The Linux NCR53C8XX/SYM53C8XX drivers README file

Written by Gerard Roudier <groudier@free.fr>

21 Rue Carnot

95170 DEUIL LA BARRE - FRANCE

29 May 1999

1. Introduction

The initial Linux ncr53c8xx driver has been a port of the ncr driver from FreeBSD that has been achieved in November 1995 by:

- Gerard Roudier <groudier@free.fr>

The original driver has been written for 386bsd and FreeBSD by:

- Wolfgang Stanglmeier <wolf@cologne.de>
- Stefan Esser <se@mi.Uni-Koeln.de>

It is now available as a bundle of 2 drivers:

- ncr53c8xx generic driver that supports all the SYM53C8XX family including the earliest 810 rev. 1, the latest 896 (2 channel LVD SCSI controller) and the new 895A (1 channel LVD SCSI controller).
- sym53c8xx enhanced driver (a.k.a. 896 drivers) that drops support of oldest chips in order to gain advantage of new features, as LOAD/STORE instructions available since the 810A and hardware phase mismatch available with the 896 and the 895A.

You can find technical information about the NCR 8xx family in the PCI-HOWTO written by Michael Will and in the SCSI-HOWTO written by Drew Eckhardt.

Information about new chips is available at LSILOGIC web server:

- http://www.lsilogic.com/

SCSI standard documentations are available at SYMBIOS ftp server:

- ftp://ftp.symbios.com/

Useful SCSI tools written by Eric Youngdale are available at tsx-11:

- ftp://tsx-11.mit.edu/pub/linux/ALPHA/scsi/scsiinfo-X.Y.tar.gz
- ftp://tsx-11.mit.edu/pub/linux/ALPHA/scsi/scsidev-X.Y.tar.gz

These tools are not ALPHA but quite clean and work quite well. It is essential you have the 'scsiinfo' package.

This short documentation describes the features of the generic and enhanced drivers, configuration parameters and control commands available through the proc SCSI file system read / write operations.

This driver has been tested OK with linux/i386, Linux/Alpha and Linux/PPC.

Latest driver version and patches are available at:

- ftp://ftp.tux.org/pub/people/gerard-roudier

or

- ftp://ftp.symbios.com/mirror/ftp.tux.org/pub/tux/roudier/drivers

I am not a native speaker of English and there are probably lots of mistakes in this README file. Any help will be welcome.

2. Supported chips and SCSI features

The following features are supported for all chips:

- Synchronous negotiation
- Disconnection
- Tagged command queuing
- SCSI parity checking

- Master parity checking

"Wide negotiation" is supported for chips that allow it. The following table shows some characteristics of NCR 8xx family chips and what drivers support them.

Chip	On board SDMS BIOS	Wide	SCSI std.	Max. sync	Supported by the generic driver	Supported by the enhanced driver
810	N	N	FAST10	10 MB/s	Y	N
810A	N	N	FAST10	10 MB/s	Y	Y
815	Y	N	FAST10	10 MB/s	Y	N
825	Y	Y	FAST10	20 MB/s	Y	N
825A	Y	Y	FAST10	20 MB/s	Y	Y
860	N	N	FAST20	20 MB/s	Y	Y
875	Y	Y	FAST20	40 MB/s	Y	Y
876	Y	Y	FAST20	40 MB/s	Y	Y
895	Y	Y	FAST40	80 MB/s	Y	Y
895A	Y	Y	FAST40	80 MB/s	Y	Y
896	Y	Y	FAST40	80 MB/s	Y	Y
897	Y	Y	FAST40	80 MB/s	Y	Y
1510D	Y	Y	FAST40	80 MB/s	Y	Y
1010	Y	Y	FAST80	160 MB/s	N	Y
1010_66 [1]	Y	Y	FAST80	160 MB/s	N	Y

[1] Chip supports 33MHz and 66MHz PCI buses.

Summary of other supported features:

Module: allow to load the driver
Memory mapped I/O: increases performance
Profiling information: read operations from the proc SCSI file system
Control commands: write operations to the proc SCSI file system
Debugging information: written to syslog (expert only)
Serial NVRAM: Symbios and Tekram formats

- Scatter / gather
- Shared interrupt
- Boot setup commands

3. Advantages of the enhanced 896 driver

3.1 Optimized SCSI SCRIPTS

The 810A, 825A, 875, 895, 896 and 895A support new SCSI SCRIPTS instructions named LOAD and STORE that allow to move up to 1 DWORD from/to an IO register to/from memory much faster that the MOVE MEMORY instruction that is supported by the 53c7xx and 53c8xx family. The LOAD/STORE instructions support absolute and DSA relative addressing modes. The SCSI SCRIPTS had been entirely rewritten using LOAD/STORE instead of MOVE MEMORY instructions.

3.2 New features of the SYM53C896 (64 bit PCI dual LVD SCSI controller)

The 896 and the 895A allows handling of the phase mismatch context from SCRIPTS (avoids the phase mismatch interrupt that stops the SCSI processor until the C code has saved the context of the transfer). Implementing this without using LOAD/STORE instructions would be painful and I didn't even want to try it.

The 896 chip supports 64 bit PCI transactions and addressing, while the 895A supports 32 bit PCI transactions and 64 bit addressing. The SCRIPTS processor of these chips is not true 64 bit, but uses segment registers for bit 32-63. Another interesting feature is that LOAD/STORE instructions that address the on-chip RAM (8k) remain internal to the chip.

Due to the use of LOAD/STORE SCRIPTS instructions, this driver does not support the following chips:

- SYM53C810 revision < 0x10 (16)
- SYM53C815 all revisions
- SYM53C825 revision < 0x10 (16)

4. Memory mapped I/O versus normal I/O

Memory mapped I/O has less latency than normal I/O. Since linux-1.3.x, memory mapped I/O is used rather than normal I/O. Memory mapped I/O seems to work fine on most hardware configurations, but some poorly designed motherboards may break this feature.

The configuration option CONFIG_SCSI_NCR53C8XX_IOMAPPED forces the driver to use normal I/O in all cases.

5. Tagged command queueing

Queuing more than 1 command at a time to a device allows it to perform optimizations based on actual head positions and its mechanical characteristics. This feature may also reduce average command latency. In order to really gain advantage of this feature, devices must have a reasonable cache size (No miracle is to be expected for a low-end hard disk with 128 KB or less). Some known SCSI devices do not properly support tagged command queuing. Generally, firmware revisions that fix this kind of problems are available at respective vendor web/ftp sites. All I can say is that the hard disks I use on my machines behave well with this driver with tagged command queuing enabled:

- IBM S12 0662
- Conner 1080S
- Quantum Atlas I
- Quantum Atlas II

If your controller has NVRAM, you can configure this feature per target from the user setup tool. The Tekram Setup program allows to tune the maximum number of queued commands up to 32. The Symbios Setup only allows to enable or disable this feature.

The maximum number of simultaneous tagged commands queued to a device is currently set to 8 by default. This value is suitable for most SCSI disks. With large SCSI disks (>= 2GB, cache >= 512KB, average seek time <= 10 ms), using a larger value may give better performances.

The sym53c8xx driver supports up to 255 commands per device, and the generic ncr53c8xx driver supports up to 64, but using more than 32 is generally not worth-while, unless you are using a very large disk or disk array. It is noticeable that most of recent hard disks seem not to accept more than 64 simultaneous commands. So, using more than 64 queued commands is probably just resource wasting.

