Linux SCSI Programmier HOWTO

Heiko Eißfeldt (heiko@colossus.escape.de) v1.5-2, 17 März 1997

Dieses Dokument behandelt die Installation und Programmierung von Linux generischem SCSI Interface.

Inhaltsverzeichnis

1 Einführung **2**

1.1 Zu diesem Dokument . . . 2

1.2 Copyright . . . 2

2 Was ist das generische SCSI Interface? **3**

3 Was sind die notwendigen Voraussetzungen? **3**

3.1 Kernel Konfiguration . . . 3

3.2 Device Deskriptoren . . . 4

3.3 Abbildung der SCSI-Geräte auf Deskriptoren . . . 4

3.3.1 Dynamisches Einfügen und Löschen von SCSI Geräten . . . 5

4 Programmierleitfaden **5**

5 Überblick der Deviceprogrammierung **6**

6 Öffnen des generischen SCSI-Devices **6**

7 Die Kopfstruktur **7**

8 Inquiry Kommandobeispiel **9**

9 Der Sensepuffer **12**

10 Beispiel mit Sensepuffer **13**

11 Ioctl Funktionen **14**

12 Treibervoreinstellungen **15**

12.1 Länge eines Transfers . . . 15

12.2 Timeout und Retry Werte . . . 15

13 Die SCSI Spezifikationen und wo sie zu bekommen sind **15**

14 Andere Informationsquellen **16**

14.1 HOWTOs und FAQs 16

14.2 Mailingliste 16

14.3 Beispielcode 16

15 Andere Nützlichkeiten **17**

15.1 Hilfen zur Gerätetreiberentwicklung 17

15.2 Hilfsprogramme 17

16 Andere SCSI Interfaces **17**

17 Schlußkommentar **17**

18 Danksagungen **18**

A Fehlerbehandlung **18**

A.1 Fehlerstatusbedeutung 19

A.2 Statuskodes 19

A.3 SCSI Sense Keys 21

A.4 Hostkodes 23

A.5 Treiberkodes 23

B Zusätzlich Prüfcodes (ASC) und zusätzliche Prüfcodequalifizierer (ASCQ) **24**

B.1 ASC und ASCQ Werte in lexikalischer Sortierung 24

B.2 ASC und ASCQ Werte in numerischer Sortierung 28

C Eine SCSI Befehlscode-Kurzübersicht **35**

D Beispielprogramme **39**

1 Einführung

1.1 Zu diesem Dokument

Dieses Dokument führt durch die Installation und Programmierung von Linux generischem SCSI Interface.

Behandelt werden Voraussetzungen im Kernel, Device Deskriptoren und das allgemeine Ansprechen von Devices. Dazu werden ein paar einfache C Beispielprogramme vorgestellt. Allgemeines Wissen über den SCSI-Befehlssatz wird hier vorausgesetzt, weitere Informationsquellen zum SCSI-Standard und allgemein zu SCSI stehen im Anhang.

Hinweis: die rohe Textvariante dieses Dokuments verfügt nicht über Querverweise (es erscheint nur “”).

1.2 Copyright

Dieses Dokument ist urheberrechtlich geschützt. Das Copyright liegt bei Heiko Eißfeldt.

Das Dokument darf gemäß der GNU *General Public License* verbreitet werden. Insbesondere bedeutet dieses, daß der Text sowohl über elektronische wie auch physikalische Medien ohne die Zahlung von Lizenzgebühren verbreitet werden darf, solange dieser Copyright Hinweis nicht entfernt wird. Eine kommerzielle Verbreitung ist erlaubt und ausdrücklich erwünscht. Bei einer Publikation in Papierform ist das Deutsche Linux HOWTO Projekt hierüber zu zu informieren.

2 Was ist das generische SCSI Interface?

Das generische SCSI Interface wurde geschrieben, um einen allgemeinen Zugriff auf beliebige SCSI Geräte zu ermöglichen (zuvor waren dazu meist Kerneländerungen notwendig). Es wurde ursprünglich von Lawrence Foard (`entropy@world.std.com`) entwickelt und von Killy Corporation gesponsert (siehe dazu die Kommentare in der Datei `scsi/sg.h`).

Dieses Interface ermöglicht spezielle Zugriffe auf SCSI-Geräte durch Benutzerapplikationen (also Codeteile, die im sogenannten Userspace laufen). Dadurch braucht kein spezieller Kerneldevicetreiber mehr entwickelt werden. Die Vorteile sind sicherere Entwicklung und Fehlerbeseitigung und Unabhängigkeit vom Kernel.

Trotzdem ist Vorsicht angebracht, falls die Applikation nicht auf Anhieb ordnungsgemäß programmiert wird. Es kann dann passieren, daß der SCSI-Bus, die Applikation oder der Kernel blockieren. Aus diesen Gründen ist es wichtig, das generische Interface richtig anzusprechen und natürlich einen Backup aller wichtigen Daten durchzuführen. Vor der Ausführung eigener Programme empfiehlt sich außerdem das Kommando `sync`, um Schreibbuffer zu leeren und einen möglichen Datenverlust zu minimieren.

Ein weiterer Vorteil dieses Interfaces ist die Unabhängigkeit von Kernelinterna. Selbst wenn innerhalb des Kernels Strukturen verändert werden, betrifft das die Applikation nur, wenn an der Schnittstelle selbst Änderungen auftreten. Bei der dynamischen Kernelentwicklung ist dies ein wichtiger Vorteil gegenüber einem Kerneldevicetreiber. Dieser muß weit häufiger an Kerneländerungen angepaßt werden.

Am häufigsten wird diese Schnittstelle verwendet, wenn spezielle, exotische oder neue SCSI-Hardware speziell angesprochen weden soll, um spezielle oder erweiterte Fähigkeiten zu benutzen. Nicht so gängige SCSI-Geräte wie Scanner, Drucker, CD-ROM Jukeboxen können dann schnell und einfach durch Applikationen bedient werden.

3 Was sind die notwendigen Voraussetzungen?

3.1 Kernel Konfiguration

Natürlich muß ein unterstützter SCSI Kontroller vorhanden sein. Außerdem muß der Kernel noch Treiber für SCSI, den/die SCSI-Hostadapter und das generische Interface beinhalten.

Die Kernelkonfiguration (mittels `make config` unter `/usr/src/linux`) sieht typischerweise so aus:

```
...
*
* SCSI support
*
SCSI support? (CONFIG_SCSI) [n] y
*
* SCSI support type (disk, tape, CDrom)
*
...
Scsi generic support (CONFIG_CHR_DEV_SG) [n] y
*
```

```
* SCSI low-level drivers
*
...
```

Falls Module möglich sind, können diese natürlich auch stattdessen verwendet werden.

3.2 Device Deskriptoren

Das generische SCSI Interface verwendet eigene Devicedeskriptoren, also nicht die der anderen SCSI Gerätetreiber. Das Skript `MAKEDEV` (zu finden im `/dev` Verzeichnis) kann die Einträge erzeugen. Der Aufruf `MAKEDEV sg` erzeugt diese Einträge:

```
crw-------   1 root     system    21,   0 Aug 20 20:09 /dev/sga
crw-------   1 root     system    21,   1 Aug 20 20:09 /dev/sgb
crw-------   1 root     system    21,   2 Aug 20 20:09 /dev/sgc
crw-------   1 root     system    21,   3 Aug 20 20:09 /dev/sgd
crw-------   1 root     system    21,   4 Aug 20 20:09 /dev/sge
crw-------   1 root     system    21,   5 Aug 20 20:09 /dev/sgf
crw-------   1 root     system    21,   6 Aug 20 20:09 /dev/sgg
crw-------   1 root     system    21,   7 Aug 20 20:09 /dev/sgh
                                    |    |
                                 Major,   Minor Device Nummern
```

Es fällt auf, daß dies Character-Deviceeinträge für Lowlevelzugriff sind.

Hinweis: Auf manchen Systemen mögen die Namen auch `/dev/{sg0,sg1,...}` lauten, die folgenden Beispiele müssen dann entsprechend angepaßt werden.

3.3 Abbildung der SCSI-Geräte auf Deskriptoren

Diese Deskriptoren werden dynamisch auf die aktiven SCSI ID/LUNs der SCSI-Busse abgebildet (LUN bedeutet logische Einheit von logical unit). Die Zuordnung (beim SCSI-Busscan) weist der Reihe nach alle LUNs von allen IDs von allen SCSI-Bussen zu. Begonnen wird bei der kleinsten LUN der kleinsten ID des ersten SCSI-Busses. Bei mehreren SCSI-Kontrollern schließen sich die folgenden Busse ohne Lücken an. Dieser Schritt findet beim Initialisieren des SCSI-Hosttreibers statt (also zum Zeitpunkt des Bootens oder beim Einfügen des Hosttreibermoduls).

Ein Beispiel: Nehmen wir an, es gibt drei angeschlossene SCSI Geräte mit den Ids 1, 3 und 5 auf dem ersten SCSI-Bus (jedes belegt eine LUN). Dann würde sich die folgende Zuordnung ergeben:

```
/dev/sga -> SCSI Id 1
/dev/sgb -> SCSI Id 3
/dev/sgc -> SCSI Id 5
```

Falls jetzt ein weiteres Gerät mit ID 4 dazukommt, verändert sich die Zuodnung (nach erneutem Busscan) zu:

```
/dev/sga -> SCSI Id 1
/dev/sgb -> SCSI Id 3
/dev/sgc -> SCSI Id 4
/dev/sgd -> SCSI Id 5
```

Beachte die Änderung bei Id 5 – das entsprechende Gerät ist nicht mehr `/dev/sgc` zugeordnet, sondern erscheint als `/dev/sgd`.

Zum Glück gibt es bei modernen Kerneln eine Möglichkeit, darauf Einfluß zu nehmen.

3.3.1 Dynamisches Einfügen und Löschen von SCSI Geräten

Wenn alls ein neuerer Kernel und das `/proc` Filesystem läuft, können Geräte, die gerade nicht benutzt werden, mitten im Betrieb ausgetragen und neu angemeldet werden. Dazu sind natürlich Superuser- privilegien erforderlich.

Zum Austragen eines angemeldeten SCSI-Gerätes wird

```
echo "scsi remove-single-device a b c d" > /proc/scsi/scsi
```

und ähnlich zum Anmelden

```
echo "scsi add-single-device a b c d" > /proc/scsi/scsi
```

verwendet, dabei ist

```
'a' die Hostadapter Id (die erste ist 0)
'b' der SCSI-Bus des Hostadapters (der erste ist 0)
'c' die ID
'd' die LUN (die erste ist 0)
```

Um also die Zuordnungen der beiden Geräte `/dev/sgc` und `/dev/sgd` zu vertauschen (siehe letztes Beispiel), geben wir Folgendes ein:

```
echo "scsi remove-single-device 0 0 4 0" > /proc/scsi/scsi
echo "scsi remove-single-device 0 0 5 0" > /proc/scsi/scsi
echo "scsi add-single-device 0 0 5 0" > /proc/scsi/scsi
echo "scsi add-single-device 0 0 4 0" > /proc/scsi/scsi
```

Das klappt, weil die Gerätezuordnung nach Anmeldereihenfolge erfolgt.

Beim Anmelden neuer SCSI-Geräte ist zu beachten, daß nur eine begrenzte Anzahl von freien Einträgen zur Verfügung steht. Der Speicher ist zur Bootzeit reserviert worden und erlaubt zwei weitere Gerätezuordnungen.

4 Programmierleitfaden

Die folgenden Kapitel sind für ProgrammiererInnen gedacht, die das Interface in ihren eigenen Applikationen einsetzen wollen. Ein Beispiel führt den INQUIRY Befehl durch, ein anderes benutzt das TESTUNITREADY Kommando.

Bitte beachte die folgenden Punkte zu diesen Codebeispielen;

- Die Headerfiles `sg.h` und `scsi.h` sind ab der Kernelversion 1.3.98 auf einen anderen Platz gewandert. Diese Dateien stehen jetzt unter `/usr/src/linux/include/scsi`. Sie sollten über einen symbolischen Link von `/usr/include/scsi` zu erreichen ist. Zuvor standen sie unter `/usr/src/linux/drivers/scsi`. Wir setzen im Folgenden neuere Kernel voraus.
- Ab Kernelversion 1.1.68 ist das generische SCSI Interface erweitert worden; die Beispiele erfordern mindestens die Version. Außerdem sind die folgenden Kernelversionen wegen nichtfunktionierender Schnittstelle zu vermeiden: 1.1.77 bis 1.1.89 und 1.3.52 bis 1.3.56.
- Die Konstante DEVICE im Kopfteil beschreibt das verwendete SCSI-Gerät und sollte vorher auf das gewünschte der verfügbaren Geräte gesetzt werden (siehe auch Kapitel 7.

