

# **Atapi\_Cd**

Georg Campana

Copyright © 1995 CD++

---

**COLLABORATORS**

	<i>TITLE :</i> Atapi_Cd		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Georg Campana	February 12, 2023	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>Atapi_Cd</b>	<b>1</b>
1.1	Atapi_Cd.doc	1
1.2	cd.device/CD_ADDCHANGEINT	2
1.3	cd.device/CD_ATTENUATE	3
1.4	cd.device/CD_CHANGENUM	4
1.5	cd.device/CD_CHANGESTATE	4
1.6	cd.device/CD_CONFIG	4
1.7	cd.device/CD_EJECT	6
1.8	cd.device/CD_GETGEOMETRY	6
1.9	cd.device/CD_INFO	7
1.10	cd.device/CD_MOTOR	8
1.11	cd.device/CD_PAUSE	8
1.12	cd.device/CD_PLAYLSN	9
1.13	cd.device/CD_PLAYMSF	10
1.14	cd.device/CD_PLAYTRACK	11
1.15	cd.device/CD_PROTSTATUS	12
1.16	cd.device/CD_QCODELSN	12
1.17	cd.device/CD_QCODEMSF	13
1.18	cd.device/CD_READ	14
1.19	cd.device/CD_READXL	15
1.20	cd.device/CD_REMCHANGEINT	16
1.21	CD SCSI_DIRECT	17
1.22	cd.device/CD_SEEK	18
1.23	cd.device/CD_TOCLSN	18
1.24	cd.device/CD_TOCMSF	20
1.25	CD <sup>32</sup> Compatibility	21
1.26	SCSI_II	22

---

# Chapter 1

## Atapi\_Cd

### 1.1 Atapi\_Cd.doc

~CD\_ADDCHANGEINT~

~CD\_ATTENUATE~

~CD\_CHANGENUM~

~CD\_CHANGESTATE~

~CD\_CONFIG~

~CD\_EJECT~

~CD\_GETGEOMETRY~

~CD\_INFO~

~CD\_MOTOR~

CD\_PAUSE

CD\_PLAYLSN

CD\_PLAYMSF

CD\_PLAYTRACK

~CD\_PROTSTATUS~

~CD\_QCODELSN~

~CD\_QCODEMSF~

~CD\_READ~

~CD\_READXL~

~CD\_REMCHANGEINT~

```

~CD_REMFRAMEINT~

~CD SCSI_DIRECT~

~CD_SEEK~

~CD_TOCLSN~

~CD_TOCMSF~

Compatibility

SCSI II Emulation

```

## 1.2 cd.device/CD\_ADDCHANGEINT

### NAME

CD\_ADDCHANGEINT -- add a disk change software interrupt handler.

### FUNCTION

With this command you can add an interrupt handler to the Atapi disk device that gets invoked whenever a disk insertion or removal occurs.

You must pass in a properly initialized Exec Interrupt structure and be prepared to deal with disk insertions/removals immediately. The interrupt is generated by the exec Cause function, so you must preserve A6 .

To set up the handler, an Interrupt structure must be initialized. This structure is supplied as the io\_Data to the CD\_ADDCHANGEINT command. The handler then gets linked into the handler chain and gets invoked whenever a disk change happens. You must remove the handler before you exit.

This command only returns when the handler is removed. That is, the device holds onto the IO request until the

```

CD_REMCHANGEINT
command

```

is executed with that same IO request. Hence, you must use SendIO() with this command.

