

# [RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-11/msg05993.html>

---

- *From:* Adrian Bunk <[bunk@xxxxxxxx](mailto:bunk@xxxxxxxx)>
  - *Date:* Mon, 20 Nov 2006 03:23:38 +0100
- 

The SCSI\_SEAGATE driver has:

- already been marked as BROKEN in 2.6.0 three years ago and
- is still marked as BROKEN.

Drivers that had been marked as BROKEN for such a long time seem to be unlikely to be revived in the foreseeable future.

But if anyone wants to ever revive this driver, the code is still present in the older kernel releases.

Signed-off-by: Adrian Bunk <[bunk@xxxxxxxx](mailto:bunk@xxxxxxxx)>

---

drivers/scsi/Kconfig | 13  
drivers/scsi/Makefile | 3  
drivers/scsi/seagate.c | 1666 -----  
3 files changed, 1682 deletions(-)

--- linux-2.6.19-rc5-mm2/drivers/scsi/Kconfig.old 2006-11-19 17:54:09.000000000 +0100  
+++ linux-2.6.19-rc5-mm2/drivers/scsi/Kconfig 2006-11-19 17:54:23.000000000 +0100  
@@ -1277,19 +1277,6 @@

This lpfc driver supports the Emulex LightPulse Family of Fibre Channel PCI host adapters.

```
-config SCSI_SEAGATE
- tristate "Seagate ST-02 and Future Domain TMC-8xx SCSI support"
- depends on X86 && ISA && SCSI && BROKEN
- ---help---
- These are 8-bit SCSI controllers; the ST-01 is also supported by
- this driver. It is explained in section 3.9 of the SCSI-HOWTO,
- available from <http://www.tldp.org/docs.html#howto>. If it
- doesn't work out of the box, you may have to change some macros at
- compiletime, which are described in <file:drivers/scsi/seagate.c>.
-
- To compile this driver as a module, choose M here: the
- module will be called seagate.
-
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

# definitely looks not 64bit safe:

config SCSI\_SIM710

tristate "Simple 53c710 SCSI support (Compaq, NCR machines)"

--- linux-2.6.19-rc5-mm2/drivers/scsi/Makefile.old 2006-11-19 17:54:34.000000000 +0100

+++ linux-2.6.19-rc5-mm2/drivers/scsi/Makefile 2006-11-19 17:57:41.000000000 +0100

@@ -16,7 +16,6 @@

CFLAGS\_aha152x.o = -DAHA152X\_STAT -DAUTOCONF

CFLAGS\_gdth.o = # -DDEBUG\_GDTH=2 -D\_\_SERIAL\_\_ -D\_\_COM2\_\_ -DGDTH\_STATISTICS

-CFLAGS\_seagate.o = -DARBITRATE -DPARITY -DSEAGATE\_USE\_ASM

subdir-\$(CONFIG\_PCMCIA) += pcmcia

@@ -88,8 +87,6 @@

obj-\$(CONFIG SCSI\_QLA\_ISCSI) += qla4xxx/

obj-\$(CONFIG SCSI\_LPFC) += lpfc/

obj-\$(CONFIG SCSI\_PAS16) += pas16.o

-obj-\$(CONFIG SCSI\_SEAGATE) += seagate.o

-obj-\$(CONFIG SCSI\_FD\_8xx) += seagate.o

obj-\$(CONFIG SCSI\_T128) += t128.o

obj-\$(CONFIG SCSI\_DM3191D) += dm3191d.o

obj-\$(CONFIG SCSI\_DTC3280) += dtc.o

--- linux-2.6.19-rc5-mm2/drivers/scsi/seagate.c 2006-11-14 18:33:18.000000000 +0100

+++ /dev/null 2006-09-19 00:45:31.000000000 +0200

@@ -1,1666 +0,0 @@

\_/\*

- \* seagate.c Copyright (C) 1992, 1993 Drew Eckhardt

- \* low level scsi driver for ST01/ST02, Future Domain TMC-885,

- \* TMC-950 by Drew Eckhardt <drew@xxxxxxxxxxxx>

- \*

- \* Note : TMC-880 boards don't work because they have two bits in

- \* the status register flipped, I'll fix this "RSN"

- \* [why do I have strong feeling that above message is from 1993? :-)

- \* pavel@xxxxxx]

- \*

- \* This card does all the I/O via memory mapped I/O, so there is no need

- \* to check or allocate a region of the I/O address space.

- \*/

-

\_/\* 1996 - to use new read{b,w,l}, write{b,w,l}, and phys\_to\_virt

- \* macros, replaced assembler routines with C. There's probably a

- \* performance hit, but I only have a cdrom and can't tell. Define

- \* SEAGATE\_USE\_ASM if you want the old assembler code -- SJT

- \*

- \* 1998-jul-29 - created DPRINTK macros and made it work under

- \* linux 2.1.112, simplified some #defines etc. <pavel@xxxxxx>

- \*

- \* Aug 2000 - aeb - deleted seagate\_st0x\_biosparam(). It would try to

- \* read the physical disk geometry, a bad mistake. Of course it doesn't

- \* matter much what geometry one invents, but on large disks it

- \* returned 256 (or more) heads, causing all kind of failures.

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

## [RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

- \* Of course this means that people might see a different geometry now,
- \* so boot parameters may be necessary in some cases.
- \*/
- 
- /\*
- \* Configuration :
- \* To use without BIOS -DOVERRIDE=base\_address -DCONTROLLER=FD or SEAGATE
- \* -DIRQ will override the default of 5.
- \* Note: You can now set these options from the kernel's "command line".
- \* The syntax is:
- \*
- \* st0x=ADDRESS,IRQ (for a Seagate controller)
- \* or:
- \* tmc8xx=ADDRESS,IRQ (for a TMC-8xx or TMC-950 controller)
- \* eg:
- \* tmc8xx=0xC8000,15
- \*
- \* will configure the driver for a TMC-8xx style controller using IRQ 15
