Burroughs

Reference Manual

B 22 Systems

Tape Streamer Interface

(Relative to Release Level 4.0)

Distribution Code SA

Priced Item
Printed in U.S.A.
February 1985

1180080

Burroughs

Reference Manual

B 22 Systems Tape Streamer Interface

(Relative to Release Level 4.0)

Distribution Code SA

Priced Item
Printed in U.S.A.
February 1985

1180080

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued from time to time to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded, using the Documentation Evaluation Form at the back of the manual, or remarks may be addressed directly to Burroughs Corporation, Corporate Product Information East, 209 W. Lancaster Ave., Paoli, PA 19301, U.S.A.

LIST OF EFFECTIVE PAGES

Page	Issue
iii	Original
iv	Blank
v thru vii	Original
viii	Blank
1-1 thru 1-3	Original
1-4	Blank
2-1 thru 2-3	Original
2-4	Blank
3-1 thru 3-9	Original
3-10	Blank
4-1	Original
4-2	Blank
5-1 thru 5-5	Original
5-6	Blank
6-1 thru 6-13	Original
6-14	Blank
7-1 thru 7-3	Original
7-4	Blank
8-1 thru 8-14	Original
9-1 thru 9-3	Original
9-4	Blank
A-1 thru A-4	Original
B-1 thru B-6	Original
C-1 thru C-12	Original
1, 2	Original

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

OVERVIEW

This reference manual provides information for interfacing the Burroughs Magnetic Tape Streamer Unit (MTSU) to a B 22 workstation. Magnetic tape provides more efficient and faster storage than floppy disks.

This document is organized as follows:

Section 1 provides an Overview, a brief description of the items needed to install and use the Burroughs Magnetic Tape Streamer Unit (MTSU).

Section 2, Installing Tape Streamer, is designed for the user installing Tape Streamer.

Section 3, Software Structure and Tape Format, provides general information about the software interface and the tape media.

Sections 4 through 7 describe the system from the point of view of the user who wants to operate the tape utilities. These sections presume a familiarity with the basic B 20 operations.

Section 8, Notes for the Programmer, provides more advanced software information. This section is helpful to the user writing tape application programs.

Section 9, Diagnostics, is designed for the person who will be responsible for troubleshooting hardware problems.

Appendix A lists the various error codes associated with Tape Streamer.

Appendix B, Accessing The Tapestreamer From The Languages, is designed for users who write applications to interface with the MTSU.

Appendix C gives examples of Pascal programs using the Tape calls.

INTRODUCTION

Using the Tape Streamer Utilities requires the following software and additional documentation:

Software

- B 20 Tape Streamer 4.0.
- BTOS 4.0.

Additional Documentation

- B 20 Operating System (BTOS) Reference Manual 4.0.
- B 20 Systems Standard Software Operation Guide 4.0.
- B 20 Systems Programmer's Guide Part 1.

TAPE STREAMER SOFTWARE

The B 22 Tape Streamer software provides three levels of access to the Burroughs Magnetic Tape Streamer Unit (MTSU). These are: Tape Backup Utilities (which include Tape Backup Volume, Tape Selective Backup, and Tape Restore), Tape Bytestreams (which include Tape Copy), and Direct Tape Services.

The Tape Backup Utilities function similarly to the standard archiving utilities with the convenience of magnetic tape. The Tape Backup Utilities can be used either from a standalone, master, or cluster workstation.

The Tape Bytestreams allow programs to access the tape in a device-independent manner consistent with BTOS Standard Access Method (SAM), which is common to all devices. Tape Bytestreams are useful for transferring information to and from non-B 20 Systems. Tape Bytestreams can only be used in cases where the tape format is not too machine specific.

The Direct Tape Services, also referred to as the Tape Server, allow direct control of tape through BTOS requests. The Tape Server is useful for exporting and importing data from magnetic tapes that have been created with very machine-specific information, such as blocking information imbedded in tape blocks, and so forth. The Tape Server is also used by the Tape Backup Utilities (when used from a cluster workstation) and Tape Bytestreams.

The choice of the above mentioned utilities depends greatly on the application. For disk backup and file transfer within the BTOS environment, the Tape Backup and Restore Utilities are the quickest and most convenient. We suggest using the Tape Backup Utilities rather than the Tape Copy Utility. The Tape Backup Utilities copy file information on the tape and contain redundant information to assure more reliable data transfers between systems. Use the Tape Copy Utility for importing and exporting data from non-B 20 systems.

SECTION 2

INSTALLING TAPE STREAMER

The Magnetic Tape Streamer Unit (MTSU) can be installed on any B 22 workstation. If the B 22 is a master workstation and the Tape Streamer controller board resides in that workstation, Tape Streamer is then accessible to all cluster workstations, whether B 22, B 21, or B 25. If the B 22 is a standalone or cluster workstation and the Tape Streamer controller board resides in that station, then Tape Streamer can only be used by that station.

HARDWARE INSTALLATION

MTSU Hardware is to be installed only by a qualified Burroughs Field Engineer.

DIAGNOSTICS

Refer to Section 9, Tape Streamer Diagnostics.

Contents of theDistribution Diskette

The B20TS4 consists of one diskette.

```
Directory <Sys>
   TapeBs.Obj                    TapeRq.Obj


Directory <Burroughs>
   TapeCopy.Run                  TapeRestore.Run
   TapeBackupVolume.Run          TapeInstall.Run
   TapeSelectiveBackup.Run       TapeConfigure.Run
```

SOFTWARE INSTALLATION

Follow the instructions below to install the B 20 Tape Streamer Software on the B 22 with a Winchester Mass Storage Unit.

- Set your Path as follows:

```
Path
  [Volume]                    Sys RETURN
  [Directory]                 Sys RETURN
  [Default file prefix]
  [Password]
  [Node]
```

  If your hard disk has a volume password, enter the password into the [Password] field before pressing the **GO** key.

- Insert the Distribution Diskette (B20TS4) in the drive [f0]. To install the software, the Software Installation command must be invoked. Do this by keying in:

  Software Installation

  Then press **GO**

If the Software Installation command is not available, the following Submit command may be used instead:

```
Submit
  File List           [F0]<Sys>Install.Sub
  [Parameters]
  [Force Expansion?]
  [Show Expansion?]
```

Then press **GO**

During the installation of the software, follow the instructions that are displayed on the screen.

- Remove the installation diskette and save it as the master copy.

SECTION 3

SOFTWARE STRUCTURE AND TAPE FORMAT

INTRODUCTION

The Tape Streamer software has functions that vary from high to low level. Some of these functions or capabilities depend upon others in the hierarchy.

TAPE BACKUP AND RESTORE UTLITIES

At the highest level are the Tape Backup utilities:

Tape Backup Volume
Tape Selective Backup
Tape Restore

These utilities allow the user to archive data in the same way one would backup using other archiving media, such as floppy disks. Archiving is much faster with tape than with floppy disks, and a hard disk can be archived on one reel of tape instead of several dozen floppy disks.

The safety features built into the Tape Backup utilities provide the preferred method for copying a file that was created on a B 20 system for transfer to another B 20 system.

The Tape Backup utilities are discussed further in Section 6.

TAPE BYTESTREAMS AND TAPE COPY

Tape Bytestreams are a set of procedures that read a tape as a purely sequential device. It looks for a tape pattern of file marks that designate the beginning and the end of a file.

Tape Bytestreams can interpret data input from a tape that may not have been generated on a B 20 system. For example, Tape Bytestreams allow a program running on a B 20 to read from a tape generated on non-B 20 system. This is possible only if the format of the tape conforms to certain rules. These rules are discussed in further detail in Section 7.

Tape Copy is a copy utility that depends on or uses Tape Bytestreams. (Tape Bytestreams can also be used by other programs. See Section 8.

DIRECT TAPE SERVICES

Direct Tape Services, also called the **Tape Server**, allows direct control of tape through BTOS requests. Details about the Tape Server and how to call its services are discussed in Section 8.

The importance of the Tape Server is that both the Tape Backup utilities and the Tape Bytestreams use and depend on its services. The Tape Server is the lowest level, or in a sense, the lowest common denominator of Tape Streamer software. In most cases the Tape Server must be installed if a Tape Backup utility or Tape Bytestreams (Tape Copy) is to be used. To determine when the Tape Server must be installed, refer to Section 4.

The hierarchical relationship of the parts of Tape Streamer software is shown in figure 3-1.

TAPE FORMAT

A tape that is produced or read by Tape Backup or Tape Copy or is used with Tape Bytestreams is shown in figure 3-2. A more detailed format for a tape file read or written by Tape Bytestreams is shown in figure 3-3. Detailed file format for the Tape Backup utilities appears in figure 3-4.

Any general tape format other than that shown in figure 3-2 (such as one with two tape marks used as a label within a file) cannot be used with these utilities and requires the writing of a special program based on the Tape Server. If this is the case, refer to Section 8.

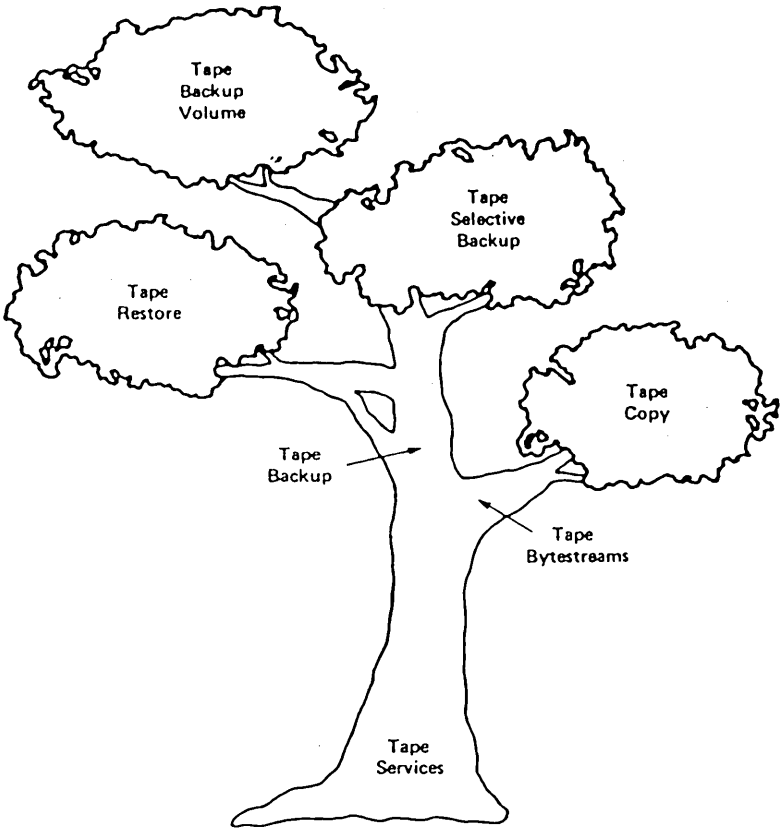

Figure 3-1. Tape Software Hierarchy.

