The Linux SCSI programming HOWTO

Table of Contents

Table of Contents

The Linux SCSI programming HOWTO

Heiko Eißfeldt `heiko@colossus.escape.de`

v1.5, 7 May 1996

This document deals with programming the Linux generic SCSI interface.

8.The Header Structure

9.Inquiry Command Example

10.The Sense Buffer

11.Example Using Sense Buffer

12.Ioctl Functions

13.Driver Defaults

14.Obtaining The Scsi Specifications

15.Related Information Sources

16.Other useful stuff

17.Other SCSI Access Interfaces

18.Final Comments

19.Acknowledgments

Appendix

20.Appendix

21.Error handling

22.Additional sense codes and additional sense code qualifiers

23.A SCSI command code quick reference

24.Example programs

1.What's New?

Newer kernels have changed the interface a bit. This affects a section formerly entitled 'rescanning the devices'. Now it is possible to add/remove SCSI devices on the fly.

Since kernel 1.3.98 some important header files have been moved/split (sg.h and scsi.h).

Some stupid bugs have been replaced by newer ones.

2.Introduction

This document is a guide to the installation and programming of the Linux generic SCSI interface.

It covers kernel prerequisites, device mappings, and basic interaction with devices. Some simple C programming examples are included. General knowledge of the SCSI command set is required; for more information on the SCSI standard and related information, see the appendix to this document.

Note the plain text version of this document lacks cross references (they show up as ``'').

3.What Is The Generic SCSI Interface?

The generic SCSI interface has been implemented to provide general SCSI access to (possibly exotic) pieces of SCSI hardware. It was developed by Lawrence Foard (`entropy@world.std.com`) and sponsored by Killy Corporation (see the comments in `scsi/sg.h`).

The interface makes special device handling possible from user level applications (i.e. outside the kernel). Thus, kernel driver development, which is more risky and difficult to debug, is not necessary.

However, if you don't program the driver properly it is possible to hang the SCSI bus, the driver, or the kernel. Therefore, it is important to properly program the generic driver and to first back up all files to avoid losing data. Another useful thing to do before running your programs is to issue a `sync` command to ensure that any buffers are flushed to disk, minimizing data loss if the system hangs.

Another advantage of the generic driver is that as long as the interface itself does not change, all applications are independent of new kernel development. In comparison, other low−level kernel drivers have to be synchronized with other internal kernel changes.

Typically, the generic driver is used to communicate with new SCSI hardware devices that require special user applications to be written to take advantage of their features (e.g. scanners, printers, CD−ROM jukeboxes). The generic interface allows these to be written quickly.

4.What Are The Requirements To Use It?

4.1 Kernel Configuration

You must have a supported SCSI controller, obviously. Furthermore, your kernel must have controller support as well as generic support compiled in. Configuring the Linux kernel (via `make config` under /usr/src/linux) typically looks like the following:

```
 ...
*
* 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
*
 ...
```

If available, modules can of course be build instead.

4.2 Device Files

The generic SCSI driver uses its own device files, separate from those used by the other SCSI device drivers. They can be generated using the `MAKEDEV` script, typically found in the `/dev` directory. Running `MAKEDEV sg` produces these files:

```
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 numbers
```

Note that these are character devices for raw access. On some systems these devices may be called `/dev/{sg0,sg1,...}`, depending on your installation, so adjust the following examples accordingly.

4.3 Device Mapping

These device files are dynamically mapped to SCSI id/LUNs on your SCSI bus (LUN = logical unit). The mapping allocates devices consecutively for each LUN of each device on each SCSI bus found at time of the SCSI scan, beginning at the lower LUNs/ids/buses. It starts with the first SCSI controller and continues without interruption with all following controllers. This is currently done in the initialisation of the SCSI driver.

For example, assuming you had three SCSI devices hooked up with ids 1, 3, and 5 on the first SCSI bus (each having one LUN), then the following mapping would be in effect:

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

If you now add a new device with id 4, then the mapping (after the next rescan) will be:

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

Notice the change for id 5 −− the corresponding device is no longer mapped to `/dev/sgc` but is now under `/dev/sgd`.

Luckily newer kernels allow for changing this order.

Dynamically insert and remove SCSI devices

If a newer kernel and the `/proc` file system is running, a non−busy device can be removed and installed 'on the fly'.