### IO REQUEST INPUT

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_ADDCHANGEINT
io_Length      sizeof(struct Interrupt)
io_Data        pointer to Interrupt structure

```

### IO REQUEST RESULT

```

io_Error - 0 for success, or an error code as defined in
          <devices/Atapi_Cd.h>

```

### NOTE

Full compatible with the CD\$^3\$^2\$

SEE ALSO

```

        CD_REMCHANGEINT
        , <devices/Atapi_Cd.h>, <exec/interrupts.h>,
exec.library/Cause()

```

### 1.3 cd.device/CD\_ATTENUATE

NAME

CD\_ATTENUATE -- Attenuate CD audio volume (only immediately )

IO REQUEST

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_ATTENUATE
io_Data        NULL
io_Length      NULL ( not used now, will be the attenuate time
                  in future version of the device)
io_Offset      target volume level (0 - 0x7FFF) (-1 = status only)

```

RESULTS

```

io_Actual      last volume level (in future versions you can
                  monitoring the fade in the time )

```

FUNCTION

This command will set the CD audio volume to the value contained in io\_Offset. The range is 0 (silence) to 0x7FFF (full volume). If -1 is specified as the target, the attenuation will not be modified; the current attenuation value will be returned in io\_Actual.

In future versions io\_Length will contain the duration of the fade. In seconds, this is io\_Length divided by the current frame rate (75).

EXAMPLE

NOTES

The Atapi cd.device supports only the immediate attenuation . This command has no effect on Amiga audio volume, only CD audio. If the drive does not support volume attenuation, but does support mute, a value of under \$0800 should be considered mute, and equal to or above should be full volume. If chunky attenuation is supported, the drive should do the best it can. Even if only mute is supported, if gradual attenuation is requested, the device should still emulate the volume command based on the \$0800 boundary.

BUGS

SEE ALSO

CD\_INFO

## 1.4 cd.device/CD\_CHANGENUM

### NAME

CD\_CHANGENUM -- return the current value of the disk-change counter.

### FUNCTION

The command returns the current value of the disk-change counter which is incremented each time a disk is inserted or removed from the Atapi drive.

### IO REQUEST INPUT

io\_Device            preset by the call to OpenDevice()  
io\_Unit              preset by the call to OpenDevice()  
io\_Command          CD\_CHANGENUM

### IO REQUEST RESULT

io\_Error - 0 for success, or an error code as defined in  
          <devices/Atapi\_Cd.h>  
io\_Actual - if io\_Error is 0, this contains the current value of the  
          disk-change counter.

## 1.5 cd.device/CD\_CHANGESTATE

### NAME

CD\_CHANGESTATE -- check if a there is a valid disk in the cd-drive.

### FUNCTION

This command checks to see if there is a valid disk in a drive.

### IO REQUEST INPUT

io\_Device            preset by the call to OpenDevice()  
io\_Unit              preset by the call to OpenDevice()  
io\_Command          CD\_CHANGESTATE

### IO REQUEST RESULT

io\_Error - 0 for success, or an error code as defined in  
          <devices/Atapi\_Cd.h>  
io\_Actual - 0 means there is a disk while anything else indicates  
          there is no disk.

### NOTES

A valid disk is a disk with a readable table of contents ,  
but a pure Audio-Cd is considered a not valid disk ( with only  
Audio Tracks )

## 1.6 cd.device/CD\_CONFIG

---

## NAME

CD\_CONFIG -- Set the drive preferences

## IO REQUEST

io\_Device            preset by the call to OpenDevice()  
 io\_Unit             preset by the call to OpenDevice()  
 io\_Command          CD\_CONFIG  
 io\_Data             pointer to first entry of TagList  
 io\_Length           0

## RESULTS

io\_Error            0 for success, or an error code as defined in  
 <devices/Atapi\_Cd.h>

## FUNCTION

You can set one or more of the configuration items.  
 The configuration items are:

fix)	TAGCD_PLAYSPEED	Default: 75 (at present this remains
cd-drive)	TAGCD_READSPEED	75,150,300 (it depends on the
	TAGCD_READXLSPEED	" " " " "
	TAGCD_SECTORSIZE	Default: 2048
	TAGCD_XLECC	(not supported)
Diskchange)	TAGCD_EJECTRESET	(Is always 0 -> No reset at

The speed setting is described number of frames per second.  
 All CD-ROM drives are capable of the 75 frames/second rate.  
 The 2x drives are capable of 150 frames/second, and some even more  
 and more (4x and 6x).  
 To determine the maximum frame rate of the drive, use the

## CD\_INFO

command.

You should always make sure the drive is capable of the  
 Check to see if there was an error , if the asked speed is not ok .