- \* with a base address of 0xC8000.
- \*
- \* -DARBITRATE
- \* Will cause the host adapter to arbitrate for the
- \* bus for better SCSI-II compatibility, rather than just
- \* waiting for BUS FREE and then doing its thing. Should
- \* let us do one command per Lun when I integrate my
- \* reorganization changes into the distribution sources.
- \*
- \* -DDEBUG=65535
- \* Will activate debug code.
- \*
- \* -DFAST or -DFAST32
- \* Will use blind transfers where possible
- \*
- \* -DPARITY
- \* This will enable parity.
- \*
- \* -DSEAGATE\_USE\_ASM
- \* Will use older seagate assembly code. should be (very small amount)
- \* Faster.
- \*
- \* -DSLOW\_RATE=50
- \* Will allow compatibility with broken devices that don't
- \* handshake fast enough (ie, some CD ROM's) for the Seagate
- \* code.
- \*
- \* 50 is some number, It will let you specify a default
- \* transfer rate if handshaking isn't working correctly.
- \*
- \* -DOLDCNTDATASCEME There is a new sceme to set the CONTROL
- \* and DATA reigsters which complies more closely
- \* with the SCSI2 standard. This hopefully eliminates

## [RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- * the need to swap the order these registers are
- * 'messed' with. It makes the following two options
- * obsolete. To reenale the old sceme define this.
- *
- * The following to options are patches from the SCSI.HOWTO
- *
- * -DSWAPSTAT This will swap the definitions for STAT_MSG and STAT_CD.
- *
- * -DSWAPCNTDATA This will swap the order that seagate.c messes with
- * the CONTROL an DATA registers.
- */
-
-#include <linux/module.h>
-#include <linux/interrupt.h>
-#include <linux/spinlock.h>
-#include <linux/signal.h>
-#include <linux/string.h>
-#include <linux/proc_fs.h>
-#include <linux/init.h>
-#include <linux/blkdev.h>
-#include <linux/stat.h>
-#include <linux/delay.h>
-#include <linux/io.h>
-
-#include <asm/system.h>
-#include <asm/uaccess.h>
-
-#include <scsi/scsi_cmnd.h>
-#include <scsi/scsi_device.h>
-#include <scsi/scsi.h>
-
-#include <scsi/scsi_dbg.h>
-#include <scsi/scsi_host.h>
-
-#ifdef DEBUG
-#define DPRINTK( when, msg... ) do { if ( (DEBUG & (when)) == (when) ) printk( msg ); } while (0)
-#else
-#define DPRINTK( when, msg... ) do { } while (0)
-#endif
-#define DANY( msg... ) DPRINTK( 0xffff, msg );
-
-#ifndef IRQ
-#define IRQ 5
-#endif
-
-#ifdef FAST32
-#define FAST
-#endif
-
-#undef LINKED /* Linked commands are currently broken! */
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
-
-#if defined(OVERRIDE) && !defined(CONTROLLER)
-#error Please use -DCONTROLLER=SEAGATE or -DCONTROLLER=FD to override controller type
-#endif
-
-#ifndef __i386__
-#undef SEAGATE_USE_ASM
-#endif
-
-/*
- Thanks to Brian Antoine for the example code in his Messy-Loss ST-01
- driver, and Mitsugu Suzuki for information on the ST-01
- SCSI host.
-*/
-
-/*
- CONTROL defines
-*/
-
-#define CMD_RST 0x01
-#define CMD_SEL 0x02
-#define CMD_BSY 0x04
-#define CMD_ATTN 0x08
-#define CMD_START_ARB 0x10
-#define CMD_EN_PARITY 0x20
-#define CMD_INTR 0x40
-#define CMD_DRVR_ENABLE 0x80
-
-/*
- STATUS
-*/
-#ifdef SWAPSTAT
-#define STAT_MSG 0x08
-#define STAT_CD 0x02
-#else
-#define STAT_MSG 0x02
-#define STAT_CD 0x08
-#endif
-
-#define STAT_BSY 0x01
-#define STAT_IO 0x04
-#define STAT_REQ 0x10
-#define STAT_SEL 0x20
-#define STAT_PARITY 0x40
-#define STAT_ARB_CMPL 0x80
-
-/*
- REQUESTS
-*/
-
-#define REQ_MASK (STAT_CD | STAT_IO | STAT_MSG)
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```

-#define REQ_DATAOUT 0
-#define REQ_DATAIN STAT_IO
-#define REQ_CMDOUT STAT_CD
-#define REQ_STATIN (STAT_CD | STAT_IO)
-#define REQ_MSGOUT (STAT_MSG | STAT_CD)
-#define REQ_MSGIN (STAT_MSG | STAT_CD | STAT_IO)
-
-extern volatile int seagate_st0x_timeout;
-
-#ifdef PARITY
-#define BASE_CMD CMD_EN_PARITY
-#else
-#define BASE_CMD 0
-#endif
-
-/*
- Debugging code
-*/
-
-#define PHASE_BUS_FREE 1
-#define PHASE_ARBITRATION 2
-#define PHASE_SELECTION 4
-#define PHASE_DATAIN 8
-#define PHASE_DATAOUT 0x10
-#define PHASE_CMDOUT 0x20
-#define PHASE_MSGIN 0x40
-#define PHASE_MSGOUT 0x80
-#define PHASE_STATUSIN 0x100
-#define PHASE_ETC (PHASE_DATAIN | PHASE_DATAOUT | PHASE_CMDOUT | PHASE_MSGIN |
PHASE_MSGOUT | PHASE_STATUSIN)
-#define PRINT_COMMAND 0x200
-#define PHASE_EXIT 0x400
-#define PHASE_RESELECT 0x800
-#define DEBUG_FAST 0x1000
-#define DEBUG_SG 0x2000
-#define DEBUG_LINKED 0x4000
-#define DEBUG_BORKEN 0x8000
-
-/*
- * Control options – these are timeouts specified in .01 seconds.
- */
-
-/* 30, 20 work */
-#define ST0X_BUS_FREE_DELAY 25
-#define ST0X_SELECTION_DELAY 25
-
-#define SEAGATE 1 /* these determine the type of the controller */
-#define FD 2
-
-#define ST0X_ID_STR "Seagate ST-01/ST-02"
-#define FD_ID_STR "TMC-8XX/TMC-950"