To remove a SCSI device:

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

and similar, to add a SCSI device, do

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

where

```
a == hostadapter id (first one being 0)
b == SCSI channel on hostadapter (first one being 0)
c == ID
d == LUN (first one being 0)
```

So in order to swap the `/dev/sgc` and `/dev/sgd` mappings from the previous example, we could do

```
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
```

since generic devices are mapped in the order of their insertion.

When adding more devices to the scsi bus keep in mind there are limited spare entries for new devices. The memory has been allocated at boot time and has room for 2 more devices.

5.Programmers Guide

The following sections are for programmers who want to use the generic SCSI interface in their own applications. An example will be given showing how to access a SCSI device with the INQUIRY and the TESTUNITREADY commands.

When using these code examples, note the following:

- the location of the header files `sg.h` and `scsi.h` has changed in kernel version 1.3.98. Now these files are located at `/usr/src/linux/include/scsi`, which is hopefully linked to `/usr/include/scsi`. Previously they were in `/usr/src/linux/drivers/scsi`. We assume a newer kernel in the following text.
- the generic SCSI interface was extended in kernel version 1.1.68; the examples require at least this version. But please avoid kernel version 1.1.77 up to 1.1.89 and 1.3.52 upto 1.3.56 since they had a broken generic scsi interface.
- the constant DEVICE in the header section describing the accessed device should be set according to your available devices (see section sec–header .

6.Overview Of Device Programming

The header file `include/scsi/sg.h` contains a description of the interface (this is based on kernel version 1.3.98):

```
struct sg_header
 {
  int pack_len;
                   /* length of incoming packet (including header) */
  int reply_len;   /* maximum length of expected reply */
  int pack_id;     /* id number of packet */
  int result;      /* 0==ok, otherwise refer to errno codes */
  unsigned int twelve_byte:1;
               /* Force 12 byte command length for group 6 & 7 commands  */
  unsigned int other_flags:31;                  /* for future use */
  unsigned char sense_buffer[16]; /* used only by reads */
  /* command follows then data for command */
 };
```

This structure describes how a SCSI command is to be processed and has room to hold the results of the execution of the command. The individual structure components will be discussed later in section sec−header .

The general way of exchanging data with the generic driver is as follows: to send a command to an opened generic device, `write()` a block containing these three parts to it:

```
struct sg_header
SCSI command
data to be sent with the command
```

To obtain the result of a command, `read()` a block with this (similar) block structure:

```
struct sg_header
data coming from the device
```

This is a general overview of the process. The following sections describe each of the steps in more detail.

NOTE: Up to recent kernel versions, it is necessary to block the SIGINT signal between the `write()` and the corresponding `read()` call (i.e. via `sigprocmask()`). A return after the `write()` part without any `read()` to fetch the results will block on subsequent accesses. This signal blocking has not yet been included in the example code. So better do not issue SIGINT (a la ^C) when running these examples.

7.Opening The Device

A generic device has to be opened for read and write access:

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

(This is the case even for a read−only hardware device such as a cdrom drive).

We have to perform a `write` to send the command and a `read` to get back any results. In the case of an error the return code is negative (see section sec−errorhandling for a complete list).

8.The Header Structure

The header structure `struct sg_header` serves as a controlling layer between the application and the kernel driver. We now discuss its components in detail.

int pack_len

: defines the size of the block written to the driver. This is defined within the kernel for internal use.

int reply_len

: defines the size of the block to be accepted at reply. This is defined from the application side.

int pack_id

: This field helps to assign replies to requests. The application can supply a unique id for each request. Suppose you have written several commands (say 4) to one device. They may work in parallel, one being the fastest. When getting replies via 4 reads, the replies do not have to have the order of the requests. To identify the correct reply for a given request one can use the `pack_id` field. Typically its value is incremented after each request (and wraps eventually). The maximum amount of outstanding requests is limited by the kernel to SG_MAX_QUEUE (eg 4).

int result

: the result code of a `read` or `write` call. This is (sometimes) defined from the generic driver (kernel) side. It is safe to set it to null before the `write` call. These codes are defined in `errno.h` (0 meaning no error).

unsigned int twelve_byte:1

This field is necessary only when using non−standard vendor specific commands (in the range 0xc0 − 0xff). When these commands have a command length of 12 bytes instead of 10, this field has to be set to one before the write call. Other command lengths are not supported. This is defined from the application side.

unsigned char sense_buffer[16]

This buffer is set after a command is completed (after a `read()` call) and contains the SCSI sense code. Some command results have to be read from here (e.g. for `TESTUNITREADY`). Usually it contains just zero bytes. The value in this field is set by the generic driver (kernel) side.