There are three different types of CD-ROM sectors. Mode 1 sectors  
 (2048 bytes), mode 2 form 1 sectors (2048 bytes), and mode 2 form 2  
 sectors (2328 bytes). Normally, disks are encoded in Mode 1 format.  
 Mode 2 form 1 is basically the same as mode 1; however, the mode 2  
 form 2 sector format contains no CD-ROM error correction information  
 and is usable for MPEG streams .

In order to read information encoded in this sector format, the  
 drive's sector size must be configured to 2328 byte sectors.  
 (this option is at present not supported , to read this sector  
 you must use the SCSI\_Direct command with READ\_CD ) .

## EXAMPLE

## NOTES

## BUGS

TAG\_IGNORE, TAG\_MORE, and TAG\_SKIP do not work.  
Do not use these in the TagList.

## SEE ALSO

CD\_INFO

## 1.7 cd.device/CD\_EJECT

## NAME

CD\_EJECT -- Open or close the CD's drive tray

## IO REQUEST

io_Command	CD_EJECT
io_Data	NULL
io_Length	requested state of drive door (0 == close, 1 == open)
io_Offset	0

## RESULTS

io_Error	0 for success, or an error code as defined in <devices/Atapi_Cd.h>
io_Actual	previous state of drive door

## FUNCTION

This command causes the CD-ROM drive's tray to open or close.  
The previous state of the drive door is returned in io\_Actual.

## EXAMPLE

## NOTES

## BUGS

## SEE ALSO

## 1.8 cd.device/CD\_GETGEOMETRY

## NAME

CD\_GETGEOMETRY -- return the geometry of the drive.

## FUNCTION

This command returns the informations about the geometry of the drive.

The information is returned in the DriveGeometry structure  
pointed to by io\_Data.

## IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()

```

io_Command      CD_GETGEOMETRY
io_Data         pointer to a DriveGeometry structure
io_Length       sizeof(struct DriveGeometry)

```

#### IO REQUEST RESULT

```

io_Error - 0 for success, or an error code as defined in
          <devices/Atapi_Cd.h>
io_Actual - length of data transferred.

```

#### SEE ALSO

```

CD_GETNUMTRACKS, <devices/trackdisk.h>

```

## 1.9 cd.device/CD\_INFO

#### NAME

```

CD_INFO -- Return informations and status of the CD-Drive

```

#### IO REQUEST

```

io_Device       preset by the call to OpenDevice()
io_Unit         preset by the call to OpenDevice()
io_Command      CD_INFO
io_Data         pointer to CDInfo structure
io_Length       sizeof(struct CDInfo)

```

#### RESULTS

```

io_Error        0 for success, or an error code as defined in
                <devices/Atapi_Cd.h>
io_Actual       length of data transferred

```

#### FUNCTION

This command returns the current configuration and status of the Atapi drive .

#### EXAMPLE

```

struct CDInfo Info;

ior->io_Command = CD_INFO;           /* Retrieve drive info. */
ior->io_Data    = (APTR)Info;        /* Here's where we want it */
ior->io_Length  = sizeof(struct CDInfo); /* Return whole structure */
DoIO(ior);

if (!ior->io_Error) {                /* Command succeeded */

    if (Info.Status & CDSTS_PLAYING) printf("Audio is playing\n");
    else                             printf("Audio not playing\n");
}

```

#### NOTES

#### BUGS

#### SEE ALSO

<devices/Atapi\_Cd.h>

## 1.10 cd.device/CD\_MOTOR

### NAME

CD\_MOTOR -- control the state of the drive motor.

### FUNCTION

The command is present only for compatibilty with the CD<sup>3</sup>^2\$ but it does NOTHING because some Filesystems put off the motor after each read sequence .

### IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CD_MOTOR
io_Length	the requested state of the motor, 0 to turn the motor off, and 1 to turn the motor on.

### IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/Atapi_Cd.h>
io_Actual	- if io_Error is 0 this contains the previous state of the drive motor.

