Summary of *HDIO_* ioctl calls

- Edward A. Falk <efalk@google.com>

November, 2004

This document attempts to describe the ioctl(2) calls supported by the HD/IDE layer. These are by-and-large implemented (as of Linux 5.11) drivers/ata/libata-scsi.c.

ioctl values are listed in <linux/hdreg.h>. As of this writing, they are as follows:

ioctls that pass argument pointers to user space:

HDIO_GETGEO	get device geometry
HDIO_GET_32BIT	get current io_32bit setting
HDIO_GET_IDENTITY	get IDE identification info
HDIO_DRIVE_TASKFILE	execute raw taskfile
HDIO_DRIVE_TASK	execute task and special drive command
HDIO_DRIVE_CMD	execute a special drive command

ioctls that pass non-pointer values:

HDIO_SET_32BIT	change io_32bit flags

The information that follows was determined from reading kernel source code. It is likely that some corrections will be made over time.

General:

Unless otherwise specified, all ioctl calls return 0 on success and -1 with errno set to an appropriate value on error.

Unless otherwise specified, all ioctl calls return -1 and set errno to EFAULT on a failed attempt to copy data to or from user address space.

Unless otherwise specified, all data structures and constants are defined in <linux/hdreg.h>

HDIO_GETGEO

get device geometry

usage:

```
struct hd_geometry geom;

ioctl(fd, HDIO_GETGEO, &geom);
```

inputs:

none

outputs:

hd_geometry structure containing:

heads	number of heads
sectors	number of sectors/track
cylinders	number of cylinders, mod 65536
start	starting sector of this partition.

error returns:

- EINVAL

  if the device is not a disk drive or floppy drive, or if the user passes a null pointer

notes:

Not particularly useful with modern disk drives, whose geometry is a polite fiction anyway. Modern drives are addressed purely by sector number nowadays (lba addressing), and the drive geometry is an abstraction which is actually subject to change. Currently (as of Nov 2004), the geometry values are the "bios" values -- presumably the values the drive had when Linux first booted.

In addition, the cylinders field of the hd_geometry is an unsigned short, meaning that on most architectures, this ioctl will not return a meaningful value on drives with more than 65535 tracks.

The start field is unsigned long, meaning that it will not contain a meaningful value for disks over 219 Gb in size.

HDIO_GET_IDENTITY

get IDE identification info

usage:

```
unsigned char identity[512];

ioctl(fd, HDIO_GET_IDENTITY, identity);
```

inputs:

none

outputs:

ATA drive identity information. For full description, see the IDENTIFY DEVICE and IDENTIFY PACKET DEVICE commands in the ATA specification.

error returns:

- EINVAL Called on a partition instead of the whole disk device
- ENOMSG IDENTIFY DEVICE information not available

notes:

Returns information that was obtained when the drive was probed. Some of this information is subject to change, and this ioctl does not re-probe the drive to update the information.

This information is also available from /proc/ide/hdX/identify

HDIO_GET_32BIT

get current io_32bit setting

usage:

```
long val;

ioctl(fd, HDIO_GET_32BIT, &val);
```

inputs:

none

outputs:

The value of the current io_32bit setting

notes:

0=16-bit, 1=32-bit, 2,3 = 32bit+sync

HDIO_DRIVE_TASKFILE

execute raw taskfile

Note:

If you don't have a copy of the ANSI ATA specification handy, you should probably ignore this ioctl.

- Execute an ATA disk command directly by writing the "taskfile" registers of the drive. Requires ADMIN and RAWIO access privileges.

usage:

```
struct {

  ide_task_request_t req_task;
  u8 outbuf[OUTPUT_SIZE];
  u8 inbuf[INPUT_SIZE];
} task;
memset(&task.req_task, 0, sizeof(task.req_task));
task.req_task.out_size = sizeof(task.outbuf);
task.req_task.in_size = sizeof(task.inbuf);
...
ioctl(fd, HDIO_DRIVE_TASKFILE, &task);
...
```

inputs:

(See below for details on memory area passed to ioctl.)

io_ports[8]	values to be written to taskfile registers

hob_ports[8]	high-order bytes, for extended commands.
out_flags	flags indicating which registers are valid
in_flags	flags indicating which registers should be returned
data_phase	see below
req_cmd	command type to be executed
out_size	size of output buffer
outbuf	buffer of data to be transmitted to disk
inbuf	buffer of data to be received from disk (see [1])

outputs:

io_ports[]	values returned in the taskfile registers
hob_ports[]	high-order bytes, for extended commands.
out_flags	flags indicating which registers are valid (see [2])
in_flags	flags indicating which registers should be returned
outbuf	buffer of data to be transmitted to disk (see [1])
inbuf	buffer of data to be received from disk

error returns:

- EACCES CAP_SYS_ADMIN or CAP_SYS_RAWIO privilege not set.
- ENOMSG Device is not a disk drive.
- ENOMEM Unable to allocate memory for task
- EFAULT req_cmd == TASKFILE_IN_OUT (not implemented as of 2.6.8)
- EPERM

  req_cmd == TASKFILE_MULTI_OUT and drive multi-count not yet set.
- EIO Drive failed the command.

notes:

[1] READ THE FOLLOWING NOTES *CAREFULLY*. THIS IOCTL IS FULL OF GOTCHAS. Extreme caution should be used with using this ioctl. A mistake can easily corrupt data or hang the system.

