted to read from *pathname*. Only writing is allowed for the first I/O to a new file. - USER

`TM131 -` *message_type* `: error sending station message:` *reason*
The tape subsystem received an error when it tried to send a station message to a servicing front end. *reason* was received as the reason for the error. - OPERATOR

`TM132 -` *message_type*`: file id` *file id* `- access denied`
The servicing front end denied access to the file ID *file id*. - USER

`TM133 -` *message_type*`: mainframe is not secure`
The servicing front end is not secure. - USER

`TM134 -` *message_type*`: file id` *file id* `already exists in catalog`
The servicing front end returned an error because *file id* was specified as a new file and it already exists in the catalog. - USER

`TM135 -` *message_type*`: file id` *file id* `not in catalog`
The servicing front end returned an error because *file id* was specified as an existing file and it does not exist in the catalog. - USER

`TM136 - not used`

`TM137 - `*`message_type`*` : file id `*`file id`*` - dataset update failed`
The servicing front end returned an error because the catalog update failed. - USER

`TM138 - `*`message_type`*` : volume `*`volume`*` - access denied `*`optional-reason`*
The servicing front end denied access to volume *volume*. - USER

`TM139 - `*`message_type`*` : volume `*`volume`*` not in volume catalog`
The servicing front end returned an error because volume *volume* does not exist in the volume catalog. - USER

`TM140 - `*`message_type`*` : volume `*`volume`*` - volume update failed`
The servicing front end returned an error because the volume catalog update failed. - USER

`TM141 - `*`message_type`*` : error building station message (ITSP)`
An error occurred during an attempt to build a station message. Contact your system support staff - USER/OPERATOR

`TM142 - invalid station message type `*`type`*` (ITSP)`
This is an invalid station message type. Contact your system support staff. - USER/OPERATOR

`TM143 - bad station message word count `*`count`*`, expected `*`count`*` (ITSP)`
The text of this station message is not of the size expected. Contact your system support staff. - USER/OPERATOR

`TM144 - error reading station message reply, errno = `*`errno`*` (ITSP)`
An error occurred while a station message reply was read from `USCP`. Contact your system support staff. - USER/OPERATOR

`TM145 - `*`message_type`*` : error `*`operation string`*` : errno = `*`errno`*
An error occurred while the specified operation was being performed. - USER

`TM146 - `*`message_type`*` : error reading `*`string`*` : read `*`count`*`, expected `*`count`*

A `read`(2) operation returned a count different from that expected. - USER

`TM147 - invalid front-end id` *id*
Front-end ID *id* is invalid. - USER

`TM148 - error sending message to front-end`
An error occurred during an attempt to send a station message. - USER

`TM149 - request rejected by front-end`
The servicing front end rejected your request. - USER

`TM150 - bad station message reply,` *structure* `structure missing (ITSP)`
A required table is missing from a station message reply. Contact your system support staff. - USER/OPERATOR

`TM151 -` *message_type* `: front-end servicing is turned off`
Front-end servicing is turned off. - USER

`TM152 - path` *path* `not found`
Path *path* does not exist, or it does not belong to you. - OPERATOR

`TM153 - no path to device`
This component cannot be configured up because there is no path to it. - OPERATOR

`TM154 - associated devices still up`
This component cannot be configured down because it would leave another component without a path to it. - OPERATOR

`TM155 -` *message_type* `: timed out waiting for front-end reply`
Timed out while waiting for a reply from the servicing front end. - USER

`TM156 -` *message_type* `: front-end` *front-end id* `not accepting station messages`
The specified servicing front end does not accept type `3` station messages. - USER

`TM157 -` *message_type* `: file must be closed when cataloging`
The `tpcatalog`(1) command can be used only when the file is closed. - USER

`TM158 - unable to send` *message* `for` *user job-id*`,` *nqs-id* `to front-end` *target-id*`(`*reason*`), reply cancel or retry`
message to server or front-end ID cannot be sent to *user* because *reason*. - OPERATOR

`TM159 - operator message aborted by UNICOS operator`
The system operator canceled the message that could not be sent to the front end. - USER

`TM160 - user volume` *vsn* `closed during EOV processing`
Volume identifier *vsn* was closed during user EOV processing. - USER

`TM161 - Unable to open file` *filename* `(errno =` *errno*`)`

The *filename* file, could not be opened; the error returned is *errno*. Check to see if this file exists or whether you have the correct permissions for opening the *filename* file. - USER/OPERATOR

`TM162 - extra parameters at end of command :` *string*
Extra characters (*string*) were specified in the `tpmnt`(1) command. - USER

`TM163 -` *program name* `(pid` *process id*`): server` *server name*`:` *error text*`: reply retry or cancel`
`program name` encountered a problem communicating with server `server name` on behalf of one or more requests. The problem is described in `error text`. A reply of cancel aborts all requests that have encountered this problem. A reply of retry requeues all requests that encountered this problem and allocates more time for these requests to wait for the communication with server `server name`. - OPERATOR

`TM164 - Unable to read file` *filename* `(errno =` *errno*`)`
An error occurred when attempting to read the *filename* file; the error returned is *errno*. Check to see if this file exists or whether you have the correct permissions for reading the *filename* file. - USER/OPERATOR

`TM165 -` *pathname* `: you must use list i/o for this function`
You requested a user EOV function, but tapelist I/O is not being used for *pathname*. - USER

`TM166 - You must begin end-of-volume processing before requesting another function`

You reached end-of-tape during user EOV processing, but you requested an invalid function (such as tape positioning) before starting special processing. - USER

`TM167 - tpmnt -b value` *text*
The maximum block size specified with the `-b` option of the `tpmnt`(1) command is zero or exceeds the maximum block size specified in the configuration file. Modify the maximum block size specified with the `-b` option of the `tpmnt`(1) command. - USER

`TM168 - You must be at end of tape before requesting start EOV for tape file` *file*
A request to start user end-of-volume special processing was issued before the end-of-tape status was detected. Modify the tape job to wait for the end-of-tape status (`ENDSPC`) to be returned before beginning user end-of-volume processing. - USER

`TM169 - No tape devices defined in the CNT - tape daemon terminating`
No tape drives were defined in the tape configuration file. Modify the configuration file and restart the tape daemon. - OPERATOR

`TM170 -` *pathname* `: device status` *status* `: errno` *error* `: user close required, no further I/O allowed`
The user received an error on a tape request. No further requests will be accepted except for a close request. Close and reopen the tape file. - USER

`TM171 - '-l st' may not be used with -T or fseq > 1`
Single tape mark format tapes are not allowed with the `-T` option or with a file sequence number (`-q` option) greater than 1. - USER

`TM172 - secure label violation`
The label of the tape prevents the user from accessing the file. - USER

`TM173 - bypass or unlabel permission required`
You requested nonlabeled or bypass-label processing on the `tpmnt`(1) command, but you do not have permission. See your system administrator. - USER

`TM174 -` *filename* `: file already opened`
You requested the opening of a file that is already open. The `open`(2) request has been terminated. - USER

`TM175 - logical and physical device pointers error (ITSP)`
The tape daemon reselects from one tape device to another by modifying data structures that correspond to the tape devices. This message is issued when the tape daemon is unable to modify the data structures because of a system error. Contact your system support staff. - USER

`TM176 through TM178 - not used`

`TM179 - Operator request aborted; loader in unattended mode`
With the tape loader in unattended mode, operator requests are not valid; thus the request was aborted. - OPERATOR

`TM180 - unable to find out if tape daemon is active`
An attempt to communicate with the tape daemon failed. Contact your system support staff. - USER

`TM181 -` *pathname* `: path not found to concatenate`
The *pathname* specified in the `-c` option of the `tpmnt`(1) command does not exist. - USER

`TM182 - must not specify -p or -P with -c`
The `-p`, `-P`, and `-c` options on the `tpmnt`(1) command are mutually exclusive. You may enter only one of the three. - USER

`TM183 - cannot concatenate new/append files`
The `-c` option of the `tpmnt`(1) command was used to request that multiple tape files be read as though they were one tape file. This feature can be specified with either the `-n` or `-a` option. Correct the option specified and reissue the `tpmnt`(1) command. - USER

`TM184 - only valid positioning is rewind with concatenation`
The position request was terminated because the tape file is a concatenated file. The only valid positioning request for concatenated files is the rewind request. - USER

`TM185 - rls was received and close was not issued for file` *filename*
The *filename* for this message is replaced by the path name.

