NAME

ahci - Serial ATA Advanced Host Controller Interface driver

SYNOPSIS

To compile this driver into the kernel, place the following lines in your kernel configuration file:

```
device pci
device scbus
device ahci
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
ahci_load="YES"
```

The following tunables are settable from the loader(8):

hint.ahci.X.msi
controls Message Signaled Interrupts (MSI) usage by the specified controller.

- 0 MSI disabled;
- 1 single MSI vector used, if supported;
- 2 multiple MSI vectors used, if supported (default);

hint.ahci.X.ccc
controls Command Completion Coalescing (CCC) usage by the specified controller. Non-zero value enables CCC and defines maximum time (in ms), request can wait for interrupt, if there are some more requests present on controller queue. CCC reduces number of context switches on systems with many parallel requests, but it can decrease disk performance on some workloads due to additional command latency.

hint.ahci.X.direct
controls whether the driver should use direct command completion from interrupt thread(s), or queue them to CAM completion threads. Default value depends on number of MSI interrupts supported and number of implemented SATA ports.

hint.ahci.X.em
controls whether the driver should implement virtual enclosure management device on top of SGPIO or other interface. Default value depends on controller capabilities.

hint.ahcich.X.pm_level

controls SATA interface Power Management for the specified channel, allowing some power to be saved at the cost of additional command latency. Possible values:

0 interface Power Management is disabled (default);
1 device is allowed to initiate PM state change, host is passive;
2 host initiates PARTIAL PM state transition every time port becomes idle;
3 host initiates SLUMBER PM state transition every time port becomes idle.
4 driver initiates PARTIAL PM state transition 1ms after port becomes idle;
5 driver initiates SLUMBER PM state transition 125ms after port becomes idle.

Some controllers, such as ICH8, do not implement modes 2 and 3 with NCQ used. Because of artificial entering latency, performance degradation in modes 4 and 5 is much smaller then in modes 2 and 3.

Note that interface Power Management complicates device presence detection. A manual bus reset/rescan may be needed after device hot-plug, unless hardware implements Cold Presence Detection.

hint.ahcich.X.sata_rev
setting to nonzero value limits maximum SATA revision (speed). Values 1, 2 and 3 are respectively 1.5, 3 and 6Gbps.

hw.ahci.force
setting to nonzero value forces driver attach to some known AHCI-capable chips even if they are configured for legacy IDE emulation. Default is 1.

DESCRIPTION

This driver provides the CAM(4) subsystem with native access to the SATA ports of AHCI-compatible controllers. Each SATA port found is represented to CAM as a separate bus with one target, or, if HBA supports Port Multipliers, 16 targets. Most of the bus-management details are handled by the SATA-specific transport of CAM. Connected ATA disks are handled by the ATA protocol disk peripheral driver ada(4). ATAPI devices are handled by the SCSI protocol peripheral drivers cd(4), da(4), sa(4), etc.

Driver features include support for Serial ATA and ATAPI devices, Port Multipliers (including FIS-based switching, when supported), hardware command queues (up to 32 commands per port), Native Command Queuing, SATA interface Power Management, device hot-plug and Message Signaled Interrupts.

Driver supports "LED" enclosure management messages, defined by the AHCI. When supported by hardware, it allows to control per-port activity, locate and fault LEDs via the led(4) API or emulated ses(4) device for localization and status reporting purposes. Supporting AHCI controllers may transmit

that information to the backplane controllers via SGPIO interface. Backplane controllers interpret received statuses in some way (IBPI standard) to report them using present indicators.

HARDWARE

The **ahci** driver supports AHCI compatible controllers having PCI class 1 (mass storage), subclass 6 (SATA) and programming interface 1 (AHCI).

Also, in cooperation with atamarvell and atajmicron drivers of ata(4), it supports AHCI part of legacy-PATA + AHCI-SATA combined controllers, such as JMicron JMB36x and Marvell 88SE61xx.

The **ahci** driver also supports AHCI devices that act as PCI bridges for nvme(4) using Intel Rapid Storage Technology (RST). To use the nvme(4) device, either one must set the SATA mode in the BIOS to AHCI (from RST), or one must accept the performance with RST enabled due to interrupt sharing. FreeBSD will automatically detect AHCI devices with this extension that are in RST mode. When that happens, **ahci** will attach nvme(4) children to the **ahci** device.

FILES

/dev/led/ahci.*.act* activity LED device nodes

/dev/led/ahci.*.fault* fault LED device nodes

/dev/led/ahci.*.locate* locate LED device nodes

SYSCTL

dev.ahcich.X.disable_phy

Set to 1 to disable the phy for the drive on channel X. Set to 0 to enable the phy. Useful for turning off troublemakers. Also useful for debugging when you need the ada drive to come and go.

SEE ALSO

ada(4), ata(4), cam(4), cd(4), da(4), sa(4), ses(4)

HISTORY

The **ahci** driver first appeared in FreeBSD 8.0.

AUTHORS

Alexander Motin <*mav@FreeBSD.org*>