TOPS–10
Tape Processing Manual

AA–L412A–TB

July 1982

This manual describes the user interface to the TOPS–10 tape subsystem. The use of labeled and unlabeled tapes is addressed.

This is a new manual.

OPERATING SYSTEM: TOPS–10 V7.01

SOFTWARE: GALAXY V4.1

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid–Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

First Printing, July 1982

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

DEC	DECnet	IAS
DECUS	DECsystem–10	MASSBUS
DECSYSTEM–20	PDT	PDP
DECwriter	RSTS	UNIBUS
DIBOL	RSX	VAX
EduSystem	VMS	VT
digital	RT	

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

Contents

Chapter 5 Magnetic Tape Structure

Chapter 6 Magnetic Tape Data Modes

Chapter 7 Record Formats

Chapter 8 Label Formats and Contents

Chapter 9 The Mount Process for Labeled Tapes

Chapter 10 The Label Processor's Search Algorithm

Chapter 11 The TAPOP. Monitor Call

Chapter 12 Crash Procedures

Appendix A Label–Processing Stopcodes

Appendix B Tape Label Implementation Levels

Appendix C Glossary

Figures

Tables

Preface

Magnetic tape labels are reserved records on the tape used to hold information pertaining to that reel of tape and the files it contains. This manual discusses label–processing operations in addition to the use and formatting of magnetic tape labels. It also discusses unlabeled tape organization and processing. You may not need to read the entire manual. Scan the table of contents to determine the portions of the manual that concern you.

This manual has been divided into fields of interest for the following people:

- *Terminal users* utilizing magnetic tapes. Refer to Chapters 1, 3, and 4.
- *Operators* mounting and initializing magnetic tapes as well as handling label–processing errors. Refer to Chapters 1, 2, and 9.
- *Programmers* and other users creating tapes to interchange between systems and interchange between users. Refer to Chapters 1, 5, 6, 7, 8, 10, and 11.
- *System administrators* making judgments concerning magnetic tape use and operation. Refer to Chapter 1.
- *System programmers* analyzing label–processing crashes. Refer to Chapter 12.

Chapter 1
Introduction

This manual discusses labeled and unlabeled tapes that are under the control of the PULSAR and QUASAR programs. When the operator gives the command SET TAPE–DRIVE AVAILABLE, these programs are given control of the specified device. When they have control, the following automatic features can be used:

- Automatic Volume Recognition (labeled tapes only)
- Label processing (labeled tapes only)
- Volume switching

Automatic Volume Recognition and label processing are discussed in Section 1.2, Labeled Tapes. Volume switching is discussed below.

"Volume switching" refers to the internal and external operations involved in dismounting a volume and mounting the next one of the set. These operations are required during the reading or writing of a file that is large enough to span volumes. In these situations, the software that was activated with the original user MOUNT command is automatically reinvoked. For example, the operator receives a directive to mount the new volume; the new volume is assigned; and the user receives notification (if he specified /NOTIFY with the MOUNT command).

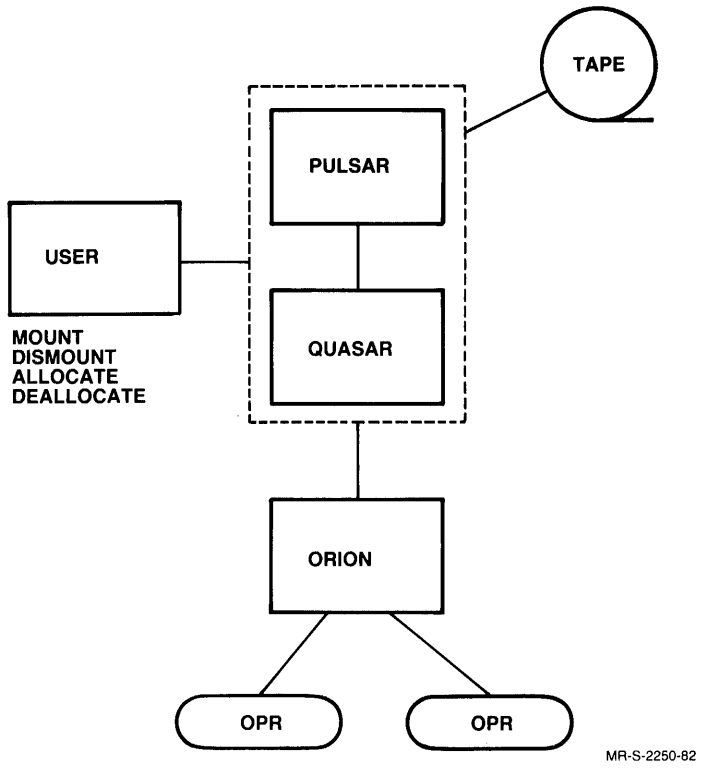
This automatic activity and the operator's response constitute volume switching. Chapters 3, Using Labeled Tapes; 4, Using Unlabeled Tapes; and 9, The Mount Process for Labeled Tapes, further describe this topic.

PULSAR and QUASAR are the chief components of the tape–processing system. OPR and ORION, the software components serving as the operator

interface to the system, also play a part. All these components communicate with each other through IPCF, the Inter–Process Communication Facility (described in the *TOPS–10 Monitor Calls Manual*).

Figure 1–1 shows the relationship among these components.

Figure 1–1: The Tape Processing System



1.1 Magnetic Tape Characteristics

Sections 1.1.1 through 1.1.5 describe some of the logical and physical properties of magnetic tapes. Chapter 5, Magnetic Tape Structure, describes these properties in detail.

1.1.1 Records

A record consists of related data that a program treats as a unit of information. When you output information to a magnetic tape, the system writes the information in sequential records on the tape. The default size for data records is 128 words. You can modify this value with the TAPOP. monitor call. (Chapter 11 discusses this monitor call.) Note that the label records are always 80 characters in length, regardless of the data record size.

1.1.2 Blocks

A block is a collection of data that PULSAR reads or writes as a unit. Each physical tape block may comprise one or more complete records or spanned record segments. (Chapter 7, Record Formats, discusses records and blocks.)

Where labels are concerned, however, a block contains only one record. The default block size is 128 words, which is the equivalent of a data record. But you can change the block size with the TAPOP. monitor call or the SET BLOCKSIZE monitor command. (Refer to the *TOPS–10 Operating System Commands Manual* for details on this command.) Blocks are separated from each other by spaces called interrecord gaps.

1.1.3 Volume Sets

A magnetic tape may contain one or more files or file sections. If the end–of–tape marker is reached before a file is completely written, that file is continued onto another tape and is known as a *multivolume* file. As with the first tape, this and all subsequent continuation tapes (up to 60) may contain files and/or file sections. The files contained on one magnetic tape or spanning multiple magnetic tapes make up a file set. And the tapes holding the files make up a volume set ("volume" being an equivalent term for magnetic tape reel).

1.1.4 Recording Density

Tape densities of 200, 556, 800, 1600, and 6250 bits per inch are supported by TOPS–10. Not all tape drives support all densities, however. Refer to Section 5.2.3, Tape Density, for details.

1.1.5 Track Status

The PULSAR–QUASAR system supports 7– and 9–track tapes. However, only 9–track tapes can be used with label processing. All 7–track tapes are considered to be unlabeled. (This is not to say that all 9–track tapes are considered to be labeled.)

1.2 Labeled Tapes

Magnetic tape is a widely used communications medium for the exchange of information between users, user programs, and computer systems. This communication may call for processing of tapes on foreign (non–TOPS–10) systems and accessing of tapes by nonowners of the tapes. All the information required for these interchanges is written directly on the magnetic tape in the form of distinct identifiable records called *tape labels*. Tape labels also serve useful purposes when no interchange is involved; for example, when a TOPS–10 user accesses his own TOPS–10–generated tapes.

Labels function in the following ways:

- Eliminating problems caused by the mounting of incorrect tapes. The system reads labels to verify that the correct tape has been mounted.
- Providing data protection by preventing premature overwriting of tape data and unauthorized access to tape files. The expiration date and accessibility–related fields contained in the tape labels provide this protection.

- Identifying and delimiting files for selective processing by user programs.
- Permitting users minimal involvement in reading from and writing to tapes (volumes). One task a user might ordinarily perform is end–of–volume processing; that is, reading or writing labels at the end of the tape and then determining whether another tape is required to continue processing a file. Using tape labels, the system automatically accomplishes this task for the user. (Refer to the discussion on volume switching in Section 1.2.2.3.)
- Identifying the system on which each file on the tape was created.
- Defining the magnetic tape record and file formats.
- Allowing user programs to perform error checking during tape reads. Such label fields as block count, file section number, and file number can serve as audit tools.
- Offering on–line mechanisms for controlling and maintaining the magnetic tape resources at an installation. For example, such label fields as the volume identifier and project–programmer number can facilitate tape library maintenance.

Chapter 8, Label Formats and Contents, describes all the label fields mentioned above.

1.2.1 Labels

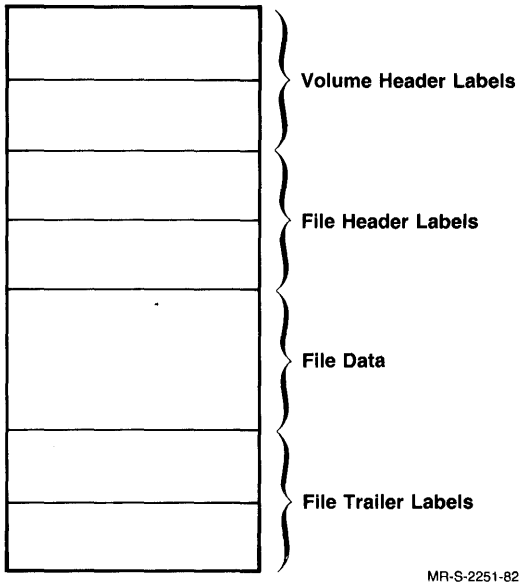
Magnetic tape labels are initially generated when the tape is *initialized.* (Refer to Chapter 2, Tape Initialization, for details.) As you output files to a volume set, the system writes additional labels as required to define each new file and each new volume in the set.

A labeled tape contains the following types of labels:

- Volume header labels.
- File header labels.
- File trailer or end–of–file labels.
- End–of–volume labels.

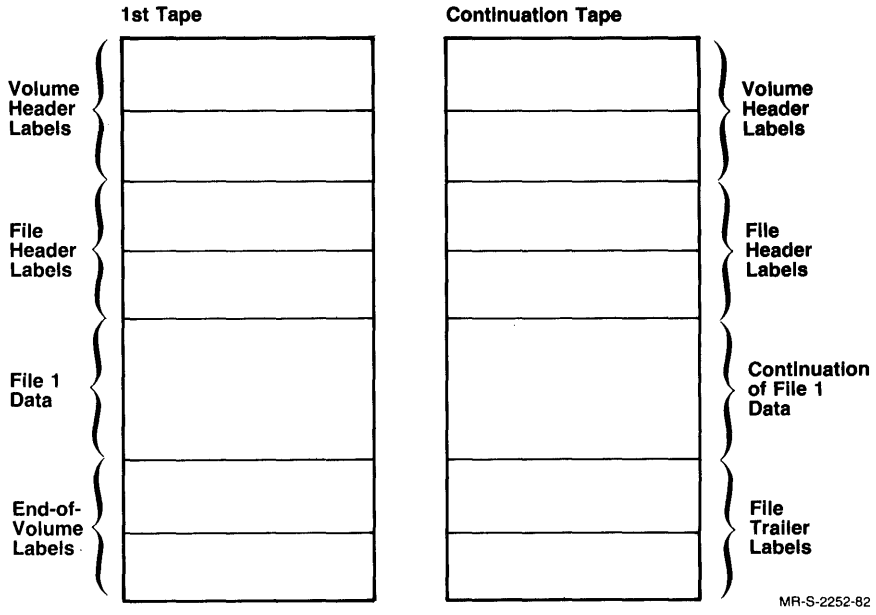
Volume header labels are the two labels at the beginning of the tape that describe the magnetic tape volume. The two labels that precede a file are called file header labels. These labels define the file that immediately follows. The two labels that follow a file are called file trailer or end–of–file labels. Figure 1–2 illustrates a labeled tape containing one file.

Figure 1–2: A Labeled Tape with One File



For multivolume files, the two end–of–volume labels are written at the end of the tape. These labels, written instead of the file trailer labels, indicate that the last file on the tape is not a complete file. See Figure 1–3.

Figure 1–3: Continuation Files in a Volume Set



Chapter 5, Magnetic Tape Structure, discusses the sequence of the tape labels in greater detail, and describes the other items that are included on a tape (such as tape marks and interrecord gaps). Chapter 8, Label Formats and Contents, describes the contents of each tape label.

1.2.1.1 System Labels — Labels that are specified in the ANSI standard and that are processed by PULSAR are called system labels. System labels include the volume and file header and trailer labels. Their 3–character identifiers are:

- VOL volume header
- UVL 2nd volume header
- HDR file header
- EOV end–of–volume
- EOF file trailer

An identifier followed by a sequence number (1 or 2) constitutes a label name and is part of the information contained within a label.

1.2.1.2 User Labels — Labels that are specified in the ANSI standard and that are processed by user programs are called user labels. The user label identifiers are:

- UVL user volume header
- UHL user file header
- UTL user trailer

The TOPS–10 system does not support user labels. When you transfer tapes that contain these types of labels from other systems, the label processor ignores them. UVL1 is considered to be a system label, however. It is written by the system and reserved for its use.

1.2.2 Label Processing

"Label processing" refers to the reading or writing of tape labels by PULSAR, the TOPS–10 label processor. Data files are not touched during label processing. And normally, users are unaware of this activity and of the labels' presence.

In performing its tasks, PULSAR interacts heavily with the QUASAR program. Through QUASAR, it sends messages to and receives them from other system components. For example, all operator requests for label processing (RECOGNIZE and SET TAPE–DRIVE INITIALIZE, for example) pass through QUASAR.

PULSAR also receives commands directly from QUASAR. Automatic Volume Recognition is an example of this. With AVR, QUASAR detects an on–line interrupt, then signals for PULSAR to verify labels.

Label processing takes place at various times as discussed in the following subsections.

1.2.2.1 Automatic Volume Recognition (AVR) — AVR is the facility that recognizes that a tape has been mounted, verifies its volume header labels, and then assigns the tape to a waiting user. If the tape has been premounted, that is, mounted before a program has asked for it, assignment takes place when it is requested.

AVR may be enabled or disabled (using operator commands) for a particular drive. When it is enabled, the operator need do nothing more than mount the correct labeled tape in response to a request; no type–in is required. The operator may even *premount* a volume in anticipation of a request. (Refer to Section 9.4.4, Premounted Volumes, for details.) Just the same, AVR recognizes a premounted volume and assigns the volume when it is needed.

1.2.2.2 Label Verification — As stated above, the label processor verifies labels when a tape is mounted. It also verifies labels when a user accesses a particular file. It checks the file header labels to see, for example, if the user is allowed access to the file or if the expiration date allows output to the file.

1.2.2.3 Volume Switching — On input, if PULSAR encounters the EOV label group, indicating a multivolume file, it requests mounting of the next volume of the set. This action is repeated until it encounters an EOF label group.

Similarly, if during output operations PULSAR encounters the end–of–tape marker, it requests mounting of the next volume so that operations can continue.

1.2.2.4 Label Creation and Updating — PULSAR writes labels:

- Before the first data is written on a tape
- As the second and subsequent output files in a set are opened and closed
- As the end of the first volume, and beginning and end of all subsequent volumes in a set are reached during output operations

The first case refers to *volume initialization*, a procedure required of all tapes intended for label processing. Initialization sets up the beginning–of–volume label set and all the file–related label sets required for the first output file. Chapter 2, Tape Initialization, describes this procedure.

The rest of the labels in a volume set are created or updated as needed during output operations.

Programmers can write label information by executing various TAPOP. monitor call functions. Refer to Chapter 11, The TAPOP. Monitor Call.

1.2.3 TOPS–10 Implementation Level

The TOPS–10 magnetic tape label processor partially implements the ANSI x3.27 – 1977 standard. The system reads and writes labels according to the standard, but does not ensure that the file data conforms to label specifications. In this respect, it does not fully conform to the ANSI standard. For example, the actual record format, block size, and record size could contradict the associated label parameters. User programs must enforce conformance.

1.2.3.1 ANSI Labels — The TOPS–10 label processor reads and writes level 4 labels. (See Appendix B for a definition of levels.)

1.2.3.2 OS/VS Labels — The TOPS–10 label processor implements IBM tape labels as defined for VS1 Release 3, and VS2 Release 2. (See IBM publication GC26–3795–2 for a complete description.)

1.3 Unlabeled Tapes

Unlabeled tapes are defined as tapes having no labeling information that describes the tape and its contents. Thus, use of unlabeled tapes puts extra responsibility on system administrators, users, and operators, who must be vigilant in handling the tapes. They must take care to see that the tapes are well catalogued, that correct tapes are mounted, and that unauthorized users do not gain access to tape data.

1.3.1 Bypass Label Processing

The system can process labeled tapes in unlabeled mode. This action is known as *bypass label processing*. Although tapes designated for this mode of processing contain labels, users may want to process the labels themselves rather than rely upon the label processor. Or, there may be other reasons why they would not want the system to process labels. For example, the tapes might contain nonstandard labels. The system usually requires that the user have privileges for this type of processing.

1.4 The System Administrator

There are several decisions that the system administrator must make regarding tapes. Some of these decisions are:

- Who should use magnetic tapes at the installation
- How many tapes should be assigned to a particular user or a particular project
- Who should be responsible for initializing magnetic tapes (if not the operator)
- Who should be responsible for cataloging magnetic tapes, that is, for maintaining the magnetic tape library

- How should the magnetic tapes be catalogued
- During the GALGEN procedure, what should the system defaults be for label type, track type, and recording density
- What information should the visual identification labels hold (refer to Section 1.4.2)

1.4.1 Volume Identifiers

The system administrator should verify that there is no duplication of volume identifiers (VOLIDs – specified at tape initialization) among the installation's labeled tapes. Switches to the SET TAPE–DRIVE INITIALIZE command provide automatic incrementation and assignment of alphanumeric VOLIDs to tapes.

1.4.2 Visual Identification Labels

Visual identification labels (VIDs) give an external indication of the contents of magnetic tapes. These labels are not machine–readable; they adhere to the outside of the tape. Although the information written on these labels is of the installation's choosing and need not conform to any standard, it should unambiguously identify the tape. For example, a visual identification label could contain the following information:

- File names
- File sequence numbers
- File creation dates
- Number of volumes in set if a multivolume set
- Owner of files
- Volume set name
- VOLID of next/previous volume in a multivolume volume set.

1.4.3 Unlabeled to Labeled Tape Conversion

To convert unlabeled tapes to labeled ones, the system administrator should follow these steps:

1. Initialize new or old scratch tapes. Follow the initialization procedures required for labeled tapes, as specified in Chapter 2, Tape Initialization, and in the *TOPS–10 Operator Command Language Reference Manual.*
2. Copy the contents of a designated unlabeled tape to one of the initialized tapes. This process is to be repeated for each unlabeled tape intended for conversion. The data transfer can be accomplished with the COPY command. **Note that COPY assumes a blocksize of 128 characters.**

Other than handling the issues above, the system administrator can leave tape labeling to the operator, the user, and the label processor. The rest of this manual discusses the user and operator interfaces to the tape–processing system.

1.5 The Operator

Several activities fall under the operator's responsibility. Some of them are:

- Initializing magnetic tapes
- Mounting and dismounting magnetic tapes
- Handling label–processing and tape mount errors
- Maintaining the visual identification labels

The operator should refer to the *TOPS–10 Operator's Guide* and to the *TOPS–10 Operator Command Language Reference Manual* for details on these and other topics.

1.5.1 Initializing Magnetic Tapes

In order to use label processing, the operator, or someone else with privileges, must have initialized all tapes by using the OPR command, SET TAPE–DRIVE INITIALIZE. This command writes the initial labels that identify the volume and that ready it for the first file to be written. Chapter 2, Tape Initialization, describes this process.

NOTE

New **unlabeled** tapes that have never been used must also be initialized. No labels are written during this process, however. An empty record followed by two tape marks is written on the tapes. These indicate that the tapes will be used for unlabeled purposes.

1.5.2 Mounting and Dismounting Tapes

When the operator receives a mount request for a labeled tape, he/she examines the various magnetic tape reels contained in the installation's tape library. After finding and mounting the requested tape, the operator waits for the system's response. This response is an acknowledgement that the tape has been mounted and that all is well, or it is an error message. Chapter 9, The Mount Process for Labeled Tapes, describes the internal aspects of the mounting process. The external aspects are covered in the operator–specific manuals.

For unlabeled tapes, the operator mounts a tape and issues the appropriate command to let the system know that the requested tape has been mounted. No message is then issued to indicate that the correct tape has been mounted.

1.5.3 Handling Errors

Any errors that occur during label processing or mounting result in messages to the operator, who then rectifies the situation or notifies users that the tape operation has been aborted. For example, if, in reading the volume header labels, the label processor finds that an incorrect tape has been mounted, it sends an error message to the operator. Then, the operator searches for the correct tape and mounts it.

Or, the label processor might find that the user is not allowed access to the specific tape that he/she has requested. Again, the operator receives notification. In this case, the operator informs the user, who takes the next action. The user might issue a mount request for another volume or try to learn why he/she was denied access to the volume. (Chapter 3, Using Labeled Tapes, describes protection codes.)

Also, if a labeled tape is mounted when an unlabeled one is expected, or vice versa, the operator receives an error message.

1.5.4 Maintaining the Visual Identification Labels

The operator is also responsible for the visual identification labels affixed to magnetic tapes. These labels should be updated when the tape is first used, overwritten, or returned to the installation's scratch pool.

Chapter 2
Tape Initialization

Chapter 2 discusses initialization for labeled and unlabeled tapes.

2.1 Initializing Labeled Tapes

All magnetic tapes intended for label processing must be initialized before any data can be written on them (with the exception of bypass label–processing tapes). During initialization, the system writes:

- The beginning–of–volume label group.
- Two tape marks framing an empty file.
- The end–of–file label group.
- A double tape mark — logical EOT.

The format of an initialized labeled tape is represented in Figure 2–1:

The newly written labels contain sufficient information to establish a system–accessible single–file volume. But because initialization neither requires nor allows any file data to be written, the file is empty.

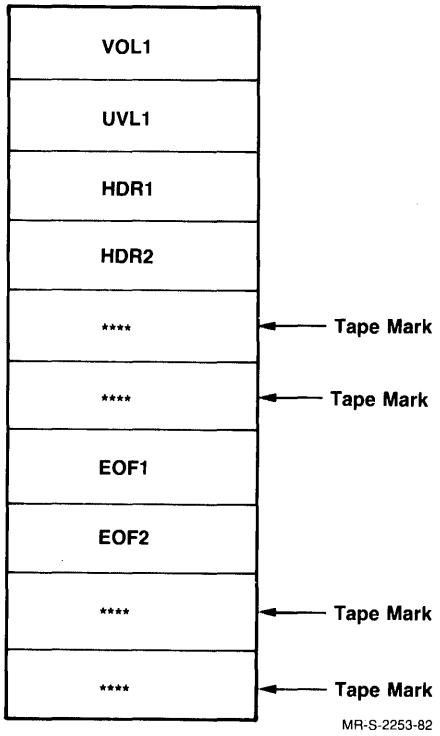
When the tapes are later used for output, the system updates the fields within these labels and creates additional labels as needed. Such fields as the file creation date and expiration date are updated when files are created on initialized tapes, and the end–of–volume label set is written when needed.

The operator command, SET TAPE–DRIVE INITIALIZE, is used to initialize magnetic tapes. With this command, the operator specifies such information as the following to be assigned to the labels: project–programmer

number, expiration date, and protection code for the entire volume (different from the protection code for individual files). The system augments these inputs with all other information required to create the initial labels.