### NOTE

To put off really the CD-Motor use the SCSI\_Direct command

The Atapi device will also stop the motor after 10 mins of inactivity (if no kind off audio playing is in progress )

## 1.11 cd.device/CD\_PAUSE

### NAME

CD\_PAUSE -- Pause or unPause play command.

### IO REQUEST

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CD_PAUSE
io_Data	NULL
io_Length	pausemode : 1 = pause play; 0 = do not pause play;
io_Offset	0

### RESULTS

io_Actual	- if io_Error is 0, this contains the previous pause state.
-----------	---

### FUNCTION

This command will place the CD in, or take the CD out of pause mode. The desired pause state is placed in io\_Length. This command only

effects play commands. When the audio is playing and the pausemode is set, this command will immediately pause the audio output suspending the play command until the play is unpaused. When audio is not playing and the pausemode is set, this command will set the pause mode (having no immediate effect). When a play command is submitted, the laser will seek to the appropriate position and pause at that spot. The play command will be suspended until the play is unpaused (or the play is aborted).

EXAMPLE

NOTES

BUGS

SEE ALSO

## 1.12 cd.device/CD\_PLAYLSN

NAME

CD\_PLAYLSN -- Play a selected portion of CD audio (LSN form).

IO REQUEST

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CD_PLAYLSN
io_Data	NULL
io_Length	length of play
io_Offset	starting position

RESULTS

io_Error	0 for success, or an error code as defined in <devices/Atapi_Cd.h>
----------	--

FUNCTION

This command causes the drive to start playing CD audio from the specified position until the specified length has passed.

io\_Offset specifies the starting position. io\_Length contains the amount of time to play. All data is specified in LSN format.

A DoIO() will not return until the requested number of sectors have been played. A SendIO() will return as soon as the PLAY has been started. At this time other commands can be sent (like CD\_PAUSE). To stop a play before the specified length has been reached, use AbortIO().

EXAMPLE

```
/* Play two minutes, ten seconds of audio starting at 20 minutes, */
/* 58 seconds, and 10 frames. */

ior->io_Command = CD_PLAYLSN; /* Play CD audio */
ior->io_Offset = 94360; /* 20*(60*75) + 58*75 + 10 */
ior->io_Length = 9750; /* 02*(60*75) + 10*75 + 00 */
DoIO (ior);
```

NOTES

BUGS

SEE ALSO

```

    CD_PLAYTRACK
    ,
    CD_PAUSE
    ,
    CD_ATTENUATE

```

### 1.13 cd.device/CD\_PLAYMSF

NAME

CD\_PLAYMSF -- Play a selected portion of CD audio (MSF form).

IO REQUEST

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_PLAYMSF
io_Data        NULL
io_Length      length of play
io_Offset       starting position

```

RESULTS

```

io_Error       0 for success, or an error code as defined in
               <devices/Atapi_Cd.h>

```

FUNCTION

This command causes the drive to start playing CD audio from the specified position until the specified length has passed.

io\_Offset specifies the starting position. io\_Length contains the amount of time to play. All data is specified in MSF format.

A DoIO() will not return until the requested number of sectors have been played. A SendIO() will return as soon as the PLAY has been started. At this time other commands can be sent (like CD\_PAUSE). To stop a play before the specified length has been reached, use AbortIO().

EXAMPLE

```

/* Play two minutes, ten seconds of audio starting at 20 minutes, */
/* 58 seconds, and 10 frames.                                     */

ior->io_Command = CD_PLAYMSF; /* Play CD audio          */
ior->io_Offset  = 0x00143A0A; /* $14=20, $3A=58, $0A=10 */
ior->io_Length  = 0x00020A00; /* $02=02, $0A=10, $00=00 */
DoIO (ior);

```

NOTES

BUGS

SEE ALSO

```

    CD_PLAYTRACK
    ,
    CD_PAUSE
    ,
    CD_ATTENUATE

```

## 1.14 cd.device/CD\_PLAYTRACK

NAME

CD\_PLAYTRACK -- Play one or more tracks of CD audio.

IO REQUEST

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_PLAYTRACK
io_Data        NULL
io_Length      number of tracks to play
io_Offset      start playing at beginning of this track

```

RESULTS

```

io_Error       0 for success, or an error code as defined in
               <devices/Atapi_Cd.h>

```

FUNCTION

This command causes the drive to play the specified audio track(s). The command will return when the audio has completed.

io\_Offset specifies the track number (starting from 1).

io\_Length specifies the number of tracks to play (0 is invalid).

