NAME

fdc - PC architecture floppy disk controller driver

SYNOPSIS

device fdc

In */boot/device.hints*:
hint.fdc.0.at="isa"
hint.fdc.0.port="0x3F0"
hint.fdc.0.irq="6"
hint.fdc.0.drq="2"
hint.fdc.0.flags="0x0"
hint.fd.0.at="fdc0"
hint.fd.0.drive="0"
hint.fd.0.flags="0x0"
hint.fd.1.at="fdc0"
hint.fd.1.drive="1"
hint.fd.1.flags="0x0"

DESCRIPTION

Device Usage

This driver provides access to floppy disk drives. Floppy disks using either FM (single-density) or MFM (double or high-density) recording can be handled.

Floppy disk controllers can connect up to four drives each. The **fdc** driver can currently handle up to two drives per controller (or four drives on ACPI). Upon driver initialization, an attempt is made to find out the type of the floppy controller in use. The known controller types are either the original NE765 or i8272 chips, or alternatively *enhanced* controllers that are compatible with the NE72065 or i82077 chips. These enhanced controllers (among other enhancements) implement a FIFO for floppy data transfers that will automatically be enabled once an enhanced chip has been detected. This FIFO activation can be disabled using the per-controller flags value of *0x1*.

By default, this driver creates a single device node */dev/fdN* for each attached drive with number *N*. For historical reasons, device nodes that use a trailing UFS-style partition letter (ranging from ‘a’ through ‘h’) can also be accessed, which will be implemented as symbolic links to the main device node.

Accessing the main device node will attempt to autodetect the density of the available medium for multi-density devices. Thus it is possible to use either a 720 KB medium or a 1440 KB medium in a high-density 3.5 inch standard floppy drive. Normally, this autodetection will only happen once at the first call to open(2) for the device after inserting the medium. This assumes the drive offers proper

changeline support so media changes can be detected by the driver. To indicate a drive that does not have the changeline support, this can be overridden using the per-drive device flags value of *0x10* (causing each call to open(2) to perform the autodetection).

When trying to use a floppy device with special-density media, other device nodes can be created, of the form */dev/fdN.MMMM*, where *N* is the drive number, and *MMMM* is a number between one and four digits describing the device density. Up to 15 additional subdevices per drive can be created that way. The administrator is free to decide on a policy how to assign these numbers. The two common policies are to either implement subdevices numbered 1 through 15, or to use a number that describes the medium density in kilobytes. Initially, each of those devices will be configured to the maximal density that is possible for the drive type (like 1200 KB for 5.25 inch HD drives or 1440 KB for 3.5 inch HD drives). The desired density to be used on that subdevice needs to be configured using fdcontrol(8).

Drive types are configured using the lower four bits of the per-drive device flags. The following values can be specified:

1 5.25 inch double-density device with 40 cylinders (360 KB native capacity)

2 5.25 inch high-density device with 80 cylinders (1200 KB native capacity)

3 3.5 inch double-density device with 80 cylinders (720 KB native capacity)

4 3.5 inch high-density device with 80 cylinders (1440 KB native capacity)

5 3.5 inch extra-density device with 80 cylinders (2880 KB native capacity, usage currently restricted to at most 1440 KB media)

6 Same as type 5, available for compatibility with some BIOSes

On IA32 architectures, the drive type can be specified as 0 for the drives. In that case, the CMOS configuration memory will be consulted to obtain the value for that drive. The ACPI probe automatically determines these values via the _FDE and _FDI methods, but this can be overridden by specifying a drive type hint.

Normally, each configured drive will be probed at initialization time, using a short seek sequence. This is intended to find out about drives that have been configured but are actually missing or otherwise not responding. (The ACPI probe method does not perform this seek.) In some environments (like laptops with detachable drives), it might be desirable to bypass this drive probe, and pretend a drive to be there so the driver autoconfiguration will work even if the drive is currently not present. For that purpose, a per-drive device flags value of *0x20* needs to be specified.

