NAME

cdio – CD-ROM control operations

SYNOPSIS

#include <sys/cdio.h>

DESCRIPTION

The set of **ioctl**(2) commands described below are used to perform audio and **CD-ROM** specific operations. Basic to these **cdio** ioctl requests are the definitions in **<sys/cdio.h>**.

Several **CD-ROM** specific commands can report addresses either in LBA (Logical Block Address) format or in MSF (Minute, Second, Frame) format. The **READ HEADER**, **READ SUBCHANNEL**, and **READ TABLE OF CONTENTS** commands have this feature.

LBA format represents the logical block address for the **CD-ROM** absolute address field or for the offset from the beginning of the current track expressed as a number of logical blocks in a **CD-ROM** track relative address field. MSF format represents the physical address written on **CD-ROM** discs, expressed as a sector count relative to either the beginning of the medium or the beginning of the current track.

IOCTLS

The following I/O controls do not have any additional data passed into or received from them.

CDROMSTART This **ioctl( )** spins up the disc and seeks to the last address requested.

CDROMSTOP This **ioctl( )** spins down the disc.

CDROMPAUSE This **ioctl( )** pauses the current audio play operation.

CDROMRESUME This **ioctl( )** resumes the paused audio play operation.

CDROMEJECT This **ioctl( )** ejects the caddy with the disc.

The following I/O controls require a pointer to the structure for that **ioctl( )**, with data being passed into the **ioctl( )**.

CDROMPLAYMSF This **ioctl( )** command requests the drive to output the audio signals at the specified starting address and continue the audio play until the specified ending address is detected. The address is in MSF format. The third argument of this **ioctl( )** call is a pointer to the type **struct cdrom_msf**.

```
/*
 * definition of play audio msf structure
 */
struct cdrom_msf {
        unsigned char   cdmsf_min0;      /* starting minute*/
        unsigned char   cdmsf_sec0;      /* starting second*/
        unsigned char   cdmsf_frame0;    /*starting frame*/
        unsigned char   cdmsf_min1;      /* ending minute */
        unsigned char   cdmsf_sec1;      /* ending second */
        unsigned char   cdmsf_frame1;    /* ending frame */
};
```

The **CDROMREADTOCENTRY** ioctl request may be used to obtain the start time for a track. An approximation of the finish time can be obtained by using the **CDROMREADTOCENTRY** ioctl request to retrieve the start time of the track following the current track.

The leadout track is the next consecutive track after the last audio track. Hence, the start time of the leadout track may be used as the effective finish time of the last audio track.

CDROMPLAYTRKIND
This **ioctl( )** command is similar to **CDROMPLAYMSF**. The starting and ending address is in track/index format. The third argument of the **ioctl( )** call is a pointer to the type **struct cdrom_ti**.

```
/*
 * definition of play audio track/index structure
 */
struct cdrom_ti {
        unsigned char   cdti_trk0;      /* starting track*/
        unsigned char   cdti_ind0;      /* starting index*/
        unsigned char   cdti_trk1;      /* ending track */
        unsigned char   cdti_ind1;      /* ending index */
};
```

CDROMVOLCTRL This **ioctl( )** command controls the audio output level. The SCSI command allows the control of up to four channels. The current implementation of the supported **CD-ROM** drive only uses channel 0 and channel 1. The valid values of volume control are between 0x00 and 0xFF, with a value of 0xFF indicating maximum volume. The third argument of the **ioctl( )** call is a pointer to **struct cdrom_volctrl** which contains the output volume values.

```
/*
 * definition of audio volume control structure
 */
struct cdrom_volctrl {
        unsigned char   channel0;
        unsigned char   channel1;
        unsigned char   channel2;
        unsigned char   channel3;
};
```

The following I/O controls take a pointer that will have data returned to the user program from the **CD-ROM** driver.

CDROMREADTOCHDR
This **ioctl( )** command returns the header of the table of contents (TOC). The header consists of the starting tracking number and the ending track number of the disc. These two numbers are returned through a pointer of **struct cdrom_tochdr**. While the disc can start at any number, all tracks between the first and last tracks are in

contiguous ascending order.

```
/*
 * definition of read toc header structure
 */
struct cdrom_tochdr {
        unsigned char   cdth_trk0;      /* starting track*/
        unsigned char   cdth_trk1;      /* ending track*/
};
```

