Programmierung

Um Drittapplikationen die Möglichkeit zu geben, die Funktionen des RSJ CD-Writer Dateisystems zu steuern, wird in diesem Kapitel die Schnittstelle dokumentiert, die von den mitgelieferten Programmen (cdattach, cdcopy, ...) verwendet wird.

Hinweis: Die Programmschnittstelle wird mit zukünftigen Versionen des Dateisystems erweitert werden.

Die Programmschnittstelle ist für Micrsosoft C6.00 bzw. IBM CSet/2 und CSet++ ausgelegt. Sowohl die Bibliotheken als auch die Includedateien können von beiden Compilern verwendet werden.

CDWFSCTL.H

Um die Programmschnittstelle zu verwenden, muß die Includedatei "cdwfsctl.h" in das Programm eingefügt werden. Diese Datei definiert die Datentypen und Konstanten, die zur Kommunikation mit dem Dateisystem verwendet werden.

Im Folgenden werden die einzelnen API-Funktionen des Dateisystems erläutert.

ATTACH

Der ATTACH-Befehl dient zum Anmelden einer CD auf dem angegebenen Laufwerksbuchstaben. Dazu wird die Struktur ATTACH_INFO verwendet:

```
 /****************************************************************************

  'ATTACH_INFO' contains the information which is used to attach a drive
  letter to the file system

****************************************************************************/

typedef struct {
  short       len;              /* length of this structure */
  char device[20];              /* name of the SCSI device */
  short       sessions_to_skip; /* number of sessions to skip (open last...) */
  short       formatted;        /* != 0, if medium is formatted */
  } ATTACH_INFO;
```

Die einzelnen Felder haben folgende Bedeutung:

len	Input. Gesamtlänge der Struktur
device	Input. Name des SCSI Devicetreibers. Im allgemeinen sollte hier der Name RSJSCSI$ verwendet werden.
sessions_to_skip	Input. Anzahl der zu überspringenden Sessions. 0 = aktuelle Session, 1 = letzte Session, 2 = vorletzte Session, usw.
formatted	Output. Gibt an, ob die CD bereits formatiert ist.

Beispiel:

```
#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
ATTACH_INFO attach_info;
APIRET ret;

/* initialize attach info */
attach_info.len = sizeof(ATTACH_INFIO);
attach_info.sessions_to_skip = 0;
```

```
strcpy(attach_info.device, "RSJSCSI$");

/* attach drive Z: */
ret = DosFSAttach("z:",
                  "cdwfs",
                  (void *) &attach_info,
                  sizeof(ATTACH_INFO),
                  FS_ATTACH);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

DETACH

Der DETACH-Befehl dient zum Abmelden einer CD. Dazu wird die Struktur DETACH_INFO verwendet:

```
/*****************************************************************************

  'DETACH_INFO' contains the information which is written into the primary
  volume descriptor of the CD when 'flush_mode' is greater that FLUSH_CACHE.

*****************************************************************************/

typedef struct {
  short       len;                  /* length of this structure */
  FLUSH_MODE  flush_mode;           /* type of flush requested */
  char        vol_set_id[128];      /* volume set identifier */
  char        publisher_id[128];    /* publisher identifier */
  char        preparer_id[128];     /* data preparer identifier */
  char        app_id[128];          /* application identifier */
  char        cpyrght_file[37];     /* name of copyright file in root */
  char        abstrct_file[37];     /* name of abstract file in root */
  char        biblio_file[37];      /* name of bibliographic file in root */
  } DETACH_INFO;
```

Die einzelnen Felder haben folgende Bedeutung:

len	Input. Gesamtlänge der Struktur
flush_mode	Input. Der flush_mode kann auf einen der folgenden Werte gesetzt werden:
FLUSH_NONE	Das Laufwerk wird abgemeldet, ohne daß die seit dem Anmelden geschriebenen Daten gesichert werden.
FLUSH_CACHE	Die Daten im Zwischenspeicher werden vor dem Abmelden auf die CD geschrieben. Da die Verzeichnisinformationen nicht aktualisiert werden, kann auf diese Daten jedoch nicht zugegriffen werden. Dieser Modus wird intern verwendet.
FLUSH_DIRECTORY	Die Daten im Zwischenspeicher werden vor dem Abmelden auf die CD geschrieben und das aktuelle Verzeichnis gespeichert. Dies entspricht dem Befehl "cdattach <Laufwerk> -c".
FLUSH_SESSION	Die Daten im Zwischenspeicher werden vor dem Abmelden auf die CD geschrieben und das aktuelle Verzeichnis gespeichert. Danach wird die aktuelle Session geschlossen und eine neue Session geöffnet. Dies entspricht dem Befehl "cdattach <Laufwerk> -s".
FLUSH_SEAL	Wie FLUSH_SESSION, nur wird nach dem Öffnen der neuen Session kein Track reserviert. Die CD ist nach diesem Befehl schreibgeschützt. Der Schreibschutz kann mit "format <Laufwerk> /UNSEAL" aufgehoben werden
vol_set_id, publisher_id, preparer_id, app_id, app_id, cpyrght_file, abstrct_file, biblio_file	Input. Diese Felder werden im Primary Volume Descriptor gespeichert und können u.a. mit dem Befehl "chkdsk <Laufwerk> /V" ausgegeben werden.

Beispiel:

