# cdplay

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>cdplay | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | March 1, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# cdplay

## 1.1 cdplay.library -- Dokumentation

```
                          cdplay.library -- (c) Copyright ´96 by Patrick Hess
        cdplayer.library -- (c) Copyright ´95 by Patrick Hess

                        Library-Funktionen

                Strukturen und Daten

                Übersicht: Wie gehts?
               Die  cdplayer.library  ist  eine  Library  zur Ansteuerung von ↩
                  SCSI-CD-Rom-
Laufwerken.  Sie erlaubt eine komfortable Ansteuerung der Audiofunktionen,
wenn eine Audio-CD im Laufwerk ist.

Mein  größter  Dank geht an Gary Duncan aus dessen SCSIUtil ich die meisten
Dingen  für  diese  Library gelernt habe.  Ausserdem geht mein Dank noch an
Thomas  Herold,  der  mir  bei der Überwindung anfänglicher Schwierigkeiten
der alten cdplayer.library geholfen hat.

Diese  Library  und  die  mitgelieferten  Files  sind  alle  FreeWare,  das
Copyright  verbleibt bei mir, die Nutzung ist kostfrei.  Diese Library darf
frei kopiert werden und in Mailboxen/Mailboxnetzen/Internet gespeichert und
vertrieben werden, solange an diesem Packet kein Profit gemacht wird.

Die Benutzung dieser Library in eigenen Programmen ist natürlich gestattet,
jedoch  mit  (geringen)  Auflagen  verbunden.  Diese  hängen  mit  der
Vertriebsart/Form der Software zusammen:

- Programm benutzt cdplay.library und ist FreeWare:

  Hier bitte ich um eine e-Mail (falls möglich) das an einem Projekt  gear-
  beitet wird, wenn dieses fertig ist, muß mir eine Kopie des Programms zu-
  gestellt werden (am besten via e-Mail, aber Post ist auch O.K.)

- ... ist ShareWare:

  Solange der Betrag der ShareWare unter  50 DM bleibt, reicht  eine regis-
  trierte Version der Software, aber  auch für Folgeversionen, die die Lib-
  rary nutzen.
```

- ... kommerzielle Software

  Nachfragen!

Sollte ein Projekt nicht in eine dieser Kathegorien fallen, gilt das selbe wie bei kommerziellen Produkten: fragen... ;)

Bugreports, Verbesserungs- und Erweiterungsvorschläge usw. an folgende Adresse, e-Mail SEHR bevorzugt...

e-Mail: poseidon@newswire.de
hess@neuss.netsurf.de (http://www.neuss.netsurf.de/~patrhess

Postadresse: Patrick Hess
Holsteinstr. 27
41564 Kaarst

Telefon: V+49-2131-766672Q

## 1.2 Datenstrukturen der CDPlay Library

/include/libraries/cdplay.h:

Grundlage der cdplay.library ist der CDRequest. Um ein CDRom als Audioplayer zu verwenden wird die Funktion CDOpenDriver() benutzt. Als Parameter bekommt sie das SCSI-Device des Laufwerks und die ID des CD-Rom's. Geliefert wird dann ein Pointer auf eine Struktur vom Typ CDRequest. Diese Struktur benötigt JEDE Funktion der Library. Nach dem Gebrauch wird die Struktur mit CDCloseDrive() wieder gelöscht, Speicher frei- gegeben und das Device geschlossen. Nach dem Öffnen muß selbstständig geprüft werden, ob es sich auch um eine Audio-CD handelt.

