

```

/*****
/*                                C D D E V . H                                */
/**-----**/
/* Task                : Header File for CDDEV.C                */
/**-----**/
/* Author              : Michael Tischer / Bruno Jennrich      */
/* Developed on         : 04/08/1994                            */
/* Last update         : 04/18/1994                            */
/*****

#ifndef _CDDEV_H
#define _CDDEV_H

/*- Device Driver Requests -----*/
/*- Device-Header -----*/
typedef struct tagDeviceHeader
{ LPVOID      lpNextHdr;                /* Address of next device driver */
  WORD        wDevAttributes;          /* Driver attributes */
  WORD        wStrategy;               /* Offset of strategy routine */
  WORD        wInterrupt;              /* Offset of interrupt routine */
  BYTE        bName[ 8 ];              /* Name of driver */
  WORD        wReserved;
  BYTE        bDriveLetter;            /* Drive letter */
  BYTE        bNumberOfSubUnits;       /* Number of SubUnits */
} DeviceHeader;
typedef DeviceHeader _FP *LPDeviceHeader;

```

```

/*- Request-Header -----*/
typedef struct tagRequestHeader
{
    BYTE bLength;                /* Length of request structure */
    BYTE bSubUnit;               /* Number of SubUnit */
    BYTE bCommand;               /* Command to execute */
    WORD wStatus;                /* Error status */
    BYTE bReserved[ 8 ]; /* will be redefined according to command the */
} RequestHeader;
typedef RequestHeader _FP *LPRequestHeader;

/*- RequestHeader.wStatus Bits : -----*/
#define DEV_ERROR ( 1 << 15 )    /* Set, then error in Bits 0-7 */
#define DEV_BUSY ( 1 << 9 )      /* Device is busy */
#define DEV_DONE ( 1 << 8 )      /* Device has finished last operation */

/*- With set error bit following error messages in Low-Byte -----*/
#define DEV_WRITE_PROTECT_VIOLATION    0
#define DEV_UNKNOWN_UNIT                1
#define DEV_DRIVE_NOT_READY            2
#define DEV_UNKNOWN_COMMAND            3
#define DEV_CRC_ERROR                  4
#define DEV_BAD_DRIVE_REQUEST_LEN      5
#define DEV_SEEK_ERROR                 6
#define DEV_UNKNOWN_MEDIA              7
#define DEV_SECTOR_NOT_FOUND           8
#define DEV_PRINTER_OUT_OF_PAPER       9
#define DEV_WRITE_FAULT                10

```

```

#define DEV_READ_FAULT 11
#define DEV_GENERAL_FAILURE 12
#define DEV_RESERVED1 13
#define DEV_RESERVED2 14
#define DEV_INVALID_DISK_CHANGE 15

/*- Device Commands -----*/
#define DEVCMD_INIT 0
/* INIT command is called only once during booting, after that */
/* no more. */
typedef struct tagMSCDEX_Init
{
    RequestHeader ReqHdr;
    BYTE          bNumberOfUnits;
    LPVOID        lpEndAddress;
    LPVOID        lpBPBArray;
    BYTE          bBlockDeviceNumber;
} MSCDEX_Init;
typedef MSCDEX_Init _FP *LPMSCDEX_Init;

#define DEVCMD_IOCTL_READ 3
/* The IOCTL_READ command is used to query diverse status */
/* information about the driver, drive and inserted CD. */
typedef struct tagMSCDEX_IOCTLIO
{
    RequestHeader ReqHdr;
    BYTE          bMediaDescriptor; /* always 0 */
    LPVOID        lpTransferAddress; /* = Address of Control-Block */
    WORD          wTransferCount;    /* = Size of Control-Block */
}

