NAME

ipmi - OpenIPMI compatible IPMI interface driver

SYNOPSIS

device ipmi

To manually specify I/O attachment in */boot/device.hints*:
hint.ipmi.0.at="isa"
hint.ipmi.0.port="0xCA2"
hint.ipmi.0.spacing="8"
hint.ipmi.0.mode="KCS"

To manually specify memory attachment in */boot/device.hints*:
hint.ipmi.0.at="isa"
hint.ipmi.0.maddr="0xf0000000"
hint.ipmi.0.spacing="8"
hint.ipmi.0.mode="SMIC"

Meaning of *spacing*:

8	8 bit alignment
16	16 bit alignment
32	32 bit alignment

If the *port* and *spacing* are not specified the interface type default will be used. Only specify either the *port* for I/O access or *maddr* for memory access.

DESCRIPTION

The IPMI (Intelligent Platform Management Interface) is a standard for monitoring system hardware by permitting generic code to detect and monitor the sensors in a system. The IPMI standard offers watchdog support, an FRU database, and other support extensions. It is currently being adopted by the makers of many single board and embedded system manufacturers.

The **ipmi** driver in FreeBSD is heavily adopted from the standard and Linux driver; however, not all features described in the standard are supported.

The **ipmi** driver implements the power cycling option to shutdown(8) to implement power cycling of the system. The motherboard’s BMC must support the chassis device and the optional power cycle subcomand of the chassis control command as described in section 28.3 of the IPMI standard. The length of time the system is off will be at least one second, but may be longer if the power cycle interval has been set (see section 28.9).

IOCTLS

Sending and receiving messages through the **ipmi** driver requires the use of ioctl(2). The ioctls are used due to the complexity of data sent to and from the device. The ioctl(2) command codes below are defined in <*sys/ipmi.h*>. The third argument to ioctl(2) should be a pointer to the type indicated.

Currently the following ioctls are supported:

IPMICTL_RECEIVE_MSG (*struct ipmi_recv*)
Receive a message. Possible error values:

[EAGAIN] No messages are in the process queue.

[EFAULT] An address supplied was invalid.

[EMSGSIZE] The address could not fit in the message buffer and will remain in the buffer.

IPMICTL_RECEIVE_MSG_TRUNC (*struct ipmi_recv*)
Like IPMICTL_RECEIVE_MSG but if the message cannot fit into the buffer, it will truncate the contents instead of leaving the data in the buffer.

IPMICTL_SEND_COMMAND (*struct ipmi_req*)
Send a message to the interface. Possible error values:

[EFAULT] An address supplied was invalid.

[ENOMEM] Buffers could not be allowed for the command, out of memory.

IPMICTL_SET_MY_ADDRESS_CMD (*unsigned int*)
Set the slave address for source messages.

IPMICTL_GET_MY_ADDRESS_CMD (*unsigned int*)
Get the slave address for source messages.

IPMICTL_SET_MY_LUN_CMD (*unsigned int*)
Set the slave LUN for source messages.

IPMICTL_GET_MY_LUN_CMD (*unsigned int*)
Get the slave LUN for source messages.

Unimplemented Ioctls

IPMICTL_REGISTER_FOR_CMD (*struct ipmi_cmdspec*)
Register to receive a specific command. Possible error values:

[EFAULT] An supplied address was invalid.

[EBUSY] The network function/command is already in use.

[ENOMEM] Could not allocate memory.

IPMICTL_UNREGISTER_FOR_CMD (*struct ipmi_cmdspec*)
Unregister to receive a specific command. Possible error values:

[EFAULT] An address supplied was invalid.

[ENOENT] The network function/command was not found.

Stub Only Ioctl

IPMICTL_SET_GETS_EVENTS_CMD (*int*)
Set whether this interface receives events. Possible error values:

[EFAULT] An address supplied was invalid.

SEE ALSO

ioctl(2), watchdog(4), reboot(8), shutdown(8), watchdog(8), watchdogd(8), watchdog(9)

HISTORY

The **ipmi** driver first appeared in FreeBSD 6.2.

AUTHORS

The **ipmi** driver was written by Doug Ambrisko <*ambrisko@FreeBSD.org*>. This manual page was written by Tom Rhodes <*trhodes@FreeBSD.org*>.

BUGS

Not all features of the MontaVista driver are supported.

Currently, IPMB and BT modes are not implemented.