If the operator chooses not to specify the various values, the system takes such default action as the following: it designates [0,0] as the owner, assigns a protection code of 000, and uses the current date as the expiration date. This type of tape is recognizable to the label processor as a fully labeled but unprotected tape. Refer to the list of SET TAPE–DRIVE INITIALIZE switches in Table 2–1 for complete default information.

Figure 2–1: An Initialized Labeled Tape



2.2 Initializing Unlabeled Tapes

Unlabeled tapes contain only files and tape marks. Before the first file can be written however, the operator must initialize these tapes with the SET TAPE–DRIVE INITIALIZE command.

Initialization is performed on tapes intended for unlabeled use to establish this intent in writing. (The system makes sure that these tapes are not used when labeled ones are required.) Thus, when the operator initializes a tape as unlabeled, the system writes a record of 80 blank frames at the beginning of the tape, followed by two tape marks (logical EOT). The system recognizes this as an unlabeled tape.

When a user issues a write instruction with the tape positioned at the beginning point, the blank frames and tape marks are overwritten by the user's data.

2.3 The SET TAPE–DRIVE INITIALIZE Command

Volume initialization can be performed in a few minutes following a user request for a new tape.

To initialize one or more tapes, the operator gives the following command to the OPR prompt:

OPR>SET TAPE–DRIVE device: INITIALIZE/switch1,/switch2,...RET

where "device" is the physical name of the tape drive.

Table 2–1 describes the valid switches for the SET TAPE–DRIVE INITIALIZE command. Refer to the *TOPS–10 Operator Command Language Reference Manual* for details on initialization.

Table 2–1: SET TAPE–DRIVE INITIALIZE Switches

Switch	Meaning
/COUNT:	specifies the number of tapes to be initialized. The /COUNT: switch, when specified, provides a mechanism for initializing multiple tapes having the same attributes. After initialization, the tape is automatically unloaded and the tape drive is ready to accept the next tape to initialize.
/DENSITY:	specifies the density of the tapes to be initialized. If the /DENSITY: switch is not specified, the density defaults to that specified during GALGEN (usually 1600 bits/inch).
/INCREMENT:	specifies the numeric value by which to increment the tape volids. This switch is valid only when specified with the /COUNT: and /VOLUME–ID: switches. If this switch is not included, the default numeric value is 1. For example, to initialize a set of 10 tapes with VOLIDs from 442000 through 442900, the switches /COUNT:10, /INCREMENT:100, and /VOLUME–ID:442000 are given. Every tape mounted on the tape drive is initialized with the next sequence number as specified with the /INCREMENT: switch.
/LABEL–TYPE:type	specifies the type of label to be written on the tape. The type can be ANSI, EBCDIC, or UNLABELED. The default label is ANSI when this switch is specified. If the /LABEL–TYPE: switch is not specified, the default is UNLABELED.
/OVERRIDE–EXPIRATION: (YES or NO)	specifies whether to check the expiration date of the data on the tape. If NO is specified and there is an attempt to reinitialize a labeled tape whose first file has not expired, the system issues an error message and does not reinitialize the tape. If YES is specified, each tape is reinitialized unconditionally. When initializing virgin tapes, the operator specifies YES to prevent the tape drive from "running away" when the label processor tries to read a label from the tape. If this switch is not specified, the default is NO.

Table 2–1: SET TAPE–DRIVE INITIALIZE Switches (Cont.)

Switch	Meaning
/OWNER:[proj,prog]	specifies the project–programmer number for the user who owns the tape(s) to be initialized. If the /OWNER: switch is not specified, the tape is initialized as owner [0,0].
/PROTECTION:nnn	specifies a 3–digit octal number as the protection code of the tape. The number is usually from 000 (where anyone can use the tape) through 777 (where no one can use the tape). If the /PROTECTION: switch is not specified, the default is 000.
/TAPE–DISPOSITION: condition	specifies what is to be done to the tape after it is initialized. The condition can be either HOLD or UNLOAD. If HOLD is specified, the tape is initialized and is not unloaded. A user requesting the tape can use it without the operator's having to reload it. If UNLOAD is specified, the tape is initialized and is unloaded from the tape drive. The default is UNLOAD. If a value greater than 1 is specified with the /COUNT: switch, the /TAPE–DISPOSITION: switch is ignored and UNLOAD is assumed.
/VOLUME–ID:"volid"	specifies a volume identification for the tape. The VOLID is from one to six characters long. If the VOLID is to contain any nonalphanumeric characters, it must be enclosed in double quotes ("). This switch is ignored if the tape is being initialized as an unlabeled tape.

Chapter 3
Using Labeled Tapes

This chapter discusses the user interface to the tape–labeling system.

3.1 The Terminal User

When sitting at a terminal writing to a magnetic tape and reading from a magnetic tape, you need not do anything different with labeled tapes than with unlabeled tapes. In either case, you issue the MOUNT command to assign a magnetic tape unit to your job and to alert the operator to mount a tape for you.

Under most conditions you need not worry about magnetic tape labels. The label–processing software reads and writes tape labels without your intervention.

3.2 The Programmer

As a programmer, you can control many label–processing operations with the various TAPOP. monitor call functions. For example, you can use the .TFLPR function to position the tape to the beginning of any file or to the end of the last file of the volume set. Using this function, you can also supply or change the data for many of the tape label fields. Several other TAPOP. functions are discussed in the remainder of this chapter.

Refer to Chapters 5 through 12 for more complete programmer information.

3.3 Accessing Tape Data

You can access a tape only when you, the operator, and the label processor can uniquely identify the desired magnetic tape reel.

Primary identification is the unique name assigned to each tape. This name is called a VOLID (volume identifier), and it is assigned to a tape during initialization. You specify the VOLID(s) when you issue the MOUNT command, letting the system and the operator know which tape(s) you wish to use. For input, all VOLIDs must be specified for the tapes in a volume set. The label processor then directs the operator to mount and dismount these tapes as you process files in the set.

The visual identification label also provides identification. This identification is on a label that is glued to the magnetic–tape reel; it is not machine readable. The system administrator determines the VID label content. The operator uses this information to select tapes that you request through the MOUNT command.

3.4 The MOUNT Command

The MOUNT command causes a magnetic tape to be physically mounted by the operator and assigned to your job.

Command format:

MOUNT resource:logical–name:/switches

where resource can be the:

- Physical device name
- Logical–name previously associated with a resource
- Volume set definition, for example: BILLS–80(NOV,DEC)

Logical–name is the name your program will use to refer to the resource. The default logical name is taken from the first six characters of the volume set name, or from fewer characters if a nonalphanumeric is encountered. For example, the volume set name above would produce this logical name: BILLS.

The MOUNT command is described in detail in the *TOPS–10 Operating System Commands Manual.*

The switches that you specify in the MOUNT command that relate directly to label processing are described below.

3.4.1 The /LABEL–TYPE: Switch

The /LABEL–TYPE switch identifies the format of the labels. The parameters for this switch are:

- ANSI
- IBM
- EBCDIC
- NOLABELS

- NONE
- UNLABELED
- USER–EOT
- BYPASS
- BLP

ANSI indicates that the labels conform to ANSI standards. IBM and EBCDIC are equivalent parameters that indicate IBM–formatted labels. NOLABELS, NONE, UNLABELED, and USER–EOT are equivalent parameters indicating that the tape does not contain standard labels. With NOLABELS, NONE, and UNLABELED, the label processor performs automatic volume switching; it does not perform it with USER–EOT. BYPASS and BLP are equivalent parameters that result in the label processor performing no processing. These two parameters are usually reserved for privileged users. Chapter 4 further discusses the parameters for unlabeled and bypass–labeled tapes.

3.4.2 The /VOLID: Switch

The /VOLID switch must be specified in your MOUNT command unless /SCRATCH is specified. This switch specifies which volume (VOLID) or volumes (VOLID(1),...VOLID(n)) you want an operator to mount. If you specify fewer volumes on output than are needed, the operator can use scratch volumes as the extra volumes. For multivolume files, you must enter (on input) all volume names associated with the file when you request mounting of the first reel.

3.4.3 The /NOTIFY Switch

The /NOTIFY switch is important. This switch causes the system to notify you of automatic volume switches and to give you the VOLID of the new volume. (Refer to Section 9.3, Mounting the 'Next' Tape of a Volume Set, for information on volume switching.) By this action, you can know the VOLIDs of scratch volumes that the operator may have had to mount for your output operations.

The /NOTIFY action is performed if you specify the /NOWAIT switch in the MOUNT command.

3.5 Protection Codes

There are two types of protection codes associated with labeled tapes. The volume protection code, written in the UVL1 label during initialization, determines who can access the volume. The file protection code, written in the HDR2 label for each new file, determines who can access the file.

Programmers can write the file protection code with the .TFLPR function of the TAPOP. monitor call. But the volume protection code is written only

when the tape is initialized. Note that the protection codes correspond to the standard disk file protection codes discussed in the *TOPS–10 Operating System Commands Manual.*

The protection code consists of nine bits, divided into the following three classes:

1. The first three bits refer to the owner of the file or volume.
2. The second three bits refer to users with the same project number as the owner (that is, any user logged in under the same project number as the owner, regardless of his programmer number).
3. The last three bits refer to all other users.

The protection codes and their meanings are listed in Tables 3–1 and 3–2.

Table 3–1: Protection Codes for Field 1 Owner Protection Codes

Code	Accessibility by Owner
7	You (the owner) have no access privileges. However, you can rename the file to change the protection code, using the RENAME command. If another user tries to access the file, the File Daemon* is called.
6	You can execute the file. If another user tries to write or rename the file, the File Daemon is called.
5	You can read and execute the file. If another user tries to write or rename the file, the File Daemon is called.
4	You can change the protection code of the file, rename, write, read, execute, update, and append to the file. If another user attempts to write, rename, or supersede the file, the File Daemon is called.
3	You can only rename the file.
2	You can write, read, execute, update, and append to the file.
1	You can rename, write, read, execute, update, and append to the file.
0	You can change the protection code of the file. You can also rename, write, execute, update, and append to the file.

* For complete information on the File Daemon, refer to the *TOPS–10 Monitor Calls Manual.*

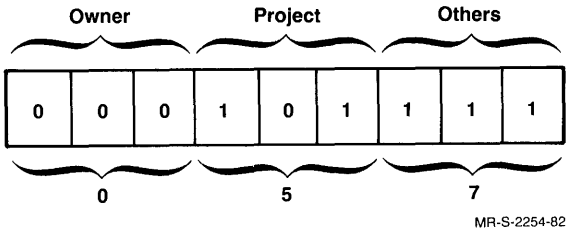
Table 3–2: Protection Codes for Fields 2 and 3 Project–members and Other User Protections

Code	Access Privileges
7	Users other than the owner cannot access the file.
6	The user can execute the file. If the user attempts to delete, read, change, rename, append–to, or run the file with the RUN command, the File Daemon is called.
5	The user can read and execute the file. If the user attempts to write or rename the file, the File Daemon is called.
4	The user can read, execute, and append–to the file. If the user attempts to write, rename, or supersede the file, the File Daemon is called.
3	The user can read, execute, append–to, and update the file.
2	The user can write, read, execute, append–to, and update the file.
1	The user can rename, write, read, execute, append–to, and update the file.
0	The user can change the protection–code of the file. He can also rename, write, read, execute, change, append–to and update the file. This code allows all users full access privileges.

Figure 3–1 illustrates the 9–bit protection code field of a file or volume having a protection code of 057. This protection code indicates three things:

1. The owner has full privileges (code 0).
2. The project members have read and execute privileges (code 5).
3. All other users have no access privileges (code 7).

Figure 3–1: Access Protection Code



3.5.1 The Accessibility Characters

There are two types of accessibility characters. The first type, written in the VOL1 label, indicates to the system whether it should check the volume protection code in UVL1. If the character is zero, anyone can use the tape and no checking is performed. If it is one, the system checks the protection code. During initialization, the system writes this character in accordance with the volume protection code.

Likewise, the second type of character, written in HDR1 for each new file, determines whether the file will be protection checked according to the protection code in the file's HDR2 label. Thus, if this field contains a space, anyone is allowed access to the file. If it contains a one, the protection code is checked before access is allowed.

Refer to the descriptions of the Volume Header Label and of the first File Header Label in Chapter 8, Label Formats and Contents, for further details on the accessibility characters.

3.6 Writing to a Magnetic Tape

After you have been notified of a successful mount, you can open the magnetic tape file and record data.

Your first output attempt causes the label processor to reverify the volume–header label set. This action duplicates the action taken during mounting. (Refer to Chapter 9, The Mount Process for Labeled Tapes.) Here, the system double–checks to ensure that no one has tampered with the tape since it was mounted. Again, the system checks your project–programmer number and other label parameters. If you are updating a file, it checks the protection code of the current file to ensure that you are allowed to modify it.

In performing output operations, the system first updates fields in the labels that were written during initialization, then positions the tape by filename or file position if necessary. (Refer to Chapter 10, The Label Processor's Search Algorithm, for positioning information.)

You can write one or more files on a magnetic tape, or you can write just one segment of a file on a tape. In the latter case, the remainder of the file is placed on additional magnetic tapes. When a file spans more than one magnetic tape, it is called a multivolume file. A multivolume file can occupy up to 60 volumes.

As you add files to a volume set, the system overwrites the last of the logical EOT tape marks, leaving one tape mark between the end–of–file label group and the following beginning–of–file label group.

If, as is the case with multivolume files, your file is large enough to reach the physical end–of–tape mark before you have closed the file, the label processor will write an end–of–volume label set, indicating that your file is to be continued onto another volume. In addition, the label processor will automatically alert the operator to mount the next volume for the continued section of the file. (Refer to Section 9.3, Mounting the 'Next' Tape of a Volume Set, for details on volume switching.) Thus, you need not be concerned with end–of–tape operations while processing a file. If your data requires more tapes than were specified in the MOUNT or ALLOCATE command, the system requests the necessary scratch tapes for your job.

3.6.1 Updating Files in a Volume Set

When you update a file on a tape, all files beyond this file in the volume set become lost. You lose these files because when you close a file, the system closes the volume set. (Refer to Section 3.6.4, Closing a File, for details.) Therefore, you should use some alternative method for updating tape files. Perhaps you could begin copying the file set to another set of tapes. When the designated file has been copied, you could update it on the new tape. After you have updated the file, you could continue copying the remaining files in the set.

Or, you could copy the files to disk; update the designated file; then rewrite the files to tape.

3.6.2 Specifying End–of–Volume Processing

You can force the label processor to write an end–of–volume label set if you wish to establish the point at which the continued section of a multivolume file is to begin on the next volume. Use the .TFFEV function of the TAPOP. monitor call to force the end–of–volume label set to be written.

3.6.3 Writing Label Parameters

The .TFLPR function of the TAPOP. monitor call allows you to override the system defaults for file–specific data that is contained in the beginning–of–file, end–of–volume, and end–of–file label sets. To name a few, you can specify record length, block length, expiration date, and filename. Refer to the full description of this function in Chapter 11, The TAPOP. Monitor Call.

3.6.4 Closing a File

When you close the file (with the CLOSE monitor call, for example), the label processor writes the end–of–file label set. It then writes two tape marks. These marks indicate the logical EOT as well as the end of the volume set.

3.7 Reading from a Magnetic Tape

When you open a file for input, the system reverifies the tape labels as it does when you write to a file. It similarly positions the tape to the desired location.

The label processor directs the operator to mount and dismount volumes as required as you read through the file or the file set. Your job receives notification when end–of–file is reached.

3.7.1 Reading Label Parameters

The .TFLPR function of the TAPOP. monitor call allows you to read file–specific information contained in the file header and file trailer labels. Refer to Chapter 11, The TAPOP. Monitor Call, for details.

3.8 Positioning the Tape

You can position the tape to access specific data by using monitor commands, monitor calls, and language statements.

3.8.1 Using Monitor Commands

You can position the tape with the following monitor commands: BACKSPACE, SKIP, EOF, REWIND, and UNLOAD. In releases previous to GALAXY 4.1, the monitor was solely responsible for these commands. But with GALAXY Version 4.1, this arrangement is changed. If you issue the MOUNT command (as opposed to the ASSIGN command), and do not specify bypass label–processing mode, PULSAR controls the operation of these commands. Otherwise, the monitor handles these commands as it did previously.

PULSAR performs as follows for each command:

- BACKSPACE and SKIP — These commands cause PULSAR to perform volume switching when the tape reaches the beginning– or end–of–tape markers. PULSAR also positions the tape to the previous/next file if a file–header label set is encountered.
- EOF — This command skips the remaining data in the file and positions the tape at the beginning of the next file. For the last file of the set, the tape is positioned at the end of the data area.
- REWIND — This command positions the tape at the beginning of user data, just after the beginning–of–volume label group.
- UNLOAD — This command only rewinds the tape. You should use the DISMOUNT command to relinquish a tape drive.

3.8.2 Using Monitor Calls

You can access files by name or by position within the volume set. Use various functions of the TAPOP. monitor call to accomplish this. Chapter 10, The Label Processor's Search Algorithm, describes the operations that result when you specify a filename or sequence number to access data.

You can also access specific records using the TAPOP. monitor call.

3.8.3 Using Language Statements

You can also select records using language statements. Refer to the appropriate language reference manual or users guide for the specifics of processing tape records.

3.9 Running COBOL or FORTRAN Programs

No special procedures are required for reading and writing tapes with COBOL or FORTRAN programs. The label processor reads/writes labels and performs volume switching for your job as described above. This volume switching requires no intervention by the COBOL or FORTRAN run–time systems. Thus, the trailer records that COBOL ordinarily writes at the end of a volume are not written. Refer to Chapter 4, Using Unlabeled Tapes, for a description of differences with unlabeled tapes.

3.10 Using the BACKUP Utility

With BACKUP, the label processor performs as described in Section 3.9. BACKUP is not involved in volume switching and thus does not write trailer records. Refer to Chapter 4 for differences with unlabeled tapes.

3.11 Using the DIRECTORY Command

The DIRECTORY command can provide information on labeled tapes. It can read both ANSI and EBCDIC labels. An example is given below:

```
.MOUNT T(T,B):T:/WRITE/LABEL:ANSI/NOTIFY
[7:21:51]
[Mount request T queued, request #150]
[Magtape T mounted on MTA263 with logical name T]

.DIR T:

  Read Density:1600 Parity:Odd 9-Track Write enabled REELID:T

  Labeled tape file information:
    File Name:"FILE.001          "
    Protection:000, Creation Date:14-May-81, Expiration Date:14-May-81
    Block size:640, Record size:640
    Record Format:Undefined, Form control:None
```

Chapter 4
Using Unlabeled Tapes

When you choose not to use labeled tapes, you must specify the label type as either "NOLABELS", "NONE", "UNLABELED", or "USER–EOT" in the MOUNT command. These label types have particular significance in regard to volume switching.

4.1 Volume Switching

With unlabeled tapes, volume switching is performed for tapes specified with the NOLABELS, NONE, and UNLABELED parameters. For USER–EOT, your program sees the end–of–tape condition and has to then issue the .TFFEV function of the TAPOP. monitor call in order to switch to the next reel. It is important to specify the /NOTIFY switch in the MOUNT command if you anticipate volume switching on output. Section 3.4, The MOUNT Command, discusses this switch.

4.1.1 Volume Switching with COBOL

When using COBOL programs to read or write tape data, you specify the USER–EOT parameter. This parameter gives the COBOL run–time system the responsibility for handling end–of–volume operations. Thus, when you reach the end of a tape while reading or writing files, your job stops and waits for the next action to come from the COBOL run–time system (rather than from PULSAR). The COBOL system responds by writing the COBOL–specific trailer records and issuing the .TFFEV function of the TAPOP. monitor call. This function calls upon PULSAR to perform the volume switch.

4.1.2 Volume Switching with BACKUP

Follow the same procedure with BACKUP as with COBOL. This causes volume switching to occur in the same manner.

NOTE

If you write tapes in USER–EOT mode, you must also read them in this mode. Likewise, any tapes written in non–USER–EOT mode must be read in this mode.

4.2 Reading from an Unlabeled Tape

When an unlabeled tape is mounted and verified that it contains no standard volume labels, the system positions the tape before the first record. If the tape has more than one file and you want to use other than the first file on the tape, you can position the tape to the specific file by giving the SKIP command.

You can also position a tape to a specific file by giving the BACKSPACE command. When you give either command, the tape is spaced forward or backward the number of files you specified. You should use these commands only if you know the location of the file relative to the tape's current position. Refer to Section 3.8.1 for a complete discussion of positioning the tape using monitor commands.

You can also access tape data using the TAPOP. monitor call or language statements.

When your program encounters an end–of–file condition (reads a tape mark), the system informs your program and leaves the tape positioned after the tape mark and before the next file on the tape. For multivolume files, either you or the system can handle the required volume switch. As discussed earlier, control over volume switching depends upon the parameters you specify with the /LABEL–TYPE switch in the MOUNT command. Automatic volume switching takes place when PULSAR reaches the logical EOT (two tape marks).

4.3 Writing to an Unlabeled Tape

When an unlabeled tape is mounted for output, the system positions the tape before the first record on the tape. If the tape has been initialized (refer to Chapter 2) but never used, it contains a record of 80 blank frames followed by two consecutive tape marks. When you issue the first write instruction, the system overwrites the current contents of the tape with your data record. When you close the file (with the CLOSE monitor call, for example), the system writes two tape marks. These marks indicate the logical EOT as well as the end of the volume set.

If you want to write the new data following an existing file, you can position the tape beyond an existing file with the SKIP command. Your first output to the new file causes the last tape mark to be overwritten, leaving just one tape mark between the files.

You can also use other monitor commands to position the tape on output. (Refer to Section 3.8.1.) In addition, as mentioned above, monitor calls and language statements allow you to access tape data.

Volume switching is handled the same as when reading a multivolume file.

4.3.1 Updating Files in a Volume Set

When you update a file on a tape, all files beyond this file in the volume set become lost. You lose these files because two tape marks are written when you close a file. These marks indicate the end of the volume set. Therefore, you should use some alternative method for updating tape files. Perhaps you could begin copying the file set to another set of tapes. When the designated file has been copied, you could update it on the new tape. After you have updated the file, you could continue copying the remaining files in the set.

Or, you could copy the files to disk; update the designated file; then rewrite the files to tape.

Chapter 5
Magnetic Tape Structure

5.1 Introduction

This chapter discusses the physical and logical structure of magnetic tapes. In the discussions, structure is linked to the two tape–data segmentation modes: physical and logical.

Physical segmentation refers to the separation of data by marks or interrecord gaps (gaps that are lengths of magnetic tape containing no software–accessible data). Blocks, marks, and gaps relate to a tape's physical structure.

With logical segmentation, data separation is established by the software. The software accesses logical segments of data and is made aware of the marks and gaps only through detection of these physical delimiters by the tape unit. Volume sets, files, and records relate to a tape's logical structure.

Often physical and logical segmentations of tape data coincide. For instance, when block length equals record length, the physical record (block) is the same length as the software–accessible record. In this case a gap separates each logical record.

5.2 The Physical Structure

Sections 5.2.1 through 5.2.4 discuss the physical structure of magnetic tapes.

5.2.1 Blocks

A block is a physical record, a collection of characters that the system reads or writes as a unit. The default length of a block is 128 words. You can reset the default to any value up to 4094 words with the .TFBSZ function of the TAPOP. monitor call. (Refer to Chapter 11, The TAPOP. Monitor Call, for information.) You can also reset this value with the SET BLOCKSIZE monitor command.

5.2.2 Tape Marks and Interrecord Gaps

Tape marks and gaps are physical delimiters that are recorded on a magnetic tape by the tape unit. Exactly what is recorded is of concern only to the hardware. Figure 5–2 and the figures in Section 5.3.2.2, Tape Label Sequences, show the placement of marks and gaps in the nondata areas of a volume.