`TM186 - not used`

`TM187 - Cannot position past the beginning of file` *file*
A request to position backward by blocks was terminated because the beginning of the file was detected. - USER

`TM188 - Cannot position past the end of file` *file*
A request to position forward by blocks was terminated because the end of the file was detected. - USER

`TM189 - Secure mount violation`
The security label is outside the device group security label range. See your security administrator. - USER

`TM190 - **WARNING** device` *device_name* `(autoloader:` *autoloader_name*`, server:` *server_name*`) is in state:` *state*
During the initialization of the *server_name* server for the *autoloader_name* autoloader, the *device_name* tape drive was reported by the server to be in the *state* state. Either check the state of the tape drive with the server and alter its state so that it can be used on the Cray Research system or do not attempt to configured it up. - OPERATOR

`TM191 -` *message_type* `: file has not been accessed yet`
The `tpcatalog`(1) command can be used only after the file has been opened and closed. - USER

`TM192 - no servicing front-end id`
The `tpcatalog`(1) command cannot be used if a servicing front end is not being used. - USER

`TM193 -` *message_type* `:` *operation* `permission denied by front-end`
The servicing front end has not given permission to perform the specified operation. - USER

`TM194 - resending operator message...`
The message is being sent again to the front-end operator. - USER

`TM195 - Not operational`
This tape device (drive) cannot be configured. - OPERATOR

`TM196 - Not available`
This tape device (controller or channel) cannot be configured. - OPERATOR

`TM197 - parameter error`

The option specified on the tape command is invalid. The command has been terminated. - USER

`TM198 - AVR turned off because of -d option`

If the `-d` option of `tpdaemon`(8) is specified, it causes AVR to be turned off. - OPERATOR

`TM199 - Tape daemon not available`

The tape daemon is not responding to tape requests. Either the tape daemon is not running or there is a tape daemon error. - USER

`TM200 - supplementary logfile message from` *filename*

This message issues a supplementary log file message from the front end. The message corresponds to the front-end request. - USER

`TM201 - Tape daemon not active`

A tape daemon request could not complete because the tape daemon is not active. - USER

`TM202 - User end of volume processing has already been selected, request ignored`

A request to select user end-of-volume processing was ignored because user end-of-volume processing has already been selected. - USER

`TM203 - User end of volume processing is not enabled, disable request ignored`

A request to deselect user end-of-volume processing was ignored because user end-of-volume processing is not currently selected. - USER

`TM204 - A configuration request is pending, request ignored`

A `tpconfig`(8) command cannot be completed because a previous configuration command is still pending. - OPERATOR

`TM205 - tpmnt -q file_seq_number required for file:` *file*

An invalid file sequence number was specified. If the multifile volume allocation was specified, unique files were not specified or the file specified does not exist. - USER

`TM206 - Mount failure on drive` *device* `(`*reason*`)`

A mount request failed on drive *device* because of *reason*. - USER

`TM207 - Should short tape <`*vsn*`> on` *device* `be used? Reply y/n`
No description available.

`TM208 - A device name must be specified`
The command requires that a device name be specified. Reissue the command specifying a device name. - USER

`TM209 -` *filename*`: can't write to read-only file`
You have attempted to write to a file that does not have `write`(2) permission. The `write`(2) request has been aborted. - USER

`TM210 - Invalid Media Loader specification.`
Either an invalid communication path was specified when a loader was defined in the configuration or parameter file, or an invalid loader was specified on a `tpscr`(8) request. - USER

`TM211 - Invalid loader type` *type*`.`
An invalid loader type was specified when the loaders were defined. - USER

`TM212 - Invalid communication path` *pathname.*
An invalid communication path was specified when a loader was defined in the configuration or parameter file. - USER

`TM213 - Loader` *name* `cannot change ring status, user aborted.`
A tape volume in the loader does not have the correct ring status. The loader is unable to correct the ring status. The tape request has been aborted. - USER

`TM214 - Loader Reselect (for` *name*`) Not Supported.`
The loader cannot remount a volume for reason *name.* The tape request has been aborted. - USER

`TM215 - Invalid Message Routing Code =` *code*
A request was made to issue a message to an invalid destination. This is a system error. - USER/OPERATOR

`TM216 - unable to send` *operator message* `for` *user id jid*`,` *NQSid* `to front-end, '`*feid*`' (`*frontendid*`), (`*reason*`) reply cancel, retry, or ignore`
The tape daemon was unable to send a message to the front end. The operator should reply with `cancel` to abort the original request, `retry` to reissue the original request, or `ignore` to ignore the error condition. Multiple messages

are generated when a tape is being mounted. If the operator specifies ignore, it is assumed that the tape mount request will be satisfied as a result of one of the other messages issued. - OPERATOR

`TM217 - Scratch Volume Request denied.`
A request was made to mount a scratch tape to a loader that does not support the type of scratch tape specified in the request. The mount request has been terminated. - USER

`TM218 - Unable to` *action*`.`
The tape daemon was unable to perform the function specified with *action*. The request has been terminated. - USER

`TM219 - Specified Media Loader is busy, request failed`
A request to change the configuration of a loader cannot be completed if devices allocated to the loader have been assigned. The request has been terminated. - USER

`TM220 - Device must be down to change Media Loader`
The media loader for a device cannot be changed if the loader is not configured down. The `tpconfig`(8) request has been terminated. Configure the loader down, and reissue the loader change request. - USER

`TM221 - Associated Media Loader is down`
You attempted to change or configure up the media loader for a device, and the loader is not configured down. The `tpconfig`(8) request has been terminated. - USER

`TM222 - Volume` *volno* `scratched.`
A request to `scratch` volume *volno* succeeded. - USER

`TM223 - Volume` *volno* `scratch request failed: reason` *reason.*
A request to `scratch` a tape has failed. - USER

`TM224 - Loader Unavailable.`
A request was made to `scratch` a volume within the loader. This request could not be completed because the loader has been configured `down`. - USER

`TM225 - Invalid State for Loader Type.`
A request was made to configure a loader to a state that is invalid. The request has been terminated. - USER

`TM226 - `*vsn*` : Invalid vsn:` *vsn*

An invalid volume was specified on the `tpmnt`(1) command. The volume specification must be alphanumeric. - USER

`TM227 - `*info*` : network request failed:` *request*

The tape daemon was unable to issue a request to the front end. - USER/OPERATOR

`TM228 - `*path*` : waiting for vsn` *vsn*

A request was made to mount volume *vsn*. However, this volume is currently in use. The tape daemon will place the mount request in a waiting state, and reissue the request when the volume is free. - USER

`TM229 - vsn request rejected by user exit`

Your mount request was terminated because of your site's verification specifications. Check to see that you have permission to mount this volume. - USER

`TM230 - `*action*` < `*vsn*` >? Reply y(yes)/n(no)/q(requeue).`

Request that you `import` or `export` a volume. Reply `y` if you wish to import or export, `n` to abort the job that requested the volume, or `q` to queue the mount request. - OPERATOR

`TM231 - Reply y/n when import is complete.`

You replied `y` to an `import` or `export` request. When you complete the `import` or `export`, reply `y` to this message. - OPERATOR

`TM232 - Eject vsn (`*vsn*`) from loader (`*loader*`) and change ring? Reply y(yes) or n(no)`

The operator must respond either `y` if the system is to eject the specified VSN and change the state of the tape ring before returning the tape to the loader domain or `n` if the operator does not want these actions to occur. - OPERATOR

`TM233 - Volume` *volname* `Not Scratchable, enter retry / cancel`

The mounted volume cannot be made into a scratch tape. Reply `retry` to mount another scratch tape or `cancel` to cancel the tape mount request. - OPERATOR

`TM234 - Embedded tape marks are not allowed on volume` *vsn*

Embedded tape marks are not allowed on volume *vsn*. - USER

`TM235 - Microcode file not found for channel` *channo*`, iop` *iop*`, ios` *ios*

No microcode file was specified in the configuration file for channel *channo*, iop *iop*, ios *ios*. - OPERATOR

`TM236 - not used`

`TM237 - Density not valid with this device group`

A density was specified with the `-d` option of the `tpmnt`(1) command with a device group that does not allow different densities. - USER

`TM238 - is volume` *vsn* `on device` *dvn* `a valid scratch volume for user` *uname jid*`? reply y/n`

- OPERATOR

`TM239 - Device` *dvn* `does not exist`

A device name, *dvn*, was specified on a tape daemon command that does not exist. Correct the device name and reissue the command. - USER

`TM240 - cannot find host entry for sd=`*socket-descriptor*

The entry for *socket-descriptor* is set, indicating that it is expecting a reply from another mainframe. No entry has been queued within the `tcpnet()` function expecting such a reply. Check the network for proper functionality or contact your system support group. - OPERATOR

`TM241 - error from socket` *operation*`, sd=`*socket-descriptor*`, host=`*host*`, rc=`*rc*`, errno=`*errno*`: ` *error-description*

The error *error-description* has occurred while trying to do *operation* on *socket-descriptor* that is connected to *host*. Check the network or mainframe *host* for proper functionality or contact your system support group. - OPERATOR

`TM242 - host name` *host* `not found`

An attempt to obtain the network host entry for mainframe *host* from the file `/etc/hosts` or from the file `/etc/host.bin` has failed. Contact your system support group to correct the network files or to correct the tape configuration file. - OPERATOR

`TM243 - Invalid block size requested on tpmnt`

The maximum block size specified with the `-b` option of the `tpmnt`(1) command is 0 or exceeds the maximum specified in the configuration file.

