Using the Character-Special Tape Interface [6]

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

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

6.1 Using character-special tapes

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

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

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

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

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

6.2 Writing C applications

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

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

6.2.1 Opening files

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

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

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

6.2.2 Closing files

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

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

6.2.3 Using I/O

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

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

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

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

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

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

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

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

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

6.2.4 Using `ioctl`(2) requests

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

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

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

6.2.4.1 MTIOCACKERR call

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

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

6.2.4.2 MTIOCATTR call

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

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

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

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

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

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

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

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

6.2.4.3 MTIOCGET call

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

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

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

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

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

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

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

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

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

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

`MT_VALID_BLKNO` Specifies that `mt_blkno` is valid

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

6.2.4.4 `MTIOCTOP` call

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

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

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

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

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

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

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

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

6.2.4.4.1 MTWEOF

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

6.2.4.4.2 MTFSF and MTBSF

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

6.2.4.4.3 MTFSR and MTBSR

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

6.2.4.4.4 MTREW

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

6.2.4.4.5 MTOFFL

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

6.2.4.4.6 MTSYNC

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

6.2.4.4.7 MTRDLOG and MTCLRLOG

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

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

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

6.2.4.4.8 MTGABS and MTPABS

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

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

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

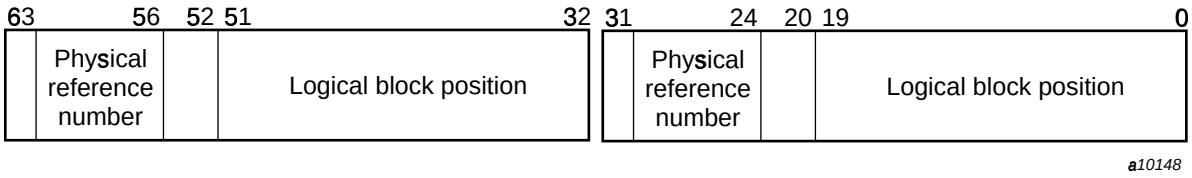

Figure 61. Block identifiers

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

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

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

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

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

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

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

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

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

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

6.2.4.4.9 MTGPOS

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

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

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

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

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

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

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

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

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

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

`tpc_time_code_a` and `tpc_time_code_b` specify the time code. The character-special tape interface does not support time-stamping. These fields do not contain valid values if they are created with the character-special tape interface.

A beginning-of-media (BOM) zone is created at the beginning of each partition when the tape is formatted. It consists of special physical blocks identifying the

logical beginning of a partition. `tpc_pos_bom` is set to 1 if the logical position is at the BOM. After a position to the BOM, a ER90 device is ready to process the first block of the first file section of the current partition.

The end-of-media warning (EMW) is the tenth physical block from the end of the partition. It provides a warning that the tape is positioned near the end of the partition. `tpc_pos_emw` is set to 1, for write operations, if the tape is positioned at or beyond the EMW of the current partition. It is set for read operations if the logical position is at or beyond the EMW of the current partition.

The early-end-of-media warning (EEW) is a tape location defined by the host. It provides a warning when the end of the partition approaches. `tpc_pos_eew` is set to 1, for write operations, if the tape is positioned at or beyond the EEW of the current partition. It is set for read operations if the logical position is at or beyond the EEW of the current partition.

The beginning-of-tape (BOT) is an area, located at the physical beginning of tape, used for tape loads and unloads. `tpc_pos_bot` is set to 1 if the tape is positioned at the BOT. There is no address associated with this area. The logical data block number, physical data block number, file section number, partition number, and absolute address fields are not valid when positioned at the BOT.

The end-of-recording (EOR) is recorded by the ER90 device after the last user data of the partition. `tpc_pos_eor` is set to 1 if the tape is positioned at the EOR.

The end-of-tape (EOT) is an area, located at the physical end of tape, used for tape loads and unloads. `tpc_pos_eot` is set to 1 if the tape is positioned at the EOT. There is no address associated with this area. The logical datablock number, physical datablock number, file section number, partition number, and absolute address fields are not valid when positioned at the EOT.

System zones are created on a tape volume when the volume is formatted. They provide an area of tape, other than the BOT and EOT zones, for loading and unloading a cassette. `tpc_pos_sysz` is set to 1 if the tape is positioned within a system zone. The system zone number is specified in `tpc_sysz_number`.

The end-of-media (EOM) is the end of the recording for a partition. `tpc_pos_eom` is set to 1 if the tape is positioned at the EOM of the current partition.