5 Überblick der Deviceprogrammierung

Das Headerfile `include/scsi/sg.h` enthält die Beschreibung des Interface (dies basiert auf Kernelversion 1.3.98):

```
struct sg_header
 {
  int pack_len;    /* Länge des eingehenden Pakets (mit Header) */
  int reply_len;   /* Maximallänge des erwarteten Antwortpakets */
  int pack_id;     /* Id Nummer des Pakets */
  int result;      /* Ergebnis 0 bedeutet ok, sonst einer der errno Codes */
  unsigned int twelve_byte:1;
               /* Erzwinge 12 byte Kommandolänge für Gruppe 6 & 7
Kommandos  */
  unsigned int other_flags:31;                  /* für Erweiterungen */
  unsigned char sense_buffer[16]; /* nur bei Antwortpaketen benutzt */
  /* Als nächstes folgen direkt der Block mit dem SCSI-Kommando und
eventuell Daten zum Befehl */
 };
```

Diese Struktur legt fest, wie ein SCSI Kommando bearbeitet werden soll und nimmt das Ergebnis nach erfolgter Abarbeitung auf. Die einzelnen Komponenten werden später in Kapitel 7 erläutert.

Die allgemeine Prozedur zum Datenaustausch läuft wie folgt: ein Kommando wird mit `write()` an ein geöffnetes Gerät geschickt. Der Block dazu enthält diese drei Abschnitte:

```
struct sg_header
SCSI Kommando
zu schickende Daten zu diesem Kommando
```

Das Ergebnis eines Kommandos wird mit `read()` gelesen. Der Block hat eine ähnliche Struktur:

```
struct sg_header
gelieferte Daten vom SCSI-Gerät
```

Dies war der Überblick der Prozedur. Die folgenden Kapitel beschreiben die einzelnen Schritte detaillierter.

Hinweis: Bis hin zu aktuellen Kerneln ist es notwendig, das SIGINT signal zwischen `write()` und entsprechendem `read()` Aufruf zu blockieren (z.B. mittels `sigprocmask()`). Eine Rückkehr nach der `write()` Hälfte ohne den `read()` Aufruf führt zur Blockade bei nachfolgenden Zugriffen. Diese Behandlung ist in den Beispielen nicht enthalten. Es sollte daher beim Laufenlassen der Beispiele auf das Signal SIGINT (Drücken von ^C) verzichtet werden.

6 Öffnen des generischen SCSI-Devices

Ein generisches SCSI-Device muß für Lese- und Schreibzugriff geöffnet werden:

```
int fd = open (device_name, O_RDWR);
```

(Das gilt sogar für reine Nurlesegeräte wie zum Beispiel CDROM-Laufwerke).

Wir müssen dann einen `write` Aufruf mit dem Kommando abschicken und einen `read` Aufruf, um die Ergebnisse abzuholen. Falls Fehler auftreten, ist der Rückgabewert des `read` negativ (siehe Kapitel A für die Liste der Fehlercodes).

7 Die Kopfstruktur

Die Kopfstruktur `struct sg_header` dient als Übergabeschnittstelle zwischen Applikation und Kerneltreiber. Jetzt kommen wir zu den einzelnen Komponenten dieser Struktur.

int pack_len

definiert die Größe des an den Treiber gesendeten Blocks. Das Feld wird vom Kernel zur internen Verwaltung gefüllt.

int reply_len

definiert die Größe des erwarteten Antwortblocks. Das Feld wird von der Applikation bestimmt.

int pack_id

Dieses Feld dient zum Wiederfinden einer Antwort zur gestarteten Anfrage. Die Applikation kann einen neuen ID für jedes Kommando angeben. Die Antworten sind dann den Kommandos zuzuordnen, auch wenn sie in anderer Reihenfolge erscheinen. Bis zu SG_MAX_QUEUE (z.B. 4) parallele Kommandos sind möglich.

int result

Der Rückgabewert eines `read` oder `write` Aufrufs. Dieses Feld sollte vom Kernel definiert werden (leider nicht immer). Daher ist es am besten, das Feld vor den `write` Aufrufen explizit auf Null zu setzen. Die Codes sind in `errno.h` aufgeführt (0 bedeutet kein Fehler).

unsigned int twelve_byte:1

Das Feld wird nur bei Nichtstandardbefehlen (vendor specific Kommandos) im Bereich 0xc0 - 0xff ausgewertet. Wenn Kommandos aus diesem Bereich eine Länge von 12 Bytes statt 10 Bytes (ohne eventuelle Daten) haben, muß dieses Feld vor dem `write` Aufruf auf Eins gesetzt werden. Andere Längen werden nicht unterstützt. Dieses Feld wird von der Applikation gefüllt.

unsigned char sense_buffer[16]

Dieser Puffer wird nach Abarbeitung eines Kommandos (`read()` Aufruf) vom Kernel gefüllt und enthält den sogenannten SCSI sense code. Manche SCSI Kommandoergebnisse stehen hier (z.B. die des `TESTUNITREADY` Kommandos). Ansonsten enthält der Puffer Nullen.

Die folgende Beispielfunktion stellt den Kontakt mit dem Interface her. Erst wird die Kopfstruktur definiert, dann der Befehl per `write` geschickt, das Ergebnis per `read` abgeholt und Fehlerstati gecheckt. Der Sensepuffer ist im Ausgabepuffer untergebracht, es sei denn ein NULL-Zeiger wurde angegeben, dann wird der Eingabepuffer verwendet. Wir werden ihn in einem der Beispiele verwenden.

Hinweis: Setze den Wert für `DEVICE` auf einen Deiner Gerätedeskriptoren.

```
#define DEVICE "/dev/sgc"

/* Beispielprogramm zur Demonstration des generischen SCSI-Interface */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <scsi/sg.h>


#define SCSI_OFF sizeof(struct sg_header)
```

```
static unsigned char cmd[SCSI_OFF + 18];      /* SCSI Kommandopuffer */
int fd;                                 /* SCSI device/file-Deskriptor */

/* einen kompletten SCSI Befehl abarbeiten. Verwende das generische
SCSI-Interface. */
static int handle_SCSI_cmd(unsigned cmd_len,         /* Kommandolänge */
                           unsigned in_size,         /* Eingabedatengröße */
                           unsigned char *i_buff,    /* Eingabepuffer */
                           unsigned out_size,        /* Ausgabedatengröße */
                           unsigned char *o_buff     /* Ausgabepuffer */
                           )
{
    int status = 0;
    struct sg_header *sg_hd;

    /* Sicherheitsüberprüfungen */
    if (!cmd_len) return -1;            /* erfordert cmd_len != 0 */
    if (!i_buff) return -1;             /* erfordert Eingabepuffer != NULL */
#ifdef SG_BIG_BUFF
    if (SCSI_OFF + cmd_len + in_size > SG_BIG_BUFF) return -1;
    if (SCSI_OFF + out_size > SG_BIG_BUFF) return -1;
#else
    if (SCSI_OFF + cmd_len + in_size > 4096) return -1;
    if (SCSI_OFF + out_size > 4096) return -1;
#endif

    if (!o_buff) out_size = 0;      /* kein Ausgabepuffer, keine
Ausgabegröße */

    /* Aufbau der generischen SCSI-Device Kopfstruktur */
    sg_hd = (struct sg_header *) i_buff;
    sg_hd->reply_len   = SCSI_OFF + out_size;
    sg_hd->twelve_byte = cmd_len == 12;
    sg_hd->result = 0;
#if     0
    sg_hd->pack_len    = SCSI_OFF + cmd_len + in_size; /* nicht notwendig */
    sg_hd->pack_id;     /* unbenutzt */
    sg_hd->other_flags; /* unbenutzt */
#endif

    /* Befehl schicken */
    status = write( fd, i_buff, SCSI_OFF + cmd_len + in_size );
    if ( status < 0 || status != SCSI_OFF + cmd_len + in_size ||
                       sg_hd->result ) {
        /* ein Fehler ist aufgetreten */
        fprintf( stderr, "write(generic) result = 0x%x cmd = 0x%x\n",
                    sg_hd->result, i_buff[SCSI_OFF] );
        perror("");
        return status;
    }

    if (!o_buff) o_buff = i_buff;       /* Pufferpointer checken */
    sg_hd = (struct sg_header *) o_buff;

    /* Ergebnis abholen */
```

```
    status = read( fd, o_buff, SCSI_OFF + out_size);
    if ( status < 0 || status != SCSI_OFF + out_size || sg_hd->result ) {
        /* ein Fehler ist aufgetreten */
        fprintf( stderr, "read(generic) status = 0x%x, result = 0x%x, "
                         "cmd = 0x%x\n",
                         status, sg_hd->result, o_buff[SCSI_OFF] );
        fprintf( stderr, "read(generic) sense "
                "%x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x\n",
                sg_hd->sense_buffer[0],         sg_hd->sense_buffer[1],
                sg_hd->sense_buffer[2],         sg_hd->sense_buffer[3],
                sg_hd->sense_buffer[4],         sg_hd->sense_buffer[5],
                sg_hd->sense_buffer[6],         sg_hd->sense_buffer[7],
                sg_hd->sense_buffer[8],         sg_hd->sense_buffer[9],
                sg_hd->sense_buffer[10],        sg_hd->sense_buffer[11],
                sg_hd->sense_buffer[12],        sg_hd->sense_buffer[13],
                sg_hd->sense_buffer[14],        sg_hd->sense_buffer[15]);
        if (status < 0)
            perror("");
    }
    /* Nachsehen, ob wir bekommen haben, was wir wollten */
    if (status == SCSI_OFF + out_size) status = 0; /* alles bekommen */

    return status;  /* 0 bedeutet kein Fehler */
}
```

Auf den ersten Blick mag dies abschreckend wirken, jedoch dient der meiste Code zur Überprüfung und zum Melden von Fehlern. Das ist nützlich, selbst wenn der Code fehlerfrei gemacht wurde.

`Handle_SCSI_cmd` hat eine generalisierte Form für alle Arten von SCSI Kommandos, was den Transfer von Daten angeht. Es gibt diese Kategorien:

```
       Datenmodus                        | Beispiel Kommando
=============================================================
weder Eingabe- noch Ausgabe-Daten        | test unit ready
keine Eingabedaten, aber Ausgabedaten    | inquiry, read
Eingabedaten, aber keine Ausgabedaten    | mode select, write
Eingabedaten und Ausgabedaten            | mode sense
```

8 Inquiry Kommandobeispiel

Eines der elementarsten SCSI-Kommandos ist das `INQUIRY` Kommando, das zur Identifikation des SCSI-Geräts verwendet wird. Hier ist die Definition aus der SCSI-2 Spezifikation (weitere Details sind dem SCSI-2 Standard zu entnehmen).

Tabelle 44: INQUIRY Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (12h)							
1	Logical Unit Number			Reserved				EVPD

2	Page Code							
3	Reserved							
4	Allocation Length							
5	Control							

Die Ausgabedaten sind dann wie folgt:

Tabelle 45: Standard INQUIRY Data Format

Bit / Byte	7	6	5	4	3	2	1	0
0	Peripheral Qualifier			Peripheral Device Type				
1	RMB	Device-Type Modifier						
2	ISO Version		ECMA Version			ANSI-Approved Version		
3	AENC	TrmIOP	Reserved			Response Data Format		
4	Additional Length (n-4)							
5	Reserved							
6	Reserved							
7	RelAdr	WBus32	WBus16	Sync	Linked	Reserved	CmdQue	SftRe
8	(MSB)							
- -	---	Vendor Identification						---
15								(LSB)
16	(MSB)							
- -	---	Product Identification						---
31								(LSB)
32	(MSB)							
- -	---	Product Revision Level						---
35								(LSB)
36								
- -	---	Vendor Specific						---
55								
56								
- -	---	Reserved						---
95								
	Vendor-Specific Parameters							
96								

```
|- - -+---                          Vendor Specific                          ---|
| n   |                                                                          |
+==========================================================================+
```

Das folgende Beispiel benutzt die Lowlevel-Funktion `handle_SCSI_cmd` zum Abschicken des Inquiry SCSI Kommandos.

Zuerst hängen wir den Kommandoblock an den generischen Header, dann rufen wir `handle_SCSI_cmd` auf. Bitte beachte, daß der Parameter für die Größe des Ausgabepuffers die Kopfstrukturgröße nicht beinhaltet. Nach Abarbeitung des Kommandos enthält der Ausgabepuffer die angeforderte Information, falls kein Fehler vorlag.

```
#define INQUIRY_CMD     0x12
#define INQUIRY_CMDLEN  6
#define INQUIRY_REPLY_LEN 96
#define INQUIRY_VENDOR  8       /* Versatz zum Herstellernamen im Antwortblock */

/* Herstellernamen und Typ anfordern */
static unsigned char *Inquiry ( void )
{
  unsigned char Inqbuffer[ SCSI_OFF + INQUIRY_REPLY_LEN ];
  unsigned char cmdblk [ INQUIRY_CMDLEN ] =
      { INQUIRY_CMD,  /* Kommando */
                0,  /* lun/reserviert */
                0,  /* page code */
                0,  /* reserviert */
  INQUIRY_REPLY_LEN,  /* Allokationslänge */
                0 };/* reserviert/flag/link */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- cmd
   * +------------------+
   * | Kopie von cmdblk | <- cmd + SCSI_OFF
   * +------------------+
   */

  if (handle_SCSI_cmd(sizeof(cmdblk), 0, cmd,
                      sizeof(Inqbuffer) - SCSI_OFF, Inqbuffer )) {
      fprintf( stderr, "Inquiry scheiterte\n" );
      exit(2);
  }
  return (Inqbuffer + SCSI_OFF);
}
```

Das oben angegebene Beispiel hält sich an diese Struktur. Die Inquiry-Funktion kopiert ihren Kommandoblock hinter den generischen Kopf (durch `SCSI_OFF` gegeben). Dieser Befehl hat keine weiteren Daten zu übergeben. Funktion `Handle_SCSI_cmd` erzeugt daraus die Kopfstruktur.

