Technote DV22

CD-ROM Driver Calls

By Chris Brown, Neil Day, and Brian Bechtel May 1993
Revised by: Sérgio J. Henriques Jan. 1997
Updated by: Mark Cookson Sept. 1998
Updated by: Vinnie Moscaritolo March. 1999

Apple Worldwide Developer Technical Support

This Technical Note discusses the public interface to the Apple CD-ROM driver. This information supersedes the "Macintosh CD-ROM Device Driver" chapter of the *AppleCD SC Developer's Guide* available through APDA. If you're writing special purpose application software that needs to access the audio or data portions of a CD-ROM directly, this Note will be of interest to you.

Changes since Sept. 1998: Added Quiescence mode.

Changes since May 1994: Added power management, play list calls and an icon for CD-Extra discs, which is available starting in version 5.1.X of the Apple CD-ROM driver extension. Removed SCSI-specific calls and support for block sizes of 256 and 1024 bytes/block, which are not available starting in version 5.2.X of the Apple CD-ROM driver extension. Fixed the definition of the "CD Features" flags in `GetCDFeatures`. Corrected some minor errors.

Topics

- Status & Control calls for the Apple CD-ROM Driver.
- Differences between the various Apple CD-ROM drives.

This Technote describes version 4.0.X through version 5.2.X of the Apple CD-ROM driver. Previous versions of the Apple CD-ROM driver are documented in the *AppleCD SC Developer's Guide* from APDA.

Version 5.2.X History: Version 5.2 adds support for ATAPI CD-ROM drives.

Version 5.1.X History: Version 5.1 adds support for the AppleCD 600, support for multi-session CDs, support for an independent icon for CD-Extra, support for asynchronous calls, and support for playing audio track lists with 5 new calls: `SetTrackList`, `GetTrackList`, `GetTrackIndex`, `SetPlayMode`, and `GetPlayMode`.

Version 5.0.X History: Version 5.0 adds support for the AppleCD 300 Plus and three new calls: `DriverGestalt`, `ReturnDeviceIdent`, and `GetCDFeatures`. Version 5.0 is also the first version to support asynchronous I/O using SCSI Manager 4.3. The `ReadMCN` and `ReadISRC` calls did not return the correct information in version 4.0.X when using an AppleCD 300. Those two calls now work in version 5.0, but the format of the returned parameters has changed for all drives. Version 5.0.1 fixes the following problems: an empty tray can be ejected using the `Eject` control call; when third-party SCSI accelerator cards are installed, the driver now sees the CD-ROM drives, and the `WhoIsThere` call works correctly; and `DriverGestalt` will no longer modify `csParam[0]`.

Version 4.0.X History: Version 4.0 of the Apple CD-ROM extension was the first version to

support dual-speed operation and multiple-session Photo CDs on the AppleCD 300. Version 4.0 also supports single-session Photo CDs on all other Apple CD-ROM drives. Version 4.0 had problems with certain CD-ROMs that used encryption technology to restrict access to files. When the customer was given a decryption key to gain access to a file, the file would not always appear in the Finder. Version 4.0.1 was produced to fix this problem. However, this version was tested only on the Macintosh IIvx and Performa 600. Version 4.0.2 was tested on all Macintosh computers, and a few changes were made to ensure compatibility across the Macintosh product line. Version 4.0.2 corrected problems when sending audio commands using block addressing only. Audio calls using minute-second-frame addressing continued to work correctly.

Audio compact discs and CD-ROM discs conform to two standards, called the Red Book and the Yellow Book. The Red book specifies audio standards; the Yellow book specifies additional standards for CD-ROM. "Red Book" is the common name of the "Compact Disc Digital Audio Standard" standard, CEI IEC 908. When a disc conforms to the Red Book standard, it will usually have "digital audio" printed below the "disc" logo. Most music CDs conform to this standard. "Yellow Book" is the standard for CD-ROM, ISO 10149. When a disc conforms to the Yellow Book, the words "data storage" usually are printed beneath the "disc" logo.

There are several other standards usually associated with CD-ROM technology: the "Green Book" standard defines CD-I (Compact Disc Interactive); "Orange Book" defines write-once compact discs; and "Blue Book" defines CD-Extra (Compact Disc Digital Audio plus Data).

You can get the Red Book and Yellow Book from:

ANSI
Attn: Sales
1430 Broadway
New York, NY 10018
(212) 642-4900

Red Book: CEI IEC 908
Yellow Book: ISO 10149:1989

You can get the Green Book from:

American CD-I Association
11111 Santa Monica, Suite 750
Los Angeles, CA 90025
(310) 444-6619

The other standards are available only to Sony/Philips licensees. Contact Sony or Philips for licensing details.

Beginning with version 4.0 of the Apple CD-ROM software, several new `Gestalt` selectors are installed by Foreign File Access and various file system translators. Each selector is a version number, as defined in *Inside Macintosh volume VI*, page VI-3-34. The list includes:

Apple Photo Access	`'kpcd'`
Foreign File Access	`'ufox'`
High Sierra File Access	`'hscd'`
ISO 9660 File Access, version 1.0 through 4.0	`'hscd'`
ISO 9660 File Access, version 5.0	`'iscd'`
Audio CD Access	`'aucd'`

To call the Macintosh driver, you need to understand the possible addressing formats for audio blocks on a CD. The three addressing formats are:

1. logical block
2. absolute minute, second, and frame
3. track number

An audio CD contains up to 74 minutes of sound. (By decreasing the inter-track gap and other tricks, you can put more than 74 minutes of sound on a CD. Doing this is only marginally within the standard. Such CDs may not be playable on all drives.) These minutes are divided into 60 seconds. Each second contains 75 frames. There is a two-second lead-in area at the beginning of the disc that contains table of contents information about the tracks on that CD. You can have a maximum of 99 tracks on a CD.

You can address sound down to the frame level, that is, down to the 1/75th of a second. You can address sound by specifying an absolute block number (e.g., start playing at absolute block 1,234,567 from the start of the disc), or by absolute minute, second, and frame number (e.g., start playing at minute 42, second 30, frame 15 on the disc), or by logical track number (e.g., start playing at track 2).

The driver requires that the logical track number or absolute minute-second-frame numbers be specified in BCD (e.g., the decimal number 12 is stored as hex `0x0012`, not as `0x000C`).

Optical Positioning Type

Many calls require an optical pick-up positioning type. This is used to specify what kind of address is being used in other parameters of a given call. There are four different optical pick-up positioning types:

Type	Description
`0x0000`	32-bit Logical Block Address
`0x0001`	An AMIN-ASEC-AFRAME descriptor that gives the running time from the beginning of the disc (in BCD)
`0x0002`	A track number (in BCD)
`0x0003`	An index into the driver's track play list

Track Control Field

Many calls return a control field. This field describes the format of the current track, and has the following values:

Bits 3210	Function

00x0	2 audio channels without pre-emphasis
00x1	2 audio channels with pre-emphasis
10x0	4 audio channels without pre-emphasis
10x1	4 audio channels with pre-emphasis
01x0	data track
01x1	reserved
11xx	reserved
xx0x	digital copy prohibited
xx1x	digital copy permitted

Audio Play Mode

Many calls require a play mode. This field describes how to play the audio track, and this field has the following values:

Bits 3210	Effect
0000	Muting on (no audio)
0001	Right channel through right channel only
0010	Left channel through right channel only
0011	Left and right channel through right channel only *
0100	Right channel through left channel only
0101	Right channel through left and right channel
0110	Right channel through left channel, left channel through right channel
0111	Right channel through left channel, left and right channel through right channel *
1000	Left channel through left channel only
1001	Left channel through left channel, right channel through right channel (stereo)
1010	Left channel through left and right channel
1011	Left channel through left channel, left and right channel through right channel *
1100	Left and right channel through left channel *
1101	Left and right channel through left channel, right channel through right channel *
1110	Left and right channel through left channel, left channel through right channel *
1111	Left and right channel through left channel, left and

1111	Left and right channel through left channel; left and right channel through right channel (monaural) *

*Not available on AppleCD 300 Plus

In the absolute minute-second-frame address, the ranges are as follows:

Minutes	00 to 99 (effectively, from 00 to 75)
Seconds	00 to 59
Frames	00 to 74

Making a High-Level Driver Call

The Apple CD-ROM driver does not conform to the normal design of a driver. In particular, the Apple CD-ROM driver returns some additional status information in some of its control calls, and expects to receive some additional control information in some of its status calls. The high-level control and status calls weren't designed with this in mind. The glue for control does not fill in the `csParamPtr` field with the results of any changes due to a control call, and the glue for status does not use the `csParam` field passed in as part of the call it creates on the stack. Do not use high-level control and status calls to access the Apple CD-ROM driver; use only the `PBControl` and `PBStatus` calls.