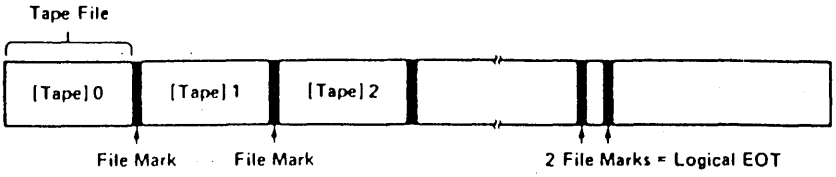

Figure 3-2. General Tape Format.

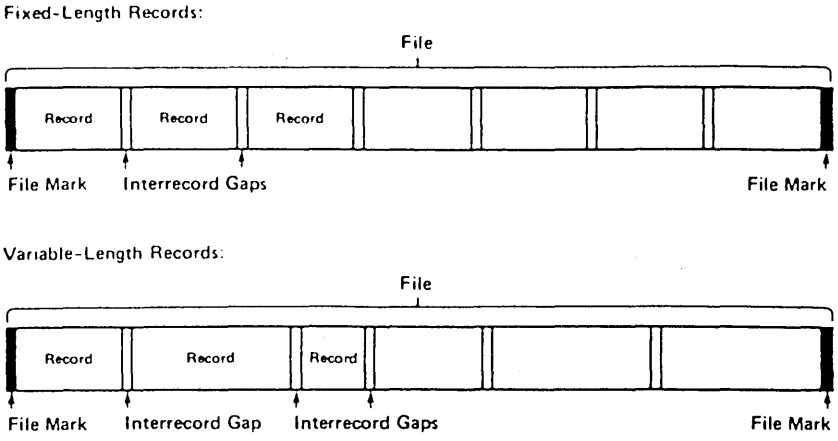

Figure 3-3. File Format Detail for Tape Bytestreams (Tape Copy).

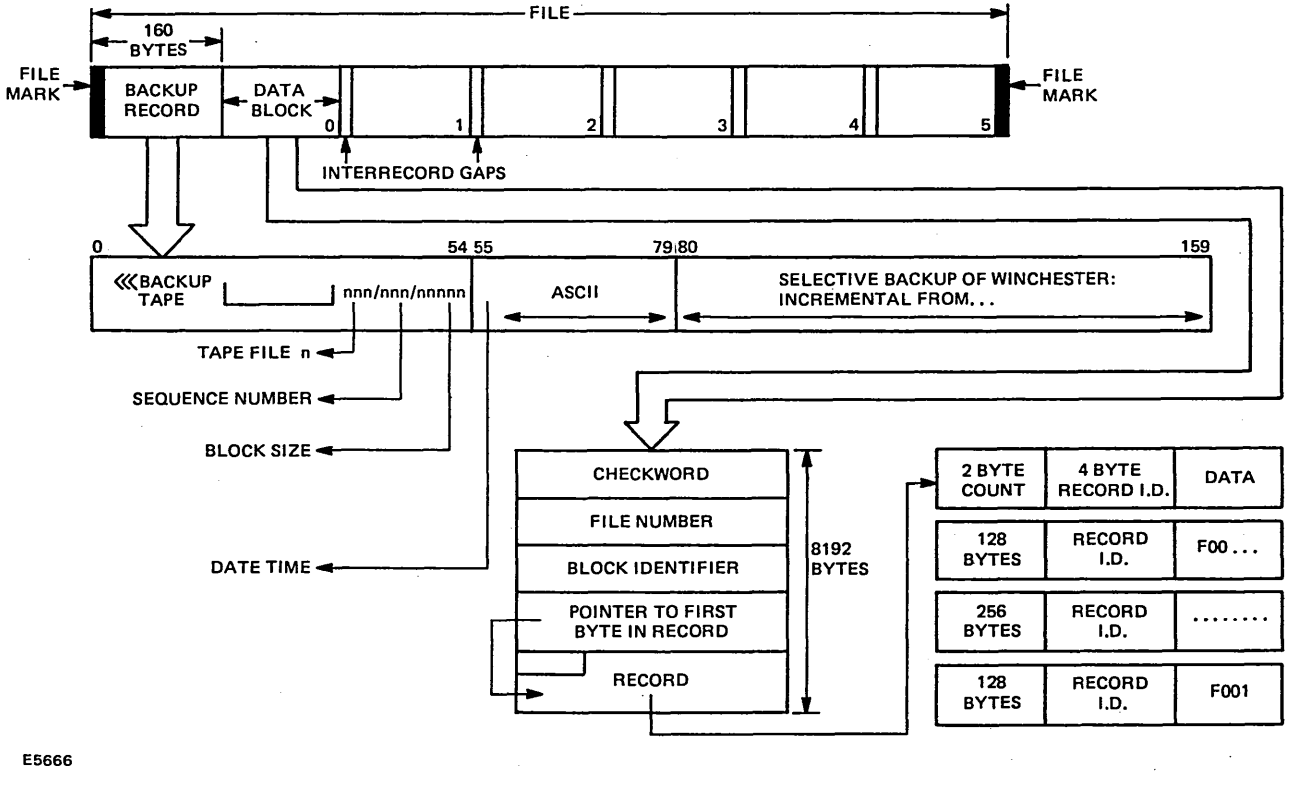

Figure 3-4. File Format for Tape Backup Utilities.

BASIC FORMAT

To understand the material in Sections 4 through 7, one does not need to understand all the details of the Tape Backup file format in figure 3-4. It is sufficient to see the tape's basic stucture. One tape file follows another, with one file mark between each file. The logical end of tape (EOT), which is the point after which nothing has been written, is indicated by two consecutive file marks.

When Tape Backup or Tape Copy are used to write a file to tape, the software causes the tape drive to write two tape marks at the end of any file written. The drive then backs up so the tape read/write heads are positioned between these two marks. Thus, this pair of marks may end up being the logical end of tape (EOT). If it is necessary to write another file to tape, the drive will overwrite the second file mark with the desired file leaving two files written to tape separated by one file mark. This procedure continues until the last file is written to tape ending with two file marks (the logical end of tape (EOT)).

TAPE FILES

Files on a tape differ from disk files in that they do not have filenames or directories. They can be refered to only by sequential numbers, in this case 0, 1, 2, and so on. Also, a tape file can contain several disk files. For example, when Tape Backup Volume is used to archive an entire disk, all the files archived from the disk are placed in one tape file.

If a file were to contain only one structure on the tape, there would be a large risk of losing data. The slightest amount of damage on the tape would prohibit the read head from reading a file. This is part of the reason why the tape file is divided into data blocks. If one block is damaged, the other data blocks are not effected and still can be read.

BLOCKS AND BLOCK SIZES

The user chooses the size of the blocks within a file. The effects of a large block size must be weighed against those of a small block size. Since the system leaves spaces between blocks, called "interrecord gap", the user gets more data on a single tape if the data blocks are large. Also, it is faster to write one big block than to write a series of smaller ones. On the other hand, it is safer to make blocks very small so that damage to one block does not cause the loss of much data. The user must reach a compromise between these factors in deciding the size of a data block. This topic is discussed futher in Section 5.

Sometimes the terms **block** and **record** are used interchangeably to refer to the units of data within a file. However, figure 3-4 shows that a data block can contain a small amount of header information in addition to containing a record or records. The terms **record** and **records** cannot be used interchangeably, because under the tape backup utilities, a block may contain many records, whereas under tape bytestreams, a block may contain only one record.

TAPE FILENAMES

Tape filenames have the format:

[TAPE]n

where n = 0, 1, 2, ...

The 'n' field specifies individual tape file positions on the tape. The tape files created by the B 20 systems are not directory devices. This means individual files on a tape cannot be referenced or assigned a name, as is the case on disk. Tapes, however, contain file marks (discussed earlier) which mark the separation of adjacent files on the tape. One specifies the name [Tape]0 for the file at the beginning of the tape, [Tape]1 for the file after the first file mark, and so on.

Two special cases of tape file names are particularly useful. They are [TAPE] and [Tape]+. The file name [TAPE] refers to the current position of the tape. By using this name, the system is told to write or read at the next meaningful location on tape (whether there is something there or not). Even if the number of the next file is known, [TAPE] may be used instead of [TAPE]n because [TAPE]n causes the tape to rewind to the beginning in order to count file marks, whereas [TAPE] just continues from the current position. The file name [Tape]+ refers to the logical end of tape (EOT). When this file name is used, the system is told to pass all existing files on tape and write after the last one.

HOW THE TAPE TRANSPORT WORKS

The MTSU has two heads, one behind the other, that float very close to the tape media and read and write to it. When the drive is writing, the first head (write head) writes and the second head (read head) reads what has been written. If the drive is writing and the read head cannot read what was written, a write error is reported.

SOFT ERRORS

If during a write, the read head cannot read what has been written, it reports a soft error. If you find that too many errors are being reported, first clean the drive heads. Dirty heads are the most frequent cause of "soft errors" of this kind. Further causes could be a faulty tape or a MTSU malfunction.

TAPE DRIVE MODES

The MTSU can operate in either start/stop mode or streaming mode.

In start/stop mode, the drive writes a file and then stops within the interrecord gap. To avoid damage to the tape, this mode runs the tape slowly.

In streaming mode, the drive runs tape much more quickly and does not stop within the interrecord gap at the end of a file. Instead, it ramps down slowly until it stops. It then backs up to a point considerably before the interrecord gap at the end of the file just written. When it is called upon to write the next file, it ramps up until it reaches the correct speed and then starts writing as it passes the end of the previously written file.

When the MTSU is accessed from the B 22 workstation where the tape controller resides, streaming mode can be used. When the MTSU is accessed from a cluster workstation, start/stop mode should be used because of the time required for the cluster to communicate with the master.

Keep in mind that what is written to the tape is the same, regardless of which mode is used during the write. Thus a tape can be read in start/stop mode that was written in streaming mode, and vice versa.

SECTION 4

TAPE SERVER

The Tape Server is an installed system service which performs tape handling functions. The Tape Server provides the BTOS interface for an application to access the MTSU (refer to Appendix B).

See figure 4-1 to determine when to install the Tape Server.

