NAME

dkio – disk control operations

SYNOPSIS

#include <sys/dkio.h>
#include <sys/vtoc.h>

DESCRIPTION

Disk drivers support a set of **ioctl**(2) requests for disk controller, geometry, and partition information. Basic to these **ioctl( )** requests are the definitions in **<sys/dkio.h>**.

IOCTLS

The following **ioctl( )** requests set and/or retrieve the current disk controller, partitions, or geometry information:

DKIOCINFO The argument is a pointer to a **dk_cinfo** structure (described below). This structure tells the type of the controller and attributes about how bad-block processing is done on the controller.

```
/*
 * Structures and definitions for disk I/O control commands
 */
#define DK_DEVLEN      16     /* device name max length, */
                              /* including unit # and NULL */
/*
 * Used for controller info
 */
struct dk_cinfo {
        char     dki_cname[DK_DEVLEN];  /* controller name (no unit #)*/
        u_short  dki_ctype;             /* controller type */
        u_short  dki_flags;             /* flags */
        u_short  dki_cnum;              /* controller number */
        u_int    dki_addr;              /* controller address */
        u_int    dki_space;             /* controller bus type */
        u_int    dki_prio;              /* interrupt priority */
        u_int    dki_vec;               /* interrupt vector */
        char     dki_dname[DK_DEVLEN];  /* drive name (no unit #) */
        u_int    dki_unit;              /* unit number */
        u_int    dki_slave;             /* slave number */
        u_short  dki_partition;         /* partition number */
        u_short  dki_maxtransfer;       /* maximum transfer size */
                                        /* in DEV_BSIZE */
};
/*
 * Controller types
 */
#define DKC_UNKNOWN        0
#define DKC_CDROM          1             /* CD-ROM, SCSI or
                                            otherwise */
```

```
#define DKC_WDC2880         2
#define DKC_XXX_0           3              /* unassigned */
#define DKC_XXX_1           4              /* unassigned */
#define DKC_DSD5215         5
#define DKC_XY450           6
#define DKC_ACB4000         7
#define DKC_MD21            8
#define DKC_XXX_2           9              /* unassigned */
#define DKC_NCRFLOPPY       10
#define DKC_XD7053          11
#define DKC_SMSFLOPPY       12
#define DKC_SCSI_CCS        13             /* SCSI CCS compatible */
#define DKC_INTEL82072      14             /* native floppy chip */
#define DKC_PANTHER         15
#define DKC_SUN_IPI1        DKC_PANTHER    /* Sun Panther */
                                           /* VME/IPI controller */
#define DKC_MD              16             /* meta-disk (virtual-disk) */
                                           /* driver */
#define DKC_CDC_9057        17             /* CDC 9057-321 (CM-3) */
                                           /* IPI String Controller */
#define DKC_FJ_M1060        18             /* Fujitsu/Intellistor */
                                           /* IM1060 PI-3 SC */
#define DKC_INTEL82077      19             /* 82077 floppy disk */
                                           /* controller */
#define DKC_DIRECT          20             /* Intel direct attached */
                                           /* device (IDE) */
#define DKC_PCMCIA_MEM      21             /* PCMCIA memory disk-like */
                                           /* type */
#define DKC_PCMCIA_ATA      22             /* PCMCIA AT Attached type */

/*
 * Sun reserves up through 1023
 */

#define DKC_CUSTOMER_BASE     1024

/*
 * Flags
 */

#define DKI_BAD144          0x01  /* use DEC std 144 bad sector fwding */
#define DKI_MAPTRK          0x02  /* controller does track mapping */
#define DKI_FMTTRK          0x04  /* formats only full track at a time */
#define DKI_FMTVOL          0x08  /* formats only full volume */
                                  /* at a time*/
#define DKI_FMTCYL          0x10  /* formats only full cylinders */
                                  /* at a time*/
#define DKI_HEXUNIT         0x20  /* unit number printed as 3 hex */
```

```
                                        /* digits */
#define DKI_PCMCIA_PFD    0x40   /* PCMCIA pseudo-floppy memory card */
```

DKIOCGAPART The argument is a pointer to a **dk_allmap** structure (described below). This **ioctl( )** gets the controller's notion of the current partition table for disk drive.

DKIOCSAPART The argument is a pointer to a **dk_allmap** structure (described below). This **ioctl( )** sets the controller's notion of the partition table without changing the disk itself.

```
/*
 * Partition map (part of dk_label)
 */

struct dk_map {
        daddr_t dkl_cylno;     /* starting cylinder */
        daddr_t dkl_nblk;      /* number of blocks */
};

/*
 * Used for all partitions
 */

struct dk_allmap {
        struct dk_map     dka_map[NDKMAP];
};
```

DKIOCPARTINFO
x86: The argument is a pointer to a **part_info** structure (described below). This **ioctl( )** gets the driver's notion of the size and extent of the partition or slice indicated by the file descriptor argument.

```
/*
 * Used by applications to get partition or slice information
 */

struct part_info {
        daddr_t     p_start;
        int         p_length;
 };
```

DKIOCGGEOM The argument is a pointer to a **dk_geom** structure (described below). This **ioctl( )** gets the controller's notion of the current geometry of the disk drive.

DKIOCSGEOM The argument is a pointer to a **dk_geom** structure (described below). This **ioctl( )** sets the controller's notion of the geometry without changing the disk itself.

DKIOCG_PHYGEOM
x86: The argument is a pointer to a **dk_geom** structure (described below). This **ioctl( )** gets the driver's notion of the physical geometry of the disk drive. It is functionally identical to the **DKIOCGGEOM ioctl( )**.