Making a Low-Level Driver Call

The low-level calls require that you pass in two parameters: a pointer to a parameter block, and a flag indicating synchronous or asynchronous completion. You must fill in the parameter block yourself, and may find it easier to define your own custom parameter block variants for the various calls. Here's some example MPW C code to get audio status:

```
#include <Devices.h>

short gDrvRefNum;               // set up somewhere else

// this is a custom version of the CntrlParam parameter
// block defined on page II-181 and II-183 or in
// Devices.h

#if PRAGMA_STRUCT_ALIGN
        #pragma options align=mac68k
#endif
typedef struct {
        QElemPtr qLink;
        int qType;
        int ioTrap;
        Ptr ioCmdAddr;
        ProcPtr ioCompletion;
        OsErr ioResult;
        StringPtr ioNamePtr;
        int ioVRefNum;
        int ioRefNum;
        int csCode;
        // Everything below this replaces: short csParam[11]
        unsigned char status;
        unsigned char play;
        unsigned char control;
        unsigned char minute;
        unsigned char second;
```

```
        unsigned char frame;
        char unused[16]; // for the rest
} CDCntrlParam;
#if PRAGMA_STRUCT_ALIGN
        #pragma options align=reset
#endif

// GetAudioStuff
//      requires one input, the driver reference number that we
//      got by calling PBOpen().
//      fills in six parameters with appropriate information
//      from the driver call.
//      We're assuming that only one CD player is attached.
//      To generalize this code, fill in the ioVRefNum dynamically.

OSErr GetAudioStuff(status, play, control, minute, second, frame)
unsigned char *status, *play, *control, *minute, *second, *frame;
{
        OSErr result;
        CDCntrlParam myPB;
        myPB.ioCompletion = 0;
        myPB.ioVRefNum = 1; // assuming only 1 volume
        myPB.ioRefNum = gDrvRefNum; // determined elsewhere
        myPB.csCode = 107;
        result = PBControl(&myPB, false);
        if (result == noErr)
        {
        *status = myPB.status;
        *play = myPB.play;
        *control = myPB.control;
        *minute = myPB.minute;
        *second = myPB.second;
        *frame = myPB.frame;
        }
        return(result);
}
```

Driver Summary

Name

.AppleCD

Conventions

- Each call description will follow this format:
    - **Function Name**

      A detailed description of the device driver function is given here.

      Input Parameters:

      Relevant `csCode` and `csParam[]` requirements are listed here.

      Output Produced:

      Any returned data is listed here.

Any returned data is listed here.

Status Return Codes:

A list of possible status return codes this function can return are listed here.

- Reserved fields are assumed to be filled with zeros, to ensure compatibility with future releases of the driver.
- Several status return codes are generic to any driver call and will not be repeated within each call description. Suggested possible meanings for these error codes are given in *Inside Macintosh*. The generic status return codes are:

```
controlErr
badUnitErr
notOpenErr
ioErr
wPrErr
paramErr
offLinErr
```

- All control calls not listed below will return `controlErr`.

Supported Control Calls

csCode	Function Name	Description
Standard Driver Calls		
1	`KillIO`	Interrupt all asynchronous driver operations.
5	`VerifyTheDisc`	Verify the data on the disc.
6	`FormatTheDisc`	Format the disc.
7	`EjectTheDisc`	Eject disc from drive.
21	`GetDriveIcon`	Return `ICN#` information for the drive hardware.
22	`GetMediaIcon`	Return `ICN#` information for the drive media.
23	`DriveInfo`	Return drive characteristics information.
65	`accRun`	Driver-specific `needTime` code.
Special CD-ROM Control Calls		
70	`SetPowerMode`	Allow users to set the power mode of a CD-ROM drive.
76	`ModifyPostEvent`	Enable/disable `PostEvent()`s for non-HFS discs.
79	`SetBlockSize`	Modify block size of device.
80	`SetUserEject`	Enable/disable the button on the front of the CD-ROM device.
81	`SetPollFreq`	Modify period of the `VBL/dNeedTime` task execution.
1079	`Quiescence`	Enable/Disable the CD driver Quiescence mode.
Audio track Control Calls		
100	`ReadTOC`	Return a disc's Table of Contents (TOC) information.
101	`ReadTheQSubcode`	Return Q-Subcode information.
102	`ReadHeader`	Return 4-byte header information for a specified block.
103	`AudioTrackSearch`	Search disc for a specified track.
104	`AudioPlay`	Cause drive to play a given range of audio.

105	AudioPause	Cause drive to enter/exit the Hold Track State.
106	AudioStop	Specify an address to cause the CD-ROM to stop.
107	AudioStatus	Request the play status information from a drive.
108	AudioScan	Cause drive to fast-forward/reverse.
109	AudioControl	Set volume level for drive (not available on AppleCD SC).
110	ReadMCN	Return Media Catalog Number (not available on AppleCD SC).
111	ReadISRC	Return International Standard Recording Code (not available on AppleCD SC).
112	ReadAudioVolume	Return Audio Volume Control Data (not available on AppleCD SC).
113	GetSpindleSpeed	Return the current spindle speed.
114	SetSpindleSpeed	Set the spindle speed.
115	ReadAudio	Return digital audio data (not supported by AppleCD SC, SC Plus/AppleCD 150).
116	ReadAllSubcodes	Return subcodes while playing audio (not supported by AppleCD SC, AppleCD SC Plus/AppleCD 150).

Audio track list Control Calls

122	SetTrackList	Set the audio tracks playlist.
123	GetTrackList	Get the current audio tracks playlist.
124	GetTrackIndex	Get the current audio tracks playlist index.
125	SetPlayMode	Set the audio tracks play mode (including repeat).
126	GetPlayMode	Get the current audio tracks play mode (including repeat).

Special System Control Calls

-1	GoodBye	Executes necessary cleanup before shutdown.

Supported Status Calls

csCode	Function Name	Description
8	DriveStatus	Provides drive/disc information about a specified device ID.
43	DriverGestalt	Returns various driver information.
70	GetPowerMode	Returns the current power mode of a CD drive at a given device ID.
95	Get2KOffset	Returns last Prime call's offset from start of 2K physical block.
96	GetDriveType	Returns the type of CD drive at a given device ID.
98	GetBlockSize	Returns the block size of the disc at a given device ID.
120	ReturnDeviceIdent	Returns the DeviceIdent of the CD-ROM drive.
121	GetCDFeatures	Returns various CD-ROM drive features (spindle speed, drawer or tray type, etc.).

Control Call Descriptions

The purpose of this call is to allow the interruption of all asynchronous device driver operations. This call is not supported and returns `noErr` if called.

Input Parameters:

`csCode`	1
`csParam[0-10]`	Reserved.

Status Return Codes:

`noErr`	This is the only expected status return code.

VerifyTheDisc

The purpose of this call is to ensure the validity of the data on the disc. Since CD-ROMs cannot be changed (they are read-only, as the name implies), when this control call is made, the CD-ROM driver will return `noErr` (as long as a disc is mounted). If no disc is mounted, the CD-ROM driver will return `offLinErr`.

Input Parameters:

`csCode`	5
`csParam[0-10]`	Reserved.

Status Return Codes:

`noErr`	The entire disc was successfully verified with no errors detected.
`offLinErr`	No disc inserted at the specified device ID.

FormatTheDisc

The purpose of this call is to format and initialize the CD-ROM disc for use in the Macintosh. However, a CD-ROM cannot be modified after it has been pressed, so this command always returns a `writErr` status code.

Input Parameters:

`csCode`	6
`csParam[0-10]`	Reserved.

Status Return Codes:

`writErr`	This control call is not valid for this device.

EjectTheDisc

The purpose of this call is to cause a disc eject at a specified device ID. First, a `Prevent/Allow Medium Removal` command is issued to reactivate the external Eject button on the front panel of the CD-ROM drive. Then, the specified device is sent an "eject" command to eject the disc. Next, the `dNeedTime` flag in the specified device ID's DCE is set so that new disc insertions will be periodically checked for (see the `accRun` control call for more details). Finally, the drive queue entry associated with this drive is marked as being Off-Line and Ejected.

Input Parameters:

csCode	7
csParam[0-10]	Reserved.

Status Return Codes:

noErr	The entire disc was successfully ejected.
offLinErr	No disc inserted at the specified device ID.

GetDriveIcon

The purpose of this call is to return ICN#-style icon data and name string that is typically displayed by the Startup Disk Control Panel module. The drive icon is the same as the media icon in this driver. There is no true drive icon because there is no way of determining if an internal or external drive should represented. Starting on version 5.1, the driver returns a special icon when a CD-Extra is present in the drive, and a generic icon when any other type of CD is present. See Figure 1 for a picture of the icons returned.