```

```

        WORD                wStartingSectorNo;                /* always 0 */
        LPVOID              lpVolumeID;                       /* always NULL */
    } MSCDEX_IOCTLIO;
typedef MSCDEX_IOCTLIO _FP *LPMSCDEX_IOCTLIO;

#define IOCTLI_GET_DEVHDR_ADRESS 0
/*- Returns the address of the DeviceHeader in memory -----*/
typedef struct tagMSCDEX_Raddr
{
    BYTE                bControlBlockCode;                    /* = 0 */
    LPDeviceHeader lpDeviceHeaderAddress;    /* DeviceHeader address */
} MSCDEX_Raddr;
typedef MSCDEX_Raddr _FP *LPMSCDEX_Raddr;

#define IOCTLI_GET_HEAD_LOCATION 1
/*- Gets Read/Write Head location -----*/
typedef struct tagMSCDEX_LocHead
{
    BYTE bControlBlockCode;                                    /* = 1 */
    BYTE bAdressingMode;                                        /* = HSG (0) or REDBOOK (1) */
    LONG lLocation;      /* Location of Read/Write head (sector) */
} MSCDEX_LocHead;
typedef MSCDEX_LocHead _FP *LPMSCDEX_LocHead;

#define IOCTLI_RESERVED1 2
/*- Reserved Command -----*/

#define IOCTLI_ERROR_STATISTICS 3
/*- Gets error statistics (not yet specified). -----*/

```

```

typedef struct tagMSCDEX_ErrStat
{ BYTE bControlBlockCode; /* = 3 */
  BYTE bErrStat[ 1 ]; /* Size of array not yet defined */
} MSCDEX_ErrStat;
typedef MSCDEX_ErrStat _FP *LPMSCDEX_ErrStat;

#define IOCTL_GET_AUDIO_CHANNEL_INFO 4
/*- Determines which CD channels are on which CD player output -*/
/*- as well as how loud they are. -*/
typedef struct tagMSCDEX_AudInfo
{ BYTE bControlBlockCode; /* = 4 */
  BYTE bOutput0, bVolume0; /* Which audio tracks are re- */
  BYTE bOutput1, bVolume1; /* directed to which CD output, and */
  BYTE bOutput2, bVolume2; /* how loud are the CD outputs? */
  BYTE bOutput3, bVolume3;
} MSCDEX_AudInfo;
typedef MSCDEX_AudInfo _FP *LPMSCDEX_AudInfo;

#define IOCTL_READ_DRIVE_BYTES 5
/*- Read data directly from drive (data is hardware dependent) -*/
typedef struct tagMSCDEX_DrvBytes
{ BYTE bControlBlockCode; /* = 5 */
  BYTE bNumReadBytes; /* = Number of bytes to read */
  BYTE bReadBuffer[ 128 ]; /* In 128 byte blocks */
} MSCDEX_DrvBytes; /* (previous blocks are discarded) */
typedef MSCDEX_DrvBytes _FP *LPMSCDEX_DrvBytes;

```

```

#define IOCTLI_GET_DEV_STAT 6
/*- Get drive status -----*/
typedef struct tagMSCDEX_DevStat
{ BYTE bControlBlockCode; /* = 6 */
  LONG lDeviceStatus; /* Contains drive status after call */
} MSCDEX_DevStat;
typedef MSCDEX_DevStat _FP *LPMSCDEX_DevStat;

#define DS_DOOR_OPEN 0x00000001L
/*- Door open -----*/
#define DS_DOOR_UNLOCKED 0x00000002L
/*- Door not locked -----*/
#define DS_READS_COOKED_AND_RAW 0x00000004L
/*- Data can be read Cooked and Raw, else only Cooked -----*/
#define DS_READ_WRITE 0x00000008L
/*- Drive can only write (e.g., WORM), else only read -----*/
#define DS_AUDIO_VIDEO 0x00000010L
/*- Drive supports audio and video Cds, else only data -----*/
#define DS_ISO_9600_INTERLEAVE 0x00000020L
/* Bit 6 reserved */
#define DS_PREFETCH 0x00000080L
/*- Prefetch possible -----*/
#define DS_MANIPULATE_AUDIO_CHANNEL 0x00000100L
/*- Audio channels can be redirected to CD outputs -----*/
#define DS_HSG_AND_REDBOOK 0x00000200L
/*- Addressing modes HSG and REDBOOK, else only HSG -----*/
/* BIT 10 reserved */