With labeled tapes, tape marks are recorded before and after every data file to separate a data file from its preceding and following tape labels. They are also recorded between the end–of–file label group and the following beginning–of–file label group. Two tape marks are written after the last label group (EOV or EOF) on the volume.

With unlabeled tapes, a tape mark is recorded after every file. Two tape marks are recorded at the end of meaningful information on the tape. Also, if the last file continues onto another volume, only one tape mark is written at the end of the volume.

Gaps are used to separate physical records (blocks) on the tape.

5.2.3 Tape Density

Density refers to the number of characters that can be recorded on one inch of magnetic tape. The possible densities for tapes processed under TOPS–10 are 200, 556, 800, 1600, and 6250. These numbers, which represent bits–per–inch, are specified with the MOUNT command. The rate of transferring data from a tape unit to memory, or vice versa, depends on the tape density and the speed of the unit in use. If you try to record at too high a density you may lose data.

Note that not all devices can process tapes at all densities. See your system operator for the valid densities of the tape drives on your system.

Files and labels are written at the same density for all volumes in the set.

5.2.4 Data Modes

Magnetic tape data can be recorded in one of the following data modes:

- DIGITAL–compatible core dump mode (7– and 9–track)
- 8–bit mode, 4 bytes per word
- 6–bit mode, 6 bytes per word (9–track, TU70 only)
- 7–bit mode, 5 bytes per word (ANSI–ASCII mode, TU70 only)

These modes govern the transfer of data between memory and the magnetic tape and determine how the monitor records the bits of a 36–bit word onto tape. Chapter 6, Magnetic Tape Data Modes, discusses this topic in detail.

5.3 The Logical Structure

Magnetic tapes consist of files and records and may contain labels coded in either industry–compatible ASCII or IBM–compatible EBCDIC. (The information in both types of labels is recorded in 8–bit bytes.)

With labeled tapes, sets of labels preceding and following each tape file indicate, for example, the record length (in characters) and the file length (in blocks). Also, the first two records of each volume are volume labels that contain information common to all files on the volume. They are called the Volume Header Label and the Extended Volume Label. The Extended Volume Label is present only for those volumes created on a TOPS–10 system. This does not present a problem when transferring tapes between systems.

5.3.1 Volume Sets, Files, Records

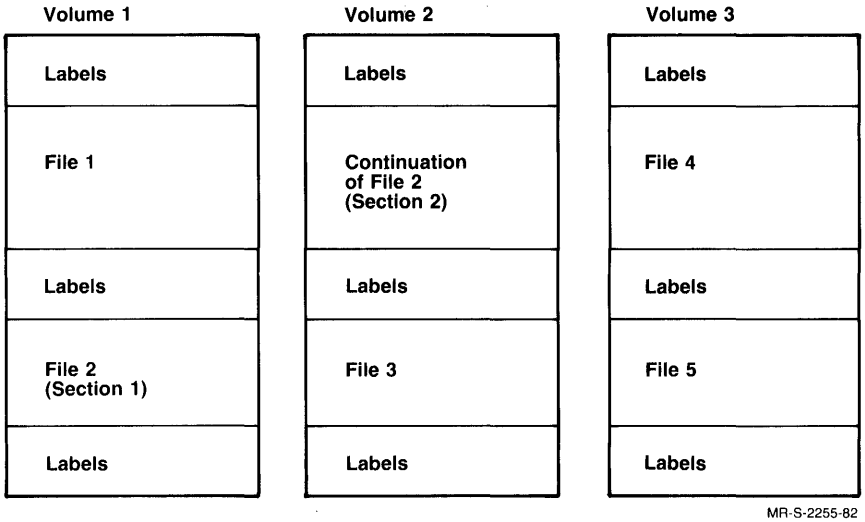
Data recorded by the operating system on magnetic tape is logically divided into volume sets, files, and records.

Volume Sets

A volume set comprises one or more (up to 60) magnetic tape reels logically treated as one volume. That is, to user programs, a volume set is a collection of files on one volume.

A tape within a volume set may contain one or more files, or it may contain one section of a multivolume file. A volume set may consist of a number of files, any of which may contain multiple sections. For example, a volume set could contain three volumes, as represented in Figure 5–1.

Figure 5–1: Volume Set of Labeled Tapes

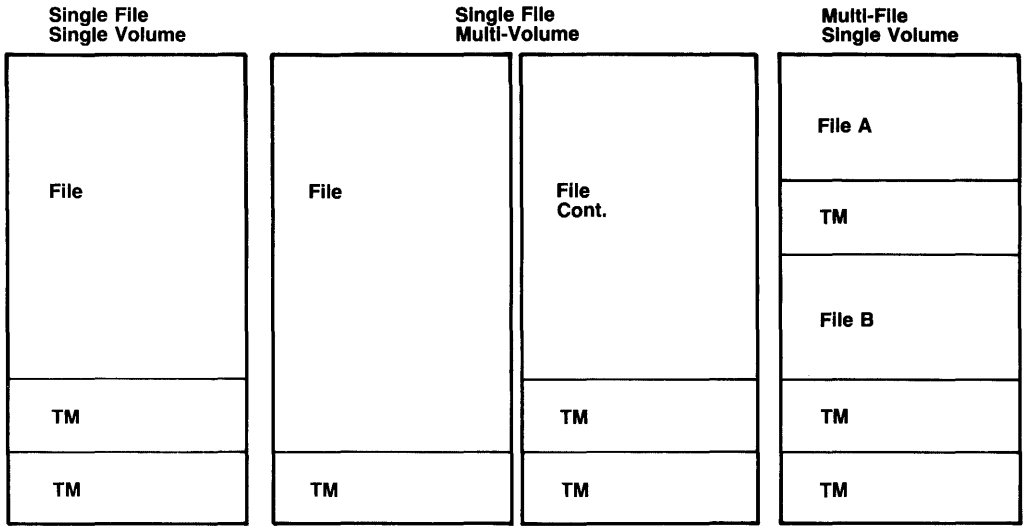


Volumes 1 and 2 contain a complete file and a file section. Volume 3 contains two complete files. Note that all sections of a file are contiguous. Refer to Section 5.3.2.2, Tape Label Sequences, for illustrations of a single file and multiple files on single– and multiple–tape volume sets.

Each volume within the set must have a unique VOLID, specified during initialization.

The organization of files on one or more unlabeled tape volumes is shown in Figure 5–2.

Figure 5–2: Volume Sets of Unlabeled Tapes



Files

The files in a volume set are known as a file set. Each file can consist of one or more records. When there is only a section of a file on a magnetic tape volume, the file is said to be a multiple–volume or multivolume file. Tape labels define the position of each file/file section of a file set. Each file within the set must have a unique filename.

Records

A logical record is usually a segment of a file.

Record format refers to the relation of blocks to records. By default, the record and block lengths are equal and are established by the operating system. If you use the system default value when you create a file, the labels correctly indicate the file format. However, if you choose a different record format, you must override the system default values before you initiate the first output to the file. You can change the record format with the .TFLPR function of the TAPOP. monitor call. Using this function, you can indicate the presence of variable–length records, spanned records, or multiple–record blocks. (Refer to Chapter 7, Record Formats.)

5.3.2 Labels

Each label is an 80–character record. With only one label per block, the labels are separated from each other by interrecord gaps. Labels are separated from file data by tape marks. The record size for all labels generated by the system remains at 80 characters regardless of the logical record and block sizes selected by the system or the user program for file data.

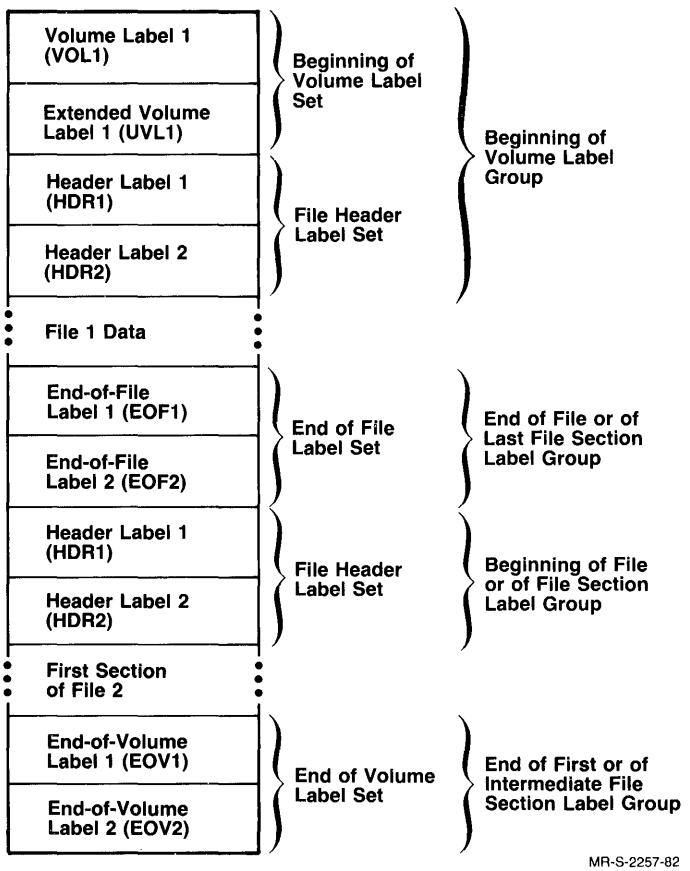
Although a label generated by the system is 80 characters in length, longer labels generated by other systems are accepted, but only the first 80 characters are processed.

Section 5.3.2.2, Tape Label Sequences, shows how labels are placed throughout a volume set. Various volume set configurations are compared. By way of introduction, the section below (5.3.2.1) explains the label–grouping scheme.

5.3.2.1 Classification of Labels — As mentioned in Chapter 1, there are several types of labels (HDR, EOV, and others). Each type usually includes more than one label. This is because a single 80–character label is sometimes not long enough to contain all the information necessary to describe the file data. Together, the labels within a type constitute a label set. Label set members carry the same label identifier but differ in respect to label number. (The label identifier and label number are among the information fields in a label — refer to Chapter 8, Label Formats and Contents.) For example, there are two file header labels, named HDR1 and HDR2; and there are two file trailer labels, called EOF1 and EOF2.

A label group contains all the adjacent label sets required to define the following file or the preceding file. Each label group is separated from the file data by a tape mark. A tape mark also separates an end–of–file label group from the following beginning–of–file label group. The label groups along with the labels and label sets they contain are shown in Figure 5–3.

Figure 5–3: Label Groupings



Note that at the beginning of a volume, the file–header label set belongs to the beginning–of–volume label group.

Also note that in most cases a label set corresponds to a label group. The nonsupport of user–label sets on TOPS–10 systems causes this situation. For example, the ANSI–specified user–header label set, if implemented with TOPS–10, would belong to the beginning–of–file or beginning–of–file–section label group along with the file–header label set.

5.3.2.2 Tape Label Sequences — The sequence of the various tape labels differs according to the number of files contained on a volume and whether a file is continued onto another tape. The volume set variations, which determine the label sequences, are listed below:

- Single file on a single volume
- Single file continued onto one or more volumes.
- Multiple files on a single volume.
- Multiple files on multiple volumes.

Figures 5–4 through 5–7 illustrate the tape label sequencing possibilities based on these variations. In the figures, IRG stands for interrecord gap.

Note that a tape ends with either an end–of–file tape label or an end–of–volume tape label; but never both. If it ends with an end–of–file label, the last file on the tape is not continued onto another tape, and this volume is either the last volume or the only volume in a set. If the last label on a tape is an end–of–volume label, the last file on the tape is continued onto another volume in the set.

Figure 5–4: Single File on a Single Volume

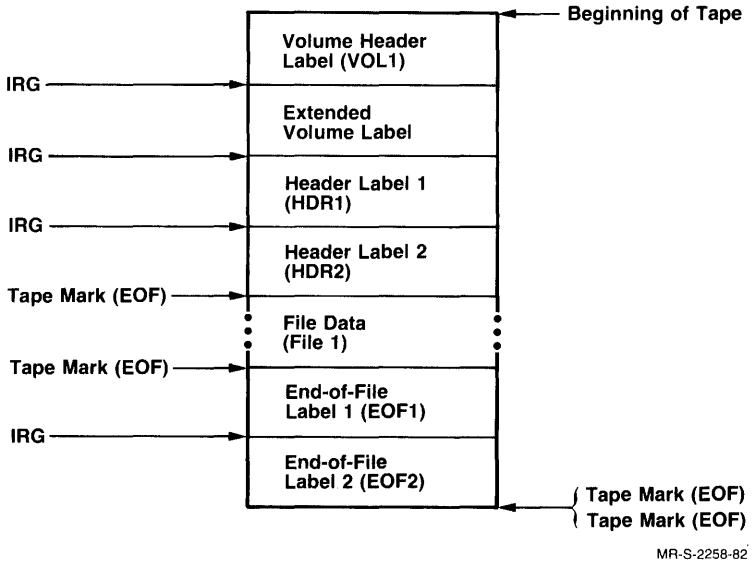


Figure 5–5: Single File on Multiple Volumes

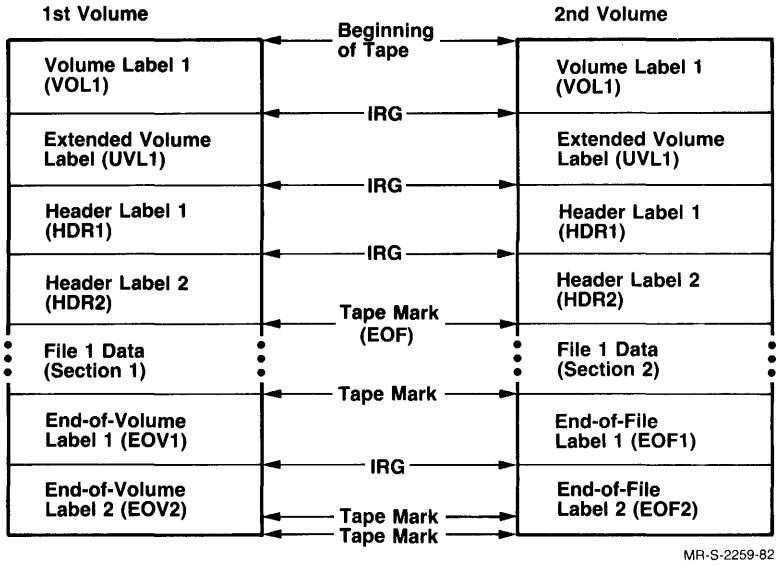


Figure 5–6: Multiple Files on a Single Volume

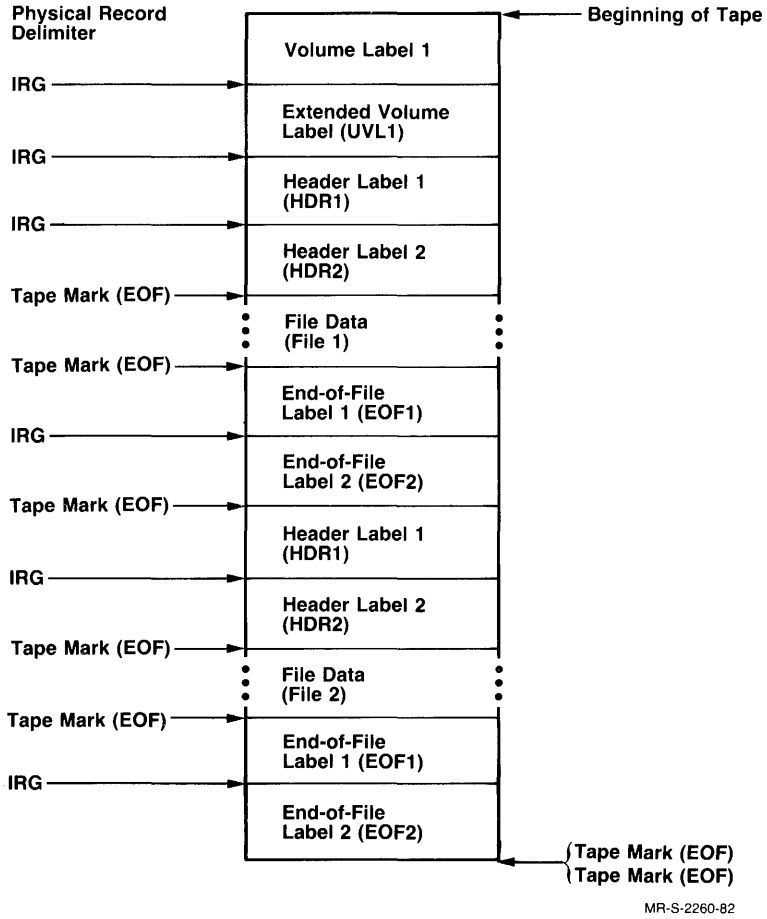
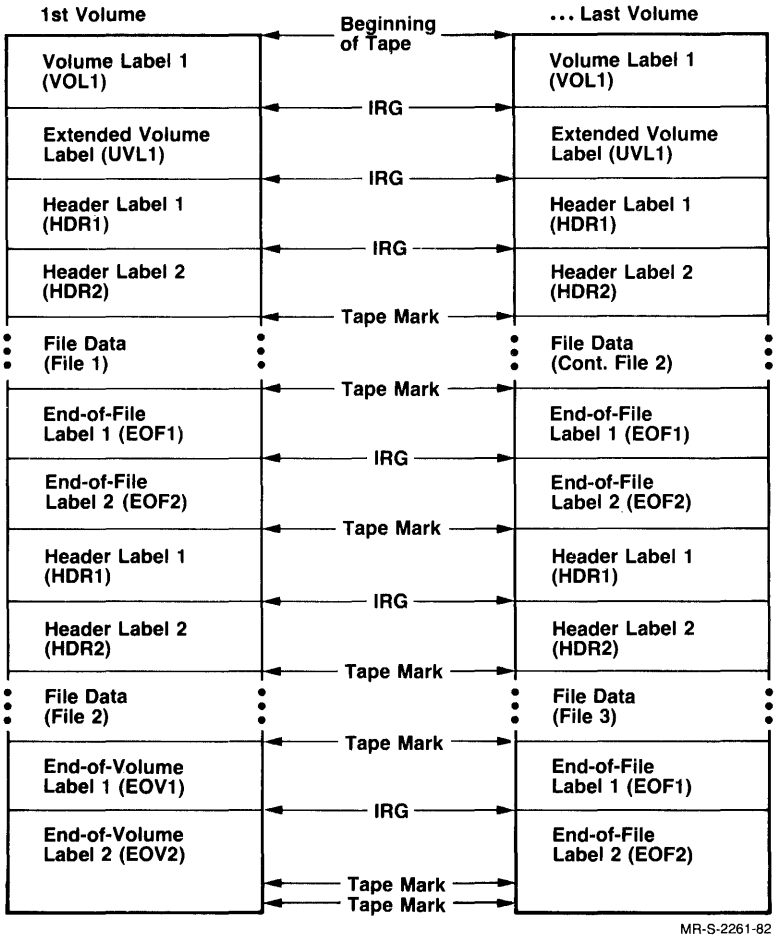


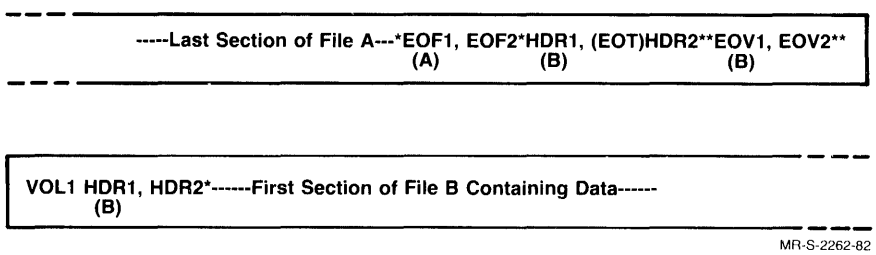
Figure 5–7: Multiple Files on Multiple Volumes



Several situations cause tape labels to be written in sequences that differ from those shown in Figures 5–4 through 5–7:

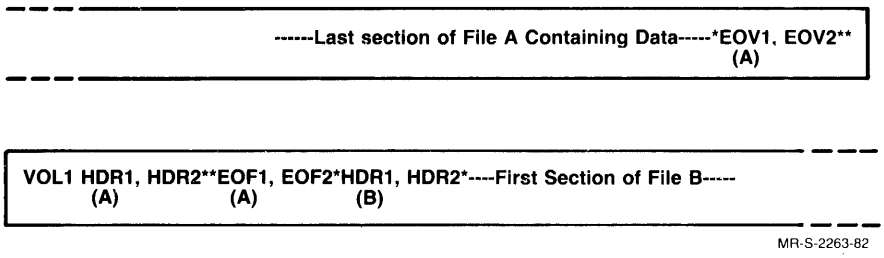
- If the system senses the end–of–tape marker while it is writing the beginning–of–file label group, it finishes the beginning–of–file label group but does not write any data blocks in this file section. Instead, it closes the volume and "continues" the file on the next volume. The file section number field of HDR1 is 1 on the first volume for this file and 2 on the continuation volume. (See Figure 5–8.)

Figure 5–8: Coincidence of Beginning–of–File Group and End–of–Tape Marker



- If the system senses the end–of–tape marker while it is writing the last data block of a file, it completes the data block and closes the volume. Then it continues the file onto the next volume even though there are no more data blocks to be written for this file. An end–of–file label group follows the empty file section on the new volume. (See Figure 5–9.)

Figure 5–9: Coincidence of Last Data Block and End–of–Tape Marker



- If the system senses the end–of–tape marker while it is writing the end–of–file label group and the file is not the last file of the set, it finishes the end–of–file label group and writes the beginning–of–file–section label group for the next file. It then closes the volume without writing any data blocks for the new file. The file is "continued" onto the next volume. The result of this situation is the same as the first situation described above.

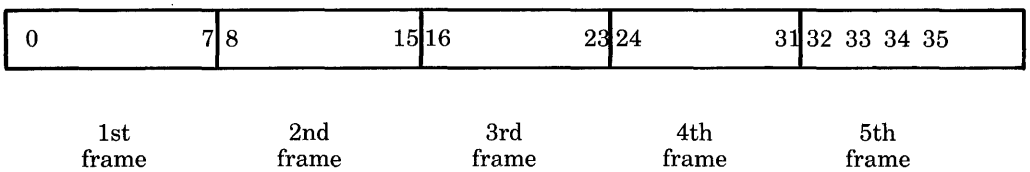
Chapter 6
Magnetic Tape Data Modes

Chapter 6 describes how data is stored on magnetic tapes in each of the data modes. These are hardware data modes that relate to a tape's physical structure. They can be set with the .TFMOD function of the TAPOP. monitor call (Code 1007/2007). (Refer to Chapter 11, The TAPOP. Monitor Call, for information on how to set the desired data mode.)

Sections 6.1 through 6.4 describe the placement of data bits on tape tracks and indicate the number of frames required to store a word of data. The diagrams in these sections are logical representations of data on a magnetic tape. Actually, the parity frame is in the center of the tape, and the order of the other frames is not necessarily as pictured.

6.1 DEC–Compatible Core Dump Mode (7– or 9–Track)

Code 0 (.TFMDD) sets DEC–compatible core dump mode for either 7–track or 9–track magnetic tape (MTAPE 100 also sets this mode.) The choice of types depends on the magnetic tape unit used. This data mode function code is the equivalent of code 1 (.TFMID) for 9–track tapes and code 5 (.TFM7T) for 7–track tapes. On a 9–track tape, DEC–compatible core dump mode stores one 36–bit word of data in five frames. Note that the last frame is only half used. One word actually requires four and one–half frames. Each data word in 9–track DEC–compatible core dump mode is formatted as shown below.



For each data word in memory, there are five tape bytes per 36–bit word, with parity bits unavailable to the user. Bytes are read and written on the tape as shown below.