DKIOCG_VIRTGEOM
x86: The argument is a pointer to a **dk_geom** structure (described below). This **ioctl( )** gets the controller's (and hence the driver's) notion of the virtual geometry of the disk drive. Virtual geometry is a view of the disk geometry maintained by the firmware in a host bus adapter or disk controller.

```
/*
 * Definition of a disk's geometry
 */
struct dk_geom {
        unsigned short   dkg_ncyl;               /* # of data */
                                                 /* cylinders */
        unsigned short   dkg_acyl;               /* # of alternate*/
                                                 /* cylinders */
        unsigned short   dkg_bcyl;               /* cyl offset (for */
                                                 /* fixed head area) */
        unsigned short   dkg_nhead;              /* # of heads */
        unsigned short   dkg_obs1;               /* obsolete */
        unsigned short   dkg_nsect;              /* # of sectors */
                                                 /* per track */
        unsigned short   dkg_intrlv;             /* interleave factor */
        unsigned short   dkg_obs2;               /* obsolete */
        unsigned short   dkg_obs3;               /* obsolete */
        unsigned short   dkg_apc;                /* alternates per*/
                                                 /* cyl (SCSI only) */
        unsigned short   dkg_rpm;                /* revolutions per min*/
        unsigned short   dkg_pcyl;               /* # of physical */
                                                 /* cylinders */
        unsigned short   dkg_write_reinstruct;   /* # sectors to */
                                                 /* skip, writes */
        unsigned short   dkg_read_reinstruct;    /* # sectors to */
                                                 /* skip, reads */
        unsigned short   dkg_extra[7];           /* for compatible*/
                                                 /* expansion */
};
#define dkg_gap1    dkg_extra[0]    /* for application */
                                    /* compatibility */
#define dkg_gap2    dkg_extra[1]    /* for application */
                                    /* compatibility */
```

DKIOCGVTOC The argument is a pointer to a **vtoc** structure (described below). This

ioctl( ) returns the device's current VTOC (volume table of contents).

DKIOCSVTOC The argument is a pointer to a **vtoc** structure (described below). This **ioctl( )** changes the VTOC associated with the device.

```
struct partition {
        ushort     p_tag;      /* ID tag of partition */
        ushort     p_flag;     /* permission flags */
        daddr_t    p_start;    /* start sector of partition */
        long       p_size;     /* # of blocks in partition */
};
```

If **DKIOCSVTOC** is used with a floppy diskette, the **p_start** field must be the first sector of a cylinder. Multiply the number of heads by the number of sectors per track to compute the number of sectors per cylinder.

```
struct vtoc {
    unsigned long    v_bootinfo[3];                  /* info needed */
                                                     /* by mboot */
                                                     /* (unsupported) */
    unsigned long    v_sanity;                       /* to verify vtoc */
                                                     /* sanity */
    unsigned long    v_version;                      /* layout version */
    char             v_volume[LEN_DKL_VVOL];         /* volume name */
    ushort           v_sectorsz;                     /* sector size in */
                                                     /* bytes */
    ushort           v_nparts;                       /* number of */
                                                     /* partitions */
    unsigned long    v_reserved[10];                 /* free space */
    struct partition v_part[V_NUMPAR];               /* partition */
                                                     /* headers*/
    time_t           timestamp[V_NUMPAR];            /* partition */
                                                     /* timestamp */
                                                     /* (unsupported) */
    char             v_asciilabel[LEN_DKL_ASCII];    /* compatibility */
};

/*
 * Partition permission flags
 */

#define V_UNMNT      0x01    /* Unmountable partition */
#define V_RONLY      0x10    /* Read only */

/*
 * Partition identification tags
 */

#define V_UNASSIGNED     0x00    /* unassigned partition */
#define V_BOOT           0x01    /* Boot partition */
#define V_ROOT           0x02    /* Root filesystem */
```

```
#define V_SWAP        0x03   /* Swap filesystem */
#define V_USR         0x04   /* Usr filesystem */
#define V_BACKUP      0x05   /* full disk */
#define V_STAND       0x06   /* Stand partition */
#define V_VAR         0x07   /* Var partition */
#define V_HOME        0x08   /* Home partition */
#define V_ALTSCTR     0x09   /* Alternate sector partition */
```

DKIOCADDBAD
x86: This **ioctl( )** forces the driver to re-examine the alternates slice and rebuild the internal bad block map accordingly. It should be used whenever the alternates slice is changed by any method other than the **addbadsec**(1M) utility.

DKIOCEJECT This **ioctl( )** requests the disk drive to eject its disk, if that drive supports removable media.

DKIOCLOCK SPARC: This **ioctl( )** requests the disk drive to lock the door, for those devices with removable media.

DKIOCUNLOCK
SPARC: This **ioctl( )** requests the disk drive to unlock the door, for those devices with removable media.

DKIOCSTATE SPARC: This **ioctl( )** blocks until the state of the drive, inserted or ejected, is changed. The argument is a pointer to a **dkio_state**, enum, whose possible enumerations are listed below. The initial value should be either the last reported state of the drive, or **DKIO_NONE**. Upon return, the enum pointed to by the argument is updated with the current state of the drive.

```
enum dkio_state {
        DKIO_NONE,        /* Return disk's current state */
        DKIO_EJECTED,     /* Disk state is 'ejected' */
        DKIO_INSERTED     /* Disk state is 'inserted' */
};
```

SEE ALSO

ioctl(2), **cdio**(7I), **fdio**(7I)

SPARC Only
hdio(7I), **ipi**(7D), **sd**(7D), **xd**(7D), **xy**(7D)

x86 Only
cmdk(7D)