EXAMPLE

```

ior->io_Command = CD_PLAYTRACK;    /* Play audio tracks    */
ior->io_Offset  = STARTTRACK;      /* Start with this track */
ior->io_Length  = 3;               /* Play three tracks    */
DoIO(ior);

```

NOTES

PLAY commands are asynchronous with many other CD commands. Using a separate I/O request, other commands can be sent to the device that can change the behavior of the PLAY command.

BUGS

SEE ALSO

```

    CD_PLAYMSF
    ,
    CD_PLAYLSN

```

```

    ,
    CD_PAUSE
    ,
    CD_ATTENUATE

```

## 1.15 cd.device/CD\_PROTSTATUS

### NAME

CD\_PROTSTATUS -- return whether the current disk is write-protected.

### FUNCTION

This command is used to determine whether the current disk is write-protected.  
At present, this function always returns write-protected status.  
In future if CD-WO are made available this may change.

### IO REQUEST INPUT

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_PROTSTATUS

```

### IO REQUEST RESULT

```

io_Error - 0 for success, or an error code as defined in
           <devices/Atapi_Cd.h>
io_Actual - 0 means the disk is NOT write-protected, while any other
            value indicates it is.

```

## 1.16 cd.device/CD\_QCODELSN

### NAME

CD\_QCODELSN -- Report current disk position.

### IO REQUEST

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_QCODELSN
io_Data        pointer to QCode structure
io_Length      0 - MUST be zero (for future compatibility)

```

### RESULTS

```

io_Error      0 for success, or an error code as defined in
              <devices/Atapi_Cd.h>

```

### FUNCTION

This command reports current subcode Q channel time information. This command only returns data when CD Audio is playing (or paused). At any other time, an error is returned. The Q-Code packet consists of:

```

struct QCode {
    UBYTE      CtlAdr;          /* Data type / QCode type      */

```

```

    UBYTE      Track;          /* Track number                */
    UBYTE      Index;         /* Track subindex number      */
    UBYTE      Zero;          /* The "Zero" byte of Q-Code */
    union LSNMSF TrackPosition; /* Position from start of track */
    union LSNMSF DiskPosition; /* Position from start of disk  */
};

```

## EXAMPLE

```

struct QCode qcode;

ior->io_Command = CD_QCODELSN; /* Retrieve TOC information */
ior->io_Length  = 0;           /* MUST be zero             */
ior->io_Data    = (APTR)qcode; /* Here's where we want it */
DoIO (ior);

if (!ior->io_Error) {          /* Command succeeded        */

    printf("Current position is: %ld\n", qcode.DiskPosition.LSN);
}

```

## NOTES

This function may not return immediately. It may take several frames to pass by before a valid Q-Code packet can be returned. Use SendIO() and CheckIO() if response time is critical, and the information is not.

## BUGS

## SEE ALSO

<devices/Atapi\_Cd.h>

## 1.17 cd.device/CD\_QCODEMSF

## NAME

CD\_QCODEMSF -- Report current disk position.

## IO REQUEST

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_QCODEMSF
io_Data        pointer to QCode structure
io_Length      0 - MUST be zero (for future compatibility)

```

## RESULTS

```

io_Error       0 for success, or an error code as defined in
               <devices/Atapi_Cd.h>

```

## FUNCTION

This command reports current subcode Q channel time information. This command only returns data when CD Audio is playing (or paused). At any other time, an error is returned. The Q-Code packet consists of:

```

struct QCode {

```

---

```

    UBYTE      CtlAdr;          /* Data type / QCode type          */
    UBYTE      Track;          /* Track number                    */
    UBYTE      Index;         /* Track subindex number           */
    UBYTE      Zero;          /* The "Zero" byte of Q-Code packet */
    union LSNMSF TrackPosition; /* Position from start of track    */
    union LSNMSF DiskPosition; /* Position from start of disk     */
};

```

#### EXAMPLE

```

struct QCode qcode;

ior->io_Command = CD_QCODEMSF; /* Retrieve TOC information */
ior->io_Length  = 0;           /* MUST be zero             */
ior->io_Data    = (APTR)qcode; /* Here's where we want it */
DoIO (ior);

if (!ior->io_Error) {          /* Command succeeded        */

    printf("Current position is: %02d:%02d:%02d\n",
           qcode.DiskPosition.MSF.Minute,
           qcode.DiskPosition.MSF.Second,
           qcode.DiskPosition.MSF.Frame);
}

```

#### NOTES

This function may not return immediately. It may take several frames to pass by before a valid Q-Code packet can be returned. Use SendIO() and CheckIO() if response time is critical, and the information is not.

#### BUGS

#### SEE ALSO

<devices/Atapi\_Cd.h>

## 1.18 cd.device/CD\_READ

### NAME

CD\_READ -- read data from disk.

### FUNCTION

Reads data from the ATAPI CD into memory. Data may be accessed on WORD boundaries ( and not only sector multiplies as with normal disk devices). Data lengths can also be described in WORD amounts.

### IO REQUEST INPUT

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_READ
io_Data        pointer to the buffer where the data should be put
io_Length      number of bytes to read, must be a WORD multiple.
io_Offset      byte offset from the start of the disk describing
               where to read data from, must be a WORD multiple.

```

## IO REQUEST RESULT

io\_Error - 0 for success, or an error code as defined in  
           <devices/Atapi\_Cd.h>  
 io\_Actual - if io\_Error is 0, number of bytes actually transferred

## NOTES

If an error occurs the device will retry the read for 4 times .  
 Reads with sector bounded Lengths and Offsets are very fast .

## SEE ALSO

CD\_READXL

## 1.19 cd.device/CD\_READXL

## NAME

CD\_READXL -- Read from ATAPI DRIVE into mem following a transfer list.

## IO REQUEST

io\_Device            preset by the call to OpenDevice()  
 io\_Unit             preset by the call to OpenDevice()  
 io\_Command          CD\_READXL  
 io\_Data             pointer to the transfer list (must be struct List \*).  
 io\_Length           maximum transfer length (WORD multiple) or 0.  
 io\_Offset           byte offset from the start of the disk describing  
                       where to read data from, must be a WORD multiple.

## RESULTS

io\_Error            0 for success, or an error code as described in  
                       <devices/Atapi\_Cd.h>  
 io\_Actual           if io\_Error is 0, number of bytes actually transferred

## FUNCTION

This command starts reading data off the disk at the specified location and deposits it into memory according to the nodes in a transfer list. The pointer to the list of transfer nodes is placed in io\_Data. If you have a non-circular transfer list, simply set io\_Length to 0 (0 is special and means ignore io\_Length) -- your transfer will end when your transfer list has been exhausted. If you have a circular transfer list, the list will never end. In this case, the transfer will terminate when io\_Length bytes have been transferred.

The fields in the CDXL node structure are:

```
struct CDXL {
    struct MinNode Node;          /* double linkage          */
    char *Buffer;                /* data destination       */
    LONG Length;                 /* must be even # bytes   */
    LONG Actual;                 /* bytes transferred      */
    APTR IntData;                /* interrupt server data segment */
}
```

```

VOID          (*IntCode)(); /* interrupt server code entry */
};

```

The philosophy here is that you set up the buffers you want filled, create CDXL nodes describing the locations and sizes of these buffers, link all the nodes together in the order that you'd like (even make a circular list for animations), and execute the command. The data will be streamed into the appropriate buffers until the list has been exhausted, an entry with a Length of zero is encountered, `io_Length` bytes have been transferred (if `io_Length` is non-zero), or the command is aborted with `AbortIO()`.