```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
-
-static int internal_command (unsigned char target, unsigned char lun,
- const void *cmdnd,
- void *buff, int buflen, int reselect);
-
-static int incommand; /* set if arbitration has finished
- and we are in some command phase. */
-
-static unsigned int base_address = 0; /* Where the card ROM starts, used to
- calculate memory mapped register
- location. */
-
-static void __iomem *st0x_cr_sr; /* control register write, status
- register read. 256 bytes in
- length.
- Read is status of SCSI BUS, as per
- STAT masks. */
-
-static void __iomem *st0x_dr; /* data register, read write 256
- bytes in length. */
-
-static volatile int st0x_aborted = 0; /* set when we are aborted, ie by a
- time out, etc. */
-
-static unsigned char controller_type = 0; /* set to SEAGATE for ST0x
- boards or FD for TMC-8xx
- boards */
-static int irq = IRQ;
-
-module_param(base_address, uint, 0);
-module_param(controller_type, byte, 0);
-module_param(irq, int, 0);
-MODULE_LICENSE("GPL");
-
-#define retcode(result) (((result) << 16) | (message << 8) | status)
-#define STATUS ((u8) readb(st0x_cr_sr))
-#define DATA ((u8) readb(st0x_dr))
-#define WRITE_CONTROL(d) { writeb((d), st0x_cr_sr); }
-#define WRITE_DATA(d) { writeb((d), st0x_dr); }
-
-#ifndef OVERRIDE
-static unsigned int seagate_bases[] = {
- 0xc8000, 0xca000, 0xcc000,
- 0xce000, 0xdc000, 0xde000
-};
-
-typedef struct {
- const unsigned char *signature;
- unsigned offset;
- unsigned length;

```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- unsigned char type;
- } Signature;
-
-static Signature __initdata signatures[] = {
- { "ST01 v1.7 (C) Copyright 1987 Seagate", 15, 37, SEAGATE},
- { "SCSI BIOS 2.00 (C) Copyright 1987 Seagate", 15, 40, SEAGATE},
-
- /*
- * The following two lines are NOT mistakes. One detects ROM revision
- * 3.0.0, the other 3.2. Since seagate has only one type of SCSI adapter,
- * and this is not going to change, the "SEAGATE" and "SCSI" together
- * are probably "good enough"
- */
-
- { "SEAGATE SCSI BIOS ", 16, 17, SEAGATE},
- { "SEAGATE SCSI BIOS ", 17, 17, SEAGATE},
-
- /*
- * However, future domain makes several incompatible SCSI boards, so specific
- * signatures must be used.
- */
-
- { "FUTURE DOMAIN CORP. (C) 1986-1989 V5.0C2/14/89", 5, 46, FD},
- { "FUTURE DOMAIN CORP. (C) 1986-1989 V6.0A7/28/89", 5, 46, FD},
- { "FUTURE DOMAIN CORP. (C) 1986-1990 V6.0105/31/90", 5, 47, FD},
- { "FUTURE DOMAIN CORP. (C) 1986-1990 V6.0209/18/90", 5, 47, FD},
- { "FUTURE DOMAIN CORP. (C) 1986-1990 V7.009/18/90", 5, 46, FD},
- { "FUTURE DOMAIN CORP. (C) 1992 V8.00.004/02/92", 5, 44, FD},
- { "IBM F1 BIOS V1.1004/30/92", 5, 25, FD},
- { "FUTURE DOMAIN TMC-950", 5, 21, FD},
- /* Added for 2.2.16 by Matthias_Heidbrink@xxxxxxxxxx */
- { "IBM F1 V1.2009/22/93", 5, 25, FD},
- };
-
-#define NUM_SIGNATURES ARRAY_SIZE(signatures)
-#endif /* n OVERRIDE */
-
- /*
- * hostno stores the hostnumber, as told to us by the init routine.
- */
-
- static int hostno = -1;
- static void seagate_reconnect_intr (int, void *);
- static irqreturn_t do_seagate_reconnect_intr (int, void *);
- static int seagate_st0x_bus_reset(struct scsi_cmnd *);
-
-#ifdef FAST
- static int fast = 1;
-#else
-#define fast 0
-#endif
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
-
-#ifdef SLOW_RATE
-/*
- * Support for broken devices :
- * The Seagate board has a handshaking problem. Namely, a lack
- * thereof for slow devices. You can blast 600K/second through
- * it if you are polling for each byte, more if you do a blind
- * transfer. In the first case, with a fast device, REQ will
- * transition high-low or high-low-high before your loop restarts
- * and you'll have no problems. In the second case, the board
- * will insert wait states for up to 13.2 usecs for REQ to
- * transition low->high, and everything will work.
- *
- * However, there's nothing in the state machine that says
- * you *HAVE* to see a high-low-high set of transitions before
- * sending the next byte, and slow things like the Trantor CD ROMS
- * will break because of this.
- *
- * So, we need to slow things down, which isn't as simple as it
- * seems. We can't slow things down period, because then people
- * who don't recompile their kernels will shoot me for ruining
- * their performance. We need to do it on a case per case basis.
- *
- * The best for performance will be to, only for borken devices
- * (this is stored on a per-target basis in the scsi_devices array)
- *
- * Wait for a low->high transition before continuing with that
- * transfer. If we timeout, continue anyways. We don't need
- * a long timeout, because REQ should only be asserted until the
- * corresponding ACK is received and processed.
- *
- * Note that we can't use the system timer for this, because of
- * resolution, and we *really* can't use the timer chip since
- * gettimeofday() and the beeper routines use that. So,
- * the best thing for us to do will be to calibrate a timing
- * loop in the initialization code using the timer chip before
- * gettimeofday() can screw with it.
- *
- * FIXME: this is broken (not borken :-). Empty loop costs less than
- * loop with ISA access in it! -- pavel@xxxxxx
- */
-
-
-static int borken_calibration = 0;
-
-
-static void __init borken_init (void)
- {
- register int count = 0, start = jiffies + 1, stop = start + 25;
-
- /* FIXME: There may be a better approach, this is a straight port for
- now */
- preempt_disable();
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- while (time_before (jiffies, start))
- cpu_relax();
- for (; time_before (jiffies, stop); ++count)
- cpu_relax();
- preempt_enable();
-
-/*
- * Ok, we now have a count for .25 seconds. Convert to a
- * count per second and divide by transfer rate in K. */
-
- borken_calibration = (count * 4) / (SLOW_RATE * 1024);
-
- if (borken_calibration < 1)
- borken_calibration = 1;
- }
-
- static inline void borken_wait (void)
- {
- register int count;
-
- for (count = borken_calibration; count && (STATUS & STAT_REQ); --count)
- cpu_relax();