The following example function interfaces directly with the generic kernel driver. It defines the header structure, sends the command via `write`, gets the result via `read` and does some (limited) error checking. The sense buffer data is available in the output buffer (unless a NULL pointer has been given, in which case it's in the input buffer). We will use it in the examples which follow.

Note: Set the value of `DEVICE` to your device descriptor.

```
#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;      /* no output buffer, no output size */

    /* 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 */

    /* 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) 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("");
    }
    /* 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 */
}
```

While this may look somewhat complex at first appearance, most of the code is for error checking and reporting (which is useful even after the code is working).

`Handle_SCSI_cmd` has a generalized form for all SCSI commands types, falling into each of these categories:

```
       Data Mode               | Example Command
===============================================
neither input nor output data  | test unit ready
 no input data, output data    | inquiry, read
 input data, no output data    | mode select, write
   input data, output data     | mode sense
```

9.Inquiry Command Example

One of the most basic SCSI commands is the `INQUIRY` command, used to identify the type and make of the device. Here is the definition from the SCSI–2 specification (for details refer to the SCSI–2 standard).

```
                              Table 44: INQUIRY Command
+=====-========-========-========-========-========-========-========-========+
|  Bit|   7    |   6    |   5    |   4    |   3    |   2    |   1    |   0    |
|Byte |        |        |        |        |        |        |        |        |
|=====+=======================================================================|
| 0   |                           Operation Code (12h)                        |
|-----+-----------------------------------------------------------------------|
| 1   | Logical Unit Number      |                  Reserved         |  EVPD  |
|-----+-----------------------------------------------------------------------|
| 2   |                           Page Code                                   |
|-----+-----------------------------------------------------------------------|
| 3   |                           Reserved                                    |
|-----+-----------------------------------------------------------------------|
| 4   |                           Allocation Length                           |
|-----+-----------------------------------------------------------------------|
| 5   |                           Control                                     |
+=============================================================================+
```

The output data are as follows:

```
                       Table 45: Standard INQUIRY Data Format
+=====-========-========-========-========-========-========-========-========+
|  Bit|   7    |   6    |   5    |   4    |   3    |   2    |   1    |   0    |
|Byte |        |        |        |        |        |        |        |        |
|=====+==========================+============================================|
| 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   |                                                                             |
+===================================================================================+
```

The next example uses the low−level function `handle_SCSI_cmd` to perform the Inquiry SCSI command.

We first append the command block to the generic header, then call `handle_SCSI_cmd`. Note that the output buffer size argument for the `handle_SCSI_cmd` call excludes the generic header size. After command completion the output buffer contains the requested data, unless an error occurred.

```
#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);
}
```

The example above follows this structure. The Inquiry function copies its command block behind the generic header (given by `SCSI_OFF`). Input data is not present for this command. `Handle_SCSI_cmd` will define the header structure. We can now implement the function `main` to complete this working example program.

```
void 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 );
}
```

We first open the device, check for errors, and then call the higher level subroutine. Then we print the results in human readable format including the vendor, product, and revision.

Note: There is more information in the Inquiry result than this little program gives. You may want to extend the program to give device type, ANSI version etc. The device type is of special importance, since it determines the mandatory and optional command sets for this device. If you don't want to program it yourself, you may want to use the scsiinfo program from Eric Youngdale, which requests nearly all information about an SCSI device. Look at tsx−11.mit.edu in pub/Linux/ALPHA/scsi.