If your controller does not have NVRAM or if it is managed by the SDMS BIOS/SETUP, you can configure tagged queueing feature and device queue depths from the boot command-line. For example:

```
ncr53c8xx=tags:4/t2t3q15-t4q7/t1u0q32
```

will set tagged commands queue depths as follow:

- target 2 all luns on controller 0 --> 15
- target 3 all luns on controller 0 --> 15
- target 4 all luns on controller 0 --> 7
- target 1 lun 0 on controller 1 --> 32
- all other target/lun --> 4

In some special conditions, some SCSI disk firmwares may return a QUEUE FULL status for a SCSI command. This behaviour is managed by the driver using the following heuristic:

- Each time a QUEUE FULL status is returned, tagged queue depth is reduced to the actual number of disconnected commands.
- Every 1000 successfully completed SCSI commands, if allowed by the current limit, the maximum number of queueable commands is incremented.

Since QUEUE FULL status reception and handling is resource wasting, the driver notifies by default this problem to user by indicating the actual number of commands used and their status, as well as its decision on the device queue depth change. The heuristic used by the driver in handling QUEUE FULL ensures that the impact on performances is not too bad. You can get rid of the messages by setting verbose level to zero, as follow:

1st method:
boot your system using 'ncr53c8xx=verb:0' option.
2nd method:
apply "setverbose 0" control command to the proc fs entry corresponding to your controller after boot-up.

6. Parity checking

The driver supports SCSI parity checking and PCI bus master parity checking. These features must be enabled in order to ensure safe data transfers. However, some flawed devices or mother boards will have problems with parity. You can disable either PCI parity or SCSI parity checking by entering appropriate options from the boot command line. (See 10: Boot setup commands).

7. Profiling information

Profiling information is available through the proc SCSI file system. Since gathering profiling information may impact performances, this feature is disabled by default and requires a compilation configuration option to be set to Y.

The device associated with a host has the following pathname:

```
/proc/scsi/ncr53c8xx/N     (N=0,1,2 ....)
```

Generally, only 1 board is used on hardware configuration, and that device is:

```
/proc/scsi/ncr53c8xx/0
```

However, if the driver has been made as module, the number of the hosts is incremented each time the driver is loaded.

In order to display profiling information, just enter:

```
cat /proc/scsi/ncr53c8xx/0
```

and you will get something like the following text:

```
General information:
Chip NCR53C810, device id 0x1, revision id 0x2
IO port address 0x6000, IRQ number 10
Using memory mapped IO at virtual address 0x282c000
Synchronous transfer period 25, max commands per lun 4
Profiling information:
num_trans    = 18014
num_kbytes   = 671314
num_disc     = 25763
num_break    = 1673
num_int      = 1685
num_fly      = 18038
ms_setup     = 4940
ms_data      = 369940
ms_disc      = 183090
ms_post      = 1320
```

General information is easy to understand. The device ID and the revision ID identify the SCSI chip as follows:

Chip	**Device id**	**Revision Id**
810	0x1	< 0x10
810A	0x1	>= 0x10
815	0x4	
825	0x3	< 0x10
860	0x6	
825A	0x3	>= 0x10
875	0xf	
895	0xc	

The profiling information is updated upon completion of SCSI commands. A data structure is allocated and zeroed when the host adapter is attached. So, if the driver is a module, the profile counters are cleared each time the driver is loaded. The "clearprof" command allows you to clear these counters at any time.

The following counters are available:

("num" prefix means "number of", "ms" means milli-seconds)

num_trans
: Number of completed commands Example above: 18014 completed commands

num_kbytes
: Number of kbytes transferred Example above: 671 MB transferred

num_disc
: Number of SCSI disconnections Example above: 25763 SCSI disconnections

num_break
: number of script interruptions (phase mismatch) Example above: 1673 script interruptions

num_int
: Number of interrupts other than "on the fly" Example above: 1685 interruptions not "on the fly"

num_fly
: Number of interrupts "on the fly" Example above: 18038 interruptions "on the fly"

ms_setup
: Elapsed time for SCSI commands setups Example above: 4.94 seconds

ms_data
: Elapsed time for data transfers Example above: 369.94 seconds spent for data transfer

ms_disc
: Elapsed time for SCSI disconnections Example above: 183.09 seconds spent disconnected

ms_post
: Elapsed time for command post processing (time from SCSI status get to command completion call) Example above: 1.32 seconds spent for post processing

Due to the 1/100 second tick of the system clock, "ms_post" time may be wrong.

In the example above, we got 18038 interrupts "on the fly" and only 1673 script breaks generally due to disconnections inside a

segment of the scatter list.

8. Control commands

Control commands can be sent to the driver with write operations to the proc SCSI file system. The generic command syntax is the following:

```
echo "<verb> <parameters>" >/proc/scsi/ncr53c8xx/0
(assumes controller number is 0)
```

Using "all" for "<target>" parameter with the commands below will apply to all targets of the SCSI chain (except the controller).

Available commands:

8.1 Set minimum synchronous period factor

setsync <target> <period factor>

target: target number
period: minimum synchronous period. Maximum speed = 1000/(4*period factor) except for special cases below.

Specify a period of 255, to force asynchronous transfer mode.

- 10 means 25 nano-seconds synchronous period
- 11 means 30 nano-seconds synchronous period
- 12 means 50 nano-seconds synchronous period

8.2 Set wide size

setwide <target> <size>

target: target number
size: 0=8 bits, 1=16bits

8.3 Set maximum number of concurrent tagged commands

settags <target> <tags>

target: target number
tags: number of concurrent tagged commands must not be greater than SCSI_NCR_MAX_TAGS (default: 8)

8.4 Set order type for tagged command

setorder <order>

order: 3 possible values:

simple:
use SIMPLE TAG for all operations (read and write)

ordered:
use ORDERED TAG for all operations

default:
use default tag type, SIMPLE TAG for read operations ORDERED TAG for write operations

8.5 Set debug mode

setdebug <list of debug flags>

Available debug flags:

alloc	print info about memory allocations (ccb, lcb)
queue	print info about insertions into the command start queue
result	print sense data on CHECK CONDITION status
scatter	print info about the scatter process
scripts	print info about the script binding process
tiny	print minimal debugging information
timing	print timing information of the NCR chip
nego	print information about SCSI negotiations

phase	print information on script interruptions

Use "setdebug" with no argument to reset debug flags.

8.6 Clear profile counters

clearprof

The profile counters are automatically cleared when the amount of data transferred reaches 1000 GB in order to avoid overflow. The "clearprof" command allows you to clear these counters at any time.

8.7 Set flag (no_disc)

setflag <target> <flag>

target: target number

For the moment, only one flag is available:

no_disc: not allow target to disconnect.

Do not specify any flag in order to reset the flag. For example:

setflag 4
will reset no_disc flag for target 4, so will allow it disconnections.
setflag all
will allow disconnection for all devices on the SCSI bus.

8.8 Set verbose level

setverbose #level

The driver default verbose level is 1. This command allows to change th driver verbose level after boot-up.

8.9 Reset all logical units of a target

resetdev <target>

target: target number

The driver will try to send a BUS DEVICE RESET message to the target. (Only supported by the SYM53C8XX driver and provided for test purpose)

8.10 Abort all tasks of all logical units of a target

cleardev <target>

target: target number

The driver will try to send a ABORT message to all the logical units of the target.

(Only supported by the SYM53C8XX driver and provided for test purpose)

9. Configuration parameters

If the firmware of all your devices is perfect enough, all the features supported by the driver can be enabled at start-up. However, if only one has a flaw for some SCSI feature, you can disable the support by the driver of this feature at linux start-up and enable this feature after boot-up only for devices that support it safely.

CONFIG_SCSI_NCR53C8XX_IOMAPPED (default answer: n)

Answer "y" if you suspect your mother board to not allow memory mapped I/O.