```

```

#define DS_NO_DISC_IN_DRIVE          0x00000800L
/*- No disc in drive -----*/
#define DS_SUPPORT_RW_SUB_CHANNELS  0x00001000L
/*- Sub channel info can be read -----*/

#define IOCTLI_SECTOR_SIZE 7
/*- Gets sector size in bytes -----*/
typedef struct tagMSCDEX_SectSize
{
    BYTE bControlBlockCode; /* = 7 */
    BYTE bReadMode; /* COOKED or RAW */
    WORD wSectorSize; /* contains sector size after call */
} MSCDEX_SectSize; /* COOKED: 2048 or RAW: 2352 bytes */
typedef MSCDEX_SectSize _FP *LPMSCDEX_SectSize;

#define IOCTLI_VOLUME_SIZE 8
/*- Location of lead-out tracks (after end of CD) -----*/
typedef struct tagMSCDEX_VolSize
{
    BYTE bControlBlockCode; /* = 8 */
    LONG lVolumeSize; /* after call location of last track + 1 */
} MSCDEX_VolSize;
typedef MSCDEX_VolSize _FP *LPMSCDEX_VolSize;

#define IOCTLI_MEDIA_CHANGED 9
/*- CD change occurred? -----*/
typedef struct tagMSCDEX_MedChng
{
    BYTE bControlBlockCode; /* = 9 */
    BYTE bMedChng; /* after call: NOT_CHANGED, DONT_KNOW, CHANGED */
}

```

```

    } MSCDEX_MedChng;
typedef MSCDEX_MedChng _FP *LPMSCDEX_MedChng;
#define NOT_CHANGED 1
#define DONT_KNOW 0
#define CHANGED 0xFF

#define IOCTL_AUDIO_DISK_INFO 10
/*- Gets CD information */
typedef struct tagMSCDEX_DiskInfo
{
    BYTE bControlBlockCode; /* = 10 */
    BYTE bLowestTrack; /* first track */
    BYTE bHighestTrack; /* last track */
    LONG bLeadOutTrack; /* Size of CD (REDBOOK) */
} MSCDEX_DiskInfo;
typedef MSCDEX_DiskInfo _FP *LPMSCDEX_DiskInfo;

#define IOCTL_AUDIO_TRACK_INFO 11
/*- Gets information about a specific track -----*/
typedef struct tagMSCDEX_TnoInfo
{
    BYTE bControlBlockCode; /* = 11 */
    BYTE bTrackNo; /* = Number of track to be searched */
    LONG lStartingPoint; /* Address in REDBOOK Format */
    BYTE bTrackControlInfo; /* Info about track */
} MSCDEX_TnoInfo;
typedef MSCDEX_TnoInfo _FP *LPMSCDEX_TnoInfo;

/*- Definitions for TrackControlInfo -----*/

```



```

#define TCI_4AUDIO_CHANNELS          0x80
#define TCI_DATA_TRACK               0x40
#define TCI_TRACK_MASK               0xC0
#define TCI_PRE_EMPHASIS             0x10
#define TCI_DIGITAL_COPY_PROHIBITED 0x20

#define IOCTL_AUDIO_QUERY_CHANNEL 12
/*- Get current track and disk playing time -----*/
typedef struct tagMSCDEX_QInfo
{
    BYTE bControlBlockCode; /* = 12 */
    BYTE bCONTROLandADR; /* TrackControl, Addressing (see TnoInfo) */
    BYTE bTrackNo; /* Number of current track (song) */
    BYTE bIndex; /* Current index within current track */
    BYTE bTrackMin; /* Current track playing time */
    BYTE bTrackFrame;
    BYTE bZero; /* always 0 */
    BYTE bDiskMin; /* Total playing time of CD */
    BYTE bDiskSec;
    BYTE bDiskFrame;
} MSCDEX_QInfo;
typedef MSCDEX_QInfo _FP *LPMSCDEX_QInfo;

#define IOCTL_AUDIO_SUB_CHANNEL_INFO 13
/*- Gets Sub Channel Info during playback -----*/
typedef struct tagMSCDEX_SubChanInfo
{
    BYTE bControlBlockCode; /* = 13 */

