Hardwarenahe Programmierung in C/C++

Disketten

Christian Zahler

Diskette

Technische Laufwerksdaten eines Diskettenlaufwerks

Daten eines Disketten-/Festplattenlaufwerks:

```
INT 21 - DOS 3.2+ - IOCTL - GENERIC BLOCK DEVICE REQUEST
      AX = 440Dh
      BL = drive number (00h=default,01h=A:,etc)
      CH = category code (see #1245)
      CL = minor code (function) (see #1246)
      DS:DX -> (DOS) parameter block (see #1247,#1249,#1250,#1251,#1252)
      SI:DI -> (OS/2 comp box) parameter block (see #1253,#1255,#1256,#1259)
Return: CF set on error
      AX = error code (01h,02h,etc.) (see #1366 at AH=59h/BX=0000h)
      CF clear if successful
      DS:DX -> data block if CL=60h or CL=61h
Notes: DOS 4.01 seems to ignore the high byte of the number of directory
      entries in the BPB for diskettes.
      functions 46h and 66h undocumented in DOS 4.x, documented for DOS 5+
      the DUBLDISK.SYS v2.6 driver only supports minor codes 60h and 67h
      DR DOS 3.41-6.0 only support minor codes 40h-42h and 60h-62h; all
      other minor codes return error code 16h
      some PCMCIA calls reportedly appear to be dangerous for MS-DOS versions
        prior to 5.0
      minor code 60h normally produces no I/O except with AutoMount=1 for
        DBLSPACE/DRVSPACE
SeeAlso: AX=440Ch,AX=440Dh/CX=084Ah,AX=440Dh/CX=0871h,AH=69h,INT 2F/AX=0802h
SeeAlso: INT 2F/AX=122Bh

(Table 1245) gibt CH-Register-Werte an, die gesetzt werden müssen
Values for block device IOCTL category code:
 08h  disk drive
 48h  FAT32 disk drive
 00h-7Fh reserved for Microsoft
 80h-FFh reserved for OEM/user-defined

(Table 1246) gibt CL-Register-Werte an, die gesetzt werden müssen
Values for generic block IOCTL minor code:
 00h  (OS/2)      \ used to lock/unlock a drive
 01h  (OS/2)      /
 40h  set device parameters (see #1247)
 41h  write logical device track (see #1249)
 42h  format and verify logical device track (see #1250)
 46h  (DOS 4.0+) set volume serial number (see #1252,AH=69h)
 47h  (DOS 4.0+) set access flag (see #1253)
 48h  (Enh. Disk Drive Spec) set media lock state (see #1254,INT 13/AH=45h)
 49h  (Enh. Disk Drive Spec) eject media in drive (see INT 13/AH=49h)
      no parameter block required
 4Ah  (MS-DOS 7.0) lock logical volume (see AX=440Dh/CX=084Ah)
 4Bh  (MS-DOS 7.0) lock physical volume (see AX=440Dh/CX=084Bh)
 50h  (PCMCIA) attribute memory write
 51h  (PCMCIA) common memory write
 52h  (PCMCIA) force media change (DOS 5+ ???) (see #1255)
 53h  (PCMCIA) erase drive
 54h  (PCMCIA) erase media
 56h  (PCMCIA) set erase status callback
 57h  (PCMCIA) append Card Information Structure (CIS) tuple
 58h  (PCMCIA) erase CIS tuples
 60h  get device parameters (see #1247)
 61h  read logical device track (see #1249)
 62h  verify logical device track (see #1251)
 66h  (DOS 4.0+) get volume serial number (see #1252,AH=69h)
 67h  (DOS 4.0+) get access flag (see #1253)
 68h  (DOS 5.0+) sense media type (see #1256)
 6Ah  (MS-DOS 7.0) unlock logical volume (see AX=440Dh/CX=086Ah)
      no parameter block required
 6Bh  (MS-DOS 7.0) unlock physical volume (see AX=440Dh/CX=086Bh)
      no parameter block required
 6Ch  (MS-DOS 7.0) get lock flag (see AX=440Dh/CX=086Ch)
      no parameter block required
 6Dh  (MS-DOS 7.0) enumerate open files (see AX=440Dh/CX=086Dh)
 6Eh  (MS-DOS 7.0) find swap file (see AX=440Dh/CX=086Eh)
 6Fh  (MS-DOS 7.0) get drive map information (see #1257)
 70h  (PCMCIA) attribute memory read
 70h  (MS-DOS 7.0) get current lock state (see AX=440Dh/CX=0870h)
      no parameter block required
 71h  (MS-DOS 7.0) get first cluster (see AX=440Dh/CX=0871h)
 73h  (PCMCIA) get memory media information (DOS 5+ ???) (see #1259)
 76h  (PCMCIA) get erase status callback
 77h  (PCMCIA) get first Card Information Structure (CIS) tuple
 78h  (PCMCIA) get next CIS tuple
 7Fh  (PCMCIA) get ??? information (see #1260,#1261)

Format of parameter block for functions 40h, 60h:
Offset Size       Description (Table 1247)
 00h  BYTE        special functions
                  bit 0 set if function to use current BPB, clear if Device
                              BIOS Parameter Block field contains new default
BPB
                  bit 1 set if function to use track layout fields only
                            must be clear if CL=60h
                  bit 2 set if all sectors in track same size (should be set)
                  bits 3-7 reserved (MS-DOS, Novell DOS 7)
                  bit 5: skip head settling time (WinDOS 2.11)
                  bit 6: format access flag (WinDOS 2.11)
 01h  BYTE        device type (see #1248)
 02h  WORD        device attributes
                  bit 0 set if nonremovable medium
                  bit 1 set if door lock ("changeline") supported
                  bits 2-15 reserved
 04h  WORD        number of cylinders
 06h  BYTE        media type
                  for 1.2M drive
                      00h 1.2M disk (default)
                      01h 320K/360K disk
                  F8h for DUBLDISK.SYS v2.6 expanded drives
                  always 00h for other drive types
 07h 31 BYTEs     device BPB (see #1349 at AH=53h), bytes after BPB offset 1Eh
                    omitted; final six bytes only transferred on function 40h
                    with BYTE 00h bit 0 set for MS-DOS 5.0
--function 40h only--
 26h  WORD        number of sectors per track (start of track layout field)
                  (maximum 63)
 28h  N word pairs: number,size of each sector in track
--category code 48h (FAT32), function 40h--
 07h 53 BYTEs     extended BPB (see #1350)
 3Ch 32 BYTEs     reserved
 5Ch  WORD        number of track table entries
 5Eh 2N WORDs     sector table (word pairs: number/size of each sector in track)
--category code 48h (FAT32), function 60h--
 07h 53 BYTEs     extended BPB (see #1350)
 3Ch 32 BYTEs     reserved

(Table 1248)
Values for device type:
 00h  320K/360K disk
 01h  1.2M disk
 02h  720K disk
 03h  single-density 8-inch disk
 04h  double-density 8-inch disk
 05h  fixed disk
 06h  tape drive
 07h  (DOS 3.3+) other type of block device, normally 1.44M floppy
 08h  read/write optical disk
 09h  (DOS 5+) 2.88M floppy

Format of parameter block for functions 41h, 61h:
Offset Size       Description (Table 1249)
 00h  BYTE        special functions (reserved, must be zero)
 01h  WORD        number of disk head
 03h  WORD        number of disk cylinder
 05h  WORD        number of first sector to read/write
 07h  WORD        number of sectors
```

```
 09h  DWORD       transfer address
Note: under Windows95, a volume must be locked (see AX=440Dh/CX=084Bh) in
        order to perform direct accesses such as track reads and writes
        with this IOCTL function

Format of parameter block for function 42h:
Offset Size       Description (Table 1250)
 00h  BYTE        reserved, must be zero (DOS <3.2)
                    bit 0=0: format/verify track
                             1: format status call (DOS 3.2+), don't actually
format
                    bit 1: format multiple tracks, require additional WORD
                             (hard disks only)
                    bits 2-7 reserved, must be zero
                  value on return (DOS 3.3+):
                    00h      specified tracks, sectors/track supported by BIOS
                    01h      function not supported by BIOS
                    02h      specified tracks, sectors/track not allowed for
drive
                    03h      no disk in drive
 01h  WORD        number of disk head
 03h  WORD        number of disk cylinder
--BYTE 00h bit 1 set--
 05h  WORD        number of tracks to format

Format of parameter block for function 62h:
Offset Size       Description (Table 1251)
 00h  BYTE        reserved, must be zero (DOS <3.2)
                    bit 0=0: verify single track
                             1: verify multiple tracks
