Microsoft MS-DOS CD-ROM Extensions

Version 2.21

Hardware-Dependent Device Driver Specification

This document describes the CD-ROM hardware-dependent device driver and its interface with MSCDEX.EXE, the MS-DOS CD-ROM Extensions resident program. Differences between CD-ROM drives and hard- or floppy-disk drives account for the differences in this device driver specification from the normal MS-DOS block and character device driver specification. For more information on device drivers, see the *MS-DOS Programmer's Reference Manual* and *Advanced MS-DOS Programming* by Ray Duncan. (Both books are published by Microsoft Press.) For information on the terms describing for CD-ROM operating characteristics, see the IEC 908 and ISO 10149 specifications.

The MS-DOS operating system reads CONFIG.SYS and installs the device. MSCDEX.EXE performs an open system call on the device driver name in order to communicate with it and uses an IOCTL call to ask the device driver for the address of its device header. From the device header address, MSCDEX.EXE locates the device driver's interrupt and strategy routines. After that, all requests the device driver receives come directly from MSCDEX.EXE, not MS-DOS. To avoid reentrancy problems and allow MSCDEX to monitor all media changes, all other applications that wish to communicate directly with CD-ROM device drivers should do so through the *Send Device Driver Request* INT 2Fh function 1510h. MSCDEX.EXE interfaces with MS-DOS so that normal requests for I/O with files on a CD-ROM drive down to the MS-DOS INT 21h service layer will work just as they would for a normal MS-DOS device.

For maximum compatibility between your CD-ROM device driver, MSCDEX, and other components in the system, observe the following guidelines when developing your device driver:

- Avoid disabling interrupts for extended periods.
- CD-ROM drives and drivers that use DMA must use Windows virtual DMA device services for 386 enhanced mode operation. For information for using Windows virtual DMA device services, see the *Microsoft Windows Device Driver Adaptation Guide* included in the Microsoft Windows Device Development Kit.

Installation

The device driver will be installed in the same way as any other device with an entry in CONFIG.SYS. The syntax is:

DEVICE=<*filename*> /D:<*device_name*> /N:<number of drives>

The following are examples:

DEVICE=HITACHI.SYS /D:MSCD001 /D:MSCD002
DEVICE=SONY.SYS /D:MSCD003 /N:2

The arguments will be the character device names that will be used on the command line when starting MSCDEX.EXE so that it can find and communicate with the device driver.

A device driver may support one or more physical drives or logical disks. This may be done by having multiple device headers in the device driver file (in which case it will be necessary to have more than one *device_name* on the command line - one for each device header; see the HITACHI.SYS example above) or through the use of subunits. Each disk handled by a device

driver that supports multiple disks using subunits is addressed by the subunit field of the request header when a request is made for that disk. A device driver that supports more than one disk can share code and data instead of requiring separate device drivers for each disk. A "jukebox" CD-ROM system would be an example of a CD-ROM device that might wish to support more than one drive or a disk pack using a single device driver.
Device drivers that use multiple subunits should use the optional switch /n:<number of drives> to say how many drives are present. If not present, the default number of drives is 1. If the driver can tell how many drives are installed without a command line switch, then this argument is not necessary. Unless there are special considerations, it is better practice to support multiple drives using subunits than to have multiple device headers in the same device driver file.

Device header

The device header is an extension to what is described in the *MS-DOS Programmer's Refrence Manual*.

```
DevHdr      DD      -1          ; Ptr to next driver in file or -1 if last driver
            DW      ?                   ; Device attributes
            DW      ?                   ; Device strategy entry point
            DW      ?                   ; Device interrupt entry point
            DB      8 dup (?)           ; Character device name field
            DW      0                   ; Reserved
            DB      0                   ; Drive letter
            DB      ?                   ; Number of units
```

The following are the device attributes for MSCDEX.EXE device drivers:

```
    Bit 15          1     - Character device
    Bit 14          1     - IOCTL supported
    Bit 13          0     - Output 'till  busy
    Bit 12          0     - Reserved
    Bit 11          1     - OPEN/CLOSE/RM supported
    Bit 10-4 0      - Reserved
    Bit 3           0     - Dev is CLOCK
    Bit 2           0     - Dev is NUL
    Bit 1           0     - Dev is STO
    Bit 0           0     - Dev is STI
```

MSCDEX.EXE device drivers will be character devices that understand IOCTL calls and handle OPEN/CLOSE/RM calls.
The drive letter field is a read-only field for the device driver and is initialized to 0. The field is for MSCDEX.EXE to use when it assigns the device driver to a drive letter (A = 1, B = 2...Z = 26). It should never be modified by the device driver. For drivers that support more than one unit, the drive letter will indicate the first unit, and each successive unit is assigned the next higher drive letter. For example, if the device driver has four units defined (0-3), it requires four drive letters. The position of the driver in the list of all drivers determines which units correspond to which drive letters. If driver ALPHA is the first driver in the device list, and it defines 4 units (0-3), they will be A, B, C, and D. If BETA is the second driver and defines three units (0-2), they will be E, F, and G, and so on. The theoretical limit to the number of drive letters is 63, but it should be noted that the device installation code will not allow the installation of a device if it would result in a drive letter > 'Z' (5Ah). All block device drivers present in the standard resident BIOS will be placed ahead of installable device drivers in the list.
NOTE: It is important that one set *lastdrive=<letter>* in CONFIG.SYS to accommodate the additional drive letters that CD-ROM device drivers will require.
The number-of-units field is set by the device driver to the number of disks that are supported. Normal character devices do not support more than one unit and MS-DOS does not expect a character device to handle more than one unit or have a nonzero subunit value in the request header. Since these device drivers are not called by MS-DOS directly, this is not a problem.