	Configuration 1		Configuration 2	
	B 22 Master with MTSU	B 20 Cluster	B20 Master	B 22 Cluster with MTSU
Install Tape Server to use Tape Backup?	NO	YES	N/A	NO
Install Tape Server to use Tape Copy?	YES	YES	N/A	YES

Figure 4-1. Tape Server Installation

INSTALLING TAPE SERVER

To install the Tape Server, give the command Install Tape Server. The following form appears:

```
Install Tape Server
 [Buffer size ? (default 2560 bytes)]
```

The default buffer size is the maximum size for a Burroughs cluster Operating System.

NOTE: The Tape Server must be installed on the B 22 workstation that contains the tape controller.

SECTION 5

TAPE CONFIGURATION FILES

A configuration file is a collection of parameters needed to govern the way a device such as a printer or a tape device interacts with the workstation. The parameters are placed in this file and the user is given access to them.

For the interface with tape, there are three kinds of configuration files, one for Tape Backup from the master workstation, one for Tape Backup from the cluster workstation, and one for the use of Tape Bytestreams. They are similiar but not identical. (As a user, your most likely contact with Tape Bytestreams is through Tape Copy.)

Default tape configuration files are created at installation of the Tape Streamer software, or when using the Tape Configuration Utility. To examine the default parameters invoke the Configure Tape File command (described below). These configuration files are changed only if new values are specified for the optional parameters in the form.

CONFIGURATION FILES FOR TAPE BACKUP

For Tape Backup utilities, the configuration files allow the size of data blocks to be specified within a file on tape. Use of stop/start mode or streaming mode can also be specified, and whether the tape is to be rewound at the end of the operation.

The default configuration file names are <Sys>Mws>TapeBackupConfig.sys for the master workstation and <Sys>Cws>TapeBackupConfig.sys for a cluster workstation.

NOTE: When a B 22 is clustered to a B 22 Master with a tape controller board installed and a tape command is given from the cluster, the master tape configuration file is used. The file which should be used is the cluster tape configuration file. This results in an error 45 whenever a Tape Backup utility is used. To avoid this error use the Configure Tape File command and change the record size of the master configuration file from 8192 bytes to 2048 bytes, and change Streaming mode from YES to NO.

CONFIGURATION FILES FOR TAPE BYTESTREAMS

The configuration files for Tape Bytestreams allows specification of the same parameters for Tape Backup, and also permits choice of a variable record length. The tape bytestream configuration file also provides a method of specifying parameters for use with tapes produced on a non-B20 system.

The name of the default configuration file for Tape Bytestreams is <Sys>TapeConfig.sys.

Tape Configure Utility

Access to the tape configuration files is gained by invoking the Configure Tape File command while in the executive. The following form appears:

```
Configure Tape File
  Configuration type (Master, Cluster, or Bytestream) ____
  [File name]                                         _____
```

Fields:

Configuration type (Master, Cluster, or Bytestream)

Specifies the type of configuration file to reconfigure. Appropriate entries are Master (or M), Cluster (C) or Bytestream (B).

[File name]

Specify a file name to create a configuration file other than that of the default configuration file name. Once this alternative configuration file is created, use it within the Tape name field that appears within the Tape Backup Utilities or Tape Copy forms by appending it to the tape file specification.

For example, suppose you want to perform a Selective Backup to the third file on the tape and you want to use the parameters specified in a specially created configuration file. In this case we will call the file SheriTapeConfig.file. Use the file specification [TAPE]2&SheriTapeConfig.sys in the Tape name field of the Tape Selective Backup form.

When the form has been filled out, press GO. A list of options are displayed (the default values for the tape configuration files that were created during software installation appear in figure 5-1). For Master and Cluster types, the options appear as follows:

```
Tape Parameters
  [Record Size]
  [Streaming Mode?]
  [High Density?]
  [Rewind Tape On Close?]
```

For the Bytestream type, the Tape Parameters form is the same, with the addition of a final field:

[Variable Length Records?]

[Record Size]

Specify the size, in bytes, of each physical tape record (For the Tape Backup utilities, this is the same as specifying the data block size). For the Tape Backup utilities, the acceptable range is from 1024 to 16384 bytes.

For Tape Bytestreams, the acceptable range is from 1 to 16384 bytes. If the variable length records option is selected, this parameter specifies the largest record that can occur.

[Streaming Mode?]

Specify YES to perform read/write operations in streaming mode. It is advised that bytestreams and backups from a cluster workstation not be used in streaming mode.

[High Density?]

Currently not implemented.

[Rewind Tape On Close?]

Specify YES if the tape is to be rewound to the load point when the tape operation has completed.

[Variable length records?]

The Default is NO if the tape records are not of variable length.

Specify YES if the tape records are not all the same size. This parameter is used for tape bytestreams only.

Example:

If you are creating a configuration file to read or write a tape where the size of each record is 50 bytes and the user wants 50 records per block, the record size will be:

50 Bytes/Record X 50 Records/Block = 2500

The record size (sometimes referred to as the block size) will be 2500 Bytes/Block.

Field	Default Value		
	Master	Cluster	ByteStreams
[Record Size]	8192	2048	512
[Streaming Mode?]	Yes	No	Yes
[High Density?]	No	No	No
[Rewind Tape on Close?]	No	No	No
[Variable length record?]	N/A	N/A	No

Figure 5-1. Installed Parameter Values for Tape Configuration Files

SECTION 6

TAPE BACKUP UTILITIES

INTRODUCTION

These three utilities have the same user interface as their floppy counterparts.

The tape utilities support the following features:

- Multiple backups on one tape.
- Backup to multiple tape volumes.
- Recovery of tape errors.

The Tape Backup Utilities are invoked either from the master or cluster workstations. If the utilities are invoked by cluster workstations, the Tape Server must be installed at the B 22 workstation which contains the tape controller.

MULTIPLE BACKUPS ON ONE TAPE

Multiple backups can be placed on one tape. Fill in [Tape]0, [Tape]1, [Tape]2, and so on, at each invocation of the backup command. If backups are performed in succession, use [TAPE] for successive invocations of the backup utilities.

BACKUP TO MULTIPLE TAPE VOLUMES

A large backup may use more than one tape. In this case, the last record on the tape indicates continuation to another tape. A new tape is mounted and the backup continues on that tape.

There is one limitation however; the continuation must take place at the beginning of the second tape.

RECOVERY OF TAPE ERRORS

Each tape block is numbered sequentially on the tape. The sequence number is surrounded by a checkword, followed by the file number where the data block is located, and then by a sequential block identifier number. If an error occurs when the tape block is written, it is re-written again (further down the tape) with the same sequence number.

As the tape is read, each sequence number is monitored. Duplicate sequence numbers are ignored and not passed to the backup program as data.

If either the tape or header records are unreadable, the tape is scanned until a valid data block is found. The tape is considered a valid backup tape by Tape Restore if the

data block conforms with the format of a tape backup data block. Tape Backup does not allow data to be appended to a tape which has unreadable tape or file headers.

TAPE BACKUP VOLUME

The Tape Backup Volume Utility is used to archive an entire disk to one or several tapes.

Tape Backup Volume can:

- archive all files,
- archive only those files modified on or after a specified date, or date and time (incremental backup),
- verify the integrity of the volume control structures without backup of the files,
- recognize a formatted volume that is too scrambled to be automatically recognized by the Operating System, and
- optionally write the log of the Backup Volume operation to a file or printer.

One question, before the actual backup is performed, is "Should Clusters be disabled?". The master workstation can be backed up without disabling the clusters. In this case the files that are exclusively open are not backed up. To back up every file on disk, disable the clusters beforehand.

OPERATION

Tape Backup Volume is able to recover all the file information from a volume (even a scrambled one) by accessing only one of the two Volume Home Blocks and one of each pair of File Header Blocks.

NOTE

Tape Backup Volume is intended for use by the system manager. It displays passwords for all the directories.

INVOKING THE TAPE BACKUP VOLUME UTILITY

Enter the command Tape Backup Volume and the following form appears:

Form:

```
Tape Backup Volume
  Volume or device name                                    _____________
  [Volume or device password]                              _____________
  [Incremental from (e.g., Mon Jun 1 1981 8:00 pm)]        _____
  [Suppress backup?]                                       _____________
  [Suppress verification?]                                 _____________
  [Tape name]                                              _____________
  [Delete existing archive file?]                          _____________
  [Log file]                                               _____________
  [Display structures?]                                    _____________
```

Fields:

Volume or device name

is the volume or device name of the disk to be backed up.

[Volume or device password]

is the password (if one exists) of the volume or device to be backed up.

The volume password must be supplied if the volume is recognized by BTOS. Otherwise, the device password must be specified.

[Incremental from (e.g., Mon June 1 1981 8:00 pm)]

is the date and time from which files are to be backed up. Only files modified on or after the specified date and time are backed up.

[Suppress backup?]

is Yes or No (the default).

Yes verifies the integrity of the volume control structures without performing a backup.

[Suppress verification?]

is Yes or No (the default).

Yes performs only the backup pass, and suppresses the verification pass.

[Tape file]

is the name of the tape file at which to archive the files.

The Default parameter is [TAPE]. Specify this parameter to begin backup at the current tape position. This is useful when archiving several disks on one tape.

Specify [Tape]0 to begin backup at the beginning of the tape.

Specify [Tape]1 to begin backup after the first file mark, [Tape]2 after the second file mark, and so on.

Specify [Tape]+ to backup the volume at the logical end of tape.

To use a a specially created configuration file, append it to the tape file name, for example :

```
[TAPE]2&SheriTapeConfig.file
```

[Delete existing archive file?]

is Yes or No (the default).

If [Delete existing archive file?] is No, Tape Backup Volume prompts the following if the tape already contains a valid archive file:

```
Tape already contains a backup performed on
mm/dd/yy/time.  Backup of XXX

[OVERWRITE OK?]
(Press GO to confirm, CANCEL to deny, or FINISH to
return to the Executive.)
```

[Log file]

is the name of the file to which to write a report of the backup.

If the log file exists, the log is appended to it. If it does not exist, it is created.

If a log file is not named, the log appears only on the video display.

NOTE: During a Tape Backup, if the log file exceeds 99 pages, the following page numbers (that is, 100, 101, and so forth) are displayed incorrectly.

[Display structures?]

is Yes or No (the default).

Yes gives a detailed analysis of the volume control structures.

Once this form is completed, press **GO** to start the backup. Now the system takes over and instructs with screen messages as needed. A warning message may be encountered: "volume verification may be suspect". This message appears because you are using a file on disk. The message means that there are open files on the disk. These open files will not be backed up.

As the backup pass is made, the names of the files are written to the screen as they are archived. After the backup pass is complete, and YES was not specified to suppress verification, the system appears to sit idle while a verification pass is performed. Eventually the screen displays the results of the verification and returns to the Executive.