TRACKS									
9	8	7	6	5	4	3	2	1	
B0	B1	B2	B3	B4	B5	B6	B7	P	FRAMES
B8	B9	B10	B11	B12	B13	B14	B15	P	
B16	B17	B18	B19	B20	B21	B22	B23	P	
B24	B25	B26	B27	B28	B29	B30	B31	P	
0	0	0	0	B32	B33	B34	B35	P	

When writing on TM10s, bits 30 and 31 are written twice. First they are written as shown in the diagram above, and then they are copied in tracks 7 and 6 of byte 5. Tracks 9 and 8 of byte 5 remain zero. When writing with a DX10, TM02–C, or TM10–C, bits 30 and 31 are written only once. Frames 9, 8, 7, and 6 contain zero.

When reading from a TM10, parity bits 9 and 8 of frame 5 are ignored. Frame 4, tracks 2 and 3 are ORed with Frame 5, tracks 6 and 7. These are bits 30 and 31 of the data word. On a DX10, TM02–C, and TC10–C, the parity bits along with frames 9, 8, 7, and 6 are ignored.

In 7–track core dump mode, one 6–bit byte is stored in each frame of the tape. Six frames are required to store one 36–bit word. This mode is the only hardware mode supported for 7–track magnetic tapes.

Each data word in 7–track DEC–compatible core dump mode is formatted as shown below.

0 5	6 11	12 17	18 23	24 29	30 35
1st frame	2nd frame	3rd frame	4th frame	5th frame	6th frame

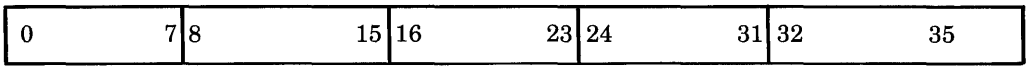
Bits are read and written on the tape as shown below.

TRACKS							
7	6	5	4	3	2	1	
B0	B1	B2	B3	B4	B5	P	FRAMES
B6	B7	B8	B9	B10	B11	P	
B12	B13	B14	B15	B16	B17	P	
B18	B19	B20	B21	B22	B23	P	
B24	B25	B26	B27	B28	B29	P	
B30	B31	B32	B33	B34	B35	P	

6.2 Eight–Bit Mode

Code 2 (.TFM8B) sets 8–bit mode for 9–track tapes. This mode can also be set with MTAPE channel, 101. This mode is compatible with any machine that reads and writes 8–bit bytes (for example, System 360/370 and PDP–11s). When a read operation is performed, four frames (that is, 8–bit bytes) are read into each word (left justified), with the remaining four bits of the word containing zeroes or copies of the parity bits. The TM02 and TM10–C copy the parity bits. The DX10 and TC10C return zeroes in these four bits. The information read into bits 32 through 35 is not user data. On a write operation, the leftmost four 8–bit bytes of each word are written out in four frames; the remaining 4 rightmost bits (32 through 35) are ignored. Only bits 0 through 31 are written onto the tape.

Each data word in 9–track industry–compatible core dump mode is formatted as shown below.



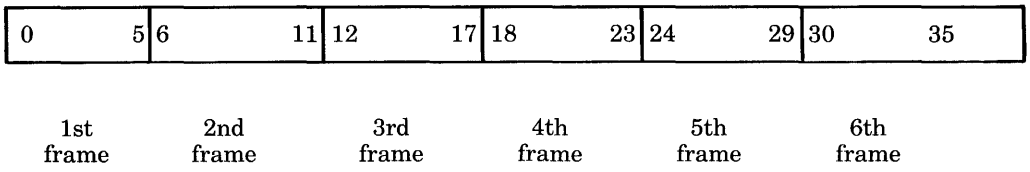
Bits are read and written on the tape as shown below.

TRACKS									
9	8	7	6	5	4	3	2	1	
B0	B1	B2	B3	B4	B5	B6	B7	P	FRAMES
B8	B9	B10	B11	B12	B13	B14	B15	P	
B16	B17	B18	B19	B20	B21	B22	B23	P	
B24	B25	B26	B27	B28	B29	B30	B31	P	

When using industry–compatible mode and buffered I/O, your program must insert the byte size in the byte pointer before issuing the first IN, INPUT, OUT, or OUTPUT monitor call.

6.3 Sixbit Mode

CODE 3 (.TFM6B) sets 6–bit (SIXBIT) mode for 9–track magnetic tapes. This mode is available only on TU70s. The format of the data word is shown below.



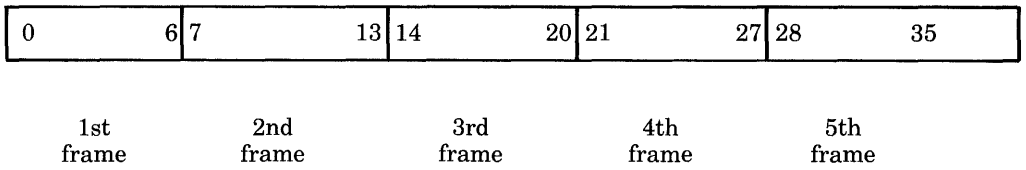
Bits are read and written on the tape as shown below.

TRACKS									
9	8	7	6	5	4	3	2	1	
0	0	B0	B1	B2	B3	B4	B5	P	FRAMES
0	0	B6	B7	B8	B9	B10	B11	P	
0	0	B12	B13	B14	B15	B16	B17	P	
0	0	B18	B19	B20	B21	B22	B23	P	
0	0	B24	B25	B26	B27	B28	B29	P	
0	0	B30	B31	B32	B33	B34	B35	P	

6.4 ANSI–ASCII Mode

Code 4 (.TFM7B) sets 7–bit mode, called ANSI–ASCII (not available on TM10s and TC10–Cs). This mode stores one 7–bit ASCII byte in each frame of the tape. This mode is useful for transferring ASCII data from TOPS–10 systems to 8–bit byte–oriented machines (for example, PDP–11s, System 360/370). Data must be stored in ANSI–ASCII mode to meet ANSI standards. Five left–justified (in core) 7–bit bytes are stored in five frames on the magnetic tape. Bit 35 must be zero to conform to ANSI standards. Bit 35 is written into the high–order bit of the last frame of each word. The other high–order bits are set to zero on write operations. When the tape is read, all five high–order bits are 0Red, and the result is stored in bit 35.

Each data word in ANSI–ASCII mode is formatted as shown below.



Data is read and written on the tape in the following format.

TRACKS									
9	8	7	6	5	4	3	2	1	
0	B0	B1	B2	B3	B4	B5	B6	P	FRAMES
0	B7	B8	B9	B10	B11	B12	B13	P	
0	B14	B15	B16	B17	B18	B19	B20	P	
0	B21	B22	B23	B24	B25	B26	B27	P	
B35	B28	B29	B30	B31	B32	B33	B34	P	

Chapter 7
Record Formats

The record formats used on a labeled tape conform to the ANSI standard. There are three formats available to meet the diverse needs of your applications and hardware. Sections 7.2 through 7.5 describe the various record formats. Note that records of only one format type can be recorded in the same file.

7.1 Blocking of Records

Logical records may be blocked; that is, more than one logical record may reside in a physical tape block. Records are usually blocked when the physical block is at least twice the logical block (record) size, and the record type is either fixed or variable. Spanned records are also likely to be blocked, no matter what the ratio is between block size and record size. In short, a blocked record is a record that is not equal in size to the block. An unblocked record is equal in size to the block.

The ANSI standard allows you to distinguish between the demands of the hardware and those of your program by allowing you to specify the physical block size independent of the logical record structure. The block size you select should be optimal for the recording density, the drive type, and the hardware interface necessary to transmit the data. You should select the logical record size and type so that your program can process the data as efficiently as possible.

NOTE

Your program must perform its own blocking, deblocking, and data translation operations. The system does not automatically perform these functions.

Also note that neither the TOPS–10 monitor nor the label processor enforces the characteristics specified in the tape labels. Thus, for example, if your program attempts to read or write a logical record of a size different from that specified, it receives no error. You are free to read or write data in any format.

7.2 Fixed–Length Records (F Format)

With fixed–length records, the logical record length is the same for all records. The block length is also the same for all blocks. In addition, the block length must be an integral multiple of the record length. The blocking factor for fixed format is 1 (for unblocked records) or greater than 1 (for blocked records).

Fixed–length record blocks may contain fewer records than the possible maximum. Thus, even though the block length is the same for all blocks, you do not need to completely fill all blocks. For example, your program might write its last record in the middle of a block.

Figures 7–1 and 7–2 illustrate the formats of fixed–length records.

Figure 7–1: Fixed–Length Records (F), Unblocked

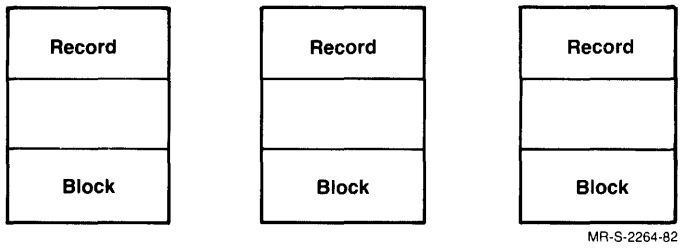
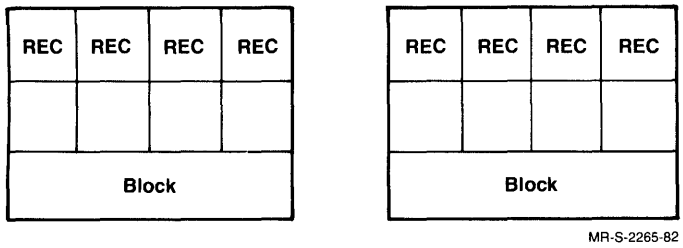


Figure 7–2: Fixed–Length Records (F), Blocked



7.3 Variable–Length Records (D Format)

Variable–length records provide an increased level of sophistication over fixed–length records. With variable–length records, your program can make a decision to block or not block as each record is produced. You should use variable length when the record size is less than or equal to the block size, or when records are not of uniform length.

With variable–length records, a record control word (RCW) precedes each record. The RCW consists of four characters. The record length (in characters), including the length of the RCW, is expressed as a decimal number occupying the entire RCW. The RCW is necessary so that a program reading the data can find out the length of each record. If your file is blocked, each block may contain multiple records, each of which is preceded by its own RCW. Because variable–length records are not of uniform length, the number of records that are in each block can vary from one block to another. Also, the block sizes can vary from one block to another. Figures 7–3, 7–4, and 7–5 illustrate the formats of variable–length records.

Figure 7–3: Variable–Length Records (D), Unblocked

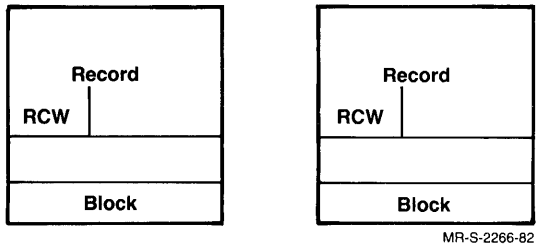


Figure 7–4: Variable–Length Records (D), Blocked

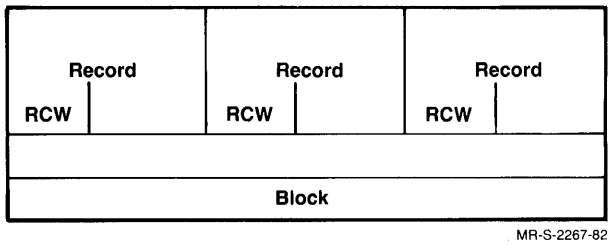
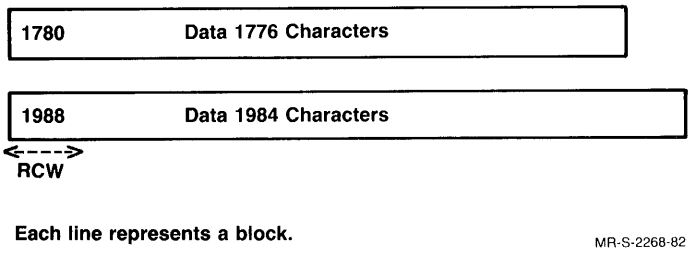


Figure 7–5: Unblocked Variable–Length Records



7.4 Spanned Records (S Format)

The logical record size that you wish to use could be larger than the ideal physical block size for the device. To accommodate this situation, you could use spanned records. A spanned record is one that may occupy more than one physical tape block, and may also occupy tape blocks on different volumes. Varying length blocks are permitted for segments of spanned records.

With spanned records, the first five characters of each segment make up the segment control word (SCW). The first character of the SCW is called the Spanning Indicator. It has the following meaning:

0– record begins and ends in this segment

1– record begins but does not end in this segment

2– record neither begins nor ends in this segment

3– record ends but does not begin in this segment

The next four characters of the SCW indicate the segment length (in characters), including the SCW. This value is expressed as a decimal number.

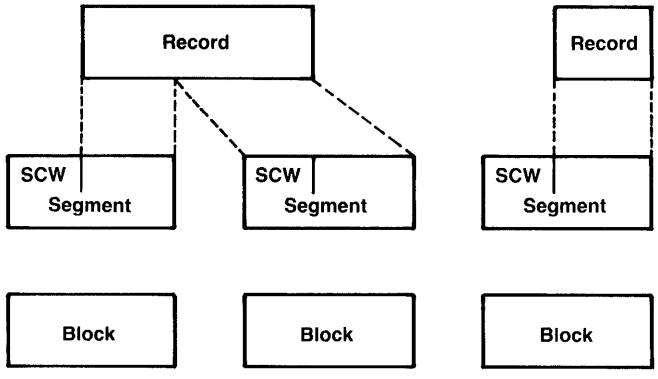
For spanned records, there is no explicit record control word that expresses the total record length.

These records may span volumes. The record length is unbounded in that there is no limit to the number of segments in one record. But, there is only one segment of the same record in a block.

The segments of a record are written in consecutive order. That is, segment n+1 is written immediately following segment n. No segments from other records are interspersed.

Figures 7–6, 7–7, 7–8, and 7–9 illustrate spanned record formats.

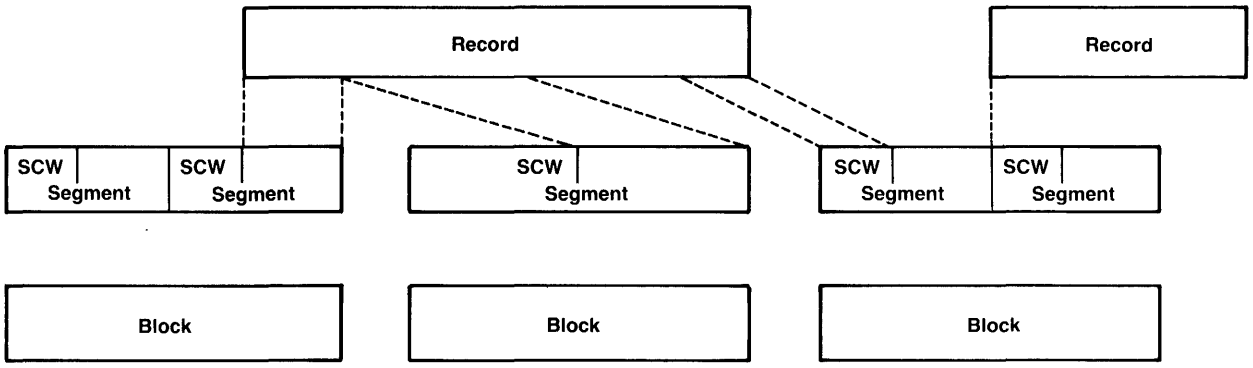
Figure 7–6: Spanned Records (S), Unblocked



The first block shows the maximum blocksize.

MR-S-2269-82

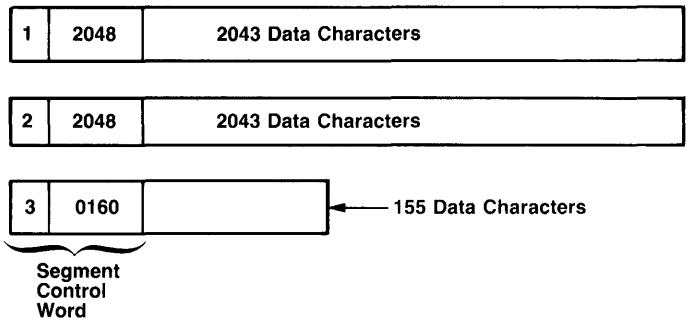
Figure 7–7: Spanned Records (S), Blocked



All blocks show the maximum blocksize;
the last record continues in the next subsequent block.

MR-S-2270-82

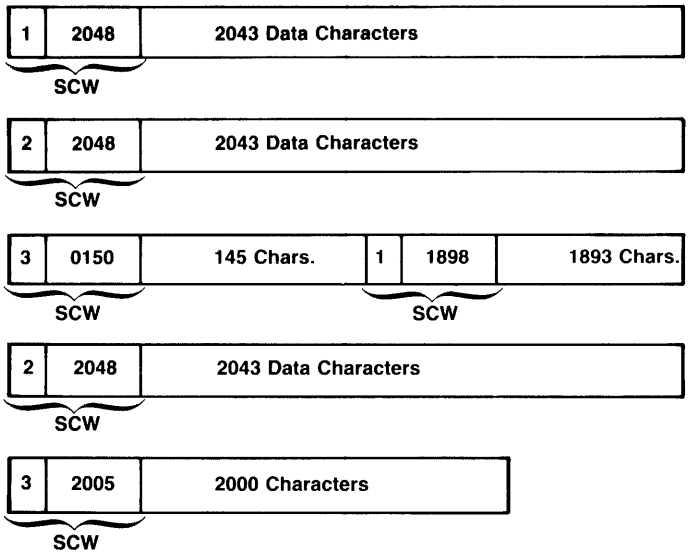
Figure 7–8: Unblocked Spanned Record



The record lenght is 4241 characters;
each line represents a block.

MR-S-2271-82

Figure 7–9: Blocked Spanned Records



Record 1: Length of record is 4231 characters.
Record 2: Length of record is 5936 characters.
Each line represents a block.

MR-S-2272-82

7.5 Undefined Records (U Format)

When records do not meet the specifications described in Sections 7.2, 7.3, or 7.4, the records are undefined in format.

Chapter 8
Label Formats and Contents

This chapter describes DEC– (ANSI–) and IBM–formatted tape labels. All DEC–formatted labels follow the ANSI Tape Label Standard; therefore, the terms DEC–formatted and ANSI–formatted are equivalent. There are some fields defined in the ANSI Tape Label Standard as being reserved for operating system information. The DEC–formatted tape labels illustrated and described in this chapter define the contents of these fields to agree with the contents of the DIGITAL Tape Label Standard. Any differences between DEC–formatted tape labels and IBM–formatted tape labels are described in the text of this chapter.

The types of tape labels are listed below. These are described in the remainder of this chapter.

- Volume header label.
- Extended volume header label.
- File header label 1.
- File header label 2.
- End–of–file label 1.
- End–of–file label 2.
- End–of–volume label 1.
- End–of–volume label 2.

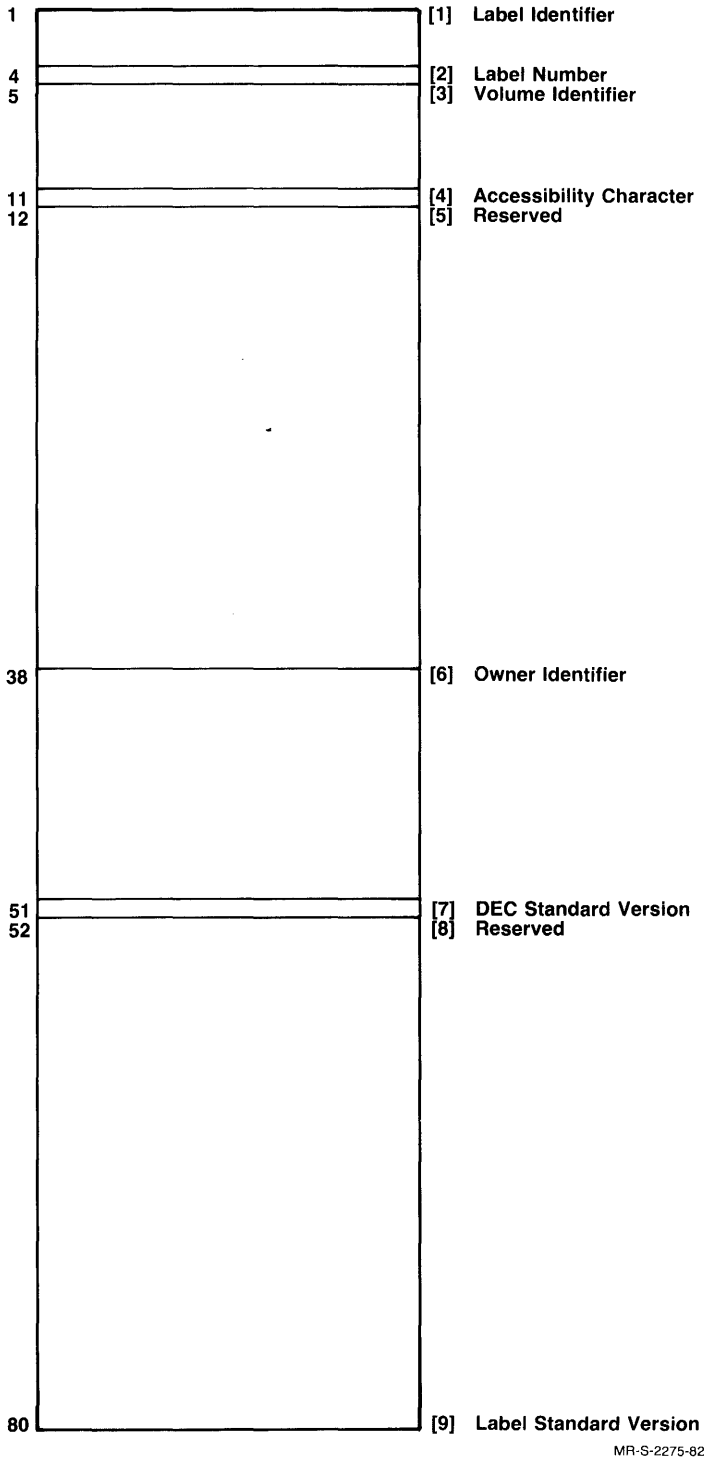
Each label is an 80–character record on the magnetic tape. On the following pages, each type of label is illustrated with the 80 characters divided into information fields. Solid lines separate these fields from one another. The field name appears on the right side of the illustration. The numbers on the left side of the illustration indicate the first character position of

that information field within the tape label. And the numbers within square brackets on the right side of the illustration correspond to the textual description following each tape label illustration.

Note that fields in the EOF and EOV label sets match those in the HDR label set. This allows identical label processing when reading a file backward.

All labels contain a label identifier and label number in addition to the various control information fields. The first three (1–3) characters form the identifier field; and the fourth character, a numeric, indicates the position of the label within the set. (Refer to Section 5.3.2.1, Classification of Labels, for a description of label sets.) Characters 5 through 80 make up the control information fields.

Volume Header Label (VOL1)



The VOL1 label is always the first label on a magnetic tape reel. Its main function is to facilitate the interchange of tapes among installations by identifying the sender.