Modify the maximum block size specified with the `-b` option of the `tpmnt`(1) command. - USER

`TM244 - The label write did not complete successfully`
The `tplabel`(1) command did not complete successfully. Check the `errno` and the `tape.msg` file to determine the cause of the failure. - USER

`TM245 - not used`

`TM246 - Only bypass label is valid with option -z`
A label type other than bypass label type was requested with the `-z` option. Reissue the `tpmnt`(1) command with `-l blp`. - USER

`TM247 - Option -`*option* `can only be specified for ER90 volumes`
The option *option* was specified on a tape command that is not valid for devices other than ER90 devices. Reissue the command without the option. - USER

`TM248 - A format id can only be specified for ER90 volumes`
A format identifier was specified on a command issued to a non-ER90 device. Reissue the command without the format identifier or issue the command to an ER90 device. - USER

`TM249 - A partition number can only be specified for ER90 volumes`
A partition number was specified on a command issued to a non-ER90 device. Reissue the command without the partition number or issue the command to an ER90 device. - USER

`TM250 - The block size for an ER90 device must be in increments of 8`
The user specified a block size, using the `-b` option on the `tpmnt`(1) command, which is not a multiple of eight. The `tpmnt`(1) command is terminated. Correct the block size to a multiple of eight and reissue the command. - USER

`TM251 - The block size for an ER90 device must be in the range` *min-block-size* `to` *max-block-size*
The user specified a block size, using the `-b` option on the `tpmnt`(1) command, which is not within the valid range for an ER90 device. The block size must be greater than *min-block-size* but cannot exceed *max-block-size*. The `tpmnt`(1) command is terminated. - USER

`TM252 - Cannot specify a non-numeric value,` *value*`, for option` *-option*

The user specified a nonnumeric value for option *-option*. Reissue the command specifying a numeric value for option *-option*. - USER/OPERATOR

`TM253 - The` *parameter* `specified,` *param*`, cannot exceed` *n* `characters`

An invalid parameter, *param*, was specified. The parameter cannot exceed *n* characters. Correct the parameter length and reissue the command. - USER/OPERATOR

`TM254 - Unable to complete the` *reqt* `request because of a system error (errno=`*errno*`)`

An unexpected system error occurred when processing request, *reqt*. Contact your system support staff. - USER/OPERATOR

`TM255 - Option` *option* `must be specified`

A required option, *option*, has not been specified. Reissue the command specifying this option. - USER/OPERATOR

`TM256 - The system zone spacing must be zero if no system zones were requested`

The user specified options `-z` and `-l` on the `tpformat`(8) command. These options are mutually exclusive. It is invalid to specify a length between system zones using the `-l` option. Correct the options and reissue the `tpformat`(8) command. - OPERATOR

`TM257 - The number of [`*A*`|`*B*`] partitions specified,` *num-of-partitions*`, must be in the range` *min-partitions* `to` *max-partitions*

The user specified a partition count, using the `-n` option of the `tpformat`(8) command, which is not in the range *min-partition* to *max-partition*. Correct the partition count and reissue the `tpformat`(8) command. - OPERATOR

`TM258 - If a single partition tape was requested ('A' size is zero), the 'A' count must be 1`

The user requested that a single partition tape be created by specifying zero on the `-s` option of the `tpformat`(8) command. The number of partitions specified, by using the `-n` option, must be one. Correct the option(s) and reissue the command. - OPERATOR

`TM259 - If a single partition tape was requested ('A' size is zero), the 'B' count and size must be 0`

The user requested that a single partition tape be created by specifying zero on the `-s` option of the `tpformat`(8) command. The number and size of the B partitions requested, by using the `-n` and `-s` options, must be zero. Correct the option(s) and reissue the command. - OPERATOR

`TM260 - The size of partition [`*A*`|`*B*`],` *part-size*`, must be in the range` *min-part-size* `to` *max-part-size*

The user specified a partition size, using the `-s` option of the `tpformat`(8) command, which is not within the valid range. Correct the size specified and reissue the command. - OPERATOR

`TM261 - A non-zero 'B' partition size must be specified if more than one 'B' partition is requested`

The user specified a value other than 1 for a B partition count when a B partition size of 0 has been specified. Modify the B partition size to a nonzero value or modify the B partition count to 1 and reissue the `tpformat`(8) command. - OPERATOR

`TM262 - The system zone spacing specified,` *spacing-length*`, must be in the range` *min-length* `to` *max-length*

The user specified an invalid system zone spacing value, using the `-l` option on the `tpformat`(8) command. The length specified must be in the range *min-length* to *max-length*. Correct the length specified and reissue the command. - OPERATOR

`TM263 - The syntax of the` *-opt* `option argument is incorrect`

The syntax of the value specified with the *-opt* option is invalid. Correct the syntax and reissue the command. - OPERATOR

`TM264 - A partition size cannot be specified if the number of partitions requested is zero`

The user specified a partition count of zero, using the `-n` option of the `tpformat`(8) command. Because no partitions were requested, it is invalid to also specify a partition size with the `-s` option. Correct the options and reissue the command. - OPERATOR

`TM265 - Option` *-opt* `cannot be specified for single partition volumes`

Option *-opt* cannot be specified if a single partition volume is requested on the `tpformat`(8) command. Reissue the command without option *-opt*. - OPERATOR

`TM266 - The 'A' partition size must be zero if the volume is to be created during write operations`

The user requested that the volume be formatted during write operations by specifying the `-q` option on the `tpformat`(8) command. Multiple partition tapes cannot be created during write operations. Reissue the `tpformat`(8) command with an A partition size of zero or without specifying the `-s` option. - OPERATOR

`TM267 - The specified partition size,` *partition-size*`, exceeds the tape length`

The ER90 device was unable to format even one partition, of size *partition-size*, on the volume. Specify a smaller partition size and reissue the command. - OPERATOR

`TM268 - Permission to format volume` *volume* `denied`

Permission to format ER90 volumes has been denied. The `tpformat`(8) command is terminated. Contact your system support staff to obtain permission to format ER90 volumes. - OPERATOR

`TM269 - A` *reqt* `request cannot be issued to an active device`

The *reqt* request was rejected because the device is active. The command is terminated. Notify your system support staff of the problem. - OPERATOR

`TM270 - An invalid parameter was specified on the` *reqt* `request`

An invalid parameter was specified on the *reqt* request. The command is terminated. Contact your system support staff for more information on the parameter in error. - OPERATOR

`TM271 - A volume format request is not valid for the requested device type`

A volume format request was issued to a device that does not support format requests. Format requests are only valid for ER90 devices. Reissue the `tpformat`(8) command to an ER90 device. - OPERATOR

`TM272 - Unable to issue the` *reqt* `request because of an existing device error`

Request *reqt* was rejected because an error occurred on the device on a previous request and has not been acknowledged. Check your `tape.msg` file for an error. If no error has occurred, contact your system support staff. - OPERATOR

`TM273 - Unable to issue the` *reqt* `request because the device has been cleared`

The *reqt* request was rejected because it has been issued to a device that has been cleared but not reset. The command is terminated. Reset the device by configuring the device down and then up again. Reissue the request. - USER/OPERATOR

`TM274 - Unable to complete the` *reqt* `request (errno =` *errno*`)`

An unexpected error, error *errno*, occurred when processing request, *reqt*. Contact your system support staff. - USER/OPERATOR

`TM275 - A sync must be issued before positioning when EOV processing is selected`