CDROMREADTOCENTRY

This **ioctl( )** command returns the information of a specified track. The third argument of the function call is a pointer to the type **struct cdrom_tocentry**. The caller needs to supply the track number and the address format. This command will return a 4-bit **adr** field, a 4-bit **ctrl** field, the starting address in MSF format or LBA format, and the data mode if the track is a data track. The **ctrl** field specifies whether the track is data or audio.

```
/*
 * definition of read toc entry structure
 */
struct cdrom_tocentry {
        unsigned char   cdte_track;
        unsigned char   cdte_adr        :4;
        unsigned char   cdte_ctrl       :4;
        unsigned char   cdte_format;
        union {
                struct {
                        unsigned char   minute;
                        unsigned char   second;
                        unsigned char   frame;
                } msf;
                int     lba;
        } cdte_addr;
        unsigned char   cdte_datamode;
};
```

To get the information from the leadout track, the following value is appropriate for **cdte_track** the field:

CDROM_LEADOUT Leadout track

To get the information from the data track, the following value is appropriate for **cdte_ctrl** the field:

CDROM_DATA_TRACK Data track

The following values are appropriate for the cdte_adr field:

CDROM_LBA	**LBA** format
CDROM_MSF	**MSF** format

CDROMSUBCHNL This **ioctl( )** command reads the Q sub-channel data of the current block. The subchannel data includes track number, index number, absolute **CD-ROM** address, track relative **CD-ROM** address, control data and audio status. All information is returned through a pointer to **struct cdrom_subchnl**. The caller needs to supply the address format for the returned address.

```
struct cdrom_subchnl {
        unsigned char  cdsc_format;
        unsigned char  cdsc_audiostatus;
        unsigned char  cdsc_adr:       4;
        unsigned char  cdsc_ctrl:      4;
        unsigned char  cdsc_trk;
        unsigned char  cdsc_ind;
        union {
                struct {
                        unsigned char  minute;
                        unsigned char  second;
                        unsigned char  frame;
                } msf;
                int     lba;
        } cdsc_absaddr;
        union {
                struct {
                        unsigned char  minute;
                        unsigned char  second;
                        unsigned char  frame;
                } msf;
                int     lba;
        } cdsc_reladdr;
};
```

The following values are valid for the audio status field returned from **READ SUBCHANNEL** command:

CDROM_AUDIO_INVALID	Audio status not supported
CDROM_AUDIO_PLAY	Audio play operation in progress
CDROM_AUDIO_PAUSED	Audio play operation paused
CDROM_AUDIO_COMPLETED	Audio play successfully completed
CDROM_AUDIO_ERROR	Audio play stopped due to error
CDROM_AUDIO_NO_STATUS	No current audio status to return

CDROMREADOFFSET

This **ioctl()** command returns the absolute **CD-ROM** address of the first track in the last session of a Multi-Session **CD-ROM**. The third argument of the **ioctl()** call is a pointer to an **int**.

CDROMCDDA SPARC: This **ioctl()** command returns the **CD-DA** data or the subcode data. The third argument of the **ioctl()** call is a pointer to the type **struct cdrom_cdda**. In addition to allocating memory and supplying its address, the caller needs to supply the starting address of the data, the transfer length, and the subcode options. The caller also needs to issue the **CDROMREADTOCENTRY ioctl()** to find out which tracks contain **CD-DA** data before issuing this **ioctl()**.

```
/*
 * Definition of CD-DA structure
 */
struct cdrom_cdda {
        unsigned int    cdda_addr;
        unsigned int    cdda_length;
        caddr_t         cdda_data;
        unsigned char   cdda_subcode;
};
```

To get the subcode information related to **CD-DA** data, the following values are appropriate for the **cdda_subcode** field:

CDROM_DA_NO_SUBCODE	**CD-DA** data with no subcode
CDROM_DA_SUBQ	**CD-DA** data with sub Q code
CDROM_DA_ALL_SUBCODE	**CD-DA** data with all subcode
CDROM_DA_SUBCODE_ONLY	All subcode only

To allocate the memory related to **CD-DA** and/or subcode data, the following values are appropriate for each data block transferred:

CD-DA data with no subcode	2352 bytes
CD-DA data with sub Q code	2368 bytes
CD-DA data with all subcode	2448 bytes
All subcode only	96 bytes

CDROMCDXA SPARC: This **ioctl( )** command returns the **CD-ROM XA** (CD-ROM Extended Architecture) data according to **CD-ROM XA** format. The third argument of the **ioctl( )** call is a pointer to the type **struct cdrom_cdxa**. In addition to allocating memory and supplying its address, the caller needs to supply the starting address of the data, the transfer length, and the format. The caller also needs to issue the **CDROMREADTOCENTRY ioctl( )** to find out which tracks contain **CD-ROM XA** data before issuing this **ioctl( )**.

```
/*
 * Definition of CD-ROM XA structure
 */
struct cdrom_cdxa {
        unsigned int    cdxa_addr;
        unsigned int    cdxa_length;
        caddr_t         cdxa_data;
        unsigned char   cdxa_format;
};
```