May slow down performance a little. This option is required by Linux/PPC and is used no matter what you select here. Linux/PPC suffers no performance loss with this option since all IO is memory mapped anyway.

CONFIG_SCSI_NCR53C8XX_DEFAULT_TAGS (default answer: 8)

Default tagged command queue depth.

CONFIG_SCSI_NCR53C8XX_MAX_TAGS (default answer: 8)

This option allows you to specify the maximum number of tagged commands that can be queued to a device. The maximum supported value is 32.

CONFIG_SCSI_NCR53C8XX_SYNC (default answer: 5)

This option allows you to specify the frequency in MHz the driver will use at boot time for synchronous data transfer negotiations. This frequency can be changed later with the "setsync" control command. 0 means "asynchronous data transfers".

CONFIG_SCSI_NCR53C8XX_FORCE_SYNC_NEGO (default answer: n)

Force synchronous negotiation for all SCSI-2 devices.

Some SCSI-2 devices do not report this feature in byte 7 of inquiry response but do support it properly (TAMARACK scanners for example).

CONFIG_SCSI_NCR53C8XX_NO_DISCONNECT (default and only reasonable answer: n)

If you suspect a device of yours does not properly support disconnections, you can answer "y". Then, all SCSI devices will never disconnect the bus even while performing long SCSI operations.

CONFIG_SCSI_NCR53C8XX_SYMBIOS_COMPAT

Genuine SYMBIOS boards use GPIO0 in output for controller LED and GPIO3 bit as a flag indicating singled-ended/differential interface. If all the boards of your system are genuine SYMBIOS boards or use BIOS and drivers from SYMBIOS, you would want to enable this option.

This option must NOT be enabled if your system has at least one 53C8XX based scsi board with a vendor-specific BIOS. For example, Tekram DC-390/U, DC-390/W and DC-390/F scsi controllers use a vendor-specific BIOS and are known to not use SYMBIOS compatible GPIO wiring. So, this option must not be enabled if your system has such a board installed.

CONFIG_SCSI_NCR53C8XX_NVRAM_DETECT

Enable support for reading the serial NVRAM data on Symbios and some Symbios compatible cards, and Tekram DC390W/U/F cards. Useful for systems with more than one Symbios compatible controller where at least one has a serial NVRAM, or for a system with a mixture of Symbios and Tekram cards. Enables setting the boot order of host adaptors to something other than the default order or "reverse probe" order. Also enables Symbios and Tekram cards to be distinguished so CONFIG_SCSI_NCR53C8XX_SYMBIOS_COMPAT may be set in a system with a mixture of Symbios and Tekram cards so the Symbios cards can make use of the full range of Symbios features, differential, led pin, without causing problems for the Tekram card(s).

10. Boot setup commands

10.1 Syntax

Setup commands can be passed to the driver either at boot time or as a string variable using 'insmod'.

A boot setup command for the ncr53c8xx (sym53c8xx) driver begins with the driver name "ncr53c8xx="(sym53c8xx). The kernel syntax parser then expects an optional list of integers separated with comma followed by an optional list of comma-separated strings. Example of boot setup command under lilo prompt:

```
lilo: linux root=/dev/hda2 ncr53c8xx=tags:4,sync:10,debug:0x200
```

- enable tagged commands, up to 4 tagged commands queued.
- set synchronous negotiation speed to 10 Mega-transfers / second.
- set DEBUG_NEGO flag.

Since comma seems not to be allowed when defining a string variable using 'insmod', the driver also accepts <space> as option separator. The following command will install driver module with the same options as above:

```
insmod ncr53c8xx.o ncr53c8xx="tags:4 sync:10 debug:0x200"
```

For the moment, the integer list of arguments is discarded by the driver. It will be used in the future in order to allow a per controller setup.

Each string argument must be specified as "keyword:value". Only lower-case characters and digits are allowed.

In a system that contains multiple 53C8xx adapters insmod will install the specified driver on each adapter. To exclude a chip use the 'excl' keyword.

The sequence of commands:

```
insmod sym53c8xx sym53c8xx=excl:0x1400
insmod ncr53c8xx
```

installs the sym53c8xx driver on all adapters except the one at IO port address 0x1400 and then installs the ncr53c8xx driver to the adapter at IO port address 0x1400.

10.2 Available arguments

10.2.1 Master parity checking

mpar:y	enabled
mpar:n	disabled

10.2.2 Scsi parity checking

spar:y	enabled
spar:n	disabled

10.2.3 Scsi disconnections

disc:y	enabled
disc:n	disabled

10.2.4 Special features

Only apply to 810A, 825A, 860, 875 and 895 controllers. Have no effect with other ones.

specf:y	(or 1) enabled
specf:n	(or 0) disabled
specf:3	enabled except Memory Write And Invalidate

The default driver setup is 'specf:3'. As a consequence, option 'specf:y' must be specified in the boot setup command to enable Memory Write And Invalidate.

10.2.5 Ultra SCSI support

Only apply to 860, 875, 895, 895a, 896, 1010 and 1010_66 controllers. Have no effect with other ones.

ultra:n	All ultra speeds enabled
ultra:2	Ultra2 enabled
ultra:1	Ultra enabled
ultra:0	Ultra speeds disabled

10.2.6 Default number of tagged commands

tags:0 (or tags:1)	tagged command queuing disabled
tags:#tags (#tags > 1)	tagged command queuing enabled

#tags will be truncated to the max queued commands configuration parameter. This option also allows to specify a command queue depth for each device that support tagged command queueing.

Example:

```
ncr53c8xx=tags:10/t2t3q16-t5q24/t1u2q32
```

will set devices queue depth as follow:

- controller #0 target #2 and target #3 -> 16 commands,
- controller #0 target #5 -> 24 commands,
- controller #1 target #1 logical unit #2 -> 32 commands,
- all other logical units (all targets, all controllers) -> 10 commands.

10.2.7 Default synchronous period factor

sync:255	disabled (asynchronous transfer mode)	
sync:#factor	#factor = 10	Ultra-2 SCSI 40 Mega-transfers / second
	#factor = 11	Ultra-2 SCSI 33 Mega-transfers / second
	#factor < 25	Ultra SCSI 20 Mega-transfers / second
	#factor < 50	Fast SCSI-2

In all cases, the driver will use the minimum transfer period supported by controllers according to NCR53C8XX chip

type.

10.2.8 Negotiate synchronous with all devices

(force sync nego)

fsn:y	enabled
fsn:n	disabled

10.2.9 Verbosity level

verb:0	minimal
verb:1	normal
verb:2	too much

10.2.10 Debug mode

<table>
<tr><td>debug:0</td><td>clear debug flags</td></tr>
<tr><td>debug:#x</td><td>set debug flags

#x is an integer value combining the following power-of-2 values:

<table>
<tr><td>DEBUG_ALLOC</td><td>0x1</td></tr>
<tr><td>DEBUG_PHASE</td><td>0x2</td></tr>
<tr><td>DEBUG_POLL</td><td>0x4</td></tr>
<tr><td>DEBUG_QUEUE</td><td>0x8</td></tr>
<tr><td>DEBUG_RESULT</td><td>0x10</td></tr>
<tr><td>DEBUG_SCATTER</td><td>0x20</td></tr>
<tr><td>DEBUG_SCRIPT</td><td>0x40</td></tr>
<tr><td>DEBUG_TINY</td><td>0x80</td></tr>
<tr><td>DEBUG_TIMING</td><td>0x100</td></tr>
<tr><td>DEBUG_NEGO</td><td>0x200</td></tr>
<tr><td>DEBUG_TAGS</td><td>0x400</td></tr>
<tr><td>DEBUG_FREEZE</td><td>0x800</td></tr>
<tr><td>DEBUG_RESTART</td><td>0x1000</td></tr>
</table>
</td></tr>
</table>

You can play safely with DEBUG_NEGO. However, some of these flags may generate bunches of syslog messages.