End-of-volume processing was selected, but not initiated. The user output data to tape and then issued a position request. The tape daemon requires that a synchronize request be issued on position requests issued to an output file if the user has selected but not initiated user end-of-volume processing. The position request is terminated. Modify the request sequence to issue a synchronize request before positioning. - USER

`TM276 - Cannot append to a blank tape`

The user attempted to append to a tape that is blank. The write request is terminated. Reissue the `tpmnt`(1) command without the `-a` option. - USER

`TM277 - Cannot read a blank tape`

The user issued a request to read a tape that is blank. The read request is terminated. - USER

`TM278 - Option` *-opt* `requires bypass label or tape manager permission`

The user specified an option, *-opt*, that requires bypass label or tape manager permission. The command is terminated. Contact your system support staff for permission to use the option. - USER

`TM279 - The` *parameter* `number specified,` *param*`, must be in the range` *min-value* `to` *max-value*

The user specified a value for *parameter* that is invalid. The value must in the range *min-value* to *max-value*. Correct the number specified and reissue the command. - OPERATOR

`TM280 - Partition` *partition* `does not exist`

The user requested a partition that does not exist. Check that the correct partition numbers and volumes were specified on the `tpmnt`(1) command and that the volumes have been formatted correctly. - USER

`TM281 - A volume id must be specified with option -v`

The user did not specify any VSN with option `-v`. Correct the `-v` parameter and reissue the command. - USER

`TM282 - An internal volume id must be specified`

The user specified an external VSN or format ID without also specifying an internal VSN (`-v =EXTID` or `-v ==FMTID`). Correct the VSN specified and reissue the command. - USER

`TM283 - A partition number must be specified after the '/' delimiter`

The user did not specify a partition number following the / delimiter on the `-v` option of the `tpmnt`(1) command. Correct the volume list specified and reissue the command. - USER

`TM284 - Extraneous characters,` *`string`*`, following the partition`

The user specified extraneous characters following the partition number. Correct the volume list specified and reissue the command. - USER

`TM285 - Cannot specify a non-numeric value` *`parameter`* `for a partition number`

The user specified a nonnumeric value, *parameter*, for a partition number. Specify a numeric value for all partition numbers and reissue the command. - USER

`TM286 - A volume description cannot begin with the '=' delimiter`

The user began the volume description with the = delimiter rather than with the internal volume ID. Correct the volume description and reissue the command. - USER

`TM287 - The` *`delim`* `delimiter cannot be specified after the format id`

The syntax of the volume description is incorrect. The user specified the *delim* delimiter after the format ID. Correct the volume description and reissue the request. - USER

`TM288 - An external volume id must be specified after the first '=' delimiter`

The syntax of the volume description is incorrect. A VSN must follow the = delimiter. Correct the volume description and reissue the request. - USER

`TM289 - A format id must be specified after the second '=' delimiter`

The syntax of the volume description is incorrect. Two = delimiters were specified but were not followed by a format ID. Correct the syntax of the volume description and reissue the request. - USER

`TM290 - A partition number must follow the '//' delimiter`

`TM291 - The absolute track address contains a value that is not within the valid range`

The user specified an invalid absolute track address. Correct the address so that each value comprising the address is within the valid range and reissue the request. - USER

`TM292 - Permission to position to the requested tape address denied`

The user does not have permission to position to the requested tape address. The user must have tape manager privilege to position outside the current partition or bypass label privilege to position outside the current file. Special permission is not required to position within the current file. Contact your system support staff for privilege required. - USER

`TM293 - Unable to obtain the user permission bits for user` *user*

The tape daemon was unable to validate the user's request by checking the User Database (UDB). The request is terminated. Contact your system support staff. - USER

`TM294 - The` *reqt* `request has been interrupted`

The *reqt* request has been interrupted. - USER

`TM295 - The absolute track address specified is invalid`

The user specified an absolute track address that does not exist on the tape. Correct the address specified or the volume specified on the `tpmnt`(1) command and reissue the position request. - USER

`TM296 - The` *cmd* `command requires bypass label or tape manager permission`

The user does not have permission to use the *cmd* command. The command is terminated. Contact your system support staff for the privilege required to use this command. - USER

`TM297 - Unable to complete the` *reqt* `request within the device timeout period specified in the configuration file`

The *reqt* request did not complete within the time-out period specified in the configuration file. Either the time-out value specified in the configuration is not long enough to allow the request to complete or a problem with the device is preventing the completion of the request. Check the device or contact your administrator to change the configuration file. - USER/OPERATOR

`TM298 - Cannot specify a negative value for the` *parameter*

An invalid parameter, *parameter*, was specified. The parameter cannot be negative. The command is terminated. Correct the parameter and reissue the command. - USER/OPERATOR

`TM299 - Unable to complete the` *cmd* `command (errno =` *errno*`)`

An unexpected error occurred when processing command *cmd*. Contact your system support staff. - USER/OPERATOR

`TM300 - CRL vsn/vid` *string* `invalid on scratch request`

You have used the `-o` option of the `tpmnt`(1) command to specify volume identifier *string*. However, the volume specified by *string* has been cataloged to be used as a scratch tape. Check to see that the VSN/volume identifier specified is correct. - USER

`TM301 - not used`

`TM302 - CRL number of vid's exceeds maximum -` *maxvids*

The number of VSN/volume identifiers in your list exceeds the maximum, *maxvids*. Either specify fewer VSN/volume identifiers or ask your administrator to change the number of VSN/volume identifiers allowed. - USER

`TM303 - CRL invalid vid:` *vid*

The VSN/volume identifier *vid* contains illegal characters. The VSN/volume identifier specification must be alphanumeric. - USER

`TM304 - CRL` *string*`: Cray/REELlibrarian enabled`

Front-end servicing options specified by *string* are invalid for a system with Cray/REELlibrarian enabled. Disable these specifications in the tape configuration file. - OPERATOR

`TM305 - CRL invalid option -X: Cray/REELlibrarian disabled`
The `-X` option was specified for a system which does not contain the Cray/REELlibrarian product. Remove the `-X` option from the `tpmnt`(1) command. - USER

`TM306 - CRL volume set allocate error` *errno* `for` *volset*
The error *errno* was issued by the Cray/REELlibrarian daemon when it attempted to allocate a volume for the volume set *volset*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM307 - CRL ambiguous CRL filename:` *filename*
The file name *filename* you specified does not identify a unique file. Correct the file name on the `tpmnt`(1) command. - USER

`TM308 - CRL Volume record error` *errno* `for VID =` *vid*
The error *errno* was received from the Cray/REELlibrarian daemon when it attempted to read the volume record for *vid*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM309 - CRL Volume record error` *errno* `for volset =` *volset*
The error number *errno* was received from the Cray/REELlibrarian daemon when it attempted to read the volume set record for *volset*. Contact your Cray/REELlibrarian administrator for further information. - USER

`TM310 - CRL` *parameter* `mismatch: (fit->)`*string1*`, (frec->)`*string2*
A mismatch was encountered in the parameter *parameter* of the file record. The value was specified as *string1*; the value retrieved from the file record is *string2*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM311 - CRL` *parameter* `mismatch: (fit->)`*string1*`, (frec->)`*string2*
A mismatch was encountered in the parameter *parameter* of the file record. The value was specified as *string1*; the value retrieved from the file record is *string2*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM312 - CRL volume` *vid* `is not mountable from location` *location*
A tape mount request was made for a volume that is classified as not mountable. The volume is not accessible for use. - USER

`TM313 - CRL` *type* `permission denied for` *name*

You do not have permission to access the *type* (file or volume) identified by *name*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM314 - CRL` *type* `password error for` *name*

You did not specify the correct password to access the *type* (file or volume) identified by *name*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM315 - CRL` *filename* `must be file sequence 1`

You specified an invalid sequence number for the file *filename*. Correct the sequence number specification on the `tpmnt`(1) command. - USER

`TM316 - CRL Volume list failure` *errno* `for` *volset*

The error *errno* was issued by the Cray/REELlibrarian daemon when it attempted to read the volume set volume list for *volset*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM317 - CRL invalid` *type* `mode:` *mode*

You specified an invalid access mode for the *type* (file or volume). The mode should be three octal digits between 000 and 777. - USER

`TM318 - CRL owner name too long -` *name*

The owner name *name* exceeds the owner name length maximum specified in the Volume Management Facility catalog. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM319 - CRL invalid generation number -` *generation*