Um das Beispiel vollständig zu machen, implementieren wir nun noch die Funktion `main`.

```
int main( void )
{
  fd = open(DEVICE, O_RDWR);
```

```
  if (fd < 0) {
    fprintf( stderr, "Lese-Schreiberlaubnis erforderlich für "DEVICE".\n"
);
    exit(1);
  }

  /* ein paar Felder des Inquiry-Ergebnisses ausgeben */
  printf( "%s\n", Inquiry() + INQUIRY_VENDOR );
  return 0;
}
```

Zuerst öffnen wir das Gerät, prüfen auf Fehler und rufen dann das Unterprogramm Inquiry auf. Die Ergebnisse Hersteller, Produkt und Version werden schließlich in lesbarer Form ausgegeben.

Hinweis: Der Inquiry-Befehl liefert noch weitere Informationen, mehr als diese kleine Programm ausgibt. Vielleicht möchtest Du es so erweitern, daß Gerätetyp, ANSI-Version usw. behandelt werden. Der Gerätetyp ist zum Beispiel von besonderer Bedeutung, da er für das Gerät den festen (minimalen) und erweiterten Befehlsumfang festlegt. Wenn Du das nicht selbst programmieren möchtest, gibt es auch noch das Programm scsiinfo von Eric Youngdale. Es liefert nahezu alle Angaben zu einem SCSI-Gerät. Du findest es per FTP unter:

```
tsx-11.mit.edu:/pub/Linux/ALPHA/scsi
```

9 Der Sensepuffer

SCSI-Befehle ohne Ausgabedaten können Statusinformationen über den Sensepuffer zurückgeben (er befindet sich in der Kopfstruktur). Diese Daten (Sensedaten) sind definiert wenn das vorangehende Kommando mit dem Status CHECK CONDITION beendet wurde. In diesem Fall holt der Kernel automatisch die Sensedaten mit einem REQUEST SENSE Kommando und füllt sie in den Sensepuffer. Die Struktur des Puffer ist:

Bit / Byte	7	6	5	4	3	2	1	0
0	Valid	Error Code (70h or 71h)						
1	Segment Number							
2	Filemark	EOM	ILI	Reserved	Sense Key			
3	(MSB)							
- -	---	Information						---
6								(LSB)
7	Additional Sense Length (n-7)							
8	(MSB)							
- -	---	Command-Specific Information						---
11								(LSB)
12	Additional Sense Code							
13	Additional Sense Code Qualifier							

14				Field Replaceable Unit Code				
15	SKSV							
17				Sense-Key Specific				
18								
n				Additional Sense Bytes				

Hinweis: Die wichtigsten Felder sind der 'Sense Key' (siehe auch Kapitel A.3), 'Additional Sense Code' and 'Additional Sense Code Qualifier' (siehe auch Kapitel B). Die beiden letzten Felder werden zusammen als ein Paar benutzt.

10 Beispiel mit Sensepuffer

Wir führen ein TEST UNIT READY Kommando durch, um zu überprüfen, ob ein Medium im Gerät geladen ist. Die Kopf-Deklarationen und die Funktion `handle_SCSI_cmd` aus dem Inquiry-Bespiel werden hier auch benötigt.

Tabelle 73: TEST UNIT READY Command

Bit / Byte	7	6	5	4	3	2	1	0
0				Operation Code (00h)				
1	Logical Unit Number				Reserved			
2				Reserved				
3				Reserved				
4				Reserved				
5				Control				

Mit dieser Funktion implementieren wir die Abfrage.

```
#define TESTUNITREADY_CMD 0
#define TESTUNITREADY_CMDLEN 6

#define ADD_SENSECODE 12
#define ADD_SC_QUALIFIER 13
#define NO_MEDIA_SC 0x3a
#define NO_MEDIA_SCQ 0x00

int TestForMedium ( void )
{
  /* request READY status */
  static unsigned char cmdblk [TESTUNITREADY_CMDLEN] = {
      TESTUNITREADY_CMD, /* command */
                    0, /* lun/reserved */
```

```
                        0, /* reserved */
                        0, /* reserved */
                        0, /* reserved */
                        0};/* control */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- cmd
   * +------------------+
   * | copy of cmdblk   | <- cmd + SCSI_OFF
   * +------------------+
   */

  if (handle_SCSI_cmd(sizeof(cmdblk), 0, cmd,
                            0, NULL)) {
      fprintf (stderr, "Test unit ready scheiterte\n");
      exit(2);
  }

  return
   *(((struct sg_header*)cmd)->sense_buffer +ADD_SENSECODE) !=
                                                    NO_MEDIA_SC ||
   *(((struct sg_header*)cmd)->sense_buffer +ADD_SC_QUALIFIER) !=
                                                    NO_MEDIA_SCQ;
}
```

Mit einer neuen `main` Funktion können wir die Abfrage laufen lassen.

```
int main( void )
{
  fd = open(DEVICE, O_RDWR);
  if (fd < 0) {
    fprintf( stderr, "Lese-Schreiberlaubnis erforderlich für "DEVICE".\n"
);
    exit(1);
  }

  /* Abfrage, ob sich ein Medium im Gerät befindet */

  if (!TestForMedium()) {
    printf("Gerät enthält kein Medium\n");
  } else {
    printf("Gerät enthält ein Medium\n");
  }
  return 0;
}
```

Die Datei `generic_demo.c` im Anhang enthält beide Beispiele.

11 Ioctl Funktionen

Zwei Ioctl-Funktionen stehen zur Verfügung:

- `ioctl(fd, SG_SET_TIMEOUT, &Timeout);` setzt den Timeout-Wert auf `Timeout` * 10 Millisekunden. `Timeout` muß als int deklariert werden.
- `ioctl(fd, SG_GET_TIMEOUT, &Timeout);` ermittelt den aktuellen Timeoutwert. `Timeout` muß als int deklariert werden.

12 Treibervoreinstellungen

12.1 Länge eines Transfers

Zur Zeit (zumindest bis zur Kernelversion 1.1.68) müssen Ein- und Ausgabedaten kleiner als 4097 Bytes sein, es sei denn der Kernel ist mit definiertem Symbol `SG_BIG_BUFF` kompiliert worden. In diesem Fall sind die Größen auf diesen Wert begrenzt (meistens 32768 Bytes). Die Größen beinhalten die generische Kopfstruktur, bei Eingaben den Kommandoblock und mögliche Daten. `SG_BIG_BUFF` kann bis auf 131072 Bytes vergrößert werden. Damit die Änderung wirksam wird, muß der Kernel natürlich kompiliert und gebootet werden.

12.2 Timeout und Retry Werte

Der Standardtimeoutwert ist auf eine Minute (`Timeout` = 6000) gesetzt. Er kann mit einem ioctl-Aufruf geändert werden (see section 11). Der Standardretrywert ist Eins.

13 Die SCSI Spezifikationen und wo sie zu bekommen sind

Standards für SCSI-1, SCSI-2 und SCSI-3 sind verfügbar. Sie sind weitestgehend aufwärtskompatibel.

Der SCSI-1 Standard ist (nach Meinung des Autors) weitgehend hinfällig, SCSI-2 ist der meist benutzte. SCSI-3 ist noch sehr neu und entsprechend teuer. Die standarisierten Befehlssätze unterteilen in zwingend vorgeschriebene und optionale Befehle. SCSI-Gerätehersteller können darüber hinaus eigene Befehle implementieren. Die optionalen bzw. herstellerspezifischen Befehle sind nach Möglichkeit zu vermeiden, wenn das Programm möglichst viele Geräte bedienen können soll (Portabilität). Desweiteren sind Informationen zu herstellerspezifischen Befehlen schwer zu bekommen. Es gibt natürlich Fälle, in denen darauf nicht verzichtet werden kann.

Kopien der letzten Draftversionen der SCSI-Standards sind hier per anonymem ftp verfügbar:

- `ftp.cs.tulane.edu:/pub/scsi`
- `ftp.symbios.com:/pub/standards`
- `ftp.cs.uni-sb.de:/pub/misc/doc/scsi`

(Ich benutzte die SCSI Spezifikation von meiner Yggdrasil Linux CD-ROM im Verzeichnis `/usr/doc/scsi-2` und `/usr/doc/scsi-1`).

Das *SCSI FAQ* beinhaltet die folgenden Bezugsquellen für englisches, gedrucktes Material:

Die SCSI Spezifikation:

```
Global Engineering Documents
15 Inverness Way East
Englewood Co  80112-5704
(800) 854-7179
  SCSI-1: X3.131-1986
```

```
        SCSI-2: X3.131-199x
        SCSI-3 X3T9.2/91-010R4 Working Draft

(Global Engineering Documentation in Irvine, CA (714)261-1455??)

SCSI-1: Doc \# X3.131-1986 von ANSI, 1430 Broadway, NY, NY 10018

IN-DEPTH EXPLORATION OF SCSI ist erhältlich von
Solution Technology, Attn: SCSI Publications, POB 104, Boulder Creek,
CA 95006, (408)338-4285, FAX (408)338-4374

THE SCSI ENCYLOPEDIA und das SCSI BENCH REFERENCE sind erhältlich bei
ENDL Publishing, 14426 Black Walnut Ct., Saratoga, CA 95090,
(408)867-6642, FAX (408)867-2115

SCSI: UNDERSTANDING THE SMALL COMPUTER SYSTEM INTERFACE wurde von
Prentice-Hall veröffentlicht, ISBN 0-13-796855-8
```

Hinweise auf gute deutsche Publikationen sind erwünscht.

14 Andere Informationsquellen

14.1 HOWTOs und FAQs

Das Linux *SCSI HOWTO* von Drew Eckhardt behandelt alle unterstützten SCSI-Kontroller und gerätespezifische Fragen. Es werden eine Menge Tips zur Fehlerbehebung gegeben. Es ist bei `metalab.unc.edu` in `/pub/Linux/docs/LDP` und seinen Mirrorsites verfügbar.

Allgemeine Fragen zum Thema SCSI werden im *SCSI-FAQ* und der Newsgruppe

```
comp.periphs.scsi
```

beantwortet (verfügbar auf `tsx-11.mit.edu` in `pub/linux/ALPHA/scsi` und den Mirrorsites).

14.2 Mailingliste

Es gibt eine **Mailingliste** für Bugreports und Fragen zur SCSI-Entwicklung unter Linux. Um sich eintragen zu lassen, schick eine email an `majordomo@vger.rutgers.edu` mit der Zeile `subscribe linux-scsi` in der Email. Beiträge sollten an `linux-scsi@vger.rutgers.edu` geschickt werden. Ein englischer Hilfstext kann durch Schicken der Zeile 'help' an `majordomo@vger.rutgers.edu` erhalten werden.

14.3 Beispielcode

metalab.unc.edu: apps/graphics/hpscanpbm-0.3a.tar.gz

Dieses Paket bedient einen HP Scanjet Scanner mit dem generischen SCSI Interface.

tsx-11.mit.edu: BETA/cdrom/private/mkisofs/cdwrite-1.3.tar.gz

Das cdwrite Paket benutzt das generische SCSI Interface zum Schreiben von CD-Images auf einen CD-Brenner.

ftp.gwdg.de:/pub/linux/misc/cdda2wav/cdda2wav*.src.tar.gz

Dies ist meine eigene Applikation, die Audio-CD Tracks in WAV-Dateien kopiert.

15 Andere Nützlichkeiten

Tools, die für Entwickler nützlich sein können. Falls neuere oder bessere Versionen/Varianten verfügbar sind, schreibt mir bitte.

15.1 Hilfen zur Gerätetreiberentwicklung

Alessandro Rubini hat ein (englisches) Buch zum Thema Treiberentwicklung unter Linux (Verlag O'Reilly) geschrieben (ab November 1997).

15.2 Hilfsprogramme

tsx-11.mit.edu: ALPHA/scsi/scsiinfo*.tar.gz

Ein Programm zum Abfragen der Betriebsparameter für SCSI-Geräte wie z.B. Defektlisten, usw. Es gibt ein grafisches X11-Frontend, das Tk/Tcl/wish voraussetzt. Damit können Einstellungen auch einfach verändert werden.

tsx-11.mit.edu: ALPHA/kdebug

Eine gdb Erweiterung für Kerneldebugging.

16 Andere SCSI Interfaces

In Linux existiert noch eine veraltete andere Zugriffsart, um auf SCSI Geräte zuzugreifen. Es gibt einen ioctl Befehl (SCSI_IOCTL_SEND_COMMAND), der früher dazu verwendet wurde. Spezielle Programme wie 'scsiinfo' verwenden ihn noch.

Es gibt in Unixwelt noch andere Interfaces zu einem ähnlichem Zweck, die unter Linux allerdings nicht verfügbar sind:

1. CAM (Common Access Method) (von Future Domain entwickelt). Linux hat nur in soweit Support dafür, wie es zum Booten von einer Festplatte enötigt wird. Allgemein unterstützt die CAM-Spezifikation sogar den sogenannten 'target mode', so daß der Linuxrechner in die Rolle eines Peripheriegeräts schlüpfen könnte. Dieses Feature ist für ein kleines 'SCSI-Netz' notwendig.
2. ASPI (Advanced SCSI Programming Interface) (entwickelt von Adaptec). Dies ist wohl der de facto Standard für MS-DOS Rechner geworden. Die Spezifikation ist wesentlich weniger allgemein als die von CAM.