-
- #if (DEBUG & DEBUG_BORKEN)
- if (count)
- printk ("scsi%d : borken timeout\n", hostno);
- #endif
- }
-
- #endif /* def SLOW_RATE */
-
- /* These beasts only live on ISA, and ISA means 8MHz. Each ULOOP()
- * contains at least one ISA access, which takes more than 0.125
- * usec. So if we loop 8 times time in usec, we are safe.
- */
-
- #define ULOOP( i ) for (clock = i*8;;)
- #define TIMEOUT (!(clock--))
-
- int __init seagate_st0x_detect (struct scsi_host_template * tpnt)
- {
- struct Scsi_Host *instance;
- int i, j;
- unsigned long cr, dr;
-
- tpnt->proc_name = "seagate";
- /*
- * First, we try for the manual override.
- */
- DANY ("Autodetecting ST0x / TMC-8xx\n");
-
- }
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- if (hostno != -1) {
- printk (KERN_ERR "seagate_st0x_detect() called twice?!\n");
- return 0;
- }
-
-/* If the user specified the controller type from the command line,
- controller_type will be non-zero, so don't try to detect one */
-
- if (!controller_type) {
-#ifdef OVERRIDE
- base_address = OVERRIDE;
- controller_type = CONTROLLER;
-
- DANY ("Base address overridden to %x, controller type is %s\n",
- base_address,
- controller_type == SEAGATE ? "SEAGATE" : "FD");
-#else /* OVERRIDE */
-/*
- * To detect this card, we simply look for the signature
- * from the BIOS version notice in all the possible locations
- * of the ROM's. This has a nice side effect of not trashing
- * any register locations that might be used by something else.
- *
- * XXX - note that we probably should be probing the address
- * space for the on-board RAM instead.
- */
-
- for (i = 0; i < ARRAY_SIZE(seagate_bases); ++i) {
- void __iomem *p = ioremap(seagate_bases[i], 0x2000);
- if (!p)
- continue;
- for (j = 0; j < NUM_SIGNATURES; ++j)
- if (check_signature(p + signatures[j].offset, signatures[j].signature, signatures[j].length)) {
- base_address = seagate_bases[i];
- controller_type = signatures[j].type;
- break;
- }
- iounmap(p);
- }
-#endif /* OVERRIDE */
- }
- /* (! controller_type) */
- tpnt->this_id = (controller_type == SEAGATE) ? 7 : 6;
- tpnt->name = (controller_type == SEAGATE) ? ST0X_ID_STR : FD_ID_STR;
-
- if (!base_address) {
- printk(KERN_INFO "seagate: ST0x/TMC-8xx not detected.\n");
- return 0;
- }
-
- cr = base_address + (controller_type == SEAGATE ? 0x1a00 : 0x1c00);
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- dr = cr + 0x200;
- st0x_cr_sr = ioremap(cr, 0x100);
- st0x_dr = ioremap(dr, 0x100);
-
- DANY("%s detected. Base address = %x, cr = %x, dr = %x\n",
- tpnt->name, base_address, cr, dr);
-
- /*
- * At all times, we will use IRQ 5. Should also check for IRQ3
- * if we lose our first interrupt.
- */
- instance = scsi_register (tpnt, 0);
- if (instance == NULL)
- return 0;
-
- hostno = instance->host_no;
- if (request_irq (irq, do_seagate_reconnect_intr, IRQF_DISABLED, (controller_type == SEAGATE) ?
"seagate" : "tmc-8xx", instance)) {
- printk(KERN_ERR "scsi%d : unable to allocate IRQ%d\n", hostno, irq);
- return 0;
- }
- instance->irq = irq;
- instance->io_port = base_address;
-#ifdef SLOW_RATE
- printk(KERN_INFO "Calibrating borken timer... ");
- borken_init();
- printk(" %d cycles per transfer\n", borken_calibration);
-#endif
- printk (KERN_INFO "This is one second... ");
- {
- int clock;
- ULOOP (1 * 1000 * 1000) {
- STATUS;
- if (TIMEOUT)
- break;
- }
- }
-
- printk ("done, %s options:"
-#ifdef ARBITRATE
- " ARBITRATE"
-#endif
-#ifdef DEBUG
- " DEBUG"
-#endif
-#ifdef FAST
- " FAST"
-#ifdef FAST32
- "32"
-#endif
-#endif
-#endif
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```

-#ifdef LINKED
- " LINKED"
-#endif
-#ifdef PARITY
- " PARITY"
-#endif
-#ifdef SEAGATE_USE_ASM
- " SEAGATE_USE_ASM"
-#endif
-#ifdef SLOW_RATE
- " SLOW_RATE"
-#endif
-#ifdef SWAPSTAT
- " SWAPSTAT"
-#endif
-#ifdef SWAPCNTDATA
- " SWAPCNTDATA"
-#endif
- "\n", tpnt->name);
- return 1;
-}
-
-static const char *seagate_st0x_info (struct Scsi_Host *shpnt)
-{
- static char buffer[64];
-
- snprintf(buffer, 64, "%s at irq %d, address 0x%05X",
- (controller_type == SEAGATE) ? ST0X_ID_STR : FD_ID_STR,
- irq, base_address);
- return buffer;
-}
-
-/*
- * These are our saved pointers for the outstanding command that is
- * waiting for a reconnect
- */
-
-static unsigned char current_target, current_lun;
-static unsigned char *current_cmnd, *current_data;
-static int current_nobuffs;
-static struct scatterlist *current_buffer;
-static int current_bufflen;
-
-#ifdef LINKED
-/*
- * linked_connected indicates whether or not we are currently connected to
- * linked_target, linked_lun and in an INFORMATION TRANSFER phase,
- * using linked commands.
- */
-
-static int linked_connected = 0;
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
-static unsigned char linked_target, linked_lun;
-#endif
-
-static void (*done_fn) (struct scsi_cmnd *) = NULL;
-static struct scsi_cmnd *SCint = NULL;
-
-/*
- * These control whether or not disconnect / reconnect will be attempted,
- * or are being attempted.
- */
-
-#define NO_RECONNECT 0
-#define RECONNECT_NOW 1
-#define CAN_RECONNECT 2
-
-/*
- * LINKED_RIGHT indicates that we are currently connected to the correct target
- * for this command, LINKED_WRONG indicates that we are connected to the wrong
- * target. Note that these imply CAN_RECONNECT and require defined(LINKED).
- */
-
-#define LINKED_RIGHT 3
-#define LINKED_WRONG 4
-
-/*
- * This determines if we are expecting to reconnect or not.
- */
-
-static int should_reconnect = 0;
-
-/*
- * The seagate_reconnect_intr routine is called when a target reselects the
- * host adapter. This occurs on the interrupt triggered by the target
- * asserting SEL.
- */
-
-static irqreturn_t do_seagate_reconnect_intr(int irq, void *dev_id)
-{
- unsigned long flags;
- struct Scsi_Host *dev = dev_id;
-
- spin_lock_irqsave (dev->host_lock, flags);
- seagate_reconnect_intr (irq, dev_id);
- spin_unlock_irqrestore (dev->host_lock, flags);
- return IRQ_HANDLED;
-}
-
-static void seagate_reconnect_intr (int irq, void *dev_id)
-{
- int temp;
- struct scsi_cmnd *SCTmp;
```