10.2.11 Burst max

burst:0	burst disabled
burst:255	get burst length from initial IO register settings.
burst:#x	burst enabled (1<<#x burst transfers max) #x is an integer value which is log base 2 of the burst transfers max. The NCR53C875 and NCR53C825A support up to 128 burst transfers (#x = 7). Other chips only support up to 16 (#x = 4). This is a maximum value. The driver set the burst length according to chip and revision ids. By default the driver uses the maximum value supported by the chip.

10.2.12 LED support

led:1	enable LED support
led:0	disable LED support

Donnot enable LED support if your scsi board does not use SDMS BIOS. (See 'Configuration parameters')

10.2.13 Max wide

wide:1	wide scsi enabled
wide:0	wide scsi disabled

Some scsi boards use a 875 (ultra wide) and only supply narrow connectors. If you have connected a wide device with a 50 pins to 68 pins cable converter, any accepted wide negotiation will break further data transfers. In such a case, using "wide:0" in the bootup command will be helpful.

10.2.14 Differential mode

diff:0	never set up diff mode
diff:1	set up diff mode if BIOS set it
diff:2	always set up diff mode
diff:3	set diff mode if GPIO3 is not set

10.2.15 IRQ mode

irqm:0	always open drain
irqm:1	same as initial settings (assumed BIOS settings)
irqm:2	always totem pole
irqm:0x10	driver will not use IRQF_SHARED flag when requesting irq

(Bits 0x10 and 0x20 can be combined with hardware irq mode option)

10.2.16 Reverse probe

revprob:n	probe chip ids from the PCI configuration in this order: 810, 815, 820, 860, 875, 885, 895, 896
revprob:y	probe chip ids in the reverse order.

10.2.17 Fix up PCI configuration space

pcifix:<option bits>

Available option bits:

0x0	No attempt to fix PCI configuration space registers values.
0x1	Set PCI cache-line size register if not set.
0x2	Set write and invalidate bit in PCI command register.
0x4	Increase if necessary PCI latency timer according to burst max.

Use 'pcifix:7' in order to allow the driver to fix up all PCI features.

10.2.18 Serial NVRAM

nvram:n	do not look for serial NVRAM
nvram:y	test controllers for onboard serial NVRAM

(alternate binary form) nvram=<bits options>

0x01	look for NVRAM (equivalent to nvram=y)
0x02	ignore NVRAM "Synchronous negotiation" parameters for all devices
0x04	ignore NVRAM "Wide negotiation" parameter for all devices
0x08	ignore NVRAM "Scan at boot time" parameter for all devices
0x80	also attach controllers set to OFF in the NVRAM (sym53c8xx only)

10.2.19 Check SCSI BUS

buschk:<option bits>

Available option bits:

0x0:	No check.
0x1:	Check and do not attach the controller on error.
0x2:	Check and just warn on error.
0x4:	Disable SCSI bus integrity checking.

10.2.20 Exclude a host from being attached

excl=<io_address>

Prevent host at a given io address from being attached. For example 'ncr53c8xx=excl:0xb400,excl:0xc000' indicate to the ncr53c8xx driver not to attach hosts at address 0xb400 and 0xc000.

10.2.21 Suggest a default SCSI id for hosts

hostid:255	no id suggested.
hostid:#x	(0 < x < 7) x suggested for hosts SCSI id.

If a host SCSI id is available from the NVRAM, the driver will ignore any value suggested as boot option. Otherwise, if a suggested value different from 255 has been supplied, it will use it. Otherwise, it will try to deduce the value previously set in the hardware and use value 7 if the hardware value is zero.

10.2.22 Enable use of IMMEDIATE ARBITRATION

(only supported by the sym53c8xx driver. See 10.7 for more details)

iarb:0	do not use this feature.	
iarb:#x	use this feature according to bit fields as follow:	
	bit 0 (1)	enable IARB each time the initiator has been reselected when it arbitrated for the SCSI BUS.
	(#x >> 4)	maximum number of successive settings of IARB if the initiator win arbitration and it has other commands to send to a device.

Boot fail safe

safe:y load the following assumed fail safe initial setup

master parity	disabled	mpar:n
scsi parity	enabled	spar:y
disconnections	not allowed	disc:n
special features	disabled	specf:n
ultra scsi	disabled	ultra:n
force sync negotiation	disabled	fsn:n
reverse probe	disabled	revprob:n
PCI fix up	disabled	pcifix:0
serial NVRAM	enabled	nvram:y
verbosity level	2	verb:2
tagged command queuing	disabled	tags:0
synchronous negotiation	disabled	sync:255
debug flags	none	debug:0
burst length	from BIOS settings	burst:255
LED support	disabled	led:0
wide support	disabled	wide:0
settle time	10 seconds	settle:10
differential support	from BIOS settings	diff:1
irq mode	from BIOS settings	irqm:1
SCSI BUS check	do not attach on error	buschk:1
immediate arbitration	disabled	iarb:0

10.3 Advised boot setup commands

If the driver has been configured with default options, the equivalent boot setup is:

```
ncr53c8xx=mpar:y,spar:y,disc:y,specf:3,fsn:n,ultra:2,fsn:n,revprob:n,verb:1\
          tags:0,sync:50,debug:0,burst:7,led:0,wide:1,settle:2,diff:0,irqm:0
```

For an installation diskette or a safe but not fast system, boot setup can be:

```
ncr53c8xx=safe:y,mpar:y,disc:y
ncr53c8xx=safe:y,disc:y
ncr53c8xx=safe:y,mpar:y
ncr53c8xx=safe:y
```

My personal system works flawlessly with the following equivalent setup:

```
ncr53c8xx=mpar:y,spar:y,disc:y,specf:1,fsn:n,ultra:2,fsn:n,revprob:n,verb:1\
          tags:32,sync:12,debug:0,burst:7,led:1,wide:1,settle:2,diff:0,irqm:0
```

The driver prints its actual setup when verbosity level is 2. You can try "ncr53c8xx=verb:2" to get the "static" setup of the driver, or add "verb:2" to your boot setup command in order to check the actual setup the driver is using.

10.4 PCI configuration fix-up boot option

pcifix:<option bits>

Available option bits:

0x1	Set PCI cache-line size register if not set.
0x2	Set write and invalidate bit in PCI command register.

Use 'pcifix:3' in order to allow the driver to fix both PCI features.

These options only apply to new SYMBIOS chips 810A, 825A, 860, 875 and 895 and are only supported for Pentium and 486 class processors. Recent SYMBIOS 53C8XX scsi processors are able to use PCI read multiple and PCI write and invalidate commands. These features require the cache line size register to be properly set in the PCI configuration space of the chips. On the other hand, chips will use PCI write and invalidate commands only if the corresponding bit is set to 1 in the PCI command register.

Not all PCI bioses set the PCI cache line register and the PCI write and invalidate bit in the PCI configuration space of 53C8XX chips. Optimized PCI accesses may be broken for some PCI/memory controllers or make problems with some PCI boards.

This fix-up worked flawlessly on my previous system. (MB Triton HX / 53C875 / 53C810A) I use these options at my own risks as you will do if you decide to use them too.

10.5 Serial NVRAM support boot option

nvram:n	do not look for serial NVRAM
nvram:y	test controllers for onboard serial NVRAM

This option can also been entered as an hexadecimal value that allows to control what information the driver will get from the NVRAM and what information it will ignore. For details see '17. Serial NVRAM support'.