Der "struct CDRequest" enthält folgende Einträge, Strukturen sind weiter unten näher beschrieben:

  struct IOStdReq *cdr_Request;

  Ein IOStdReq für die Kommunikation der Library mit Device. Im Prinzip ist diese Struktur privat, wer sie für eigene Zwecke benutzen möchte tut dieses auf eigenes Risiko.

  struct MsgPort *cdr_MsgPort;

  Ein Messageport für die Devicekommunikation.

  struct CDCapacity *cdr_Capacity;

  Dieser Struct gibt Auskunft über einige Parameter der CD, z.B. Größe/Länge etc.

  struct CDInquiry *cdr_Inquiry;

  Informationen über das Device.

```
  struct  CDPTOC    *cdr_TOC;

  Das Inhaltsverzeichniss der CD.

  struct  CDTime    *cdr_Time;

  Informationen  über die aktuellen Trackzeiten, Gesammtspielzeit der
  CD und ähnliche Angaben.

  struct  CDVolume  *cdr_Volume;

  Lautstärke des CD-Rom's, zur Zeit noch unbenutzt.

  UBYTE cdr_ID[20];

  Jede CD bekommt eine ID, die (hoffentlich ;) ) nur einmal auftritt.
        Dadurch kann die CD jederzeit wieder identifziert werden.  Diese ID
  ist kompatibel zu der, die MCDP oder der BGUIPlayer verwenden.

  UBYTE cdr_Active;

  Gibt den aktuellen Zustand des Laufwerks aus, möglich sind

    SCSI_STAT_NO_DISK keine CD im Laufwerk
    SCSI_STAT_PLAYING es wird eine Audio-CD abgespielt
    SCSI_STAT_STOPPED CD im Laufwerk, wird nicht gespielt
    SCSI_STAT_PAUSED        Pausefunktion ist aktiv

  UBYTE cdr_CurrentTrack;

  Der  aktuelle  Track,  der gerade abgespielt wird.  Dieser Wert ist
  nur  gültig,  wenn cdr_Active auf SCSI_STAT_PLAYING oder ..._PAUSED
  ist, in allen anderen Fällen ist das Resultat undefiniert.

  ULONG   cdr_CurrentAddress;

  Aktuelle Addresse auf der CD, dieser Wert kann z.B.  mit den Macros
  base2min() und base2sec() in eine reale Zeit umgerechnet werden.

  UBYTE *cdr_SCSISense;
  UBYTE *cdr_SCSIData;
  UBYTE *cdr_TOCBuf;

  3  interne  Puffer,  FINGER  WEG!  Manipulationen an diesen Puffern
  können recht seltsame Folgen haben...


Der "struct CDCapacity" gibt 3 Informationen über die eingelegte CD:

  ULONG cdc_MaxSector;

  Anzahl der Sektoren der CD.

  ULONG cdc_SectorSize;

  Größe eines Sektors (in der Regel dürften das 2048 Bytes sein)
```

```
  ULONG cdc_Capacity;

  Die Kapazität der CD (cdr_MaxSector * cdr_Capacity)
```

Der "struct CDInquiry" gibt Informationen über das verwendete CD-Rom:

```
  ULONG cdi_Flags;

  Die  möglichen  Flags  sind  in  libraries/cdplay.h definiert, eine
  Erklärung  ist  nicht  nötig, um die Library voll nutzen zu können.
  Wenn die Info jemanden doch interessieren, wird er mit den Namen im
  Include genügend anfangen können.

  UBYTE cdi_DeviceType;

    DEVTYPE_DIRECT_ACCESS   Festplatten etc.
    DEVTYPE_SEQUENTIAL_ACCESS Streamer etc.
    DEVTYPE_PRINTER     Drucker
    DEVTYPE_PROCESSOR   Prozessordevices
    DEVTYPE_WRITE_ONCE    WORM Laufwerke
    DEVTYPE_CDROM     CDRom, wenn ein Device  mit
            der  cdplay.library benutzt
            werden soll, muß auf dieses
            Flag getestet werden.
    DEVTYPE_SCANNER     Scanner
    DEVTYPE_OPTICAL     Optische Medien
    DEVTYPE_MEDIUM_CHANGER    Jukeboxen, Wechsler, ...
    DEVTYPE_COMMUNICATIONS    Kommunikationsdevices
    DEVTYPE_ASC_IT8_1   \
    DEVTYPE_ASC_IT8_2    >~erklärt sich selbst
    DEVTYPE_UNKNOWN     /

  UBYTE cdi_ANSIVersion;

  Die ANSI Version des Devices, hier sind folgende Werte möglich:

    ANSI_NONE nicht genormt
    ANSI_SCSI_1 SCSI-I Device
    ANSI_SCSI_2 SCSI-II Device

  [ ... ]

  UBYTE cdi_VendorID[9];

  Herstellername als ASCII, 0 terminiert.

  UBYTE cdi_ProductID[17];

  Produktbezeichnung, 0 terminiert.

  UBYTE cdi_RevisionLevel[5];

  Version der Firmware, wieder 0 terminiert.

  UBYTE cdi_VendorSpecific[21];
```

Herstellerspezifischer Text/Daten.