                    bits 1-7 reserved, must be zero
                  value on return (DOS 3.3+):
                    00h      specified tracks, sectors/track supported by BIOS
                    01h      function not supported by BIOS
                    02h      specified tracks, sectors/track not allowed for
drive
                    03h      no disk in drive
 01h  WORD        number of disk head
 03h  WORD        number of disk cylinder
 05h  WORD        number of tracks to verify (equivalent to 255 or fewer sectors)

Format of parameter block for functions 46h, 66h:
Offset Size       Description (Table 1252)
 00h  WORD        (call) info level (should be 0000h)
 02h  DWORD       disk serial number (binary)
 06h 11 BYTEs     volume label or "NO NAME    "
 11h  8 BYTEs     filesystem type "FAT12   " or "FAT16   "
                    (generally CL=66h only, but MS-DOS 5.0 will write the
                    given filesystem type to the disk)
Note: under MS-DOS 7.0 or a Windows95 DOS box, the volume label field can
        return as all blanks even when a volume label has been set (the Win95
        installation seems to blank the volume label field in the partition
        boot sector; once LABEL has been run, the volume label is reported
        correctly)
SeeAlso: AH=69h

Format of parameter block for functions 47h, 67h:
Offset Size       Description (Table 1253)
 00h  BYTE        special-function field (must be zero)
 01h  BYTE        disk-access flag, nonzero if access allowed by driver

Format of parameter block for function 48h:
Offset Size       Description (Table 1254)
 00h  BYTE        (call) locking operation
                  00h lock media in drive
                  01h unlock media
                  02h get locking status
 01h  BYTE        (ret) drive's lock status (number of pending locks on drive)
Note: also supported by MS-DOS 7.0

Format of parameter block for function 52h:
 00h  BYTE        (call) unused???        (Table 1255)
                  (ret) 00h if flash/ATA drive but no card inserted
                        unchanged otherwise
Notes: the absence of a flash card should be tested by checking the DOS error
        code rather than the returned byte
       the parameter byte is cleared to 00h erroneously by the Award
        PCDISK.EXE v1.02c PCMCIA/ATA driver if no ATA card is inserted
        (bug corrected in PCDISK.EXE v1.02h and later)
       not supported by the SystemSoft ATADRV.EXE and the Phoenix PCMATA.SYS
        PCMCIA/ATA drivers

Format of parameter block for function 68h:
Offset Size       Description (Table 1256)
 00h  BYTE        01h for default media type, 00h for any other media type
                  (see also INT 13/AH=20h"Compaq")
 01h  BYTE        02h for 720K, 07h for 1.44M, 09h for 2.88M

Format of parameter block for function 6Fh:
Offset Size       Description (Table 1257)
 00h  BYTE        (call) length of this buffer (in bytes)
 01h  BYTE        (ret) number of bytes in parameter block actually used
 02h  BYTE        (ret) drive flags (see #1258)
 03h  BYTE        (ret) physical drive number
                  00h-7Fh floppy
                  80h-FEh hard
                  FFh no physical drive
 04h  DWORD       (ret) bitmap of logical drives associated with physical drive
                  bit 0 = drive A:, etc.
 08h  QWORD       (ret) relative block address of partition start

Bitfields for Get Drive Map Information drive flags:
Bit(s) Description (Table 1258)
 0    protected-mode driver for logical drive
 1    protected-mode driver in use for physical drive corresponding to the
        logical drive
 2    drive available only in protected mode
 3    protected-mode drive supports media ejection
 4    drive issues media insertion and removal notifications
SeeAlso: #1257

Format of parameter block for function 73h:
Offset Size       Description (Table 1259)
 00h  BYTE        ???
