aic7xxx_old.txt
AIC7xxx Driver for Linux

Introduction

The AIC7xxx SCSI driver adds support for Adaptec (http://www.adaptec.com) SCSI controllers and chipsets. Major portions of the driver and driver development are shared between both Linux and FreeBSD. Support for the AIC-7xxx chipsets have been in the default Linux kernel since approximately linux-1.1.x and fairly stable since linux-1.2.x, and are also in FreeBSD 2.1.0 or later.

Supported cards/chipsets

Adaptec Cards

AHA-274x
AHA-274xT
AHA-2842
AHA-2910B
AHA-2920C
AHA-2930
AHA-2930U
AHA-2930CU
AHA-2930U2
AHA-2940
AHA-2940W
AHA-2940U
AHA-2940UW
AHA-2940UW-PRO
AHA-2940AU
AHA-2940U2W
AHA-2940U2
AHA-2940U2B
AHA-2940U2BOEM
AHA-2944D
AHA-2944WD
AHA-2944UD
AHA-2944UWD
AHA-2950U2
AHA-2950U2W
AHA-2950U2B
AHA-29160M
AHA-3940
AHA-3940U
AHA-3940W
AHA-3940UW
AHA-3940AUW
AHA-3940U2W
AHA-3950U2B
AHA-3950U2D
AHA-3960D
AHA-39160M
AHA-3985
AHA-3985U
AHA-3985W
AHA-3985UW

Motherboard Chipsets

AIC-777x
AIC-785x
AIC-786x
AIC-787x
AIC-788x
AIC-789x
AIC-3860

Bus Types

W - Wide SCSI, SCSI-3, 16bit bus, 68pin connector, will also support SCSI-1/SCSI-2 50pin devices, transfer rates up to 20MB/s.
U - Ultra SCSI, transfer rates up to 40MB/s.
U2- Ultra 2 SCSI, transfer rates up to 80MB/s.
D - Differential SCSI.
T - Twin Channel SCSI. Up to 14 SCSI devices.

AHA-274x - EISA SCSI controller
AHA-284x - VLB SCSI controller
AHA-29xx - PCI SCSI controller
AHA-394x - PCI controllers with two separate SCSI controllers on-board.
AHA-398x - PCI RAID controllers with three separate SCSI controllers on-board.

Not Supported Devices

Adaptec Cards

AHA-2920 (Only the cards that use the Future Domain chipset are not supported, any 2920 cards based on Adaptec AIC chipsets, such as the 2920C, are supported)
AAA-13x Raid Adapters
AAA-113x Raid Port Card

Motherboard Chipsets

AIC-7810

Bus Types

R - Raid Port busses are not supported.

The hardware RAID devices sold by Adaptec are *NOT* supported by this driver (and will people please stop emailing me about them, they are a totally separate beast from the bare SCSI controllers and this driver cannot be retrofitted in any sane manner to support the hardware RAID features on those cards - Doug Ledford).

People

Justin T Gibbs gibbs@plutotech.com
(BSD Driver Author)

Dan Eischen deischen@iworks.InterWorks.org
(Original Linux Driver Co-maintainer)
Dean Gehnert deang@teleport.com
(Original Linux FTP/patch maintainer)
Jess Johnson jester@frenzy.com
(AIC7xxx FAQ author)
Doug Ledford dledford@redhat.com
(Current Linux aic7xxx-5.x.x Driver/Patch/FTP maintainer)

Special thanks go to John Aycock (aycock@cpsc.ucalgary.ca), the original author of the driver. John has since retired from the project. Thanks again for all his work!

Mailing list

There is a mailing list available for users who want to track development and converse with other users and developers. This list is for both FreeBSD and Linux support of the AIC7xxx chipsets.