TAPE SELECTIVE BACKUP

The Tape Selective Backup utility copies individual files or directories from one disk volume to tape. (The Tape Backup Volume utility, in contrast, copies an entire volume.)

Tape Selective Backup allows users to archive their personal files and requires only read access to the files being archived.

Tape Selective Backup can:

- archive individual files or directories,
- archive only those selected files modified on or after a specified date, or date and time (incremental backup),
- optionally write the log of the Selective Backup operation to a file or printer. (The log always appears on the video display.)

Invoking Tape Selective Backup

Enter the command Tape Selective Backup and the following form appears:

```
Tape Selective Backup
  File List
  [Incremental from (e.g., Mon Jun 1 1981 8:00 pm)]
  [Confirm each?]
  [Tape name]
  [Delete existing archive file?]
  [Log file]
```

Fields:

File list

is a list of the files to archive. The list can include single files, directories or sets of files (using wild cards).

[Incremental from (e.g. Mon Jun 1 1981 8:00 pm)]

is the date and time from which files are to be archived. Only files modified on or after the specified date and time are archived.

NOTE: When a future date is entered the utility will proceed with the archive. It will not display the error message "The date given is in the future" as is done in the Tape Backup Volume command.

[Confirm each?]

is Yes or No (the default). If [Confirm each?] is No, the files are backed up without individual confirmation.

[Tape name]

is the name of the tape file at which to archive the files.

The Default parameter is [TAPE]. Specify this parameter to begin backup at the current tape position. This is useful when archiving several disks on one tape.

Specify [Tape]0 to begin backup at the beginning of the tape.

Specify [Tape]1 to begin backup after the first file mark, [Tape]2 after the second file mark, etc.

Specify [Tape]+ to backup the files at the logical end of tape.

To use a specially created configuration file, append it to the tape file name, for example :

[TAPE]2&SheriTapeConfig.file

[Delete existing archive file?]

is Yes or No (the default).

If [Delete existing archive file?] is Yes, any existing archive file is automatically overwritten.

If [Delete existing archive file?] is No, Tape Backup Volume prompts the following if the tape already contains a valid archive file:

> Tape already contains a backup performed on mm/dd/yy/time. Backup of XXX
>
> [OVERWRITE OK?]
> (Press GO to confirm, CANCEL to deny, or FINISH to return to the Executive.)

[Log file]

is the name of the file to which to write a report of the backup.

If the log file exists, the log is appended to it. If it does not exist, it is created.

If a log file is not named, the log appears only on the video display.

TAPE RESTORE

The Tape Restore utility restores files from an archive tape created by either the Tape Backup Volume or Tape Selective Backup utilities.

Tape Restore:

- Restores an entire volume, creating any needed directories on the destination volume,
- Restores selected files to the same or different file specifications,
- Restores a file's characteristics (creation date, protection level, etc.) as they existed at the time of backup, and
- optionally writes the log of the Restore operation to a file or printer. (The log always appears on the video display.)

Invoking Tape Restore

Tape Restore is similar to the other Tape Backup utilities, except that data is transferred from the tape to disk. Enter the command Tape Restore and the following command form appears:

Form:

```
Tape Restore
  [Tape name]                          _____
  [File list from]                     _____
  [File list to]                       _____
  [Overwrite ok?]                      _____
  [Confirm each?]                      _____
  [Sequence number]                    _____
  [Merge with existing file?]          _____
  [List files only?]                   _____
  [Log file]                           _____
```

Fields:

[Tape name]

is the name of the tape file to be restored.

The default is [Tape], the current position of the tape.

[TAPE]+ is not a valid entry in the case of Tape Restore.

A specially created configuration file may be used by appending it to the tape file name, for example:

[Tape]2&SheriTapeConfig.file.

However, Tape Restore will restore the files using block size information embedded in the tape file. It will not use the block size as indicated by the record size field of the specified configuration file.

[File list from]

is a list of the files to restore. The file specifications are of the form:

<dirname>filename

No volume name is permitted.

If this field is defaulted, all the files on the archive file are restored. In this case, there must be nothing entered in the [File list to] field.

To restore an entire directory, type:

<dirname>*

[File list to]

is a list of the file specifications to receive the restored files. The file specifications are of the form:

[volname]<dirname>filename

The volume name and directory name are optional. If only a file name is specified, the files are restored to the volume and directory specified at 'Login'.

If this field is defaulted, the files are restored to the same directory and file as when archived. However, the files will be restored to the logged-in volume.

The wild card character * can be used for directory name and file name.

[Overwrite ok?]

is Yes or No or may be left blank (the default).

If [Overwrite ok?] is left blank (the Default), the user is prompted for confirmation before an existing file is deleted.

If [Overwrite ok?] is Yes, an existing file of the same name is automatically deleted before a new one is restored.

If [Overwrite ok?] is No, the files will not be restored, if there are exisiting files of the same name.

[Confirm each?]

is Yes or No (the default).

If [Confirm each?] is Yes, Restore prompts for user confirmation before an existing file is restored.

If [Confirm each?] is No, Restore proceeds without user interaction.

[Sequence number]

is the first volume with which to begin the Tape Restore operation. Enter a value in this field to restore from an archive which required two or more reels of tape.

The default is 1, the first volume of the archive file.

If the Restore operation does not begin with the first volume, Restore creates any required directories 10 pages in size with a protection level of 15 (unprotected).

[Merge with existing file?]

is Yes or No (the default).

If [Merge with existing file?] is Yes, and the corresponding sector of the archive file is unreadable, the sectors of the target file are bypassed.

If [Merge with existing file?] is No, and any input/output error is detected on the archive file, the sectors of the target file are written with 0s.

[List files only?]

is Yes or No (the default).

Yes lists the files on the archive file: it does <u>not</u> restore them.

[Log file]

is the name of the file to which to write a report of the Restore operation.

If the log file exists, the log is appended to it. If it does not exist, it is created.

If a log file is not named, the log appears only on the video display.

NOTE: When a Tape Restore is performed on a tape file which was created through the Tape Selective Backup utility, any directories which are created have a page count of 10. This is true regardless of the page count of the original directories. A way around this is to use Tape Backup Volume or create any needed directories before the Tape Restore is used.

When a Tape Restore is performed on a tape file which was created through the Tape Selective Backup utility, any directories that are created have a protection level of 15. This is true regardless of the protection level of the original directories. A way around this is to use Tape Backup Volume or create any needed directories before the Tape Restore is used.

SECTION 7

TAPE COPY UTILITY

Tape copy allows users to transport data between B 20 and non-B 20 systems. Its operation is similar to the Executive's Copy Utility.

Before using the Tape Copy Utility to create or read tapes produced from other systems, it is important to follow the checklist below.

1. What is the size of records on the tape?

   If the answer is known, use the Tape Configuration utility to specify this record size.

   If the answer is not known, there is no way to determine it. In this case, specify the maximum possible record size, (using the Configure Tape File command) and enter yes in [Variable Records?].

2. Does the end of each tape block contain padding?

   If this is the case, Tape Copy copies the padding. To remove the padding, write a post processing program.

3. Is each tape block fixed in length or does each block vary in size?

   Specify the fixed length in the record size field of the Tape Configuration utility or enter YES in the variable records field for records of variable length.

4. Is the data EBCDIC or ASCII?

   If the tape data is in EBCDIC, tape copy can read it; however, a conversion program must be written to translate the copied file. If the file being read is in ASCII, then no translation program is required.

   If the output file to tape is to be written in EBCDIC, then a conversion must be written to translate the file from ASCII to EBCDIC. After the file has been translated, it may be copied with the Tape Copy utility.

5. Do multiple file marks exist at the end of each tape file?

   Tape Bytestreams are set up to read two file marks in a row as the logical end of tape (EOT). If the tape contains two file marks which mean anything else besides the logical EOT, a program using direct tape services (Tape Server) to interpret it must be written.

6. Is control information embedded within the data of each record?

   If this is the case, write a conversion utility.

7. Is the tape to be created or read at 1600 bpi?

   The MTSU reads and writes at 1600 bpi.

Once the answers to these questions have been found, make the determination whether or not, the Tape Copy Utility or Tape Bytestreams can be used. If the tape contains fixed-length tape records, with one file mark at the end of each tape file, and two marks at the logical end of tape, Tape Bytestreams and the Tape copy Utility can be used.

For more complex tape formats, use the Direct Tape Services provided by the Tape Server.

INVOKING THE TAPE COPY UTILITY

Before using the Tape Copy Utility, install the Tape Server at the B 22 workstation that contains the tape controller. In addition, use the Tape Configure command to reflect the parameters of the tape to be created or read (if necessary). Enter the command Tape Copy. The following form appears:

```
Tape Copy
  File From                    _____
  File To                      _____
  [Overwrite ok?]              _____
  [Confirm each?]              _____
```

Fields:

File From

Specify the source file name. If copying a file from tape to disk, specify the tape name. If copying a file from disk to tape, specify the disk file name.

File To

Specify the destination file name. If copying a file from tape to disk, specify the disk file name. If copying a file from disk to tape, specify the tape name.

NOTE: Wildcards are not permitted in the "File from" and "File to" parameters.

[Overwrite ok?]

is Yes or No or may be left blank (the default). This field is only applicable when copying from tape to disk.

If [Overwrite ok?] is left blank (the Default), the user is prompted for confirmation before an existing file is deleted.

If [Overwrite ok?] is Yes, an existing file of the same name is automatically deleted before a new one is restored.

If [Overwrite ok?] is No, and if there are existing files with the same name, the files will not be restored.

[Confirm each?]

is YES or NO (the default).

Specify YES if you wish to confirm the operation.

Note: A null file (one containing nothing) cannot be written to tape with this utility. If attempted, 128 sectors of meaningless information is written to the tape.

SECTION 8

NOTES FOR THE PROGRAMMER

The Tape utilities can be used without knowledge or awareness of the underlying tape software structures (through Tape Bytestreams and Tape Server). In some cases, however, the need to write a special application may arise. This section describes some of those situations and the facilities available to the programmer to solve them.

An example is to read from a tape that was not generated on a B 20. If the tape is in EBCDIC but has the correct file format, it can be read through an application using Tape Bytestreams. From within this application, the file can be translated from EBCDIC to ASCII.

In order to use Tape Bytestreams or Tape Services, the Tape Server must be installed. (Refer to section 4.)

USING TAPE BYTESTREAMS

Tape Byte Streams allow users to read or write to tape using the standard B 20 bytestream interface (this topic is discussed in detail in the BTOS Operating System Manual). With Tape Bytestreams, you can read or write to tape using the standard Burroughs bytestreams interface.

In read mode, records are read from the tape as a sequence of bytes until a filemark is encountered. The user is not aware of the record blocking (size of records).