When this option is enabled, the driver tries to detect all boards using a Serial NVRAM. This memory is used to hold user set up parameters.

The parameters the driver is able to get from the NVRAM depend on the data format used, as follow:

	Tekram format	Symbios format
General and host parameters		
• Boot order	N	Y
• Host SCSI ID	Y	Y
• SCSI parity checking	Y	Y
• Verbose boot messages	N	Y
SCSI devices parameters		
• Synchronous transfer speed	Y	Y
• Wide 16 / Narrow	Y	Y
• Tagged Command Queuing enabled	Y	Y
• Disconnections enabled	Y	Y
• Scan at boot time	N	Y

In order to speed up the system boot, for each device configured without the "scan at boot time" option, the driver forces an error on the first TEST UNIT READY command received for this device.

Some SDMS BIOS revisions seem to be unable to boot cleanly with very fast hard disks. In such a situation you cannot configure the NVRAM with optimized parameters value.

The 'nvram' boot option can be entered in hexadecimal form in order to ignore some options configured in the NVRAM, as follow:

nvram=<bits options>

0x01	look for NVRAM (equivalent to nvram=y)
0x02	ignore NVRAM "Synchronous negotiation" parameters for all devices
0x04	ignore NVRAM "Wide negotiation" parameter for all devices
0x08	ignore NVRAM "Scan at boot time" parameter for all devices
0x80	also attach controllers set to OFF in the NVRAM (sym53c8xx only)

Option 0x80 is only supported by the sym53c8xx driver and is disabled by default. Result is that, by default (option not set), the sym53c8xx driver will not attach controllers set to OFF in the NVRAM.

The ncr53c8xx always tries to attach all the controllers. Option 0x80 has not been added to the ncr53c8xx driver, since it has been reported to confuse users who use this driver since a long time. If you desire a controller not to be attached by the ncr53c8xx driver

at Linux boot, you must use the 'excl' driver boot option.

10.6 SCSI BUS checking boot option.

When this option is set to a non-zero value, the driver checks SCSI lines logic state, 100 micro-seconds after having asserted the SCSI RESET line. The driver just reads SCSI lines and checks all lines read FALSE except RESET. Since SCSI devices shall release the BUS at most 800 nano-seconds after SCSI RESET has been asserted, any signal to TRUE may indicate a SCSI BUS problem. Unfortunately, the following common SCSI BUS problems are not detected:

- Only 1 terminator installed.
- Misplaced terminators.
- Bad quality terminators.

On the other hand, either bad cabling, broken devices, not conformant devices, ... may cause a SCSI signal to be wrong when te driver reads it.

10.7 IMMEDIATE ARBITRATION boot option

This option is only supported by the SYM53C8XX driver (not by the NCR53C8XX).

SYMBIOS 53C8XX chips are able to arbitrate for the SCSI BUS as soon as they have detected an expected disconnection (BUS FREE PHASE). For this process to be started, bit 1 of SCNTL1 IO register must be set when the chip is connected to the SCSI BUS.

When this feature has been enabled for the current connection, the chip has every chance to win arbitration if only devices with lower priority are competing for the SCSI BUS. By the way, when the chip is using SCSI id 7, then it will for sure win the next SCSI BUS arbitration.

Since, there is no way to know what devices are trying to arbitrate for the BUS, using this feature can be extremely unfair. So, you are not advised to enable it, or at most enable this feature for the case the chip lost the previous arbitration (boot option 'iarb:1').

This feature has the following advantages:

a. Allow the initiator with ID 7 to win arbitration when it wants so.
b. Overlap at least 4 micro-seconds of arbitration time with the execution of SCRIPTS that deal with the end of the current connection and that starts the next job.

Hmmm... But (a) may just prevent other devices from reselecting the initiator, and delay data transfers or status/completions, and (b) may just waste SCSI BUS bandwidth if the SCRIPTS execution lasts more than 4 micro-seconds.

The use of IARB needs the SCSI_NCR_IARB_SUPPORT option to have been defined at compile time and the 'iarb' boot option to have been set to a non zero value at boot time. It is not that useful for real work, but can be used to stress SCSI devices or for some applications that can gain advantage of it. By the way, if you experience badnesses like 'unexpected disconnections', 'bad reselections', etc... when using IARB on heavy IO load, you should not be surprised, because force-feeding anything and blocking its arse at the same time cannot work for a long time. :-))

11. Some constants and flags of the ncr53c8xx.h header file

Some of these are defined from the configuration parameters. To change other "defines", you must edit the header file. Do that only if you know what you are doing.

SCSI_NCR_SETUP_SPECIAL_FEATURES (default: defined)

If defined, the driver will enable some special features according to chip and revision id.

For 810A, 860, 825A, 875 and 895 scsi chips, this option enables support of features that reduce load of PCI bus and memory accesses during scsi transfer processing: burst op-code fetch, read multiple, read line, prefetch, cache line, write and invalidate, burst 128 (875 only), large dma fifo (875 only), offset 16 (875 only). Can be changed by the following boot setup command:

```
ncr53c8xx=specf:n
```

SCSI_NCR_IOMAPPED (default: not defined)

If defined, normal I/O is forced.

SCSI_NCR_SHARE_IRQ (default: defined)

If defined, request shared IRQ.

SCSI_NCR_MAX_TAGS (default: 8)

Maximum number of simultaneous tagged commands to a device.

Can be changed by "settags <target> <maxtags>"

SCSI_NCR_SETUP_DEFAULT_SYNC (default: 50)

Transfer period factor the driver will use at boot time for synchronous negotiation. 0 means asynchronous.

Can be changed by "setsync <target> <period factor>"

SCSI_NCR_SETUP_DEFAULT_TAGS (default: 8)

Default number of simultaneous tagged commands to a device.

< 1 means tagged command queuing disabled at start-up.

SCSI_NCR_ALWAYS_SIMPLE_TAG (default: defined)

Use SIMPLE TAG for read and write commands.

Can be changed by "setorder <ordered|simple|default>"

SCSI_NCR_SETUP_DISCONNECTION (default: defined)

If defined, targets are allowed to disconnect.

SCSI_NCR_SETUP_FORCE_SYNC_NEGO (default: not defined)

If defined, synchronous negotiation is tried for all SCSI-2 devices.

Can be changed by "setsync <target> <period>"

SCSI_NCR_SETUP_MASTER_PARITY (default: defined)

If defined, master parity checking is enabled.

SCSI_NCR_SETUP_SCSI_PARITY (default: defined)

If defined, SCSI parity checking is enabled.

SCSI_NCR_PROFILE_SUPPORT (default: not defined)

If defined, profiling information is gathered.

SCSI_NCR_MAX_SCATTER (default: 128)

Scatter list size of the driver ccb.

SCSI_NCR_MAX_TARGET (default: 16)

Max number of targets per host.

SCSI_NCR_MAX_HOST (default: 2)

Max number of host controllers.

SCSI_NCR_SETTLE_TIME (default: 2)

Number of seconds the driver will wait after reset.

SCSI_NCR_TIMEOUT_ALERT (default: 3)

If a pending command will time out after this amount of seconds, an ordered tag is used for the next command.

Avoids timeouts for unordered tagged commands.

SCSI_NCR_CAN_QUEUE (default: 7*SCSI_NCR_MAX_TAGS)

Max number of commands that can be queued to a host.

SCSI_NCR_CMD_PER_LUN (default: SCSI_NCR_MAX_TAGS)

Max number of commands queued to a host for a device.

SCSI_NCR_SG_TABLESIZE (default: SCSI_NCR_MAX_SCATTER-1)

Max size of the Linux scatter/gather list.