10.The Sense Buffer

Commands with no output data can give status information via the sense buffer (which is part of the header structure). Sense data is available when the previous command has terminated with a CHECK CONDITION status. In this case the kernel automatically retrieves the sense data via a REQUEST SENSE command. Its structure is:

```
+=====-========-========-========-========-========-========-========-========+
|  Bit|   7    |   6    |   5    |   4    |   3    |   2    |   1    |   0    |
|Byte |        |        |        |        |        |        |        |        |
|=====+========+===============================================================|
| 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  |                                                               |
|- - -+------------               Sense-Key Specific                        ---|
| 17  |                                                                        |
|-----+------------------------------------------------------------------------|
| 18  |                                                                        |
|- - -+---                        Additional Sense Bytes                    ---|
| n   |                                                                        |
+==============================================================================+
```

Note: The most useful fields are Sense Key (see section sec−sensekeys), Additional Sense Code and Additional Sense Code Qualifier (see section sec−sensecodes). The latter two are used combined as a pair.

11.Example Using Sense Buffer

Here we will use the TEST UNIT READY command to check whether media is loaded into our device. The header declarations and function `handle_SCSI_cmd` from the inquiry example will be needed as well.

```
                    Table 73: TEST UNIT READY Command
+=====-========-========-========-========-========-========-========-========+
|  Bit|   7    |   6    |   5    |   4    |   3    |   2    |   1    |   0    |
|Byte |        |        |        |        |        |        |        |        |
|=====+=======================================================================|
| 0   |                           Operation Code (00h)                        |
|-----+-----------------------------------------------------------------------|
| 1   | Logical Unit Number      |                  Reserved                  |
|-----+-----------------------------------------------------------------------|
| 2   |                           Reserved                                    |
|-----+-----------------------------------------------------------------------|
| 3   |                           Reserved                                    |
|-----+-----------------------------------------------------------------------|
| 4   |                           Reserved                                    |
|-----+-----------------------------------------------------------------------|
| 5   |                           Control                                     |
+=============================================================================+
```

Here is the function which implements it:

```
#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 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;
}
```

Combined with this `main` function we can do the check.

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

  /* look if medium is loaded */

  if (!TestForMedium()) {
    printf("device is unloaded\n");
  } else {
    printf("device is loaded\n");
  }
}
```

The file `generic_demo.c` from the appendix contains both examples.

12.Ioctl Functions

There are two ioctl functions available:

- `ioctl(fd, SG_SET_TIMEOUT, &Timeout);` sets the timeout value to `Timeout` * 10 milliseconds. `Timeout` has to be declared as int.
- `ioctl(fd, SG_GET_TIMEOUT, &Timeout);` gets the current timeout value. `Timeout` has to be declared as int.

13.Driver Defaults

13.1 Transfer Lengths

Currently (at least up to kernel version 1.1.68) input and output sizes have to be less than or equal than 4096 bytes unless the kernel has been compiled with `SG_BIG_BUFF` defined, if which case it is limited to `SG_BIG_BUFF` (e.g. 32768) bytes. These sizes include the generic header as well as the command block on input. `SG_BIG_BUFF` can be safely increased upto (131072 − 512). To take advantage of this, a new kernel has to be compiled and booted, of course.

13.2 Timeout And Retry Values

The default timeout value is set to one minute (`Timeout` = 6000). It can be changed through an ioctl call (see section sec−ioctl). The default number of retries is one.

14.Obtaining The Scsi Specifications

There are standards entitled SCSI−1 and SCSI−2 (and possibly soon SCSI−3). The standards are mostly upward compatible.

The SCSI−1 standard is (in the author's opinion) mostly obsolete, and SCSI−2 is the most widely used. SCSI−3 is very new and very expensive. These standardized command sets specify mandatory and optional commands for SCSI manufacturers and should be preferred over the vendor specific command extensions which are not standardized and for which programming information is seldom available. Of course sometimes there is no alternative to these extensions.

Electronic copies of the latest drafts are available via anonymous ftp from:

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

(I got my SCSI specification from the Yggdrasil Linux CD−ROM in the directory /usr/doc/scsi−2 and /usr/doc/scsi−1).

The SCSI FAQ also lists the following sources of printed information:

The SCSI specification: Available from:

```
          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 from ANSI, 1430 Broadway, NY, NY 10018

IN-DEPTH EXPLORATION OF SCSI can be obtained from
Solution Technology, Attn: SCSI Publications, POB 104, Boulder Creek,
CA 95006, (408)338-4285, FAX (408)338-4374