In write mode, the record size is obtained from the tape configuration file (see Section 5).

Certain problems cannot be handled through the use of Tape Bytestreams. One of the most important, is reading a tape in which file marks have been used in pairs to designate something other than the logical end of tape (EOT).

The tape services are as follows:

- OpenTape
- CloseTape
- ReadTapeRecords
- WriteTapeRecords
- TapeOperation
- PurgeTapeUser
- TapeStatus

The Tape Server (described in section 4) **does not** perform record blocking outside the user buffer area. This implies that cluster users may not write tape records larger than the maximum size request available to the cluster (default is 2560 bytes). Otherwise, an error 45 will be returned to the application.

Services

OpenTape

The OpenTape service reserves the tape drive for the requestor's exclusive use. The tape is not positioned by the OpenTape service.

A "tape handle" is returned to the user. This tape handle must be provided to the user on subsequent tape requests.

Procedural Interface

```
OpenTape (pbTapeName, cbTapeName, fStreaming,
          fHighDensity, fClear, mode, pThRet) : ErcType
```

where:

pbTapeName cbTapeName	is the memory address of tape to open (for example, [Tape]).
fStreaming	is a flag which indicates whether data transfer operations are to be performed in streaming mode.
fHighDensity	Must be true.
fClear	is a flag which indicates whether data errors require a TapeStatus request to be issued to clear the error condition.
mode	is a word which contains 'mr' or 'mw' for mode read and mode write respectively. An error is returned if a tape without a write-ring is opened for write.
pThRet	is the memory address of a word in which the tape handle is returned.

Request Block

Offset	Field	size (bytes)	Contents
0	sCntInfo	2	6
2	nReqPbCb	1	1
3	nRespPbcb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	190
12	fClear	1	
13	fStreaming	1	
14	fHighDensity	1	
15	reserved	1	
16	mode	2	
18	pbTapeName	4	
22	cbTapeName	2	
24	pThRet	4	
28	cbThRet	1	2

CloseTape

The CloseTape service closes the tape making it available to other users.

The tape position is not affected by the CloseTape request.

Procedural Interface

```
CloseTape (th) : ErcType
```

where:

th	is the tape handle returned by the OpenTape service.

Request Block

Offset	Field	size (bytes)	Contents
0	sCntInfo	2	2
2	nReqPbCb	1	0
3	nRespPbCb	1	0
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	194
12	th	2	

ReadTapeRecords

The ReadTapeRecords service reads the next "n" fixed-length records from tape into the user buffer.

The length of the record is computed by dividing the number of records by the size of the buffer.

If nRecords is negative, the number of records to read is interpreted as the 2's complement of nRecords and the records are read in the reverse direction (towards the load point).

If an error condition occurs, pCRecordsRet reflects the number of records successfully read. pCbRet is updated with the number of bytes read.

If an error condition occurred and the fClear flag was false when the tape was opened, a TapeStatus must be issued prior to any subsequent reads. Tape reads issued without the intervening Tapestatus request result in the Tape Error Outstanding status code (9039). This is to allow the user to return to the application where tape error recovery may be performed.

Procedural Interface

```
ReadTapeRecords (th, nRecords, pb, cb, pNRecordsRet,
                 PCbRet) : ErcType
```

Where:

th	is the tape handle returned by the OpenTape service.
nRecords	is the number of records to read.
pb	is the memory address in the user area to store the tape records as read.
cb	is the buffer size in bytes of the user area. The size of each record is computed as cb/nRecords.
pNRecordsRet	is the memory address to store the number of records which were successfully read.
pCbRet	is the memory address to store the number of bytes transferred to the user buffer.

Request Block

Offset	Field	size (bytes)	Contents
0	scntinfo	2	6
2	nReqPbCb	1	0
3	nRespPbcb	1	3
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	191
12	th	2	
14	nRecords	2	
18	pb	4	
22	cb	2	
24	pNRecordsRet	4	
28	cbRet	1	2
30	pCbRet	4	
34	cbRet	1	2

WriteTapeRecords

The WriteTapeRecords service writes one or several fixed-length records.

The length of each record is computed by dividing the size of the buffer by the number of records to be written.

If an error condition occurs, the actual number of records written is returned and the count of bytes written is returned.

Procedural Interface

```
WriteTapeRecords (th, nRecords, pb, cb, pCRecordsRet,
                  pCbRet) : ErcType
```

where:

th	is the tape handle returned by the OpenTape service.
nRecords	is the number of records to write.
pb	is the memory address of the bytes to write on tape.
cb	is the number of bytes to write. The size of each record is computed as cb/n Records.
pCRecordsRet	is the memory address to store the number of records which were successfully read.
pCbRet	is the memory address to store the number of bytes written to tape.

Request Block

Offset	Field	size (bytes)	Contents
0	scntinfo	2	6
2	nReqPbCb	1	1
3	nRespPbcb	1	2
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	192
12	th	2	
14	nRecords	2	
16	Reserved	2	
18	pb	4	
22	cb	1	2
24	pCRecordsRet	4	
28	cbRet	1	2
30	pCbRet	4	
34	cbRet	1	2

TapeOperation

The TapeOperation service allows users to issue various commands to the tape drive. These include such commands as: Rewind, Unload, Skip Record(s), Search Mark(s), Write Tape Mark(s), Erase and Erase Tape.

Operations which do not make use of the "subOp" field described below are labelled with subOp = "x". Operations which make use of the subOp field have "n" labels. If n is greater than 0 the Operations proceed with the tape moving forward. If n is less than 0, the operations proceed with the tape moving in the reverse direction, for example:

TapeOperation (th, 0, 3, -1) backspaces one record.

Procedural Interface

```
TapeOperation (th, fFast, command, subOp) : ErcType
```

where:

th	is the tape handle returned by the OpenTape service.
fFast	is a flag which indicates whether these operations should be done in streaming mode or not. This flag overrides the parameter given in the OpenTape command. This allows the user to read/write records in incremental mode (25 ips), and to skip records in streaming mode (100 ips).
Command	is the command code of the operation to be performed.
argument	is the argument to be supplied to certain commands. For example, TapeOperation (SkipRecord, 3) skips the next 3 records.

Commands

Rewind (1,X)	Rewinds the tape to the load point.
Unload (2,X)	Rewinds and unloads the tape.
SkipRecord (3,n)	Skips "n" records.
SearchMark (4,n)	Searches for "n" tapemarks.
WriteTapeMark (5,n)	Writes "n" tapemarks.
Erase (6,n)	Erases "n" * 3.5 inches of tape.
EraseTape (7,n)	Erases the tape until the physical end of tape (EOT) is reached.

Request Block

Offset	Field	size (bytes)	Contents
0	scntinfo	2	6
2	nReqPbCb	1	0
3	nRespPbcb	1	0
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	193
12	th	2	
14	fFast	1	
15	command	1	
16	subop	2	

PurgeTapeUser

The PurgeTapeUser service aborts all the user's outstanding tape requests and releases all the tape drives reserved by the user.

Note: Issuing a PurgeTapeUser request while a tape is rewinding causes the tape drive to stop and an error 4, is returned to the calling procedure.

Procedural Interface

```
PurgeTapeUser  (work) : ErcType

work                  is a temporary work area for the
                      service.
```

Request Block

Offset	Field	size (bytes)	Contents
0	scntinfo	2	2
2	nReqPbCb	1	0
3	nRespPbcb	1	0
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	195
12	work	2	

TapeStatus

The TapeStatus request allows users to determine the status of the tape drive. The TapeStatus requests also clears outstanding error conditions.

The status returned is as indicated below. The number within parentheses specifies the bit value of the status indication (that is, status = 4Ah indicates that the tape is on-line and write-protected).

Write Protect (2)	Indicates whether the tape is currently write-protected.
Formatter Busy (4)	Indicates that the formatter is busy and is not ready to accept commands.
Ready (8)	Indicates the tape drive is currently ready for new commands.
End of Tape (10h)	Indicates that the end of the tape has been reached.
Load Point (20h)	Indicates that the tape is at the load point.
On-Line (40h)	Indicates that the tape drive is on-line.

Procedural Interface

```
TapeStatus  (th, pStatusRet) : ErcType
```

where:

th	is the tape handle returned by the OpenTape service.
pStatusRet	is the memory address of the one word tape status to be updated.

Request Block

Offset	Field	size (bytes)	Contents
0	scntinfo	2	6
2	nReqPbCb	1	0
3	nRespPbcb	1	1
4	userNum	2	
6	exchResp	2	
8	ercRet	2	
10	rqcode	2	196
12	th	2	
18	pStatusRet	4	
22	cbRet	1	2

SECTION 9
TAPE STREAMER DIAGNOSTICS

The Tape Streamer diagnostics test the following:

- Proper controller/cabling installation.
- Proper tape operation.

INSTALLING MAGTAPE DIAGNOSTICS

- Insert the Distribution Diskette (B20TS4) in the drive [f0].

  Method 1

  Invoke the bootstrap command as follows:

  ```
  Bootstrap
     Filename [f0]<sys>SysImage.sys  GO
  ```

  Method 2

  Use this method if the bootstrap command is not available:

  Reboot the B 22 which contains the tape controller and the Winchester mass storage unit.

OPERATION

The Tape streamer diagnostic's operation is similar to other B 20 diagnostics.

The diagnostic is loaded into the workstation's memory, displays a header, and prompts:

Change Parameters?

To run a test sequence with standard parameters, type N followed by RETURN. The diagnostic will run through 5 tests verifying the proper installation and operation of the tape drive.

If Y RETURN is answered, the tests available are displayed. This prompt is displayed:

Test # to run:

Type in each test number to run followed by RETURN. A beep is sounded if no such test number exists. When all tests to run have been specified, press the RETURN key.

All diagnostic errors are text messages that suggest actions be taken.

TEST DESCRIPTIONS

Test 1: Tape Controller Installation

Test 1 verifies that the Tape Controller has been correctly installed. The diagnostic will attempt to detect improper switch settings. Some installation errors may cause the diagnostic to freeze, however. If this occurs, check the following: switch setting S2 on the CPU board, pins 31-50 on the Tape Controller board.

Test 2: Tape Controller Interrupts

Test 2 verifies the correct operation of interrupts from the Tape Controller.

Test 3: Tape Drive Status Test

Test 3 verifies the correct installation of the cables to the tape drive and the functioning of the tape drive itself. Most errors from this test are most likely caused by incorrect cable installation.

Test 4: Tape Drive Positioning Test

Test 4 performs various tape positioning functions and tests the results.

Test 5: Tape Drive Write/Read Tests

Test 5 writes and reads one hundred records. Errors from this test indicate drive, cabling and/or media problems.