If you fill in the `(*IntCode)()` field with a pointer to an interrupt routine, your routine will be called when the transfer for the node is complete. Your code will be called before the driver proceeds to the next node. The interrupt should follow the same rules as standard interrupts (see `AddIntServer` of `Exec` autodocs). Register A2 will contain a pointer to the node just completed. You may manipulate the list from within the interrupt. Your code must be brief (this is an interrupt). When returning from this interrupt, D0 should be cleared and an RTS instruction should be used to return.

Servers are called with the following register conventions:

```

D0 - scratch
D1 - scratch

A0 - scratch
A1 - server is_Data pointer (scratch)
A2 - pointer to CDXL node just completed

A5 - jump vector register (scratch)

all other registers must be preserved

```

#### EXAMPLE

#### NOTES

Try to make sure that small buffers are not overused. Each time a node is completed, an interrupt is generated. If you find that your computer is acting sluggish, or the `CD_READXL` command is aborting, you are probably generating too many interrupts. It is not efficient to have more than a few of these interrupts generated within a vertical blank.

#### BUGS

#### SEE ALSO

```

CMD_READ,
    CD_SEEK
, Autodocs - AddIntServer

```

## 1.20 cd.device/CD\_REMCHANGEINT

## NAME

CD\_REMCHANGEINT -- remove a disk change software interrupt handler.

## FUNCTION

This command removes a disk change software interrupt added by a previous use of

CD\_ADDCHANGEINT

.

## IO REQUEST INPUT

The same IO request used for

CD\_ADDCHANGEINT

.

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CD_REMCHANGEINT
io_Length	sizeof(struct Interrupt)
io_Data	pointer to Interrupt structure

## IO REQUEST RESULT

io\_Error - 0 for success, or an error code as defined in <devices/Atapi\_Cd.h>

## SEE ALSO

CD\_ADDCHANGEINT  
, <devices/Atapi\_Cd.h>

## 1.21 CD SCSI DIRECT

## NAME

CD SCSI\_DIRECT -- send a SCSI command sequence to the ATAPI Drive

## IO REQUEST

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CD SCSI_DIRECT
io_Data	pointer to a struct SCSI_Cmd
io_Length	sizeof(struct SCSI_Cmd)

## RESULTS

io\_Error 0 for success, or an error code as defined in <devices/Atapi\_Cd.h>

## FUNCTION

This command allows you to send directly a SCSI command to the Atapi CD-Drive and is full compatible with HD\_SCESICMD of the scsi.device .

For Version < 2.10 this is the only way to Play Audiotracks on the Drive .

Note: Play operation started with SCSI\_Direct are always asynchronous .  
Using CD\_PLAYLSN etc. you have to wait the end of the play operation or abort the command .

The drive appears with this command like a Scsi CD-Drive , with the exeption for some SCSI II commands .

Our device emulates some of these SCSI II commands , to have a better compatibility .

EXAMPLE

NOTES

This command is not present in the real cd.device of the CD<sup>3</sup> but it is included allowing you to use all the Filesystems for SCSI CD-Drives and so much other programs like players etc..

BUGS

SEE ALSO

<devices/Atapi\_Cd.h> <devices/scsidisk.h>

## 1.22 cd.device/CD\_SEEK

NAME

CD\_SEEK -- position laser at specified location.

FUNCTION

CD\_SEEK moves the laser to the approximate position specified. The io\_Offset field should be set to the offset to which the head is to be positioned.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CD_SEEK
io_Offset	position where head is to be moved (always LSN format)

IO REQUEST RESULT

io\_Error - 0 for success, or an error code as defined in <devices/Atapi\_Cd.h>

NOTES

Do not use this command , because there is no reason .

## 1.23 cd.device/CD\_TOCLSN

---

## NAME

CD\_TOCLSN -- Get table of contents information from CD (LSN form).