Input Parameters:

csCode	21
csParam[0-10]	Reserved.

Output Produced:

csParam[0-1]	Address of ICN# and name string.

Status Return Codes:

noErr	This is the only possible status return code.

Generic CD icon

CD-Extra icon

Figure 1: Icons returned by GetDriveIcon control call

GetMediaIcon

The purpose of this call is to return ICN#-style icon data and name string that is displayed whenever the disc media is shown. This icon can be seen on the desktop or when the "Get Info" (Cmd-I) command from the Finder is executed under System 7. See Figure 2 for a picture of the icons returned.

Input Parameters:

csCode	22
csParam[0-10]	Reserved.

Output Produced:

csParam[0-1]	Address of ICN# and name string.

Status Return Codes:

noErr	This is the only possible status return code.

Generic CD icon

CD-Extra icon

Figure 2 : Icons returned by `GetDriveIcon` control call

`DriveInfo`

The purpose of this call is to return information describing drive characteristics for the specified device ID. The values to be returned are defined in *Inside Macintosh, volume V*, pages 470-471. The 32-bit value returned by the Apple CD-ROM driver is `$00000B01`. Bits 7-0 equal to `0x01` means that this is an "unspecified" drive. Bit 8 equal to 1 means that this is an "external" drive (this will be set even if the drive is actually mounted internally). Bit 9 equal to 1 means that this is a SCSI drive (this will be set even if the drive is actually an ATAPI drive). Bit 10 equal to 0 means that this is a removable media drive. Bit 11 equal to 1 means that this is a "secondary" drive. All other bits are reserved.

Input Parameters:

`csCode`	23
`csParam[0-10]`	Reserved.

Output Produced:

`csParam[0-1]`	Drive information (`$00000B01`).

Status Return Codes:

`noErr`	This is the only possible status return code.

`accRun`

The purpose of this call is to perform operations required by the driver on an "as-needed" basis.

There is no need for an application to issue this call. Almost all Macintosh programs make a call to the MacOS routine `SystemTask()`, and one of the functions of that routine is occasionally to make some CPU time available to all devices that request it by executing its `accRun` call. (See *Inside Macintosh, Volume II,* pg. 189 for a further discussion of `accRun`). This is done by setting the `dNeedTime` flag in the Device Control Entry (DCE) for a given device.

Since the CD-ROM device is a removable media device and does not generate interrupts when a disc is inserted, the only way to detect disc insertion is by occasionally polling all devices known by the driver to be "empty" (disc is not inserted or is still off-line). So, the main function of the `accRun` control call is testing for disc insertions.

In addition, during the Macintosh boot process, if the driver is installed before the System file is read in, the driver cannot install its custom CD-ROM Disk Switch subroutine. When the System file is read in and installed, it will overwrite the low-memory global that defines the address of the standard Disk Switch routine. So, the installation of our custom Disk Switch routine is held off until the first invocation of `SystemTask()`. At that time, the System file has already been loaded and the standard Disk Switch routine has been initialized. The `accRun` control call also handles track play lists.

Input Parameters:

csCode	65
csParam[0-10]	Reserved.

Status Return Codes:

noErr	This is the only possible status return code.

SetPowerMode

The purpose of this call is to select the power mode of a CD-ROM drive. The currently available power modes are:

- pmActive
- pmStandby
- pmIdle
- pmSleep

Input Parameters:

csCode	70
csParam[0]:15-8	New power mode to be set: 0 -- Active, normal operation. 1 -- Standby, minimal energy-saving state. 2 -- Idle, substantial energy-saving state. 3 -- Sleep, maximum energy-saving state.
csParam[0]:7-0	Reserved.
csParam[1-10]	Reserved.

Status Return Codes:

noErr	Drive set to the new power mode.
paramErr	Invalid power mode sent to the driver.

ModifyPostEvent

The purpose of this call is to allow an application to modify the driver's behavior when non-HFS discs are inserted into the CD-ROM drive. The default action of the driver is to post a disc-inserted event for all discs that are inserted into the CD-ROM drive, regardless of whether or not the disc actually contains valid partition information and HFS file system. There may be some reason why the driver should not post the disc-inserted event and this call allows that "feature" to be turned off. When this feature is turned off for a specified device ID, it will stay off until it is reactivated via another call to ModifyPostEvent, or until the machine is restarted.

Input Parameters:

csCode	76
csParam[0]	0 -- Do not issue disc-insert events for non-HFS CD-ROM discs ¬0 -- Issue disc-insert events for non-HFS CD-ROM discs
	Reserved.

`csParam[1-10]`	Reserved

Status Return Codes:

`noErr`	This is the only possible status return code.

ChangeBlockSize

The purpose of this call is to allow an application to modify the "block size" the drive is currently using to read data. The standard HFS file system block size is 512 bytes/block. Applications and file system translators dealing with non-HFS file systems may require other block sizes to read their data correctly.

The supported block sizes are 512, 2048, 2056, 2336, 2340, 2352, 2646, and 2647 bytes. The AppleCD SC cannot handle block sizes of 2056, 2352, 2646, and 2647 bytes. The AppleCD SC Plus cannot handle block sizes of 2352, 2646, and 2647 bytes.

The following tables show the type of data that is returned for each of the valid block sizes:

CD-ROM Mode 1 Data Format:

Block Size (bytes)	Block Contents (size in bytes)
512	User data (512)
2048	User data (2048)
2336	User data (2048) + EDC (4) + Zero (8) + ECC (276)
2340	Header (4) + User data (2048) + EDC (4) + Zero (8) + ECC (276)
2352	Sync (12) + Header (4) + User data (2048) + EDC (4) + Zero (8) + ECC (276)

CD-ROM Mode 2 Data Format:

Block Size (bytes)	Block Contents (size in bytes)
2336	Mode 2 User data (2336)
2340	Header (4) + Mode 2 User data (2336)
2352	Sync (12) + Header (4) + Mode 2 User data (2336)

CD-ROM Mode 2 Form 1 Data Format (CD-XA or CD-I):

Block Size (bytes)	Block Contents (size in bytes)
512	User data (512)
2048	User data (2048)
2056	Subheader (8) + User data (2048)
2336	Subheader (8) + User data (2048) + EDC (4) + ECC (276)
2340	Header (4) + Subheader (8) + User data (2048) + EDC (4) + ECC (276)
2352	Sync (12) + Header (4) + Subheader (8) + User data (2048) + EDC (4) + ECC (276)
2646	Sync (12) + Header (4) + Subheader (8) + User data (2048) + EDC (4) + ECC (276) + Byte Error Flags (294)

2647	Sync (12) + Header (4) + Subheader (8) + User data (2048) + EDC (4) + ECC (276) + Block Error Flag (1) + Byte Error Flags (294)

CD-ROM Mode 2 Form 2 Data Format (CD-XA or CD-I):

Block Size (bytes)	Block Contents (size in bytes)
2336	Subheader (8) + User data (2324) + Reserved or EDC (4)
2340	Header (4) + Subheader (8) + User data (2324) + Reserved or EDC (4)
2352	Sync (12) + Header (4) + Subheader (8) + User data (2324) + Reserved or EDC (4)
2646	Sync (12) + Header (4) + Subheader (8) + User data (2324) + Reserved or EDC (4) + Byte Error Flags (294)
2647	Sync (12) + Header (4) + Subheader (8) + User data (2324) + Reserved or EDC (4) + Block Error Flag (1) + Byte Error Flags (294)

Input Parameters:

`csCode`	79
`csParam[0]`	Block size the drive should use from now on.
`csParam[1-10]`	Reserved.

Status Return Codes:

`noErr`	The block size change completed successfully.
`paramErr`	An invalid block size was specified.

SetUserEject

The purpose of this call is to allow the user to enable/disable the physical *Eject* button on the front of the CD-ROM device while a disc is in the drive.

The CD-ROM device does not generate an interrupt when a disc is inserted or ejected, so if the button is left active we have no way of accurately knowing whether or not there is really a disc in the drive. The MacOS assumes it has final say over when an ejectable disc is actually ejected, and issues an `EjectTheDisc` control call when it wishes to eject the disc.

This call provides a method to disable/enable the manual eject button for the drive on the specified device ID. Internal variables are maintained for each device ID that reflect the current state of the manual eject button. If the button is enabled, the driver sets the `dNeedTime` flag in the appropriate DCE, telling the driver to periodically poll the device to see whether a disc has been ejected, or inserted. This is the only way the driver has to ensure the validity of its global variables in this situation. When a disc is inserted, the button setting reverts back to disabled.