                  00h ATA card inserted ???
                  80h ATA card not inserted ???
 01h  BYTE        length of parameter block ???
                  apparently always 40h
 02h  BYTE        ???
                  00h ATA card not inserted ???
                  0Dh ATA card inserted ???
 03h  2 BYTEs     ??? (apparently always 00h)
 05h  BYTE        drive number (0=first) ???
 06h  BYTE        total number of drives ???
 07h  BYTE        ???
                  00h ATA card not inserted ???
                  01h ATA card inserted ???
 08h 17 BYTEs     ???
 19h  BYTE        ???
                  00h ATA card not inserted ???
                  01h ATA card inserted ???
 1Ah  BYTE        ??? (apparently always 01h)
 1Bh  BYTE        ???
                  00h ATA card not inserted ???
                  01h ATA card inserted ???
 1Ch  2 BYTEs     ??? (apparently always 0015h)
 1Eh  2 BYTEs     ???
 20h  2 BYTEs     ??? (apparently always 0110h)
 22h 15 BYTEs     ???
 31h  2 BYTEs     ??? (apparently always 7000h)
 33h 11 BYTEs     driver signature
                  "AWARD PDISK" for Award PCDISK.EXE PCMCIA/ATA driver
                  "MS-BIOS    " for HP 200LX generic ATA driver
 3Eh  2 BYTEs     ???
Notes: parameter structure possibly depends on driver
       this function is not supported by the SystemSoft ATADRV.EXE and the
        Phoenix PCMATA.SYS PCMCIA/ATA drivers

Format of parameter block for function 7Fh for SystemSoft ATADRV.EXE:
Offset Size       Description (Table 1260)
 00h  DWORD       -> unknown location within driver
Note: function supported by the SystemSoft ATADRV.EXE PCMCIA/ATA driver
        but not by the Award PCDISK.EXE PCMCIA/ATA driver
SeeAlso: #1261

Format of parameter block for function 7Fh for Phoenix PCMATA.SYS:
Offset Size       Description (Table 1261)
 00h  8 BYTEs     ???
Note: this function supported by the Phoenix PCMATA.SYS PCMCIA/ATA driver
        but not by the Award PCDISK.EXE PCMCIA/ATA driver
SeeAlso: #1260
```

```
Format of BIOS Parameter Block:
Offset Size       Description (Table 1349)
 00h   WORD       number of bytes per sector
 02h   BYTE       number of sectors per cluster
 03h   WORD       number of reserved sectors at start of disk
 05h   BYTE       number of FATs
 06h   WORD       number of entries in root directory
 08h   WORD       total number of sectors
                  for DOS 4.0+, set to zero if partition >32M, then set DWORD at
                    15h to actual number of sectors
 0Ah   BYTE       media ID byte (see #1044)
 0Bh   WORD       number of sectors per FAT
--DOS 2.13--
 0Dh   WORD       number of sectors per track
 0Fh   WORD       number of heads
 11h   WORD       number of hidden sectors
--DOS 3.0+ --
 0Dh   WORD       number of sectors per track
 0Fh   WORD       number of heads
 11h   DWORD      number of hidden sectors
 15h 11 BYTEs     reserved
--DOS 4.0+ --
 15h   DWORD      total number of sectors if word at 08h contains zero
 19h  6 BYTEs     ???
 1Fh   WORD       number of cylinders
 21h   BYTE       device type
 22h   WORD       device attributes (removable or not, etc)
--DR DOS 5+ --
 15h   DWORD      total number of sectors if word at 08h contains zero
 19h  6 BYTEs     reserved
--European MS-DOS 4.00--
 15h   DWORD      total number of sectors if word at 08h contains zero
                  (however, this DOS does not actually implement >32M partitions)
SeeAlso: #1083,#1350
```

```
(Table 1044)
Values for media ID byte:
 FFh   floppy, double-sided, 8 sectors per track (320K)
 FEh   floppy, single-sided, 8 sectors per track (160K)
 FDh   floppy, double-sided, 9 sectors per track (360K)
 FCh   floppy, single-sided, 9 sectors per track (180K)
 FAh   HP 200LX D: ROM disk, 16 sectors per track (995K)
       HP 200LX E: (Stacker host drive ???)