[2] Both the input and output buffers are copied from the user and written back to the user, even when not used.

[3] If one or more bits are set in out_flags and in_flags is zero, the following values are used for in_flags.all and written back into in_flags on completion.

- IDE_TASKFILE_STD_IN_FLAGS | (IDE_HOB_STD_IN_FLAGS << 8) if LBA48 addressing is enabled for the drive
- IDE_TASKFILE_STD_IN_FLAGS if CHS/LBA28

The association between in_flags.all and each enable bitfield flips depending on endianness; fortunately, TASKFILE only uses inflags.b.data bit and ignores all other bits. The end result is that, on any endian machines, it has no effect other than modifying in_flags on completion.

[4] The default value of SELECT is (0xa0|DEV_bit|LBA_bit) except for four drives per port chipsets. For four drives per port chipsets, it's (0xa0|DEV_bit|LBA_bit) for the first pair and (0x80|DEV_bit|LBA_bit) for the second pair.

[5] The argument to the ioctl is a pointer to a region of memory containing a ide_task_request_t structure, followed by an optional buffer of data to be transmitted to the drive, followed by an optional buffer to receive data from the drive.

Command is passed to the disk drive via the ide_task_request_t structure, which contains these fields:

io_ports[8]	values for the taskfile registers
hob_ports[8]	high-order bytes, for extended commands
out_flags	flags indicating which entries in the io_ports[] and hob_ports[] arrays contain valid values. Type ide_reg_valid_t.
in_flags	flags indicating which entries in the io_ports[] and hob_ports[] arrays are expected to contain valid values on return.
data_phase	See below
req_cmd	Command type, see below
out_size	output (user->drive) buffer size, bytes
in_size	input (drive->user) buffer size, bytes

When out_flags is zero, the following registers are loaded.

HOB_FEATURE	If the drive supports LBA48
HOB_NSECTOR	If the drive supports LBA48
HOB_SECTOR	If the drive supports LBA48
HOB_LCYL	If the drive supports LBA48
HOB_HCYL	If the drive supports LBA48
FEATURE	
NSECTOR	
SECTOR	
LCYL	
HCYL	
SELECT	First, masked with 0xE0 if LBA48, 0xEF otherwise; then, or'ed with the default value of SELECT.

If any bit in out_flags is set, the following registers are loaded.

HOB_DATA	If out_flags.b.data is set. HOB_DATA will travel on DD8-DD15 on little endian machines and on DD0-DD7 on big endian machines.
DATA	If out_flags.b.data is set. DATA will travel on DD0-DD7 on little endian machines and on DD8-DD15 on big endian machines.
HOB_NSECTOR	If out_flags.b.nsector_hob is set
HOB_SECTOR	If out_flags.b.sector_hob is set
HOB_LCYL	If out_flags.b.lcyl_hob is set
HOB_HCYL	If out_flags.b.hcyl_hob is set
FEATURE	If out_flags.b.feature is set
NSECTOR	If out_flags.b.nsector is set
SECTOR	If out_flags.b.sector is set
LCYL	If out_flags.b.lcyl is set
HCYL	If out_flags.b.hcyl is set
SELECT	Or'ed with the default value of SELECT and loaded regardless of out_flags.b.select.

Taskfile registers are read back from the drive into {io|hob}_ports[] after the command completes iff one of the following conditions is met; otherwise, the original values will be written back, unchanged.

1. The drive fails the command (EIO).
2. One or more than one bits are set in out_flags.
3. The requested data_phase is TASKFILE_NO_DATA.