`tpc_valid_rem_part` is set to 1 if the `tpc_rem_partition` variable is valid. `tpc_rem_partition` specifies, in millions of bytes, the amount of data that can be recorded between the current position and the EOM.

`tpc_valid_rem_doubleframes` is set to 1 if the `tpc_rem_doubleframes` variable is valid. `tpc_rem_doubleframes` specifies the approximate number of double-frames (physical blocks) between the current position and the EOT.

6.2.4.4.10 MTSEEK

`MTSEEK` positions to a tape area specified in the `tpc_er90_seek` structure. This request is only valid for ER90 device files. `mt_arg` is a pointer to this structure. It is defined as follows:

```
struct tpc_er90_seek  {
  int  tpc_pos_flag;
  int  tpc_sysz_number;
}
```

`tpc_pos_flag` is a flag specifying the tape entity or area to position to. It is constructed from one of the following flags:

Flag	Description
`TPC_POS_EMW`	Positions to the EMW of the current partition
`TPC_POS_BOM`	Positions to the BOM of the current position
`TPC_LOAD`	Positions to a volume format information (VFI) zone
`TPC_POS_BOT`	Positions to the BOT
`TPC_INIT_POS`	Positions the tape to the BOM and initializes the volume
`TPC_POS_SYSZONE`	Positions to the system zone specified in `tpc_sysz_number`
`TPC_POS_EOT`	Positions the tape to the EOT
`TPC_POS_EOR`	Positions the tape to the EOR of the current partition
`TPC_PARK`	Positions the tape to the nearest system zone, in the BOT direction, and unthreads the tape

The `TPC_LOAD` request involves searching for and then reading the volume format information (VFI). This information is recorded when the cassette is formatted and consists of the format ID plus system zone and partition information. The operation is performed automatically when a cassette is loaded and should not have to be requested.

The TPC_INIT_POS request positions to BOM and then initializes the tape so that the tape is formatted during write operations. The tape is formatted with a NULL format ID, system zones, and one partition spanning the length of the tape. This request cannot be used on a cassette with an existing format that has a nonzero format ID.

The TPC_PARK request is used to minimize head wear. It positions to a system zone and then unthreads the tape from the helical scanner. Tape processing can resume at the current position without losing any buffered data and without issuing any additional requests.

For information on positioning with MTGPOS, see section Section 6.2.4.4.9, page 132.

6.2.4.4.11 MTEXTS

MTEXTS returns the extended status of a device for ER90 and block multiplexer device files.

For ER90 device files, it consists of the responses to commands: Report Addressee Status, Attribute, Operating Mode, and Report Position.

The Report Addressee Status Response describes the state of the ER90 device (ready/not ready or on-line/off-line), a description of the mounted volume, and the ER90 detailed status. The Attribute Response returns the operational characteristics of the ER90, for example, the data block size, burst size, early-end-of-media warning (EEW) location, and so on. The Operating Mode Response describes those attributes that have been defined only for as long as the tape is positioned within the current partition. The Report Position Response contains the current absolute track address, the remaining partition capacity, and other tape location information.

mt_arg is a pointer to the ctl_extsts structure. This structure is defined as follows:

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

```
}
```

To receive responses to all commands, `rep_addr`, `attributes`, `oper_mode`, and `report_pos` must be set to pointers to the memory into which the response packets are copied. To receive only select portions of the extended device status, the memory pointers of the response packets that are not desired must be set to `NULL`. For each command requested, the amount of memory allocated for the command must be set in the `len_rep_addr`, `len_attributes`, `len_oper_mode`, or `len_report_pos`. The length of each response packet is returned in these variables. Field device is not used for the character-special tape interface.

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

`MTEXTS` returns the sense information of a device. This information contains the device status, tape position, recoverable error counters, and other information. `mt_arg` is a pointer to the buffer into which sense information is read. `mt_size` specifies the size of the buffer receiving the sense information. The buffer size must be at least 64 bytes.

6.2.4.4.12 MTFMT

`MTFMT` formats a cassette for ER90 device files. Formatting records a volume identifier, creates partitions (logical volumes), and, if requested, creates system zones. `mt_arg` is a pointer to a structure defining this format. The structure is defined as follows:

```
struct tpc_format {
     uint  tpc_preformat     : 1,
           tpc_syszone       : 1,
           tpc_pack          : 1,
           tpc_extend        : 1,
           tpc_waste         : 1,
           tpc_verify_volume : 1,
           tpc_unused_0      : 10,
           tpc_fmtid         : 48;
     uint  tpc_count_a       : 16,
           tpc_count_b       : 16,
           tpc_sysz_spacing  : 32;
     uint  tpc_size_a        : 32,
           tpc_size_b        : 32;
     uint  tpc_old_fmtid     : 48,
```

```
            tpc_unused_1      : 16;
}
```

`tpc_preformat` specifies whether the tape should be preformatted. If set to 1, the volume is preformatted with the information provided in the `tpc_format` structure. If `tpc_preformat` is set to 0, the tape is formatted during write operations. Multiple partitions cannot be requested if the tape is formatted during write operations.