Input Parameters:

`csCode`	80
`csParam[0]`	1 Enable the manual eject button. ¬1 Disable the manual eject button. {default}
`csParam[1-10]`	Reserved.

Status Return Codes:

noErr	The button was successfully enabled/disabled.
offLinErr	No disc inserted in the drive.

SetPollFreq

The purpose of this call is to allow an application to specify a polling frequency (in Macintosh time ticks) different from the driver default value (120). This polling frequency is the value used for determining how often the `accRun` control call is issued, as well as the polling frequency for a VBL task initiated during the execution of the driver's custom Disk Switch routine.

Input Parameters:

`csCode`	81
`csParam[0]`	Number of Macintosh ticks to be used for polling frequency.
`csParam[1-10]`	Reserved.

Status Return Codes:

`noErr`	This is the only possible status return code.

Quiescence

The purpose of this call is to allow the user to enable/disable the driver quiescent mode. This is used when an application wishes to communicate directly to the drive, in effect bypassing the CD driver. In this case the procedure for communicate directly with the drive, is as follows:

1. Wake up the drive using the SetPowerMode command.
2. Place the driver into quiescent mode.

   During quiescent mode, the driver will be completely inactive. **Note that if the driver is not put into quiescent mode immediately after it awakens the drive, the driver will during the first accRun put that empty drive back to sleep.**
3. After the application is done talkimg to the drive, remove the driver from quiescent mode.

Input Parameters:

`csCode`	1079	
`csParam[0]`	0	Enable Quiescence.
	1	Disable Quiescence.
`csParam[1-10]`	Reserved.	

Status Return Codes:

noErr	Quiescence mode was successfully enabled/disabled.

ReadTOC

The purpose of this call is to allow an application to receive various types of Table Of Contents (TOC) data from the drive. There are six varieties of the `ReadTOC` call:

csParam[0]	

csParam[0] (word)	Description
1	Transfers the first track number of the first (or only) session and the last track number of the last (or only) session. AppleCD SC, AppleCD SC Plus, and AppleCD 150: Transfers the first and last user track number (in BCD) of the first (or only) session.
2	Transfers the starting address of the Lead Out area of the last (or only) session in BCD MIN-SEC-FRAME format. AppleCD SC, AppleCD SC Plus, and AppleCD 150: Transfers the starting address of the Lead Out area of the first (or only) session in BCD MIN-SEC-FRAME format.
3	Transfers the starting address of each track in a range of specified tracks, starting from the specified starting track and proceeding in ascending order until the specified buffer is filled or the data for the last track on the disc is transferred, whichever is less. Four bytes of information are transferred to describe the starting address. AppleCD SC, AppleCD SC Plus, and AppleCD 150: Only the first session TOC information can be accessed in a multi-session disc.
4	Transfers the entire table of contents for a disc (all track entries and points A0, A1, and A2) to the specified buffer. AppleCD SC: Not available.
5	Transfers information about the number of sessions on the disc and the location of the last session. AppleCD SC, AppleCD SC Plus, and AppleCD 150: Not available.
6	Transfers all Q-subcode entries in the lead-in areas of a disc (including TOC information), starting from the specified session number and proceeding in ascending order until the specified buffer is filled, or the data for the last Q-subcode entry on the disc is transferred, whichever is less. AppleCD SC, AppleCD SC Plus, and AppleCD 150: Not available.

For transfer types *1*, *2*, and *5*, the TOC information is returned directly in the caller's `csParam[0-10]` array. For types *3*, *4*, and *6*, the caller is required to specify a buffer to place the information into. In the case of a type *4* transfer, the buffer must be 512 bytes long.

Input Parameters:

`csCode` 100

For Type 1:

`csParam[0]` The transfer type value 1.

`csParam[1-10]` Reserved.

For Type 2:

`csParam[0]` The transfer type value 2.

`csParam[1-10]` Reserved.

For Type 3:

`csParam[0]` The transfer type value 3.

`csParam[1-2]` The address of a buffer to return the requested information.

`csParam[3]` The size of the buffer specified in `csParam[1-2]`, in bytes.

`csParam[4]` The MSByte (the upper 8 bits) of this 16-bit word must contain the starting track number (in BCD).

For Type 4 (Not available on AppleCD SC):

`csParam[0]` The transfer type value 4.

`csParam[1-2]` The address of a buffer to return the requested information.

For Type 5 (Not available on AppleCD SC, AppleCD SC Plus, and AppleCD):

`csParam[0]` The transfer type value 5.

`csParam[1-10]` Reserved.

For Type 6 (Not available on AppleCD SC, AppleCD SC Plus, and AppleCD):

`csParam[0]` The transfer type value 6.

`csParam[1-2]` The address of a buffer to return the requested information.

`csParam[3]` The size of the buffer specified in `csParam[1-2]`, in bytes.

`csParam[4]` The `MSByte` (the upper 8 bits) of this 16-bit word must contain the starting session number (in BCD).

Output Produced:

For Type 1:

`csParam[0]:15-8` First user track number (in BCD).

`csParam[0]:7-0` Last user track number (in BCD).

For Type 2:

`csParam[0]:15-8` MIN field of a MIN-SEC-FRAME descriptor that describes the Lead Out starting address (in BCD).

`csParam[0]:7-0` SEC field of the Lead Out starting address descriptor.

`csParam[1]:15-8` FRAME field of the Lead Out starting address descriptor.

For Type 3:

The user must specify a return buffer address and the size of the buffer. The buffer is filled with 4-byte entries for each track in the requested range. Each entry is of the form:

Entry (byte offset)	Contents
0	Bits 7-4: Reserved. Bits 3-0: Control Field (see Track Control Field at the beginning of this document for more information).
1	The MIN field of a MIN-SEC-FRAME descriptor that describes the start of the associated track.
2	The SEC field of the track starting address descriptor.
3	The FRAME field of the track starting address descriptor.

Note: no entries will be returned for data tracks on CD-I discs.

For Type 4 (Not available on AppleCD SC):

The user must specify a return buffer address. The buffer must be 512 bytes long, and is filled with the entire table of contents, which is returned in the form:

Entry (byte offset)	Contents
0	Reserved.
1-5	TOC entry for point A0 (first track number).
6-10	TOC entry for point A1 (last track number).
11-15	TOC entry for point A2 (address of beginning of lead out area).
16-20	TOC entry for track 1.

. .
. .
. .

506-510 TOC entry for track 99.

Each five-byte entry is of the form:

Entry (byte offset)	Contents
0	Reserved. Bits 7-4: Reserved. Bits 3-0: Control Field (see Track Control Field at the beginning of this document for more information).
1	Point or track number.
2	PMIN.
3	PSEC.
4	PFRAME.

Note: this option is not valid for the AppleCD SC drive.

Type 5 (Not available on AppleCD SC, APPLECD SC Plus, and AppleCD 150):

`csParam[0]`	The first session number (in BCD).
`csParam[1]`	The last session number (in BCD).
`csParam[2]`	First track number of last session (in BCD).
`csParam[3]:15-12`	Reserved
`csParam[3]:11-8`	Control Field of first track in last session. (see Track Control Field at the beginning of this document for more information).
`csParam[3]:7-0`	The MIN field of a MIN-SEC-FRAME descriptor that describes the start of the first track of the last session (in BCD).
`csParam[4]:15-8`	The SEC field of the track starting address descriptor (in BCD).
`csParam[4]:7-0`	The FRAME field of the track starting address descriptor (in BCD).

For Type 6 (Not available on AppleCD SC, AppleCD SC Plus, and AppleCD 150):

The user must specify a return buffer address and the size of the buffer. The buffer is filled with a 4-byte header and 8-byte entries for each Q-subcode entry in the requested range. The first three entries are for points A0, A1, and A2, and are followed by the track entries for that session. After that, the Q-subcode entries are sorted in ascending order using the ADR field and within each ADR group by ascending order of the POINT field. This pattern is repeated for each session on the disc.

The 4-byte header is of the form:

Header (byte offset)	Contents
0-1	Total number of TOC data bytes transferred excluding these two length bytes.
2	First session number on the disc (in BCD).
3	Last session number on the disc (in BCD).

Each TOC entry is of the form:

Entry (byte offset)	Contents
0	The session number (in hex).
1	Bits 7-4: ADR field, i.e., Q-subcode mode field Bits 3-0: Control Field (see Track Control Field at the beginning of this document for more information)
2	TNO - Track number.
3	POINT.
4	MIN.
5	SEC.
6	FRAME.
7	ZERO.
8	PMIN.
9	PSEC.
10	PFRAME.