Außerdem gibt es dann noch die SCSI-Interfaces von SCO(TM), NeXT(TM), Silicon Graphics(TM) und SUN(TM).

17 Schlußkommentar

Das generische SCSI-Interface schließt die Lücke zwischen Benutzerapplikationen und den SCSI-Geräten. Statt jede dieser Applikationen mit eigenen (ähnlichen) Routinen zur Behandlung von bestimmten Geräten zu versehen, wäre die Entwicklung von allgemeinen Libraries mit generischen Funktionen zu bevorzugen. Ein Hauptziel sollte die Entwicklung von unabhängigen Schichten mit wohldefinierten Interfaces sein. Ein gutes Design würde innerhalb einer Applikation zwischen geräteunabhängigen Funktionen und allgemein verwendbaren Routinen zum Ansprechen der Hardware trennen. Die Hardware-Routinen könnten in eine Library verpackt werden und anderen Applikationen zur Verfügung gestellt werden. Standarisierte Interfaces sollten dabei natürlich bevorzugt werden.

Tja, nun weißt Du langsam mehr als ich über Linux generisches SCSI Interface. Jetzt sollte Dir nichts mehr im Wege stehen, um für die globale Linux-Gemeinde gute SCSI Applikationen zu entwickeln.

18 Danksagungen

Eine besonderen Dank an Jeff Tranter, der mir bei der englischen Version sehr geholfen hat und an Carlos Puchol für nützliche Hinweise. Drew Eckhardts und Eric Youngdales Hilfe bei meinen ersten unbeholfenen Versuchen und Fragen zur Benutzung dieses Interfaces war sehr willkommen.

A Fehlerbehandlung

Die Funktionen `open`, `ioctl`, `write` und `read` können Fehler melden. In diesen Fällen ist der Rückgabewert -1 und die globale Variable errno enthält die Fehlernummer. Die Werte dieser Variable sind definiert in `/usr/include/errno.h`. Die möglichen Werte sind:

Funktion	Fehler	Beschreibung
open	ENXIO	kein gültiges Gerät
	EACCES	der Zugriffsmodus ist nicht read/write (O_RDWR)
	EBUSY	Gerät soll nicht blockieren,
		ist aber in Benutzung.
	ERESTARTSYS	Dies wäre ein interner Fehler im Kernel.
		Er sollte reproduzierbar sein und an die SCSI
		Mailingliste geschickt werden
		(siehe Drew Eckhardts SCSI-HOWTO).
ioctl	ENXIO	kein gültiges Gerät
read	EAGAIN	das Gerät würde jetzt blockieren. Bitte
nochmal		
		versuchen
	ERESTARTSYS	Dies wäre ein interner Fehler im Kernel.
		Er sollte reproduzierbar sein und an die SCSI
		Mailingliste geschickt werden
		(siehe Drew Eckhardts SCSI-HOWTO).
write	EIO	die Länge ist zu klein (kleiner als die
Größe		
		des generischen Kopfes). Achtung: z.Z. gibt es
		keine Absicherung gegen zu große Längen!
	EAGAIN	das Gerät würde jetzt blockieren. Bitte
nochmal		
		versuchen
	ENOMEM	der notwendige Speicher konnte nicht allokiert
		werden. Bitte nochmal versuchen, es sei denn
		die Maximalgröße wurde überschritten
(siehe oben)		
select		keine
close		keine

Bei read/write bedeuten positive Rückgabewerte wie üblich die Anzahl erfolgreich übertragener Bytes. Das sollte dem entsprechen, was übertragen werden sollte.

A.1 Fehlerstatusbedeutung

Weiterhin existiert ein detaillierteria the kernels `hd_status` and the devices `sense_buffer` (see section 9) both from the generic header structure.

Die Bedeutung von `hd_status` kann in `drivers/scsi/scsi.h` nachgelesen werden. Der Wert vom Typ `unsigned int` setzt sich aus verschiedenen Teilen zusammen. Dabei steht lsb (least significant byte) für das niederwertigste Byte und msb für das höchstwertigste Byte.

lsb	...	...	msb
status	sense key	host code	driver byte

Diese Makros aus `drivers/scsi/scsi.h` sind definiert, können aber leider nicht so einfach verwendet werden (wegen unsauberer Headerdatei- Abhängigkeiten). Dies muß noch bereinigt werden.

Makro	Beschreibung
status_byte(hd_status)	Der SCSI-Gerätestatus. Siehe Section Statuskodes
msg_byte(hd_status)	Vom SCSI-Gerät. Siehe Sektion SCSI sense keys
host_byte(hd_status)	Vom Kernel. Siehe Sektion Hostkodes
driver_byte(hd_status)	Vom Kernel. Siehe Sektion midlevel codes

A.2 Statuskodes

Vom SCSI-Gerät können folgende Statuskodes (defined in `scsi/scsi.h`) verwendet werden.

Wert	Symbol
0x00	GOOD
0x01	CHECK_CONDITION
0x02	CONDITION_GOOD
0x04	BUSY
0x08	INTERMEDIATE_GOOD
0x0a	INTERMEDIATE_C_GOOD
0x0c	RESERVATION_CONFLICT

Bitte beachten, daß diese Werte **einmal nach rechts geschoben wurden**. Wenn der Statuskode CHECK_CONDITION ist, stehen gültige Sensedaten im Sensepuffer zur Verfügung (vor allem zum Prüfen der additional sense code (ASC) und additional sense code qualifier (ASCQ) Größen).

Die Werte entsprechen diesen Definitionen aus der SCSI-2 Spezifikation:

Tabelle 27: Status Byte Code

Bits of Status Byte								Status
7	6	5	4	3	2	1	0	
R	R	0	0	0	0	0	R	GOOD
R	R	0	0	0	0	1	R	CHECK CONDITION
R	R	0	0	0	1	0	R	CONDITION MET
R	R	0	0	1	0	0	R	BUSY
R	R	0	1	0	0	0	R	INTERMEDIATE

R	R	0	1	0	1	0	R	INTERMEDIATE-CONDITION MET
R	R	0	1	1	0	0	R	RESERVATION CONFLICT
R	R	1	0	0	0	1	R	COMMAND TERMINATED
R	R	1	0	1	0	0	R	QUEUE FULL
All Other Codes								Reserved
Key: R = Reserved bit								

A definition of the status byte codes is given below.

GOOD. This status indicates that the target has successfully completed the command.

CHECK CONDITION. This status indicates that a contingent allegiance condition has occurred (see 6.6).

CONDITION MET. This status or INTERMEDIATE-CONDITION MET is returned whenever the requested operation is satisfied (see the SEARCH DATA and PRE-FETCH commands).

BUSY. This status indicates that the target is busy. This status shall be returned whenever a target is unable to accept a command from an otherwise acceptable initiator (i.e., no reservation conflicts). The recommended initiator recovery action is to issue the command again at a later time.

INTERMEDIATE. This status or INTERMEDIATE-CONDITION MET shall be returned for every successfully completed command in a series of linked commands (except the last command), unless the command is terminated with CHECK CONDITION, RESERVATION CONFLICT, or COMMAND TERMINATED status. If INTERMEDIATE or INTERMEDIATE-CONDITION MET status is not returned, the series of linked commands is terminated and the I/O process is ended.

INTERMEDIATE-CONDITION MET. This status is the combination of the CONDITION MET and INTERMEDIATE statuses.

RESERVATION CONFLICT. This status shall be returned whenever an initiator attempts to access a logical unit or an extent within a logical unit that is reserved with a conflicting reservation type for another SCSI device (see the RESERVE and RESERVE UNIT commands). The recommended initiator recovery action is to issue the command again at a later time.

COMMAND TERMINATED. This status shall be returned whenever the target terminates the current I/O process after receiving a TERMINATE I/O PROCESS message (see 5.6.22). This status also indicates that a contingent allegiance condition has occurred (see 6.6).

QUEUE FULL. This status shall be implemented if tagged queuing is implemented. This status is returned when a SIMPLE QUEUE TAG, ORDERED QUEUE TAG, or HEAD OF QUEUE TAG message is received and the command queue is full. The I/O process is not placed in the command queue.

A.3 SCSI Sense Keys

Diese symbolischen Konstanten aus der Datei `scsi/scsi.h` sind vordefiniert.

```
Wert   | Symbol
======|================
0x00   | NO_SENSE        (es liegt keine Senseinformation vor)
0x01   | RECOVERED_ERROR (fehlerfrei nach Widerholung wegen Fehler)
0x02   | NOT_READY       (Gerät ist (noch) nicht bereit)
0x03   | MEDIUM_ERROR    (das Medium ist beschädigt)
0x04   | HARDWARE_ERROR  (die Hardware des Geräts ist beschädigt)
0x05   | ILLEGAL_REQUEST (nichtdurchführbarer Befehl)
0x06   | UNIT_ATTENTION  (Gerät erfordert Anfrage oder Eingriff)
0x07   | DATA_PROTECT    (wegen eines Schutzes erfolgt kein Datenzugriff)
0x08   | BLANK_CHECK     (z.B. ein CD-Brenner findet kein neues Medium vor)
0x0a   | COPY_ABORTED    (ein COPY oder COMPARE-Kommando wurde abgebrochen)
0x0b   | ABORTED_COMMAND (das Kommando wurde abgebrochen)
0x0d   | VOLUME_OVERFLOW (Überlauf eines Volumes/einer Partition)
0x0e   | MISCOMPARE      (ein Vergleich (VERIFY) lieferte Unterschiede)
```

Hier ist die Liste aus den SCSI-2 Spezifikationen (Section 7.2.14.3):

Tabelle 69: Sense Key (0h-7h) Descriptions

Sense Key	Description
0h	NO SENSE. Indicates that there is no specific sense key information to be reported for the designated logical unit. This would be the case for a successful command or a command that received CHECK CONDITION or COMMAND TERMINATED status because one of the filemark, EOM, or ILI bits is set to one.
1h	RECOVERED ERROR. Indicates that the last command completed successfully with some recovery action performed by the target. Details may be determinable by examining the additional sense bytes and the information field. When multiple recovered errors occur during one command, the choice of which error to report (first, last, most severe, etc.) is device specific.
2h	NOT READY. Indicates that the logical unit addressed cannot be accessed. Operator intervention may be required to correct this condition.
3h	MEDIUM ERROR. Indicates that the command terminated with a non-recovered error condition that was probably caused by a flaw in the medium or an error in the recorded data. This sense key may also be returned if the target is unable to distinguish between a flaw in the medium and a specific hardware failure (sense key 4h).
4h	HARDWARE ERROR. Indicates that the target detected a non-recoverable hardware failure (for example, controller failure, device failure, parity error, etc.) while performing the command or during a self test.

5h	ILLEGAL REQUEST. Indicates that there was an illegal parameter in the command descriptor block or in the additional parameters supplied as data for some commands (FORMAT UNIT, SEARCH DATA, etc.). If the target detects an invalid parameter in the command descriptor block, then it shall terminate the command without altering the medium. If the target detects an invalid parameter in the additional parameters supplied as data, then the target may have already altered the medium. This sense key may also indicate that an invalid IDENTIFY message was received (5.6.7).
6h	UNIT ATTENTION. Indicates that the removable medium may have been changed or the target has been reset. See 6.9 for more detailed information about the unit attention condition.
7h	DATA PROTECT. Indicates that a command that reads or writes the medium was attempted on a block that is protected from this operation. The read or write operation is not performed.

Tabelle 70: Sense Key (8h-Fh) Descriptions

Sense Key	Description
8h	BLANK CHECK. Indicates that a write-once device or a sequential-access device encountered blank medium or format-defined end-of-data indication while reading or a write-once device encountered a non-blank medium while writing.
9h	Vendor Specific. This sense key is available for reporting vendor specific conditions.
Ah	COPY ABORTED. Indicates a COPY, COMPARE, or COPY AND VERIFY command was aborted due to an error condition on the source device, the destination device, or both. (See 7.2.3.2 for additional information about this sense key.)
Bh	ABORTED COMMAND. Indicates that the target aborted the command. The initiator may be able to recover by trying the command again.
Ch	EQUAL. Indicates a SEARCH DATA command has satisfied an equal comparison.
Dh	VOLUME OVERFLOW. Indicates that a buffered peripheral device has reached the end-of-partition and data may remain in the buffer that has not been written to the medium. A RECOVER BUFFERED DATA command(s) may be issued to read the unwritten data from the buffer.
Eh	MISCOMPARE. Indicates that the source data did not match the data read from the medium.
Fh	RESERVED.

A.4 Hostkodes

Die folgenden Hostkodes (Rückgabewerte der Kernelseite) sind in der Datei `scsi.h` definiert. Sie werden innerhalb des Kerneltreibers gesetzt.