`tpc_syszone` specifies whether system zones are created on the tape. System zones are data-free areas on the tape that can be used to load and unload the cassette. If `tpc_syszone` is set to 1, the volume is formatted with system zones. Otherwise, no system zones are created. If a volume is formatted without system zones, the volume is positioned to the beginning-of-tape (BOT) or the end-of-tape (EOT) when it is unloaded. It could take up to 185 seconds to complete the unload. If the default system zone spacing is used, the unload time can be reduced to approximately 16 seconds for small cassettes, 21 seconds for medium cassettes, and 24 seconds for large cassettes.

`tpc_pack` is set to 1 to allow partitions to span system zones. This option must be specified if a single partition is requested or if no system zones are requested. `tpc_pack`, `tpc_extend`, and `tpc_waste` are mutually exclusive.

`tpc_extend` is set to 1 to request that the ER90 attempt to minimize the amount of system zone discontinuities in a partition. If the ER90 device determines that a partition should be created after a system zone, the previous partition is extended to the system zone dividing the two partitions. This option cannot be specified if a single partition is requested or if no system zones are requested. `tpc_pack`, `tpc_extend`, and `tpc_waste` are mutually exclusive.

`tpc_waste` is set to 1 to request that the ER90 attempt to minimize the number of system zone discontinuities within a partition. If the ER90 device determines that a partition should be created after a system zone, the previous partition is not extended to the system zone dividing the two partitions. Instead, the area between the previous partition and the system zone is wasted. This option cannot be specified if a single partition is requested or if no system zones are requested. `tpc_pack`, `tpc_extend`, and `tpc_waste` are mutually exclusive.

`tpc_verify_volume` is used to request volume verification. If set to 1, the value specified in `tpc_old_fmtid` is compared with the ID recorded on the volume to be formatted. If the volume IDs do not match, the request is terminated with the `ETPD_BAD_REQT` error code.

`tpc_fmtid` specifies the identifier to be recorded on the tape during the volume format. The format identifier must not be longer than 6 alphanumeric characters.

`tpc_count_a` and `tpc_count_b` specify the number of A partitions and the number of B partitions that should be formatted. The number of A partitions specified must be in the range 1 through 255; the size is specified with `size_a` field.

The A partitions are formatted on the volume until all partitions have been created or the end of the tape is detected. If tape remains after formatting the A partitions and no B partitions are requested, the tape is formatted with A partitions until the EOT is detected.

The number of B partitions specified must be in the range 0 through 255. B partitions are created following the last A partition. If one B partition is requested with a size of 0, the volume is formatted with one B partition spanning the remainder of the volume. If you specify more than one B partition, the volume is formatted with B partitions until all partitions are formatted or until the EOT is detected.

If the end of the volume is not detected after creating the B partitions, formatting continues, beginning again with A partitions.

`tpc_size_a` and `tpc_size_b` specify the size of the partitions. The size of the partition is specified in millions of bytes and must be in the range 0, 0xF0 through 0x1312D00 (240 through 20,000,000).

If the A partition size is 0, one partition is created spanning the length of the volume. Any size specified for the B partition is then not valid. If the A partition size is 0, one B partition is created spanning the length of the tape remaining after the A partitions.

Nonstandard system zone spacing can be requested with field `tpc_sysz_spacing`. `tpc_sysz_spacing` specifies the length, in double frames, between system zones. The length specified must be in the range 0x842 through 0xFFFFFF. If this field is set to 0, the default system zone spacing is used.