Programming Interface

In addition to the normal read and write functionality, the **fdc** driver offers a number of configurable options using ioctl(2). In order to access any of this functionality, programmers need to include the header file <*sys/fdcio.h*> into their programs. The call to open(2) can be performed in two possible ways. When opening the device without the O_NONBLOCK flag set, the device is opened in a normal way, which would cause the main device nodes to perform automatic media density selection, and which will yield a file descriptor that is fully available for any I/O operation or any of the following ioctl(2) commands.

When opening the device with O_NONBLOCK set, automatic media density selection will be bypassed, and the device remains in a half-opened state. No actual I/O operations are possible, but many of the ioctl(2) commands described below can be performed. This mode is intended for access to the device without the requirement to have an accessible media present, like for status inquiries to the drive, or in order to format a medium. O_NONBLOCK needs to be cleared before I/O operations are possible on the descriptor, which requires a prior specification of the density using the FD_STYPE command (see below). Operations that are not allowed on the half-opened descriptor will cause an error value of EAGAIN.

The following ioctl(2) commands are currently available:

FD_FORM Used to format a floppy disk medium. Third argument is a pointer to a *struct fd_formb* specifying which track to format, and which parameters to fill into the ID fields of the floppy disk medium.

FD_GTYPE Returns the current density definition record for the selected device. Third argument is a pointer to *struct fd_type*.

FD_STYPE Adjusts the density definition of the selected device. Third argument is a pointer to *struct fd_type*. For the fixed-density subdevices (1 through 15 per drive), this operation is restricted to a process with superuser privileges. For the auto-selecting subdevice 0, the operation is temporarily allowed to any process, but this setting will be lost again upon the next autoselection. This can be used when formatting a new medium (which will require to open the device using O_NONBLOCK, and thus to later adjust the density using FD_STYPE).

FD_GOPTS Obtain the current drive options. Third argument is a pointer to *int*, containing a bitwise union of the following possible flag values:

FDOPT_NORETRY Do not automatically retry operations upon failure.

FDOPT_NOERRLOG Do not cause "hard error" kernel logs for failed I/O operations.

FDOPT_NOERROR Do not indicate I/O errors when returning from read(2) or write(2) system calls. The caller is assumed to use FD_GSTAT calls in order to inquire about the success of each operation. This is intended to allow even erroneous data from bad blocks to be retrieved using normal I/O operations.

FDOPT_AUTOSEL Device performs automatic density selection. Unlike the above flags, this one is read-only.

FD_SOPTS Set device options, see above for their meaning. Third argument is a pointer to *int*. Drive options will always be cleared when closing the descriptor.

FD_CLRERR
Clear the internal low-level error counter. Normally, controller-level I/O errors are only logged up to FDC_ERRMAX errors (currently defined to 100). This command resets the counter. Requires superuser privileges.

FD_READID Read one sector ID field from the floppy disk medium. Third argument is a pointer to *struct fdc_readid*, where the read data will be returned. Can be used to analyze a floppy disk medium.

FD_GSTAT Return the recent floppy disk controller status, if available. Third argument is a pointer to *struct fdc_status*, where the status registers (ST0, ST1, ST2, C, H, R, and N) are being returned. EINVAL will be caused if no recent status is available.

FD_GDTYPE
Returns the floppy disk drive type. Third argument is a pointer to *enum fd_drivetype*. This type is the same as being used in the per-drive configuration flags, or in the CMOS configuration data or ACPI namespace on IA32 systems.

FILES

*/dev/fd** floppy disk device nodes

SEE ALSO

fdread(1), fdwrite(1), ioctl(2), open(2), read(2), write(2), fdcontrol(8), fdformat(8)

AUTHORS

This man page was initially written by Wilko Bulte, and later vastly rewritten by Jörg Wunsch.