You specified a generation number, *generation*, on a `tpmnt`(1) command file specification that contains nonnumeric characters. Correct the specification and reissue the `tpmnt`(1) command. - USER

`TM320 - CRL invalid version number -` *version*

You specified a version number, *version*, on a `tpmnt`(1) command file specification that contains nonnumeric characters. Correct the specification and reissue the `tpmnt`(1) command. - USER

`TM321 - CRL invalid section number -` *section*

You specified a section number, *section*, on a `tpmnt`(1) command file specification that contains nonnumeric characters. Correct the specification and reissue the `tpmnt`(1) command. - USER

`TM322 - CRL invalid FID character -` *char*

You specified an invalid character, *char*, on a `tpmnt`(1) command file specification. This character should be either `g`, `G`, `v`, `V`, `s`, or `S`. Correct the specification and reissue the `tpmnt`(1) command. - USER

`TM323 - file read error for volset` *vname*`, fseq` *fseq*`, fsect` *fsect*

In a secure environment, the attempt to reread the file record for MAC revocation failed. Contact the system administrator with the error output for resolution. - USER

`TM324 - CRL unable to get` *type* `name for id` *id*

The *type* (user or group) name for ID *id* exceeds the user or group name maximum allowed in the Cray/REELlibrarian catalog. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM325 - CRL record format conversion error for` *format attribute*

An error occurred trying to convert the fit record format (*format*) and block attribute (*attribute*) to Cray/REELlibrarian format. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM326 - CRL file record update error on` *filename*

An error was encountered by the Cray/REELlibrarian daemon when it attempted to update the file record for *filename*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM327 - CRL volume record update error on` *volset*

An error was encountered by the Cray/REELlibrarian daemon when it attempted to update the volume record for *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM328 - CRL vid=vsn, do not specify both`

You specified both the `vsn` and `vid` options on the `tpmnt`(1) command. On this Cray Research computer system, the VSN is identical to the volume identifier. Reissue the `tpmnt`(1) command with only one of the options specified. - USER

`TM329 - CRL fit/vrec vsn mismatch:` *fitvsn*`,` *vrecvsn*

A mismatch was encountered between the mounted VSN/volume identifier (*fitvsn*) and that returned in the volume record (*vrecvsn*). Contact the Cray/REELlibrarian administrator for further information. - USER

`TM330 - CRL request/vrec` *type* `mismatch:` *fitvsn*`,` *vrecvsn*
A mismatch was encountered between the *type* (VSN/volume identifier) specified by the user (*fitvsn*) and that returned in the volume record (*vrecvsn*). Contact the Cray/REELlibrarian administrator for further information. - USER

`TM331 - CRL vlock error` *errno* `for vid` *vid*
The Cray/REELlibrarian daemon encountered error *errno* while attempting to lock VSN/volume identifier *vid*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM332 - CRL lock operation` *type* `valid for vid` *vid*
The *type* (lock or unlock) request was successful for the VSN/volume identifier *vid*. This is an informational message. - USER

`TM333 - CRL palloc error` *errno* `for nvol` *nvol*`, vsid =` *vsid*
The error *errno* was encountered by the Cray/REELlibrarian daemon when it attempted to allocate a new volume for the volume set that has a first VSN/volume identifier of *vsid*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM334 - CRL file sequence mismatch: request =` *fitseq*`, frec =` *frecseq*
A mismatch was encountered by the Cray/REELlibrarian daemon in the file sequence number between the fit (*fitseq*) and the file record (*frecseq*). Contact the Cray/REELlibrarian administrator for more information. - USER

`TM335 - CRL end of file record failed for vsid =` *vsid*
The Cray/REELlibrarian daemon encountered an error while attempting to write an end-of-file record for the volume set that has a first VSN/volume identifier of *vsid*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM336 - CRL file section` *section* `record missing for` *filename*
The Cray/REELlibrarian daemon could not retrieve file section *section* record for the file *filename*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM337 - CRL file record not written for` *filename*

The file record for *filename* was not written to the catalog because of the errors encountered during its processing. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM338 - CRL created file section` *section* `record for` *filename*
The Cray/REELlibrarian daemon created file section *section* record for file *filename*. This is an informational message. - USER

`TM339 - CRL flist call failed for volset` *volset*
The Cray/REELlibrarian daemon could not retrieve the file list for volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM340 - CRL sequence error:` *seq1* `to` *seq2* `on` *volset*
The Cray/REELlibrarian daemon encountered missing sequence numbers for the volume set *volset*. The missing sequence numbers are between *seq1* and *seq2*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM341 - CRL section error: fseq` *sequence* `1st section is` *section* `on` *volset*
The Cray/REELlibrarian daemon encountered file sequence *sequence* on volume set *volset*, which has a first section of *section* rather than 1. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM342 - CRL section error:` *sequence* `to` *section* `on` *volset*
The Cray/REELlibrarian daemon encountered a missing file section record for file sequence *sequence* on volume set *volset* at section *section*. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM343 - CRL fseq` *sequence* `end-of-status, not last file on` *volset*
The Cray/REELlibrarian daemon returned a file list, which contained an end-of-list record that was not the last record in the list. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM344 - CRL no EOT/EOL found for non-empty volset` *volset*
The Cray/REELlibrarian daemon returned a file list, for volume set *volset*, which did not contain an end-of-list record. Contact the Cray/REELlibrarian administrator for further information. - USER

`TM345 - CRL hdr1` *type val1* `found looking for` *type val2* `in volset` *volset*

A mismatch was encountered in *type* (sequence or section) between the expected value (*val1*) and the value in the file record (*val2*) for volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM346 - CRL file seq gap` *seq1* `to` *seq2* `for volset` *volset*

The Cray/REELlibrarian daemon encountered missing file sequence records for volume set *volset*. The missing sequence numbers are between *seq1* and *seq2*. Contact the Cray/REELlibrarian administrator for more information. - USER.

`TM347 - CRL addfrec for seq` *sequence* `failed for volset` *volset*

The Cray/REELlibrarian daemon encountered an error while attempting to add file sequence record *sequence* for volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM348 - CRL expdate conversion error for fseq` *sequence* `in volset` *volset*

An error occurred converting the expiration date for file sequence *sequence* in volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM349 - CRL fid mismatch: hdr1 =` *fid1*`, frec =` *fid2*

A file ID mismatch was encountered between the header1 label file ID (*fid1*) and the file record file ID (*fid2*). Contact the Cray/REELlibrarian administrator for more information. - USER

`TM350 - CRL` *type* `mismatch: hdr1 =` *val1*`, frec =` *val2*

A mismatch was encountered in the *type* (generation or version) field between the header1 label (*val1*) and the file record (*val2*). Contact the Cray/REELlibrarian administrator for more information. - USER

`TM351 - CRL volset` *volset* `at EOL, file destruction possible, no write`

The Cray/REELlibrarian daemon does not have enough information about volume set *volset* to allow your `write` request. Usually, this error occurs for imported volume sets. The user should read the entire volume set to end-of-data to allow the daemon to complete its file list, or, if the `write` request is absolutely necessary, to completely disable file tracking for the volume set. - USER

`TM352 - CRL volset` *volset* `flags neither EOL or EOT`

Volume set *volset* contains no end-of-list record. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM353 - CRL write would destroy unexpired files on volset` *volset*

The `write` operation you requested will destroy unexpired files that exist on the volume set *volset*. - USER

`TM354 - CRL scratch volset` *volset* `has unexpired file sequence` *sequence*

The scratch volume set *volset* contains an unexpired file at sequence number *sequence*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM355 - CRL missing section` *section* `for fseq` *sequence* `on volset` *volset*

The Cray/REELlibrarian daemon has encountered missing file section record *section* for file sequence *sequence* on volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM356 - CRL eot at fseq` *seq1* `instead of fseq` *seq2* `for volset` *volset*

The Cray/REELlibrarian daemon encountered the end-of-tape list at sequence number *seq1* instead of *seq2* for volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM357 - CRL file sequence` *sequence* `section` *section* `error on volset` *volset*

While adding a new file record, the Cray/REELlibrarian daemon encountered file section *section* for file sequence *sequence*, which was not the last sequence number before the end-of-list record. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM358 - CRL hdr1 missing for old file` *filename* `on volset` *volset*

No header1 label was found for file *filename* accessed by the `-o` option of the `tpmnt`(1) command for volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM359 - CRL file name syntax error for file id` *fid*

The file ID specified on the `tpmnt`(1) command contained incorrect CRL file ID syntax. Correct the parameter specification and reissue the command. - USER