[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- current_target = SCpnt->device->id;
- current_lun = SCpnt->device->lun;
- current_cmnd = SCpnt->cmnd;
- current_data = (unsigned char *) SCpnt->request_buffer;
- current_bufflen = SCpnt->request_bufflen;
- SCint = SCpnt;
- if (recursion_depth)
- return 1;
- recursion_depth++;
- do {
-#ifdef LINKED
- /*
- * Set linked command bit in control field of SCSI command.
- */
-
- current_cmnd[SCpnt->cmd_len] |= 0x01;
- if (linked_connected) {
- DPRINTK (DEBUG_LINKED, "scsi%d : using linked commands, current I_T_L nexus is ", hostno);
- if (linked_target == current_target && linked_lun == current_lun)
- {
- DPRINTK(DEBUG_LINKED, "correct\n");
- reconnect = LINKED_RIGHT;
- } else {
- DPRINTK(DEBUG_LINKED, "incorrect\n");
- reconnect = LINKED_WRONG;
- }
- } else
-#endif /* LINKED */
- reconnect = CAN_RECONNECT;
-
- result = internal_command(SCint->device->id, SCint->device->lun, SCint->cmnd,
- SCint->request_buffer, SCint->request_bufflen, reconnect);
- if (msg_byte(result) == DISCONNECT)
- break;
- SCtmp = SCint;
- SCint = NULL;
- SCtmp->result = result;
- done_fn(SCtmp);
- }
- while (SCint);
- recursion_depth--;
- return 0;
-}
-
-static int internal_command (unsigned char target, unsigned char lun,
- const void *cmnd, void *buff, int bufflen, int reselect)
-{
- unsigned char *data = NULL;
- struct scatterlist *buffer = NULL;
- int clock, temp, nobuffs = 0, done = 0, len = 0;
-#ifdef DEBUG
```

## [RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- int transfered = 0, phase = 0, newphase;
-#endif
- register unsigned char status_read;
- unsigned char tmp_data, tmp_control, status = 0, message = 0;
- unsigned transfersize = 0, underflow = 0;
-#ifdef SLOW_RATE
- int borken = (int) SCint->device->borken; /* Does the current target require
- Very Slow I/O ? */
-#endif
-
-
- incommand = 0;
- st0x_aborted = 0;
-
-#if (DEBUG & PRINT_COMMAND)
- printk("scsi%d : target = %d, command = ", hostno, target);
- __scsi_print_command((unsigned char *) cmnd);
-#endif
-
-#if (DEBUG & PHASE_RESELECT)
- switch (reselect) {
- case RECONNECT_NOW:
- printk("scsi%d : reconnecting\n", hostno);
- break;
-#ifdef LINKED
- case LINKED_RIGHT:
- printk("scsi%d : connected, can reconnect\n", hostno);
- break;
- case LINKED_WRONG:
- printk("scsi%d : connected to wrong target, can reconnect\n",
- hostno);
- break;
-#endif
- case CAN_RECONNECT:
- printk("scsi%d : allowed to reconnect\n", hostno);
- break;
- default:
- printk("scsi%d : not allowed to reconnect\n", hostno);
- }
-#endif
-
- if (target == (controller_type == SEAGATE ? 7 : 6))
- return DID_BAD_TARGET;
-
- /*
- * We work it differently depending on if this is "the first time,"
- * or a reconnect. If this is a reselect phase, then SEL will
- * be asserted, and we must skip selection / arbitration phases.
- */
-
- switch (reselect) {
- case RECONNECT_NOW:
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- DPRINTK (PHASE_RESELECT, "scsi%d : phase RESELECT \n", hostno);
- /*
- * At this point, we should find the logical or of our ID
- * and the original target's ID on the BUS, with BSY, SEL,
- * and I/O signals asserted.
- *
- * After ARBITRATION phase is completed, only SEL, BSY,
- * and the target ID are asserted. A valid initiator ID
- * is not on the bus until IO is asserted, so we must wait
- * for that.
- */
- ULOOP (100 * 1000) {
- temp = STATUS;
- if ((temp & STAT_IO) && !(temp & STAT_BSY))
- break;
- if (TIMEOUT) {
- DPRINTK (PHASE_RESELECT, "scsi%d : RESELECT timed out while waiting for IO .\n", hostno);
- return (DID_BAD_INTR << 16);
- }
- }
-
- /*
- * After I/O is asserted by the target, we can read our ID
- * and its ID off of the BUS.
- */
-
- if (!(temp = DATA) & (controller_type == SEAGATE ? 0x80 : 0x40)) {
- DPRINTK (PHASE_RESELECT, "scsi%d : detected reconnect request to different target.\n\tData bus =
%d\n", hostno, temp);
- return (DID_BAD_INTR << 16);
- }
-
- if (!(temp & (1 << current_target))) {
- printk(KERN_WARNING "scsi%d : Unexpected reselect interrupt. Data bus = %d\n", hostno, temp);
- return (DID_BAD_INTR << 16);
- }
-
- buffer = current_buffer;
- cmdnd = current_cmdnd; /* WDE add */
- data = current_data; /* WDE add */
- len = current_bufflen; /* WDE add */
- nobuffs = current_nobuffs;
-
- /*
- * We have determined that we have been selected. At this
- * point, we must respond to the reselection by asserting
- * BSY ourselves
- */
-
-#if 1
- WRITE_CONTROL (BASE_CMD | CMD_DRVR_ENABLE | CMD_BSY);
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
–#else
– WRITE_CONTROL (BASE_CMD | CMD_BSY);
–#endif
–
– /*
– * The target will drop SEL, and raise BSY, at which time
– * we must drop BSY.
– */
–
– ULOOP (100 * 1000) {
– if (!(STATUS & STAT_SEL))
– break;
– if (TIMEOUT) {
– WRITE_CONTROL (BASE_CMD | CMD_INTR);
– DPRINTK (PHASE_RESELECT, "scsi%d : RESELECT timed out while waiting for SEL.\n", hostno);
– return (DID_BAD_INTR << 16);
– }
– }
– WRITE_CONTROL (BASE_CMD);
– /*
– * At this point, we have connected with the target
– * and can get on with our lives.
– */
– break;
– case CAN_RECONNECT:
–#ifdef LINKED
– /*
– * This is a bltcherous hack, just as bad as the Unix #!
– * interpreter stuff. If it turns out we are using the wrong
– * I_T_L nexus, the easiest way to deal with it is to go into
– * our INFORMATION TRANSFER PHASE code, send a ABORT
– * message on MESSAGE OUT phase, and then loop back to here.
– */
–connect_loop:
–#endif
– DPRINTK (PHASE_BUS_FREE, "scsi%d : phase = BUS FREE \n", hostno);
–
– /*
– * BUS FREE PHASE
– *
– * On entry, we make sure that the BUS is in a BUS FREE
– * phase, by insuring that both BSY and SEL are low for
– * at least one bus settle delay. Several reads help
– * eliminate wire glitch.
– */
–
–#ifndef ARBITRATE
–#error FIXME: this is broken: we may not use jiffies here – we are under cli(). It will hardlock.
– clock = jiffies + STOX_BUS_FREE_DELAY;
–
– while (((STATUS | STATUS | STATUS) & (STAT_BSY | STAT_SEL)) && (!st0x_aborted) &&
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
time_before (jiffies, clock))
- cpu_relax();