```

```

        LONG        lStartingSector;                /* = REDBOOK address */
        LPVOID       lpTransferAddress;             /* = Address of buffer */
        LONG         lNumberOfSectorsToRead;        /* = Amount of Sub Channel */
    } MSCDEX_SubChanInfo;                          /* information to be read. */
    typedef MSCDEX_SubChanInfo _FP *LPMSCDEx_SubChanInfo;

#define IOCTLI_UPCODE 14
/*- Get UPC/EAN number of inserted CD. -----*/
    typedef struct tagMSCDEx_UPCode
    {
        BYTE bControlBlockCode;                    /* = 14 */
        BYTE bCONTROLandADR;                        /* s. TnoInfo */
        BYTE bUPC_EAN[ 7 ];                        /* 13 figure BCD number */
        BYTE bZero;                                /* (last nibble unused) */
        BYTE bAFrame;
    } MSCDEX_UPCode;
    typedef MSCDEX_UPCode _FP *LPMSCDEx_UPCode;

#define IOCTLI_AUDIO_STATUS_INFO 15
/*- Gets Audio Status -----*/
    typedef struct tagMSCDEx_AudStat
    {
        BYTE bControlBlockCode;                    /* = 15 */
        WORD wAudioStatus;                          /* Bit 0 = Audio Pause, 1-15 reserved */
        LONG lResumeStart; /* Start position for DEVCMD_RESUME (REDBOOK) */
        LONG lResumeEnd; /* End position for DEVCMD_RESUME (REDBOOK) */
    } MSCDEX_AudStat;
    typedef MSCDEX_AudStat _FP *LPMSCDEx_AudStat;

```

```

#define DEVCMD_INPUTFLUSH 7
/*- INPUTFLUSH uses Standard RequestHeader -----*/
#define DEVCMD_OUTPUTFLUSH 11
/*- OUTPUTFLUSH uses Standard RequestHeader -----*/

#define DEVCMD_IOCTL_WRITE 12
/* Some drive parameters are changed using the WRITE command.          */
/* The MSCDEX_IOCTLIO block is used for this purpose (see above)        */
#define IOCTL_EJECT_DISK 0
/*- Opens CD door -----*/
typedef struct tagMSCDEX_Eject
{ BYTE bControlBlockCode;                      /* = 0 */
  } MSCDEX_Eject;
typedef MSCDEX_Eject _FP *LPMSCDEX_Eject;

#define IOCTL_LOCK_DOOR 1
/*- Locks door -----*/
typedef struct tagMSCDEX_LockDoor
{ BYTE bControlBlockCode;                      /* = 1 */
  BYTE bLock;                                  /* = 1: Lock, 0: Open the door */
  } MSCDEX_LockDoor;
typedef MSCDEX_LockDoor _FP *LPMSCDEX_LockDoor;

#define IOCTL_RESET_DRIVE 2
/*- Resetting of drive -----*/
typedef struct tagMSCDEX_Reset
{ BYTE bControlBlockCode;                      /* = 2 */