`TM360 - CRL volume record not found for -o file id` *filename*

The Cray/REELlibrarian daemon was unable to retrieve the volume record for file ID *filename*, which should exist. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM361 - CRL NOFID status in volset` *volset*`, fseq` *sequence* `is not PSEUDO`

The Cray/REELlibrarian daemon encountered a file record for file sequence *sequence* in volume set *volset*, which had the `NOFID` status enabled, but did not have the `PSEUDO` status enabled. This indicates that an unlabeled tape may have been encountered. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM362 - CRL file id` *filename* `not in catalog`

The Cray/REELlibrarian daemon was unable to retrieve the file record for *filename* from the catalog. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM363 - CRL file id` *fid* `already exists in catalog`

You attempted to access file *fid* with the `-n` option of the `tpmnt`(1) command, but this file record already exists. - USER

`TM364 - CRL end of file record (fseq =` *sequence*`) on volset` *volset*

The Cray/REELlibrarian daemon encountered an error while attempting to read the end-of-list file record at file sequence *sequence* for volume set *volset*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM365 - CRL` *command* `failed for` *parameter*`, terrno =` *errno*

The Cray/REELlibrarian command *command* failed for parameter *parameter* with error number *errno*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM366 - CRL volume set name mismatch: vrec =` *vsname1*`, tpmnt =` *vsname2*

The user specified a volume set name with the `-S tpmnt`(1) option and a list of VSNs (or volume identifiers) with the `-v (-j)` option that are members of a different volume set. Correct either the `-S` or `-v (-j)` parameter and reissue the command. - USER

`TM367 - CRL can't normalize volume set name` *vsname*

The user specified a volume set name with the `-S tpmnt`(1) option that CRL cannot normalize due to syntactical errors. Correct the parameter and reissue the command. - USER

`TM368 - CRL and FES cannot both be enabled`
You attempted to enable both Cray/REELlibrarian and front-end servicing by using the `tpset`(8) command. - OPERATOR

`TM369 - CRL Undefined or ambiguous vsn reference:` *vsn*
The Cray/REELlibrarian daemon was unable to retrieve the volume record for *vsn*. Either *vsn* does not exist, or more than one *vsn* volume exists. - USER

`TM370 - CRL 1st VUX not sent for veronly of vid` *vid*
The `verify only` volume update request made by the tape daemon did not have the correct status set for vid *vid*. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM371 - CRL does not have a unique scratch volume with vsn` *vsn*
While attempting to read the volume record for *vsn* to extend a volume set, the Cray/REELlibrarian daemon encountered an error other than `unknown vsn`. Because *vsn* is not a scratch volume, the mount request will be rejected, and the mount message will be reissued. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM372 - CRL file id mismatch: request =` *x*`, frec =` *X*

`TM373 - CRL CRL_NEW_VREC not set for vmf_dex recall`
You specified a volume record to be read. No volume record was found, nor was the volume record read flag set. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM374 - CRL vmf_dex recall error`
An error was encountered while attempting to access the volume record for a scratch CRL submission. Contact the Cray/REELlibrarian administrator for more information. - USER

`TM375 - CRL using VID` *vid* `for ambiguous VSN` *vsn* `in mount msg`
The *vsn* requested to be mounted is not unique in CRL. The volume ID (*vid*) will be used in the operator mount message so the ambiguity is resolved. This is informational. - USER

`TM376 - CRL VID` *vid* `has ambiguous vsn and is longer than 6 characters`

The external VSN field is only 6 characters. In attempting to use the volume identifier to resolve ambiguity, it was discovered that the *vid* is too long to use as the external VSN. Contact the CRL administrator to resolve the VSN ambiguity or specify an appropriate external VSN. - USER

`TM377 - CRL permission denied for volset` *object name*

The user does not have access permission to the requested object (file, volume, or volume set). - USER

`TM378 - CRL ambiguous filename creation request for` *filename*

A new file creation request was made that could not be completed because the file name already exists in the catalog. Change the file name specification and reissue the command. - USER

`TM397 - CRL no vsn list for -y Z bypass request`

The user has requested to bypass CRL processing but failed to supply a list of VSNs/volume identifiers to be checked for catalog existence. Add the `-v` option to the `tpmnt`(1) request and reissue the command. - USER

`TM398 - CRL found vsn` *vsn* `for -y Z request`

The user has requested to bypass CRL processing but specified a VSN/volume identifier that existed in the catalog, causing the request to fail. Re-specify the VSN/volume identifier list and reissue the command. - USER

`TM399 - Bypassing CRL processing for -y Z request`

CRL processing is being bypassed as requested. This is informational. - USER

`TM400 -` *program data-structure-1* `V` *version-number-1* `is incompatible with system` *data-structure-2* `V` *version-number-2*`, abort` *program*

The *program* name has been created with *data-structure-1* name version *version-number-1*. This is incompatible with the *data-structure-2* name version *version-number-2* that was used to build the tape daemon. Contact your system support group to change either one of the components to create a matching set. - OPERATOR

`TM401 -` *structure name*`:` *text* `discrepency: expected:` *number*`; received:` *number*`.`

structure name is one of the following: `chiolh` or `chiole`. *text* is one of the following: `quantity` or `size`. *number* is a decimal number. When the tape

daemon starts a child process, it delivers a data structure to this process. The data structure contains the number of data structures that will follow and the size of each structure. This message indicates that there is a discrepancy either between the number delivered and the number the child process has calculated or between the delivered size and the calculated size of one or more data structures. Contact your system support group to resolve the discrepancy. - OPERATOR

`TM402 - not used`

`TM403 - Unable to stop IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

`TM404 - Unable to start IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

`TM405 - Configuration Table Error: invalid value for` *parameter* `=` *entry*

Configuration file table error with *parameter*; entry *entry* is incorrect. - OPERATOR

`TM406 - Unable to add channel` *channel* `to IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

The tape subsystem was unable to complete the configuration of the tape subsystem because it was unable to add channel *channel* to IOP *iop*, cluster *cluster*; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM407 - Unable to add bank` *bank* `to IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

The tape subsystem was unable to complete the configuration of the tape subsystem because it was unable to add bank *bank* to IOP *iop*, cluster *cluster*; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM408 - Unable to add slave` *slave* `to IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

The tape subsystem was unable to complete the configuration of the tape subsystem because it was unable to add slave *slave* to IOP *iop*, cluster *cluster*; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM409 - Unable to add device` *device* `to IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

The tape subsystem was unable to complete the configuration of the tape subsystem because it was unable to add device *device* to IOP *iop*, cluster *cluster*; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM410 - User Exit Function :` *function* `: returned :` *return code*

The user exit function *function* returned *return code*. This is a site-defined message. - USER

`TM411 - Unable to add channel pair` *value* `to IOP` *iop*`, cluster` *cluster* `(errno =` *errno*`)`

The tape subsystem was unable to complete the configuration of the tape subsystem because it was unable to add channel pair *value* to IOP *iop*, cluster *cluster*; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM412 - TM416 - not used`

`TM417 - file` *file*`, line` *line* `at "text", offset` *offset*`: Parameter` *param* `is invalid for the IOP type`

The parameter, *param*, is not valid for an IOP of the type being defined. Correct the parameter and reissue the command. - OPERATOR

`TM418 - file` *file*`, line` *line* `at "`*text*`", offset` *offset*`: Channel pair` *channel1*`:`*channel2* `has not been defined`

The channel pair, *channel1:channel2*, has not been defined. Define the channel pair and reissue the command. - OPERATOR

`TM419 - file` *file*`, line` *line* `at "`*text*`", offset` *offset*`: The input and output channels must be unique`

The same value was specified for both the input channel and output channel defined for a slave. Unique values must be specified. Correct the channel pair values and reissue the command. - OPERATOR

`TM420 - file` *file*`, line` *line* `at "`*text*`", offset` *offset*`:` *value* `must be in the range of` *lowvalue* `to` *highvalue*

A value was specified that is not in the range *lowvalue* to *highvalue*. Correct the value and reissue the command. - OPERATOR

`TM421 - vendor address needs to be specified for autoloader tape drive:` *device name*

Specify the autoloader address of the tape drive with the device name specified by the vendor supplied software for that autoloader. For a STK autoloader this

address is in the format (*acs,lsm,panel,drive*), and for the EMASS autoloader this address is a single number. Contact your system support group to specify the correct autoloader address for drive name. - OPERATOR

`TM422 - line` *line-number* `at "`*char-string*`", offset` *char-offset*`: vendor address:` *value1* `can not be <` *value2*