```
#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static DETACH_INFO detach_info;
APIRET ret;

/* initialize detach info */
detach_info.len = sizeof(DETACH_INFIO);
detach_info-flush_mode = FLUSH_SESSION

strcpy(detach_info.vol_set_id, "My first CD");
strcpy(detach_info.publisher_id, "RSJ Software GmbH");
strcpy(detach_info.preparer_id, "Bugs Bunny");
strcpy(detach_info.app_id, "RSJ CD-Writer File System");
strcpy(detach_info.cpyrght_file, "");
strcpy(detach_info.abstrct_file, "");
strcpy(detach_info.biblio_file, "");

/* detach drive Z: */
ret = DosFSAttach("z:",
                  "cdwfs",
                  (void *) &detach_info,
                  sizeof(DETACH_INFO),
                  FS_DETACH);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

SET_SPEED

Der SET_SPEED-Befehl dient zum Einstellen der Schreibgeschwindigkeit des CD-Recorders.
Für den SET_SPEED-Befehl wird die Struktur SPEED_INFO verwendet:

```
/*****************************************************************************

  'SPEED_INFO' is used to specify the recording speed as well as the write
  mode (emulation write or physical write).

*****************************************************************************/

typedef struct {
  short       speed_factor;         /* 1 = 150K, 2 = 300K, 4 = 600K, ... */
  short       emulation_write;      /* if set, the CD will not be modified */
  } SPEED_INFO;
```

Die einzelnen Felder haben folgende Bedeutung:

speed_factor — Input. Dieses Feld enthält den Geschwindigkeitsfaktor (1 = 150KB/s, 2 = 300KB/s, 4 = 600KB/s, ...).

emulation_write Input. Wird dieses Feld auf einen Wert ungleich 0 gesetzt, werden die Daten nicht auf die CD geschrieben, sondern nur der Schreibvorgang simuliert. Dies dient im allgemeinen zum Testem der Übetragungsgeschwindigkeit zwischen PC und CD-Recorder.

Hinweis: Im Gegensatz zu den anderen FSCtl-Aufrufen kann bei diesem Aufruf das Dateisystem auf zwei verschiedene Arten angesprochen werden:

- Unter Angabe eines Laufwerksbuchstabens (z.B. "z:\\") mit FSCTL_PATHNAME. Hierbei wird die Geschwindigkeit des an dem Laufwerk angeschlossenen CD-Recorders gleich mitgeändert.
- Unter Verwendung des Dateisystem-Namens (CDWFS) mit FSCTL_FSDNAME. Diese Möglichkeit erlaubt es, die Standardgeschwindigkeit zu Ändern, ohne daß dabei eine CD angemeldet sein muß.

Beispiel:

```
#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static SPEED_INFO speed_info;
USHORT parm_len = sizeof(SPEED_INFO);
USHORT data_len = 0;

/* select double speed and no emulation write */
speed_info.speed_factor = 2;
speed_info.emulation_write = 0;

/* call SPEED_INFO entry point in CDWFS */
ret = DosFSCtl(NULL,
               data_len,
               &data_len,
               (PBYTE) &speed_info,
               parm_len,
               &parm_len,
               CDWFS_SET_SPEED,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

FORMAT

Der FORMAT-Befehl dient zum Formatieren von neuen bzw. "versiegelten" Medien. Der FORMAT-Befehl erwartet einen Parameter vom Typ FORMAT_MODE, in dem die Art der gewünschten Formatierung angegeben wird.

Dieser Formatierungsmodus kann folgende Werte haben:

FORMAT_EMPTY_MEDIUM Gibt an, daß nur leere Medien formatiert werden sollen.

FORMAT_UNSEAL Hebt eine "Versiegelung" auf, d.h. die CD wird wieder schreibfähig gemacht.

Beispiel:

```
 #include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
FORMAT_MODE format_mode = FORMAT_EMPTY_MEDIUM;
USHORT data_len = 0;
USHORT parm_len = sizeof(FORMAT_MODE);

/* call FORMAT entry point in CDWFS */
ret = DosFSCtl(NULL,
               data_len,
               &data_len,
               (PBYTE) &format_mode,
               parm_len,
               &parm_len,
               CDWFS_FORMAT,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CHKDSK

Der CHKDSK-Befehl liefert die Informationen, die von der Datei "ucdwfs.dll" verwendet werden, um die Ausgabe des OS/2-Befehls "CHKDSK" zu erzeugen.