Nonetheless, the number of units returned by the device driver in the number-of-units field during the INIT call must be 0, since MS-DOS makes the INIT call and does not expect a nonzero value for a character device. MSCDEX.EXE will never see what is returned anyway, and relies on the number-of-units field in the device header.
The signature field is necessary for MSCDEX confirmation that the device driver is a valid cd-rom device driver and consists of the 4 bytes 'MSCD' followed by two ascii digits for the version which is '00' at present.
Sample device header:

```
HsgDrv  DD      -1              ; Pointer to next device
        DW      0c800h          ; Device attributes
        DW      STRATEGY        ; Pointer to device strategy routine
        DW      DEVINT          ; Pointer to device interrupt routine
        DB      'MSCD003 '      ; 8-byte character device name field
        DW      0               ; Reserved (must be zero)
        DB      0               ; Drive letter (must be zero)
        DB      1               ; Number of units supported
                                ; (one or more)
```

As with other MS-DOS device drivers, the code originates at offset 0, not 100H. The first device header will be at offset 0 of the code segment. The pointer to the next driver is a double word field (offset/segment) that is the address of the next device driver in the list, or -1 if the device header is the only one or the last in the list. The strategy and interrupt entry points are word fields and must be offsets into the same segment as the device header. The device driver is expected to overwrite the name(s) in each of its one or more device headers with the <*device_name*> command line arguments during its initialization.
MSCDEX.EXE will call the device driver in the following manner:

1. MSCDEX.EXE makes a far call to the strategy entry.
2. MSCDEX.EXE passes device driver information in a request header to the strategy routine.
3. MSCDEX.EXE makes a far call to the interrupt entry.

Request header

MSCDEX.EXE will call the device's strategy routine with the address of a request header in ES:BX. The format of the request header is the same as what is described in the *MS-DOS Programmer's Reference Manual.*

```
ReqHdr  DB      ?               ; Length in bytes of request header
        DB      ?               ; Subunit code for minor devices
        DB      ?               ; Command code field
        DW      ?               ; Status
        DB      8 dup (?)       ; Reserved
```

Status
The status word also has the same format as described in the *MS-DOS Programmer's Reference Manual*. It is 0 on entry and is set by the device driver.

Bit 15 - Error bit
Bit 14-10 - Reserved
Bit 9 - Busy

Bit 8 - Done
Bit 7-0 - Error code (bit 15 on)

Bit 15, the error bit, is set by the device driver if an error is detected or if an invalid request is made to the driver. The low 8 bits indicate the error code.
Bit 9, the busy bit, should be set by the device driver when the drive is in audio play mode. Device drivers should fail all requests to the physical device that require head movement when the device is playing and return the request with this bit and the error bit set and an error code. Requests that would not interrupt audio play may return without error but will also have this bit set when the drive is in audio play mode. Play mode can be terminated prematurely with a reset or STOP AUDIO request and a new request can be made at that point. Monitoring this bit in each successive request, an Audio Q-Channel Info IOCTL for example, will tell when play mode is complete.
Bit 8, the done bit, is set by the device driver when the operation is finished.
Error codes include the following:

0	Write-protect violation
1	Unknown unit
2	Drive not ready
3	Unknown command
4	CRC error
5	Bad drive request structure length
6	Seek error
7	Unknown media
8	Sector not found
9	Printer out of paper
A	Write fault
B	Read fault
C	General failure
D	Reserved
E	Reserved
F	Invalid disk change

While MS-DOS reserves D as a general error code, the audio PLAY command uses it for a "PARAMETER OUT OF RANGE" error.

Command code field

The following values are valid command codes:

*	0	INIT
	1	MEDIA CHECK (block devices)
	2	BUILD BPB (block devices)
*	3	IOCTL INPUT
	4	INPUT (read)
	5	NONDESTRUCTIVE INPUT NO WAIT
	6	INPUT STATUS
*	7	INPUT FLUSH
	8	OUTPUT (write)
	9	OUTPUT WITH VERIFY
	10	OUTPUT STATUS
#	11	OUTPUT FLUSH
*	12	IOCTL OUTPUT
*	13	DEVICE OPEN
*	14	DEVICE CLOSE
	15	REMOVABLE MEDIA (block devices)
	16	OUTPUT UNTIL BUSY
*	128	READ LONG
	129	Reserved

*	130	READ LONG PREFETCH
*	131	SEEK
+	132	PLAY AUDIO
+	133	STOP AUDIO
#	134	WRITE LONG
#	135	WRITE LONG VERIFY
+	136	RESUME AUDIO

* Supported by a basic CD-ROM device driver (required)
+ Supported by an extended CD-ROM device driver (required for multimedia driver)
See also IOCTL INPUT
Supported by writable CD-ROM device drivers for authoring systems

Unsupported or illegal commands will set the error bit and return the error code for *Unknown Command.* This includes command codes 1, 2, 4, 5, 6, 8, 9, 10, 15, 16, and 129; and 11, 134 and 135 for systems that do not support writing.