A value is specified for *value1* in the vendor address that is less than the minimum boundary, *value2*, that can be specified for *value1* in the vendor address. Contact your system support group to change the value for *value1* to a number that is larger than *value2*. - OPERATOR

`TM423 - line` *line-number* `at "`*char-string*`", offset` *char-offset*`: vendor address:` *value1* `can not be >` *value2*

A value is specified for *value1* in the vendor address that is greater than the maximum boundary, *value2*, that can be specified for *value1* in the vendor address. Contact your system support group to change the value for *value1* to a number that is less than *value2*. - OPERATOR

`TM424 - line` *line-number* `at "`*char-string*`", offset` *char-offset*`: STK vendor address format: (acs,lsm,panel,drive)`

The STK autoloader vendor address has not been specified correctly: more values were encountered than the four making up the STK autoloader vendor address. Contact your system support group to specify a correct STK autoloader address. - OPERATOR

`TM425 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`:` *text*

Correct the error, *text*, and run again - OPERATOR

`TM426 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: too many` *items*`, max =` *number*

More than *number* of *items* were specified. Reduce the number of *items* and reissue the command - OPERATOR

`TM427 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: invalid server name`

The server name specified by *char-string* is invalid. Specify a correct server name and reissue the command. - OPERATOR

`TM428 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: '`*keyword-parameter* `=' required in preceeding statement`

The *keyword-parameter* is required in the preceding statement. Specify the keyword parameter and reissue the command. - OPERATOR

`TM429 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: name too long, max =` *number*

The name *char-string* is too long. The maximum length is *number*. Reduce length of name and reissue the command. - OPERATOR

`TM430 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: already defined`

The parameter *char-string* has already been defined. Remove this instance of the parameter and reissue the command. - OPERATOR

`TM431 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: must specify full pathname`

The *char-string* is expected to be a full path name and it is not. Specify a full path name and reissue the command. - OPERATOR

`TM432 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: invalid device id, max =` *number*

The device ID specified by *char-string* is greater than *number*. Specify a device ID less than or equal to *number*. - OPERATOR

`TM433 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`:` *item* `not defined`

The item specified by *item* has not been defined. Add the definition of *item*. - OPERATOR

`TM434 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: servicing frontend and cray reel librarian are mutually exclusive`

Servicing front end and Cray/REELlibrarian may not be active at the same time. Select one or the other. - OPERATOR

`TM435 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: incorrect length, expected length =` *number*

The length of *char-string* is not *number*. Re-specify the *char-string*. - OPERATOR

`TM436 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: incorrect message type`

Message type *char-string* is not correct. Specify correct message type. - OPERATOR

`TM437 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: cannot locate loader '`*loader*`'`
The loader *loader* is not defined. Define *loader* before referencing it. - OPERATOR

`TM438 - binary config file "`*file*`" not created because of previous error`
The binary file *file* is not created because of previous error. Correct previous error and reissue the command. - OPERATOR

`TM439 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: controller on the wrong type of iop`
The controller defined by *char-string* cannot be attached to the IOP being processed. Check the configuration file and the hardware connection to make sure that the hardware configuration is correct. - OPERATOR

`TM440 - file` *config-file-name*`, line` *line-number* `at "`*char-string*`", offset` *char-offset*`: unknown iop type`
The IOP type specified is unknown to the tape system. - OPERATOR

`TM441 - binary tape config record version mis_match. Use correct version of tpconf to rebuild binary config file.`
The binary configuration file version does not match that of the tape demon. Rebuild the binary configuration file using the `tpconf`(8) command and restart tape demon. - OPERATOR

`TM442 - Cannot configure a device of this type`
The device specified in the `tpconfig`(8) command refers to a diagnostic device that cannot be configured up or down. - OPERATOR

`TM443 - maximum device exceeded, only` *number* `out of` *max* `devices defined in "`*file*`" are configured`
The number of devices defined in the configuration file, *file*, is greater than the number, *max*, specified by the `TAPE_MAX_DEV` parameter in the boot parameter file. Only the first *number* devices are configured in the tape system. Increase the value of `TAPE_MAX_DEV` or decrease the number of devices defined or do both. - OPERATOR

`TM444 - can't stat file` *filename*`, errno =` *errno*

`TM445 - file` *config-file*`, line` *line-number* `at "`*text*`", offset` *off-set* `: invalid channel address`

Channel address *text* in the configuration file *config-file*, line *line-number*, character *offset*, is invalid. Acceptable values are 030, 032, 034, and 036. - USER

`TM446 - file` *config-file*`, line` *line-number* `at "`*text*`", offset` *off-set* `: bank number already defined`

Bank number *text* in the configuration file *config-file*, line *line-number*, character *offset*, has already been specified or used. Specify a unique bank number. - USER

`TM447 - file %s, line %d at "%s", offset %d : invalid bank number"`

`TM448 - file` *config-file*`, option at line` *line-number* `ignored - this option has been replaced by the permbit "PERMBITS_BYPASSLABEL" in the udb.`

The option at line *line-number* in the configuration file *config-file* is no longer used in the tape daemon. It has been replaced by the permbit `PERMBITS_BYPASSLABEL` in the user database. - OPERATOR

`TM449 - file` *config-file*`, line` *number* `at` *text*`, offset` *number*`: iscp interface no longer supported`

In the tape configuration file called *config-file* at line *number* is the following text that describes the USCP option: *text*. *number* is the offset in the text.

This message informs the user that the USCP references in the configuration file are not longer valid. - USER

`TM450 - The` *string1* `command is only valid for` *string2* `devices`

A command was issued to a device not supporting that command. Recheck your command or device and reissue. - USER

`TM451 - Density` *value* `is not valid for device type` *type*

A density value *value* was specified with the `-i` option of the `tpmnt`(1) command. A density cannot be specified for the device type *type*. - USER

`TM452 - Release and reissue rsv for the correct device type. See tape message file for correct device type`

A request was issued that is not valid for the device type. Release the current tape resource and reissue the `rsv`(1) command, specifying a device type that is valid for the requests that are to be issued. - USER

`TM453 - The -D option is not valid for a device with avr`

The `-D` option of the `tpmnt`(1) command was used to request that a volume be mounted on an AVR device. It is not valid to specify an AVR device with the `-D` option. - USER

`TM454 - Asynchronous event "`*event*`" received on channel` *channel*`, control unit` *control unit*

`TM455 - Channel` *channel* `configured down due to asynchronous event "`*event*`"`

`TM456 - Control Unit` *control unit* `configured down due to asynchronous event` *event*

`TM457 - Device` *device* `configured down due to` *event* `event`

`TM458 - WARNING. file` *config-file*`, device` *device-name*`: channel` *octal-channel-number* `specified as DOWN: control unit (`*octal-channel-number*`,` *decimal-controller-address*`) downed.`

A conflict has been detected within tape configuration file *config-file*, such that the status of the channel with channel number *octal-channel-number* for a channel/control unit combination has been specified as `DOWN`, but the status of the control unit with control unit address *octal-channel-number*, *decimal-controller-address* has been specified as `UP`. This combination prompts the software to configure *octal-channel-number*, *decimal-controller-address* as `DOWN`. - OPERATOR

`TM459 - WARNING. file` *config-file*`, device` *device-name*`:` *text* `DOWN: device` *device-name* `downed.`

A conflict has been detected within tape configuration file *config-file*, such that the status of tape device with device name *device-name* has been specified as `UP`, while the status of all channels that have been specified in the *config-file* and all control units that have been specified in the `config-file` for this tape device have a status of `DOWN`.

The status of these components being `DOWN` could be the result of the software having configured channels `DOWN` or that tape configuration file *config-file* has these components specified with a status of `DOWN`. As a result, the software has configured the tape device with device name *device-name* `DOWN`. The variable *text* could read *its channel is* or *all its channels are.* The choice is dependent on whether one channel or all channels for a tape drive have been configured `DOWN`. - OPERATOR

TM460 - file *file*, **line** *line* **at "***text***", offset** *offset*: **invalid** *address* **address, 0***xXX* **- valid range is** *low* **-** *high*

TM461 - file *file*, **line** *line* **: previous control unit has an invalid protocol for corresponding channel adaptor or IOP type**

TM462 - file *file*, **line** *line* **: previous device has an invalid type for corresponding channel adaptor or IOP type**

TM463 - file *file*, **line** *line* **: previous channel adaptor and IOP type combination are not valid**

TM464 - Process *process* **aborted with error** *error*

TM465 - Device must be down to change Device Group Name.