 F9h   floppy, double-sided, 15 sectors per track (1.2M)
       floppy, double-sided, 9 sectors per track (720K,3.5")
 F8h   hard disk
 F0h   other media
       (e.g. floppy, double-sized, 18 sectors per track - 1.44M,3.5")
```

```
/* Programm        Diskette
   Version         1.0
*/

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <process.h>

int Drive;
int Fehler;
int AnzBlock;
int FreiBlock;
int SektBlock;
unsigned long Sektoren;

struct PARAMS
{
  unsigned char spezfkt;
  unsigned char devtype;
  unsigned int devattr;
  unsigned int numcyl;
  unsigned char mediatype;
  unsigned int bytesekt;
  unsigned char sktclust;
  unsigned int ressekt;
  unsigned char numfats;
  unsigned int rootentr;
  unsigned int totalsekt;
  unsigned char mediades;
  unsigned int sektfat;
  unsigned int sectspur;
  unsigned int heads;
  unsigned long hiddsekt;
  unsigned long res1;
};

REGS register_in, register_out, reg;

void Diskinfo(int lw)
{
  register_in.x.dx=lw;
  register in.h.ah=0x36;
  int86(0x21,&register_in,&register_out);
  SektBlock = register out.x.ax;
  FreiBlock = register out.x.bx;
  AnzBlock = register out.x.dx;
}

void Version(char *vers)
{
  int haupt, neben;
  register_in.h.ah=0x30;
  int86(0x21,&register in,&register out);
  haupt = register_out.h.al;
  neben = register out.h.ah;
  sprintf(vers,"%d.%d",haupt,neben);
} // End Version


int Parameter(int lw, void *dev)
{
  reg.x.ax=0x440D;
  reg.x.bx=lw;
  reg.x.cx=0x0860;
  reg.x.dx=(int)(dev);
  intdos(&reg,&reg);

  return reg.x.ax;
}

void main()
{
  struct PARAMS device;
  int laufwerk;
  char *ver = "";
  clrscr();
  puts("\t\tProgramm DISKETTEN-INFO");
  puts("\tLiest die technischen Laufwerksdaten.");

  Version(ver); // Funktion holt die DOS-Version
  printf("\nMS-DOS Version %s installiert.",ver);

  printf("\nWelches Laufwerk: *\b");
  laufwerk = getch();
  Drive = (laufwerk & 0x5f) - 0x40;
  if ((Drive < 1) || (Drive > 26))
    Drive = 0;
  putch(Drive+0x40);
  Fehler = Parameter(Drive,&device);
  if (device.mediades==0xf8)
    printf("\nFestplattenlaufwerk");
  else
    printf("\nDiskettenlaufwerk");
  Diskinfo(Drive);
  if (device.totalsekt == 0)
    Sektoren = device.res1;
else
    Sektoren=device.totalsekt;

  printf("\nAnzahl der Sektoren     : %lu",Sektoren);
  printf("\nSektoren pro Spur       : %u",device.sectspur);
  printf("\nReservierte Sektoren    : %u",device.ressekt);
  printf("\nVersteckte Sektoren     : %u",device.hiddsekt);
  printf("\nBytes pro Sektor        : %u",device.bytesekt);
  printf("\nAnzahl Spuren           : %u",device.numcyl);
  printf("\nAnzahl Köpfe            : %u",device.heads);
  printf("\nAnzahl Cluster          : %u",AnzBlock);
  printf("\nFreie Cluster           : %u",FreiBlock);
  printf("\nSektoren pro Cluster    : %u",SektBlock);
  printf("\nGesamt-Speicherplatz    : %lu",Sektoren*device.bytesekt);
  getch();
} // End main
```