THE SCSI ENCYLOPEDIA and the SCSI BENCH REFERENCE can be obtained from
ENDL Publishing, 14426 Black Walnut Ct., Saratoga, CA 95090,
(408)867-6642, FAX (408)867-2115

SCSI: UNDERSTANDING THE SMALL COMPUTER SYSTEM INTERFACE was published
by Prentice-Hall, ISBN 0-13-796855-8
```

15.Related Information Sources

15.1 HOWTOs and FAQs

The Linux **SCSI−HOWTO** by Drew Eckhardt covers all supported SCSI controllers as well as device specific questions. A lot of troubleshooting hints are given. It is available from sunsite.unc.edu in /pub/Linux/docs/LDP and its mirror sites.

General questions about SCSI are answered in the **SCSI−FAQ** from the newsgroup Comp.Periphs.Scsi (available on tsx−11 in pub/linux/ALPHA/scsi and mirror sites).

15.2 Mailing list

There is a **mailing list** for bug reports and questions regarding SCSI development under Linux. To join, send email to `majordomo@vger.rutgers.edu` with the line `subscribe linux-scsi` in the body of the message. Messages should be posted to `linux-scsi@vger.rutgers.edu`. Help text can be requested by sending the message line "help" to `majordomo@vger.rutgers.edu`.

15.3 Example code

sunsite.unc.edu: apps/graphics/hpscanpbm−0.3a.tar.gz

This package handles a HP scanjet scanner through the generic interface.

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

The cdwrite package uses the generic interface to write a cd image to a cd writer.

sunsite.unc.edu: apps/sound/cds/cdda2wav*.src.tar.gz

A shameless plug for my own application, which copies audio cd tracks into wav files.

16.Other useful stuff

Things that may come in handy. I don't have no idea if there are newer or better versions around. Feedback is welcome.

16.1 Device driver writer helpers

These documents can be found at the sunsite.unc.edu ftp server and its mirrors.

/pub/Linux/docs/kernel/kernel−hackers−guide

The LDP kernel hackers guide. May be a bit outdated, but covers the most fundamental things.

/pub/Linux/docs/kernel/drivers.doc.z

This document covers writing character drivers.

/pub/Linux/docs/kernel/tutorial.doc.z

Tutorial on writing a character device driver with code.

/pub/Linux/docs/kernel/scsi.paper.tar.gz

A Latex document describing howto write a SCSI driver.

/pub/Linux/docs/hardware/DEVICES

A list of device majors and minors used by Linux.

16.2 Utilities

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

Program to query a scsi device for operating parameters, defect lists, etc. An X−based interface is available which requires you have Tk/Tcl/wish installed. With the X−based interface you can easily alter the settings on the drive.

tsx−11.mit.edu: ALPHA/kdebug

A gdb extension for kernel debugging.

17.Other SCSI Access Interfaces

In Linux there is also another SCSI access method via SCSI_IOCTL_SEND_COMMAND ioctl calls, which is deprecated. Special tools like 'scsiinfo' utilize it.

There are some other similar interfaces in use in the un*x world, but not available for Linux:

1. CAM (Common Access Method) developed by Future Domain and other SCSI vendors. Linux has little support for a SCSI CAM system yet (mainly for booting from hard disk). CAM even supports target mode, so one could disguise ones computer as a peripheral hardware device (e.g. for a small SCSI net).
2. ASPI (Advanced SCSI Programming Interface) developed by Adaptec. This is the de facto standard for MS−DOS machines.

There are other application interfaces from SCO(TM), NeXT(TM), Silicon Graphics(TM) and SUN(TM) as well.

18.Final Comments

The generic SCSI interface bridges the gap between user applications and specific devices. But rather than bloating a lot of programs with similar sets of low−level functions, it would be more desirable to have a shared library with a generalized set of low−level functions for a particular purpose. The main goal should be to have independent layers of interfaces. A good design would separate an application into low−level and hardware independent routines. The low−level routines could be put into a shared library and made available for all applications. Here, standardized interfaces should be followed as much as possible before making new ones.

By now you should know more than I do about the Linux generic SCSI interface. So you can start developing powerful applications for the benefit of the global Linux community now...

19.Acknowledgments

Special thanks go to Jeff Tranter for proofreading and enhancing the text considerably as well as to Carlos Puchol for useful comments. Drew Eckhardt's and Eric Youngdale's help on my first (dumb) questions about the use of this interface has been appreciated.

20.Appendix

21.Error handling

The functions `open`, `ioctl`, `write` and `read` can report errors. In this case their return value is −1 and the global variable errno is set to the error number. The errno values are defined in `/usr/include/errno.h`. Possible values are:

```
Function | Error        | Description
=========|==============|===============================================
open     | ENXIO        | not a valid device
         | EACCES       | access mode is not read/write (O_RDWR)
         | EBUSY        | device was requested for nonblocking access,
         |              | but is busy now.
         | ERESTARTSYS  | this indicates an internal error. Try to
         |              | make it reproducible and inform the SCSI
         |              | channel (for details on bug reporting
         |              | see Drew Eckhardts SCSI-HOWTO).
ioctl    | ENXIO        | not a valid device
read     | EAGAIN       | the device would block. Try again later.
         | ERESTARTSYS  | this indicates an internal error. Try to
         |              | make it reproducible and inform the SCSI
         |              | channel (for details on bug reporting
         |              | see Drew Eckhardts SCSI-HOWTO).
write    | EIO          | the length is too small (smaller than the
         |              | generic header struct). Caution: Currently
         |              | there is no overlength checking.
         | EAGAIN       | the device would block. Try again later.
         | ENOMEM       | memory required for this request could not be
         |              | allocated. Try later again unless you
         |              | exceeded the maximum transfer size (see above)
select   |              | none
close    |              | none
```

For read/write positive return values indicate as usual the amount of bytes that have been successfully transferred. This should equal the amount you requested.

21.1 Error status decoding

Furthermore a detailed reporting is done via the kernels `hd_status` and the devices `sense_buffer` (see section sec−sensebuff) both from the generic header structure.

The meaning of `hd_status` can be found in `drivers/scsi/scsi.h`: This `unsigned int` is composed out of different parts:

```
  lsb  |    ...    |    ...    | msb
=======|===========|===========|============
status | sense key | host code | driver byte
```

These macros from `drivers/scsi/scsi.h` are available, but unfortunately cannot be easily used due to weird header file interdependencies. This has to be cleaned.

```
        Macro          | Description
=======================|=================================================
status_byte(hd_status) | The SCSI device status. See section Status codes
msg_byte(hd_status)    | From the device. See section SCSI sense keys
host_byte(hd_status)   | From the kernel. See section Hostcodes
driver_byte(hd_status) | From the kernel. See section midlevel codes
```

21.2 Status codes

The following status codes from the SCSI device (defined in `scsi/scsi.h`) are available.

```
Value | Symbol
======|======================
0x00  | GOOD
0x01  | CHECK_CONDITION
0x02  | CONDITION_GOOD
0x04  | BUSY
0x08  | INTERMEDIATE_GOOD
0x0a  | INTERMEDIATE_C_GOOD
```

```
0x0c   | RESERVATION_CONFLICT
```

Note that these symbol values have been **shifted right once**. When the status is CHECK_CONDITION, the sense data in the sense buffer is valid (check especially the additional sense code and additional sense code qualifier).

These values carry the meaning from the SCSI−2 specification:

```
                           Table 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.
```

21.3 SCSI Sense Keys

These kernel symbols (from `scsi/scsi.h`) are predefined:

```
Value | Symbol
======|================
0x00  | NO_SENSE
0x01  | RECOVERED_ERROR
0x02  | NOT_READY
0x03  | MEDIUM_ERROR
0x04  | HARDWARE_ERROR
0x05  | ILLEGAL_REQUEST
0x06  | UNIT_ATTENTION
0x07  | DATA_PROTECT
0x08  | BLANK_CHECK
0x0a  | COPY_ABORTED
0x0b  | ABORTED_COMMAND
0x0d  | VOLUME_OVERFLOW
0x0e  | MISCOMPARE
```

A verbatim list from the SCSI–2 doc follows (from section 7.2.14.3):

```
                    Table 69: Sense Key (0h-7h) Descriptions
+========-=============================================================+
| Sense  |  Description                                                |
|  Key   |                                                             |
|--------+-------------------------------------------------------------|
|   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.