```

```

    } MSCDEX_Reset;
    typedef MSCDEX_Reset _FP *LPMSCDEx_Reset;

#define IOCTL_SET_AUDIO_CHANNEL_INFO 3
/*- Equivalent of IOCTL_GET_AUDIO_CHANNEL_INFO -----*/
#define IOCTL_WRITE_DRIVE_BYTES 4
/*- Equivalent of IOCTL_READ_DRIVE_BYTES -----*/
#define IOCTL_CLOSE_TRAY 5
    typedef struct tagMSCDEX_CloseTray
    { BYTE bControlBlockCode; /* = 5 */
      } MSCDEX_CloseTray;
    typedef MSCDEX_CloseTray _FP *LPMSCDEx_CloseTray;

#define DEVCMD_DEVICE_OPEN 13
/*- Open device -----*/
#define DEVCMD_DEVICE_CLOSE 14
/*- Close device -----*/
#define DEVCMD_READLONG 128
/*- Reads data from a CD -----*/

/*- The following structure is used for reading and writing data ---*/
typedef struct tagMSCDEX_ReadWriteL
{ RequestHeader ReqHdr;
  BYTE          bAddressingMode; /* = HSG(0) or REDBOOK(1) */
  LPVOID        lpTransferAddress; /* = Address of buffer */
  WORD          wNumSec; /* = Number of sectors to transferred */
  LONG          lStartingSector; /* = Read/Write position */

```

```

        BYTE            bDataReadWriteMode;        /* = Transfer mode see bel. */
        BYTE            bInterleaveSize; /* = Number of sequential sectors */
        BYTE            bInterleaveSkipFactor;     /* = Size of gap */
    } MSCDEX_ReadWriteL;
typedef struct MSCDEX_ReadWriteL _FP *LPMSDCDEX_ReadWriteL;

/*- bDataReadWriteMode Definitions: -----*/
/* Only RAW and COOKED are used for reading: */
#define COOKED 0
#define RAW    1

/* The following constants are effective during writing: */
#define MODE0 0 /* Deleting of sectors (overwrite with 0) */
#define MODE1 1 /* Inscription of sectors with specified data */
#define MODE2FORM1 2 /* COOKED Data-Write */
#define MODE2FORM2 3 /* RAW Data-Write */

#define DEVCMD_READLONG_PREFETCH 130
/*- sends next sector to be read to driver, so that the driver can */
/*- "cache" the sector end. !>

#define DEVCMD_SEEK 131
/*- Sets location of Read/Write Head on entire CD -----*/
typedef struct MSCDEX_SeekReq
{
    RequestHeader ReqHdr;
    BYTE          bAddressingMode; /* = HSG(0) or REDBOOK(1) */
    LPVOID        lpTransferAddress; /* is ignored, therefore = set 0 */

```

```

WORD        wNumSec;          /* is ignored, therefore = set 0 */
LONG        lStartingSector;  /* = starting location */
} MSCDEX_SeekReq;
typedef MSCDEX_SeekReq _FP *LPMSCDEX_SeekReq;

#define DEVCMD_PLAY_AUDIO 132
/* Play Audio Track ----- */
typedef struct tagMSCDEX_PlayReq
{
    RequestHeader ReqHdr;
    BYTE          bAddressingMode;          /* = HSG(0) or REDBOOK(1) */
    LONG          lStartSector;             /* = first sector to play */
    LONG          lSectorCount;            /* = number of sectors to play */
} MSCDEX_PlayReq;
typedef MSCDEX_PlayReq _FP *LPMSCDEX_PlayReq;

#define DEVCMD_STOP_AUDIO 133
/*- Stop audio output (uses Standard RequestHeader) ----- */
#define DEVCMD_WRITE_LONG 134
/*- Write data (see READ_LONG) ----- */
#define DEVCMD_WRITE_LONG_VERIFY 135
/*- Compare written data with data in memory ----- */
#define DEVCMD_RESUME_AUDIO 136
/*- Resume playing music interrupted by nonrecurring STOP_AUDIO ----- */

typedef struct HSG_DIR_ENTRY
{
    BYTE    len dr;          /* Length of this structure */