Wert	Symbol	Beschreibung
0x00	DID_OK	Kein Fehler
0x01	DID_NO_CONNECT	Kein Connect innerhalb des Timeouts
0x02	DID_BUS_BUSY	BUS blieb die ganze Timeout-zeit busy
0x03	DID_TIME_OUT	TIMEOUT aus anderen Gründen
0x04	DID_BAD_TARGET	fehlerhaftes Peripheriegerät
0x05	DID_ABORT	Abbruchaufforderung von außerhalb
0x06	DID_PARITY	Paritätsfehler
0x07	DID_ERROR	Fehler in der Treiberlogik
0x08	DID_RESET	Reset von außerhalb
0x09	DID_BAD_INTR	ein unerwarteter Interrupt ist aufgetreten

A.5 Treiberkodes

In der mittleren Schicht des Treibers wird der vom unten gelieferte Status je nach Sensekey des Geräts bewertet. Daraus werden Vorschläge (wie Widerholung des Befehls, Abbruch oder Ummappen eines Blocks) gebildet und an die scsi_done-Routine weitergegeben. Dort wird je nach `host_byte()`, `status_byte()`, `msg_byte()` und Vorschlag sehr differenziert reagiert. Im Treiberbyte wird dann ein Wert gesetzt, der zeigt wie reagiert wurde. Es ist aus zwei Halbbytes (nibbles) zusammengesetzt: der Treiberzustand und der Vorschlag. Jede Hälfte wird aus den unten aufgelisteten Werten (stehen in der Datei `scsi.h`) gebildet, die dann 'zusammengeodert' werden.

Wert	Symbol	Beschreibung des Treiberzustands
0x00	DRIVER_OK	kein Fehler
0x01	DRIVER_BUSY	unbenutzt
0x02	DRIVER_SOFT	unbenutzt
0x03	DRIVER_MEDIA	unbenutzt
0x04	DRIVER_ERROR	Fehler innerhalb der Treiberlogik
0x05	DRIVER_INVALID	beendet (DID_BAD_TARGET oder DID_ABORT)
0x06	DRIVER_TIMEOUT	beendet durch Timeout
0x07	DRIVER_HARD	beendet mit fatalem Fehler
0x08	DRIVER_SENSE	Senseinformation liegt vor

Wert	Symbol	Beschreibung des Vorschlags
0x10	SUGGEST_RETRY	widerhole den SCSI-Befehl
0x20	SUGGEST_ABORT	brich den Befehl ab
0x30	SUGGEST_REMAP	den Block ummappen (nicht unterstützt)
0x40	SUGGEST_DIE	initiiere eine Kernelpanik
0x80	SUGGEST_SENSE	hole die Sensedaten vom Gerät
0xff	SUGGEST_IS_OK	nichts zu tun

B Zusätzlich Prüfcodes (ASC) und zusätzliche Prüfcodequalifizierer (ASCQ)

Falls ein ausgeführtes SCSI-Kommando den Status CHECK_CONDITION (in Deutsch etwa überprüfe den Zustand) liefert, stehen die Sensedaten danach im Sensebuffer zur Verfügung. Der zusätzliche Prüfcode (ASC) und der zusätzliche Prüfcodequalifizierer (ASCQ) liegen in diesem Puffer vor.

Ich füge aus der SCSI-2 Spezifikation noch zwei (nichtübersetzte) Tabellen für ASC/ASCQ-Paarungen ein. Die erste ist lexikalisch, die zweite numerisch sortiert.

B.1 ASC und ASCQ Werte in lexikalischer Sortierung

Die folgende Tabelle gibt den Zusammenhang zwischen ASC/ASCQ-Paaren und den Devicetypen an, für die sie definiert sind.

```
+=============================================================================+
|           D - DIRECT ACCESS DEVICE                                          |
|           .T - SEQUENTIAL ACCESS DEVICE                                     |
|           . L - PRINTER DEVICE                                              |
|           .  P - PROCESSOR DEVICE                                           |
|           .  .W - WRITE ONCE READ MULTIPLE DEVICE                           |
|           .  . R - READ ONLY (CD-ROM) DEVICE                                |
|           .  .  S - SCANNER DEVICE                                          |
|           .  .  .O - OPTICAL MEMORY DEVICE                                  |
|           .  .  . M - MEDIA CHANGER DEVICE                                  |
|           .  .  .  C - COMMUNICATION DEVICE                                 |
|           .  .  .  .                                                       |
| ASC ASCQ  DTLPWRSOMC  DESCRIPTION                                           |
| --- ----  ----------  ------------------------------------------------- |
| 13h  00h  D   W  O    ADDRESS MARK NOT FOUND FOR DATA FIELD                 |
| 12h  00h  D   W  O    ADDRESS MARK NOT FOUND FOR ID FIELD                   |
| 00h  11h       R      AUDIO PLAY OPERATION IN PROGRESS                      |
| 00h  12h       R      AUDIO PLAY OPERATION PAUSED                           |
| 00h  14h       R      AUDIO PLAY OPERATION STOPPED DUE TO ERROR             |
| 00h  13h       R      AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED           |
| 00h  04h   T    S     BEGINNING-OF-PARTITION/MEDIUM DETECTED                |
| 14h  04h   T          BLOCK SEQUENCE ERROR                                  |
| 30h  02h  DT  WR O    CANNOT READ MEDIUM - INCOMPATIBLE FORMAT              |
| 30h  01h  DT  WR O    CANNOT READ MEDIUM - UNKNOWN FORMAT                   |
| 52h  00h   T          CARTRIDGE FAULT                                       |
| 3Fh  02h  DTLPWRSOMC  CHANGED OPERATING DEFINITION                          |
| 11h  06h      WR O    CIRC UNRECOVERED ERROR                                |
| 30h  03h  DT          CLEANING CARTRIDGE INSTALLED                          |
| 4Ah  00h  DTLPWRSOMC  COMMAND PHASE ERROR                                   |
| 2Ch  00h  DTLPWRSOMC  COMMAND SEQUENCE ERROR                                |
| 2Fh  00h  DTLPWRSOMC  COMMANDS CLEARED BY ANOTHER INITIATOR                 |
| 2Bh  00h  DTLPWRSO C  COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT      |
| 41h  00h  D           DATA PATH FAILURE (SHOULD USE 40 NN)                  |
| 4Bh  00h  DTLPWRSOMC  DATA PHASE ERROR                                      |
| 11h  07h      W  O    DATA RESYCHRONIZATION ERROR                           |
| 16h  00h  D   W  O    DATA SYNCHRONIZATION MARK ERROR                       |
| 19h  00h  D      O    DEFECT LIST ERROR                                     |
| 19h  03h  D      O    DEFECT LIST ERROR IN GROWN LIST                       |
```

19h	02h	D	O	DEFECT LIST ERROR IN PRIMARY LIST
19h	01h	D	O	DEFECT LIST NOT AVAILABLE
1Ch	00h	D	O	DEFECT LIST NOT FOUND
32h	01h	D W	O	DEFECT LIST UPDATE FAILURE
40h	NNh	DTLPWRSOMC		DIAGNOSTIC FAILURE ON COMPONENT NN (80H-FFH)
63h	00h	R		END OF USER AREA ENCOUNTERED ON THIS TRACK
00h	05h	T	S	END-OF-DATA DETECTED
14h	03h	T		END-OF-DATA NOT FOUND
00h	02h	T	S	END-OF-PARTITION/MEDIUM DETECTED
51h	00h	T	O	ERASE FAILURE
0Ah	00h	DTLPWRSOMC		ERROR LOG OVERFLOW
11h	02h	DT W	SO	ERROR TOO LONG TO CORRECT
03h	02h	T		EXCESSIVE WRITE ERRORS
3Bh	07h	L		FAILED TO SENSE BOTTOM-OF-FORM
3Bh	06h	L		FAILED TO SENSE TOP-OF-FORM
00h	01h	T		FILEMARK DETECTED
14h	02h	T		FILEMARK OR SETMARK NOT FOUND
09h	02h	WR	O	FOCUS SERVO FAILURE
31h	01h	D L	O	FORMAT COMMAND FAILED
58h	00h		O	GENERATION DOES NOT EXIST

Tabelle 71: (fortgesetzt)

ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
1Ch	02h	D O	GROWN DEFECT LIST NOT FOUND
00h	06h	DTLPWRSOMC	I/O PROCESS TERMINATED
10h	00h	D W	O ID CRC OR ECC ERROR
22h	00h	D	ILLEGAL FUNCTION (SHOULD USE 20 00, 24 00, OR 26 00)
64h	00h	R	ILLEGAL MODE FOR THIS TRACK
28h	01h	M	IMPORT OR EXPORT ELEMENT ACCESSED
30h	00h	DT WR	OM INCOMPATIBLE MEDIUM INSTALLED
11h	08h	T	INCOMPLETE BLOCK READ
48h	00h	DTLPWRSOMC	INITIATOR DETECTED ERROR MESSAGE RECEIVED
3Fh	03h	DTLPWRSOMC	INQUIRY DATA HAS CHANGED
44h	00h	DTLPWRSOMC	INTERNAL TARGET FAILURE
3Dh	00h	DTLPWRSOMC	INVALID BITS IN IDENTIFY MESSAGE
2Ch	02h	S	INVALID COMBINATION OF WINDOWS SPECIFIED
20h	00h	DTLPWRSOMC	INVALID COMMAND OPERATION CODE
21h	01h	M	INVALID ELEMENT ADDRESS
24h	00h	DTLPWRSOMC	INVALID FIELD IN CDB
26h	00h	DTLPWRSOMC	INVALID FIELD IN PARAMETER LIST
49h	00h	DTLPWRSOMC	INVALID MESSAGE ERROR
11h	05h	WR	O L-EC UNCORRECTABLE ERROR
60h	00h	S	LAMP FAILURE
5Bh	02h	DTLPWRSOM	LOG COUNTER AT MAXIMUM
5Bh	00h	DTLPWRSOM	LOG EXCEPTION
5Bh	03h	DTLPWRSOM	LOG LIST CODES EXHAUSTED
2Ah	02h	DTL WRSOMC	LOG PARAMETERS CHANGED
21h	00h	DT WR	OM LOGICAL BLOCK ADDRESS OUT OF RANGE
08h	00h	DTL WRSOMC	LOGICAL UNIT COMMUNICATION FAILURE
08h	02h	DTL WRSOMC	LOGICAL UNIT COMMUNICATION PARITY ERROR
08h	01h	DTL WRSOMC	LOGICAL UNIT COMMUNICATION TIME-OUT
4Ch	00h	DTLPWRSOMC	LOGICAL UNIT FAILED SELF-CONFIGURATION

3Eh	00h	DTLPWRSOMC	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET
04h	01h	DTLPWRSOMC	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
04h	00h	DTLPWRSOMC	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
04h	04h	DTL O	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
04h	02h	DTLPWRSOMC	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
04h	03h	DTLPWRSOMC	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
25h	00h	DTLPWRSOMC	LOGICAL UNIT NOT SUPPORTED
15h	01h	DTL WRSOM	MECHANICAL POSITIONING ERROR
53h	00h	DTL WRSOM	MEDIA LOAD OR EJECT FAILED
3Bh	0Dh	M	MEDIUM DESTINATION ELEMENT FULL
31h	00h	DT W O	MEDIUM FORMAT CORRUPTED
3Ah	00h	DTL WRSOM	MEDIUM NOT PRESENT
53h	02h	DT WR OM	MEDIUM REMOVAL PREVENTED
3Bh	0Eh	M	MEDIUM SOURCE ELEMENT EMPTY
43h	00h	DTLPWRSOMC	MESSAGE ERROR
3Fh	01h	DTLPWRSOMC	MICROCODE HAS BEEN CHANGED
1Dh	00h	D W O	MISCOMPARE DURING VERIFY OPERATION
11h	0Ah	DT O	MISCORRECTED ERROR
2Ah	01h	DTL WRSOMC	MODE PARAMETERS CHANGED
07h	00h	DTL WRSOM	MULTIPLE PERIPHERAL DEVICES SELECTED
11h	03h	DT W SO	MULTIPLE READ ERRORS
00h	00h	DTLPWRSOMC	NO ADDITIONAL SENSE INFORMATION
00h	15h	R	NO CURRENT AUDIO STATUS TO RETURN
32h	00h	D W O	NO DEFECT SPARE LOCATION AVAILABLE
11h	09h	T	NO GAP FOUND
01h	00h	D W O	NO INDEX/SECTOR SIGNAL
06h	00h	D WR OM	NO REFERENCE POSITION FOUND

Tabelle 71: (fortgesetzt)

ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
02h	00h	D WR OM	NO SEEK COMPLETE
03h	01h	T	NO WRITE CURRENT
28h	00h	DTLPWRSOMC	NOT READY TO READY TRANSITION, MEDIUM MAY HAVE CHANGED
5Ah	01h	DT WR OM	OPERATOR MEDIUM REMOVAL REQUEST
5Ah	00h	DTLPWRSOM	OPERATOR REQUEST OR STATE CHANGE INPUT (UNSPECIFIED)
5Ah	03h	DT W O	OPERATOR SELECTED WRITE PERMIT
5Ah	02h	DT W O	OPERATOR SELECTED WRITE PROTECT
61h	02h	S	OUT OF FOCUS
4Eh	00h	DTLPWRSOMC	OVERLAPPED COMMANDS ATTEMPTED
2Dh	00h	T	OVERWRITE ERROR ON UPDATE IN PLACE
3Bh	05h	L	PAPER JAM
1Ah	00h	DTLPWRSOMC	PARAMETER LIST LENGTH ERROR
26h	01h	DTLPWRSOMC	PARAMETER NOT SUPPORTED
26h	02h	DTLPWRSOMC	PARAMETER VALUE INVALID
2Ah	00h	DTL WRSOMC	PARAMETERS CHANGED
03h	00h	DTL W SO	PERIPHERAL DEVICE WRITE FAULT
50h	02h	T	POSITION ERROR RELATED TO TIMING
3Bh	0Ch	S	POSITION PAST BEGINNING OF MEDIUM
3Bh	0Bh	S	POSITION PAST END OF MEDIUM
15h	02h	DT WR O	POSITIONING ERROR DETECTED BY READ OF MEDIUM
29h	00h	DTLPWRSOMC	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
42h	00h	D	POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)