To subscribe to the AIC7xxx mailing list send mail to the list server, with "subscribe AIC7xxx" in the body (no Subject: required):

```
To: majordomo@FreeBSD.ORG
---
subscribe AIC7xxx
```

To unsubscribe from the list, send mail to the list server with:

```
To: majordomo@FreeBSD.ORG
---
unsubscribe AIC7xxx
```

Send regular messages and replies to: AIC7xxx@FreeBSD.ORG

Boot Command line options

"aic7xxx=no_reset" - Eliminate the SCSI bus reset during startup. Some SCSI devices need the initial reset that this option disables in order to work. If you have problems at bootup, please make sure you aren't using this option.

"aic7xxx=reverse_scan" - Certain PCI motherboards scan for devices at bootup by scanning from the highest numbered PCI device to the lowest numbered PCI device, others do just the opposite and scan from lowest to highest numbered PCI device. There is no reliable way to autodetect this ordering. So, we default to the most common order, which is lowest to highest. Then, in case your motherboard scans from highest to lowest, we have this option. If your BIOS finds the drives on controller A before controller B but the linux kernel finds your drives on controller B before A, then you should use this option.

"aic7xxx=extended" - Force the driver to detect extended drive translation on your controller. This helps those people who have cards without a SEEPROM make sure that linux and all other operating systems think the same way about your hard drives.

"aic7xxx=scbram" - Some cards have external SCB RAM that can be used to

give the card more hardware SCB slots. This allows the driver to use that SCB RAM. Without this option, the driver won't touch the SCB RAM because it is known to cause problems on a few cards out there (such as 3985 class cards).

"aic7xxx=irq_trigger:x" - Replace x with either 0 or 1 to force the kernel to use the correct IRQ type for your card. This only applies to EISA based controllers. On these controllers, 0 is for Edge triggered interrupts, and 1 is for Level triggered interrupts. If you aren't sure or don't know which IRQ trigger type your EISA card uses, then let the kernel autodetect the trigger type.

"aic7xxx=verbose" - This option can be used in one of two ways. If you simply specify aic7xxx=verbose, then the kernel will automatically pick the default set of verbose messages for you to see. Alternatively, you can specify the command as "aic7xxx=verbose:0xXXXX" where the X entries are replaced with hexadecimal digits. This option is a bit field type option. For a full listing of the available options, search for the #define VERBOSE_xxxxxx lines in the aic7xxx.c file. If you want verbose messages, then it is recommended that you simply use the aic7xxx=verbose variant of this command.

"aic7xxx=pci_parity:x" - This option controls whether or not the driver enables PCI parity error checking on the PCI bus. By default, this checking is disabled. To enable the checks, simply specify pci_parity with no value afterwords. To reverse the parity from even to odd, supply any number other than 0 or 255. In short:

```
  pci_parity     - Even parity checking (even is the normal PCI parity)
  pci_parity:x   - Where x > 0, Odd parity checking
  pci_parity:0   - No check (default)
```

NOTE: In order to get Even PCI parity checking, you must use the version of the option that does not include the : and a number at the end (unless you want to enter exactly 2^32 - 1 as the number).

"aic7xxx=no_probe" - This option will disable the probing for any VLB based 2842 controllers and any EISA based controllers. This is needed on certain newer motherboards where the normal EISA I/O ranges have been claimed by other PCI devices. Probing on those machines will often result in the machine crashing or spontaneously rebooting during startup. Examples of machines that need this are the Dell PowerEdge 6300 machines.

"aic7xxx=seltime:2" - This option controls how long the card waits during a device selection sequence for the device to respond. The original SCSI spec says that this "should be" 256ms. This is generally not required with modern devices. However, some very old SCSI I devices need the full 256ms. Most modern devices can run fine with only 64ms. The default for this option is 64ms. If you need to change this option, then use the following table to set the proper value in the example above:

```
  0  -  256ms
  1  -  128ms
  2  -   64ms
  3  -   32ms
```

"aic7xxx=panic_on_abort" - This option is for debugging and will cause the driver to panic the linux kernel and freeze the system the first time the drivers abort or reset routines are called. This is most helpful when some problem causes infinite reset loops that scroll too fast to see. By using this option, you can write down what the errors actually are and send that information to me so it can be fixed.