```

```

    BYTE  XAR_len;           /* XAR-Length in LBN (logical block numbers) */
    ULONG loc_extentI;       /* Start-LBNs (Intel Format) */
    ULONG loc_extentM;       /* Start-LBNs (Motorola Format) */
    ULONG data_lenI;         /* File length (Intel Format) */
    ULONG data_lenM;         /* File length (Motorola Format) */
    BYTE  record_time[6];    /* Date and Time */
    BYTE  file_flags_hsg;    /* Flags */
    BYTE  reserved;          /* reserved */
    BYTE  il_size;           /* Interleave: Number of sequential sectors */
    BYTE  il_skip;           /* Interleave: Size of gap */
    WORD  VSSNI;             /* Volume Set Sequence Number (Intel Format) */
    WORD  VSSNM;             /* Volume Set Sequence Number (Motorola Format) */
    WORD  len_fi;            /* Length of name */
/*  BYTE  file_id[ ... ];Filename, variable Length (max. 32 char.) */
/*  BYTE  padding; Pad-Byte (so that system data is word-aligned) */
/*  BYTE  sys_data[ ... ];          System data (variable length) */
} HSG_DIR_ENTRY;
typedef HSG_DIR_ENTRY _FP *LPHSG_DIR_ENTRY;

typedef struct ISO_DIR_ENTRY
{
    BYTE  len_dr;            /* Length of this structure */
    BYTE  XAR_len;           /* XAR-Length in LBN (logical block numbers) */
    ULONG loc_extentI;       /* Start-LBN (Intel Format) */
    ULONG loc_extentM;       /* Start-LBN (Motorola Format) */
    ULONG data_lenI;         /* File length (Intel Format) */
    ULONG data_lenM;         /* File length (Motorola Format) */

```

```

BYTE record_time[7];          /* Date and Time */
BYTE file_flags_iso;          /* Flags */
BYTE il_size;                 /* Interleave: Number of sequential sectors */
BYTE il_skip;                 /* Interleave: Size of gap */
WORD VSSNI;                   /* Volume Set Sequence Number (Intel Format) */
WORD VSSNM;                   /* Volume Set Sequence Number (Motorola Format) */
BYTE len_fi;                  /* Length of name */
/* BYTE file_id[ ... ];Filename, variable length (max. 32 char) */
/* BYTE padding; Pad-Byte (so that system data is word-aligned) */
/* BYTE sys_data[ ... ];      System data (variable length) */
} ISO_DIR_ENTRY;

typedef ISO_DIR_ENTRY _FP *LPISO_DIR_ENTRY;

typedef struct DIR_ENTRY
{
    BYTE XAR_len;              /* XAR-Length in LBN (logical block numbers) */
    ULONG loc_extent;          /* Start-LBN */
    WORD lb_size;              /* Size of an (LBN). If LBN size = 2048,
                               /* then an LBN corresponds to a physical
                               /* sector/frame.
    ULONG data_len;            /* File length (BYTES) */
    BYTE record_time[7];      /* Date and Time */
    BYTE file_flags;          /* Flags */
    BYTE il_size;             /* Interleave: Number of sequential sectors */
    BYTE il_skip;             /* Interleave: Size of gap */
    WORD VSSN;                /* Volume Set Sequence Number */
    BYTE len_fi;              /* Length of filename */

```



```

        BYTE   file_id[38];                /* Filename (ended with '\0') */
        WORD   file_version;              /* File version */
        BYTE   len_su;                    /* Length of system data */
        BYTE   su_data[220];              /* System data */
    } DIR_ENTRY;
typedef DIR_ENTRY _FP *LPDIR_ENTRY;
/*- Prototypes -----*/

INT _CallStrategy( INT iCD_Drive_Letter, LPRequestHeader lpReqHdr );
INT cd_IsError( WORD wStatus );
INT cd_GetReqHdrError( LPRequestHeader lpReqHdr );
INT cd_IsReqHdrBusy( LPRequestHeader lpReqHdr );
INT cd_IsReqHdrDone( LPRequestHeader lpReqHdr );
INT cd_GetDeviceHeaderAdress( INT iCD_Drive_Letter,
                              LPMSCDEX_Raddr lpRA );
INT cd_GetLocationOfHead( INT iCD_Drive_Letter, LPMSCDEX_LocHead lpLH );
INT cd_GetAudioChannelInfo( INT iCD_Drive_Letter,
                             LPMSCDEX_AudInfo lpAI );
INT cd_ReadDriveBytes( INT iCD_Drive_Letter, LPMSCDEX_DrvBytes lpDB );
INT cd_GetDevStat( INT iCD_Drive_Letter, LPMSCDEX_DevStat lpDS );
VOID cd_PrintDevStat( LPMSCDEX_DevStat lpDS );
INT cd_GetSectorSize( INT iCD_Drive_Letter,
                      LPMSCDEX_SectSize lpSEC,
                      INT iReadMode );
INT cd_GetVolumeSize( INT iCD_Drive_Letter, LPMSCDEX_VolSize lpVS );
INT cd_GetMediaChanged( INT iCD_Drive_Letter, LPMSCDEX_MedChng lpMC );
INT cd_GetAudioDiskInfo( INT iCD_Drive_Letter, LPMSCDEX_DiskInfo lpDI );