If, in the time since the last request to that device driver unit, the media has changed, the device driver will return the error code for invalid disk change and set the error bit. MSCDEX.EXE will then decide whether to retry the request or abort it.

The minimal CD-ROM device driver will read cooked Mode 1 data sectors using HSG addressing mode and return appropriate values for the IOCTL calls. Most other features enhance performance or add useful capabilities.

INIT

Command code = 0
ES:BX = INIT

```
INIT    DB    13 dup (0)    ; Request header
        DB    0             ; Number of units (must be 0)
        DD    ?             ; End address
        DD    ?             ; Ptr to BPB array
        DB    0             ; Block device number
```

This call is made only once, when the device is installed. INIT and a single IOCTL call for the device header address are the only device driver calls that come directly from MS-DOS. Because the INIT function is called from MS-DOS, the number of units returned is 0, as for normal MS-DOS character devices. MSCDEX.EXE will get the number of units supported from the device header.

The device must return the END ADDRESS, which is a DWORD pointer to the end of the portion of the device driver to remain resident. Code and data following the pointer is used for initialization and then discarded. (The device driver should discard this code to reduce its resident size.) If there are multiple device drivers in a single file, the ending address returned by the last INIT call will be the one that MS-DOS uses, but it is recommended that all the device drivers in the file return the same address. The code to remain resident for all the devices in a single file should be grouped together low in memory with the initialization code for all devices following it in memory.

The pointer to BPB array points to the character after the "=" on the line in CONFIG.SYS that caused this device driver to be loaded. This data is read-only and allows the device driver to scan the invocation line for parameters. This line is terminated by a carriage return or a line feed.

During initialization, the device driver must set the device name field in the device header to the argument provided on the invocation line in CONFIG.SYS. The device driver must also check that the *device_name* command line argument is a legal 8-character filename and pad it out to 8 characters with spaces (20H) when copying it to the device name field.

The block device number and number of units are both 0 for character devices.

The device driver should return the error code for *Unknown Command* for subsequent calls to INIT.

READ (IOCTL Input)

Command code = 3
ES:BX = IOCTLI

```
IOCTLI  DB    13 dup (0)    ; Request header
        DB    0             ; Media descriptor byte from BPB
        DD    ?             ; Transfer address
        DW    ?             ; Number of bytes to transfer
        DW    0             ; Starting sector number
        DD    0             ; DWORD ptr to requested vol
                            ; ID if error 0FH
```

The media descriptor byte, starting sector number, and volume ID fields are all 0.
The transfer address points to a control block that is used to communicate with the device driver. (The types of control blocks are listed in the following table.) The first byte of the control block determines the request that is being made. If the command code is reserved or the function not supported, then the device driver will return the error code for *Unknown Command*. If, for some reason, the device driver is not able to process the request at that time, it will return the error code for *Drive Not Ready*.

	Code	Number of bytes to transfer	Function
	0	5	Return Address of Device Header
*	1	6	Location of Head
	2	?	Reserved
	3	?	Error Statistics
*	4	9	Audio Channel Info
	5	130	Read Drive Bytes
	6	5	Device Status
	7	4	Return Sector Size
	8	5	Return Volume Size
	9	2	Media Changed
*	10	7	Audio Disk Info
*	11	7	Audio Track Info
*	12	11	Audio Q-Channel Info
*	13	13	Audio Sub-Channel Info
*	14	11	UPC Code
*	15	11	Audio Status Info
	16-255	?	Reserved

* Required only for extended/multimedia drivers.

- Return Address of Device Header

```
Raddr   DB    0             ; Control block code
        DD    ?             ; Address of device header
```

The device driver will fill the 4-byte field with the address of its device header. This is used by MSCDEX.EXE to locate the device driver's strategy and interrupt routines.

- Location of Head

```
LocHead DB    1             ; Control block code
        DB    ?             ; Addressing mode
        DD    ?             ; Location of drive head
```

The device driver will return a 4-byte address that indicates where the head is located. The value will be interpreted based on the addressing mode. (See function READ LONG for more

information about addressing modes.)
NOTE: the drive could provide this information by monitoring the Q-channel on the disc.

- Error Statistics

```
ErrStat     DB      3               ; Control block code
            DB      N dup (?)       ; Error statistics
```

The format of the Error Statistics is not defined.

- Audio Channel Info

```
AudInfo     DB      4               ; Control block code
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 0
            DB      ?               ; Volume control (0 - 0xff) for output channel 0
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 1
            DB      ?               ; Volume control (0 - 0xff) for output channel 1
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 2
            DB      ?               ; Volume control (0 - 0xff) for output channel 2
            DB      ?               ; Input channel (0, 1, 2, or 3) for output
                                    ; channel 3
            DB      ?               ; Volume control (0 - 0xff) for output channel 3
```

This function returns the present settings of the audio channel control set with the Audio Channel Control Ioctl Write function. The default settings for the audio channel control are for each input channel to be assigned to its corresponding output channel (0 to 0, 1 to 1, etc.) and for the volume control on each channel is set at 0xff.