"aic7xxx=dump_card" - This option will print out the *entire* set of configuration registers on the card during the init sequence. This is a debugging aid used to see exactly what state the card is in when we finally finish our initialization routines. If you don't have documentation on the chipsets, this will do you absolutely no good unless you are simply trying to write all the information down in order to send it to me.

"aic7xxx=dump_sequencer" - This is the same as the above options except that instead of dumping the register contents on the card, this option dumps the contents of the sequencer program RAM. This gives the ability to verify that the instructions downloaded to the card's sequencer are indeed what they are supposed to be. Again, unless you have documentation to tell you how to interpret these numbers, then it is totally useless.

"aic7xxx=override_term:0xffffffff" - This option is used to force the termination on your SCSI controllers to a particular setting. This is a bit mask variable that applies for up to 8 aic7xxx SCSI channels. Each channel gets 4 bits, divided as follows:

```
bit   3   2   1   0
      |   |   |   Enable/Disable Single Ended Low Byte Termination
      |   |   En/Disable Single Ended High Byte Termination
      |   En/Disable Low Byte LVD Termination
      En/Disable High Byte LVD Termination
```

The upper 2 bits that deal with LVD termination only apply to Ultra2 controllers. Furthermore, due to the current Ultra2 controller designs, these bits are tied together such that setting either bit enables both low and high byte LVD termination. It is not possible to only set high or low byte LVD termination in this manner. This is an artifact of the BIOS definition on Ultra2 controllers. For other controllers, the only important bits are the two lowest bits. Setting the higher bits on non-Ultra2 controllers has no effect. A few examples of how to use this option:

Enable low and high byte termination on a non-ultra2 controller that is the first aic7xxx controller (the correct bits are 0011),
aic7xxx=override_term:0x3

Enable all termination on the third aic7xxx controller, high byte termination on the second aic7xxx controller, and low and high byte SE termination on the first aic7xxx controller
(bits are 1111 0010 0011),
aic7xxx=override_term:0xf23

No attempt has been made to make this option non-cryptic. It really shouldn't be used except in dire circumstances, and if that happens, I'm probably going to be telling you what to set this to anyway :)

"aic7xxx=stpwlev:0xffffffff" - This option is used to control the STPWLEV bit in the DEVCONFIG PCI register. Currently, this is one of the very few registers that we have absolutely *no* way of detecting what the variable should be. It depends entirely on how the chipset and external terminators were coupled by the card/motherboard maker. Further, a chip reset (at power up) always sets this bit to 0. If there is no BIOS to run on the chipset/card (such as with a 2910C or a motherboard controller with the BIOS totally disabled) then the variable may not get set properly. Of course, if the proper setting was 0, then that's what it would be after the reset, but if the proper setting is actually 1.....you get the picture. Now, since we can't detect this at all, I've added this option to force the setting. If you have a BIOS on your controller then you should never need to use this option. However, if you are having lots of SCSI reset problems and can't seem to get them knocked out, this may help.

Here's a test to know for certain if you need this option. Make a boot floppy that you can use to boot your computer up and that will detect the aic7xxx controller. Next, power down your computer. While it's down, unplug all SCSI cables from your Adaptec SCSI controller. Boot the system back up to the Adaptec EZ-SCSI BIOS and then make sure that termination is enabled on your adapter (if you have an Adaptec BIOS of course). Next, boot up the floppy you made and wait for it to detect the aic7xxx controller. If the kernel finds the controller fine, says scsi : x hosts and then tries to detect your devices like normal, up to the point where it fails to mount your root file system and panics, then you're fine. If, on the other hand, the system goes into an infinite reset loop, then you need to use this option and/or the previous option to force the proper termination settings on your controller. If this happens, then you next need to figure out what your settings should be.