1. The Label Identifier field always contains the three ASCII characters VOL.
2. The Label Number field always contains the character 1.
3. The Volume Identifier field consists of a combination of legal characters (A through Z and 0 through 9), left–justified. There is a restriction that all of the character positions cannot be zeros. You specify this field with the VOLID switch in the MOUNT command. The operator initially sets this field when initializing the tape; refer to Chapter 2 for a description of initialization. Each VOLID should be unique, but there is no check for uniqueness.
4. The Accessibility Character field contains a space or the character 1, and indicates whether the volume is protected. It should contain a space for any tapes that are to be interchanged between systems. Note that if the operator specifies a protection code of 0 during initialization, the accessibility character is zero or blank. For DEC–formatted tape labels, this field is always 1 if the operator specifies a protection code. For IBM–formatted tape labels, this field can be 0, 1, or 3. A 0 indicates that the volume is not protected; a 1 indicates that the volume is write–protected; a 3 indicates that the volume is read–protected.
5. Reserved fields in the tape label consist of spaces.
6. The Owner Identifier field is ignored when reading IBM–formatted tape labels and is not defined in the ANSI standard. For DEC–formatted tape labels, the Owner Identifier field is divided into the following parts:

   character 38 — is always D.

   character 39 — is always %.

   character 40 — is A, meaning the DECsystem–10. For other digital systems, character 40 is as follows:

Table 8–1 Machine Codes

Code	Machine
8	PDP–8
B	PDP–11
C	VAX–11
F	PDP–15
K	DECSYSTEM–20

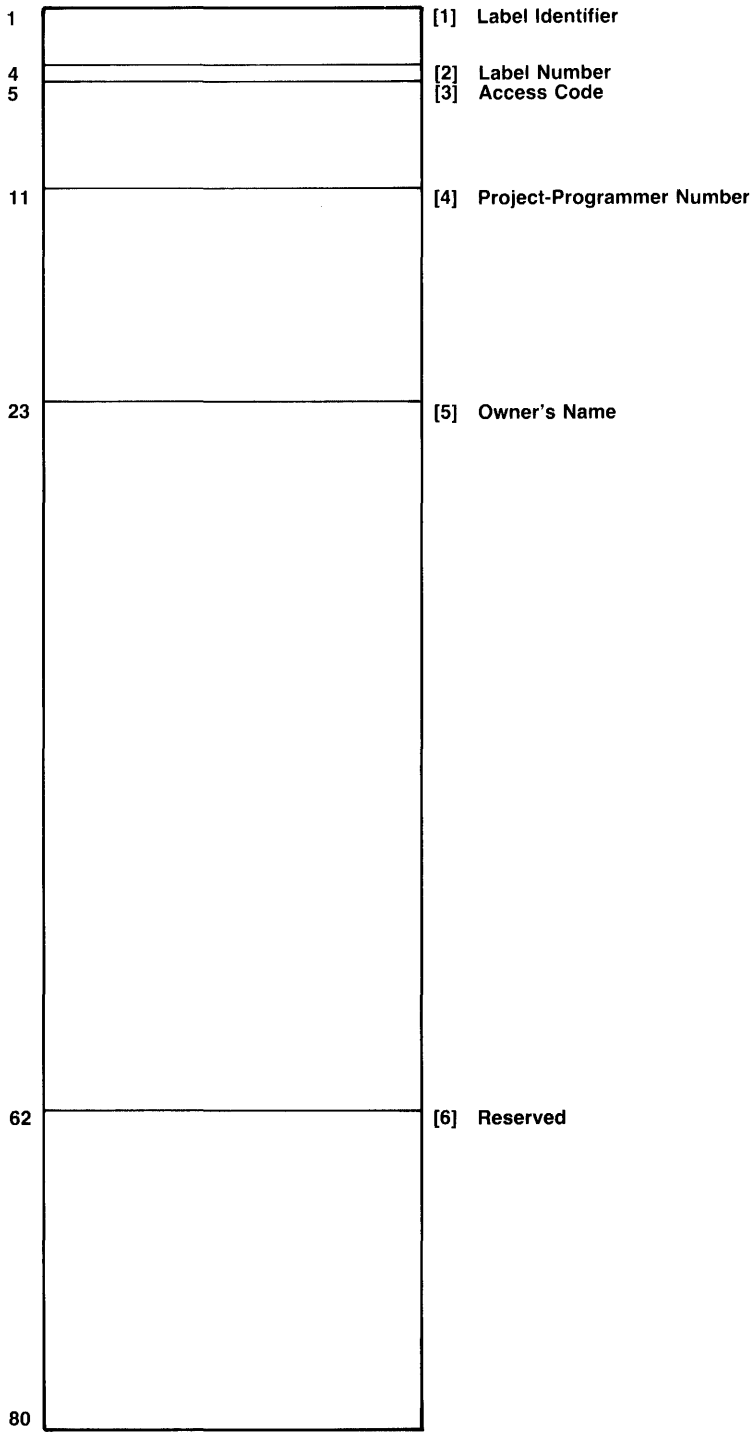
characters 41–45 are system–dependent; with TOPS–10 they make up the operating system code, which is T10 for TOPS–10.

characters 46–50 are system–dependent; with TOPS–10 they make up the system serial number, which the system administrator sets at monitor generation time through MONGEN. This number is decimal and is right–justified.

The label processor ignores this field if the first two characters are not D%.

7. The DEC Standard Version field is not used for IBM– and ANSI–formatted tape labels. For DEC–formatted tape labels it is "1".
8. Reserved fields in the tape label consist of spaces.
9. The Label Standard Version field is the version number (currently 3) of the ANSI label standard. This field is not used for IBM–formatted tape labels.

Extended Volume Header Label (UVL1)



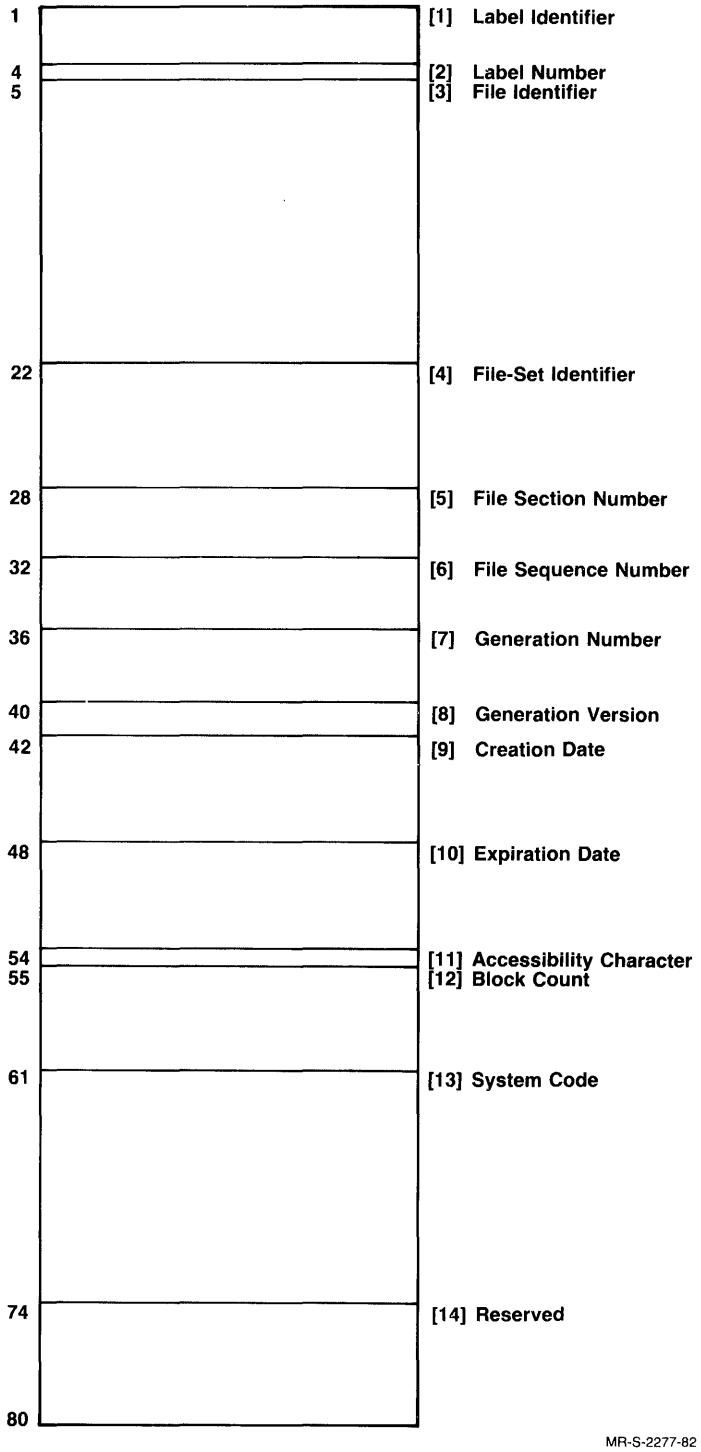
The UVL1 label is always the second label on the magnetic tape reel for tapes created on the TOPS–10 system. It supplements the VOL1 label, providing additional access information for the volume. This label is processed on input only if the Owner Identifier field in the VOL1 label contains D%A. UVL1 is processed on output for all labeled tapes.

On TOPS–10 systems, this label is treated as the second volume label, not as a user label. Because the system does not support user labels, any UVL label beyond the first one of the set is ignored.

1. The Label Identifier field always contains the ASCII characters UVL.
2. The Label Number field always contains the character 1.
3. The Access Code field contains the protection code for this volume. The operator specifies the volume protection code during initialization.

   Note that if this field contains zero, the label processor sets the Accessibility Character in the Volume Header Label (VOL1) to space for DEC–formatted tapes or zero for IBM–formatted tapes. Also, if this field contains zero, the label processor does not protection–check any of the files; the label processor allows access to all the files. You must set this field to zero if you are planning to interchange tapes between systems.
4. The Project–programmer Number field contains the project–programmer number associated with the owner of the files on this magnetic tape. The operator sets this field when initializing the tape.
5. The Owner's Name field contains the name associated with the file owner when he/she logged onto the system. The operator initially sets this field when initializing the tape.
6. Reserved fields in magnetic tape labels consist of spaces.

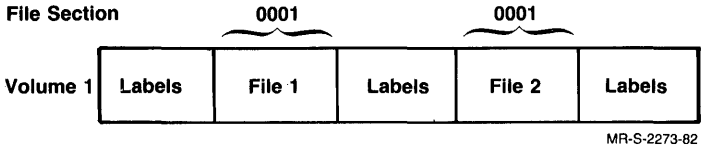
First File Header Label (HDR1)



The HDR1 label is always the third label on a magnetic tape reel. It helps the system identify, protect, and audit a file. There is one HDR1 label for each file or each section of a file contained on a reel. (Refer to Section 5.3.2.2 for information on tape label sequences for multiple files on a single reel or on multiple reels.)

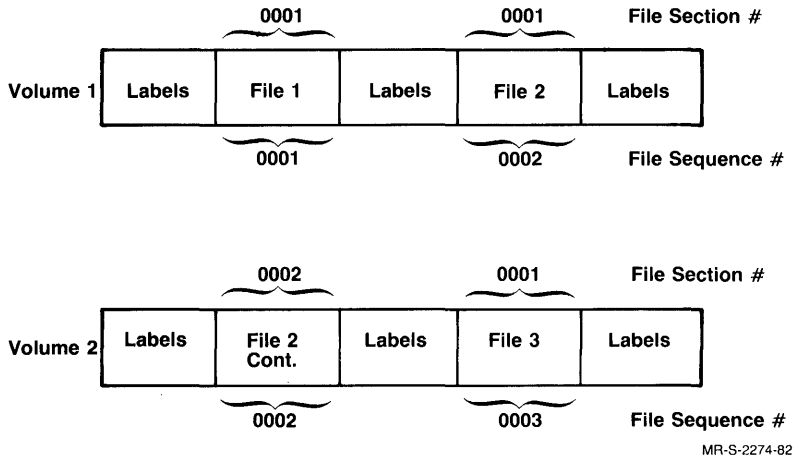
1. The Label Identifier field always contains the three ASCII characters HDR.
2. The Label Number always contains the character 1.
3. The File Identifier field (file name) is supplied by the label processor on output and is ignored on input. It can be set with the .TFLPR function of the TAPOP. monitor call. This field is written as a 17–character ASCII string.
4. The File Set Identifier is the VOLID of the first volume of the set. For example, if this field is the same as the VOLID in the VOL1 label of this reel, this reel is the first volume of the set. For IBM–formatted labels, this field is ignored on input, but written on output.
5. The File Section Identifier indicates whether a file is continued from another volume. A file always begins with a section number of 0001. If that file is continued onto another volume, the file header block for that section of the file has 0002 in the File Section field. For example, see Figure 8–1 below. There are two files on one volume. The section numbers of the files are written above each segment of the file.

Figure 8–1: File Section Numbers



6. The File Sequence Identifier field specifies the relative position of this file in the file set. See Figure 8–2, where there are three files on two volumes. The File Sequence Identifier field is incremented for each successive file in the set.

Figure 8–2: File Sequence Numbers



7. The Generation Number field always contains the character 1. The label processor ignores this field on input, but writes it on output.

8. The Generation Version field always contains 0. The label processor ignores this field on input but writes it on output.

9. The Creation Date field contains the Julian date on which the file was created, in the form yyddd. The label processor ignores this field on input, but writes it on output.

10. The Expiration Date field specifies the date after which this file and all subsequent files in the set can be overwritten. You can specify this date with the TAPOP. monitor call. If you do not specify the expiration date, the label processor assumes the creation date is the expiration date. If the current date is equal to or later than the expiration date, you can overwrite this and all following files.

11. The Accessibility Character field specifies whether this file will be protection–checked. For DEC–formatted labels, this character can be a space or a 1. A space indicates that all accesses will be allowed to the file; there will be no protection checking. A 1 indicates that the file will be protection–checked according to the protection code in the HDR2 label. For interchanging tapes between systems, this field should be blank and there should be a 0 protection code.

    For IBM–formatted tapes, this character can be 0 or 1. If it is 0, all accesses are allowed to the file; there is no protection checking. The code is checked only when the tape is written on a TOPS–10 system. If this field is 1, no access is allowed to the file. If interchanging tapes between systems, this field must be 0.

12. In HDR1, the Block Count field always contains 0. But in EOF1 and EOV1 this field is the number of blocks recorded since the preceding beginning–of–file or beginning–of–file–section label group.

    When reading backward, you can use the EOV1/EOF1 Block Count to establish a count that is decremented by one for each block read, and is compared with the HDR1 Block Count (zero) at the beginning of the file section. By this process, you can determine whether any blocks were skipped or whether any spurious blocks were inserted during the read.

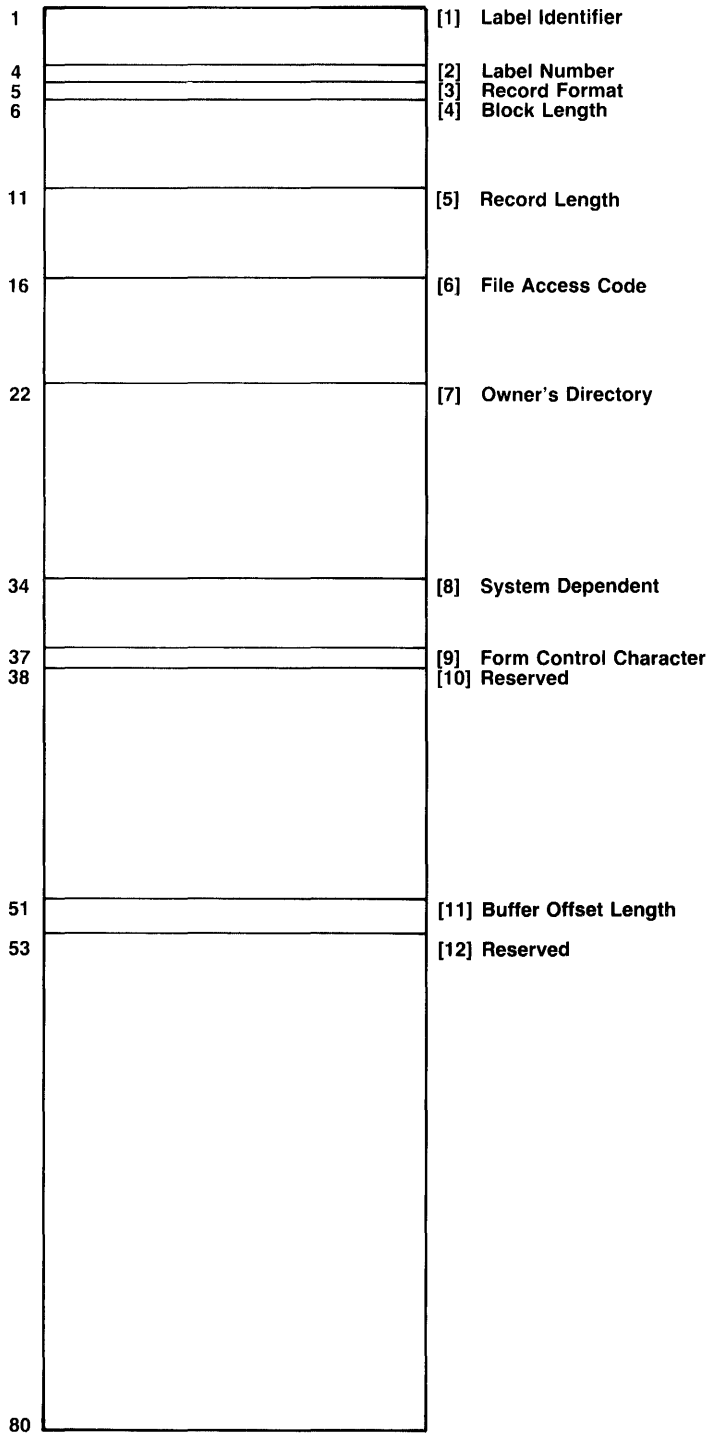
13. The System Code field is DECSYSTEM10. Codes for other DIGITAL systems are:

Table 8–2: System Codes

Code	System
DECFILE11A	RSX, IAS
DECRT11A	RT–11
DECRSTS/E	RSTS/E
DECFILE112	VMS RMS
DECSYSTEM20	TOPS–20

14. Reserved fields in labels consist of spaces.

Second File Header Label (HDR2)

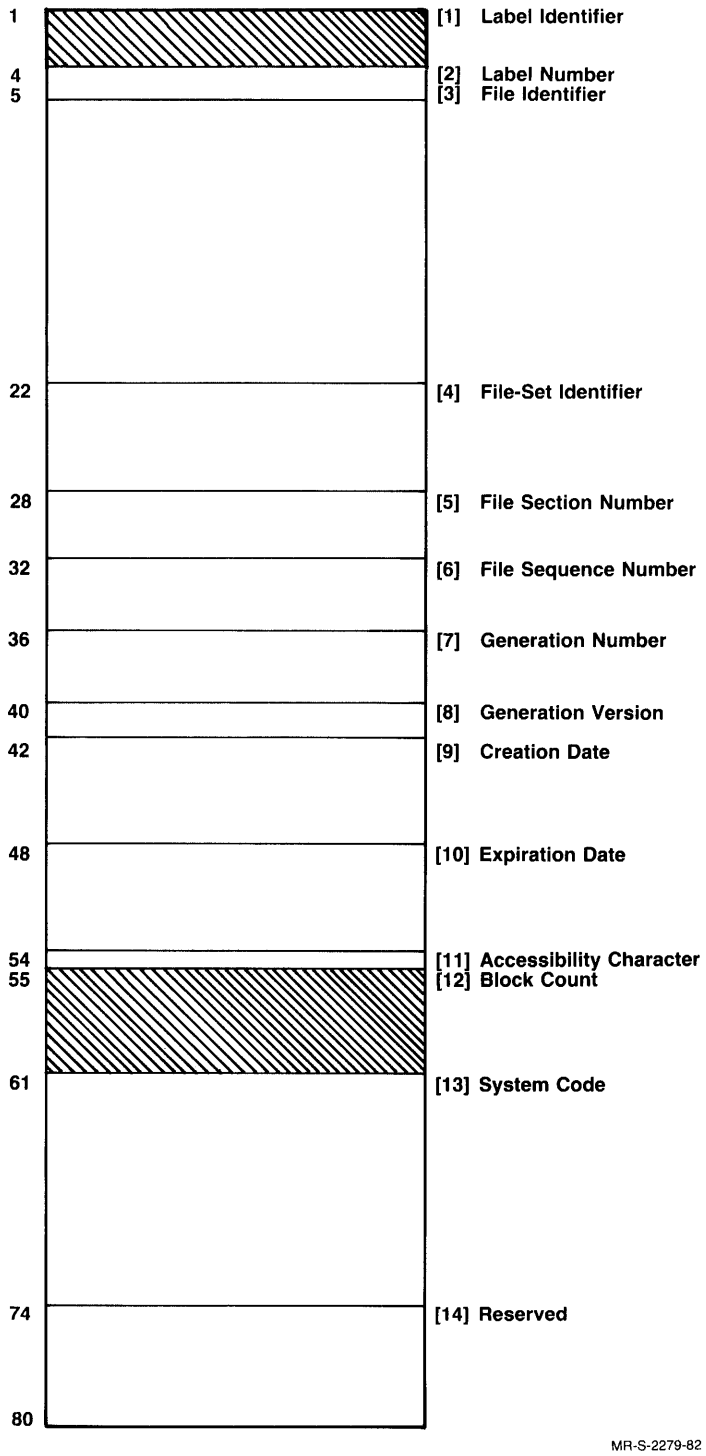


The HDR2 label is always the fourth tape label from the beginning of a magnetic tape. Also, it is always the label immediately preceding a file. HDR2 specifies the user–assigned file attributes.

1. The Label Identifier field always contains the three ASCII characters HDR.
2. The Label Number field always contains the character 2.
3. The Record Format field can contain F (fixed), D (variable), S (spanned), or U (undefined). The default format is U for both DEC and IBM records. Refer to Chapter 7 for more information on record formats.

   You can change the contents of this field with the .TFLPR function of the TAPOP. monitor call. Refer to Chapter 11 for a description of TAPOP.
4. The Block Length field contains the number of characters within a block. For DEC–formatted labels, this value can range from 0 to 99999. For IBM–formatted labels, the maximum is 32,760 characters per block.
5. The Record Length field specifies the number of characters per record. By default, this value is the same as the block length field and can range from 1 to 99999.
6. The File Access Code field contains the protection code for the file. Refer to Section 3.5 for a description of protection codes.
7. The Owner's Directory field is the project–programmer number of the file owner.
8. The System–Dependent field is unused in DEC–formatted labels. For IBM–formatted labels, it contains file usage data, which is ignored on input.
9. The Forms Control Character field contains a space, an ASCII A, or an ASCII M. A space indicates that the file contains no forms control information. That is, if carriage returns and/or linefeeds are present in the file, the label processor ignores them. An A indicates that the first character in each record of the file contains a control character. An M indicates that the file's records contain all necessary format control information. This field is processed as a space and is ignored on input if it is not a space.
10. Reserved fields in tape labels consist of spaces.
11. The Buffer Offset Length field contains the length of the information that prefixes each data block. For DEC–formatted labels, this field is written as 0, and ignored on input.
12. Reserved fields in tape labels consist of spaces.