- Read Drive Bytes

```
DrvBytes    DB      5               ; Control block code
            DB      ?               ; Number bytes read
            DB      128 dup (?)     ; Read buffer
```

Data returned from the CD-ROM drive itself can be read using this function. The number-bytes-read field returns the length of the number of bytes read, which will not exceed 128 per call. If more than this needs to be returned, the application should repeat the call until the number returned is less than 128.

The function and content of these bytes are entirely device and device driver dependent. This function is provided to allow access to device-specific features that are not addressed under any other portion of the device driver spec.

- Device Status

```
DevStat     DB      6               ; Control block code
            DD      ?               ; Device parameters
```

The device driver will return a 32-bit value. Bit 0 is the least significant bit. The bits are interpreted as follows:

Bit 0	0	Door closed
	1	Door open
Bit 1	0	Door locked
	1	Door unlocked
Bit 2	0	Supports only cooked reading
	1	Supports cooked and raw reading

Bit 3	0	Read only
	1	Read/write
Bit 4	0	Data read only
	1	Data read and plays audio/video tracks
Bit 5	0	No interleaving
	1	Supports ISO-9660 interleaving using interleave size and skip factor
Bit 6	0	Reserved
Bit 7	0	No prefetching
	1	Supports prefetching requests
Bit 8	0	No audio channel manipulation
	1	Supports audio channel manipulation
Bit 9	0	Supports HSG addressing mode
	1	Supports HSG and Redbook addressing modes
Bit 10	0	Reserved
Bit 11	0	Disc is present in drive
	1	No disc is present in drive
Bit 12	0	Doesn't support R-W sub-channels
	1	Supports R-W sub-channels
Bit 13-31	0	Reserved (all 0)

- Return Sector Size

```
SectSize    DB    7        ; Control block code
            DB    ?        ; Read mode
            DW    ?        ; Sector size
```

The device driver will return the sector size of the device given the read mode provided. In the case of CD-ROM, the value returned for cooked is 2048, and the return value for raw is 2352.

- Return Volume Size

```
VolSize     DB    8        ; Control block code
            DD    ?        ; Volume size
```

The device driver will return the number of sectors on the device. The size returned is the address of the lead-out track in the TOC converted to a binary value according to FRAME + (SEC * 75) + (MIN * 60 * 75). A disc with a lead out track starting at 31:14.63 would return a volume size of 140613. The address of the lead-out track is assumed to point to the first sector following the last addressable sector recorded on the disc.

- Media Changed

```
MedChng     DB    9        ; Control block code
            DB    ?        ; Media byte
```

The normal media check function (command code 1) is not performed on character devices and contains additional semantics that are not needed for CD-ROM device drivers. This is why there is an IOCTL request for this function.

When the device driver receives a call to see if the media has changed on that subunit since the last call to this IOCTL, it will return one of the following values:

1	Media not changed
0	Don't know if changed
-1 (0FFh)	Media changed

If the driver can assure that the media has not been changed (through a door-lock or other interlock mechanism), performance is enhanced because MSCDEX.EXE does not need to reread the Volume Descriptor and invalidate in-memory buffers for each directory access. For drives

that do not report if the media has changed, CD-ROM device drivers can utilize the same solution that has been applied to floppy disks. In some floppy-disk device drivers, if the MEDIA CHECK occurs within 2 seconds of a floppy-disk access, the driver reports "Media not changed." It is highly recommended though that drives be able to detect and report media changes.
If the drive can enforce a door lock mechanism so that the device driver is notified when the door lock has been unlocked or the device driver is requested to do so by MSCDEX.EXE, then to improve performance, the driver could return that the media has not changed without bothering to communicate with the physical device.
If the media has not been changed, MSCDEX.EXE will proceed with the disk access. If the value returned is "Don't know," or "Media changed," then MSCDEX.EXE will check to see if the disk has changed. It will continue if it has not, and reinitialize what it knows about the disk if it has. Multimedia drivers assume that after a MEDIA CHECK cached TOC (Table of Contents) information is correct.

- Audio Disk Info

```
DiskInfo    DB    10          ; Control block code
            DB    ?           ; Lowest track number
            DB    ?           ; Highest track number
            DD    ?           ; Starting point of the lead-out track
```

This function returns TOC (Table of Contents) information from the Q-Channel in the lead-in track indicating what the first and last track numbers are and the Redbook address for the lead-out track (PMIN/PSEC/PFRAME when POINT = A2). The first and last track numbers are binary values and not BCD. It is recommended that the information for Audio Disk Info and Audio Track Info should be read by the drive when the disc is initialized and made accessible to the driver so that when these functions are called, the drive or driver do not have to interrupt audio play to read them from the TOC. If the TOC is not made available to the driver and the driver must obtain the information itself from the lead-in track, the driver should read and and attempt to cache the disk and track information during the Audio Disk Info command and invalidate this information only if the media changes. Note that the lowest and highest track numbers do not include the lead-in or lead-out tracks.