-
- if (time_after (jiffies, clock))
- return retcode (DID_BUS_BUSY);
- else if (st0x_aborted)
- return retcode (st0x_aborted);
-#endif
- DPRINTK (PHASE_SELECTION, "scsi%d : phase = SELECTION\n", hostno);
-
- clock = jiffies + ST0X_SELECTION_DELAY;
-
- /*
- * Arbitration/selection procedure :
- * 1. Disable drivers
- * 2. Write HOST adapter address bit
- * 3. Set start arbitration.
- * 4. We get either ARBITRATION COMPLETE or SELECT at this
- * point.
- * 5. OR our ID and targets on bus.
- * 6. Enable SCSI drivers and asserted SEL and ATTN
- */
-
-#ifdef ARBITRATE
- /* FIXME: verify host lock is always held here */
- WRITE_CONTROL(0);
- WRITE_DATA((controller_type == SEAGATE) ? 0x80 : 0x40);
- WRITE_CONTROL(CMD_START_ARB);
-
- ULOOP (ST0X_SELECTION_DELAY * 10000) {
- status_read = STATUS;
- if (status_read & STAT_ARB_CMPL)
- break;
- if (st0x_aborted) /* FIXME: What? We are going to do something even after abort? */
- break;
- if (TIMEOUT || (status_read & STAT_SEL)) {
- printk(KERN_WARNING "scsi%d : arbitration lost or timeout.\n", hostno);
- WRITE_CONTROL (BASE_CMD);
- return retcode (DID_NO_CONNECT);
- }
- }
- DPRINTK (PHASE_SELECTION, "scsi%d : arbitration complete\n", hostno);
-#endif
-
- /*
- * When the SCSI device decides that we're gawking at it,
- * it will respond by asserting BUSY on the bus.
- *
- * Note : the Seagate ST-01/02 product manual says that we
- * should twiddle the DATA register before the control
- * register. However, this does not work reliably so we do
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- * it the other way around.
- *
- * Probably could be a problem with arbitration too, we
- * really should try this with a SCSI protocol or logic
- * analyzer to see what is going on.
- */
- tmp_data = (unsigned char) ((1 << target) | (controller_type == SEAGATE ? 0x80 : 0x40));
- tmp_control = BASE_CMD | CMD_DRV_R_ENABLE | CMD_SEL | (reselect ? CMD_ATTN : 0);
-
- /* FIXME: verify host lock is always held here */
-#ifdef OLDCNTDATASCEME
-#ifdef SWAPCNTDATA
- WRITE_CONTROL (tmp_control);
- WRITE_DATA (tmp_data);
-#else
- WRITE_DATA (tmp_data);
- WRITE_CONTROL (tmp_control);
-#endif
-#else
- tmp_control ^= CMD_BSY; /* This is guesswork. What used to be in driver */
- WRITE_CONTROL (tmp_control); /* could never work: it sent data into control */
- WRITE_DATA (tmp_data); /* register and control info into data. Hopefully */
- tmp_control ^= CMD_BSY; /* fixed, but order of first two may be wrong. */
- WRITE_CONTROL (tmp_control); /* -- Pavel@xxxxxx */
-#endif
-
- ULOOP (250 * 1000) {
- if (st0x_aborted) {
- /*
- * If we have been aborted, and we have a
- * command in progress, IE the target
- * still has BSY asserted, then we will
- * reset the bus, and notify the midlevel
- * driver to expect sense.
- */
-
- WRITE_CONTROL (BASE_CMD);
- if (STATUS & STAT_BSY) {
- printk(KERN_WARNING "scsi%d : BST asserted after we've been aborted.\n", hostno);
- seagate_st0x_bus_reset(NULL);
- return retcode (DID_RESET);
- }
- return retcode (st0x_aborted);
- }
- if (STATUS & STAT_BSY)
- break;
- if (TIMEOUT) {
- DPRINTK (PHASE_SELECTION, "scsi%d : NO CONNECT with target %d, stat = %x\n", hostno, target,
STATUS);
- return retcode (DID_NO_CONNECT);
- }
}
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- }
-
- /* Establish current pointers. Take into account scatter / gather */
-
- if ((nobuffs = SCint->use_sg)) {
- #if (DEBUG & DEBUG_SG)
- {
- int i;
- printk("scsi%d : scatter gather requested, using %d buffers.\n", hostno, nobuffs);
- for (i = 0; i < nobuffs; ++i)
- printk("scsi%d : buffer %d address = %p length = %d\n",
- hostno, i,
- page_address(buffer[i].page) + buffer[i].offset,
- buffer[i].length);
- }
- #endif
-
- buffer = (struct scatterlist *) SCint->request_buffer;
- len = buffer->length;
- data = page_address(buffer->page) + buffer->offset;
- } else {
- DPRINTK (DEBUG_SG, "scsi%d : scatter gather not requested.\n", hostno);
- buffer = NULL;
- len = SCint->request_bufflen;
- data = (unsigned char *) SCint->request_buffer;
- }
-
- DPRINTK (PHASE_DATAIN | PHASE_DATAOUT, "scsi%d : len = %d\n",
- hostno, len);
-
- break;
- #ifdef LINKED
- case LINKED_RIGHT:
- break;
- case LINKED_WRONG:
- break;
- #endif
- } /* end of switch(reselect) */
-
- /*
- * There are several conditions under which we wish to send a message :
- * 1. When we are allowing disconnect / reconnect, and need to
- * establish the I_T_L nexus via an IDENTIFY with the DiscPriv bit
- * set.
- *
- * 2. When we are doing linked commands, are have the wrong I_T_L
- * nexus established and want to send an ABORT message.
- */
-
- /* GCC does not like an ifdef inside a macro, so do it the hard way. */
- #ifdef LINKED
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- WRITE_CONTROL (BASE_CMD | CMD_DRVR_ENABLE | (((reselect == CAN_RECONNECT)||
(reselect == LINKED_WRONG))? CMD_ATTN : 0));
-#else
- WRITE_CONTROL (BASE_CMD | CMD_DRVR_ENABLE | (((reselect == CAN_RECONNECT))?
CMD_ATTN : 0));
-#endif
-
- /*
- * INFORMATION TRANSFER PHASE
- *
- * The nasty looking read / write inline assembler loops we use for
- * DATAIN and DATAOUT phases are approximately 4-5 times as fast as
- * the 'C' versions - since we're moving 1024 bytes of data, this
- * really adds up.
- *
- * SJT: The nasty-looking assembler is gone, so it's slower.
- *
- */
-
- DPRINTK (PHASE_ETC, "scsi%d : phase = INFORMATION TRANSFER\n", hostno);
-
- incommand = 1;
- transfersize = SCint->transfersize;
- underflow = SCint->underflow;
-
- /*
- * Now, we poll the device for status information,
- * and handle any requests it makes. Note that since we are unsure
- * of how much data will be flowing across the system, etc and
- * cannot make reasonable timeouts, that we will instead have the
- * midlevel driver handle any timeouts that occur in this phase.
- */
-
- while (((status_read = STATUS) & STAT_BSY) && !st0x_aborted && !done) {
-#ifdef PARITY
- if (status_read & STAT_PARITY) {
- printk(KERN_ERR "scsi%d : got parity error\n", hostno);
- st0x_aborted = DID_PARITY;
- }
-#endif
- if (status_read & STAT_REQ) {