First End–of–File Label (EOF1)



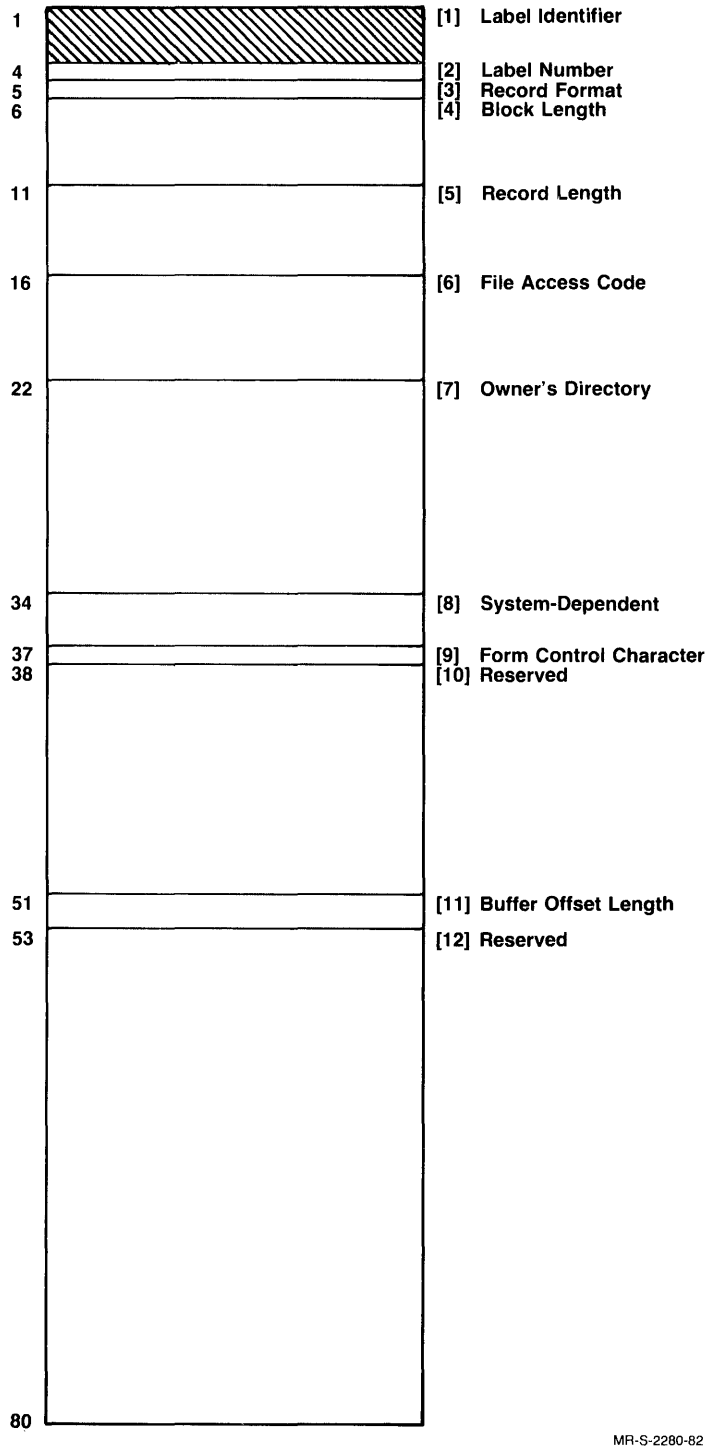
The EOF1 label is always the first tape label following a tape mark (EOF) at the end of a file. Refer to Section 5.3.2.2, Tape Label Sequences, for diagrams showing the layout of labels, marks, and data on tapes.

The EOF1 label is essentially the same as the HDR1 label. Their differences are listed below. The information fields shown in the EOF1 illustration that differ from those in HDR1 are shaded with slashes (////).

1. The Label Identifier always contains the three ASCII characters EOF.
2. The Block Count field contains the number of blocks recorded for this file section, that is, the length of this file section in blocks. This value can range from 0 to 99999.

   You can use the EOF1 Block Count field to determine, when reading a file section, whether any blocks were skipped or whether any spurious blocks were inserted. Maintain a count of blocks read and compare this count with the EOF1 Block Count.

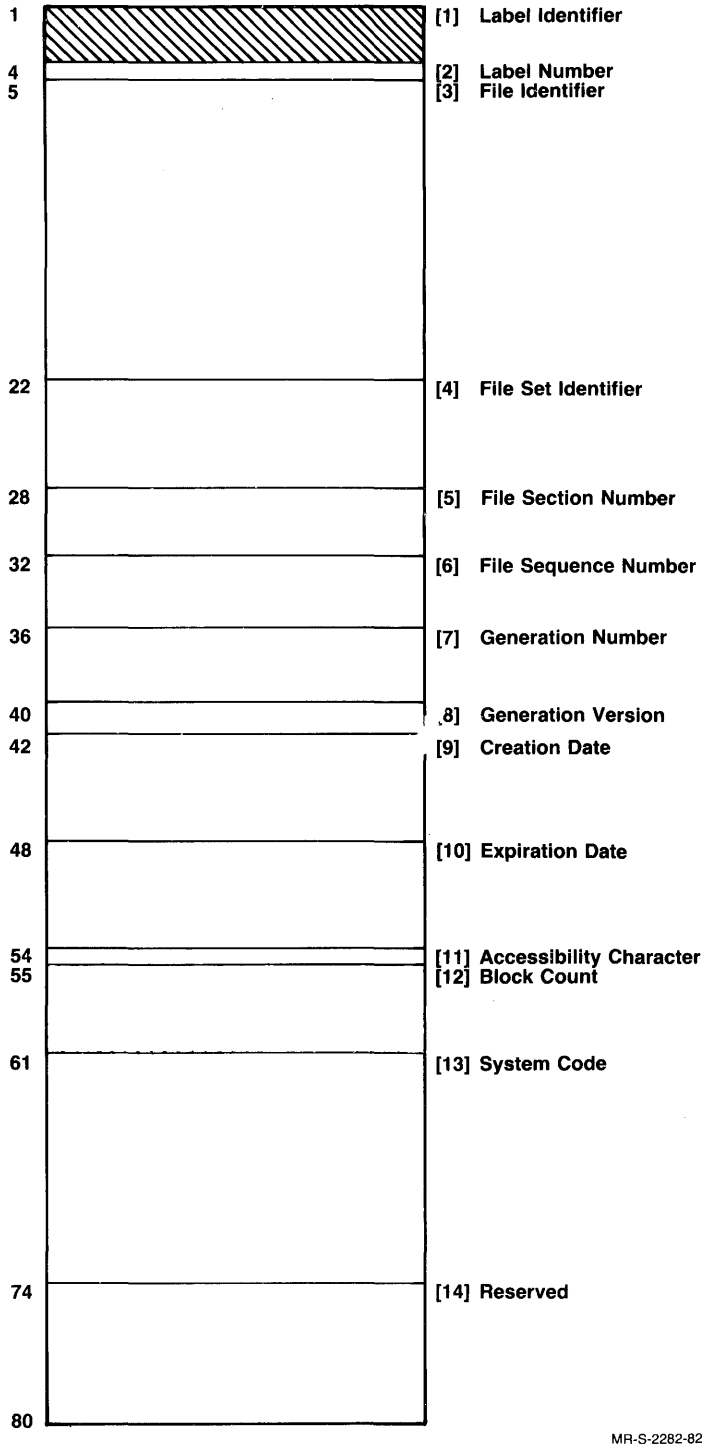
Second End–of–File Label (EOF2)



The EOF2 label always immediately follows an EOF1 label.

The EOF2 label is essentially identical to the HDR2 label. There is only one difference: the Label Identifier field (shaded above, /////) contains EOF in the second end–of–file label (EOF2).

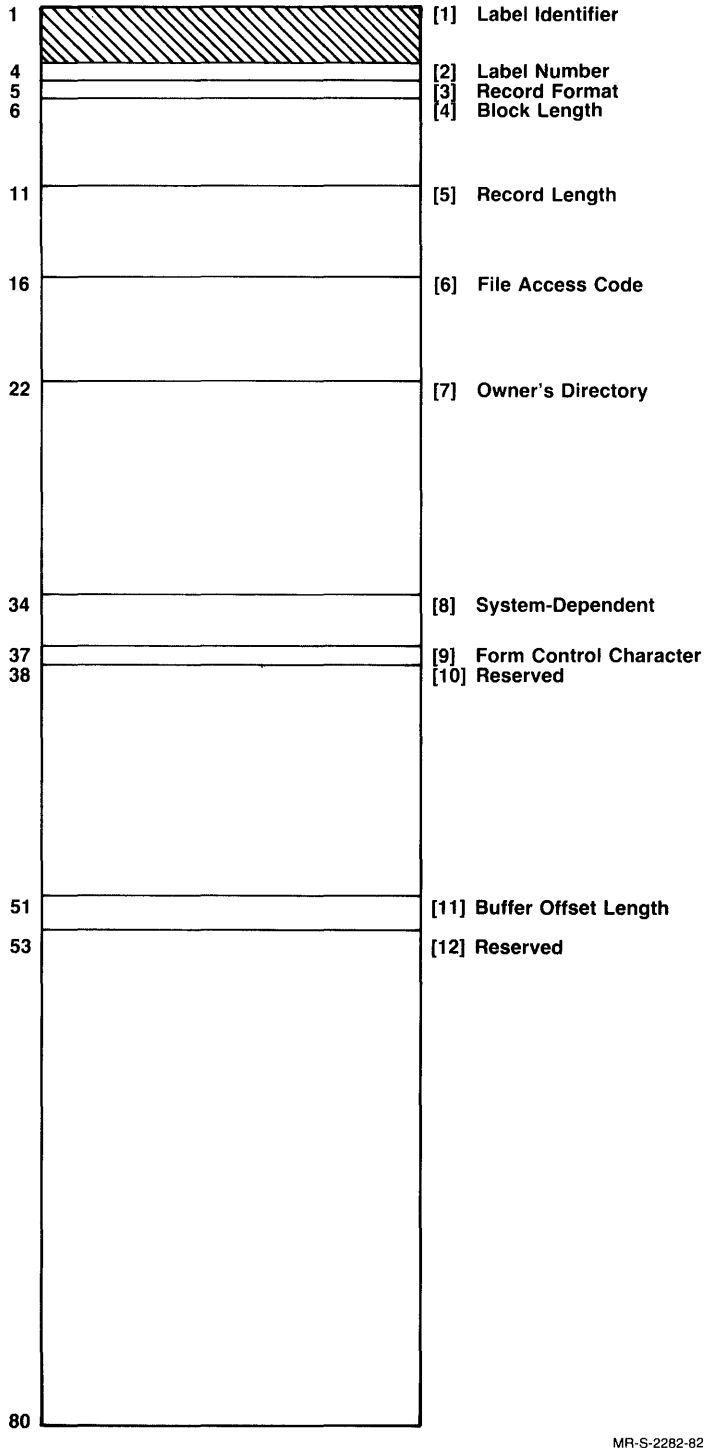
First End–of–Volume Label (EOV1)



The EOV1 label is always the first label following a file section at the end of the magnetic tape reel. That is, if end–of–file has occurred at the end of the tape, there is no EOV1 label; the EOF1 label is there instead.

The EOV1 label is essentially the same as the EOF1 label. There is only one difference: the Label Identifier field (shaded above, /////) contains EOV in the first end–of–volume label.

Second End–of–Volume Label (EOV2)



The EOV2 label is essentially identical to the EOF2 label. There is only one difference: the Label Identifier field (shaded above, /////) contains EOV in the second end–of–volume label (EOV2).

Chapter 9
The Mount Process for Labeled Tapes

9.1 Introduction

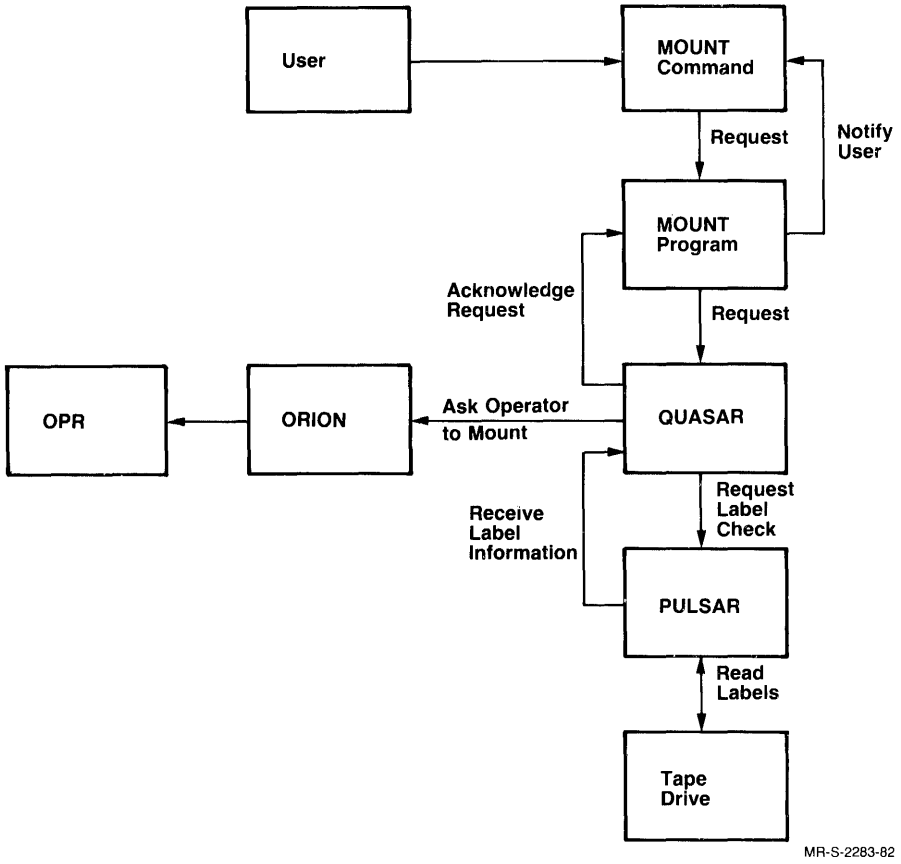
This chapter describes internal operations that are invoked by the MOUNT command and by the operator's response to the command. Automatic Volume Recognition, volume switching, and prestaging are among the topics discussed. For details on operator procedures, refer to the *TOPS–10 Operator's Guide* and to the *TOPS–10 Operator Command Language Reference Manual.*

Mount requests originate from user terminals, from batch control files, and from PULSAR. In the first two instances, the first (or only) tape of a volume set is mounted. With PULSAR requests, succeeding tapes in the volume set are mounted as needed. Sections 9.2 through 9.4 discuss the mount process in all three of these contexts.

9.2 Mounting the First Tape of a Volume Set

This section discusses the process that begins when a user issues the MOUNT command from a terminal or from a control file. Figure 9–1 gives an overview.

Figure 9–1: The MOUNT Process for the First Tape of a Volume Set



Assuming that the operator did not mount the volume before the MOUNT command was issued and that Automatic Volume Recognition is enabled, the following action takes place.

The MOUNT command sets off a cycle of events beginning with the MOUNT program's request for devices and ending with the program's notification to the user that the mount was successful. Any errors that occur break the cycle and cause appropriate messages to be sent to the operator. The operator then informs the user of the error condition.

When the MOUNT command is issued, the MOUNT program gains control. This program sends an IPCF message requesting the devices to the QUASAR program. QUASAR, in turn, sends an IPCF message to ORION, who notifies OPR. Upon receipt of ORION's message, OPR types information similar to the following on the operator's console:

```
10:55:19        -- Magtape mount request #138 --
                User: TUCKER,B [27,5342] Job# 47
                Volume-set-name: T

                Volume-ID     Write     Labels    Track    Density
                ---------     -------   ------    -----    -------
                T             Enabled   ANSI        9      1600
```

The operator then mounts the tape on an available drive and places the drive on line, at which point QUASAR, receiving notification from the monitor of an on–line interrupt, sends a message to PULSAR. This message requests PULSAR to read the volume labels.

After the labels have been read, QUASAR ensures that the mounted tape is the one that the user requested with the MOUNT command.

QUASAR also checks for other errors at this time. For example, it checks the write–ring status of the tape unit. If the user requested write–lock status, but the tape is write–enabled, or vice versa, QUASAR sends an appropriate message to the operator.

QUASAR reassigns the tape unit to the user if it foresees no conflicts then notifies the MOUNT program. Following this, a message similar to the following appears on the user's terminal (or log file):

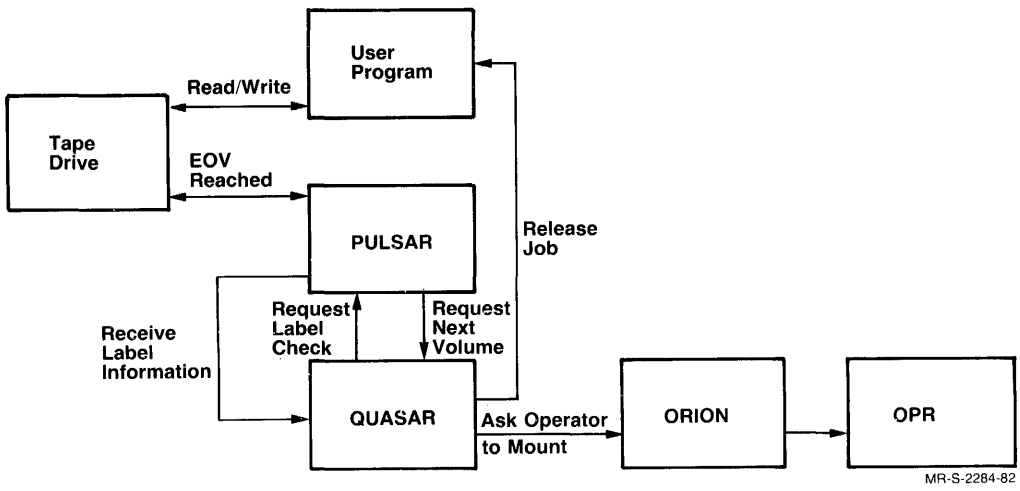
```
[Magtape T mounted on MTA261 with logical name T]
```

9.3 Mounting the 'Next' Tape of a Volume Set

Volume switching is the term that describes the action of changing from one mounted tape volume to the next one in the volume set. The operator must perform this action when a multivolume file is being read or written. This section describes what happens internally during volume switching. Figure 9–2 gives an overview of the process.

As in Section 9.2, it is assumed that the operator has not premounted the next volume and that AVR is enabled on the tape drive.

Figure 9–2: Volume Switching



When PULSAR encounters the end–of–volume label set during a tape read, it knows that this is a multivolume file. After putting the job into the event–wait (EW) state, PULSAR sends an IPCF message to QUASAR requesting the next volume; and it rewinds the recently processed volume.

The operator, by way of QUASAR, ORION, and OPR, receives notification. The message received by the operator is of the same type as that issued for the first volume of the set. After the operator mounts the requested tape on a drive, PULSAR takes the job out of the event wait state and proceeds as with the first volume of the set.

If the /NOTIFY switch was specified in the MOUNT command, a message similar to the following is sent to the user's terminal or log file:

```
From System:
Logical name T switched to volume B on MTA261
```

When PULSAR encounters the end–of–tape marker during a tape write, it writes the end–of–volume label set; asks QUASAR for the next volume as it does when reading a file; and updates fields in the beginning–of–volume label group on the newly mounted tape (and writes new labels if necessary). After this, the writing of file data can continue.

9.4 Automatic Volume Recognition

Automatic Volume Recognition (AVR) performs the following functions:

- Detects that a tape unit has come on line
- Verifies the tape on the unit by reading the labels
- Automatically assigns the newly verified tape to a waiting job

Section 9.2, Mounting the First Tape of a Volume Set, and Section 9.3, Mounting the 'Next' Tape of a Volume Set, introduced the discussion of these AVR functions. This section discusses AVR in more depth. It describes operations for premounted tapes and for tapes that satisfy multiple pending requests. The hardware aspects of AVR are also described in this section.

With AVR enabled, no operator intervention is required beyond mounting a labeled tape.

The operator enables and disables AVR with the commands:

ENABLE VOLUME–RECOGNITION arg
DISABLE

These commands can be given for all tape drives or for a specific tape drive. AVR should remain enabled except during initialization of new tapes. Disabling at this time (and specifying the /OVERRIDE–EXPIRATION:YES switch) prevents the tapes from 'running away' when PULSAR tries to verify labels.

9.4.1 Hardware

AVR can be fully utilized only on tape drives that generate on–line interrupts. The AVR process begins when the operator mounts a tape on an AVR–enabled drive, causing the hardware to generate an interrupt to the monitor when the tape is positioned at load point. The system is thereby informed that the new tape unit is on line. (The TU70 series tape subsystem generates on–line interrupts.) Label–processing operations and tape assignment then take place automatically.

For tape units that do not generate on–line interrupts, AVR can be made to operate in semiautomatic mode. That is, the operator command, RECOGNIZE, provides a way for the operator to generate the on–line interrupt.

The on–line tape unit interrupt is serviced by the physical device driver (TX1KON) contained within the TOPS–10 monitor. This interrupt is translated into an IPCF message packet and sent to QUASAR.

9.4.1.1 Ignored Interrupts — AVR acknowledges only those on–line interrupts for 9–track magnetic tape units. Since all 7–track tapes, by definition, are unlabeled, they are ignored by AVR. Also, tape units that are assigned to user jobs in nonlabel or bypass label mode are assumed to be under direct control of the operator; therefore no attempt is made to verify any labels.

9.4.2 Volume Recognition

When QUASAR receives notification of an on–line interrupt, it requests PULSAR to perform label–processing operations. PULSAR then tries to read the volume–header label. If the volume does not contain an acceptable VOL1 label as the first record, the operator is informed: "drive: Contains an UNLABELED tape". The operator then takes appropriate action.

If PULSAR detects a properly labeled volume, it sends the following information to QUASAR to be stored in QUASAR's data base: VOLID, drive name, label type (for example, ANSI), density, and tracks. QUASAR then tries to reassign the volume to a user.

9.4.3 Volume Assignment

After QUASAR receives the label information from PULSAR, it tries to associate the volume with a pending mount request. Three cases exist:

1. Only one user is waiting for the volume. In this case, QUASAR immediately reassigns the device to the user.
2. More than one user is waiting for the volume. Here, the oldest request that can be serviced using available resources while not causing a conflict with other requests is serviced.

3. No one is waiting for the volume. In this case, the operator has premounted the tape in anticipation of a reel switch or of a batch stream start–up. The tape is automatically assigned when a request for it is issued. Refer to Section 9.4.4 for information on premounted volumes.

9.4.4 Premounted Volumes

The tape–processing system allows the operator to mount labeled volumes before requests for them have been issued. This action is known as premounting, or prestaging. The operator might premount successive volumes of a set to prevent job wait during rewind of the current tape and mounting of the next tape.

As mentioned previously, premounted volumes are initially processed like other volumes; however, they cannot be immediately assigned to a user. Instead, the volume information that PULSAR passes to QUASAR remains stored in QUASAR's data base. When a mount request comes in, QUASAR searches its data base looking for volume characteristics that match those specified in the request. If it finds a match, it assigns the tape unit and volume to the user. From there, processing continues without operator intervention.

Chapter 10
The Label Processor's Search Algorithm

If you are a programmer, you should be aware of the label processor's search algorithm when you specify the filename and file sequence parameters with the .TFLPR function of the TAPOP. monitor call. This chapter describes file–positioning operations that are affected by your use of these parameters. Refer to Table 10–1 for a summary of file–positioning possibilities.

10.1 When Neither a Filename nor a Sequence Number Is Specified

When neither a filename nor sequence number is specified, the following action is taken:

Input

The volume set is opened at the current file position. No checking is done on the file name.

Output

As with input, the volume set is opened at the current file position. A new file is written using the default file name.

10.2 When a Filename Is Specified

If the filename is specified but the sequence number is not specified or is zero, the following action is taken:

Input

The volume set is rewound and a search through the volume set is made for a matching file. If the file is found, the file is opened for input. If the file is not found, the "File Not Found" error code is returned to the user program.

Output

As with input, the volume set is rewound, and a search through the volume set is made for a matching file. If the file is found, the file is opened for output. If the file is not found, the file is opened for output at the end of the volume set.

10.3 When a Sequence Number Is Specified

If the sequence number is specified but the filename is not, the following action is taken:

Input

The volume set is positioned at the location indicated by the sequence number. If a file exists there, it is opened for input. If a file does not exist at that location, such as when a file is beyond the logical end of the volume set, the positioning error return is given.

Output

As with input, the volume set is positioned at the location indicated by the sequence number. If a file exists there, it is opened for output. A default filename is written. All subsequent files in the set become useless.