The device cannot be reassigned a device group name unless it is in a down state. The **tpconfig**(8) request has been terminated. Configure the device down and reissue the command. - USER

TM466 - The tape daemon is active or the tape subsystem is being configured

A **tpdaemon**(8) or **tpinit**(8) command did not complete because either the tape daemon is already active or a previous **tpinit**(8) command is pending. Wait for the tape subsystem configuration processing to complete or stop the tape daemon with the **tpdstop**(8) command. - OPERATOR

TM467 - Unable to open the file, /dev/bmxdem, needed to configure the tape subsystem (errno = *errno*)

A **tpdaemon**(8) or **tpinit**(8) command cannot open the file used to communicate the tape subsystem configuration to the kernel and I/O processors; the error returned is *errno.* Contact your system support staff. - OPERATOR

TM468 - Unable to remove the old tape device files from directory *directory* **(errno =** *errno*)

A **tpdaemon**(8) or **tpinit**(8) command cannot remove the old tape device files found in the *directory* directory; the error returned is *errno.* Contact your system support staff. - OPERATOR

TM469 - Unable to remove directory *directory (errno = errno)*

The *directory* directory could not be removed; the error returned is *errno*. Check to see whether you have the correct permissions for removing the *directory* directory. - OPERATOR

`TM470 - Unable to modify the permissions for [file|directory]` *file*|*directory* `(errno =` *errno*`)`
The permissions of a file or directory could not be modified; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM471 - Exceeded the maximum number of tape devices allowed`
The number of tape device files defined in the tape configuration file exceeds the limit specified in the boot parameter file with the `TAPE_MAX_DEV` parameter. This parameter must specify a value that is the sum of all real tape devices, the number of tape I/O processors (IOPs), and the total number of channels. Either increase the value specified with the `TAPE_MAX_DEV` parameter and reboot your system or remove IOP, channel, or device definitions from your tape configuration file. - OPERATOR

`TM472 - Unable to obtain the status of file` *file* `(errno =` *errno*`)`
The status of the *file* file could not be obtained; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM473 - File` *file* `has an invalid length of` *length*
The length of the *file* file is not valid. Contact your system support staff. - OPERATOR

`TM474 - Cannot configure [control unit|slave]` *cu* `[up|down] on channel` *channel*`, cluster` *cluster* `iop` *iop*`, because there is no path to the [control unit|slave]`
A request to modify the state of the control unit or slave, *cu*, configured on the *channel* channel, the *cluster* cluster *iop* iop, failed because there is no path to the control unit or slave. A path exists if the channel in which the control unit or slave is attached is configured up. Configure channel *channel* up by using command `tpconfig`(8) or specify the channel state as `UP` in the tape configuration file before configuring the tape subsystem. - OPERATOR

`TM475 - Unable to configure [control unit|slave]` *cu* `[up|down] on channel` *channel* `cluster` *cluster* `iop` *iop* `(errno =` *errno*`)`

A request to modify the state of the control unit or slave, *cu*, configured on the *channel* channel, the *cluster* cluster *iop* iop failed; the error returned is *errno.* Contact your system support staff. - OPERATOR

`TM476 - A` *error-description* `error was detected on the request to configure [control unit|slave]` *cu* `[up|down] on channel` *channel* `cluster` *cluster* `iop` *iop*

A *error-description* error was detected on the request to modify the state of the control unit or slave, *cu*, configured on the *channel* channel, *cluster* cluster *iop* iop. Contact your system support staff. - OPERATOR

`TM477 - Cannot configure [control unit|slave]` *cu* `[up|down] on channel` *channel*`, cluster` *cluster* `iop` *iop*`, because the [control unit|slave] or path is` *error-description*

A request to modify the state of the control unit or slave, *cu*, configured on the *channel* channel, the *cluster* cluster *iop* iop failed because there is a hardware problem, described by *error-description*, with the control unit or slave or with the path to the control unit or slave. Contact your system support staff. - OPERATOR

`TM478 - Unable to configure channel` *channel* `[up|down] on cluster` *cluster* `iop` *iop* `(errno =` *errno*`)`

A request to modify the state of the *channel* channel, on the *cluster* cluster *iop* iop failed; the error returned is *errno.* Contact your system support staff. - OPERATOR

`TM479 - Cannot configure channel` *channel* `[up|down] on cluster` *cluster* `iop` *iop* `because the channel is` *error-description*

A request to modify the state of the *channel* channel configured on *cluster* cluster *iop* iop failed because there is a channel hardware problem described by *error-description.* Contact your system support staff. - OPERATOR

`TM480 - A` *error-description* `error was detected on the request to configure [control unit|slave]` *cu* `[up|down] on channel` *channel* `cluster` *cluster* `iop` *iop*

A *error-description* error was detected on the request to modify the state of the *channel* channel on *cluster* cluster *iop* iop. Contact your system support staff. - OPERATOR

`TM481 - Cannot configure channel` *channel* `down on cluster` *cluster* `iop` *iop* `because devices attached to the channel are configured up`

A request to configure the *channel* channel down on the *cluster* cluster *iop* iop failed because a control unit or device attached to the channel is configured up. Configure down all control units and devices that are attached to the channel and retry the request. - OPERATOR

`TM482 - An invalid cluster or iop number was specified in the configuration file`

Either the I/O processor (IOP) number specified in the tape configuration file is not within the valid range for IOP numbers, 0 through 4, or the cluster number specified is less than 0 or does not exist on the booted system. Correct the cluster or IOP number specified and reissue your request. - OPERATOR

`TM483 - The tape subsystem must be configured before starting the tape daemon`

The `tpdaemon`(8) command failed because the tape subsystem has not been configured. Configure the tape subsystem by using the `tpinit`(8) command or reissue the `tpdaemon`(8) command without the `-b` option. - OPERATOR

`TM484 - Cannot reconfigure the tape subsystem when a tape device is open`

The tape subsystem could not be reconfigured because a tape device file is open. Wait for the tape device file to be closed and then reissue the configuration request. - OPERATOR

`TM485 - Unable to read system [table|variable]` *table-or-variable* `(errno =` *errno*`)`

The system variable or table, *table-or-variable*, could not be read; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM486 - Unable to set the effective user ID of` *command* `to root (errno =` *errno*`)`

The effective user ID of the *command* command could not be set to root; the error returned is *errno*. Contact your system support staff. - OPERATOR

`TM487 - unable to set overcommit option:` *reason*

Overcommitted mount requests cannot be enabled by the `-O` option on the `tpset`(8) command because of one of the following reasons: `avr not active` or `manual` or `auto loaders in device group.` The `avr not active` message means automatic volume recognition (AVR) must be active in order for you to enable overcommitted mount requests. The `auto loaders in device group` message means you can only apply this option to device groups for

which the tapes are loaded manually. If the tapes for some devices are loaded manually and other by autoloaders, the option cannot be enabled. - OPERATOR

`TM488 - overcommit active and device group cannot have an autoloader`

The `tpconfig`(8) command is not allowed to change the loader or the device group name of a device because doing so will allow a device using an autoloader to join a device group which has overcommitment turned on. - OPERATOR

`TM489 - waiting for overcommitted mount requests to complete`

The tape subsystem has issued the maximum number of overcommitted mount requests allowed. The system is queuing the `tpmnt`(1) command that you entered until an operator reduces the number of overcommitted mount requests by responding to some of them. - OPERATOR

`TM490 - AVR cannot be turned off because overcommit is active`

AVR - OPERATOR

`TM491 - file` *filename*`, line` *number* `at` *text*`, offset` *number*`: value of overcommit_max must be greater than 0 and less than` *value*

AVR - OPERATOR

`TM492 - TM995 - not used`

`TM996 - Process %d has open tape device and can't be killed`

The tape subsystem is attempting to remove an active user and cannot kill the process that opened the tape file. - OPERATOR

`TM997 - Process %d exited , killing related pidsG with %s`

The tape subsystem has detected the exit of the user process that owns the open tape file. It is attempting to kill any related processes. - OPERATOR

`TM998 - tape subsystem terminating, request denied`

A request was terminated because the tape subsystem is in the shutdown process. Reissue the request after the tape subsystem has been brought up. - USER

`TM999 - tape subsystem busy, tape daemon termination pending`

The tape subsystem is terminating and has found an active user; this delays the termination. - OPERATOR

Index

F

G

H

I

J

L

M

T

U

V

W