6.2.4.4.13 `MTGFMT`

`MTGFMT` returns a description of the cassette and volume format of the currently loaded tape for ER90 device files. The format is described in the `tpc_fmtdesc` structure, which is copied into the buffer pointed to by `mt_arg`. The structure is defined as follows:

```
struct tpc_fmtdesc   {
     int        tpc_fmtid;
     uint       tpc_cas_not_supported : 1,
                tpc_cas_loaded         : 1.
                tpc_cas_size           : 2,
                tpc_tape_thickness     : 2,
                tpc_tape_coercivity    : 2,
                tpc_not_wrt_protected  : 1,
                tpc_not_pre_striped    : 1,
                tpc_volume_loaded      : 1,
                tpc_preformat          : 1,
                tpc_syszone            : 1,
                tpc_pack               : 1,
                tpc_extend             : 1,
                tpc_waste              : 1,
                tpc_partition_table    : 1,
                tpc_non_std_sysz_spc  : 1,
                tpc_physical_blktype  : 1,
                tpc_count_a            : 16,
                tpc_count_b            : 16,
                tpc_unused             : 9;
      uint      tpc_size_a             : 32,
                tpc_size_b             : 32;
      uint      tpc_sysz_spacing       : 32,
                tpc_sysz_size          : 32;
      uint      tpc_last_part_number   : 32,
                tpc_last_part_size     : 32;
}
```

`tpc_fmtid` specifies the identifier recorded on the tape during the volume format.

`tpc_cas_not_supported` specifies whether the cassette configuration is supported. The tape thickness, tape coercivity, the write protection mechanism, and prestripe state are evaluated to determine if the cassette is supported. This field is set to 1 if the cassette is not supported.

`tpc_cas_loaded` is set to 1 if the cassette is loaded. A cassette is loaded when it is inserted into the ER90 device, the tape cassette hubs and servo capstan hubs are interlocked, and the tape is positioned over the longitudinal heads. If this bit is set to 0, all other fields in the response are invalid.

`tpc_cas_size` specifies one of the following for the cassette size:

Setting	Description
0	Small cassette
1	Medium cassette
2	Large cassette

`tpc_tape_thickness` specifies one of the following for the tape thickness:

Setting	Description
0	16 micrometer tape
1	13 micrometer tape
3	Cleaning cassette

`tpc_tape_coercivity` specifies one of the following for the tape coercivity:

Setting	Description
0	850 oersted tape (D1)
1	1500 oersted tape (D2)
3	Cleaning cassette

`tpc_not_wrt_protected` is set to 1 if the tape is not write protected.

`tpc_not_pre_striped` is set to 1 if the tape has not been prestriped. Prestriping prerecords the longitudinal servo track.

`tpc_volume_loaded` is set to 1 if the volume in the device has been loaded. A volume reaches the loaded state after the volume format information has been read. If this bit is set to 0, the remainder of the fields in structure `tpc_fmtdesc` are invalid.

`tpc_preformat` is set to 1 if the volume has been preformatted.

`tpc_syszone` is set to 1 if the volume was formatted with system zones.

On UNICOS systems, `tpc_pack` is set to 1 if the volume was formatted with the `-z` option of the `tpformat`(8) command. `tpc_extend` specifies is set to 1 if the volume was formatted with the `-e` option of the `tpformat`(8) command. `tpc_waste` specifies is set to 1 if the volume was formatted with the `-w` option of the `tpformat`(8) command.

`tpc_partition_table` is set to 1 if the partition table has been recorded on the volume.

`tpc_non_std_sysz_spc` is set to 1 if the volume was formatted with nonstandard system zone spacing.

`tpc_physical_blktype` specifies one of the following physical block types. A physical block is the smallest unit in which data can be recorded on tape.

Type	Description
0	Type 0 physical blocks
1	Type 1 physical blocks

`tpc_count_a` specifies the number of A partitions formatted on the cassette. `tpc_count_b` specifies the number of B partitions formatted on the cassette.

`tpc_size_a` specifies the size of the A partitions, in millions of bytes. A value of 0, indicates that the partition spans the length of the tape. `tpc_size_b` specifies the size of the B partitions, in millions of bytes. A value of 0 indicates that the B partition spans the length of the tape remaining after the A partitions.

`tpc_sysz_spacing` specifies the distance between the system zones. The distance is specified in double frames.

`tpc_sysz_size` specifies the size of the system zones, in double frames. The size is fixed per cassette size. If no system zones have been formatted, the size is 0.

`tpc_last_part_number` specifies the number of the last partition formatted on the volume.

`tpc_last_part_size` specifies the size, in million of bytes, of the last partition formatted on the volume. If the volume was not preformatted, this field will be 0.