Parameter Prompts

After the test sequence is selected, test parameters can be varied. These parameters only should be modified by persons knowledgeable with Tape Controller operation.

Times to run:

The default value is 1. The test sequence selected will run for the specified number of times unless an error occurs or the FINISH key is pressed to stop testing.

Tape Controller Channel Address:

The default is 80AAh. Specifies the channel address of the Tape Controller board.

Tape Controller Interrupt Level:

The default is 6. Specifies the interrupt to be used when interrupting.

Tape Drive Number:

Must be zero.

Tape Speed/Density Option

The default is YES. Asserts the Speed/Density signal. For the MTSU asserting this signal selects streaming mode. Specify NO, to test the tape drive in incremental mode.

Data Buffer Size:

Specifies the size of the tape record during test 5. The acceptable range is 1024 to 16384 bytes.

Stop on Error?

The default is YES.

APPENDIX A

ERROR CODES

This section lists errors encountered during tape operation. Errors 9001-9031 are produced by the Tape Controller; all other errors are generated by the various Tape Streamer software.

Code	Description
9001	Timed out waiting for expected Data Busy false.
9002	Timed out waiting for expected Data Busy false, Formatter Busy false and Ready true.
9003	Timed out waiting for expected Ready false.
9004	Timed out waiting for expected Ready true.
9005	Timed out waiting for expected Data Busy true.
9006	A memory time-out occurred during a system memory reference.
9007	A blank tape was encountered where data was expected.
9008	An error occurred in the micro-diagnostic.
9009	An unexpected EOT was encountered during a forward operation, or Load Point during a reverse operation.
9010	A hard or soft error occurred which could not be eliminated by retry.
9011	A read overflow or write underflow occurred. This error indicated that the FIFO was empty when data was requested by the the tape during a write, or full, when the tape presented a byte during a read.
9012	There is no free preallocated data buffer.
9013	A read parity error occurred on the byte interface between the drive and the Tape Controller.
9014	An error was detected while calculating a checksum on the Tape Controller PROM.

9015	A tape time-out occurred because the tape drive did not supply an expected read or write strobe. This normally occurs when attempting to read a larger record than was written. It may also occur during a write due to a damaged tape.
9016	Tape not ready.
9017	A write was attempted on a tape without a write-enable ring.
9018	Not used.
9019	The diagnostic mode jumper was not installed while attempting to execute a Diagnostic command.
9020	Not used.
9021	An unexpected filemark was encountered during a tape read.
9022	An error in specifying a parameter was detected by the Tape Controller. The usual cause is a byte count which is zero or too large.
9023	Not used.
9024	An unidentifiable hardware error occurred.
9025	A streaming read or write operation was terminated by the operation system or disk.
9026-9031	Reserved.
9032	Bad tape handle. A tape request was issued with an unrecognized tape handle. This is typically caused by a error in a user program. It may also occur when using the Tape Backup Utilities, if the cluster workstation was taken off-line during the backup operation.
9033	Bad tape parameters. The tape server requests contained invalid tape parameters. These include: zero-length buffers, requests to read or write zero records, etc.
9034	No Free Tape Control Blocks. This is an internal error and should not occur.
9035	Bad Tape File Name. The tape name supplied was invalid. Tape names have the form [TAPE], [TAPE]n, or [TAPE]+.

9036	Tape is in use. The tape drive is currently in use.
9037	Tape is write-protected. The write-enable ring was not inserted on the tape. Remove the tape and insert the write-enable ring.
9038	Illegal tape operation. An illegal command code was supplied to the TapeOperation request.
9039	Tape error outstanding. A read or write request was issued while a tape error was outstanding. Issue a TapeStatus request to clear the tape error before issuing another read or write request.
9040	Unknown tape mode. Tapes may only be opened for mode read ('mr') or mode write ('mw').
9041	Tape operation is outstanding. The tape cannot be released, or closed, until all tape operations have completed.
9042	Illegal tape buffer address. The address of the buffers are illegal for tape operations.
9043	Tape record is truncated. The buffer provided was not large enough for the tape record.
9044-9050	Reserved.
9051	Corrupt tape. The tape is not a valid backup tape or has been corrupted.
9052	Invalid tape position. The tape is neither at the load point or at the start of a tape file.
9053	Invalid tape reel. The tape is not part of the backup set currently being used. Verify that the correct tape is mounted.
9054	Invalid tape sequence. The tape reel is mounted out of the correct sequence for backup (that is, the program expected tape 2 of 3 and the third tape was mounted).
9055	Invalid tape configuration file. The tape configuration file is invalid. Use the Tape Configure command to create the configuration file.
9056	Missing tape configuration file. The tape configuration file cannot be found. Use the command Configure Tape File to create the tape configuration file.

9057	Bad tape record size. Tape records must be between 1024 and 16384 bytes for backup, and between 2 and 16384 bytes for tape bytestreams. Use the Configure Tape File command to create valid parameters.
9058	Missing Tape Controller. The Tape Controller is missing or improperly configured.
9059	Reserved.
9060	Tape Controller Failure. The Tape Controller is not correctly responding. Run the Tape Streamer Diagnostic to diagnose the failure.
9061-9098	Reserved.
9099	The tape server is already installed.

APPENDIX B
ACCESSING TAPE STREAMER FROM THE LANGUAGES

The Tape declarations and object module procedures must be added to certain files. Perform the following procedures.

Copy the files TapeBs.obj and TapeRq.obj from the Tape Streamer distribution diskette to your working directory. Copy the file CTOS.Lib located on the 4.0 Language Development package to the SYS directory. Invoke the Librarian command (which is also obtainable from the Language Development package):

```
Librarian
Library File        CTOS.Lib
[Files to add]      TapeBs.obj TapeRq.obj
[Modules to delete]
[Modules to extract]
[Cross-Reference File]
[Suppress confirmation]
```

Fill in the form as shown above and press the GO key. This procedure adds the Tape object modules to CTOS.Lib.

The following entries must be added to the file SamGen.asm. This file is located on the 4.0 Language Development Package. Copy file SamGen.asm to your working directory. It is divided into an introduction followed by three sections. The first is the device open section. Using the Editor, add the following entries:

```
%DeviceOpen([Tape],OpenByteStreamTape)
%DeviceOpen([Tape0],OpenByteStreamTape)
```

to the device open section.

The second section is the TagProc section. The entries shown are syntactically correct. However, the line should not look as it is represented (See NOTES). Add the following entries:

```
%tagProc(tagTapeRead,FillBufferTape,FlushBufIllegal,
ChkptNop,ReleaseByteStreamTape)
%tagProc(tagTapeWrite,FillBufIllegal,FlushBufferTape,
ChkptNop,ReleaseByteStreamTape)
```

to this section.

NOTES: Do not leave spaces after commas when keying in these entries. When keying in the Tagproc procedures, do not use a carriage return. That is, each entry is one continuous line. The screen will wrap around, but do not force the wrap; key in the entire entry without spaces.

Invoke the assembler command (which is also available on the 4.0 language development package):

```
Assemble
  File                        SamGen.Asm
  [Errors Only?]
  [GenOnly, NoGen, Gen]
  [Object file]
  [List file]
  [Error File]
  [List on pass 1?]
```

Fill in the form as shown above and press the **GO** key.

Add the object module SamGen.Obj, to <Sys>CTOS.Lib:

```
    Librarian
    Library File              CTOS.Lib
    [Files to add]            SamGen.Obj
    [Modules to delete]
    [Modules to extract]
    [Cross-Reference File]
    [Suppress confirmation] Yes
```

Fill in the form as shown above and press the **GO** key. This procedure adds SamGen.Obj to CTOS.Lib.

For further information about customizing SamGen.Asm please refer to the B 20 Systems Programmers Guide Part 1.

Linking applications using the new CTOS.Lib will include tape bytestreams.

Now the B 20 languages can access the MTSU. There are two methods of access available:

1) SAM Bytestream calls
2) Tape Server calls

The following is a brief description of these methods for each language.

Rename SamGen.Obj to Bas-SamGen.Obj. Invoke the Librarian

```
Librarian
Library File              Basic.Lib
[Files to add]            Bas-SamGen.Obj
[Modules to delete]
[Modules to extract]
[Cross-Reference File]
[Suppress confirmation] Yes
```

Fill in the form as shown above and press the **GO** key. This procedure adds Bas-SamGen.Obj to Basic.Lib.

Regarding both tape bytestreams and tape server calls, edit file Basgen.asm (this file is found in the BASIC distribution package). Change entry:

"sSambuffer dw 512"

to read:

"sSambuffer dw 1024"

Tape Bytestreams:

Assemble file Basgen.asm and answer YES to the question " Are you calling the Sequential Access Method?". To reconfigure Basic Interpreter, refer to Reconfiguring Basic in the B 20 Systems Basic Language Reference Manual for additional information to complete this process.

Tape Server:

Using the Editor, add the following entries to the file Basgen.asm in the "Add new entries" section:

```
%TableEntry(1,8,TAPEOPERATION)
%TableEntry(1,18,OPENTAPE)
%TableEntry(1,2,CLOSETAPE)
%TableEntry(1,18,READTAPERECORDS)
%TableEntry(1,18,WRITETAPERECORDS)
%TableEntry(1,2,PURGETAPEUSER)
%TableEntry(1,6,TAPESTATUS)
```

Assemble this new BasGen .Asm to produce a new Basgen.obj. To reconfigure Basic Interpreter, refer to Reconfiguring Basic in the B 20 Systems Basic Language Reference Manual for additional information to complete this process.

To write a Basic program which invokes the Tape ByteStreams or Tape Server calls, see "Calling non-Basic procedures" in the B 20 Systems Basic Language Reference Manual.

BASIC COMPILER (RELEASE LEVEL 4.0)

Tape Bytestreams:

Assemble file Basgen.asm (this file is found in the Basic distribution package) and answer YES to the question " Are you calling the Sequential Access Method?". Add the new BasGen.Obj to your program at compile time. For additional information refer to the B 20 Sytems Basic Compiler Reference Manual for linking this application.

Tape Server:

Using the Editor, add the following entries to the file Basgen.asm in the "Add new entries" section:

```
%TableEntry(1,8,TAPEOPERATION)
%TableEntry(1,18,OPENTAPE)
%TableEntry(1,2,CLOSETAPE)
%TableEntry(1,18,READTAPERECORDS
%TableEntry(1,18,WRITETAPERECORDS)
%TableEntry(1,2,PURGETAPEUSER)
%TableEntry(1,6,TAPESTATUS)
```

The entries will be assembled into the new Basgen.obj. Add the new BasGen.Obj to your program at compile time. For additional information refer to the B 20 Systems Basic Compiler Reference Manual for linking this application.