1Ch	01h	D	O	PRIMARY DEFECT LIST NOT FOUND
40h	00h	D		RAM FAILURE (SHOULD USE 40 NN)
15h	00h	DTL	WRSOM	RANDOM POSITIONING ERROR
3Bh	0Ah		S	READ PAST BEGINNING OF MEDIUM
3Bh	09h		S	READ PAST END OF MEDIUM
11h	01h	DT	W SO	READ RETRIES EXHAUSTED
14h	01h	DT	WR O	RECORD NOT FOUND
14h	00h	DTL	WRSO	RECORDED ENTITY NOT FOUND
18h	02h	D	WR O	RECOVERED DATA - DATA AUTO-REALLOCATED
18h	05h	D	WR O	RECOVERED DATA - RECOMMEND REASSIGNMENT
18h	06h	D	WR O	RECOVERED DATA - RECOMMEND REWRITE
17h	05h	D	WR O	RECOVERED DATA USING PREVIOUS SECTOR ID
18h	03h		R	RECOVERED DATA WITH CIRC
18h	01h	D	WR O	RECOVERED DATA WITH ERROR CORRECTION & RETRIES APPLIED
18h	00h	DT	WR O	RECOVERED DATA WITH ERROR CORRECTION APPLIED
18h	04h		R	RECOVERED DATA WITH L-EC
17h	03h	DT	WR O	RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	00h	DT	WRSO	RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17h	02h	DT	WR O	RECOVERED DATA WITH POSITIVE HEAD OFFSET
17h	01h	DT	WRSO	RECOVERED DATA WITH RETRIES
17h	04h		WR O	RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17h	06h	D	W O	RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17h	07h	D	W O	RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17h	08h	D	W O	RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
1Eh	00h	D	W O	RECOVERED ID WITH ECC CORRECTION
3Bh	08h	T		REPOSITION ERROR
36h	00h	L		RIBBON, INK, OR TONER FAILURE
37h	00h	DTL	WRSOMC	ROUNDED PARAMETER
5Ch	00h	D	O	RPL STATUS CHANGE
39h	00h	DTL	WRSOMC	SAVING PARAMETERS NOT SUPPORTED
62h	00h		S	SCAN HEAD POSITIONING ERROR
47h	00h	DTLPWRSOMC		SCSI PARITY ERROR
54h	00h		P	SCSI TO HOST SYSTEM INTERFACE FAILURE
45h	00h	DTLPWRSOMC		SELECT OR RESELECT FAILURE

Tabelle 71: (Schlußteil)

ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
3Bh	00h	TL	SEQUENTIAL POSITIONING ERROR
00h	03h	T	SETMARK DETECTED
3Bh	04h	L	SLEW FAILURE
09h	03h	WR O	SPINDLE SERVO FAILURE
5Ch	02h	D O	SPINDLES NOT SYNCHRONIZED
5Ch	01h	D O	SPINDLES SYNCHRONIZED
1Bh	00h	DTLPWRSOMC	SYNCHRONOUS DATA TRANSFER ERROR
55h	00h	P	SYSTEM RESOURCE FAILURE
33h	00h	T	TAPE LENGTH ERROR
3Bh	03h	L	TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY
3Bh	01h	T	TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM
3Bh	02h	T	TAPE POSITION ERROR AT END-OF-MEDIUM
3Fh	00h	DTLPWRSOMC	TARGET OPERATING CONDITIONS HAVE CHANGED
5Bh	01h	DTLPWRSOM	THRESHOLD CONDITION MET

26h	03h	DTLPWRSOMC	THRESHOLD PARAMETERS NOT SUPPORTED
2Ch	01h	S	TOO MANY WINDOWS SPECIFIED
09h	00h	DT WR O	TRACK FOLLOWING ERROR
09h	01h	WR O	TRACKING SERVO FAILURE
61h	01h	S	UNABLE TO ACQUIRE VIDEO
57h	00h	R	UNABLE TO RECOVER TABLE-OF-CONTENTS
53h	01h	T	UNLOAD TAPE FAILURE
11h	00h	DT WRSO	UNRECOVERED READ ERROR
11h	04h	D W O	UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	0Bh	D W O	UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D W O	UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
46h	00h	DTLPWRSOMC	UNSUCCESSFUL SOFT RESET
59h	00h	O	UPDATED BLOCK READ
61h	00h	S	VIDEO ACQUISITION ERROR
50h	00h	T	WRITE APPEND ERROR
50h	01h	T	WRITE APPEND POSITION ERROR
0Ch	00h	T S	WRITE ERROR
0Ch	02h	D W O	WRITE ERROR - AUTO REALLOCATION FAILED
0Ch	01h	D W O	WRITE ERROR RECOVERED WITH AUTO REALLOCATION
27h	00h	DT W O	WRITE PROTECTED
80h	XXh	\	
THROUGH		>	VENDOR SPECIFIC.
FFh	XX	/	
XXh	80h	\	
THROUGH		>	VENDOR SPECIFIC QUALIFICATION OF STANDARD ASC.
XXh	FFh	/	
			ALL CODES NOT SHOWN ARE RESERVED.

B.2 ASC und ASCQ Werte in numerischer Sortierung

Tabelle 364: ASC und ASCQ Zuordnungen

		D - DIRECT ACCESS DEVICE	
		.T - SEQUENTIAL ACCESS DEVICE	
		. L - PRINTER DEVICE	
		. P - PROCESSOR DEVICE	
		. .W - WRITE ONCE READ MULTIPLE DEVICE	
		. . R - READ ONLY (CD-ROM) DEVICE	
		. . S - SCANNER DEVICE	
		. . .O - OPTICAL MEMORY DEVICE	
		. . . M - MEDIA CHANGER DEVICE	
		. . . C - COMMUNICATION DEVICE	
		. . . .	
ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
00	00	DTLPWRSOMC	NO ADDITIONAL SENSE INFORMATION
00	01	T	FILEMARK DETECTED
00	02	T S	END-OF-PARTITION/MEDIUM DETECTED
00	03	T	SETMARK DETECTED
00	04	T S	BEGINNING-OF-PARTITION/MEDIUM DETECTED

00	05	T S	END-OF-DATA DETECTED
00	06	DTLPWRSOMC	I/O PROCESS TERMINATED
00	11	R	AUDIO PLAY OPERATION IN PROGRESS
00	12	R	AUDIO PLAY OPERATION PAUSED
00	13	R	AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED
00	14	R	AUDIO PLAY OPERATION STOPPED DUE TO ERROR
00	15	R	NO CURRENT AUDIO STATUS TO RETURN
01	00	DW O	NO INDEX/SECTOR SIGNAL
02	00	DWR OM	NO SEEK COMPLETE
03	00	DTL W SO	PERIPHERAL DEVICE WRITE FAULT
03	01	T	NO WRITE CURRENT
03	02	T	EXCESSIVE WRITE ERRORS
04	00	DTLPWRSOMC	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
04	01	DTLPWRSOMC	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
04	02	DTLPWRSOMC	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
04	03	DTLPWRSOMC	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
04	04	DTL O	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
05	00	DTL WRSOMC	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
06	00	DWR OM NO	REFERENCE POSITION FOUND
07	00	DTL WRSOM	MULTIPLE PERIPHERAL DEVICES SELECTED
08	00	DTL WRSOMC	LOGICAL UNIT COMMUNICATION FAILURE
08	01	DTL WRSOMC	LOGICAL UNIT COMMUNICATION TIME-OUT
08	02	DTL WRSOMC	LOGICAL UNIT COMMUNICATION PARITY ERROR
09	00	DT WR O	TRACK FOLLOWING ERROR
09	01	WR O	TRA CKING SERVO FAILURE
09	02	WR O	FOC US SERVO FAILURE
09	03	WR O	SPI NDLE SERVO FAILURE

Tabelle 364: (fortgesetzt)

		D - DIRECT ACCESS DEVICE	
		.T - SEQUENTIAL ACCESS DEVICE	
		. L - PRINTER DEVICE	
		. P - PROCESSOR DEVICE	
		. .W - WRITE ONCE READ MULTIPLE DEVICE	
		. . R - READ ONLY (CD-ROM) DEVICE	
		. . S - SCANNER DEVICE	
		. . .O - OPTICAL MEMORY DEVICE	
		. . . M - MEDIA CHANGER DEVICE	
		. . . C - COMMUNICATION DEVICE	
		. . . .	
ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
---	----		---
0A	00	DTLPWRSOMC	ERROR LOG OVERFLOW
0B	00		
0C	00	T S	WRITE ERROR
0C	01	D W O	WRITE ERROR RECOVERED WITH AUTO REALLOCATION
0C	02	D W O	WRITE ERROR - AUTO REALLOCATION FAILED
0D	00		
0E	00		
0F	00		
10	00	D W O	ID CRC OR ECC ERROR
11	00	DT WRSO	UNRECOVERED READ ERROR
11	01	DT W SO	READ RETRIES EXHAUSTED

11	02	DT	W	SO	ERROR TOO LONG TO CORRECT
11	03	DT	W	SO	MULTIPLE READ ERRORS
11	04	D	W	O	UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11	05		WR	O	L-EC UNCORRECTABLE ERROR
11	06		WR	O	CIRC UNRECOVERED ERROR
11	07		W	O	DATA RESYCHRONIZATION ERROR
11	08	T			INCOMPLETE BLOCK READ
11	09	T			NO GAP FOUND
11	0A	DT		O	MISCORRECTED ERROR
11	0B	D	W	O	UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11	0C	D	W	O	UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
12	00	D	W	O	ADDRESS MARK NOT FOUND FOR ID FIELD
13	00	D	W	O	ADDRESS MARK NOT FOUND FOR DATA FIELD
14	00	DTL	WRSO		RECORDED ENTITY NOT FOUND
14	01	DT	WR	O	RECORD NOT FOUND
14	02	T			FILEMARK OR SETMARK NOT FOUND
14	03	T			END-OF-DATA NOT FOUND
14	04	T			BLOCK SEQUENCE ERROR
15	00	DTL	WRSOM		RANDOM POSITIONING ERROR
15	01	DTL	WRSOM		MECHANICAL POSITIONING ERROR
15	02	DT	WR	O	POSITIONING ERROR DETECTED BY READ OF MEDIUM
16	00	DW		O	DATA SYNCHRONIZATION MARK ERROR
17	00	DT	WRSO		RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17	01	DT	WRSO		RECOVERED DATA WITH RETRIES
17	02	DT	WR	O	RECOVERED DATA WITH POSITIVE HEAD OFFSET
17	03	DT	WR	O	RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17	04		WR	O	RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17	05	D	WR	O	RECOVERED DATA USING PREVIOUS SECTOR ID
17	06	D	W	O	RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17	07	D	W	O	RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17	08	D	W	O	RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
18	00	DT	WR	O	RECOVERED DATA WITH ERROR CORRECTION APPLIED
18	01	D	WR	O	RECOVERED DATA WITH ERROR CORRECTION & RETRIES APPLIED
18	02	D	WR	O	RECOVERED DATA - DATA AUTO-REALLOCATED
18	03		R		RECOVERED DATA WITH CIRC
18	04		R		RECOVERED DATA WITH LEC
18	05	D	WR	O	RECOVERED DATA - RECOMMEND REASSIGNMENT
18	06	D	WR	O	RECOVERED DATA - RECOMMEND REWRITE

Tabelle 364: (fortgesetzt)