-#if ((DEBUG & PHASE_ETC) == PHASE_ETC)
- if ((newphase = (status_read & REQ_MASK)) != phase) {
- phase = newphase;
- switch (phase) {
- case REQ_DATAOUT:
- printk ("scsi%d : phase = DATA OUT\n", hostno);
- break;
- case REQ_DATAIN:
- printk ("scsi%d : phase = DATA IN\n", hostno);
- break;
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- case REQ_CMDOUT:
- printk
- ("scsi%d : phase = COMMAND OUT\n", hostno);
- break;
- case REQ_STATIN:
- printk ("scsi%d : phase = STATUS IN\n", hostno);
- break;
- case REQ_MSGOUT:
- printk
- ("scsi%d : phase = MESSAGE OUT\n", hostno);
- break;
- case REQ_MSGIN:
- printk ("scsi%d : phase = MESSAGE IN\n", hostno);
- break;
- default:
- printk ("scsi%d : phase = UNKNOWN\n", hostno);
- st0x_aborted = DID_ERROR;
- }
- }
-#endif
- switch (status_read & REQ_MASK) {
- case REQ_DATAOUT:
- /*
- * If we are in fast mode, then we simply splat
- * the data out in word-sized chunks as fast as
- * we can.
- */
-
-
- if (!len) {
-#if 0
- printk("scsi%d: underflow to target %d lun %d\n", hostno, target, lun);
- st0x_aborted = DID_ERROR;
- fast = 0;
-#endif
- break;
- }
-
- if (fast && transfersize
- && !(len % transfersize)
- && (len >= transfersize)
-#ifdef FAST32
- && !(transfersize % 4)
-#endif
- ) {
- DPRINTK (DEBUG_FAST,
- "scsi%d : FAST transfer, underflow = %d, transfersize = %d\n"
- " len = %d, data = %08x\n",
- hostno, SCint->underflow,
- SCint->transfersize, len,
- data);
-
- }
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- /* SJT: Start. Fast Write */
-#ifdef SEAGATE_USE_ASM
- __asm__ ("cld\n\t"
-#ifdef FAST32
- "shr $2, %%ecx\n\t"
- "1:\t"
- "lods\n\t"
- "movl %%eax, (%%edi)\n\t"
-#else
- "1:\t"
- "lods\n\t"
- "movb %%al, (%%edi)\n\t"
-#endif
- "loop 1b;"
- /* output */ :
- /* input */ : "D" (st0x_dr),
- "S"
- (data),
- "c" (SCint->transfersize)
- /* clobbered */
- : "eax", "ecx",
- "esi");
-#else /* SEAGATE_USE_ASM */
- memcpy_toio(st0x_dr, data, transfersize);
-#endif /* SEAGATE_USE_ASM */
- /* SJT: End */
- len -= transfersize;
- data += transfersize;
- DPRINTK (DEBUG_FAST, "scsi%d : FAST transfer complete len = %d data = %08x\n", hostno, len,
data);
- } else {
- /*
- * We loop as long as we are in a
- * data out phase, there is data to
- * send, and BSY is still active.
- */
-
- /* SJT: Start. Slow Write. */
-#ifdef SEAGATE_USE_ASM
-
- int __dummy_1, __dummy_2;
-
- /*
- * We loop as long as we are in a data out phase, there is data to send,
- * and BSY is still active.
- */
- /* Local variables : len = ecx , data = esi,
- st0x_cr_sr = ebx, st0x_dr = edi
- */
- __asm__ (
- /* Test for any data here at all. */
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- "orl %%ecx, %%ecx\n\t"
- "jz 2f\n\t" "cld\n\t"
-/* "movl st0x_cr_sr, %%ebx\n\t" */
-/* "movl st0x_dr, %%edi\n\t" */
- "1:\t"
- "movb (%%ebx), %%al\n\t"
- /* Test for BSY */
- "test $1, %%al\n\t"
- "jz 2f\n\t"
- /* Test for data out phase – STATUS & REQ_MASK should be
- REQ_DATAOUT, which is 0. */
- "test $0xe, %%al\n\t"
- "jnz 2f\n\t"
- /* Test for REQ */
- "test $0x10, %%al\n\t"
- "jz 1b\n\t"
- "lods b\n\t"
- "movb %%al, (%%edi)\n\t"
- "loop 1b\n\t" "2:\n\t"
- /* output */ :="S" (data), "c" (len),
- "=b"
- (__dummy_1),
- "=D" (__dummy_2)
-/* input */
- : "0" (data), "1" (len),
- "2" (st0x_cr_sr),
- "3" (st0x_dr)
-/* clobbered */
- : "eax");
-#else /* SEAGATE_USE_ASM */
- while (len) {
- unsigned char stat;
-
- stat = STATUS;
- if (!(stat & STAT_BSY)
- || ((stat & REQ_MASK) !=
- REQ_DATAOUT))
- break;
- if (stat & STAT_REQ) {
- WRITE_DATA (*data++);
- --len;
- }
- }
-#endif /* SEAGATE_USE_ASM */
-/* SJT: End. */
- }
-
- if (!len && nobuffs) {
- --nobuffs;
- ++buffer;
- len = buffer->length;
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- data = page_address(buffer->page) + buffer->offset;
- DPRINTK (DEBUG_SG,
- "scsi%d : next scatter-gather buffer len = %d address = %08x\n",
- hostno, len, data);
- }
- break;
-
- case REQ_DATAIN:
-#ifdef SLOW_RATE
- if (borken) {
-#if (DEBUG & (PHASE_DATAIN))
- transfered += len;
-#endif
- for (; len && (STATUS & (REQ_MASK | STAT_REQ)) == (REQ_DATAIN | STAT_REQ); --len) {
- *data++ = DATA;
- borken_wait();
- }
-#if (DEBUG & (PHASE_DATAIN))
- transfered -= len;
-#endif
- } else
-#endif
-
- if (fast && transfersize
- && !(len % transfersize)
- && (len >= transfersize)
-#ifdef FAST32
- && !(transfersize % 4)
-#endif
- ) {
- DPRINTK (DEBUG_FAST,
- "scsi%d : FAST transfer, underflow = %d, transfersize = %d\n"
- " len = %d, data = %08x\n",
- hostno, SCint->underflow,
- SCint->transfersize, len,
- data);
-
- /* SJT: Start. Fast Read */
-#ifdef SEAGATE_USE_ASM
- __asm__ ("cld\n\t"
-#ifdef FAST32
- "shr $2, %%ecx\n\t"
- "1:\t"
- "movl (%%esi), %%eax\n\t"
- "stosl\n\t"
-#else
- "1:\t"
- "movb (%%esi), %%al\n\t"
- "stosb\n\t"
-#endif
-#endif
- "loop 1b\n\t"
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```

- /* output */ :
- /* input */ : "S" (st0x_dr),
- "D"
- (data),
- "c" (SCint->transfersize)
-/* clobbered */
- : "eax", "ecx",
- "edi");
-#else /* SEAGATE_USE_ASM */
- memcpy_fromio(data, st0x_dr, len);
-#endif /* SEAGATE_USE_ASM */
-/* SJT: End */
- len -= transfersize;
- data += transfersize;
-#if (DEBUG & PHASE_DATAIN)
- printk ("scsi%d: transfered += %d\n", hostno, transfersize);
- transfered += transfersize;
-#endif
-
- DPRINTK (DEBUG_FAST, "scsi%d : FAST transfer complete len = %d data = %08x\n", hostno, len,
data);
- } else {
-
-#if (DEBUG & PHASE_DATAIN)
- printk ("scsi%d: transfered += %d\n", hostno, len);
- transfered += len; /* Assume we'll transfer it all, then
- subtract what we *didn't* transfer */
-#endif
-
-/*
- * We loop as long as we are in a data in phase, there is room to read,
- * and BSY is still active
- */
-
-/* SJT: Start. */
-#ifdef SEAGATE_USE_ASM
-
- int __dummy_3, __dummy_4;
-
-/* Dummy clobbering variables for the new gcc-2.95 */
-
-/*
- * We loop as long as we are in a data in phase, there is room to read,
- * and BSY is still active
- */
- /* Local variables : ecx = len, edi = data
- esi = st0x_cr_sr, ebx = st0x_dr */
- __asm__ (
- /* Test for room to read */
- "orl %%ecx, %%ecx\n\t"
- "jz 2f\n\t" "cld\n\t"