HOB_DATA	If in_flags.b.data is set. It will contain DD8-DD15 on little endian machines and DD0-DD7 on big endian machines.
DATA	If in_flags.b.data is set. It will contain DD0-DD7 on little endian machines and DD8-DD15 on big endian machines.
HOB_FEATURE	If the drive supports LBA48
HOB_NSECTOR	If the drive supports LBA48
HOB_SECTOR	If the drive supports LBA48
HOB_LCYL	If the drive supports LBA48
HOB_HCYL	If the drive supports LBA48
NSECTOR	
SECTOR	
LCYL	
HCYL	

The data_phase field describes the data transfer to be performed. Value is one of:

TASKFILE_IN	
TASKFILE_MULTI_IN	
TASKFILE_OUT	
TASKFILE_MULTI_OUT	
TASKFILE_IN_OUT	
TASKFILE_IN_DMA	
TASKFILE_IN_DMAQ	== IN_DMA (queueing not supported)
TASKFILE_OUT_DMA	

TASKFILE_OUT_DMAQ	== OUT_DMA (queueing not supported)
TASKFILE_P_IN	unimplemented
TASKFILE_P_IN_DMA	unimplemented
TASKFILE_P_IN_DMAQ	unimplemented
TASKFILE_P_OUT	unimplemented
TASKFILE_P_OUT_DMA	unimplemented
TASKFILE_P_OUT_DMAQ	unimplemented

The req_cmd field classifies the command type. It may be one of:

IDE_DRIVE_TASK_NO_DATA	
IDE_DRIVE_TASK_SET_XFER	unimplemented
IDE_DRIVE_TASK_IN	
IDE_DRIVE_TASK_OUT	unimplemented
IDE_DRIVE_TASK_RAW_WRITE	

[6] Do not access {in|out}_flags->all except for resetting all the bits. Always access individual bit fields. ->all value will flip depending on endianness. For the same reason, do not use IDE_{TASKFILE|HOB}_STD_{OUT|IN}_FLAGS constants defined in hdreg.h.

HDIO_DRIVE_CMD

execute a special drive command

Note: If you don't have a copy of the ANSI ATA specification handy, you should probably ignore this ioctl.

usage:

```
u8 args[4+XFER_SIZE];

...
ioctl(fd, HDIO_DRIVE_CMD, args);
```

inputs:

Commands other than WIN_SMART:

args[0]	COMMAND
args[1]	NSECTOR
args[2]	FEATURE
args[3]	NSECTOR

WIN_SMART:

args[0]	COMMAND
args[1]	SECTOR
args[2]	FEATURE
args[3]	NSECTOR

outputs:

args[] buffer is filled with register values followed by any

data returned by the disk.

args[0]	status
args[1]	error
args[2]	NSECTOR
args[3]	undefined
args[4+]	NSECTOR * 512 bytes of data returned by the command.

error returns:

- EACCES Access denied: requires CAP_SYS_RAWIO
- ENOMEM Unable to allocate memory for task
- EIO Drive reports error

notes:

[1] For commands other than WIN_SMART, args[1] should equal args[3]. SECTOR, LCYL and HCYL are undefined. For WIN_SMART, 0x4f and 0xc2 are loaded into LCYL and HCYL respectively. In both cases SELECT will contain the default value for the drive. Please refer to HDIO_DRIVE_TASKFILE notes for the default value of SELECT.

[2] If NSECTOR value is greater than zero and the drive sets DRQ when interrupting for the command, NSECTOR * 512 bytes are read from the device into the area following NSECTOR. In the above example, the area would be args[4..4+XFER_SIZE]. 16bit PIO is used regardless of HDIO_SET_32BIT setting.

[3] If COMMAND == WIN_SETFEATURES && FEATURE == SETFEATURES_XFER && NSECTOR >= XFER_SW_DMA_0 && the drive supports any DMA mode, IDE driver will try to tune the transfer mode of the drive accordingly.

HDIO_DRIVE_TASK

execute task and special drive command

Note: If you don't have a copy of the ANSI ATA specification handy, you should probably ignore this ioctl.

usage:

```
u8 args[7];

...
ioctl(fd, HDIO_DRIVE_TASK, args);
```

inputs:

Taskfile register values:

args[0]	COMMAND
args[1]	FEATURE
args[2]	NSECTOR
args[3]	SECTOR
args[4]	LCYL
args[5]	HCYL
args[6]	SELECT

outputs:

Taskfile register values:

args[0]	status
args[1]	error
args[2]	NSECTOR
args[3]	SECTOR
args[4]	LCYL
args[5]	HCYL
args[6]	SELECT

error returns:

- EACCES Access denied: requires CAP_SYS_RAWIO
- ENOMEM Unable to allocate memory for task
- ENOMSG Device is not a disk drive.
- EIO Drive failed the command.

notes:

[1] DEV bit (0x10) of SELECT register is ignored and the appropriate value for the drive is used. All other bits are used unaltered.

HDIO_SET_32BIT

change io_32bit flags

usage:

```
int val;

ioctl(fd, HDIO_SET_32BIT, val);
```

inputs:
New value for io_32bit flag

outputs:
none

error return:

- EINVAL Called on a partition instead of the whole disk device
- EACCES Access denied: requires CAP_SYS_ADMIN
- EINVAL value out of range [0 3]
- EBUSY Controller busy