To get the proper **CD-ROM XA** data, the following values are appropriate for the **cdxa_format** field:

CDROM_XA_DATA	**CD-ROM XA** data only
CDROM_XA_SECTOR_DATA	**CD-ROM XA** all sector data
CDROM_XA_DATA_W_ERROR	**CD-ROM XA** data with error flags data

To allocate the memory related to **CD-ROM XA** format, the following values are appropriate for each data block transferred:

CD-ROM XA data only	2048 bytes
CD-ROM XA all sector data	2352 bytes
CD-ROM XA data with error flags data	2646 bytes

CDROMSUBCODE SPARC: This **ioctl( )** command returns raw subcode data (subcodes P ˜ W are described in the "Red Book," see **SEE ALSO**) to the initiator while the target is playing audio. The third argument of the **ioctl( )** call is a pointer to the type **struct cdrom_subcode**. The caller needs to supply the transfer length and allocate memory for subcode data. The memory allocated should be a multiple of 96 bytes depending on the transfer length.

```
/*
 * Definition of subcode structure
 */
struct cdrom_subcode {
        unsigned int    cdsc_length;
        caddr_t         cdsc_addr;
};
```

The next group of I/O controls get and set various **CD-ROM** drive parameters.

CDROMGBLKMODE SPARC: This **ioctl( )** command returns the current block size used by the **CD-ROM** drive. The third argument of the **ioctl( )** call is a pointer to an integer.

CDROMSBLKMODE SPARC: This **ioctl( )** command requests the **CD-ROM** drive to change from the current block size to the requested block size. The third argument of the **ioctl( )** call is an integer which contains the requested block size.

This **ioctl( )** command operates in exclusive-use mode only. The caller must ensure that no other processes can operate on the same CD-ROM device before issuing this **ioctl( )**. **read**(2) behavior subsequent to this **ioctl( )** remains the same: the caller is still constrained to read the raw device on block boundaries and in block multiples.

To set the proper block size, the following values are appropriate:

CDROM_BLK_512	512 bytes
CDROM_BLK_1024	1024 bytes
CDROM_BLK_2048	2048 bytes
CDROM_BLK_2056	2056 bytes
CDROM_BLK_2336	2336 bytes
CDROM_BLK_2340	2340 bytes
CDROM_BLK_2352	2352 bytes
CDROM_BLK_2368	2368 bytes
CDROM_BLK_2448	2448 bytes
CDROM_BLK_2646	2646 bytes
CDROM_BLK_2647	2647 bytes

CDROMGDRVSPEED SPARC: This **ioctl( )** command returns the current **CD-ROM** drive speed. The third argument of the **ioctl( )** call is a pointer to an integer.

CDROMSDRVSPEED SPARC: This **ioctl( )** command requests the **CD-ROM** drive to change the current drive speed to the requested drive speed. This speed setting is only applicable when reading data areas. The third argument of the **ioctl( )** is an integer which contains the requested drive speed.

To set the **CD-ROM** drive to the proper speed, the following values are appropriate:

CDROM_NORMAL_SPEED	150k/second
CDROM_DOUBLE_SPEED	300k/second
CDROM_QUAD_SPEED	600k/second
CDROM_MAXIMUM_SPEED	300k/second (2x drive) 600k/second (4x drive)

Note that these numbers are only accurate when reading 2048 byte blocks. The **CD-ROM** drive will automatically switch to normal speed when playing audio tracks and will switch back to the speed setting when accessing data.

SEE ALSO

ioctl(2), **read**(2)

N. V. Phillips and Sony Corporation, *System Description Compact Disc Digital Audio*, ("Red Book").

N. V. Phillips and Sony Corporation, *System Description of Compact Disc Read Only Memory*, ("Yellow Book").

N. V. Phillips, Microsoft, and Sony Corporation, *System Description CD-ROM XA*, 1991.

Volume and File Structure of CD-ROM for Information Interchange, ISO 9660:1988(E).

SCSI-2 Standard, document X3T9.2/86-109

NOTES

The **CDROMCDDA**, **CDROMCDXA**, **CDROMSUBCODE**, **CDROMGDRVSPEED**, **CDROMSDRVSPEED** and some of the block sizes in **CDROMSBLKMODE** are designed for new Sun-supported **CD-ROM** drives and might not work on some of the older **CD-ROM** drives.

The interface to this device is preliminary and subject to change in future releases. You are encouraged to write your programs in a modular fashion so that you can easily incorporate future changes.