UBYTE cdi_Reserved[36];

Reservierter Teil.

Der "struct CDPTOC" ist das Inhaltsverzeichniss einer CD. Er besteht aus folgenden Daten:

ULONG cdptoc_TOCSize;

Interner Wert

UBYTE cdptoc_FirstTrack;

Erster Track auf der CD.

UBYTE cdptoc_LastTrack;

Letzter Track auf der CD.

struct CDTrack cdptoc_Track[100];

Array mit Informationen über die Tracks. Auf einer CD können nicht mehr als 100 Tracks sein. Es stehen für jede CD cdtoc_LastTrack - cdtoc_FirstTrack Tracks und somit auch Trackinfos bereit.

Der "struct CDTrack" enthält Daten für jeden Track auf der CD, auch für Datenbereich, die in der Regel der erste Track sind:

ULONG tr_Position;

Startposition auf der CD, diese Adresse kann mit den Macros base2min() und base2sec() in Minuten und Sekunden gewandelt werden und ist die Startzeit.

ULONG tr_Flags;

Flags, die den Track beschreiben, z.B.

IFLAG_COPY_PROHIBITED Track darf nicht kopiert werden
IFLAG_AUDIO_TRACK Audiotrack

UBYTE tr_SubChan;

SubChan Informationen.

Der "struct CDTime" ist im CDRequest enthalten und wird durch CDUpdate() mit aktuellen Informationen über die Spielzeit gefüllt. Alle Werte können mit den base2min() und base2sec() Macros in reale Werte gewandelt werden.

Die ersten 3 Werte beziehen sich auf den aktuelle Track, die letzten 3 auf die komplette CD:

ULONG cdt_TrackCurBase;

```
  Aktuelle Zeit

  ULONG cdt_TrackRemainBase;

  Verbleibende Zeit

  ULONG cdt_TrackCompleteBase;

  Gesammtspielzeit

  ULONG cdt_AllCurBase;

  Aktuelle Zeit

  ULONG cdt_AllRemainBase;

  Verbleibende Zeit

  ULONG cdt_AllCompleteBase;

  Gesammtspielzeit
```

Die beiden Macros

```
  #define base2min(val) ((val)/75/60)
  #define base2sec(val) (((val)/75)%60)
```

dienen dazu die Position auf der CD von einer Adresse in Zeitwerte umzurechnen. Habe ich mir z.B. gerade den aktuellen Stand der CD geholt, so liefert eine Zeile wie

```
  printf ("%02d:%02d", base2min (my_cdrequest->cdr_CurrentAddress),
           base2sec (my_cdrequest->cdr_CurrentAddress));
```

zum Beispiel "12:44".

Um Bits einfacher zu testen verwende ich in meinen Sourcen i.d.Regel Macros, die ich den Benutzern meiner Library auch zur Verfügung stellen möchte:

```
  ein Bit testen:   #define btst(n,m) (n & m)
  ein Bit wandeln:  #define bchg(n,m) (n = (n & m) ? (n & (~m)) : ( n | m))
  ein Bit testen:   #define bset(n,m) (n = n | m)
```

## 1.3 Übersicht: Wie gehts?

Hier noch ein kurzer Überblick: was ist zu tun, um mit der Library eine CD abzuspielen:

- CDOpenDriver() auf ein SCSI-Laufwerk ansetzen
- in der zurückgegebenen Struktur nachsehen, ob das Device auch wirklich ein CDRom ist.
- ist der Track, den ich abspielen möchte auch ein Audiotrack?

- CDCloseDrive() am Ende nicht vergessen!