- Audio Track Info

```
TnoInfo     DB    11          ; Control block code
            DB    ?           ; Track number
            DD    ?           ; Starting point of the track
            DB    ?           ; Track control information
```

This function takes a binary track number, from within the range specified by the lowest and highest track number given by the Audio Disk Info command, and returns the Redbook address for the starting point of the track and the track control information for that track. The track control information byte corresponds to the byte in the TOC in the lead-in track containing the two 4-bit fields for CONTROL and ADR in the entry for that track. The CONTROL information is in the most significant 4 bits and the ADR information is in the lower 4 bits. The track control information is encoded as follows:

```
00x00000    - 2 audio channels without pre-emphasis
00x10000    - 2 audio channels with pre-emphasis
10x00000    - 4 audio channels without pre-emphasis
10x10000    - 4 audio channels with pre-emphasis
01x00000    - data track
01x10000    - reserved
11xx0000    - reserved
xx0x0000    - digital copy prohibited
xx1x0000    - digital copy permitted
```

- Audio Q-Channel Info

```
QInfo   DB    12      ; Control block code
        DB    ?       ; CONTROL and ADR byte
        DB    ?       ; Track number (TNO)
        DB    ?       ; (POINT) or Index (X)
                      ; Running time within a track
        DB    ?       ; (MIN)
        DB    ?       ; (SEC)
        DB    ?       ; (FRAME)
        DB    ?       ; (ZERO)
                      ; Running time on the disk
        DB    ?       ; (AMIN) or (PMIN)
        DB    ?       ; (ASEC) or (PSEC)
        DB    ?       ; (AFRAME) or (PFRAME)
```

This function reads and returns the most up to date Q-channel address presently available. It must not interrupt the present status of the drive as one of its intended purposes is to monitor the location of the read head while playing audio tracks. This function should return valid information even when no audio tracks are being played and the head is stationary. The fields returned correspond to the data that is stored in the Q-channel as described in the Redbook. The values in MIN-SEC-FRAME, AMIN-ASEC-AFRAME and PMIN-PSEC-PFRAME are converted by the driver from BCD to binary so that minutes range from 0 to 59+, seconds from 0 to 59, and frames from 0 to 74. The Control and ADR byte, TNO, and POINT/Index bytes are always passed through as they appear on the disc and are not converted. If the drive returns Q-channel information when ADR is not equal to 1, then when ADR is not equal to 1 all ten bytes of information are passed through unmodified to the caller.

Audio Sub-Channel Info

```
        DB    13      ; Control block code
        DD    ?       ; Starting sector address
        DD    ?       ; Transfer address
        DD    ?       ; Number of sectors to read
```

This function takes a Redbook address of a particular sector (also known as a block or frame) and copies the sub-channel information for each sector requested to the transfer address. If multiple sectors are requested, they are copied sequentially. Each sector contains 96 bytes of sub-channel information. These bytes do not include the two sync patterns (S0 and S1) that head the subcoding block. They contain only the subcoding symbols--each with one bit of information for the eight different channels (P-W) that follow them. In this byte, the 2 most significant bits that represent channels P and Q are undefined, the next most significant bit (bit 6) represents channel R, and the least significant bit (bit 1) represents channel W.

For the sub-channel data to be generally useful it *must* be provided during an AUDIO PLAY operation without interrupting it. Therefore, when the requested sectors lie within the current play request they should be provided in real-time. This requires data buffering (in the drive or by the driver) for at least one sector of sub-channel information. This is because an application using sub-channel information will send an AUDIO PLAY command, follow it up with successive and contiguous sub-channel information commands, and expect to get that data starting with the first sector of the play request. It is recommended that the buffer handle up to 32 sectors, to give an application time to process sub-channel data between requests.

Since implementation of this command is optional, bit 12 in the device status long word should indicate whether sub-channel support is available. The driver will return an error code of *Unknown Command* if it is not supported.

The method of data error detection and correction for R-W channels is specified in the Redbook

standard, and if not implemented by the drive, must be provided in the driver software.

- UPC Code

```
UPCCode   DB    14          ; Control block code
          DB    ?           ; CONTROL and ADR byte
          DB    7 dup (?)   ; UPC/EAN code
                            ; (last 4 bits are zero; the low-order
                            ; nibble of byte 7)
          DB    ?           ; Zero
          DB    ?           ; Aframe
```

This function returns the UPC/EAN (Universal Product Code - BAR coding) for the disc. This information is stored as a mode-2 (ADR=2) Q-channel entry. The UPC code is 13 successive BCD digits (4 bits each) followed by 12 bits of zero. The last byte is the continuation of FRAME in mode-1 though in the lead-in track (TNO=0) this byte is zero. If the CONTROL/ADR byte is zero or if the 13 digits of UPC code are all zero, then either no catalog number was encoded on the disc or it was missed by the device driver. If the command is supported but the disc does not have a UPC Code recorded, then the driver will return an error code of *Sector not Found*. If the command is not supported, then the driver will return an error code of *Unknown Command.*(ISRC is currently unsupported.)