```

```

INT cd_GetAudioTrackInfo( INT          iCD_Drive_Letter,
                          INT          iTrackNo,
                          LPMSCDEX_TnoInfo lpTI );
INT cd_IsDataTrack( LPMSCDEX_TnoInfo lpTI );
LONG cd_GetTrackLen( INT iCD_Drive_Letter, INT iTno );
INT cd_QueryAudioChannel( INT iCD_Drive_Letter, LPMSCDEX_QInfo lpQI );
INT cd_GetAudioSubChannel( INT          iCD_Drive_Letter,
                           LPMSCDEX_SubChanInfo lpSCI );
INT cd_GetUPCode( INT iCD_Drive_Letter, LPMSCDEX_UPCode lpUPC );
VOID cd_PrintUPCode( LPMSCDEX_UPCode lpUPC );
INT cd_GetAudioStatusInfo( INT iCD_Drive_Letter, LPMSCDEX_AudStat lpAS );
INT cd_Eject( INT iCD_Drive_Letter );
INT cd_LockDoor( INT iCD_Drive_Letter, INT iLock );
INT cd_ResetDrive( INT iCD_Drive_Letter );
INT cd_SetAudioChannelInfo( INT          iCD_Drive_Letter,
                           LPMSCDEX_AudInfo lpAI );
INT cd_WriteDriveBytes( INT iCD_Drive_Letter, LPMSCDEX_DrvBytes lpDB );
INT cd_CloseTray( INT iCD_Drive_Letter );
INT cd_PlayAudio( INT iCD_Drive_Letter,
                 INT iAdressMode,
                 LONG lStart,
                 LONG lLen );
INT cd_StopAudio( INT iCD_Drive_Letter );
INT cd_ResumeAudio( INT iCD_Drive_Letter );
INT cd_Seek( INT iCD_Drive_Letter, INT iAdressingMode, LONG lSector );
INT cd_ReadLong( INT          iCD_Drive_Letter,
                 INT          iAdressMode,

```

```

        LONG        lStart,
        INT         iNumSec,
        LPVOID      lpReadData,
        INT         iDataMode );
INT cd_ReadLongPrefetch( INT   iCD_Drive_Letter,
                        INT   iAdressMode,
                        LONG lStart );
INT cd_WriteLong( INT   iCD_Drive_Letter,
                INT   iAdressMode,
                LONG   lStart,
                INT   iNumSec,
                LPVOID lpWriteData,
                INT   iDataMode );
INT cd_WriteLongVerify( INT   iCD_Drive_Letter,
                      INT   iAdressMode,
                      LONG   lStart,
                      INT   iNumSec,
                      LPVOID lpVerifyData,
                      INT   iDataMode );
VOID cd_PrintDiskTracks( INT iCD_Drive_Letter );
VOID cd_FastForward( INT iCD_Drive_Letter );
INT cd_PrintActPlay( INT iCD_Drive_Letter );
VOID cd_PrintSector( LPVOID lpSector, INT iSize, INT iCols, INT iRows );
VOID cd_PrintDirEntry( LPDIR_ENTRY lpDirEntry );

#endif

```