If a file does not exist at that location but the file sequence number is equal to 99,999 or equal to the number of files contained in the volume set plus one, the file is opened at the logical end of tape. A default file name is written.

10.4 When a Filename and a Sequence Number Are both Specified

When the filename and sequence number are both specified, the following action is taken:

Input

The volume set is positioned at the location indicated by the sequence number. If a file exists, a comparison is made between the filename you specified and the filename on the label. If they match, the file is opened for input; otherwise the positioning error is returned.

Output

The volume set is positioned at the location indicated by the sequence number. No check is performed for a matching file name. The file is opened for output, and the filename you specified supersedes any filename that may exist.

Table 10–1: File Positioning Summary

Is File Name Given	Is Sequence Number Given*	Does Filename Exist Where it Should be Found **	Result for Input	Result for Output
YES	NO	YES	The requested file is opened for input.	The requested file is opened for output.
YES	NO	NO	ERROR (File not found)	The requested file is opened. The position follows the last file of the set.
YES	YES	YES	The requested file is opened for input.	The requested file is opened for output.
YES	YES	NO	ERROR (Positioning Error)	The requested file is opened for output, thereby overwriting an old file.*
NO	NO		The "current" file is opened for input.	The "current" file is opened for output. Default file label is written.
NO	YES		The requested file (defined by pos) is opened for input.	The requested file is opened for output. Default file label is written.*

* If a position is greater than the number of files on the volume set (+1 for output), the "File Not Found" error is set. To clear tape–labeling errors, you must use function code 1 of the .TFURQ TAPOP. function.

** Sequence number takes precedence over file name. If sequence number is given, the volume set is positioned first. If no sequence number is given but a filename is given, the whole volume set is searched.

Chapter 11
The TAPOP. Monitor Call

Function

The TAPOP. monitor call controls many tape–processing operations. This chapter defines the various TAPOP. functions.

Several TAPOP. functions are identical to extensions of other monitor calls, such as MTAPE and MTCHR. All TAPOP. functions assume that the specified device has been assigned to your job by the ASSIGN command or the OPEN/INIT monitor call, or that the calling job has SPY privileges.

Calling Sequence

```
        MOVE            ac,[XWD arglength,arglst]
        TAPOP.          ac,
          error return
        normal return
        . . .
arglst: EXP             fcncode

        SIXBIT/device/
        EXP             channo
        EXP             udx
        . . .
        last argument
```

Where: **arglength** is the length of the argument list.

arglist is the address of the argument list.

fcncode is one of the function codes described below.

device is the SIXBIT physical or logical name of a device.

channo is the number of an initialized channel. For information on initializing devices, refer to the *TOPS–10 Monitor Calls Manual.*

udx is the universal device index for a device. The words up through **last argument** are arguments for the given function.

On an error return, an error code is returned in the ac. The TAPOP. error codes are listed at the end of this chapter.

The function codes fall into four groups:

0 – 777	Perform specific actions.
1000 – 1777	Read parameters.
2000 – 2777	Set parameters. These function codes are not explicitly listed in the descriptions below. To set a parameter, use the corresponding read mnemonic plus the offset .TFSET (=1000). For example, to set the density indicator, use the read density indicator mnemonic plus .TFSET: .TFDEN+.TFSET
3000 – 3777	Reserved for customer–defined functions.

The function codes and their meanings are:

Code	Symbol	Function
1	.TFWAT	Waits for I/O to be completed.
2	.TFREW	Rewinds tape to load point.
3	.TFUNL	Rewinds and unloads tape.
4	.TFFSB	Skips forward one block.
5	.TFFSF	Skips forward one file.
6	.TFSLE	Skips to logical end–of–tape.
7	.TFBSB	Skips backward one block.
10	.TFBSF	Skips backward one file.
11	.TFWTM	Writes a tape mark.
12	.TFWLG	Writes three inches of blank tape.
13	.TFDSE	Erases entire tape (data security, TX01/TX02 on DX10/DX20 only). You cannot use this function for tapes that are under the control of PULSAR and QUASAR.
14	.TFWLE	Writes logical end–of–tape (two tape marks for unlabeled tapes; one tape mark followed by the end–of–file label set followed by two tape marks; for labeled tapes).
15	.TFLBG	Gets the tape label device data block. Returns the name in the ac. This is a privileged function for use by the label processor.
16	.TFLRL	Releases the tape label device data block. This is a privileged function for use by the label processor.
17	.TFLSU	Swaps units. This is a privileged function for use by the label processor.
20	.TFLDD	Destroys the tape label data base. This is a privileged function for use by the label processor.

21	.TFFEV	Forces end–of–volume processing. This function allows the label processor to write the end–of–volume (EOV) label set before it finds the end–of–tape mark. The label processor then assumes a multivolume file and automatically issues a MOUNT request for the next volume.
22	.TFURQ	Function code 1 (.TFCLE) clears the tape–label error bits from a previous label error. You retrieve the extended error bits with the .DFRES function of the DEVOP. monitor call. Function 1 causes the tape to be positioned at BOT.
23	.TFSMM	Sets maintenance mode on the tape controller. This is a privileged function.
24	.TFCMM	Clears maintenance mode on the tape controller. This is a privileged function.
25	.TFCEC	Clears error counters. This is a privileged function for use by the label processor.
1000	.TFTRY	Returns in the ac the number of retries on the last error.
1001	.TFDEN	Returns in the ac the density code for the tape. To set the density code, use .TFDEN + .TFSET; the monitor reads the new density code from **arglst + 2**. Note that in order to set the density with this function code, IO.DEN must be zero. The density codes and their meanings are:

Code	Symbol	Density
0	.TFD00	Unit default.
1	.TFD20	200 bits/inch (8.1 rows/mm).
2	.TFD55	556 bits/inch (22.5 rows/mm).
3	.TFD80	800 bits/inch (32.2 rows/mm).
4	.TFD16	1600 bits/inch (65.3 rows/mm).
5	.TFD62	6250 bits/inch (255.5 rows/mm).
6–17	Reserved	To DIGITAL.

1002	.TFKTP	Returns in the ac the controller type code for the tape. To set the controller type code, use .TFKTP + .TFSET; the monitor reads the new controller type code from **arglst + 2**. The controller type codes and their meanings are:

Code	Symbol	Controller Type
0	.TFKTA	TM10A (TU10/20/30/40/41).
1	.TFKTB	TM10B (TU10/20/30/40/41).
2	.TFKTC	TM10C (TU43).
3	.TFKTX	TX01/TX02 on DX10 (TU70,71,72).
4	.TFKTM	TM02/TM03 on RH10/RH20 (TU16,TU45).
5	.TFKRH	TM02/TM03 on RH11 (TU45).
6	.TFKD2	TX02 on DX20 (TU70,71,72).

1003	.TFRDB	Returns in the ac the read–backwards bit (TM02/TX01/TX02 only). The bit is on (1) if the tape is set for read–backwards, or off (0) if not. Refer to the *TOPS–10 Monitor Calls Manual* for information on reading backwards. To set the read–backwards bit, use .TFRDB + .TFSET; the monitor reads the bit from **arglst + 2**.
1004	TFLTH	Returns in the ac the bit for read next record at low threshold (TM10A/B only). The bit is on (1) if the tape is set for low threshold, or off (0) if not. To set the bit ,use .TFLTH + .TFSET; the monitor reads the bit from **arglst + 2**.

1005 .TFPAR Returns in the ac the status of the even parity bit (for 7–track tapes only). To set the status of the even parity bit, use .TFPAR+.TFSET; the monitor reads the status from **arglst+2**.

1006 .TFBSZ Returns in the ac the block size for the tape. The returned value is one greater than the number of data words per record. To set the block size, use .TFBSZ+.TFSET; the monitor reads the block size from **arglst+2**.

1007 .TFMOD Returns in the ac the data mode code for the tape. To set the data mode code, use .TFMOD+.TFSET; the monitor reads the data mode code from arglst +2. The data mode codes and their meanings are:

Code	Symbol	Data Mode
0	.TFMDD	DIGITAL–compatible core dump mode (7–track and 9–track).
1	.TFMID	Industry–compatible core dump mode (9–track).
2	.TFM8B	8–bit mode, 4 bytes per word.
3	.TFM6B	6–bit mode, 6 bytes per word (9–track, TU70 only).
4	.TFM7B	7–bit mode, 5 bytes per word (TU70 only).
5	.TFM7T	DIGITAL–compatible 7–track core dump mode (SIXBIT).

1010 .TFTRK Returns in the ac the track status bit for the tape (0 for 9–track, 1 for 7–track). To set the track status bit, use .TFTRK+.TFSET; the monitor reads the bit from **arglst+2**.

1011 .TFWLK Returns in the ac the write–lock bit for the tape (1 if write–locked, 0 if not).

1012 .TFCNT Returns in the ac the character count of the last record (the actual record length).

1013 .TFRID Returns in the ac the SIXBIT reel identification for the tape. To set the reel identification, use .TFRID+.TFSET; the monitor reads the SIXBIT reel identification from **arglst+2**.

1014 .TFCRC Returns in the ac the last cyclic redundancy character (9–track NRZI only).

1015 .TFSTS Returns in the ac the unit status flags for the tape. To set the flags, use .TFSTS+.TFSET; the monitor reads the flags from **arglst+2**. The unit status flags and their meanings are:

Flag	Symbol	Meaning
18	TF.UNS	Unit is not schedulable.
19	TF.BOT	Unit is positioned at beginning–of–tape mark.
20	TF.WLK	Unit is write–locked.
21	TF.REW	Unit is rewinding.
22–32		Reserved.
33	TF.STA	Unit is started.
34	TF.SEL	Unit is selected.
35	TF.OFL	Unit is off–line.

1016 TFSTA

Returns unit statistics for the tape device. Your program supplies the function code and device at **arglst** and **arglst+1**. (These values are identical to those returned for the MTCHR. monitor call.) The monitor returns the device statistics at **arglst** in the format:

Offset	Symbol	Contents
0	.TSFUN	Function code (user–supplied).
1	.TSDEV	Device (user–supplied).
2	.TSRID	SIXBIT reel identifier.
3	.TSFIL	Number of files since beginning of tape (current file number).
4	.TSREC	Number of records since last end–of–file mark (current record number).
5	.TSCRD	Number of characters read since system reload.
6	.TSCWR	Number of characters written since system reload.
7	.TSSRE	Soft read errors since system reload.
10	.TSHRE	Hard read errors since system reload.
11	.TSSWE	Soft write errors since system reload.
12	.TSHWE	Hard write errors since system reload.
13	.TSTME	Total errors since MOUNT.
14	.TSTDE	Total device errors since system start–up.
15	.TSTUN	Total unloads since system reload.
16	.TSRTY	Number of retries to resolve last error.
17	.TSCCR	Character count of last record read or written.
20	.TSPBE	Position before last error: file number (in left half); record number (in right half).
21	.TSFES	Final error state. See SYSERR documentation.

1017 .TFIEP Returns in the ac the initial error pointer.

1020 .TFFEP Returns in the ac the final error pointer.

NOTE

Function codes 1021 and 1022 return the blocks pointed to by 1017 and 1020. These blocks are for communication of errors to DAEMON and may change without notice.

1021 .TFIER Returns in the ac the initial error status.

1022 .TFFER Returns in the ac the final error status.

1023 .TFFED Returns in the ac the final error disposition.

1024 .TFLBL

Returns in the ac the label–processing type code. To set the label–processing type code, use .TFLBL+.TFSET; the monitor reads the new code from **arglst+2**. The set function requires JP.POK, [1,2], or JACCT privileges. The label–processing type codes and their meanings are:

Code	Symbol	Label–Processing Type
0	.TFLBP	Bypass label processing.
1	.TFLAL	ANSI labels.
2	.TFLAU	ANSI labels with user labels.
3	.TFLIL	IBM labels.
4	.TFLIU	IBM labels with user labels.
5	.TFLTM	Leading tape mark.
6	.TFLNS	Nonstandard labels.
7	.TFLNL	No labels. When tapes are processed with no labels, the label processor is used only to verify that the tape does not contain a tape label. Unlabeled tapes can be copied to create a labeled tape.
10	.TFCBA	DEC COBOL ASCII labels.
11	.TFCBS	DEC COBOL SIXBIT labels.
12	.TFLNV	Same as .TFLNL except that user program is responsible for dealing with an EOT. This type is the default. To switch reels at EOT, use TAPOP. function .TFFEV.

1025 .TFPLT

Performs same function as .TFLBL (1024).

1026 .TFLTC

Returns last tape–labeling return code. The return codes and their meanings are:

Code	Symbol	Termination Code
1	.TFTCP	Continue processing
2	.TFTRE	Return EOF
3	.TFTLT	Label type error
4	.TFTHL	Header label error
5	.TFTTL	Trailer label error
6	.TFTVL	Volume label error
7	.TFTDV	Device error
10	.TFTDE	Data error
11	.TFTWL	Write lock error
12	.TFPSE	Positioning error
13	.TFBOT	BOT
14	.TFIOP	Illegal operation
15	.TFFNF	File not found
16	.TFCAN	OPR cancelled request
17	.TFTMV	Too many volumes requested

Refer also to the extended I/O error code.

1027 .TFDMS

Returns in the ac the diagnostic mode set bit (TX01/TX02 on DX10 only). This bit is 1 for diagnostic mode, otherwise 0. To set this bit, use .TFDMS+.TFSET; the monitor reads the new bit from **arglst+2**.

1030 .TFFSO

Returns in the ac the bit showing whether a forced sense command will be issued to the controller (TX01/TX02 on DX10/DX20 only) after the completion of every operation. To set the bit, use

.TFFSO+.TFSET; the monitor reads the new bit from **arglst+2**. This bit should be set by diagnostic programs only, since it slows down tape operation considerably.

1031 .TFMFC

Returns in the ac the maximum frame count. To set the count, use .TFMFC+.TFSET; the monitor reads the new count from **arglst+2**. Use this function to speed tape throughput for a TU16, or TU45, TU70, TU71, or TU72 that does not have an integral number of bytes per word. The count stays in effect until your program performs a RESET with another TAPOP. monitor call or until the tape is RELEASed (if the device was ASSIGNed). This function allows a TU70 or a TU16 to read and write tapes that do not have an integral number of bytes per word. This function provides tape compatibility with other systems.

1032 .TFPDN

Returns in the ac flags showing the possible densities for a tape. The flags and their meanings are:

Flag	Symbol	Density
31	TF.DN5	6250 bits/inch (255.5 rows/mm)
32	TF.DN4	1600 bits/inch (65.3 rows/mm)
33	TF.DN3	800 bits/inch (32.2 rows/mm)
34	TF.DN2	556 bits/inch (22.5 rows/mm)
35	TF.DN1	200 bits/inch (8.1 rows/mm)

1033 .TFLPR

Returns at **arglst+2** the following tape label parameters. Function code 1033 always returns numbers. This function causes the first input label processing if there is not file open for input. To set the parameters, use .TFLPR+.TFSET; the monitor reads the parameters beginning at **arglst+2**. This set function is legal only if there is no file open for output on the given channel. The parameters given apply to the next file to be written. The format of the parameters at **arglst** is:

Offset	Symbol	Contents
0	.TPFUN	Function code (user–supplied).
1	.TPDEV	Device (user–supplied).
2	.TPREC	Record format and form control.

Bits	Symbol	Meaning
18–35	TR.RFM	Record format byte.

Code	Symbol	Meaning
0	TRF.DX	Default (fixed).
1	TRF.FX	Fixed (F).
2	TRF.VR	Variable (D).
3	TRF.SP	Spanned (S).
4	TRF.UN	Undefined (U).

Bits	Symbol	Meaning
0–17	TR.FCT	Forms Control byte.

Code	Symbol	Meaning
1	TFC.NO	Records on tape do not contain form control characters.

	Code	Symbol	Meaning
	2	TFC.AS	First character of each record is a form control character.
	3	TFC.AM	Records on tape contain all required form control characters.

3	.TPRSZ	Record size in characters.
4	.TPBSZ	Block size in characters.
5	.TPEXP	Expiration date in 15–bit format.

Bits	Symbol	Meaning
18–35	TPE.EX	Expiration date.
0–17	TPE.CR	Creation date.

6	.TPPRO	Protection Code.
7	.TPSEQ	File sequence number.
10	.TPFNM	Filename (17 chars ASCII).
14	.TPGEN	Generation and version numbers.

Bits	Symbol	Meaning
0–17	TP.GEN	Generation number.
18–35	TP.VER	Generation version number.

15	.TPLEN	.TPGEN + 1	Length of block

Normal Return

The function is performed.

Error Return

One of the following error codes is returned in the ac:

Code	Symbol	Error
–1	TPACS%	Address check storing answer.
0	TPIFC%	Illegal function code.
1	TPPRV%	Not enough privilege.
2	TPNMT%	Not a magtape device.
3	TPVOR%	Specified value out of range.
4	TPACR%	Address check reading arguments.
5	TPCBS%	Parameter cannot be set.
6	TPNIA%	Tape not initialized or assigned.
7	TPNLP%	No label process.
10	TPETC%	Termination code error.
11	TPIJN%	Illegal job number.
12	TPLRF%	Label release function required.
13	TPLSI%	Label parameter set illegal after first output.
14	TPLOE%	Label DDB owned by someone else on label get.
15	TPDNC%	Drive not capable of specified density.
16	TPWWL%	Write attempted to write–locked tape.

Example

The following example uses TAPOP. functions to generate a map of a labeled tape. A tape map is similar to a disk directory in that it lists all the files stored on the tape along with their attributes. A sample run of the program follows the listing. Note that the program makes use of the GALAXY library (GLXLIB).

```
                TITLE   TAPMAP - Labeled tape mapper program
                SUBTTL  D. Cornelius 21-Nov-79

                SEARCH  GLXMAC
                PROLOG  (TAPMAP)

                SUBTTL  Data storage

                TAPDEV==SIXBIT/STR/             ;Logical name to read from
                OCHN==10                        ;Channel upon which to do I/O

OPNBLK:         EXP     .IODMP
DVNAM:          EXP     TAPDEV                  ;logical drive name
                XWD     0,0

VOLID:          BLOCK   1
FILCNT:         BLOCK   1
PDL:            BLOCK   100

ERTAB:          [ASCIZ/No errors/]
                [ASCIZ/Continue processing/]
                [ASCIZ/EOF/]
                [ASCIZ/Label type error/]
                [ASCIZ/Header label error/]
                [ASCIZ/Trailer label error/]
                [ASCIZ/Volume label error/]
                [ASCIZ/Device error/]
                [ASCIZ/Data error/]
                [ASCIZ/Write-lock error/]
                [ASCIZ/Positioning error/]
                [ASCIZ/BOT/]
                [ASCIZ/Illegal operation/]
                [ASCIZ/File not found/]

RECFMT:         [ASCIZ/Def/]
                [ASCIZ/Fix/]
                [ASCIZ/Var/]
                [ASCIZ/Span/]
                [ASCIZ/Undf/]

FORCON:         [ASCIZ/????/]
                [ASCIZ/None/]
                [ASCIZ/ANSI/]
                [ASCIZ/Embd/]

LPBLK:          EXP     .TFLPR                  ;Get label parameters
                EXP     TAPDEV                  ;Device name
                BLOCK   .TPLEN-2                ;Label parameters come back here
                LPLEN==.-LPBLK

                SUBTTL  Main Routine

START:          RESET
                MOVE    P,[IOWD 100,PDL]
                SETZ    S1,
                PUSHJ   P,I%INIT
RETRY:          MOVE    S1,DVNAM                ;Get the generic device we use
                DEVNAM  S1,                     ;Does the user have it around?
                SKIPA                           ;No, complain
                JRST    DEVOK                   ;Yes, keep going
                $TEXT   (,<Please assign your tape as logical name ^W/DVNAM/:, and type CONTINUE>)
                EXIT    1,
                JRST    RETRY
```

```
DEVOK:  MOVE    S1,[2,,T1]              ;Point at the arg block
        MOVX    T1,.TFLBL               ;Code to get label type
        MOVE    T2,DVNAM                ;Get the drive name
        TAPOP.  S1,                     ;Find out the label type
        SKIPA                           ;Can't, so complain
        JRST    LABELD                  ;Got it, so use it
        $TEXT   (,<Device —^W/DVNAM/: is not a tape!>)
        EXIT    1,
        JRST    RETRY

LABELD: CAIL    S1,.TFLAL               ;Is it ANSI ?
        CAILE   S1,.TFLIU               ;Or EBCDIC (IBM)
        SKIPA                           ;No, complain
        JRST    ANSLBL                  ;Yes
        $TEXT   (,<Device —^W/DVNAM/: is not a labeled tape!>)
        EXIT    1,
        JRST    RETRY

ANSLBL: MOVE    S1,[2,,T1]              ;Aim at the arguments
        MOVX    T1,.TFREW               ;Code to rewind the tape
        TAPOP.  S1,                     ;Do it
        SKIPA                           ;Can't
        JRST    REWDON                  ;Got it
        $TEXT   (,<Can't REWIND —^W/DVNAM/:>)
        EXIT    1,
        JRST    RETRY

REWDON: MOVE    S1,[2,,T1]              ;Aim at the arguments
        MOVX    T1,.TFRID               ;Code to get the reelid
        TAPOP.  S1,                     ;Ask the monitor
        SKIPA                           ;Can't
        JRST    GOTRID                  ;Wins
        $TEXT   (,<Can't get volume id from ^W/DVNAM/:>)
        EXIT    1,
        JRST    RETRY

GOTRID: MOVEM   S1,VOLID                ;Save the reelid
        OPEN    IOCHN,OPNBLK            ;Open up the tape
        SKIPA                           ;Can't
        JRST    TAPOPN                  ;Got it
        $TEXT   (,<Can't OPEN device ^W/DVNAM/:>)
        EXIT    1,
        JRST    RETRY

TAPOPN: $TEXT   (,<Start of tape map for volume ^W/VOLID/ on drive ^W/DVNAM/: at ^H/[-1]/>)

        SETZM   FILCNT                  ;Clear the number of files seen
        $TEXT   (,<File    File Name  Pro Gen Ver BlkSiz RecSiz Creation Expiration RecF Form>)
        $TEXT   (,<------  ---------- ----- ------ ----- -------- --------- ---------- ---------- ------ -------->)
FILLOP: MOVE    S1,[LPLEN,,LPBLK]       ;Aim at the label parameter block
        TAPOP.  S1,                     ;Read the label params of the next file
        JRST    ALLDON                  ;Can't, see about EOF
        AOS     FILCNT                  ;Wins, count this file
        PUSHJ   P,TYPFIL                ;Print the info about this file
        MOVE    S1,[2,,T1]              ;Aim at the argument block
        MOVX    T1,.TFFSF               ;Code to skip a file
        MOVE    T2,DVNAM                ;On this tape
        TAPOP.  S1,                     ;Skip it
        SKIPA                           ;Can't, so complain
        JRST    FILLOP                  ;Wins, go get params on this file
        $TEXT   (,<Can't skip file on ^W/DVNAM/: file ^D/FILCNT/>)
        EXIT    1,
        JRST    RETRY

;Here when we can't get params on some file.

ALLDON: GETSTS  IOCHN,S1                ;Read the standard device status
        TXNN    S1,IO.ERR!IO.EOF        ;Errors, or EOF?
        JRST    QUIT                    ;No errors, nor EOF, strange, so quit
        TXC     S1,IO.ERR               ;Complement the error bits
        TXCE    S1,IO.ERR               ;All four error bits up?
        JRST    EOFCHK                  ;No, Could have been EOF
        MOVE    S1,[2,,T1]              ;Aim at the argument block
        MOVX    T1,.DFRES               ;Read extended device status
```

```
        MOVE    T2,DVNAM        ;On this tape
        DEVOP.  S1,             ;Read 'em
        SKIPA                   ;Can't!?
        JRST    GOTERS          ;Got 'em, see what the problem is
        $TEXT   (,<Can't Get label error status for ^W/DVNAM/: file ^D/FILCNT/>)
        EXIT    1,
        JRST    RETRY

GOTERS: $TEXT   (,<Label error on ^W/DVNAM/: file ^D/FILCNT/
Error code:^D/S1/, ^T/@ERTAB(S1)/>)
        JRST    QUIT

EOFCHK: TXNN    S1,IO.ERR       ;Things other than EOF
        JRST    QUIT            ;No, just EOF, so that's alright
        $TEXT   (,<I/O error on ^W/DVNAM/: file ^D/FILCNT/, I/O status = ^O/S1/>)
        EXIT    1,
        JRST    RETRY

QUIT:   $TEXT   (,<End of tape map for volume ^W/VOLID/ on drive ^W/DVNAM/: at ^H/[-1]/
Total of ^D/FILCNT/ files.>)

        EXIT
        SUBTTL  TYPFIL - Print info about the current file

        ;This routine dumps the known info about the file
        ; that is currently open.
        ;It is assumed that the .TFLPR to get the label parameters has just
        ; been done.
        ;Most of the info typed by this routine comes from LPBLK.
        ;Other inputs:
        ; FILCNT/  # of the current file

TYPFIL: $SAVE   <P1,P2,P3,P4>

        ;Do date-time conversion of creation date

        SETZ    S1,             ;No time, just date
        LOAD    S2,LPBLK+.TPEXP,TPE.CR
                                ;Get the creation date
        PUSHJ   P,CNVDT         ;Convert to UDT
        MOVE    P1,S1           ;Save it

        ;Do date-time conversion of expiration date

        SETZ    S1,             ;No time, just date
        LOAD    S2,LPBLK+.TPEXP,TPE.EX
                                ;Get the expiration date
        PUSHJ   P,CNVDT         ;Convert to UDT
        MOVE    P2,S1           ;Save expiration UDT
        LOAD    S1,LPBLK+.TPREC,TR.RFM
                                ;Get the record format code
        LOAD    S2,LPBLK+.TPREC,TR.FCT
                                ;Get the form control code
        $TEXT   (,<^D4L/FILCNT/ ^T17L /LPBLK+.TPFNM/
                   ^O3R0/LPBLK+.TPPRO/ ^D4R/LPBLK+.TPGEN,TP.GEN/
                   ^D3R/LPBLK+.TPGEN,TP.VER/ ^D6R/LPBLK+.TPBSZ/
                   ^D6R/LPBLK+.TPRSZ/ ^H9/P1/ ^H9/P2/ ^T4L /@RECFMT(S1)/
                   ^T4L
        /@FORCON(S2)/>)
        $RETT
        SUBTTL  CNVDT -         CONVERT DATE TO UDT

        ;THIS ROUTINE WILL MAKE A UDT OUT OF AN ARBITRARY DATE
        ;
        ;CALL    S1/             TIME IN SECONDS
        ;       S2/             DATE IN SYSTEM FORMAT
        ;                       (Y*12+M)*31+DAY SINCE 1/1/64
        ;
        ;RETURN  S1/             UDT
        ;
        ;NOTE:   LEFT HALF DIVIDED BY 7 GIVES THE DAY OF THE WEEK

        RADIX 10  ;UNDER RADIX 10 **** NOTE WELL ****
```

```
CNVDT::  PUSHJ   P,.SAVET          ;PRESERVE T3
         PUSH    P,S1              ;SAVE TIME FOR LATER
         IDIVI   S2,12*31          ;S2 = YEARS–1964
         CAILE   S2,2217–1964      ;SEE IF BEYOND 2217
         JRST    GETNW2            ;YES—RETURN -1
         IDIVI   T1,31             ;T1 = MONTHS–JAN, T2 = DAYS–1
         ADD     T2,MONTAB(T1)     ;T2 = DAYS–JAN 1
         MOVEI   T3,0              ;LEAP YEAR ADDITIVE IF JAN, FEB
         CAIL    T1,2              ;CHECK MONTH
         MOVEI   T3,1              ;ADDITIVE IF MAR–DEC
         MOVE    S1,S2             ;SAVE YEARS FOR REUSE
         ADDI    S2,3              ;OFFSET SINCE LEAP YEAR DOES NOT
                                   ; GET COUNTED
         IDIVI   S2,4              ;HANDLE REGULAR LEAP YEARS
         CAIE    T1,3              ;SEE IF THIS IS LEAP YEAR
         MOVEI   T3,0              ;NO—WIPE OUT ADDITIVE
         ADDI    T2,<1964–1859>*365+<1964–1859>/4+<31–18>+31(S2)
                                   ;T2 = DAYS BEFORE JAN 1,1964 +SINCE JAN 1
                                   ; +ALLOWANCE FOR ALL LEAP YEARS SINCE 64
         MOVE    S2,S1             ;RESTORE YEARS SINCE 1964
         IMULI   S2,365            ;DAYS SINCE 1964
         ADD     T2,S2             ;T2 = DAYS EXCEPT FOR 100 YR. FUDGE
         HRREI   S2,64–100–1(S1)   ;S2 = YEARS SINCE 2001
         JUMPLE  S2,GETNW1         ;ALL DONE IF NOT YET 2001
         IDIVI   S2,100            ;GET CENTURIES SINCE 2001
         SUB     T2,S2             ;ALLOW FOR LOST LEAP YEARS
         CAIE    T1,99             ;SEE IF THIS IS A LOST L.Y.
GETNW1:  ADD     T2,T3             ;ALLOW FOR LEAP YEAR THIS YEAR
         CAILE   T2,^O377777       ;SEE IF TOO BIG
GETNW2:  SETOM   T2                ;YES—SET –1

         POP     P,S1              ;GET MILLISEC TIME
         MOVEI   S2,0              ;CLEAR OTHER HALF
         ASHC    S1,–17            ;POSITION
         DIV     S1,[24*60*60*1000]  ;CONVERT TO 1/2**18 DAYS
         CAMLE   S2,[^D24*^D60*^D60*^D1000/2]
                                   ; OVER 1/2 TO NEXT?
         ADDI    S1,1              ;YES, SHOULD ACTUALLY ROUND UP
         HRL     S1,T2             ;INCLUDE DATE
GETNWX:  POPJ    P,                ;RETURN
MONTAB:  EXP     0,31,59,90,120,151,181,212,243,273,304,334,365
         RADIX   8

         END     START
```

