

NAME

mtio - FreeBSD magtape interface

DESCRIPTION

The special files named */dev/[en]sa** refer to SCSI tape drives, which may be attached to the system. */dev/sa*.ctl* are control devices that can be used to issue ioctls to the SCSI tape driver to set parameters that are required to last beyond the unmounting of a tape.

The rewind devices automatically rewind when the last requested read, write or seek has finished, or the end of the tape has been reached. The letter 'n' is prepended to the name of the no-rewind devices. The letter 'e' is prepended to the name of the eject devices.

Tapes can be written with either fixed length records or variable length records. See *sa(4)* for more information. Two filemarks mark the end of a tape, and one filemark marks the end of a tape file. If the tape is not to be rewound it is positioned with the head in between the two tape marks, where the next write will over write the second end-of-file marker.

All of the magtape devices may be manipulated with the *mt(1)* command.

A number of *ioctl(2)* operations are available on raw magnetic tape. The following definitions are from *<sys/mtio.h>*:

```
#ifndef _SYS_MTIO_H_
#define _SYS_MTIO_H_

#ifndef _KERNEL
#include <sys/types.h>
#endif
#include <sys/ioccom.h>

/*
 * Structures and definitions for mag tape io control commands
 */

/* structure for MTIOCTOP - mag tape op command */
struct mtop {
    short    mt_op;           /* operations defined below */
    int32_t  mt_count;       /* how many of them */
};
```

```

/* operations */
#define MTWEOF      0      /* write an end-of-file record */
#define MTFSF      1      /* forward space file */
#define MTBSF      2      /* backward space file */
#define MTFSR      3      /* forward space record */
#define MTBSR      4      /* backward space record */
#define MTREW      5      /* rewind */
#define MTOFFL     6      /* rewind and put the drive offline */
#define MTNOP      7      /* no operation, sets status only */
#define MTCACHE    8      /* enable controller cache */
#define MTNOCACHE  9      /* disable controller cache */

#if defined(__FreeBSD__)
/* Set block size for device. If device is a variable size dev      */
/* a non zero parameter will change the device to a fixed block size */
/* device with block size set to that of the parameter passed in.   */
/* Resetting the block size to 0 will restore the device to a variable */
/* block size device. */

#define MTSETBSIZ    10

/* Set density values for device. Sets the value for the opened mode only. */

#define MTSETDNSTY   11

#define MTERASE      12    /* erase to EOM */
#define MTEOD        13    /* Space to EOM */
#define MTCOMP       14    /* select compression mode 0=off, 1=def */
#define MTRETENS     15    /* re-tension tape */
#define MTWSS        16    /* write setmark(s) */
#define MTFSS        17    /* forward space setmark */
#define MTBSS        18    /* backward space setmark */
#define MTLOAD       19    /* load tape in drive */
#define MTWEOFI      20    /* write an end-of-file record without waiting*/

#define MT_COMP_ENABLE           0xffffffff
#define MT_COMP_DISABLED        0xffffffe
#define MT_COMP_UNSUPP          0xffffffd

/*

```

```

* Values in mt_dsreg that say what the device is doing
*/
#define MTIO_DSREG_NIL      0      /* Unknown */
#define MTIO_DSREG_REST    1      /* Doing Nothing */
#define MTIO_DSREG_RBSY    2      /* Communicating with tape (but no motion) */
#define MTIO_DSREG_WR      20     /* Writing */
#define MTIO_DSREG_FMK     21     /* Writing Filemarks */
#define MTIO_DSREG_ZER     22     /* Erasing */
#define MTIO_DSREG_RD      30     /* Reading */
#define MTIO_DSREG_FWD     40     /* Spacing Forward */
#define MTIO_DSREG_REV     41     /* Spacing Reverse */
#define MTIO_DSREG_POS     42     /* Hardware Positioning (direction unknown) */
#define MTIO_DSREG_REW     43     /* Rewinding */
#define MTIO_DSREG_TEN     44     /* Retensioning */
#define MTIO_DSREG_UNL     45     /* Unloading */
#define MTIO_DSREG_LD      46     /* Loading */

#endif /* __FreeBSD__ */

/* structure for MTIOCGET - mag tape get status command */

struct mtget {
    short    mt_type; /* type of magtape device */
    /* the following two registers are grossly device dependent */
    short    mt_dsreg; /* “drive status” register */
    short    mt_erreg; /* “error” register */
    /* end device-dependent registers */
    /*
     * Note that the residual count, while maintained, may be
     * be nonsense because the size of the residual may (greatly)
     * exceed 32 K-bytes. Use the MTIOCERRSTAT ioctl to get a
     * more accurate count.
     */
    short    mt_resid; /* residual count */
#ifdef __FreeBSD__
    int32_t  mt_blksiz; /* presently operating blocksize */
    int32_t  mt_density; /* presently operating density */
    uint32_t mt_comp; /* presently operating compression */
    int32_t  mt_blksiz0; /* blocksize for mode 0 */
    int32_t  mt_blksiz1; /* blocksize for mode 1 */
#endif
};