## IO REQUEST

io\_Device            preset by the call to OpenDevice()  
 io\_Unit             preset by the call to OpenDevice()  
 io\_Command          CD\_TOCLSN  
 io\_Data             pointer to buffer where TOC is to be stored  
 io\_Length           number of CDTOC entries to be fetched  
 io\_Offset           entry to begin at (entry 0 is summary information)

## RESULTS

io\_Error            0 for success, or an error code as defined in  
                     <devices/Atapi\_Cd.h>  
 io\_Actual           Actual number of entries copied

## FUNCTION

This command returns the table of contents of the disk currently in the drive. The table of contents consists of up to 100 entries. Entry zero is summary information describing the number of tracks and the total number of minutes on the disk. Entries 1 through N contain information about each individual track. All position information will be in LSN format.

The io\_Data field points to an array of CDTOC structures to receive the TOC data.

The io\_Length field specifies the total number of entries to be fetched. The array pointed to by io\_Data must be at least this many elements in size.

The io\_Offset field specifies the entry number at which to start copying TOC data into \*io\_Data.

Entry zero (the summary entry) contains the following:

```
struct TOCSummary {
    UBYTE            FirstTrack;     /* First track on disk (always 1) */
    UBYTE            LastTrack;      /* Last track on disk            */
    union LSNMSF LeadOut;           /* Beginning of lead-out track */
};
```

Track entries (entries 1 through number of tracks) contain:

```
struct TOCEntry {
    UBYTE            CtlAdr;         /* Q-Code info                   */
    UBYTE            Track;          /* Track number                   */
    union LSNMSF Position;         /* Start position of this track */
};
```

CDTOC is described as a union between these two structures:

```
union CDTOC {
```

---

```

struct TOCSummary Summary; /* First entry is summary info. */
struct TOCEntry Entry; /* Entries 1-N are track entries */
};

```

EXAMPLE

NOTES

BUGS

SEE ALSO

## 1.24 cd.device/CD\_TOCMSF

NAME

CD\_TOCMSF -- Return table of contents information from CD (MSF form).

IO REQUEST

```

io_Device      preset by the call to OpenDevice()
io_Unit        preset by the call to OpenDevice()
io_Command     CD_TOCMSF
io_Data        pointer to array where TOC is to be stored
io_Length      number of CDTOC entries to be fetched
io_Offset      entry to begin at (entry 0 is summary information)

```

RESULTS

```

io_Error       0 for success, or an error code as defined in
               <devices/Atapi_Cd.h>
io_Actual      Actual number of entries copied

```

FUNCTION

This command returns the table of contents of the disk currently in the drive. The table of contents consists of up to 100 entries. Entry zero is summary information describing the number of tracks and the total number of minutes on the disk. Entries 1 through N contain information about each individual track. All position information will be in MSF format.

The `io_Data` field points to an array of CDTOC structures to receive the TOC data.

The `io_Length` field specifies the total number of entries to be fetched. The array pointed to by `io_Data` must be at least this many elements in size.

The `io_Offset` field specifies the entry number at which to start copying TOC data into `*io_Data`.

Entry zero (the summary entry) contains the following:

```

struct TOCSummary {
    UBYTE      FirstTrack; /* First track on disk (always 1) */

```

```

    UBYTE          LastTrack;      /* Last track on disk          */
    union LSNMSF   LeadOut;        /* Beginning of lead-out track */
};

```

Track entries (entries 1 through number of tracks) contain:

```

struct TOEntry {

    UBYTE          CtlAdr;         /* Q-Code info                */
    UBYTE          Track;         /* Track number                */
    union LSNMSF   Position;      /* Start position of this track */
};

```

CDTOC is described as a union between these two structures:

```

union CDTOC {

    struct TOCSummary Summary;    /* First entry is summary info. */
    struct TOEntry   Entry;      /* Entries 1-N are track entries */
};

```

EXAMPLE

NOTES

BUGS

SEE ALSO

## 1.25 CD<sup>32</sup> Compatibility

ATAPI CD DEVICE

© 1995 , 1996 by Georg Campana ---> CD++

Compatibilty with the CD<sup>32</sup> :

Not Implemented commands of the original CD<sup>32</sup>:

```

CD_ADDFRAMEINT  \
                 I think i will never implement this commands
CD_REMFRAMEINT  /

```

CD\_SEARCH