The values come directly from the disc and will be in either BCD or hexadecimal format, depending on the field. The session number, ADR/Control and POINT fields are in hexadecimal. All other fields are in BCD. For more information on the format of the TOC, please refer to the "Yellow Book" specification for CD-ROM discs, ISO 10149.

In some early CD-300 units, the values for Entries *2* and *3* are swapped. This is due to a firmware bug, which was corrected on later drives. You should always `OR` the two values together to always receive the proper value.

Status Return Codes:

`noErr`	TOC data successfully retrieved.
`paramErr`	An invalid TOC type specified. This can happen due to the TOC type being out of range, or due to the call not being supported by the drive in question (e.g., Type 6 call issued to AppleCD SC drive).

ReadTheQSubcode

The purpose of this call is to allow an application access to the Q-subcode information for the current track. The current track is the track that the optical pickup is currently over on a disc.

Input Parameters:

`csCode`	101
`csParam[0-4]`	Reserved (Q-Subcode data will be returned here).
`csParam[5-10]`	Reserved.

Output Produced:

`csParam[0]:[15-8]`	Control Field information (see Track Control Field at the beginning of this document for more information).
`csParam[0]:[7-0]`	Current track number (in BCD).
`csParam[1]:[15-8]`	Current index number in the current track.
`csParam[1]:[7-0]`	MIN field of a MIN-SEC-FRAME descriptor, which describes the relative running time from the beginning of the

track.

csParam[2]:[15-8]	SEC field for a MIN-SEC-FRAME descriptor.
csParam[2]:[7-0]	FRAME field for a MIN-SEC-FRAME descriptor.
csParam[3]:[15-8]	AMIN field of an AMIN-ASEC-AFRAME descriptor, which describes the relative running time from the beginning of the disc.
csParam[3]:[7-0]	ASEC field of an AMIN-ASEC-AFRAME descriptor.
csParam[4]:[15-8]	AFRAME field of an AMIN-ASEC-AFRAME descriptor.
csParam[4]:[7-0]	LSB unused.

Status Return Codes:

noErr	Q-subcode data successfully retrieved.

ReadHeader

The purpose of this call is to allow the user/application access to the header information for a specified logical block. This information concerns the absolute AMIN-ASEC-AFRAME address of a given logical block (the absolute running time from the beginning of the disc) and the CD-ROM Data Mode of that logical block. If a requested block is within an audio track (CD-DA) of the disc, a paramErr will be returned.

Input Parameters:

csCode	102
csParam[0-1]	32-bit logical block address from which to retrieve Header information.
csParam[2-10]	Reserved.

Output Produced:

csParam[0]:[15-8]	AMIN field of an AMIN-ASEC-AFRAME descriptor, which describes the absolute running time from the beginning of the disc. Value is in BCD format.
csParam[0]:[7-0]	ASEC field of an AMIN-ASEC-AFRAME descriptor. Value is in BCD format.
csParam[1]:[15-8]	AFRAME field of an AMIN-ASEC-AFRAME descriptor. Value is in BCD format.
csParam[1]:[7-0]	CD-ROM data mode of the logical block which defines what type of data will be within the 2048 data bytes and 288 auxiliary bytes of the logical block:

Data Mode	User Data Field (2048 bytes)	Auxiliary Field 288 bytes)
0	all bytes zero	all bytes zero
1	user data	EDC, L-EC bytes
2	user	data user data

Status Return Codes:

Status Return Codes:

`noErr`	Header information successfully retrieved.
`paramErr`	The requested block is within an audio track (CD-DA) of the disc.

AudioTrackSearch

The purpose of this call is to position the optical pick-up at the specified audio address. In addition, you can also tell the drive either to start to play at the given address, or to pause at this address until an `AudioPlay` command is executed and in what Play Mode it should output the audio information.

Input Parameters:

`csCode`	103
`csParam[0]`	The optical pick-up positioning address type (see "Optical Positioning Types" at the beginning of this document for more information).
`csParam[1-2]`	32-bit value representing the search address required by the corresponding type value in `csParam[0]` (see above).
`csParam[3]`	A boolean value indicating what the drive should do when positioning of the pick-up is complete: 0 Enter a Hold Track state ("pause") ¬0 Start playing the audio track, according to the specified Play Mode.
`csParam[4]`	The "Play Mode" (see "Audio Play Modes" at the beginning of this document for more information).
`csParam[5-10]`	Reserved.

Status Return Codes:

`noErr`	Optical pick-up successfully positioned.
`paramErr`	Invalid type, Play Mode, or Search Address value specified or tried to execute this audio operation on a data track.

AudioPlay

The purpose of this call is to position the optical pick-up at the specified audio address and begin playback of the audio at that address. It can be similar to the `AudioTrackSearch` command that automatically begins playing at the specified Search Address (see the documentation on the `AudioTrackSearch` call for more detail). However, this command can also be used to specify a "Stop Address."

The proper way to use the `AudioPlay`-related commands is:

- First, specify where the audio playback should stop, by issuing an `AudioPlay` command to mark the stopping address.
- Next, (optionally) to pre-position the optical pick-up where the audio playback should start, by issuing an `AudioTrackSearch` command and entering the "pause" state.
- At this point, playback can then be started by an `AudioPlay` or `AudioTrackSearch` command and the drive will play audio until it reaches the position marked by the first `AudioPlay` command.

If no stop address is specified before issuing the actual play command, the current active stop address in the drive may be lower than the address at which we are trying to play audio, causing the drive to return an error when the play command is issued.

when the play command is issued.

This model should always be followed irregardless of the audio play mode to be used, even when playing track lists.

Input Parameters:

`csCode`	104
`csParam[0]`	The optical pick-up positioning address type (see "Optical Positioning Types" at the beginning of this document for more information).
`csParam[1-2]`	32-bit value representing the search address required by the corresponding type value in `csParam[0]`.
`csParam[3]`	A boolean value indicating whether the address specified in `csParam[1-2]` is a starting address or a stopping address. 0 The address specified is a Starting Address. ¬0 The address specified is a Stopping Address.
`csParam[4]`	The "Play Mode" (see "Audio Play Modes" at the beginning of this document for more information).
`csParam[5-10]`	Reserved.

Status Return Codes:

`noErr`	Operation successful.
`paramErr`	Invalid type, Play Mode, or Search Address value specified or tried to execute this audio operation on a data track.

AudioPause

The purpose of this call is to cause the CD-ROM drive to enter either a Hold Track ("Pause") or Play state.

Note: This command is only valid after the issuance of an `AudioTrack` Search, or `AudioPlay` command.

Input Parameters:

`csCode`	105
`csParam[0-1]`	A (long) boolean value that tells the CD-ROM drive either to pause or continue: 0 Release the "Pause" state and continue playing at the same Q-subcode address before the last `AudioPause`. 1 Enter a Hold Track ("Pause") state.
`csParam[2-10]`	Reserved.

Status Return Codes:

`noErr`	CD-ROM device successfully paused or released from pause.
`paramErr`	Invalid pause value specified or tried to execute this audio operation on a data track.

AudioStop

The purpose of this call is to specify an address to the CD-ROM drive at which it should end the current `Play` operation.

Note: If the optical pick-up positioning address type is *0* and the stop address is also *0*, the command will immediately stop the Play operation.

Input Parameters:

`csCode`	106
`csParam[0]`	The optical pick-up positioning address type (see "Optical Positioning Types" at the beginning of this document for more information).
`csParam[1-2]`	32-bit value representing the stop address required by the corresponding type value in `csParam[0]`.
`csParam[3-10]`	Reserved.

Status Return Codes:

`noErr`	Optical pick-up successfully positioned.
`paramErr`	Invalid type or Search Address value specified or tried to execute this audio operation on a data track.

AudioStatus

The purpose of this call is to return the current audio play status information and the starting Q-subcode address for the current track. The audio play status information includes the Play Mode, and status and control information for the current track.

Input Parameters:

`csCode`	107
`csParam[0-10]`	Reserved (Audio Status data will be returned here).

Output Produced:

`csParam[0]:15-8`	The Status Field: 0 Audio Play operation in progress. 1 CD-ROM device currently in Hold Track ("Pause") state 2 MUTING-ON operation in progress 3 Audio Play completed 4 Error occurred during audio play operation 5 Audio play operation not requested
`csParam[0]:7-4`	Reserved.
`csParam[0]:3-0`	Play Mode (see "Audio Play Modes" at the beginning of this document for more information).
`csParam[1]:15-12`	Reserved.
`csParam[1]:11-8`	The Control Field (see Track Control Field at the beginning of this document for more information).
`csParam[1]:7-0`	The AMIN field of an AMIN-ASEC-AFRAME descriptor that describes the current Q-subcode.
`csParam[2]:15-8`	The ASEC field of an AMIN-ASEC-AFRAME descriptor for the current Q-subcode.
`csParam[2]:7-0`	The AFRAME field of an AMIN-ASEC-AFRAME descriptor for the current Q-subcode.