Table 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.                                                         |
+==============================================================================+
```

21.4 Host codes

The following host codes are defined in `drivers/scsi/scsi.h`. They are set by the kernel driver.

```
Value | Symbol          | Description
======|=================|========================================
0x00  | DID_OK          | No error
0x01  | DID_NO_CONNECT  | Couldn't connect before timeout period
0x02  | DID_BUS_BUSY    | BUS stayed busy through time out period
0x03  | DID_TIME_OUT    | TIMED OUT for other reason
0x04  | DID_BAD_TARGET  | BAD target
0x05  | DID_ABORT       | Told to abort for some other reason
0x06  | DID_PARITY      | Parity error
0x07  | DID_ERROR       | internal error
0x08  | DID_RESET       | Reset by somebody
0x09  | DID_BAD_INTR    | Got an interrupt we weren't expecting
```

21.5 Driver codes

The midlevel driver categorizes the returned status from the lowlevel driver based on the sense key from the device. It suggests some actions to be taken such as retry, abort or remap. The routine scsi_done from scsi.c does a very differentiated handling based on host_byte(), status_byte(), msg_byte() and the suggestion. It then sets the driver byte to show what it has done. The driver byte is composed out of two nibbles: the driver status and the suggestion. Each half is composed from the below values being 'or'ed together (found in scsi.h).

```
Value | Symbol          | Description of Driver status
======|=================|========================================
0x00  | DRIVER_OK       | No error
0x01  | DRIVER_BUSY     | not used
0x02  | DRIVER_SOFT     | not used
```

```
0x03  | DRIVER_MEDIA    | not used
0x04  | DRIVER_ERROR    | internal driver error
0x05  | DRIVER_INVALID  | finished (DID_BAD_TARGET or DID_ABORT)
0x06  | DRIVER_TIMEOUT  | finished with timeout
0x07  | DRIVER_HARD     | finished with fatal error
0x08  | DRIVER_SENSE    | had sense information available
```

```
Value | Symbol          | Description of suggestion
======|=================|========================================
0x10  | SUGGEST_RETRY   | retry the SCSI request
0x20  | SUGGEST_ABORT   | abort the request
0x30  | SUGGEST_REMAP   | remap the block (not yet implemented)
0x40  | SUGGEST_DIE     | let the kernel panic
0x80  | SUGGEST_SENSE   | get sense information from the device
0xff  | SUGGEST_IS_OK   | nothing to be done
```

22.Additional sense codes and additional sense code qualifiers

When the status of the executed SCSI command is CHECK_CONDITION, sense data is available in the sense buffer. The additional sense code and additional sense code qualifier are contained in that buffer.

From the SCSI−2 specification I include two tables. The first is in lexical, the second in numerical order.

22.1 ASC and ASCQ in lexical order

The following table list gives a list of descriptions and device types they apply to.

```
+=============================================================================+
|           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

Table 71: (continued)

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

Table 71: (continued)

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
+=============================================================================+
```

```
Table 71: (concluded)
+=============================================================================+
| 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.                      |
|------------------------------------------------------------------------------|
```

22.2 ASC and ASCQ in numerical order

```
                        Table 364: ASC and ASCQ Assignments

+==============================================================================+
|           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                                  |
+==============================================================================+
```

Table 364: (continued)

```
+==============================================================================+
|           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

Table 364: (continued)

```
+==============================================================================+
|           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                        |
+============================================================================+
```

```
Table 364: (continued)
+============================================================================+
```

```
          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
```

Table 364: (continued)

```
+=============================================================================+
|           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
+=============================================================================+
```

```
Table 364: (concluded)
+=============================================================================+
|           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.                    |
+=============================================================================+
```

23.A SCSI command code quick reference

```
Table 365 is a numerical order listing of the command operation codes.

                        Table 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                                      |
+===========================================================================+
```

```
Table 365: (continued)
+===========================================================================+
|           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                                                 |
+=============================================================================+
```

```
Table 365: (continued)
+=============================================================================+
|           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                                                                   |
+=============================================================================+
```

```
Table 365: (concluded)
+=============================================================================+
|           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                                                                   |
+=============================================================================+
```

24.Example programs

Here is the C example program, which requests manufacturer/model and reports if a medium is loaded in the device.

```
#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 */

    /* 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;
}

void 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");
  }
}
```