- Audio Status Info

```
AudStat   DB    15          ; Control block code
          DW    ?           ; Audio status bits
                            ; Bit 0 is Audio Paused bit
                            ; Bits 1-15 are reserved
          DD    ?           ; Starting location of last Play or for next Resume
          DD    ?           ; Ending location for last Play or for next Resume
```

The Audio Paused bit and Starting and Ending locations are those referred to in the RESUME command. These are recorded in Redbook addressing mode.

WRITE (IOCTL OUTPUT)

Command code = 12
ES:BX = IOCTLO

```
IOCTLO    DB    13 dup (0)  ; Request header
          DB    0           ; Media descriptor byte from BPB
          DD    ?           ; Transfer address
          DW    ?           ; Number of bytes to transfer
          DW    0           ; Starting sector number
          DD    0           ; DWORD ptr to requested vol
                            ; ID if error 0FH
```

The media descriptor byte, starting sector number, and volume ID fields are all 0.
The transfer address points to a control block that is used to communicate with the device driver. The first byte of the control block determines the request that is being made. The Length of Block is the number of bytes to transfer. With the exception of Reset Drive and Write Device Control String, the action of these request may be verified with the READ IOCTL Audio Channel Info or Device Status functions. If the command is not supported, then the driver will return an error code of *Unknown Command.*

Code	Length of Block	Function
0	1	Eject Disk

1	2	Lock/Unlock Door
2	1	Reset Drive
3	9	Audio Channel Control
4	?	Write Device Control String
5	1	Close Tray
6-255	?	Reserved

- Eject Disk

```
Eject           DB      0           ; Control block code
```

The device driver will unlock the drive and eject the CD-ROM disk from the drive unit. The door will report as being open until the user has inserted a disk into the drive unit and closed the door. The status bit for door open can be monitored to determine when a disk has been reinserted.

- Lock/Unlock Door

```
LockDoor    DB      1           ; Control block code
            DB      ?           ; Lock function
```

When this function is received, the device driver will ask the CD-ROM drive to unlock or lock the door. If lock function is 0, the device driver will unlock the door. If lock function is 1, it will lock the door.

- Reset Drive

```
ResetDrv    DB      2           ; Control block code
```

This function directs the device driver to reset and reinitialize the drive.

- Audio Channel Control

```
AudInfo     DB      3           ; Control block code
            DB      ?           ; Input channel (0, 1, 2, or 3) for output
                                ; channel 0
            DB      ?           ; Volume control (0 - 0xff) for output channel 0
            DB      ?           ; Input channel (0, 1, 2, or 3) for output
                                ; channel 1
            DB      ?           ; Volume control (0 - 0xff) for output channel 1
            DB      ?           ; Input channel (0, 1, 2, or 3) for output
                                ; channel 2
            DB      ?           ; Volume control (0 - 0xff) for output channel 2
            DB      ?           ; Input channel (0, 1, 2, or 3) for output
                                ; channel 3
            DB      ?           ; Volume control (0 - 0xff) for output channel 3
```

This function is intended to provide playback control of audio information on the disk. It allows input channels on the CD-ROM to be assigned to specific output speaker connections. The purpose of this function is to allow two independent channels to be recorded - in different languages for example - and to play back only one of them at a time or to be able to manipulate an audio signal so that the source appears to move - to make a sound seem to move from left to right for example.

Output channel 0 is the left channel, 1 is right, 2 is left prime, and 3 is right prime. The Redbook specification allows for 4 audio channels. The two "prime" channels (2 and 3) extend stereo to quadrophonic stereo.

An audio volume setting of 0 means off. Drives that don't support 4 output audio channels may ignore output to channels 2 and 3. Assignment of input channels 2 and 3 to output channels 0 and

1 may be treated as though the volume control for that channel is 0.
Drives that do not support variable audio control will treat a setting of 0 as off and 1-0xff as on. Drives that support less than 256 volume settings will do their best to break up the 256 settings among the settings they can support. For example, if there are 16 settings supported, then the first setting will cover 0x01-0x10, the second 0x11-0x20...the sixteenth 0xf1-0xff. Drives that can't play a single channel in both must play only that one channel and try to suppress the other if possible. Drives that can't swap channels should play the channel that was moved in its normal channel.

- Write Device Control String

```
_____________________
DrvBytes    DB      4               ; Control block code
            DB      N dup (?)       ; Write buffer
_____________________
```

This function is provided to allow programs to talk directly to the CD-ROM drive. All remaining bytes are sent uninterpreted to the drive unit.
The function and content of these bytes are entirely device and device driver dependent. This function is provided to allow access to device-specific features that are not addressed under any other portion of the device driver spec.

- Close Tray

```
_____________________
CloseTray   DB      5               ; Control block code
_____________________
```

This command is the logical complement to the Eject Disk command. This command will instruct drives that can do so to close the door or tray.