Status Return Codes:

`noErr`	Status information successfully read.
`paramErr`	Tried to execute this audio operation on a data track.

AudioScan

The purpose of this call is to request the CD-ROM drive to perform a fast-forward or fast-reverse operation starting from a specified address.

Input Parameters:

`csCode`	108
`csParam[0]`	The optical pick-up positioning address type (see "Optical Positioning Types" at the beginning of this document for more information).
`csParam[1-2]`	32-bit value representing the search address required by the corresponding type value in `csParam[0]`.
`csParam[3]`	The direction the scan should take place: 0Scan in the fast-forward direction ¬0Scan in the fast-reverse direction
`csParam[4-10]`	Reserved.

Status Return Codes:

`noErr`	Optical pick-up successfully positioned.
`paramErr`	Invalid type or Search Address value specified or tried to execute this audio operation on a data track.

AudioControl

The purpose of this call is to set the output volume for the drive's audio channels. Each channel can be given a value from zero to 255, where zero indicates that the channel output is muted, and 255 indicates maximum volume.

Notes:
- This call can be used even if no disc is mounted.
- Beginning with driver version 4.0.5, the last volume setting used before powering down a drive will be restored when the drive is reinitialized.
- This command is not available on the AppleCD SC.

Input Parameters:

`csCode`	109
`csParam[0]:15-8`	Left channel volume (0-`FFh`).
`csParam[0]:7-0`	Right Channel volume (0-`FFh`).
`csParam[1-10]`	Reserved.

Status Return Codes:

`noErr`	Audio volume successfully set.

ReadMCN

The purpose of this call is to allow an application access to the Media Catalog Number for the disc.

The Media Catalog Number consists of the UPC/Bar Code information for the disc. The output format was changed from binary data to a Pascal string beginning with driver version 5.0. Applications can check the upper bit of `csParam[0]` for the `MCVal` bit. If set, the application can use the version 4.0 output format. If the `MCVal` is not set, but the upper byte in `csParam[0]` is not equal to zero, then the application can use the version 5.0 output.

Notes:
- This command is not available on the AppleCD SC.
- Driver version 4.0.X does not return the correct MCN data when using an AppleCD 300.

Input Parameters:

`csCode`	110
`csParam[0-6]`	Reserved (MCN data will be returned here).
`csParam[7-10]`	Reserved.

Output Produced (version 5.X):

`csParam[0-6]`:	Pascal string containing the MCN (UPC/Bar Code). If the string length (upper eight bits of `csParam[0]`) is zero, an MCN was not found on the disc.

Output Produced (version 4.X):

`csParam[0]:[15-8]`	`MCVal` bit (bit 15). If this bit is set to one, then the Media Catalog Number data was found and the rest of the returned fields are valid. If this bit is set to zero, then the Media Catalog Number data was not found and the rest of the returned fields are invalid.
`csParam[0]:[7-0]`	MSB of the Media Catalog Number (UPC/Bar Code) data if the `MCVal` bit is set.
`csParam[1-7]`	Contains the rest of the Media Catalog Number (UPC/Bar Code) data if the `MCVal` bit is set. The LSB of `csParam[7]` contains the LSB of the Media Catalog Number data.

Status Return Codes:

`noErr`	Media Catalog Number data successfully retrieved, or no MCN was found on the disc.
`paramErr`	Tried to execute this audio operation on a data track.

ReadISRC

The purpose of this call is to allow an application access to the International Standard Recording Code information for a specified track on a disc. For more information about the ISRC, please refer to ISO 10149. The output format was changed from binary data to a Pascal string beginning with driver version 5.0. Applications can check the upper bit of `csParam[0]` for the `TCVal` bit. If set, the application can use the version 4.0 output format. If the `TCVal` is not set but the upper byte in `csParam[0]` is not equal to zero, then the application can use the version 5.0 output.

Notes:
- This command is not available on the AppleCD SC.
- Driver version 4.0.X does not return the correct ISRC data when using an AppleCD 300.

Input Parameters:

`csCode`	111
`csParam[0]`	A track number (in BCD)

`csParam[0]`	A track number (in BCD).
`csParam[1-10]`	Reserved (ISRC data will be returned here).

Output Produced (version 5.X):

`csParam[0-6]:`	Pascal string containing the ISRC. If the string length (upper eight bits of `csParam[0]`) is zero, an ISRC was not found for the specified track.

Output Produced (version 4.X):

`csParam[0]:[15-8]`	`TCVal` bit (bit 15). If this bit is set to one, then the ISRC data was found for the specified track and the rest of the returned fields are valid. If this bit is set to zero, then the ISRC data was not found for the specified track and the rest of the returned fields are invalid.
`csParam[0]:[7-0]`	MSB of the ISRC data for the specified track if the `TCVal` bit is set.
`csParam[1-7]`	Contains the rest of the ISRC data if the `TCVal` bit is set. The LSB of `csParam[7]` contains the LSB of the ISRC data.

Status Return Codes:

`noErr`	ISRC data successfully retrieved.
`paramErr`	Tried to execute this audio operation on a data track.

ReadAudioVolume

The purpose of this call is to return the current audio volume control data for the drive.

Notes:
- This call can be used even if no disc is mounted.
- This call is not available on the AppleCD SC.

Input Parameters:

`csCode`	112
`csParam[0]`	Reserved (Audio Volume data will be returned here).
`csParam[1-10]`	Reserved.

Output Produced:

`csParam[0]:15-8`	The left channel volume `(0-FFh)`.
`csParam[0]:7-0`	The right channel volume `(0-FFh)`.

Status Return Codes:

`noErr`	Status information successfully read.

GetSpindleSpeed

The purpose of this call is to return the current rotational speed of the drive. This call always return *00* on normal speed (1X) drives, like the Apple AppleCD SC and AppleCD SC Plus/AppleCD 150.

Input Parameters:

`csCode`	113
`csParam[0]`	Reserved (Speed information will be returned here).
`csParam[1-10]`	Reserved.

Output Produced:

`csParam[0]`	The current rotational speed:

00 Normal speed.

FF Maximum speed.

Status Return Codes:

`noErr` Speed information successfully read.

SetSpindleSpeed

The purpose of this call is to set the rotational speed of the drive. This call will return `paramErr` for an AppleCD SC or AppleCD SC Plus/AppleCD 150 if the input parameter is not *00* (normal speed).

Input Parameters:

`cscode` 114

`csParam[0]` Speed setting:

00 Normal speed

FF Maximum speed

`csParam[1-10]` Reserved.

Status Return Codes:

`noErr` Rotational speed successfully set.

`paramErr` An invalid speed setting was specified.

ReadAudio

The purpose of this call is to read digital audio and/or subcode data from the disc into a specified buffer. The starting audio address may be specified in one of three formats (see the "Audio Play Modes" at the beginning of this document for more information).

There are four possible formats for the audio data that is returned:

Type Values	Description
0	Audio data only--2352 bytes per audio block.
1	Audio data with Q-subcode--2352 audio bytes, 10 bytes Q-subcode,and 6 zero bytes = 2368 bytes per block.
2	Audio data with all subcode data (P~W)--2352 audio bytes and 96 subcode bytes = 2448 bytes per block.
3	All subcode data only (P~W)--96 subcode bytes per block.

The position of an audio block cannot be determined exactly. Thus, two `ReadAudio` calls with the same starting address will not necessarily return the same data in the same position. Typically the data will be shifted up to 1.5 frames (3528 bytes). To read large sections of audio data using multiple `ReadAudio` calls and not have a gap are overlap in the audio data, the `ReadAudio` calls must occur frequently enough to prevent the internal drive buffer from overflowing. If the Macintosh is too slow, a `controlErr` will be returned along with the audio data. The `controlErr` indicates that the audio data just received has a gap or overlap in it.

For type *2* and type *3*, the *P*- and *Q*-subcodes are accurately aligned to the CD block boundary, but the *R*- through *W*-subcodes may be offset by one pack (i.e. 24 subcode bytes) from the *P*- and *Q*-subcodes. Thus, in any 96-byte subcode data from block n, the last 24 bytes contain *R*- through *W*-subcodes from the next block (*n*+1).

The current volume setting has no effect on the audio data that is returned. When using `ReadAudio` with type *3*, the audio is played through any connected speakers. The audio can be muted to avoid hearing anything while reading the subcode data. For types *0*, *1*, and *2* the data rate can be adjusted by changing the drive speed prior to calling `ReadAudio`. For type *3* only, subcode data is always transferred at an average throughput of 150 KBytes per second.