```

```

int32_t mt_blksiz2; /* blocksize for mode 2 */
int32_t mt_blksiz3; /* blocksize for mode 3 */
int32_t mt_density0; /* density for mode 0 */
int32_t mt_density1; /* density for mode 1 */
int32_t mt_density2; /* density for mode 2 */
int32_t mt_density3; /* density for mode 3 */
/* the following are not yet implemented */
uint32_t mt_comp0; /* compression type for mode 0 */
uint32_t mt_comp1; /* compression type for mode 1 */
uint32_t mt_comp2; /* compression type for mode 2 */
uint32_t mt_comp3; /* compression type for mode 3 */
/* end not yet implemented */
#endif

int32_t mt_fileno; /* relative file number of current position */
int32_t mt_blkno; /* relative block number of current position */

};

/* structure for MTIOCERRSTAT - tape get error status command */
/* really only supported for SCSI tapes right now */
struct scsi_tape_errors {
    /*
     * These are latched from the last command that had a SCSI
     * Check Condition noted for these operations. The act
     * of issuing an MTIOCERRSTAT unlatches and clears them.
     */
    uint8_t io_sense[32]; /* Last Sense Data For Data I/O */
    int32_t io_resid; /* residual count from last Data I/O */
    uint8_t io_cdb[16]; /* Command that Caused the Last Data Sense */
    uint8_t ctl_sense[32]; /* Last Sense Data For Control I/O */
    int32_t ctl_resid; /* residual count from last Control I/O */
    uint8_t ctl_cdb[16]; /* Command that Caused the Last Control Sense */
    /*
     * These are the read and write cumulative error counters.
     * (how to reset cumulative error counters is not yet defined).
     * (not implemented as yet but space is being reserved for them)
     */
    struct {
        uint32_t retries; /* total # retries performed */
        uint32_t corrected; /* total # corrections performed */
        uint32_t processed; /* total # corrections successful */
    };
};

```

```

        uint32_t failures; /* total # corrections/retries failed */
        uint64_t nbytes; /* total # bytes processed */
    } wterr, rderr;
};

union mtterrstat {
    struct scsi_tape_errors scsi_errstat;
    char _reserved_padding[256];
};

struct mtrblim {
    uint32_t granularity;
    uint32_t min_block_length;
    uint32_t max_block_length;
};

typedef enum {
    MT_LOCATE_DEST_OBJECT      = 0x00,
    MT_LOCATE_DEST_FILE       = 0x01,
    MT_LOCATE_DEST_SET        = 0x02,
    MT_LOCATE_DEST_EOD        = 0x03
} mt_locate_dest_type;

typedef enum {
    MT_LOCATE_BAM_IMPLICIT     = 0x00,
    MT_LOCATE_BAM_EXPLICIT    = 0x01
} mt_locate_bam;

typedef enum {
    MT_LOCATE_FLAG_IMMED      = 0x01,
    MT_LOCATE_FLAG_CHANGE_PART = 0x02
} mt_locate_flags;

struct mtlocate {
    mt_locate_flags      flags;
    mt_locate_dest_type dest_type;
    mt_locate_bam        block_address_mode;
    int64_t               partition;
    uint64_t              logical_id;
    uint8_t               reserved[64];
};

```

```

};

typedef enum {
    MT_EXT_GET_NONE,
    MT_EXT_GET_OK,
    MT_EXT_GET_NEED_MORE_SPACE,
    MT_EXT_GET_ERROR
} mt_ext_get_status;

struct mtextget {
    uint32_t      alloc_len;
    char          *status_xml;
    uint32_t      fill_len;
    mt_ext_get_status status;
    char          error_str[128];
    uint8_t       reserved[64];
};

#define MT_EXT_GET_ROOT_NAME      "mtextget"
#define MT_DENSITY_ROOT_NAME     "mtdensity"
#define MT_MEDIA_DENSITY_NAME    "media_density"
#define MT_DENSITY_REPORT_NAME   "density_report"
#define MT_MEDIUM_TYPE_REPORT_NAME "medium_type_report"
#define MT_MEDIA_REPORT_NAME     "media_report"
#define MT_DENSITY_ENTRY_NAME    "density_entry"

#define MT_DENS_WRITE_OK         0x80
#define MT_DENS_DUP              0x40
#define MT_DENS_DEFLT           0x20

#define MT_PARAM_FIXED_STR_LEN  32
union mt_param_value {
    int64_t      value_signed;
    uint64_t     value_unsigned;
    char         *value_var_str;
    char         value_fixed_str[MT_PARAM_FIXED_STR_LEN];
    uint8_t      reserved[64];
};