6.2.4.4.14 MTVERIFY

`MTVERIFY` verifies the integrity of the data recorded on tape for ER90 device files. `mt_arg` is a pointer to a structure defining the extent to which the tape should be verified and where the verification should begin. The structure is defined as follows:

```
struct tpc_verify {
     uint       tpc_extent          : 4,
                tpc_position        : 1,
                tpc_unused_0        : 59;
     uint       tpc_valid_logdb     : 1,
```

```
                tpc_valid_absaddr     : 1,
                tpc_valid_partition   : 1,
                tpc_valid_filesec      : 1,
                tpc_valid_timecode     : 1,
                tpc_unused_1           : 11,
                tpc_logical_datablock : 48;
     uint       tpc_absolute_address  : 32,
                tpc_file_section       : 32;
     uint       tpc_partition_number   : 16,
                tpc_time_code          : 48;
}
```

`tpc_extent` specifies the extent to which the tape should be verified. It is one of the following flags:

Flag	Description
`TPC_VERIFY_FILESEC`	Verifies the integrity of the data within the specified file section
`TPC_VERIFY_PARTITION`	Verifies the integrity of the data within the specified partition

File verification leaves the tape positioned after the last data block of the file section. Partition verification leaves the tape positioned after the last data block of the last file section of the partition.

`tpc_position` is set to 1 to request that the tape be positioned to the absolute address specified before verifying the integrity of the recorded data.

For a description of the absolute address fields, see Section 6.2.4.4.8, page 130.

6.2.4.4.15 MTTRACE

`MTTRACE` reads the device trace for ER90 device files. `mt_arg` is a pointer to the buffer into which the device trace is read. The trace information is always 2,399,680 bytes in length.

The ER90 data buffer is used to transfer the trace information. This request will, therefore, destroy all user data in the device buffer.

6.2.4.4.16 MTMSG

MTMSG displays a message on a tape device. mt_arg is a pointer to a buffer containing the string to be displayed. mt_size specifies the length of the message. For ER90 devices, the length of the message is limited to 8 characters. For BMX devices, the length is limited to 16 characters.

For BMX devices, mt_count specifies the type of message display. This field must be set to one of the following flags:

Flag	Description
FMsgAcl	Specifies that a load request be sent to an automatic cartridge loader.
FMsgHigh	Specifies that the characters in bytes 8 through 15 of the message buffer be displayed. By default, the message in bytes 0 through 7 will be displayed.
FMsgBlink	Specifies that the message be displayed intermittently. The message will be displayed for 2 seconds at intervals of 0.5 seconds.
FMsgAlt	Specifies that the device alternate between displaying the characters in bytes 0 through 7 and the characters in bytes 8 through 15. Each message will be displayed for 2 seconds at intervals of 0.5 seconds.
FMsgUnload	Specifies that the message in bytes 0 through 7 be displayed until a cartridge is unloaded from the tape device. If no cartridge is loaded, the message will be displayed only briefly.
FMsgLoad	Specifies that the message in bytes 0 through 7 be displayed until the tape device is next loaded.
FMsgNone	Specifies that no message be displayed.
FMsgHighUntilLoad	Specifies that the message in bytes 8 through 15 be displayed until the device is next loaded.

6.3 Hardware error codes

When a request cannot complete because of an IOP or device-detected error, one of the following error codes is returned.

Error code	Description
ETPD_BAD_REQT	The contents or format of a request are incorrect, or the sequence of requests issued is incorrect.
ETPD_BLANK_TAPE	The command was terminated because it cannot be issued to a device with a blank tape loaded.
ETPD_BOT	The beginning of tape or beginning of partition was detected.
ETPD_DATA_ERROR	An unrecoverable data error occurred.
ETPD_DEV_HUNG	A response was not received from the tape device.
ETPD_DEVBUSY	The device is busy.
ETPD_DEVICE	A device error occurred.
ETPD_EOM	The end of media was detected.
ETPD_EOR	The end of recording was detected.
ETPD_FORMAT	The volume format is not supported.
ETPD_HPCONN	A HIPPI connection error occurred.
ETPD_HPDATA	A HIPPI parity or checksum error occurred.
ETPD_HPREQ	A HIPPI request error occurred.
ETPD_HPTRNS	A HIPPI transmission error occurred.
ETPD_IOPERR	An IOP error occurred.
ETPD_IPCONN	An IPI connection error occurred.
ETPD_LGPS	A logical position has not been established.
ETPD_MAX_IOREQT	I/O request exceeded maximum size allowed.
ETPD_MEDIA	The media is not supported.
ETPD_NO_CASSETTE	The cassette is not loaded.
ETPD_NOT_BOF	The tape is not positioned at the beginning-of-file.
EPTD_NOT_OPER	A hardware error occurred.
ETPD_NOT_READY	Device is not ready.
ETPD_POSACC_ERR	The position cannot be accessed.
ETPD_SHPI	A HIPPI controller error occurred.
ETPD_SYSTEM	A tape driver error occurred.
ETPD_TAPE_ADDR	An invalid tape address occurred.

ETPD_TAPE_ERROR	A problem with the tape media occurred.