Für den CHKDSK-Befehl wird die Struktur CHKDSK_DATA verwendet:

```
/*****************************************************************************

  'CHKDSK_DATA' defines the information which is returned by the
  FSCTL_CHKDSK request.

*****************************************************************************/

typedef struct {
  char        copyright[100];       /* copyright string with version information */
  long        file_count;           /* number of files on the CD */
  long        dir_count;            /* number of directories on the CD */
  long        file_disk_usage;      /* volume space occupied by files */
  long        dir_disk_usage;       /* volume space occupied by directories */
  short       finalized_sessions;   /* number of finalized sessions on the CD */
  short       open_session;         /* currently open session */
  short       track_count;          /* number of tracks on the CD */
  short       reserved_track;       /* currently reserved track */
  short       fixation_recommended; /* power calibration area almost full */
  short       modified;             /* CD has been modified */
  DETACH_INFO pvd_info;             /* information about the PVD */
  } CHKDSK_DATA;
```

Die einzelnen Felder haben folgende Bedeutung:

copyright	Output. Dieses Feld enthält die Copyright-Meldung, die vom OS/2-Befehl CHKDSK ausgegeben wird.
file_count	Output. Anzahl der Dateien auf der CD
dir_count	Output. Anzahl der Unterverzeichnisse auf der CD
file_disk_usage	Output. Anzahl der Bytes, die von Dateien belegt werden
dir_disk_usage	Output. Anzahl der Bytes, die von Verzeichnissen belegt werden
finalized_sessions	Output. Anzahl der bereits abgeschlossenen Sessions
open_session	Output. Aktuelle Session. Wenn dieser Wert 0 ist, ist die CD entweder voll oder es handelt sich um eine CD-ROM. Auf jeden Fall kann diese CD nicht beschrieben werden.
track_count	Output. Anzahl der Tracks auf der CD
reserved_track	Output. Nummer des reservierten Tracks. Wenn diese Nummer dem Wert in track_count entspricht, sind auf dieser CD keine Dateien geschrieben worden, nachdem die CD mit "cdattach -s" abgemeldet wurde. Sollte reserved_track den Wert 0 enthalten, ist die CD entweder voll, mit "cdattach -x" schreibgeschützt oder es handelt sich um eine CD-ROM.
fixation_recommended	Output. Ist dieses Feld ungleich 0, ist diese CD bereits so oft beschrieben worden, daß beim nächsten Abmelden auf jeden Fall die aktuelle Session geschlossen wird, da die Gefahr besteht, daß die CD nicht mehr weiter beschrieben werden kann.
modified	Output. Ist dieses Feld ungleich 0, wurde die CD seit dem Anmelden modifiziert.
pvd_info	Output. Dieses Feld enthält den Inhalt der beim letzten Abmelden mit einem flush_mode >= FLUSH_SESSION übergebenen Struktur DETACH_INFO. Dies dient zum Ausgeben der im Primary Volume Descriptor gespeicherten Zusatzinformationen wie z.B. vol_set_id.

Beispiel:

```
 #include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static CHKDDSK_DATA chkdsk_data;
USHORT data_len = sizeof(CHKDSK_DATA);
USHORT parm_len = 0;

/* call CHKDSK entry point in CDWFS */
ret = DosFSCtl((PBYTE) &chkdsk_data,
               data_len,
               &data_len,
               NULL,
               parm_len,
               &parm_len,
               CDWFS_CHKDSK,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
```

```
  }
}
```

CACHE_INFO

Der CACHE_INFO-Befehl liefert Informationen über die Gesamtgröße und den momentanen Füllgrad des Zwischenspeichers.

Für den CACHE_INFO-Befehl wird die Struktur CACHE_INFO verwendet:

```
/*****************************************************************************

  'CACHE_INFO()' contains information about the write cache.

*****************************************************************************/

typedef struct {
  long        size;                /* user-specified cache size */
  long        pos;                 /* current cache position */
  } CACHE_INFO;
```

Die einzelnen Felder haben folgende Bedeutung:

size — Output. Dieses Feld enthält die Gesamtgröße des Zwischenspeichers. Der Wert entspricht der Option "-c" im Befehl IFS=<Pfad>CDWFS.IFS, auf Sektorgröße (2048 Bytes) gerundet.

pos — Output. Momentane Position des Schreibzeigers im Zwischenspeicher. Hiermit kann überprüft werden, ob eine bestimmte Datei noch in den Zwischenspeicher paßt oder ob ein neuer Track erforderlich ist, um die Datei zu schreiben.