This call is not supported by AppleCD SC, AppleCD SC Plus/AppleCD 150. Most ATAPI drives will not support `ReadAudio` type *1*, *2* or *3*.

Input Parameters:

`cscode`	115
`csParam[0]`	Audio Type (see above)
`csParam[1-2]`	The address of the buffer to return the requested information.
`csParam[3]`	The starting address type (see "Optical Positioning Types" at the beginning of this document for more information).
`csParam[4-5]`	32-bit value representing the start address required by the corresponding type value in `csParam[3]`.
`csParam[6]`	The number of consecutive audio blocks to return.
`csParam[7-10]`	Reserved.

Status Return Codes:

`noErr`	Audio data successfully retrieved.
`paramErr`	An invalid audio type, address type, or audio address specified or tried to execute this audio operation on a data track.

ReadAllSubcodes

The purpose of this call is to read consecutive blocks of subcode information (*P~W*) as the drive is playing audio. When one of the commands is issued that begins playing audio, the subcode information corresponding to the audio blocks is stored in a buffer within the drive. `ReadAllSubcodes` retrieves the subcode information from this buffer. If the number of subcode blocks requested exceeds the quantity currently in the buffer, the control call waits until more subcode blocks are available before returning data. An option allows any subcode data currently in the buffer to be purged, allowing the host to re-synchronize to the latest subcode data.

If the Macintosh does not send `ReadAllSubcode` requests frequently enough, a `controlErr` will be returned indicating that the internal drive buffer overflowed and some subcode data was lost. A `ReadAllSubcodes` call must be made with the purge parameter set to clear the overflow condition.

Note: The *R*- through *W*-subcodes that are returned are accurately aligned to the CD block boundary, but the *P*- and *Q*-subcodes may be offset by one pack (i.e. 24 subcode bytes) from the *R*- through *W*-subcodes. Thus, in any 96-byte subcode data from block *n*, the first 24 bytes may contain the P and Q subcodes from the previous block (*n-1*).

This call is not supported by AppleCD SC, AppleCD SC Plus/AppleCD 150. Most ATAPI drives will not support this call.

Input Parameters:

`cscode`	116
`csParam[0-1]`	The address of the buffer to return the requested subcode blocks.
`csParam[2]`	A boolean value indicating whether the current information in the drive buffer should be purged before retrieving any subcode data: 0 retrieve buffered subcode blocks ¬0 delete any blocks in the subcode buffer and wait for new data
`csParam[3]`	The number of consecutive subcode blocks (96 bytes per block) to return.
`csParam[4-10]`	Reserved.

Status Return Codes:

`noErr`	Audio data successfully retrieved.
`controlErr`	A buffer overflow occurred during the `ReadAllSubcodes` command execution.
`paramErr`	Audio operation on a data track.

SetTrackList

The purpose of this call is to indicate to the driver the order in which audio tracks will be played. This ordering is passed in as a pointer to an array (of bytes)--called the track play list--containing the track numbers in the order they should be played. The track play list should have a maximum of 99 entries. The driver verifies that all elements in the track play list are valid.

The proper way to use the audio track list related commands is:

- First, use the `SetTrackList` command to send to the driver an audio track list defining the order in which you want the audio tracks to be played.
- Next, specify where the audio playback should stop, by issuing an `AudioPlay` command to mark the last track in the list to be played.
- Next (optionally), to pre-position the optical pick-up on the track in the list where the audio playback should start, by issuing an `AudioTrackSearch` command and entering the "pause" state.
- At this point, playback can then be started by an `AudioPlay` or `AudioTrackSearch` command, and the drive will play audio until it finishes the track marked by the first `AudioPlay` command.

All `AudioPlay` and `AudioTrackSearch` commands issued should use the optical positioning type (`csParam[0]`) set to *3* so the address field (`csParam[1-2]`) can be interpreted as an index into the driver's audio track play list.

Input Parameters:

`cscode`	122
`csParam[0]`	Length of the new track play list (must be less than 100).
`csParam[1-2]`	Pointer to new track play list.
`csParam[3-10]`	Reserved.

Status Return Codes:

`noErr`	Track list data successfully set.
`paramErr`	Either (1) the pointer to track list wasn't provided by the client code, (2) at least one element of the play list wasn't valid, or (3) more than 99 elements were passed in.

GetTrackList

Retrieve the current track play list from driver (see `SetTrackList` command).

Input Parameters:

`cscode`	123
`csParam[0]`	Reserved (Length of current track play list will be returned here).
`csParam[1-2]`	The address of a buffer to return the current track play list data.
`csParam[3-10]`	Reserved.

Output Produced:

`csParam[0]`	Length of current track play list.
`csParam[1-2]`	The address of the buffer containing the current track play list data.

Status Return Codes:

`noErr`	Track list data successfully retrieved.
`paramErr`	Pointer to track list wasn't provided by the client code.

GetTrackIndex

Return the current track play list index. The `AudioPlay` and `AudioTrack` Search commands set track play list index when called with optical positioning type (`csParam[0]`) set to *3*. The `AudioStop` command and `Prime` calls (i.e., `Read`) will stop audio play and reset the track play list index (to *-1*).

Input Parameters:

`cscode`	124
`csParam[0]`	Reserved (Current track index will be returned here).
`csParam[1-10]`	Reserved.

Output Produced:

`csParam[0]`	The current track index.

Status Return Codes:

`noErr`	This is the only possible status return code.

SetPlayMode

The CD-ROM driver can set the CD-ROM drive to play audio from a start address to an end address (`kNormalMode`), to randomly shuffle between tracks (`kShuffleMode`), or to play tracks in a specified order (`kProgMode`). For `kNormalMode` and `kProgMode`, play can be extended by repeating the tracks.

The `SetPlayMode` command is used to set the track play mode and the repeat mode.

The three possible track play modes are:

`kNormalMode`	0
`kShuffleMode`	1
`kProgMode`	2

Input Parameters:

`cscode`	125
`csParam[0]:15-8`	The repeat flag: True -- Repeat function is on. False -- Repeat function is off.
`csParam[0]:7-0`	The track play mode.
`csParam[1-10]`	Reserved.

Status Return Codes:

`noErr`	This is the only possible status return code.

GetPlayMode

Return current settings for repeat and track play mode (see `SetPlayMode`).

Input Parameters:

`cscode`	126
`csParam[0]`	Reserved (Track play mode will be returned here).
`csParam[1-10]`	Reserved.

Output Produced:

`csParam[0]:15-8`	The repeat flag: True -- Repeat function is on. False -- Repeat function is off.
`csParam[0]`:7-0	The track play mode.

Status Return Codes:

`noErr`	This is the only possible status return code.

GoodBye

Executes necessary cleanup before system shutdown.

Input Parameters:

`cscode`	-1
`csParam[0-10]`	Reserved.

Status Return Codes:

`noErr`	This is the only possible status return code.

Status Call Descriptions

DriveStatus

The purpose of this call is to return information about the drive/disc as described in *Inside Macintosh, Volume II*, page 215.

Input Parameters:

`cscode`	8
`csParam[0-10]`	Reserved (On exit, contains information that maps to the `DrvSts` record/structure).

Output Produced:

`csParam[0]`	Current track(always 0.
`csParam[1]:15`	Volume locked bit: 0 (volume not locked) 1 (volume locked)
`csParam[1]:7-0`	Disc in place: 0 (disc not in drive) 1 (disc in drive)
`csParam[2]:15-8`	Drive installed (always 1).
`csParam[2]:7`	Sides flag (always 0) (single-sided).
`csParam[2]:6-0`	Number of sides (always 1) (single-sided).
`csParam[3-4]`	Next queue entry (always 0).
`csParam[5]`	Reserved.
`csParam[6]`	Drive number.
`csParam[7]`	Drive reference number.
`csParam[8]`	File system identifier (always 1).
`csParam[9] 15-8`	Two-sided format (always 0) (single-sided).
`csParam[9] 7-0`	Reserved.
`csParam[10]`	Disk error count (always 0).

Status Return Codes:

`noErr`	This is the only possible status return code.

DriverGestalt

The purpose of this call is to return `Gestalt`-type information about the driver itself.

Currently defined `driverGestalt` selectors are:

`driverGestaltSyncResponse = sync; //`	True if the driver only behaves synchronously.
`driverGestaltVersResponse = vers; //`	Version number of the driver (First 4 bytes of the vers #1 resource).
`driverGestaltIntfResponse = intf; //`	Physical interface to device.
`driverGestaltDevTResponse = devt; //`	Type of device the driver is driving.
`driverGestaltBootResponse = boot; //`	PRAM value to designate this driver/drive. Valid only for ATAPI drives or when SCSI Manager 4.3 is active.