To write a Basic program which invokes the Tape ByteStreams or Tape Server calls, see "Calling non-Basic procedures" in the B 20 Systems Basic Compiler Reference Manual.

FORTRAN (RELEASE LEVEL 4.0)

Tape Bytestreams:

Assemble file Forgen.asm (this file is found in the Fortran distribution package) and answer YES to the question " Are you calling the Sequential Access Method?" Add the new ForGen.Obj to your program at compile time. For additional information refer to the B 20 Systems Fortran Reference Manual for linking this application.

Tape Server:

Add the following entries to the file Forgen.asm in the "Add new entries" section:

```
%TableEntry(OpenTape,  TAOPEN,7,r,w,w,w,w,w,r)
%TableEntry(CloseTape,  TACLOS,1,w)
%TableEntry(ReadTapeRecords,  TARDRC,6,w,w,r,w,r,r)
%TableEntry(WriteTapeRecords,  TAWRRC,6,w,w,r,w,r,r)
%TableEntry(TapeOperation,  TAOPER,4,w,w,w,w)
%TableEntry(PurgeTapeUser,  TAPUSR,1,w)
%TableEntry(TapeStatus  ,TASTAT,2,w,r)
```

The entries will be assembled into the new Forgen.obj. Add the new ForGen.Obj to your program at compile time. For additional information refer to the B 20 Sytems Fortran Reference Manual for linking this application.

To write a Fortran program which invokes the Tape ByteStreams or Tape Server calls, see "Calling non-Fortran procedures" in the B 20 Systems Fortran Reference Manual.

PASCAL (RELEASE LEVEL 4.0)

Tape Bytestreams and Tape Server:

Both groups of calls must be declared in the source program. For details, see the "Accessing B 20 Systems Operations from Pascal" in the B 20 Systems Pascal Reference Manual.

COBOL (RELEASE LEVELS 3.2 AND 4.0)

Tape Bytestreams (Release Levels 3.2 and 4.0)

Assemble the file COBOLgen.asm and answer YES to the question " Are you calling the Sequential Access Method?" Next reconfigure COBOL with this new COBOLGen.Obj. For more information, see Reconfiguring COBOL in the B 20 Systems COBOL Language Reference Manual.

Tape Server (Release Level 3.2)

Using the Editor, add the following entries to the file COBOLgen.Asm in the "Add new entries" section:

```
%TableEntry(0,w,CLOSETAPE,1,w)
%TableEntry(0,w,OPENTAPE,7,r,w,b,b,b,w,x)
%TableEntry(0,w,PURGETAPEUSER,1,w)
%TableEntry(0,w,READTAPERECORDS,6,w,w,r,w,x,x)
%TableEntry(0,w,TAPEOPERATION,4,w,b,b,w)
%TableEntry(0,w,TAPESTATUS,2,w,x)
%TableEntry(0,w,WRITETAPERECORDS,6,w,w,r,w,x,x)
```

Assemble this new COBOLgen.Asm to produce a new Cobolgen.obj. Reconfigure COBOL with this new Cobolgen.obj. For more information, see Reconfiguring COBOL in the B 20 Systems COBOL Language Reference Manual.

Tape Server (Release Level 4.0):

Assemble file COBOLgen.Asm and answer YES to the question "Are you calling Tape?". Reconfigure COBOL with this new COBOLGen.Obj. For more information, see Reconfiguring COBOL in the B 20 Systems COBOL Language Reference Manual.

APPENDIX C

SAMPLE PASCAL PROGRAMS

PROGRAM TAPEBSTREAM (INPUT, OUTPUT)

```
(*************************************************************)
(*                                                           *)
(* The sample program listed below will require modification *)
(* by the user so that it will conform to specific           *)
(* application design requirements and styling conventions.  *)
(*                                                           *)
(*************************************************************)

(* This program will write a file from disk to tape, then read *)
(* that file from tape to a disk file with any new name.       *)

Const
        modeRead = #6D72;
        modeWrite = #6D77;
        sSAMBuf = 1024;
        sUserBuf = 512;

Type
        adsWord = Ads of Word;
        ErcType = Integer;
        adsChar = Ads of Char;
        LString30 = LString(30);
        DataBuf = array[1..sUserBuf] of byte;
        UserBuf = DataBuf;

Var
        (* Define a bytestream work area for read and write *)
        (* action, must be 130 bytes long.                  *)

        RTBSWA, WTBSWA : Array [1..65] of Word;

        (* Define a buffer area for SAM for read and write  *)
        (* action.  The buffer size should be at least 1024 *)
        (* bytes long and word-aligned to ensure device     *)
        (* independence.                                    *)

        SAMBufR, SAMBufW : Array [1..512] of Word;

        i, erc, LenOfFile, BytesToRead, cbWDataRet,
        cbRDataRet: Integer;
        DiskFile, TapeFile, PrintFile : LString30;
```

```
        (* Define a user buffer area *)

        UserBuf1, UserBuf2 : DataBuf;

Function CloseBytestream (pBSWA: adsword): ErcType; Extern;

Function OpenBytestream (pBSWA: adsword; pbFileSpec: adsChar;
                         cbFileSpec: word; pbPassword: adsword;
                         cbPassword: integer; mode: word;
                         pBufferArea: adschar; sBufferArea: integer)
                         : ErcType; Extern;

Function ReadBsRecord (pBSWA: adsword; pBufferRet: adschar;
                         sBufferMax: integer; psDataRet: adsword)
                         : ErcType; Extern;

Function WriteBsRecord (pBSWA: adsword; pb: adschar; cb: integer;
                         pcbRet:adsword): ErcType; Extern;

Procedure ErrorExit (e: integer); Extern;

Procedure TapeBytestream (FileSpec1, FileSpec2: LString30;
                          Buf:DataBuf);

        (* This procedure opens two bytestreams: one for  *)
        (* reading and  one for writing; reads a file to  *)
        (* a user buffer area, and then writes from the   *)
        (* user buffer area to a new file; finally closes *)
        (* both bytestreams.                              *)

Var
        pbFileSpec1 : adsChar;
        cbFileSpec1 : word;
        pbFileSpec2 : adsChar;
        cbFileSpec2 : word;

Begin   (* TapeByteStream *)
  pbFileSpec1 := ads FileSpec1[1];
  cbFileSpec1 := Wrd(FileSpec1[0]);
  erc := OpenBytestream (ads RTBSWA, pbFileSpec1, cbFileSpec1,
            ads Null, 0, modeRead, ads SAMBufR, sSAMBuf);
  If (erc <> 0)
   Then
    Begin
    Writeln ('Error in filespec1 OpenBytestream, error code = ', erc);
    ErrorExit (erc);
    End;
```

```
  pbFileSpec2 := ads FileSpec2[1];
  cbFileSpec2 := Wrd(FileSpec2[0]);
  erc := OpenBytestream (ads WTBSWA, pbFileSpec2, cbFileSpec2,
             ads null, 0, modeWrite, ads SAMBufW, sSAMBuf);

  If erc <> 0
   Then
    Begin
    Writeln ('Error in filespec2 OpenBytestream, error code = ', erc);
    ErrorExit (erc);
    End;

  erc := ReadBsRecord (ads RTBSWA, ads Buf[1], sUserBuf,
             Ads cbRDataRet);
  Writeln ('cbRdataRet = ', cbRdataRet);

  If (erc <> 0) AND (erc <> 1)
   Then
   Begin
   Writeln ('Error in filespecl ReadBsrecord, error code = ', erc);
   ErrorExit(erc);
   End;

  erc := WriteBsRecord (ads WTBSWA, ads Buf[1], sUserBuf,
              Ads cbWDataRet);
  If erc <> 0
   Then
    Begin
    Writeln('Error in filespec2 WriteBsRecord, error code = ', erc);
    ErrorExit (erc);
    End;

  erc := CloseBytestream (ads RTBSWA);
  If erc <> 0
   Then
    Begin
    Writeln('Error in filespecl CloseBytestream, error code = ', erc);
    ErrorExit(erc);
    End;

  erc := CloseBytestream (ads WTBSWA);
  If erc <> 0
   Then
    Begin
    Writeln('Error in filespec2 CloseBytestream, error code = ', erc);
    ErrorExit(erc);
    End;

End;   (* TapeByteStream *)
```

```
Begin   (*   Main   *)
  Write ('The length of file : '); Readln (LenOfFile);
  BytesToRead := LenOfFile;
  Write('Disk file : '); Readln (DiskFile);
  Write('Enter tape file and press RETURN, tapefile: ');
  Readln(TapeFile);
  Write('Enter print file and press RETURN, printfile: ');
  Readln(PrintFile);
  TapeBytestream (DiskFile, TapeFile, UserBuf1);
  Writeln('first tapebytestream OK.');
  BytesToRead := LenOfFile;
  TapeBytestream (TapeFile, PrintFile, UserBuf2);

End.   (* Main *)
```

PROGRAM STATUS (INPUT, OUTPUT)

```
(*************************************************************)
(*                                                           *)
(* The sample program listed below will require modification *)
(* by the user so that it will conform to specific           *)
(* application design requirements and styling conventions.  *)
(*                                                           *)
(*************************************************************)

Const
        modeRead = #6d72;
        modeWrite = #6d77;
Type

        adsword = Ads of Word;
        adsChar = Ads of Char;
Var
        Th : word;
        TStatus, Status : integer;
        erc : integer;
        TapeName: LString(30);
        cbTapeName : integer;
        fStream, fDen, fClear : Byte;
        openmode : integer;
        modeOpen : word;

Function CloseTape (Thdle: word):integer; Extern;

Function OpenTape (pbTname:adsChar; cbTname: integer; fStreaming:
                   byte; fDensity:Byte; fCl: Byte; mode:word;
                   pThRet: adsword): integer; Extern;

Function TapeStatus (thdle: word; pStatusRet: adsword):Integer;
                     Extern;

Begin   (*  Main  *)

   Write('Enter tape name and press RETURN, TapeName: ');
   Readln (TapeName);
   cbTapeName := Ord(TapeName[0]);
   Write('Enter value for fStream and press RETURN, fStream= ');
   Readln (fStream);
   Write ('Enter value for fClear and press RETURN, fClear = ');
   Readln (fClear);
   fDen := 0;
   Write ('Enter the value for openmode (0 for read, 1 for write): ');
   Readln (openmode);
```

```
  If openmode = 0
    Then
      modeopen := modeRead
    Else
      modeOpen := modeWrite;

  erc := OpenTape (ads TapeName[1], cbTapeName, fStream, fHid,
                   fClear,modeopen, ads Th);
  If erc <> 0
    Then
      Writeln ('Error in OpenTape, erc = ', erc);

  erc := TapeStatus (Th, ads Status);
  If erc <> 0
    Then
      Writeln ('Error in TapeStatus, erc = ', erc)
    Else
      Begin
        TStatus := (Status DIV 16)*10 + (Status MOD 16);
        Writeln ('Tape Status = ', TStatus);
      End;

  erc := CloseTape (Th);
  If erc <> 0
    Then
      Writeln ('Error in CloseTape, erc = ', erc);

End.  (*  Main  *)
```

