This driver is for Compaq's SMART Array Controllers.

Supported Cards:

This driver is known to work with the following cards:

* SA 5300
* SA 5i
* SA 532
* SA 5312
* SA 641
* SA 642
* SA 6400
* SA 6400 U320 Expansion Module
* SA 6i
* SA P600
* SA P800
* SA E400
* SA P400i
* SA E200
* SA E200i
* SA E500
* SA P700m
* SA P212
* SA P410
* SA P410i
* SA P411
* SA P812
* SA P712m
* SA P711m

Detecting drive failures:

To get the status of logical volumes and to detect physical drive
failures, you can use the cciss_vol_status program found here:
http://cciss.sourceforge.net/#cciss_utils

Device Naming:

If nodes are not already created in the /dev/cciss directory, run as root:

```
# cd /dev
# ./MAKEDEV cciss
```

You need some entries in /dev for the cciss device. The MAKEDEV script
can make device nodes for you automatically. Currently the device setup
is as follows:

```
Major numbers:
        104     cciss0
        105     cciss1
        106     cciss2
        105     cciss3
        108     cciss4
        109     cciss5
        110     cciss6
        111     cciss7
```

Minor numbers:

```
        b7 b6 b5 b4 b3 b2 b1 b0
        |----+----| |----+----|
             |           |
             |           +-------- Partition ID (0=wholedev, 1-15 partition)
             |
             +-------------------- Logical Volume number
```

The device naming scheme is:

```
/dev/cciss/c0d0                 Controller 0, disk 0, whole device
/dev/cciss/c0d0p1               Controller 0, disk 0, partition 1
/dev/cciss/c0d0p2               Controller 0, disk 0, partition 2
/dev/cciss/c0d0p3               Controller 0, disk 0, partition 3

/dev/cciss/c1d1                 Controller 1, disk 1, whole device
/dev/cciss/c1d1p1               Controller 1, disk 1, partition 1
/dev/cciss/c1d1p2               Controller 1, disk 1, partition 2
/dev/cciss/c1d1p3               Controller 1, disk 1, partition 3
```

SCSI tape drive and medium changer support

SCSI sequential access devices and medium changer devices are supported and appropriate device nodes are automatically created. (e.g. /dev/st0, /dev/st1, etc. See the "st" man page for more details.) You must enable "SCSI tape drive support for Smart Array 5xxx" and "SCSI support" in your kernel configuration to be able to use SCSI tape drives with your Smart Array 5xxx controller.

Additionally, note that the driver will not engage the SCSI core at init time. The driver must be directed to dynamically engage the SCSI core via the /proc filesystem entry which the "block" side of the driver creates as /proc/driver/cciss/cciss* at runtime. This is because at driver init time, the SCSI core may not yet be initialized (because the driver is a block driver) and attempting to register it with the SCSI core in such a case would cause a hang. This is best done via an initialization script (typically in /etc/init.d, but could vary depending on distribution). For example:

```
        for x in /proc/driver/cciss/cciss[0-9]*
        do
                echo "engage scsi" > $x
        done
```

Once the SCSI core is engaged by the driver, it cannot be disengaged (except by unloading the driver, if it happens to be linked as a module.)

Note also that if no sequential access devices or medium changers are detected, the SCSI core will not be engaged by the action of the above script.

Hot plug support for SCSI tape drives

Hot plugging of SCSI tape drives is supported, with some caveats.
The cciss driver must be informed that changes to the SCSI bus
have been made. This may be done via the /proc filesystem.
For example:

```
        echo "rescan" > /proc/scsi/cciss0/1
```

This causes the driver to query the adapter about changes to the
physical SCSI buses and/or fibre channel arbitrated loop and the
driver to make note of any new or removed sequential access devices
or medium changers. The driver will output messages indicating what
devices have been added or removed and the controller, bus, target and
lun used to address the device. It then notifies the SCSI mid layer
of these changes.

Note that the naming convention of the /proc filesystem entries
contains a number in addition to the driver name. (E.g. "cciss0"
instead of just "cciss" which you might expect.)

Note: ONLY sequential access devices and medium changers are presented
as SCSI devices to the SCSI mid layer by the cciss driver. Specifically,
physical SCSI disk drives are NOT presented to the SCSI mid layer. The
physical SCSI disk drives are controlled directly by the array controller
hardware and it is important to prevent the kernel from attempting to directly
access these devices too, as if the array controller were merely a SCSI
controller in the same way that we are allowing it to access SCSI tape drives.

SCSI error handling for tape drives and medium changers

The linux SCSI mid layer provides an error handling protocol which
kicks into gear whenever a SCSI command fails to complete within a
certain amount of time (which can vary depending on the command).
The cciss driver participates in this protocol to some extent. The
normal protocol is a four step process. First the device is told
to abort the command. If that doesn't work, the device is reset.
If that doesn't work, the SCSI bus is reset. If that doesn't work
the host bus adapter is reset. Because the cciss driver is a block
driver as well as a SCSI driver and only the tape drives and medium
changers are presented to the SCSI mid layer, and unlike more
straightforward SCSI drivers, disk i/o continues through the block
side during the SCSI error recovery process, the cciss driver only
implements the first two of these actions, aborting the command, and
resetting the device. Additionally, most tape drives will not oblige
in aborting commands, and sometimes it appears they will not even
obey a reset command, though in most circumstances they will. In
the case that the command cannot be aborted and the device cannot be
reset, the device will be set offline.

In the event the error handling code is triggered and a tape drive is
successfully reset or the tardy command is successfully aborted, the
tape drive may still not allow i/o to continue until some command
is issued which positions the tape to a known position. Typically you

must rewind the tape (by issuing "mt -f /dev/st0 rewind" for example)
before i/o can proceed again to a tape drive which was reset.