To find the correct settings, power your machine back down, connect back up the SCSI cables, and boot back into your machine like normal. However, boot with the aic7xxx=verbose:0x39 option. Record the initial DEVCONFIG values for each of your aic7xxx controllers as they are listed, and also record what the machine is detecting as the proper termination on your controllers. NOTE: the order in which the initial DEVCONFIG values are printed out is not guaranteed to be the same order as the SCSI controllers are registered. The above option and this option both work on the order of the SCSI controllers as they are registered, so make sure you match the right DEVCONFIG values with the right controllers if you have more than one aic7xxx controller.

Once you have the detected termination settings and the initial DEVCONFIG values for each controller, then figure out what the termination on each of the controllers *should* be. Hopefully, that part is correct, but it could possibly be wrong if there is bogus cable detection logic on your controller or something similar. If all the controllers have the correct termination settings, then don't set the aic7xxx=override_term variable at all, leave it alone. Next, on any controllers that go into an infinite reset loop when you unplug all the SCSI cables, get the starting DEVCONFIG value. If the initial DEVCONFIG value is divisible by 2, then the correct

setting for that controller is 0. If it's an odd number, then the correct setting for that controller is 1. For any other controllers that didn't have an infinite reset problem, then reverse the above options. If DEVCONFIG was even, then the correct setting is 1, if not then the correct setting is 0.

Now that you know what the correct setting was for each controller, we need to encode that into the aic7xxx=stpwlev:0x... variable. This variable is a bit field encoded variable. Bit 0 is for the first aic7xxx controller, bit 1 for the next, etc. Put all these bits together and you get a number. For example, if the third aic7xxx needed a 1, but the second and first both needed a 0, then the bits would be 100 in binary. This then translates to 0x04. You would therefore set aic7xxx=stpwlev:0x04. This is fairly standard binary to hexadecimal conversions here. If you aren't up to speed on the binary->hex conversion then send an email to the aic7xxx mailing list and someone can help you out.

"aic7xxx=tag_info:{{8,8..},{8,8..},..}" - This option is used to disable or enable Tagged Command Queueing (TCQ) on specific devices. As of driver version 5.1.11, TCQ is now either on or off by default according to the setting you choose during the make config process. In order to en/disable TCQ for certain devices at boot time, a user may use this boot param. The driver will then parse this message out and en/disable the specific device entries that are present based upon the value given. The param line is parsed in the following manner:

```
{ - first instance indicates the start of this parameter values
    second instance is the start of entries for a particular
    device entry
} - end the entries for a particular host adapter, or end the entire
    set of parameter entries
, - move to next entry.  Inside of a set of device entries, this
    moves us to the next device on the list.  Outside of device
    entries, this moves us to the next host adapter
. - Same effect as , but is safe to use with insmod.
x - the number to enter into the array at this position.
    0 = Enable tagged queueing on this device and use the default
        queue depth
    1-254 = Enable tagged queueing on this device and use this
            number as the queue depth
    255 = Disable tagged queueing on this device.
    Note: anything above 32 for an actual queue depth is wasteful
          and not recommended.
```

A few examples of how this can be used:

tag_info:{{8,12,,0,,255,4}}
  This line will only effect the first aic7xxx card registered. It will set scsi id 0 to a queue depth of 8, id 1 to 12, leave id 2 at the default, set id 3 to tagged queueing enabled and use the default queue depth, id 4 default, id 5 disabled, and id 6 to 4. Any not specified entries stay at the default value, repeated commas with no value specified will simply increment to the next id without changing anything for the missing values.

tag_info:{,,,{,,,255}}
First, second, and third adapters at default values. Fourth adapter, id 3 is disabled. Notice that leading commas simply increment what the first number effects, and there are no need for trailing commas. When you close out an adapter, or the entire entry, anything not explicitly set stays at the default value.