D - DIRECT ACCESS DEVICE					
.T - SEQUENTIAL ACCESS DEVICE					
. L - PRINTER DEVICE					
. P - PROCESSOR DEVICE					
. .W - WRITE ONCE READ MULTIPLE DEVICE					
. . R - READ ONLY (CD-ROM) DEVICE					
. . S - SCANNER DEVICE					
. . .O - OPTICAL MEMORY DEVICE					
. . . M - MEDIA CHANGER DEVICE					
. . . C - COMMUNICATION DEVICE					
. . . .					
ASC	ASCQ	DTLPWRSOMC			DESCRIPTION

| --- | --- | --- | --- |
19	00	D O	DEFECT LIST ERROR
19	01	D O	DEFECT LIST NOT AVAILABLE
19	02	D O	DEFECT LIST ERROR IN PRIMARY LIST
19	03	D O	DEFECT LIST ERROR IN GROWN LIST
1A	00	DTLPWRSOMC	PARAMETER LIST LENGTH ERROR
1B	00	DTLPWRSOMC	SYNCHRONOUS DATA TRANSFER ERROR
1C	00	D O	DEFECT LIST NOT FOUND
1C	01	D O	PRIMARY DEFECT LIST NOT FOUND
1C	02	D O	GROWN DEFECT LIST NOT FOUND
1D	00	D W O	MISCOMPARE DURING VERIFY OPERATION
1E	00	D W O	RECOVERED ID WITH ECC
1F	00		
20	00	DTLPWRSOMC	INVALID COMMAND OPERATION CODE
21	00	DT WR OM	LOGICAL BLOCK ADDRESS OUT OF RANGE
21	01	M	INVALID ELEMENT ADDRESS
22	00	D	ILLEGAL FUNCTION (SHOULD USE 20 00, 24 00, OR 26 00)
23	00		
24	00	DTLPWRSOMC	INVALID FIELD IN CDB
25	00	DTLPWRSOMC	LOGICAL UNIT NOT SUPPORTED
26	00	DTLPWRSOMC	INVALID FIELD IN PARAMETER LIST
26	01	DTLPWRSOMC	PARAMETER NOT SUPPORTED
26	02	DTLPWRSOMC	PARAMETER VALUE INVALID
26	03	DTLPWRSOMC	THRESHOLD PARAMETERS NOT SUPPORTED
27	00	DT W O	WRITE PROTECTED
28	00	DTLPWRSOMC	NOT READY TO READY TRANSITION(MEDIUM MAY HAVE CHANGED)
28	01	M	IMPORT OR EXPORT ELEMENT ACCESSED
29	00	DTLPWRSOMC	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
2A	00	DTL WRSOMC	PARAMETERS CHANGED
2A	01	DTL WRSOMC	MODE PARAMETERS CHANGED
2A	02	DTL WRSOMC	LOG PARAMETERS CHANGED
2B	00	DTLPWRSO C	COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT
2C	00	DTLPWRSOMC	COMMAND SEQUENCE ERROR
2C	01	S	TOO MANY WINDOWS SPECIFIED
2C	02	S	INVALID COMBINATION OF WINDOWS SPECIFIED
2D	00	T	OVERWRITE ERROR ON UPDATE IN PLACE
2E	00		
2F	00	DTLPWRSOMC	COMMANDS CLEARED BY ANOTHER INITIATOR
30	00	DT WR OM	INCOMPATIBLE MEDIUM INSTALLED
30	01	DT WR O	CANNOT READ MEDIUM - UNKNOWN FORMAT
30	02	DT WR O	CANNOT READ MEDIUM - INCOMPATIBLE FORMAT
30	03	DT	CLEANING CARTRIDGE INSTALLED
31	00	DT W O	MEDIUM FORMAT CORRUPTED
31	01	D L O	FORMAT COMMAND FAILED
32	00	D W O	NO DEFECT SPARE LOCATION AVAILABLE
32	01	D W O	DEFECT LIST UPDATE FAILURE
33	00	T	TAPE LENGTH ERROR
34	00		
35	00		
36	00	L	RIBBON, INK, OR TONER FAILURE

Tabelle 364: (fortgesetzt)

D - DIRECT ACCESS DEVICE

```
|           .T - SEQUENTIAL ACCESS DEVICE                                    |
|           . L - PRINTER DEVICE                                             |
|           .  P - PROCESSOR DEVICE                                          |
|           .  .W - WRITE ONCE READ MULTIPLE DEVICE                          |
|           .  . R - READ ONLY (CD-ROM) DEVICE                               |
|           .  .  S - SCANNER DEVICE                                         |
|           .  .  .O - OPTICAL MEMORY DEVICE                                 |
|           .  .  . M - MEDIA CHANGER DEVICE                                 |
|           .  .  .  C - COMMUNICATION DEVICE                                |
|           .  .  .  .                                                       |
| ASC ASCQ  DTLPWRSOMC  DESCRIPTION                                          |
| --- ----              ------------------------------------------------     |
|  37  00   DTL WRSOMC  ROUNDED PARAMETER                                    |
|  38  00                                                                    |
|  39  00   DTL WRSOMC  SAVING PARAMETERS NOT SUPPORTED                      |
|  3A  00   DTL WRSOM   MEDIUM NOT PRESENT                                   |
|  3B  00    TL         SEQUENTIAL POSITIONING ERROR                         |
|  3B  01    T          TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM           |
|  3B  02    T          TAPE POSITION ERROR AT END-OF-MEDIUM                 |
|  3B  03     L         TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY     |
|  3B  04     L         SLEW FAILURE                                         |
|  3B  05     L         PAPER JAM                                            |
|  3B  06     L         FAILED TO SENSE TOP-OF-FORM                          |
|  3B  07     L         FAILED TO SENSE BOTTOM-OF-FORM                       |
|  3B  08    T          REPOSITION ERROR                                     |
|  3B  09         S     READ PAST END OF MEDIUM                              |
|  3B  0A         S     READ PAST BEGINNING OF MEDIUM                        |
|  3B  0B         S     POSITION PAST END OF MEDIUM                          |
|  3B  0C         S     POSITION PAST BEGINNING OF MEDIUM                    |
|  3B  0D           M   MEDIUM DESTINATION ELEMENT FULL                      |
|  3B  0E           M   MEDIUM SOURCE ELEMENT EMPTY                          |
|  3C  00                                                                    |
|  3D  00   DTLPWRSOMC  INVALID BITS IN IDENTIFY MESSAGE                     |
|  3E  00   DTLPWRSOMC  LOGICAL UNIT HAS NOT SELF-CONFIGURED YET             |
|  3F  00   DTLPWRSOMC  TARGET OPERATING CONDITIONS HAVE CHANGED             |
|  3F  01   DTLPWRSOMC  MICROCODE HAS BEEN CHANGED                           |
|  3F  02   DTLPWRSOMC  CHANGED OPERATING DEFINITION                         |
|  3F  03   DTLPWRSOMC  INQUIRY DATA HAS CHANGED                             |
|  40  00   D           RAM FAILURE (SHOULD USE 40 NN)                       |
|  40  NN   DTLPWRSOMC  DIAGNOSTIC FAILURE ON COMPONENT NN (80H-FFH)         |
|  41  00   D           DATA PATH FAILURE (SHOULD USE 40 NN)                 |
|  42  00   D           POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)     |
|  43  00   DTLPWRSOMC  MESSAGE ERROR                                        |
|  44  00   DTLPWRSOMC  INTERNAL TARGET FAILURE                              |
|  45  00   DTLPWRSOMC  SELECT OR RESELECT FAILURE                           |
|  46  00   DTLPWRSOMC  UNSUCCESSFUL SOFT RESET                              |
|  47  00   DTLPWRSOMC  SCSI PARITY ERROR                                    |
|  48  00   DTLPWRSOMC  INITIATOR DETECTED ERROR MESSAGE RECEIVED            |
|  49  00   DTLPWRSOMC  INVALID MESSAGE ERROR                                |
|  4A  00   DTLPWRSOMC  COMMAND PHASE ERROR                                  |
|  4B  00   DTLPWRSOMC  DATA PHASE ERROR                                     |
|  4C  00   DTLPWRSOMC  LOGICAL UNIT FAILED SELF-CONFIGURATION               |
|  4D  00                                                                    |
|  4E  00   DTLPWRSOMC  OVERLAPPED COMMANDS ATTEMPTED                        |
|  4F  00                                                                    |
```

50	00	T		WRITE APPEND ERROR
50	01	T		WRITE APPEND POSITION ERROR
50	02	T		POSITION ERROR RELATED TO TIMING
51	00	T	O	ERASE FAILURE
52	00	T		CARTRIDGE FAULT

Tabelle 364: (fortgesetzt)

D - DIRECT ACCESS DEVICE
.T - SEQUENTIAL ACCESS DEVICE
. L - PRINTER DEVICE
. P - PROCESSOR DEVICE
. .W - WRITE ONCE READ MULTIPLE DEVICE
. . R - READ ONLY (CD-ROM) DEVICE
. . S - SCANNER DEVICE
. . .O - OPTICAL MEMORY DEVICE
. . . M - MEDIA CHANGER DEVICE
. . . C - COMMUNICATION DEVICE
. . . .

ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
53	00	DTL WRSOM	MEDIA LOAD OR EJECT FAILED
53	01	T	UNLOAD TAPE FAILURE
53	02	DT WR OM	MEDIUM REMOVAL PREVENTED
54	00	P	SCSI TO HOST SYSTEM INTERFACE FAILURE
55	00	P	SYSTEM RESOURCE FAILURE
56	00		
57	00	R	UNABLE TO RECOVER TABLE-OF-CONTENTS
58	00	O	GENERATION DOES NOT EXIST
59	00	O	UPDATED BLOCK READ
5A	00	DTLPWRSOM	OPERATOR REQUEST OR STATE CHANGE INPUT (UNSPECIFIED)
5A	01	DT WR OM	OPERATOR MEDIUM REMOVAL REQUEST
5A	02	DT W O	OPERATOR SELECTED WRITE PROTECT
5A	03	DT W O	OPERATOR SELECTED WRITE PERMIT
5B	00	DTLPWRSOM	LOG EXCEPTION
5B	01	DTLPWRSOM	THRESHOLD CONDITION MET
5B	02	DTLPWRSOM	LOG COUNTER AT MAXIMUM
5B	03	DTLPWRSOM	LOG LIST CODES EXHAUSTED
5C	00	D O	RPL STATUS CHANGE
5C	01	D O	SPINDLES SYNCHRONIZED
5C	02	D O	SPINDLES NOT SYNCHRONIZED
5D	00		
5E	00		
5F	00		
60	00	S	LAMP FAILURE
61	00	S	VIDEO ACQUISITION ERROR
61	01	S	UNABLE TO ACQUIRE VIDEO
61	02	S	OUT OF FOCUS
62	00	S	SCAN HEAD POSITIONING ERROR
63	00	R	END OF USER AREA ENCOUNTERED ON THIS TRACK
64	00	R	ILLEGAL MODE FOR THIS TRACK
65	00		
66	00		
67	00		

68	00	
69	00	
6A	00	
6B	00	
6C	00	
6D	00	
6E	00	
6F	00	

Tabelle 364: (Schlußteil)

D - DIRECT ACCESS DEVICE
.T - SEQUENTIAL ACCESS DEVICE
. L - PRINTER DEVICE
. P - PROCESSOR DEVICE
. .W - WRITE ONCE READ MULTIPLE DEVICE
. . R - READ ONLY (CD-ROM) DEVICE
. . S - SCANNER DEVICE
. . .O - OPTICAL MEMORY DEVICE
. . . M - MEDIA CHANGER DEVICE
. . . C - COMMUNICATION DEVICE
. . . .

ASC	ASCQ	DTLPWRSOMC	DESCRIPTION
70	00		
71	00		
72	00		
73	00		
74	00		
75	00		
76	00		
77	00		
78	00		
79	00		
7A	00		
7B	00		
7C	00		
7D	00		
7E	00		
7F	00		
80	xxh \		
THROUGH >			VENDOR SPECIFIC.
FF	xxh /		
xxh	80 \		
THROUGH >			VENDOR SPECIFIC QUALIFICATION OF STANDARD ASC.
xxh	FF /		
			ALL CODES NOT SHOWN OR BLANK ARE RESERVED.

C Eine SCSI Befehlscode-Kurzübersicht

Tabelle 365 ist eine numerisch sortierte Liste der SCSI-Befehlscodes.

Tabelle 365: SCSI-2 Operation Codes

D - DIRECT ACCESS DEVICE	Device Column Key
.T - SEQUENTIAL ACCESS DEVICE	M = Mandatory
. L - PRINTER DEVICE	O = Optional
. P - PROCESSOR DEVICE	V = Vendor Specific
. .W - WRITE ONCE READ MULTIPLE DEVICE	R = Reserved
. . R - READ ONLY (CD-ROM) DEVICE	
. . S - SCANNER DEVICE	
. . .O - OPTICAL MEMORY DEVICE	
. . . M - MEDIA CHANGER DEVICE	
. . . C - COMMUNICATION DEVICE	
. . . .	