```

```

typedef enum {
    MT_PARAM_SET_NONE,
    MT_PARAM_SET_SIGNED,
    MT_PARAM_SET_UNSIGNED,
    MT_PARAM_SET_VAR_STR,
    MT_PARAM_SET_FIXED_STR
} mt_param_set_type;

typedef enum {
    MT_PARAM_STATUS_NONE,
    MT_PARAM_STATUS_OK,
    MT_PARAM_STATUS_ERROR
} mt_param_set_status;

#define MT_PARAM_VALUE_NAME_LEN 64
struct mtparamset {
    char                value_name[MT_PARAM_VALUE_NAME_LEN];
    mt_param_set_type value_type;
    int                 value_len;
    union mt_param_value value;
    mt_param_set_status status;
    char                error_str[128];
};

#define MT_PARAM_ROOT_NAME      "mtparamget"
#define MT_PROTECTION_NAME     "protection"

/*
 * Set a list of parameters.
 */
struct mtsetlist {
    int num_params;
    int param_len;
    struct mtparamset *params;
};

/*
 * Constants for mt_type byte. These are the same
 * for controllers compatible with the types listed.
 */

```

```

#define MT_ISTS          0x01          /* TS-11 */
#define MT_ISHT          0x02          /* TM03 Massbus: TE16, TU45, TU77 */
#define MT_ISTM          0x03          /* TM11/TE10 Unibus */
#define MT_ISMT          0x04          /* TM78/TU78 Massbus */
#define MT_ISUT          0x05          /* SI TU-45 emulation on Unibus */
#define MT_ISCPC         0x06          /* SUN */
#define MT_ISAR          0x07          /* SUN */
#define MT_ISTMSCP       0x08          /* DEC TMSCP protocol (TU81, TK50) */
#define MT_ISCY          0x09          /* CCI Cipher */
#define MT_ISCT          0x0a          /* HP 1/4 tape */
#define MT_ISFHP0x0b    /* HP 7980 1/2 tape */
#define MT_ISEXABYTE    0x0c          /* Exabyte */
#define MT_ISEXA8200    0x0c          /* Exabyte EXB-8200 */
#define MT_ISEXA8500    0x0d          /* Exabyte EXB-8500 */
#define MT_ISVIPER1     0x0e          /* Archive Viper-150 */
#define MT_ISPYTHON     0x0f          /* Archive Python (DAT) */
#define MT_ISHPDAT      0x10          /* HP 35450A DAT drive */
#define MT_ISMFOUR      0x11          /* M4 Data 1/2 9track drive */
#define MT_ISTK50       0x12          /* DEC SCSI TK50 */
#define MT_ISMT02       0x13          /* Emulex MT02 SCSI tape controller */

```

/* mag tape io control commands */

```

#define MTIOCTOP        _IOW('m', 1, struct mtop) /* do a mag tape op */
#define MTIOCGET        _IOR('m', 2, struct mtget) /* get tape status */
/* these two do not appear to be used anywhere */
#define MTIOCIEOT       _IO('m', 3) /* ignore EOT error */
#define MTIOCEEOT       _IO('m', 4) /* enable EOT error */
/*

```

* When more SCSI-3 SSC (streaming device) devices are out there
 * that support the full 32 byte type 2 structure, we'll have to
 * rethink these ioctls to support all the entities they haul into
 * the picture (64 bit blocks, logical file record numbers, etc..).
 */

```

#define MTIOCRDSPOS     _IOR('m', 5, uint32_t) /* get logical blk addr */
#define MTIOCRDHPOS     _IOR('m', 6, uint32_t) /* get hardware blk addr */
#define MTIOCSLOCATE    _IOW('m', 5, uint32_t) /* seek to logical blk addr */
#define MTIOCHLOCATE    _IOW('m', 6, uint32_t) /* seek to hardware blk addr */
#define MTIOCERRSTAT    _IOR('m', 7, union mterrstat) /* get tape errors */
/*

```

* Set EOT model- argument is number of filemarks to end a tape with.

* Note that not all possible values will be accepted.

*/

```
#define MTIOCSETEOTMODEL    _IOW('m', 8, uint32_t)
/* Get current EOT model */
#define MTIOCGETEOTMODEL    _IOR('m', 8, uint32_t)
#define MTIOCRBLIM         _IOR('m', 9, struct mtrblim) /* get block limits */
#define MTIOCEXTLOCATE     _IOW('m', 10, struct mtlocate) /* seek to position */
#define MTIOCEXTGET        _IOWR('m', 11, struct mtextget) /* get tape status */
#define MTIOCPARAMGET      _IOWR('m', 12, struct mtextget) /* get tape params */
#define MTIOCPARAMSET      _IOWR('m', 13, struct mtparamset) /* set tape params */
#define MTIOCSETLIST       _IOWR('m', 14, struct mtsetlist) /* set N params */

#ifdef _KERNEL
#define DEFTAPE             "/dev/nsa0"
#endif

#endif /* !_SYS_MTIO_H_ */
```

FILES

*/dev/[en]sa**

SEE ALSO

mt(1), tar(1), sa(4)

HISTORY

The **mtio** manual appeared in 4.2BSD. An i386 version first appeared in FreeBSD 2.2.