A final note on this option. The scanner I used for this isn't perfect or highly robust. If you mess the line up, the worst that should happen is that the line will get ignored. If you don't close out the entire entry with the final bracket, then any other aic7xxx options after this will get ignored. So, in general, be sure of what you are entering, and after you have it right, just add it to the lilo.conf file so there won't be any mistakes. As a means of checking this parser, the entire tag_info array for each card is now printed out in the /proc/scsi/aic7xxx/x file. You can use that to verify that your options were parsed correctly.

Boot command line options may be combined to form the proper set of options a user might need. For example, the following is valid:

aic7xxx=verbose,extended,irq_trigger:1

The only requirement is that individual options be separated by a comma or a period on the command line.

Module Loading command options

When loading the aic7xxx driver as a module, the exact same options are available to the user. However, the syntax to specify the options changes slightly. For insmod, you need to wrap the aic7xxx= argument in quotes and replace all ',' with '.'. So, for example, a valid insmod line would be:

insmod aic7xxx aic7xxx='verbose.irq_trigger:1.extended'

This line should result in the *exact* same behaviour as if you typed it in at the lilo prompt and the driver was compiled into the kernel instead of being a module. The reason for the single quote is so that the shell won't try to interpret anything in the line, such as {. Insmod assumes any options starting with a letter instead of a number is a character string (which is what we want) and by switching all of the commas to periods, insmod won't interpret this as more than one string and write junk into our binary image. I consider it a bug in the insmod program that even if you wrap your string in quotes (quotes that pass the shell mind you and that insmod sees) it still treats a comma inside of those quotes as starting a new variable, resulting in memory scribbles if you don't switch the commas to periods.

Kernel Compile options

The various kernel compile time options for this driver are now fairly well documented in the file Documentation/Configure.help. In order to see this documentation, you need to use one of the advanced configuration

programs (menuconfig and xconfig). If you are using the "make menuconfig" method of configuring your kernel, then you would simply highlight the option in question and hit the ? key. If you are using the "make xconfig" method of configuring your kernel, then simply click on the help button next to the option you have questions about. The help information from the Configure.help file will then get automatically displayed.

/proc support

The /proc support for the AIC7xxx can be found in the /proc/scsi/aic7xxx/ directory. That directory contains a file for each SCSI controller in the system. Each file presents the current configuration and transfer statistics (enabled with #define in aic7xxx.c) for each controller.

Thanks to Michael Neuffer for his upper-level SCSI help, and Matthew Jacob for statistics support.

Debugging the driver

Should you have problems with this driver, and would like some help in getting them solved, there are a couple debugging items built into the driver to facilitate getting the needed information from the system. In general, I need a complete description of the problem, with as many logs as possible concerning what happens. To help with this, there is a command option aic7xxx=panic_on_abort. This option, when set, forces the driver to panic the kernel on the first SCSI abort issued by the mid level SCSI code. If your system is going to reset loops and you can't read the screen, then this is what you need. Not only will it stop the system, but it also prints out a large amount of state information in the process. Second, if you specify the option "aic7xxx=verbose:0x1ffff", the system will print out *S0000* much information as it runs that you won't be able to see anything. However, this can actually be very useful if your machine simply locks up when trying to boot, since it will pin-point what was last happening (in regards to the aic7xxx driver) immediately prior to the lockup. This is really only useful if your machine simply can not boot up successfully. If you can get your machine to run, then this will produce far too much information.

FTP sites

ftp://ftp.redhat.com/pub/aic/
- Out of date. I used to keep stuff here, but too many people complained about having a hard time getting into Red Hat's ftp server. So use the web site below instead.

ftp://ftp.pcnet.com/users/eischen/Linux/
- Dan Eischen's driver distribution area

ftp://ekf2.vsb.cz/pub/linux/kernel/aic7xxx/ftp.teleport.com/
- European Linux mirror of Teleport site

Web sites

http://people.redhat.com/dledford/
- My web site, also the primary aic7xxx site with several related pages.

Dean W. Gehnert
deang@teleport.com

$Revision: 3.0 $

Modified by Doug Ledford 1998-2000