The following response buffers are defined for the `driverGestalt` selectors:

```
struct driverGestaltSyncResponse
{
    Boolean behavesSynchronously; // true/false = sync/async
};
typedef struct driverGestaltSyncResponse driverGestaltSyncResponse

struct driverGestaltVersResponse
{
    NumVersion driverVersion;
};
typedef struct driverGestaltVersResponse driverGestaltVersResponse

struct driverGestaltIntfResponse
{
    OSType interfaceType; // 'scsi' or 'atpi'
};
typedef struct driverGestaltIntfResponse driverGestaltIntfResponse

struct driverGestaltDevTResponse
{
    OSType deviceType; // 'cdrm'
};
typedef struct driverGestaltDevTResponse driverGestaltDevTResponse

typedef struct
{
    unsigned char extDev;   // SCSI = Target (upper 5 bits)
                            // SCSI = LUN (lower 3 bits)
                            // ATAPI = Bus
    unsigned char partition; // Unused
    unsigned char SIMSlot;  // SCSI = Slot
                            // ATAPI = 0x20 (internal ATA)
    unsigned char SIMsRSRC; // SCSI = sRsrcID
                            // ATAPI = 0 (unused)
} driverGestaltBootResponse;
```

Input Parameters:

`cscode`	43
`csParam[0-1]`	`driverGestaltSelector`
`csParam[2-3]`	Reserved (`driverGestaltResponse` data will be returned here).
`csParam[4-10]`	Reserved.

Output Produced:

`csParam[2-3]:`	`driverGestaltResponse`

Status Return Codes:

`noErr`	This is the only possible status return code.

GetPowerMode

The purpose of this call is to return the current power mode of a CD-ROM drive (see `SetPowerMode`).

Input Parameters:

`cscode`	70
`csParam[0-10]`	Reserved.

Output Produced:

csParam[0]:15-8	The current power mode.

Status Return Codes:

noErr	This is the only possible status return code.

Get2KOffset

The purpose of this call is to return information to the caller that allows subsequent Prime calls to be aligned to the 2K physical block size of the CD-ROM. This can be used in applications (e.g., QuickTime) to increase streaming performance by avoiding data requests that span across a 2K-disc block. This call returns the number of bytes between the beginning of the 2K physical block and the first byte read in the previous Prime call. Possible values are 0, 512, 1024, and 1536. For example, if a Prime call causes a read of the last 512 bytes within a 2K physical block, this call will return 3 x 512 = 1536 bytes. This is the offset from the beginning of the 2K block to the start of the first data byte read. Note that this call assumes that the file system actually makes a Prime call and that a file system cache did not fill the read request.

Input Parameters:

cscode	95
csParam[0-1]:	Reserved (on exit contains 2K offset).
csParam[2-10]	Reserved.

Output Produced:

csParam[0-1]	The offset between the beginning of the 2K physical block and the first byte of the last read call.

Status Return Codes:

noErr	The offset for the specified device ID was returned.
statusErr	Prime was not previously called.

GetDriveType

The purpose of this call is to return to the caller the type of CD-ROM drive located at the specified device ID.

Note: Applications should begin using the new GetCDFeatures call instead of GetDriveType.

Input Parameters:

cscode	96
csParam[0]:	Reserved (on exit contains the type of CD-ROM drive).
csParam[1-10]	Reserved.

Output Produced:

csParam[0]	A code indicating the type of CD-ROM drive located at the specified device ID. These codes are defined as follows: 1 -- Drive is an AppleCD SC 2 -- Drive is an AppleCD SC Plus / AppleCD 150 3 -- Drive is an AppleCD 300 or newer (multiple speed drives)

Status Return Codes:

noErr	The drive type for the specified device ID was returned.
statusErr	The specified device ID is not a CD-ROM drive.

GetBlockSize

The purpose of this call is to return to the caller the current block size of the CD-ROM drive.

Note: If the user has set the block size to 512, but the driver has set the actual block size to 2048 and does the 2048-byte to 512-byte conversion, this call will return 512 as the current block size.

Input Parameters:

`cscode`	98
`csParam[0]`	Reserved (on exit, it contains the block size).
`csParam[1-10]`	Reserved.

Output Produced:

`csParam[0]`	The block size of the disc inserted in the specified device ID.

Status Return Codes:

`noErr`	The block size of the disc in the specified device ID was returned.

ReturnDeviceIdent

The purpose of this call is to return the `DeviceIdent` for the CD-ROM drive being controlled by this instance (entry in the unit table) of the driver. This call was first available in driver version 5.0.

The following is the format of the `DeviceIdent`:

```
struct DeviceIdent {
  UInt8      diReserved;     /* reserved               */
  UInt8      bus;            /* SCSI - Bus Number      */
  UInt8      targetID;       /* SCSI - Target SCSI ID  */
  UInt8      LUN;            /* SCSI - LUN             */
};
typedef struct DeviceIdent DeviceIdent;
```

Input Parameters:

`cscode`	120
`csParam[0-1]`	Reserved (`DeviceIdent` will be returned here).
`csParam[2-10]`	Reserved.

Output Produced:

`csParam[0-1]`:	The `DeviceIdent`.

Status Return Codes:

`noErr`	The `DeviceIdent` was returned.

GetCDFeatures

The purpose of this call is to determine the features of a particular CD-ROM drive. Prior to driver version 5.0, this was done using the `GetDriveType` status call, which returned a specific model of CD-ROM drive.

This makes client code difficult to maintain since it must be modified each time a new CD-ROM drive is introduced. To alleviate this problem, the features of the device have been broken out into testable bits.

A fixed point number containing the sustained transfer rate of the drive relative to an AppleCD 150 is also included (e.g., AppleCD 300 Plus = 2.0, AppleCD 150 = 1.0).

This information is returned in the following structure:

```
struct CDDeviceCharacteristics
{
    unsigned char speedMajor;   // High byte of fixed point speed
    unsigned char speedMinor;   // Low byte of fixed point speed.
    unsignedshort cdFlags;      // Flags for features of this drive
};
typedef struct CDDeviceCharacteristics CDDeviceCharacteristics;

enum /* Flags for CD Features field */
{
    cdPowerInject       = 0,    // device supports power inject of media
    cdNotPowerEject     = 1,    // device does not support power eject of media
    cdMute              = 2,    // device supports audio channels muting
    cdLeftToChannel     = 3,    // device supports left channel only mono audio
    cdRightToChannel    = 4,    // device supports right channel only mono audio
    cdLeftPlusRight     = 5,    // device supports left + right channels mono audio
    cdSCSI_2            = 10,   // device supports SCSI2 command set (SCSI only)
    cdStereoVolume      = 11,   // device supports independent volume per channel
    cdDisconnect        = 12,   // device supports disconnect / reconnect (SCSI only)
    cdWriteOnce         = 13,   // device is a write-once type of drive

    cdPowerInject_Mask    = 1 << cdPowerInject,
    cdNotPowerEject_Mask  = 1 << cdNotPowerEject,
    cdMute_Mask           = 1 << cdMute,
    cdLeftToChannel_Mask  = 1 << cdLeftToChannel,
    cdRightToChannel_Mask = 1 << cdRightToChannel,
    cdLeftPlusRight_Mask  = 1 << cdLeftPlusRight,
    cdSCSI_2_Mask         = 1 << cdSCSI_2,
    cdStereoVolume_Mask   = 1 << cdStereoVolume,
    cdDisconnect_Mask     = 1 << cdDisconnect,
    cdWriteOnce_Mask      = 1 << cdWriteOnce
};
```

Input Parameters:

`cscode`	121
`csParam[0-1]`	Reserved (CD-ROM drive features will be returned here).
`csParam[2-10]`	Reserved.

Output Produced:

`csParam[0-1]`:	The CD-ROM drive features as defined above.

Status Return Codes:

`noErr`	The CD-ROM drive features were returned.

Further References

- *AppleCD SC Developer's Guide, Second Edition* (APDA A7G0023/C)
- *Apple CD-ROM Handbook* (Addison-Wesley)
- *Breaking Through: A Technical Guide for the Design and Development of CD-ROMs* (Apple Multimedia Program - 030-0879-A)
- CD-ROM and the Macintosh Computer, on the Developer CD series
- *Inside Macintosh: Devices*
- *ISO 10149 Information technology - Data interchange on read-only 120 mm optical data disks* (CD-ROM) ("Yellow Book")
- *CEI IEC 908 Compact Disc digital audio system* ("Red Book")

Downloadables

 Acrobat version of this Technote (95K).

To contact us, please use the Contact Us page.

Updated: 5-April-99

Technotes
Previous Technote | Contents | Next Technote