OP	DTLPWRSOMC	Description
00	MMMMMMMMMM	TEST UNIT READY
01	M	REWIND
01	O V OO OO	REZERO UNIT
02	VVVVVV V	
03	MMMMMMMMMM	REQUEST SENSE
04	O	FORMAT
04	M O	FORMAT UNIT
05	VMVVVV V	READ BLOCK LIMITS
06	VVVVVV V	
07	O	INITIALIZE ELEMENT STATUS
07	OVV O OV	REASSIGN BLOCKS
08	M	GET MESSAGE(06)
08	OMV OO OV	READ(06)
08	O	RECEIVE
09	VVVVVV V	
0A	M	PRINT
0A	M	SEND MESSAGE(06)
0A	M	SEND(06)
0A	OM O OV	WRITE(06)
0B	O OO OV	SEEK(06)
0B	O	SLEW AND PRINT
0C	VVVVVV V	
0D	VVVVVV V	
0E	VVVVVV V	
0F	VOVVVV V	READ REVERSE
10	O O	SYNCHRONIZE BUFFER
10	VM VVV	WRITE FILEMARKS
11	VMVVVV	SPACE
12	MMMMMMMMMM	INQUIRY
13	VOVVVV	VERIFY(06)
14	VOOVVV	RECOVER BUFFERED DATA
15	OMO OOOOOO	MODE SELECT(06)
16	M MM MO	RESERVE
16	MM M	RESERVE UNIT
17	M MM MO	RELEASE

17	MM	M	RELEASE UNIT
18	OOOOOOOO		COPY
19	VMVVVV		ERASE
1A	OMO	OOOOOO	MODE SENSE(06)
1B	O		LOAD UNLOAD
1B		O	SCAN
1B	O		STOP PRINT
1B	O	OO O	STOP START UNIT

Tabelle 365: (fortgesetzt)

	D - DIRECT ACCESS DEVICE		Device Column Key
	.T - SEQUENTIAL ACCESS DEVICE		M = Mandatory
	. L - PRINTER DEVICE		O = Optional
	. P - PROCESSOR DEVICE		V = Vendor Specific
	. .W - WRITE ONCE READ MULTIPLE DEVICE		R = Reserved
	. . R - READ ONLY (CD-ROM) DEVICE		
	. . S - SCANNER DEVICE		
	. . .O - OPTICAL MEMORY DEVICE		
	. . . M - MEDIA CHANGER DEVICE		
	. . . C - COMMUNICATION DEVICE		
	. . . .		
OP	DTLPWRSOMC		Description
1C	OOOOOOOOOO		RECEIVE DIAGNOSTIC RESULTS
1D	MMMMMMMMMM		SEND DIAGNOSTIC
1E	OO	OO OO	PREVENT ALLOW MEDIUM REMOVAL
1F			
20	V	VV V	
21	V	VV V	
22	V	VV V	
23	V	VV V	
24	V	VVM	SET WINDOW
25		O	GET WINDOW
25	M	M M	READ CAPACITY
25		M	READ CD-ROM CAPACITY
26	V	VV	
27	V	VV	
28		O	GET MESSAGE(10)
28	M	MMMM	READ(10)
29	V	VV O	READ GENERATION
2A		O	SEND MESSAGE(10)
2A		O	SEND(10)
2A	M	M M	WRITE(10)
2B	O		LOCATE
2B		O	POSITION TO ELEMENT
2B	O	OO O	SEEK(10)
2C	V	O	ERASE(10)
2D	V	O O	READ UPDATED BLOCK
2E	O	O O	WRITE AND VERIFY(10)
2F	O	OO O	VERIFY(10)
30	O	OO O	SEARCH DATA HIGH(10)
31		O	OBJECT POSITION
31	O	OO O	SEARCH DATA EQUAL(10)

	32	O	OO O	SEARCH DATA LOW(10)
	33	O	OO O	SET LIMITS(10)
	34		O	GET DATA BUFFER STATUS
	34	O	OO O	PRE-FETCH
	34	O		READ POSITION
	35	O	OO O	SYNCHRONIZE CACHE
	36	O	OO O	LOCK UNLOCK CACHE
	37	O	O	READ DEFECT DATA(10)
	38		O O	MEDIUM SCAN
	39	OOOOOOOO		COMPARE
	3A	OOOOOOOO		COPY AND VERIFY
	3B	OOOOOOOOOO		WRITE BUFFER
	3C	OOOOOOOOOO		READ BUFFER
	3D		O O	UPDATE BLOCK
	3E	O	OO O	READ LONG
	3F	O	O O	WRITE LONG

Tabelle 365: (fortgesetzt)

D - DIRECT ACCESS DEVICE	Device Column Key
.T - SEQUENTIAL ACCESS DEVICE	M = Mandatory
. L - PRINTER DEVICE	O = Optional
. P - PROCESSOR DEVICE	V = Vendor Specific
. .W - WRITE ONCE READ MULTIPLE DEVICE	R = Reserved
. . R - READ ONLY (CD-ROM) DEVICE	
. . S - SCANNER DEVICE	
. . .O - OPTICAL MEMORY DEVICE	
. . . M - MEDIA CHANGER DEVICE	
. . . . C - COMMUNICATION DEVICE	
. . . .	

OP	DTLPWRSOMC	Description
40	OOOOOOOOOO	CHANGE DEFINITION
41	O	WRITE SAME
42	O	READ SUB-CHANNEL
43	O	READ TOC
44	O	READ HEADER
45	O	PLAY AUDIO(10)
46		
47	O	PLAY AUDIO MSF
48	O	PLAY AUDIO TRACK INDEX
49	O	PLAY TRACK RELATIVE(10)
4A		
4B	O	PAUSE RESUME
4C	OOOOOOOOOO	LOG SELECT
4D	OOOOOOOOOO	LOG SENSE
4E		
4F		
50		
51		
52		
53		
54		
55	OOO OOOOOO	MODE SELECT(10)

56		
57		
58		
59		
5A	OOO OOOOOO	MODE SENSE(10)
5B		
5C		
5D		
5E		
5F		

Tabelle 365: (Schlußteil)

D - DIRECT ACCESS DEVICE	Device Column Key
.T - SEQUENTIAL ACCESS DEVICE	M = Mandatory
. L - PRINTER DEVICE	O = Optional
. P - PROCESSOR DEVICE	V = Vendor Specific
. .W - WRITE ONCE READ MULTIPLE DEVICE	R = Reserved
. . R - READ ONLY (CD-ROM) DEVICE	
. . S - SCANNER DEVICE	
. . .O - OPTICAL MEMORY DEVICE	
. . . M - MEDIA CHANGER DEVICE	
. . . C - COMMUNICATION DEVICE	
. . . .	

OP	DTLPWRSOMC	Description
A0		
A1		
A2		
A3		
A4		
A5	M	MOVE MEDIUM
A5	O	PLAY AUDIO(12)
A6	O	EXCHANGE MEDIUM
A7		
A8	O	GET MESSAGE(12)
A8	OO O	READ(12)
A9	O	PLAY TRACK RELATIVE(12)
AA	O	SEND MESSAGE(12)
AA	O O	WRITE(12)
AB		
AC	O	ERASE(12)
AD		
AE	O O	WRITE AND VERIFY(12)
AF	OO O	VERIFY(12)
B0	OO O	SEARCH DATA HIGH(12)
B1	OO O	SEARCH DATA EQUAL(12)
B2	OO O	SEARCH DATA LOW(12)
B3	OO O	SET LIMITS(12)
B4		
B5		
B5	O	REQUEST VOLUME ELEMENT ADDRESS
B6		
B6	O	SEND VOLUME TAG

```
|        B7        O   READ DEFECT DATA(12)                                  |
|        B8                                                                  |
|        B8         O  READ ELEMENT STATUS                                   |
|        B9                                                                  |
|        BA                                                                  |
|        BB                                                                  |
|        BC                                                                  |
|        BD                                                                  |
|        BE                                                                  |
|        BF                                                                  |
+============================================================================+
```

D Beispielprogramme

Hier ist nun das C Beispielprogramm. Es erfragt Hersteller und Modell und gibt aus, ob ein Medium geladen ist.

```
#define DEVICE "/dev/sgc"
/* Example program to demonstrate the generic SCSI interface */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <scsi/sg.h>

#define SCSI_OFF sizeof(struct sg_header)
static unsigned char cmd[SCSI_OFF + 18];      /* SCSI command buffer */
int fd;                                 /* SCSI device/file descriptor */

/* process a complete scsi cmd. Use the generic scsi interface. */
static int handle_scsi_cmd(unsigned cmd_len,         /* command length */
                           unsigned in_size,         /* input data size */
                           unsigned char *i_buff,    /* input buffer */
                           unsigned out_size,        /* output data size */
                           unsigned char *o_buff     /* output buffer */
                           )
{
    int status = 0;
    struct sg_header *sg_hd;

    /* safety checks */
    if (!cmd_len) return -1;            /* need a cmd_len != 0 */
    if (!i_buff) return -1;             /* need an input buffer != NULL */
#ifdef SG_BIG_BUFF
    if (SCSI_OFF + cmd_len + in_size > SG_BIG_BUFF) return -1;
    if (SCSI_OFF + out_size > SG_BIG_BUFF) return -1;
#else
    if (SCSI_OFF + cmd_len + in_size > 4096) return -1;
    if (SCSI_OFF + out_size > 4096) return -1;
#endif

    if (!o_buff) out_size = 0;
```

```
    /* generic scsi device header construction */
    sg_hd = (struct sg_header *) i_buff;
    sg_hd->reply_len   = SCSI_OFF + out_size;
    sg_hd->twelve_byte = cmd_len == 12;
    sg_hd->result = 0;
#if     0
    sg_hd->pack_len    = SCSI_OFF + cmd_len + in_size; /* not necessary */
    sg_hd->pack_id;     /* not used */
    sg_hd->other_flags; /* not used */
#endif

    /* send command */
    status = write( fd, i_buff, SCSI_OFF + cmd_len + in_size );
    if ( status < 0 || status != SCSI_OFF + cmd_len + in_size ||
                       sg_hd->result ) {
        /* some error happened */
        fprintf( stderr, "write(generic) result = 0x%x cmd = 0x%x\n",
                    sg_hd->result, i_buff[SCSI_OFF] );
        perror("");
        return status;
    }

    if (!o_buff) o_buff = i_buff;       /* buffer pointer check */
    sg_hd = (struct sg_header *) o_buff;

    /* retrieve result */
    status = read( fd, o_buff, SCSI_OFF + out_size);
    if ( status < 0 || status != SCSI_OFF + out_size || sg_hd->result ) {
        /* some error happened */
        fprintf( stderr, "read(generic) result = 0x%x cmd = 0x%x\n",
                sg_hd->result, o_buff[SCSI_OFF] );
        fprintf( stderr, "read(generic) sense "
                "%x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x\n",
                sg_hd->sense_buffer[0],         sg_hd->sense_buffer[1],
                sg_hd->sense_buffer[2],         sg_hd->sense_buffer[3],
                sg_hd->sense_buffer[4],         sg_hd->sense_buffer[5],
                sg_hd->sense_buffer[6],         sg_hd->sense_buffer[7],
                sg_hd->sense_buffer[8],         sg_hd->sense_buffer[9],
                sg_hd->sense_buffer[10],        sg_hd->sense_buffer[11],
                sg_hd->sense_buffer[12],        sg_hd->sense_buffer[13],
                sg_hd->sense_buffer[14],        sg_hd->sense_buffer[15]);
        if (status < 0)
            perror("");
    }
    /* Look if we got what we expected to get */
    if (status == SCSI_OFF + out_size) status = 0; /* got them all */

    return status;  /* 0 means no error */
}

#define INQUIRY_CMD     0x12
#define INQUIRY_CMDLEN  6
#define INQUIRY_REPLY_LEN 96
#define INQUIRY_VENDOR  8       /* Offset in reply data to vendor name */
```

```
/* request vendor brand and model */
static unsigned char *Inquiry ( void )
{
  unsigned char Inqbuffer[ SCSI_OFF + INQUIRY_REPLY_LEN ];
  unsigned char cmdblk [ INQUIRY_CMDLEN ] =
      { INQUIRY_CMD,  /* command */
                  0,  /* lun/reserved */
                  0,  /* page code */
                  0,  /* reserved */
  INQUIRY_REPLY_LEN,  /* allocation length */
                  0 };/* reserved/flag/link */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- cmd
   * +------------------+
   * | copy of cmdblk   | <- cmd + SCSI_OFF
   * +------------------+
   */

  if (handle_scsi_cmd(sizeof(cmdblk), 0, cmd,
                      sizeof(Inqbuffer) - SCSI_OFF, Inqbuffer )) {
      fprintf( stderr, "Inquiry failed\n" );
      exit(2);
  }
  return (Inqbuffer + SCSI_OFF);
}

#define TESTUNITREADY_CMD 0
#define TESTUNITREADY_CMDLEN 6

#define ADD_SENSECODE 12
#define ADD_SC_QUALIFIER 13
#define NO_MEDIA_SC 0x3a
#define NO_MEDIA_SCQ 0x00
int TestForMedium ( void )
{
  /* request READY status */
  static unsigned char cmdblk [TESTUNITREADY_CMDLEN] = {
      TESTUNITREADY_CMD, /* command */
                      0, /* lun/reserved */
                      0, /* reserved */
                      0, /* reserved */
                      0, /* reserved */
                      0};/* reserved */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- cmd
   * +------------------+
   * | copy of cmdblk   | <- cmd + SCSI_OFF
```

```
    * +------------------+
    */

  if (handle_scsi_cmd(sizeof(cmdblk), 0, cmd,
                           0, NULL)) {
      fprintf (stderr, "Test unit ready failed\n");
      exit(2);
  }

  return
   *(((struct sg_header*)cmd)->sense_buffer +ADD_SENSECODE) !=
                                                    NO_MEDIA_SC ||
   *(((struct sg_header*)cmd)->sense_buffer +ADD_SC_QUALIFIER) !=
                                                    NO_MEDIA_SCQ;
}

int main( void )
{
  fd = open(DEVICE, O_RDWR);
  if (fd < 0) {
    fprintf( stderr, "Need read/write permissions for "DEVICE".\n" );
    exit(1);
  }

  /* print some fields of the Inquiry result */
  printf( "%s\n", Inquiry() + INQUIRY_VENDOR );

  /* look if medium is loaded */
  if (!TestForMedium()) {
    printf("device is unloaded\n");
  } else {
    printf("device is loaded\n");
  }
  return 0;
}
```