SCSI_NCR_MAX_LUN (default: 8)

Max number of LUNs per target.

12. Installation

This driver is part of the linux kernel distribution. Driver files are located in the sub-directory "drivers/scsi" of the kernel source tree.

Driver files:

```
README.ncr53c8xx        : this file
ChangeLog.ncr53c8xx     : change log
ncr53c8xx.h             : definitions
ncr53c8xx.c             : the driver code
```

New driver versions are made available separately in order to allow testing changes and new features prior to including them into the linux kernel distribution. The following URL provides information on latest available patches:

ftp://ftp.tux.org/pub/people/gerard-roudier/README

13. Architecture dependent features

<Not yet written>

14. Known problems

14.1 Tagged commands with Iomega Jaz device

I have not tried this device, however it has been reported to me the following: This device is capable of Tagged command queuing. However while spinning up, it rejects Tagged commands. This behaviour is conforms to 6.8.2 of SCSI-2 specifications. The current behaviour of the driver in that situation is not satisfying. So do not enable Tagged command queuing for devices that are able to spin down. The other problem that may appear is timeouts. The only way to avoid timeouts seems to edit linux/drivers/scsi/sd.c and to increase the current timeout values.

14.2 Device names change when another controller is added

When you add a new NCR53C8XX chip based controller to a system that already has one or more controllers of this family, it may happen that the order the driver registers them to the kernel causes problems due to device name changes. When at least one controller uses NvRAM, SDMS BIOS version 4 allows you to define the order the BIOS will scan the scsi boards. The driver attaches controllers according to BIOS information if NvRAM detect option is set.

If your controllers do not have NvRAM, you can:

- Ask the driver to probe chip ids in reverse order from the boot command line: ncr53c8xx=revprob:y
- Make appropriate changes in the fstab.
- Use the 'scsidev' tool from Eric Youngdale.

14.3 Using only 8 bit devices with a WIDE SCSI controller

When only 8 bit NARROW devices are connected to a 16 bit WIDE SCSI controller, you must ensure that lines of the wide part of the SCSI BUS are pulled-up. This can be achieved by ENABLING the WIDE TERMINATOR portion of the SCSI controller card.

The TYAN 1365 documentation revision 1.2 is not correct about such settings. (page 10, figure 3.3).

14.4 Possible data corruption during a Memory Write and Invalidate

This problem is described in SYMBIOS DEL 397, Part Number 69-039241, ITEM 4.

In some complex situations, 53C875 chips revision <= 3 may start a PCI Write and Invalidate Command at a not cache-line-aligned 4 DWORDS boundary. This is only possible when Cache Line Size is 8 DWORDS or greater. Pentium systems use a 8 DWORDS cache line size and so are concerned by this chip bug, unlike i486 systems that use a 4 DWORDS cache line size.

When this situation occurs, the chip may complete the Write and Invalidate command after having only filled part of the last cache line involved in the transfer, leaving to data corruption the remainder of this cache line.

Not using Write And Invalidate obviously gets rid of this chip bug, and so it is now the default setting of the driver. However, for people like me who want to enable this feature, I have added part of a work-around suggested by SYMBIOS. This work-around resets the addressing logic when the DATA IN phase is entered and so prevents the bug from being triggered for the first SCSI MOVE of the phase. This work-around should be enough according to the following:

The only driver internal data structure that is greater than 8 DWORDS and that is moved by the SCRIPTS processor is the 'CCB header' that contains the context of the SCSI transfer. This data structure is aligned on 8 DWORDS boundary (Pentium Cache Line Size), and so is immune to this chip bug, at least on Pentium systems.

But the conditions of this bug can be met when a SCSI read command is performed using a buffer that is 4 DWORDS but not cache-line aligned. This cannot happen under Linux when scatter/gather lists are used since they only refer to system buffers that are well aligned. So, a work around may only be needed under Linux when a scatter/gather list is not used and when the SCSI DATA IN phase is reentered after a phase mismatch.

15. SCSI problem troubleshooting

15.1 Problem tracking

Most SCSI problems are due to a non conformant SCSI bus or to buggy devices. If unfortunately you have SCSI problems, you can check the following things:

- SCSI bus cables
- terminations at both end of the SCSI chain
- linux syslog messages (some of them may help you)

If you do not find the source of problems, you can configure the driver with no features enabled.

- only asynchronous data transfers

- tagged commands disabled
- disconnections not allowed

Now, if your SCSI bus is ok, your system have every chance to work with this safe configuration but performances will not be optimal.

If it still fails, then you can send your problem description to appropriate mailing lists or news-groups. Send me a copy in order to be sure I will receive it. Obviously, a bug in the driver code is possible.

My email address: Gerard Roudier <groudier@free.fr>

Allowing disconnections is important if you use several devices on your SCSI bus but often causes problems with buggy devices. Synchronous data transfers increases throughput of fast devices like hard disks. Good SCSI hard disks with a large cache gain advantage of tagged commands queuing.

Try to enable one feature at a time with control commands. For example:

```
echo "setsync all 25" >/proc/scsi/ncr53c8xx/0
```

Will enable fast synchronous data transfer negotiation for all targets.

```
echo "setflag 3" >/proc/scsi/ncr53c8xx/0
```

Will reset flags (no_disc) for target 3, and so will allow it to disconnect the SCSI Bus.

```
echo "settags 3 8" >/proc/scsi/ncr53c8xx/0
```

Will enable tagged command queuing for target 3 if that device supports it.

Once you have found the device and the feature that cause problems, just disable that feature for that device.

15.2 Understanding hardware error reports

When the driver detects an unexpected error condition, it may display a message of the following pattern:

```
sym53c876-0:1: ERROR (0:48) (1-21-65) (f/95) @ (script 7c0:19000000).
sym53c876-0: script cmd = 19000000
sym53c876-0: regdump: da 10 80 95 47 0f 01 07 75 01 81 21 80 01 09 00.
```

Some fields in such a message may help you understand the cause of the problem, as follows:

```
sym53c876-0:1: ERROR (0:48) (1-21-65) (f/95) @ (script 7c0:19000000).
............A.........B.C....D.E..F....G.H.......I.....J...K.......
```

Field A : *target number.*

SCSI ID of the device the controller was talking with at the moment the error occurs.

Field B : *DSTAT io register (DMA STATUS)*

Bit 0x40	MDPE Master Data Parity Error Data parity error detected on the PCI BUS.
Bit 0x20	BF Bus Fault PCI bus fault condition detected
Bit 0x01	IID Illegal Instruction Detected Set by the chip when it detects an Illegal Instruction format on some condition that makes an instruction illegal.
Bit 0x80	DFE Dma Fifo Empty Pure status bit that does not indicate an error.

If the reported DSTAT value contains a combination of MDPE (0x40), BF (0x20), then the cause may be likely due to a PCI BUS problem.

Field C : *SIST io register (SCSI Interrupt Status)*

Bit 0x08	SGE SCSI GROSS ERROR Indicates that the chip detected a severe error condition on the SCSI BUS that prevents the SCSI protocol from functioning properly.
Bit 0x04	UDC Unexpected Disconnection Indicates that the device released the SCSI BUS when the chip was not expecting this to happen. A device may behave so to indicate the SCSI initiator that an error condition not reportable using the SCSI protocol has occurred.
Bit 0x02	RST SCSI BUS Reset Generally SCSI targets do not reset the SCSI BUS, although any device on the BUS can reset it at any time.
Bit 0x01	PAR Parity SCSI parity error detected.