The program above produces the following results:

```
.MOU TAPE-EXAMPLE:STR/LABEL:EBCDIC/NOTIFY/NOWAIT/VOLID:HPP002
[15:20:05]
[Mount request TAPE-EXAMPLE queued, request #1274]

From system:
[Magtape HPP002 mounted on MTB263 with logical name STR ]

.RU TAPMAP

Start of tape map for volume HPP002 on drive STR: at 10-Jul-80 15:24:34

File File Name  Pro Gen Ver BlkSiz RecSiz Creation  Expiration  RecF Form
---- --------  --- --- --- ------ ------ --------  ----------  ---- ----
1    IMPRESS   000 0   0   2741   160    23-Feb-76 1-Jan-64    Undf None
2    CLIMBER   000 0   0   2741   160    23-Feb-76 1-Jan-64    Undf None
3    CALENDAR  000 0   0   2741   160    12-Mar-76 1-Jan-64    Undf None
4    GEOMETRY  000 0   0   2741   160    12-Mar-76 1-Jan-64    Undf None
```

Chapter 12
Crash Procedures

GALAXY components may crash, that is, stop performing useful work, for a number of reasons:

- A monitor call (UUO) that should not fail takes the error return and a GALAXY stopcode occurs.
- One of the consistency checks, which are built into the various GALAXY components, detects an error and a GALAXY stopcode occurs.
- An undetected error eventually causes an illegal memory reference and a GALAXY stopcode occurs.
- A component goes into event–wait (EW) state for an extended length of time waiting for an event that is not likely to occur (perhaps due to some hardware or software malfunction). In this case, the program hangs in the UUO.

Crashes may or may not be accompanied by GALAXY stopcode messages. Those crashes without stopcode messages indicate that the system is hung or has fallen into an unending loop. This chapter is concerned with crashes that involve the PULSAR and QUASAR GALAXY components. Refer to the *TOPS–10 Crash Analysis Guide* for an in–depth discussion of stopcodes and crashes that involve the monitor.

To recover from PULSAR and QUASAR crashes, users should take the steps outlined below.

12.1 Recovery Procedures

User procedures differ according to the GALAXY component involved in the crash:

PULSAR

After a PULSAR crash, users accessing tape drives may find that their jobs are unable to automatically continue when PULSAR is restarted. The user who caused the crash will find that his/her job remains in EW state. Other users will receive tape–labeling errors and will be unable to continue because PULSAR has lost record of any label–processing requests made before the crash. This is because PULSAR's data base has been reinitialized. Thus users/jobs in the post restart state must type a control–C; give the .FINISH command; then restart operations.

QUASAR

When QUASAR crashes, user requests for queue services are lost, even though the failsoft master queue file is rebuilt when QUASAR is restarted. For example, user requests for mounts/dismounts involving GALAXY–controlled devices will not be serviced. Also, programs using the QUEUE. UUO to interface with QUASAR will hang in EW state in the UUO. To recover from these crashes, type a control–C and begin anew.

Tape drives allocated to a user will be set unavailable when QUASAR is restarted. However, the user will continue to be able to access the drive, both during the crash and after the restart, without problems.

12.2 Crash Analysis

If you are a system programmer, you might want to examine the ac registers as they existed at the time of the crash to determine the cause of the crash. You can then continue the component's operations from the point following the crash.

Whenever a GALAXY component generates a stopcode, it saves the contents of the ac registers and the text description of the crash. The ac registers are saved in an area called .SACS, from locations .SACS+0 to .SACS+17. The address of the ASCIZ text that describes the crash is saved in WTOADR. A sample of the crash block information is shown below:

Figure 12–1: A Crash Block

```
?Stopcode – ILM – in module GLXINT on 16–Mar–81 on 15:49:43
 Reason: Illegal memory reference at PC 631334
 Program is ORION version 4(522) using GLXLIB version 1(1006)
 Crash block starts at location 674000
 Last GLXLIB error: 21 (No such pid)

 Contents of the ACs:
  0/      777777777777      777627      200370          120
  4/                 0           0           0           77
 10/                16           0           0          400
 14/           1000015           0           0 777474000143

 Last 9 stack locations:
 –1(P)/      200370  –2(P)/      777627  –3(P)/ 312000000700
 –4(P)/ 400000522    –5(P)/           0  –6(P)/            1
 –7(P)/           0  –8(P)/ 44622000000  –9(P)/            0
```

In the GALAXY code, the $STOP macro is used to process the fatal condition.

GALAXY stopcodes are stored as S..xxx, where xxx is the 3–character stopcode name. (Refer to Appendix A, Label–Processing Stopcodes.)

Stopcodes stop program execution, and execution cannot continue unless DDT is loaded with the program. If DDT is loaded with the program, the stopcode processor transfers control to DDT.

Stopcodes are conditionally sent to ORION to be recorded in the system log and to be sent to any jobs running OPR. A flag in the program's initialization block (in IB.STP) enables this conditional function.

To continue a program that has received a stopcode, first determine where the stopcode occurred. The address where the stopcode was executed is located at S..xxx, where xxx is the 3–character stopcode name. Next, issue the POP P,ESCX instruction to restore the stack to its precrash value. You can now continue the program from the point following the stopcode, provided that you have corrected the situation that caused the program to receive a stopcode in the first place.

Other data areas that may help resolve a crash are:

.SPC	PC of the stopcode
.SCODE	SIXBIT name of the stopcode
.SPTBL	Page table address
.SPRGM	SIXBIT program name
.SPVER	Program version number
.SPLIB	GLXLIB version number
.LGERR	Last GALAXY error code
.LGEPC	PC of the last $RETE
STPFLG	A stopcode is in progress (–1)
PIDTAB	System PID table — indexed by SP.xxx, where xxx is the processor name (QSR, OPR, and others)
RCVBLK	PDB of the last IPCF message received
PACKET	Contents of the last IPCF message received as a packet
PAGTBL	Memory page map (1 word per page)

Bits:
- PG.USE = = 1B35 – Page is in use
- PG.WRK = = 1B34 – Page is in working set
- PG.ADR = = 1B33 – Page is addressable
- PG.INI = = 1B32 – Page is part of initial core image
- PG.SWP = = 1B31 – Page is swappable on a timer trap

No bits lit indicates that the page does not exist.

Appendix A
Label–Processing Stopcodes

As discussed in Chapter 12, Crash Procedures, certain hardware and software malfunctions cause stopcodes to occur. These stopcodes result in system crashes. Appendix A lists the label–processing stopcode names and their associated messages. These appear on the operator's terminal when PULSAR experiences a stopcode crash.

ADF — AVR Label Destroy Failed
The .TFLDD TAPOP. failed. This function returns the label DDB after volume recognition is performed.

BBF — Bad Backspace File
See SSR.

BBR — Bad Backspace Record
This stopcode is like SSR.

BCN — Bad Call to NXTFIL
PULSAR is confused about the position of the tape.

BCP — Bad Call to POSTAP
POSTAP has been called with neither position by sequence nor position by filename on in the status word.

CAS — Can't Append to SPT List
The system PUSHJs here if there are problems building the SAT tables for a file structure.

CBP — Called on Bypass Label Processing
The system PUSHJs here if any activity occurs on a bypass label tape.

CCD — Can't Change Density
The .TFDEN TAPOP. failed.

CCF Can't Create Failsoft File

CCP Can't Checkpoint File

CCR Can't Check Ring Status
The .TFWLK TAPOP. failed.

CCW Can't Clear Watch Bits
The SETUUO to clear watch bits (so unload stats do not get typed on a detached TTY) failed.

CDC Can't Determine Density Capabilities
The .TFPDN TAPOP. failed.

CDF Can't Delete Failsoft File

CDK Can't Determine Kontroller Type
The .TFKTP TAPOP. failed.

CDS Can't Get DSKCHR Status for (device name) error code = S2
DSKCHR for a file structure or a disk unit failed.

CDT Can't Determine Track Status
The .TFTRK TAPOP. failed.

CEF Can't Enter Failsoft File

CGC Cannot Get a Failsoft Channel
There are no free (standard) channels for failsofting TCBs.

CGD Can't Get Density

CGS Can't GETTAB States Word

CMU Can't Make TCB

CMV Can't Make TCB

COF Can't Open Failsoft Device

CPF Clear Label Parameters Failed
The .TFLPR to set label parameters to 0 failed.

CRB Can't Read Buffer Size
PULSAR cannot determine user's buffer size to set up a default block count in HDR1.

CRM Can't Read User's Mode

CRS Can't RESDV. UFD Channel
The extended channel to read/write the UFD has gone away or been lost.

CRT Can't Reopen Tape
The operation that causes dummy output to be written after the label DDB channel is closed failed.

CSB — Can't Set Blocksize
The .TFBSZ TAPOP. failed.

CSD — Can't Set Density

CSI — Can't Set Industry–Compatible Mode

CSM — Can't Set DIGITAL–Compatible Mode
The .TFMOD to set DEC–compatible I/O failed.

CSU — Can't Switch Units
The .TFLSU TAPOP. failed.

CUF — CHKACC UUO Failed

CWT — Can't Write Tape Mark
The .TFWTM TAPOP. failed.

DNI — DEVTYP UUO Not Implemented

DNZ — Device Name Is Zero in Failsoft File

DOF — Dummy OUT Failed
The operation that causes dummy output to be written whenever a tape is opened for I/O failed.

ELF — Error Looking Up Failsoft File

FTS — Failed Reading Track Status
The .TFTRK TAPOP. failed.

FUF — FILOP. UUO Failed
A FILOP for other than IN or OUT failed.

GNF — GETTAB for User's Name Failed

GSF — GETTAB for Serial Number Failed

GUF — GETTAB UUO Failed.

IDM — Invalid Date from Monitor
The date used in expiration checking is non–numeric.

IPF — Illegal Positioning Function
An illegal SIXBIT positioning code was passed to T$POS.

IRF — IO Error Reading Failsoft File

ISN — Illegal Structure Name
QUASAR has asked PULSAR to build a structure with a bad FS name.

LDF — Label Destroy Failed
This stopcode is similar to ADF, but from a different context.

LGF — Label Get Failed
The .TFLBG TAPOP. failed. This condition was probably caused by a bad device name in a recognize or monitor message.

LRF — Label Release Failed
The .TFLRL TAPOP. failed.

MCF — MTAID. UUO Failed

NEB — No Error Bit
A call to translate DEC–10 I/O error bits to internal PULSAR error bits was made with no DEC–10 I/O bits on.

NVD — No Valid Density
PULSAR cannot find any valid density in the possible density mask.

PLM — Previous List TCB Has Been Meddled With
A return to the scheduler has been made on a process that was different from the one the scheduler believed was running.

PNR — PULSAR Not Restartable

PRF — Positioning Request Failed
The TAPOP. function to move the tape failed.

RAT — Requesting Work for Active TCB
A disk TCB has requested that the HOM blocks for a unit be read, but that unit's TCB is not idle.

RKT — Running a Killed TCB
The call to kill a TCB after all work has been finished has returned, and it should never return!

RLT — Failed Reading Label Type
The .TFLBL TAPOP. failed.

RPF — Read Label Parameters Failed
The .TFLPR TAPOP. to get user's label parameters failed.

RSE — Reschedule from Exec Level
The process scheduler was called from non–TCB level.

RSF — TAPOP. to Read Statistics Failed
PULSAR cannot determine the number of bytes the user wrote on the tape.

RTT — Releasing Tape Twice
A second call to T$RELE was done without an intervening call to T$OPEN, or the TI.OPN bit was lost.

SIO — Switch Units with OPEN Label DDB
Attempt to switch units to a drive that is active, that is, open for label I/O.

SLT — Set Label Type Failed
The .TFPLT TAPOP. failed.

SND — Switch Units with Nonexistent Device (device name)

SPF — Set Label Parameters Failed
The .TFLPR TAPOP. to return label information to user failed.

SSR — Strange Skip Record
PULSAR is confused about the positioning of a tape on receipt of a monitor message issued when a user tried to move a labeled tape.

TRB — Can't Check Read–backwards

TUF — TAPOP. UUO failed
Some TAPOP. function failed. See the stack for details.

ULS — Unit Parameter List is Short
The list of unit parameter blocks that hang off a structure TCB does not have as many entries as there are units in the FS.

WCF — Wait after Close Failed

WCL — Writing COBOL Labels
This stopcode is like CBP, but for the case of outputting COBOL labels.

WNF — Waiting TCB not Found
When a TCB doing work finished, the TCB requesting the work was not found.

WRF — Wait Request Failed
The .TFWAT TAPOP. (after every positioning except rewind and unload) failed.

Appendix B
Tape Label Implementation Levels

The ANSI standard allows for varying degrees of labeling capability among systems involved in information interchange. It does this by establishing three nested subsets of the full standard. These subsets, or levels, range from the simplest processing at level 1 to the most complex at level 4 (representing the complete model). In meeting ANSI standards, a system conforms to one of these levels.

There are three main identifiable constituents of a level: the file formats, the labels used, and the record formats. The most familiar and useful options of these are as follows:

Constituent	Option	Constituent Configurations
Files	1	Single–file single–volume + single–file multivolume
	2	Single–file single–volume + Single–file multivolume + multifile single–volume + multifile multivolume
Labels	1	VOL1 HDR1 EOF1 EOV1
	2	VOL1 HDR1 HDR2 EOV1 EOV2 EOF1 EOF2
Record Format	1	Fixed
	2	Fixed + variable
	3	Fixed + variable + spanned

Each level is made up of the option constituents as follows:

	Files	Labels	Format
Level 1	1	1	1
Level 2	2	1	1
Level 3	2	2	2
Level 4	2	2	3

Recall that TOPS–10 reads and writes level 4 labels and file formats. However, it does not ensure consistency between the file data and the label specifications. Thus, TOPS–10 does not fully conform to the ANSI standard.

Appendix C
Glossary

accessibility character

A character that indicates whether a tape volume is protected.

ANSI–ASCII mode

A data recording mode in which data is stored in 7–bit ASCII bytes (also called 7–bit mode). Data must be stored in this mode to meet ANSI standards.

Automatic Volume Recognition (AVR)

A TOPS–10 facility that automatically verifies label information when a tape is mounted, then assigns the tape to a waiting user.

beginning–of–tape marker

A physical marker on a magnetic tape used to indicate the beginning of the permissible recording area.

block

A collection of characters written or read as a unit. A physical record.

block size

The size in characters of a block. The default size is 128 words, 4 characters to a word.

blocked record

A record contained in a file in which more than one record or segment can be contained in a block.

bypass label processing

The processing mode wherein the system ignores tape labels.

density

The number of characters that can be recorded on one inch of magnetic tape.

EOT

The abbreviation for end–of–tape.

end–of–tape marker

A physical marker on a magnetic tape used to indicate the end of the permissible recording area.

expiration date

The earliest date on which a tape may be overwritten.

file section

That part of a file recorded on any one volume. The sections of a file do not have sections of other files interspersed.

file set

A collection of one or more files recorded consecutively on a volume set.

fixed–length record

A record contained in a file in which all records are the same length. No indication of the record length is contained in a fixed–length record.

frame

The smallest data aggregate on a magnetic tape. It consists of 8 bits (on a 9–track tape) or 6 bits (on a 7–track tape), plus a parity bit.

interrecord gap (IRG)

Spaces or gaps between blocks on a magnetic tape.

label

A reserved record at the beginning of a volume, or at the beginning or end of a file section, or at the end of a file, that identifies and delimits that volume or file section. A label is not considered to be part of a file.

label group

A collection of one or more contiguous label sets.

label number

A digit that indicates the sequence of labels within a label set. Label number one is always present if the label set exists.

label set

A collection of one or more contiguous labels with the same three initial characters. These three characters are called the Label Identifier.

logical record

A record defined from the standpoint of its content, function, and use rather than from its physical attributes.

multiple–volume file

A file that spans more than one magnetic tape reel.

physical record

A record identified from the standpoint of the manner or form in which it is stored and retrieved; that is, one that is meaningful with respect to access. A block.

prestaging

The operator's mounting of volumes that a user has not yet requested.

protection code

A code indicating who can and cannot access the files on the magnetic tape.

record segment

That part of a spanned record that is contained in any one block. The segments of a record do not have segments of other records interspersed.

record size

The size in characters of a record. The default size is 128 characters.

scratch tape

A new tape that has been initialized and that contains only volume labels (if a labeled tape), tape marks, and no user data, or an old tape whose data can be overwritten.

Scratch tapes are assigned to users as needed and may be used for any purpose.

spanned record

A record contained in a file in which each record may begin in one block and end in another. These records can span volumes.

tape mark

On labeled tapes, a delimiter used to indicate the boundary between label groups and between label groups and file data. On unlabeled tapes, a tape mark separates individual files. On both types of tapes, tape marks indicate the logical end–of–tape.

unblocked record

A record contained in a single block with no other records or parts of records contained in that block.

unspanned record

A record contained in a file in which each record by design ends in the same block in which it begins.

variable–length record

A record contained in a file in which the records may have different lengths.

VID

The visual identification label; the physical label that is affixed to a magnetic tape reel.

VOLID

The volume identifier; the string of characters that distinguishes a volume and is written in the VOL1 label during initialization.

volume

The equivalent term for a reel of magnetic tape. A volume may contain part of a file, a complete file, or more than one file.

volume–set

A set of volumes that the user may logically treat as one large volume. A volume set contains one file set only.

volume switch

The process of removing a volume from a tape drive and mounting the next volume of the set.

Index

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear – Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR01-2/L12
MARLBOROUGH, MA 01752

Do Not Tear – Fold Here and Tape

Cut Along Dotted Line