Beispiel:

```
#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static CACHE_INFO cache_info;
USHORT data_len = sizeof(CACHE_INFO);
USHORT parm_len = 0;

/* call CACHE_INFO entry point in CDWFS */
ret = DosFSCtl((PBYTE) &cache_info,
               data_len,
               &data_len,
               NULL,
               parm_len,
               &parm_len,
               CDWFS_GET_CACHE_INFO,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CopyToCD()

Mit CopyToCD() wird eine Datei auf die CD kopiert. Diese Funktion dient zum Kopieren von Dateien, die größer sind als die Größe des Zwischenspeichers. Weitere Informationen hierzu können in der Beschreibung des Befehls "cdcopy" nachgelesen werden.

Die Funktion CopyToCD() befindet sich in der .DLL "cdwcpy.dll". Um diese Funktion zu verwenden, muß die Bibliothek "cdwcpy.lib" zu dem Programm "gelinkt" werden.

Syntax:

```
#include <cdwfsctl.h>

extern CDW_LINKAGE CopyToCD (char *source,
                             char *target);
```

Parameter:

source Input. Vollständiger Name der Quelldatei.

target Input. Vollständiger Name der Zieldatei.

Beispiel:

```
#include <stdio.h>

#include "cdwfsctl.h"

main()
{
APIRET ret;

/* copy a huge file into a single track */
ret = CopyToCD("c:\\data\\largefile.dat", "z:\\largefile.dat");

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

XCopyToCD()

Mit XCopyToCD() werden ganze Verzeichnisbäume auf die CD kopiert. Diese Funktion dient zum Kopieren von Dateien, die größer sind als die Größe des Zwischenspeichers. Weitere Informationen hierzu können in der Beschreibung des Befehls "cdcopy" nachgelesen werden.

Hinweis: Die notwendigen Zielverzeichnisse werden automatisch erstellt.

Die Funktion XCopyToCD() befindet sich in der DLL "cdwcpy.dll". Um diese Funktion zu verwenden, muß die Bibliothek "cdwcpy.lib" zu dem Programm "gelinkt" werden.

Syntax:

```
#include <cdwfsctl.h>

extern CDW_LINKAGE XCopyToCD (char *source,
                              char *target);
```

Parameter:

source Input. Vollständiger Name der Quelldatei. Dieser Name kann Platzhalter ("?" oder "*") enthalten.

target Input. Vollständiger Name des Zielverzeichnisses. Es darf weder ein Zieldateiname noch ein Platzhalter angegeben werden.

Beispiel:

```
#include <stdio.h>
```

```
#include <cdwfsctl.h>

main()
{
APIRET ret;

/* copy complete directory tree to the CD */
ret = XCopyToCD("c:\\os2\\*", "z:\\os2bkup");

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

XCopyToCD2()

Mit XCopyToCD2() werden ganze Verzeichnisbäume auf die CD kopiert. Diese Funktion dient zum Kopieren von Dateien, die größer sind als die Größe des Zwischenspeichers. Weitere Informationen hierzu können in der Beschreibung des Befehls "cdcopy" nachgelesen werden.

Hinweis: Die notwendigen Zielverzeichnisse werden automatisch erstellt.

Die Funktion XCopyToCD2() befindet sich in der DLL "cdwcpy.dll". Um diese Funktion zu verwenden, muß die Bibliothek "cdwcpy.lib" zu dem Programm "gelinkt" werden.

Syntax:

```
#include <cdwfsctl.h>

extern CDW_LINKAGE XCopyToCD2 (char  *source,
                               char  *target,
                               short  verbose);
```

Parameter:

source — Input. Vollständiger Name der Quelldatei. Dieser Name kann Platzhalter ("?" oder "*") enthalten.

target — Input. Vollständiger Name des Zielverzeichnisses. Es darf weder ein Zieldateiname noch ein Platzhalter angegeben werden.

verbose — Input. "Gespächigkeit" während des Kopierens. Wird dieser Parameter auf einen Wert != 0 gesetzt, wird jeder Dateiname auf stdout ausgegeben, bevor die Datei kopiert wird. Diese Option wird u.a. vom Befehl "cdcopy" verwendet.

Beispiel:

```
#include <stdio.h>
#include <cdwfsctl.h>

main()
{
APIRET ret;

/* copy complete directory tree to the CD */
ret = XCopyToCD2("c:\\os2\\*", "z:\\os2bkup", 1);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```