READ LONG

Command code = 128
ES:BX = ReadL

```
_____________________
ReadL       DB      13 dup (0)      ; Request header
            DB      ?               ; Addressing mode
            DD      ?               ; Transfer address
            DW      ?               ; Number of sectors to read
            DD      ?               ; Starting sector number
            DB      ?               ; Data read mode
            DB      ?               ; Interleave size
            DB      ?               ; Interleave skip factor
_____________________
```

The request block is different from a normal character device READ to accommodate the larger size and different characteristics of CD-ROM devices.
The media descriptor byte, which has no meaning for character devices, is now the addressing mode field. The following values are recognized addressing modes:

0	HSG addressing mode
1	Redbook addressing mode
2-255	Reserved

The default addressing mode is the HSG addressing mode. Long (DWORD) address values are treated as logical block numbers, as defined by the High Sierra proposal. When Redbook addressing mode is on, all disk addresses are interpreted as Minute/Second/Frame addresses, according to the Philips/Sony Redbook standard. Each of these fields is 1 byte. The least significant byte of the address field contains data for frames, the second least significant byte contains data for seconds, the third least significant byte contains data for minutes, and the most significant byte of the 4-byte field is unused. These values are represented in binary rather than in BCD format. For example, if we are referencing the sector addressed by minute 36, second 24, frame 12, the hex long value for this would be 0x0024180C. This 4 byte value is recorded in the starting sector number field with the least significant byte first and most significant byte last.

The relationship between High Sierra sectors and Redbook frames is described by the equation:
Sector = Minute * 60 * 75 + Second * 75 + Frame - 150
The byte/sector count field becomes the number of sectors to read and the starting sector number expands from one word to two, which means we can address up to 4 giga-sectors (over 8 terabytes). The DWORD ptr for requested volume ID is eliminated and MSCDEX.EXE will keep track of what volume is needed.
MSCDEX.EXE handles buffering requests, but performance may be improved if the device driver reads ahead or uses a sector caching scheme, given the slow seek times of CD-ROM drives. The operating system will use the prefetch function when it can to give hints to the driver.
The data read mode field will be one of the following:

0	Cooked mode
1	Raw mode
2-255	Reserved

Cooked mode is the default mode in which the hardware typically handles the EDC/ECC and the device driver returns 2048 bytes of data per sector read. When raw mode is set, the driver will return all 2352 bytes of user data, including any EDC/ECC present independent of the actual sector mode (Mode 2 Form 1 vs. Mode 2 Form 2). User programs will have to consider this and allow enough room for buffer space when reading in raw mode as each sector returned will take up 2352 bytes of space. Drives that cannot return all 2352 bytes will return what they can and leave blank what they cannot. For example, drives that can return all 2336 bytes except the 16 byte header will leave a space in the first 16 bytes where the header would go so that the sectors align on 2352 byte boundaries. Drivers should do what they can to return as much of the user data per sector as possible.
The two interleave parameters are for drivers that support interleaved reading. If the driver does not support interleaving, these fields are both ignored. If it does, interleave size is the number of consecutive logical blocks or sectors that are stored sequentially, and the interleave skip factor is the number of consecutive logical blocks or sectors that separate portions of the interleaved file.

READ LONG PREFETCH

Command code = 130
ES:BX = ReadLPre

```
ReadLPre    DB      13 dup (0)      ; Request header
            DB      ?               ; Addressing mode
            DD      0               ; Transfer address
            DW      ?               ; Number of sectors to read
            DD      ?               ; Starting sector number
            DB      ?               ; Read mode
            DB      ?               ; Interleave size
            DB      ?               ; Interleave skip factor
```

This function is similar in form to READ LONG, but control returns immediately to the requesting process. The device driver is not obligated to read in the requested sectors but can instead consider the request for these sectors as hints from the operating system that they are likely to be needed. It is recommended that at a minimum, the driver seek to the location provided. The attribute in the device status for prefetching is used to distinguish drivers that do more than just seek to the given location. The requests are low priority and preemptible by other requests for service. A READ LONG PREFETCH with 0 number of sectors to read should be treated as an advisory seek, and the driver can, if it is not busy, move the head to the starting sector. Since prefetching requests are advisory, there will be no functional difference between a device driver that supports prefetching from one that does not, except in terms of performance. The transfer address is not applicable for this call as the driver is not meant to transfer any data into the user address space.

SEEK

Command code = 131

ES:BX = SeekReq

```
SeekReq     DB      13 dup (0)      ; Request header
            DB      ?               ; Addressing mode
            DD      0               ; Transfer address
            DW      0               ; Number of sectors to read
            DD      ?               ; Starting sector number
```

Control returns immediately to the caller without blocking and waiting for the seek to be completed. The number of sectors to be read and the transfer address are ignored. SEEK is used to relocate the head in order to begin playing audio or video tracks, or in anticipation of reading in a particular region on the disk. Further requests for disk activity will wait until the given SEEK is completed. This seek is not advisory and the head must move to the desired location. If the address is not within the range of the disc, return a "SEEK ERROR".