PROGRAM TAPEOP (INPUT, OUTPUT)

```
(*****************************************************************)
(*                                                               *)
(* The sample program listed below will require modification     *)
(* by the user so that it will conform to specific               *)
(* application design requirements and styling conventions.      *)
(*                                                               *)
(*****************************************************************)

(* This program accepts command and subop from keyboard for      *)
(* TapeOperation.                                                *)

Const
    modeRead = #6D72;
    modeWrite = #6D77;

Type
    adsChar= ads of Char;
    adsword= ads of word;

Var
    fStream, fDen, fClear, fFast: Byte;
    Command:Byte;
    SubOp, erc : Integer;
    Th : word;
    TapeName: LString(30);
    cbTapeName : Integer;
    openmode : integer;
    modeEntry : word;


Function CloseTape (Thdle: word): Integer; Extern;

Function OpenTape (pbTname: adsChar; cbTname:Integer; fStreaming:
            Byte; fDensity: Byte; fCl:Byte; mode: word;
            pThret: adsword):Integer; Extern;

Function TapeOperation (TapeHdle: word; flagFast: Byte;
            Commandcode:Byte; argument: Integer): Integer;
            Extern;


Begin   (* main *)

    Write('Enter tape name and press RETURN: ');
    Readln (TapeName);
    cbTapeName := Ord(TapeName[0]);
    fStream := 0;
    fDen := 0;
    fClear := 0;
    Write ('Enter value for opentape(0 for read or 1 for ');
    Write ('write ), OpenMode = ');
    Readln (OpenMode);
```

```
    If openmode = 0
     Then
       modeentry := modeRead
     Else
       modeentry := modeWrite;

    erc := OpenTape (ads TapeName[1], cbTapeName, fStream, fDen,
                fClear, modeentry, ads Th);

    If erc <> 0
    Then
     Writeln ('OpenTape failed, erc = ', erc);

    Write('Enter value for fFast and press RETURN, fFast = ');
    Readln (fFast);
    Write('Enter command code and press RETURN, Command = ');
    Readln(Command);
    Write('Enter the value for subOp press RETURN, subOp = ');
    Readln(subOp);
    erc := TapeOperation(Th,fFast,command, subOp);

    If erc <> 0
     Then
      Begin
       Writeln('Error in TapeOperation, erc =', erc);
       Writeln ('Command code = ', command, 'subOp= ', subOp);
      End;

    erc := CloseTape (Th);

    If erc <> 0
     Then
       Writeln ('CloseTape failed, erc =', erc)

End.   (*  main  *)
```

PROGRAM PURGE (INPUT, OUTPUT)

```
(*************************************************************)
(*                                                           *)
(* The sample program listed below will require modification *)
(* by the user so that it will conform to specific           *)
(* application design requirements and styling conventions.  *)
(*                                                           *)
(*************************************************************)

(* This program should be given SearchMark(4) and subOp(1) to      *)
(* TapeOperation.  The tape is expected to already have at least 6 *)
(* files consecutively.  After Search for 4 marks, PurgeTapeUser   *)
(* is called to stop the execution of the rest of the program.     *)

Const
        modeRead = #6D72;
        modeWrite = #6D77;

Type
        adsChar= ads of Char;
        adsword= ads of word;

Var
        i : integer;
        WA : word;
        fStream, fDen, fClear, fFast: Byte;
        Command:Byte;
        Th : word;
        TapeName: LString(30);
        cbTapeName, openmode, SubOp, erc  : Integer;
        modeEntry : word;

Function PurgeTapeUser (work:word): Integer; Extern;

Function CloseTape (Thdle: word): Integer; Extern;

Function OpenTape (pbTname: adsChar; cbTname:Integer; fStreaming:
                   Byte; fDensity: Byte; fCl:Byte; mode: word;
                   pThret: adsword):Integer; Extern;

Function TapeOperation (TapeHdle: word; flagFast: Byte;
                        Commandcode: Byte; argument: Integer):
                        Integer; Extern;

Begin   (*   Main   *)
   Write('Enter tape name and press RETURN: ');
   Readln (TapeName);
   cbTapeName := Ord(TapeName[0]);
   fStream := 0;
   fDen := 0;
   fClear := 0;
   Write ('Enter value for opentape (0 for read or 1 for write )');
   Write ('OpenMode = ');
```

```
  Readln (OpenMode);
  If openmode = 0
    Then
      modeentry := modeRead
    Else
      modeentry := modeWrite;

  erc := OpenTape (ads TapeName[1], cbTapeName, fStream, fDen,
                   fClear, modeentry, ads Th);
  If erc <> 0
    Then
      Writeln ('OpenTape failed, erc = ', erc);

  Write('Enter value for fFast and press RETURN, fFast = ');
  Readln (fFast);
  Write('Enter command code and press RETURN, Command = ');
  Readln(Command);
  Write('Enter the value for subOp press RETURN, subOp = ');
  Readln(subOp);

  i := 0;
  While i < 4 Do
    Begin
      erc := TapeOperation(Th,fFast,command, subOp);
      If erc <> 0
        Then
          Begin
            Writeln('Error in TapeOperation, erc =', erc);
            Writeln ('Command code = ', command, 'subOp= ', subOp);
          End;

      i := i + 1;
    End;

  Writeln ('The numer of records read, i = ', i);
  erc := PurgeTapeUser(WA);
  If (erc <> 0) AND (erc <> 4)
    Then
      Writeln ('Error in PurgeTapeUser, erc = ', erc);
  erc := TapeOperation(Th,fFast,command, subOp);
  If erc <> 0
    Then
      Begin
        Writeln('Error in TapeOperation, erc =', erc);
        Writeln ('Command code = ', command, 'subOp= ', subOp);
      End;

  erc := CloseTape (Th);
  If erc <> 0
    Then
      Writeln ('Error in CloseTape, erc = ', erc);


End.   (*   Main   *)
```

PROGRAM TAPERW (INPUT, OUTPUT)

```
(*************************************************************)
(*                                                           *)
(* The sample program listed below will require modification *)
(* by the user so that it will conform to specific           *)
(* application design requirements and styling conventions.  *)
(*                                                           *)
(*************************************************************)

(* This program will a file from the input tape, waits            *)
(* for the user to dismount the input tape and mount an output    *)
(* tape.  The output tape will contain a copy of the input file.  *)
(* The Tape Copy utility can be used to display the output file   *)
(* to [VID] . The record length is specified by the constant cbDa. *)

Const
        modeWrite = #6d77;
        modeRead = #6d72;
        cbDA = 132; (* Enter record size here. *)

Type
        adsword = Ads of Word;
        adsbyte = Ads of Byte;
        adsChar = Ads of Char;

Var
        erc : Integer;
        Th : Word;
        TapeName : LString(30);
        cbTapeName, nRecWrite, cRecWritten, cBytesWritten,nRecRead,
        cRecRead, cBytesRead : Integer;
        DA : Array [1.. cbDA] of Byte;
        fStream, fDen, fClear : Byte;

Function CloseTape (Thdle: word): integer; Extern;

Function OpenTape (pbTname: adsChar; cbTName: integer; fStreaming:
                   Byte; fDensity: Byte; fcl: Byte; mode: word;
                   pthRet: adsword): Integer; Extern;

Function ReadTapeRecords (Thdle: word; nRecords: Integer;
                          pbDataArea: adsbyte; cbDataArea: integer;
                          pcRecordsRet: adsword; pcbRet: adsword):
                          Integer; Extern;

Function WriteTapeRecords (THdle: word; nRecords: Integer;
                           pbDataArea: adsByte; cbDataArea: Integer;
                           pcRecordsRet: adsword; pcbRet: adsword):
                           Integer; Extern;

Begin
   Write ('Enter tape name and press RETURN, TapeName: ');
   Readln (TapeName);
```

```
  cbTapeName := Ord(TapeName[0]);
  Write ('Enter value for fStream, fStream = ');
  Readln (fStream);
  Write ('Enter value for fClear, fClear = ');
  Readln (fClear);
  erc := OpenTape (ads TapeName[1], cbTapeName, fStream, fDen,
                   fClear, modeRead, ads Th);

  If erc <> 0
    Then
      Writeln ('Error in OpenTape to read, erc = ', erc);

  Write ('Enter number of records to read, nRecRead = ');
  Readln (nRecRead);
  erc := ReadTapeRecords (Th, nRecRead, ads DA, cbDA,
                          ads cRecRead, ads cBytesRead);

  If erc <> 0
    Then
      Writeln ('Error in ReadTapeRecords, erc = ', erc);

  Writeln ('cRecread = ', cRecRead, 'cBytesRead = ', cBytesRead);
  erc := CloseTape (Th);

  If erc <> 0
    Then
      Writeln ('Error in CloseTape, erc = ', erc);

  Writeln;
  Writeln;
  Write ('Unload the input tape, mount the output tape, and ');
  Writeln ('press RETURN when ready.');
  Readln;
  erc := OpenTape (ads TapeName[1], cbTapeName, fStream, fDen,
                   fClear, modeWrite, ads Th);

  If erc <> 0
    Then
      Writeln ('Error in OpenTape, erc = ', erc);

  nRecWrite := nRecRead;
  erc := WriteTapeRecords (Th, nRecWrite, ads DA, cbDA,
                           ads cRecWritten, ads cBytesWritten);

  If erc <> 0
    Then
      Writeln ('Error in WriteTapeRecords, erc = ', erc);

  Write ('cRecWritten = ', cRecWritten, 'cBytesWritten = ');
  Writeln (cBytesWritten);
  erc := CloseTape (Th);
  If erc <> 0
    Then
      Writeln ('Error in CloseTape to write, erc = ', erc);
End.
```

INDEX

Documentation Evaluation Form

Title: B 22 Systems Tape Streamer Interface
Reference Manual Release Level 4.0

Form No: 1180080
Date: February, 1985

Burroughs Corporation is interested in receiving your comments and suggestions regarding this manual. Comments will be utilized in ensuing revisions to improve this manual.

Please check type of Comment/Suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error ☐ Other

Comments:

From:

Name _____

Title _____

Company _____

Address _____

_____

Phone Number _____ Date _____

Remove form and mail to:

Burroughs Corporation
Corporate Product
Information East
209 W. Lancaster Ave.
Paoli, PA 19301 U.S.A.