On a faulty SCSI BUS, any error condition among SGE (0x08), UDC (0x04) and PAR (0x01) may be detected by the chip. If your SCSI system sometimes encounters such error conditions, especially SCSI GROSS ERROR, then a SCSI BUS problem is likely the cause of these errors.

For fields D,E,F,G and H, you may look into the sym53c8xx_defs.h file that contains some minimal comments on IO register bits.

Field D : *SOCL Scsi Output Control Latch*

This register reflects the state of the SCSI control lines the chip want to drive or compare against.

Field E : *SBCL Scsi Bus Control Lines*
Actual value of control lines on the SCSI BUS.

Field F : *SBDL Scsi Bus Data Lines*
Actual value of data lines on the SCSI BUS.

Field G : *SXFER SCSI Transfer*
Contains the setting of the Synchronous Period for output and the current Synchronous offset (offset 0 means asynchronous).

Field H : *SCNTL3 Scsi Control Register 3*
Contains the setting of timing values for both asynchronous and synchronous data transfers.

Understanding Fields I, J, K and dumps requires to have good knowledge of SCSI standards, chip cores functionnals and internal driver data structures. You are not required to decode and understand them, unless you want to help maintain the driver code.

16. Synchronous transfer negotiation tables

Tables below have been created by calling the routine the driver uses for synchronisation negotiation timing calculation and chip setting. The first table corresponds to Ultra chips 53875 and 53C860 with 80 MHz clock and 5 clock divisors. The second one has been calculated by setting the scsi clock to 40 Mhz and using 4 clock divisors and so applies to all NCR53C8XX chips in fast SCSI-2 mode.

Periods are in nano-seconds and speeds are in Mega-transfers per second. 1 Mega-transfers/second means 1 MB/s with 8 bits SCSI and 2 MB/s with Wide16 SCSI.

16.1 Synchronous timings for 53C895, 53C875 and 53C860 SCSI controllers

Negotiated			NCR settings		
Factor	Period	Speed	Period	Speed	
10	25	40.000	25	40.000	(53C895 only)
11	30.2	33.112	31.25	32.000	(53C895 only)
12	50	20.000	50	20.000	
13	52	19.230	62	16.000	
14	56	17.857	62	16.000	
15	60	16.666	62	16.000	
16	64	15.625	75	13.333	
17	68	14.705	75	13.333	
18	72	13.888	75	13.333	
19	76	13.157	87	11.428	
20	80	12.500	87	11.428	
21	84	11.904	87	11.428	
22	88	11.363	93	10.666	
23	92	10.869	93	10.666	
24	96	10.416	100	10.000	
25	100	10.000	100	10.000	
26	104	9.615	112	8.888	
27	108	9.259	112	8.888	
28	112	8.928	112	8.888	
29	116	8.620	125	8.000	
30	120	8.333	125	8.000	
31	124	8.064	125	8.000	
32	128	7.812	131	7.619	
33	132	7.575	150	6.666	
34	136	7.352	150	6.666	
35	140	7.142	150	6.666	
36	144	6.944	150	6.666	
37	148	6.756	150	6.666	
38	152	6.578	175	5.714	
39	156	6.410	175	5.714	
40	160	6.250	175	5.714	
41	164	6.097	175	5.714	
42	168	5.952	175	5.714	
43	172	5.813	175	5.714	
44	176	5.681	187	5.333	
45	180	5.555	187	5.333	
46	184	5.434	187	5.333	

47	188	5.319	200	5.000	
48	192	5.208	200	5.000	
49	196	5.102	200	5.000	

16.2 Synchronous timings for fast SCSI-2 53C8XX controllers

Negotiated			NCR settings	
Factor	Period	Speed	Period	Speed
25	100	10.000	100	10.000
26	104	9.615	125	8.000
27	108	9.259	125	8.000
28	112	8.928	125	8.000
29	116	8.620	125	8.000
30	120	8.333	125	8.000
31	124	8.064	125	8.000
32	128	7.812	131	7.619
33	132	7.575	150	6.666
34	136	7.352	150	6.666
35	140	7.142	150	6.666
36	144	6.944	150	6.666
37	148	6.756	150	6.666
38	152	6.578	175	5.714
39	156	6.410	175	5.714
40	160	6.250	175	5.714
41	164	6.097	175	5.714
42	168	5.952	175	5.714
43	172	5.813	175	5.714
44	176	5.681	187	5.333
45	180	5.555	187	5.333
46	184	5.434	187	5.333
47	188	5.319	200	5.000
48	192	5.208	200	5.000
49	196	5.102	200	5.000

17. Serial NVRAM

(added by Richard Waltham: dormouse@farsrobt.demon.co.uk)

17.1 Features

Enabling serial NVRAM support enables detection of the serial NVRAM included on Symbios and some Symbios compatible host adaptors, and Tekram boards. The serial NVRAM is used by Symbios and Tekram to hold set up parameters for the host adaptor and its attached drives.

The Symbios NVRAM also holds data on the boot order of host adaptors in a system with more than one host adaptor. This enables the order of scanning the cards for drives to be changed from the default used during host adaptor detection.

This can be done to a limited extent at the moment using "reverse probe" but this only changes the order of detection of different types of cards. The NVRAM boot order settings can do this as well as change the order the same types of cards are scanned in, something "reverse probe" cannot do.

Tekram boards using Symbios chips, DC390W/F/U, which have NVRAM are detected and this is used to distinguish between Symbios compatible and Tekram host adaptors. This is used to disable the Symbios compatible "diff" setting incorrectly set on Tekram boards if the CONFIG_SCSI_53C8XX_SYMBIOS_COMPAT configuration parameter is set enabling both Symbios and Tekram boards to be used together with the Symbios cards using all their features, including "diff" support. ("led pin" support for Symbios compatible cards can remain enabled when using Tekram cards. It does nothing useful for Tekram host adaptors but does not cause problems either.)

17.2 Symbios NVRAM layout

typical data at NVRAM address 0x100 (53c810a NVRAM):

```
00 00
64 01
8e 0b

00 30 00 00 00 00 07 00 00 00 00 00 00 00 07 04 10 04 00 00

04 00 0f 00 00 10 00 50 00 00 01 00 00 62
```

```
04 00 03 00 00 10 00 58 00 00 01 00 00 63
04 00 01 00 00 10 00 48 00 00 01 00 00 61
00 00 00 00 00 00 00 00 00 00 00 00 00 00

0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00

0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00

00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00

fe fe
00 00
00 00
```

NVRAM layout details

NVRAM Address	
0x000-0x0ff	not used
0x100-0x26f	initialised data
0x270-0x7ff	not used

general layout:

```
header  -   6 bytes,
data    - 356 bytes (checksum is byte sum of this data)
trailer -   6 bytes
          ---
total     368 bytes
```

data area layout:

```
controller set up  -  20 bytes
boot configuration -  56 bytes (4x14 bytes)
device set up      - 128 bytes (16x8 bytes)
unused (spare?)    - 152 bytes (19x8 bytes)
                     ---
total                356 bytes
```

header:

```
00 00   - ?? start marker
64 01   - byte count (lsb/msb excludes header/trailer)
8e 0b   - checksum (lsb/msb excludes header/trailer)
```

controller set up:

```
00 30 00 00 00 00 07 00 00 00 00 00 00 00 07 04 10 04 00 00
```

```
                |     |           |     |
                |     |           |      -- host ID
                |     |           |
                |     |            --Removable Media Support
                |     |               0x00 = none
                |     |               0x01 = Bootable Device
                |     |               0x02 = All with Media
                |     |
                |      --flag bits 2
                |        0x00000001= scan order hi->low
                |            (default 0x00 - scan low->hi)
                    --flag bits 1
                    0x00000001 scam enable
                    0x00000010 parity enable
                    0x00000100 verbose boot msgs
```

remaining bytes unknown - they do not appear to change in my current set up for any of the controllers.