PLAY AUDIO

Command code = 132
ES:BX = PlayReq

```
PlayReq     DB      13 dup (0)      ; Request header
            DB      ?               ; Addressing mode
            DD      ?               ; Starting sector number
            DD      ?               ; Number of sectors to read
```

This function will cause the driver to play the selected audio tracks until the requested sectors have been exhausted or until play is interrupted with an AUDIO STOP request. Control returns immediately to the caller. The busy bit in the status word will indicate if the drive is presently playing audio and when the play request is completed. The busy bit in the status word of the request header, as well as the paused bit and starting and ending locations returned by the Audio Status Info IOCTL, will be updated. If the drive does not support playing audio this request is ignored and it should return the error code for *Unknown Command.* If a data address is supplied, a "READ FAULT" error should be returned and the command should fail. If the sum of the starting sector and number of sectors to read exceeds the range of the disc, or there are intermediate data tracks, return the error code for *PARAMETER OUT OF RANGE.*

STOP AUDIO

Command code = 133
ES:BX = StopPlayReq

```
StopPlayReq DB      13 dup (0)      ; Request header
```

This function is included to pause the drive unit when it is currently in play mode, or to reset the starting and ending locations when in paused mode. If applicable, at the next stopping point it reaches the drive will discontinue playing, and process the next request. The busy bit in the status word of the request header, as well as the paused bit and starting location returned by the Audio Status Info IOCTL, will be updated. If the drive does not support playing audio, it should ignore this request and return the error code for *Unknown Command.*

RESUME AUDIO

Command code = 136
ES:BX = ResumeReq

```
ResumeReq   DB      13 dup (0)      ; Request header
```

This function is used to resume playing audio tracks when a previous PLAY AUDIO call has been paused with a STOP AUDIO command. It will resume playing from the starting location

indicated by the Audio Status Info IOCTL. It will modify the Audio Paused bit returned by the Audio Status Info IOCTL, and the busy bit in the status word of the request header. If the drive does not support playing audio, it should ignore this request and return the error code for *Unknown Command.*

In the following example the playing flag corresponds to the state reported by the busy bit in the status word of the request header when the drive is in audio play mode. The paused flag corresponds to the Audio Paused bit and last_startloc and last_endloc correspond to the starting and ending location reported in the Audio Status Info IOCTL.

```
// upon RESET,  NEW DISC,  or PLAY/RESUME COMPLETED the state
//should be updated:
playing                 = FALSE;
paused                  = FALSE;
last_startloc  = 0;
last_endloc             = 0;
PLAY_AUDIO( startloc, endloc )
 {if (playing) return error;
// Play overrides pause else if ( play( startloc, endloc ) != SUCCESSFUL ) return error;
      else
{playing = TRUE;
paused = FALSE;
last_startloc = startloc
last_endloc = endloc
return no error;}}
STOP_AUDIO( void )
{if ( playing )
{last_startloc = present q-channel location
playing = FALSE;
paused = TRUE;
stop();
return no error;}
      else

{playing = FALSE;
paused = FALSE;
last_startloc = 0;
last_endloc = 0;
return no error;}}
RESUME_AUDIO()
{if ( paused )
{if ( play( last_startloc, last_endloc ) != SUCCESSFUL )
return error;
      else
{playing = TRUE;
paused = FALSE;
return no error;}}
      else
return error;}
```

WRITE LONG

Command code = 134
ES:BX = WriteL

```
WriteL      DB      (dup 13 0)      ; Request header
            DB      ?               ; Addressing mode
            DD      ?               ; Transfer address
```

```
            DW      ?               ; Number of sectors to write
            DD      ?               ; Starting sector number
            DB      ?               ; Write mode
            DB      ?               ; Interleave size
            DB      ?               ; Interleave skip factor
```

The device will copy the data at the transfer address to the CD RAM device at the sector indicated. The media must be writable for this function to work. Data is written sector by sector, depending on the current write mode and the interleave parameters. The following values are recognized as valid write modes:

0	Mode 0
1	Mode 1
2	Mode 2 Form 1
3	Mode 2 Form 2
4-255	Reserved

Writing in Mode 1 is the default and must be supported. If the device driver supports the other modes, then they can be used. If Mode 0 is used, the transfer address is ignored and all sectors are written with zeroes. If the current write mode is Mode 1 or Mode 2 Form 1, each sector will consist of 2048 bytes of data located sequentially at the transfer address. If the write mode is Mode 2 Form 2, the device driver will expect 2336 bytes of data per sector at the transfer address.

WRITE LONG VERIFY

Command code = 135
ES:BX = WriteLV

```
WriteLV     DB      (dup 13 0)      ; Request header
            DB      ?               ; Addressing mode
            DD      ?               ; Transfer address
            DW      ?               ; Number of sectors to write
            DD      ?               ; Starting sector number
            DB      ?               ; Write mode
            DB      ?               ; Interleave size
            DB      ?               ; Interleave skip factor
```

This function is identical to WRITE LONG, with the addition that the device driver is responsible for verifying the data written to the device.

INPUT FLUSH

Command code = 7
ES:BX = FlushI

```
FlushI      DB      13 dup (0)      ; Request header
```

Requests that the device driver free all input buffers and clear any pending requests.

OUTPUT FLUSH

Command code = 11
ES:BX = FlushO

```
FlushO      DB      (dup 13 0)      ; Request header
```

Requests that the device driver write all unwritten buffers to the disk.

DEVICE OPEN

DEVICE CLOSE

Command code = 13,14
ES:BX = DevOpen, DevClose

```
DevOpen     DB      13 dup (0)          ; Request header
```

Used by the device driver to monitor how many different callers are currently using the CD-ROM device driver. All new device drivers should support these calls even if nothing is done with the information.

End.