default set up is identical for 53c810a and 53c875 NVRAM (Removable Media added Symbios BIOS version 4.09)

boot configuration

boot order set by order of the devices in this table:

```
04 00 0f 00 00 10 00 50 00 00 01 00 00 62 -- 1st controller
04 00 03 00 00 10 00 58 00 00 01 00 00 63    2nd controller
04 00 01 00 00 10 00 48 00 00 01 00 00 61    3rd controller
00 00 00 00 00 00 00 00 00 00 00 00 00 00    4th controller
    |  |  |  |     |        |     |  |
    |  |  |  |     |        |      ---- PCI io port adr
    |  |  |  |     |         --0x01 init/scan at boot time
    |  |  |  |      --PCI device/function number (0xdddddfff)
    |  |   -----  ?? PCI vendor ID (lsb/msb)
        ----PCI device ID (lsb/msb)

?? use of this data is a guess but seems reasonable
```

remaining bytes unknown - they do not appear to change in my current set up

default set up is identical for 53c810a and 53c875 NVRAM

device set up (up to 16 devices - includes controller):

```
0f 00 08 08 64 00 0a 00 - id 0
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00

0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00
0f 00 08 08 64 00 0a 00 - id 15
|     |  |  |     |  |
|     |  |  |      ----timeout (lsb/msb)
|     |  |   --synch period (0x?? 40 Mtrans/sec- fast 40) (probably 0x28)
|     |  |                  (0x30 20 Mtrans/sec- fast 20)
|     |  |                  (0x64 10 Mtrans/sec- fast )
|     |  |                  (0xc8  5 Mtrans/sec)
|     |  |                  (0x00  asynchronous)
|     |   -- ?? max sync offset (0x08 in NVRAM on 53c810a)
|     |                         (0x10 in NVRAM on 53c875)
|      --device bus width (0x08 narrow)
|                         (0x10 16 bit wide)
--flag bits
    0x00000001 - disconnect enabled
    0x00000010 - scan at boot time
    0x00000100 - scan luns
    0x00001000 - queue tags enabled
```

remaining bytes unknown - they do not appear to change in my current set up

?? use of this data is a guess but seems reasonable (but it could be max bus width)

default set up for 53c810a NVRAM default set up for 53c875 NVRAM

- bus width - 0x10
- sync offset ? - 0x10
- sync period - 0x30

?? spare device space (32 bit bus ??)

```
00 00 00 00 00 00 00 00  (19x8bytes)
.
.
00 00 00 00 00 00 00 00
```

default set up is identical for 53c810a and 53c875 NVRAM

trailer:

```
fe fe   - ? end marker ?
00 00
00 00
```

default set up is identical for 53c810a and 53c875 NVRAM

17.3 Tekram NVRAM layout

nvram 64x16 (1024 bit)

Drive settings:

```
Drive ID 0-15 (addr 0x0yyyy0 = device setup, yyyy = ID)
            (addr 0x0yyyy1 = 0x0000)

    x x x x  x x x x  x x x x  x x x x
             | | |      | |  | | | |
             | | |      | |  | | |  ----- parity check   0 - off
             | | |      | |  | | |                       1 - on
             | | |      | |  | | |
             | | |      | |  | |  ------- sync neg       0 - off
             | | |      | |  | |                         1 - on
             | | |      | |  | |
             | | |      | |  |  --------- disconnect     0 - off
             | | |      | |  |                           1 - on
             | | |      | |  |
             | | |      | |   ----------- start cmd      0 - off
             | | |      | |                             1 - on
             | | |      | |
             | | |      |  -------------- tagged cmds    0 - off
             | | |      |                                1 - on
             | | |      |
             | | |       ---------------- wide neg       0 - off
             | | |                                       1 - on
             | | |
                 --------------------------- sync rate      0 - 10.0 Mtrans/sec
                                                           1 -  8.0
                                                           2 -  6.6
                                                           3 -  5.7
                                                           4 -  5.0
                                                           5 -  4.0
                                                           6 -  3.0
                                                           7 -  2.0
                                                           7 -  2.0
                                                           8 - 20.0
                                                           9 - 16.7
                                                           a - 13.9
                                                           b - 11.9
```

Global settings

Host flags 0 (addr 0x100000, 32):

```
x x x x  x x x x  x x x x  x x x x
| | | |  | | | |           | | | |
| | | |  | | | |            ----------- host ID    0x00 - 0x0f
| | | |  | | | |
| | | |  | | |  ----------------------- support for    0 - off
| | | |  | | |                          > 2 drives     1 - on
| | | |  | | |
| | | |  | |  ------------------------ support drives 0 - off
| | | |  | |                           > 1Gbytes      1 - on
| | | |  | |
| | | |  |  -------------------------- bus reset on   0 - off
| | | |  |                               power on     1 - on
```

```
| | | | |
| | | |   ----------------------------- active neg     0 - off
| | | |                                               1 - on
| | | |
| | |  -------------------------------- imm seek      0 - off
| | |                                                 1 - on
| | |
| |  --------------------------------- scan luns      0 - off
| |                                                   1 - on
| |
 ------------------------------------- removable      0 - disable
                                        as BIOS dev    1 - boot device
                                                       2 - all
```

Host flags 1 (addr 0x100001, 33):

```
x x x x  x x x x  x x x x  x x x x
           | | |             | | |
           | | |              --------- boot delay     0 -   3 sec
           | | |                                     1 -   5
           | | |                                     2 -  10
           | | |                                     3 -  20
           | | |                                     4 -  30
           | | |                                     5 -  60
           | | |                                     6 - 120
           | | |
            --------------------------- max tag cmds   0 -  2
                                                        1 -  4
                                                        2 -  8
                                                        3 - 16
                                                        4 - 32
```

Host flags 2 (addr 0x100010, 34):

```
x x x x  x x x x  x x x x  x x x x
                                |
                                 ----- F2/F6 enable   0 - off ???
                                                      1 - on  ???
```

checksum (addr 0x111111)

checksum = 0x1234 - (sum addr 0-63)

---

default nvram data:

```
0x0037 0x0000 0x0037 0x0000 0x0037 0x0000 0x0037 0x0000
0x0037 0x0000 0x0037 0x0000 0x0037 0x0000 0x0037 0x0000
0x0037 0x0000 0x0037 0x0000 0x0037 0x0000 0x0037 0x0000
0x0037 0x0000 0x0037 0x0000 0x0037 0x0000 0x0037 0x0000

0x0f07 0x0400 0x0001 0x0000 0x0000 0x0000 0x0000 0x0000
0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0xfbbc
```

18. Support for Big Endian

The PCI local bus has been primarily designed for x86 architecture. As a consequence, PCI devices generally expect DWORDS using little endian byte ordering.

18.1 Big Endian CPU

In order to support NCR chips on a Big Endian architecture the driver has to perform byte reordering each time it is needed. This feature has been added to the driver by Cort <cort@cs.nmt.edu> and is available in driver version 2.5 and later ones. For the moment Big Endian support has only been tested on Linux/PPC (PowerPC).

18.2 NCR chip in Big Endian mode of operations

It can be read in SYMBIOS documentation that some chips support a special Big Endian mode, on paper: 53C815, 53C825A, 53C875, 53C875N, 53C895. This mode of operations is not software-selectable, but needs pin named BigLit to be pulled-up. Using this mode, most of byte reorderings should be avoided when the driver is running on a Big Endian CPU. Driver version 2.5 is also, in theory, ready for this feature.