```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```

-/* "movl st0x_cr_sr, %%esi\n\t" */
-/* "movl st0x_dr, %%ebx\n\t" */
- "1:\t"
- "movb (%%esi), %%al\n\t"
-/* Test for BSY */
- "test $1, %%al\n\t"
- "jz 2f\n\t"
-/* Test for data in phase – STATUS & REQ_MASK should be REQ_DATAIN,
- = STAT_IO, which is 4. */
- "movb $0xe, %%ah\n\t"
- "andb %%al, %%ah\n\t"
- "cmpb $0x04, %%ah\n\t"
- "jne 2f\n\t"
-/* Test for REQ */
- "test $0x10, %%al\n\t"
- "jz 1b\n\t"
- "movb (%%ebx), %%al\n\t"
- "stosb\n\t"
- "loop 1b\n\t" "2:\n"
-/* output */ : "=D" (data), "=c" (len),
- "=S"
- (__dummy_3),
- "=b" (__dummy_4)
-/* input */
- : "0" (data), "1" (len),
- "2" (st0x_cr_sr),
- "3" (st0x_dr)
-/* clobbered */
- : "eax");
-#else /* SEAGATE_USE_ASM */
- while (len) {
- unsigned char stat;
-
- stat = STATUS;
- if (!(stat & STAT_BSY)
- || ((stat & REQ_MASK) !=
- REQ_DATAIN))
- break;
- if (stat & STAT_REQ) {
- *data++ = DATA;
- --len;
- }
- }
-#endif /* SEAGATE_USE_ASM */
-/* SJT: End. */
-#if (DEBUG & PHASE_DATAIN)
- printk ("scsi%d: transfered == %d\n", hostno, len);
- transfered -= len; /* Since we assumed all of Len got *
- transfered, correct our mistake */
-#endif
- }

```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
-
- if (!len && nobuffs) {
- --nobuffs;
- ++buffer;
- len = buffer->length;
- data = page_address(buffer->page) + buffer->offset;
- DPRINTK (DEBUG_SG, "scsi%d : next scatter-gather buffer len = %d address = %08x\n", hostno, len,
data);
- }
- break;
-
- case REQ_CMDOUT:
- while (((status_read = STATUS) & STAT_BSY) &&
- ((status_read & REQ_MASK) == REQ_CMDOUT))
- if (status_read & STAT_REQ) {
- WRITE_DATA (*(const unsigned char *) cmdnd);
- cmdnd = 1 + (const unsigned char *)cmdnd;
- #ifdef SLOW_RATE
- if (borken)
- borken_wait ();
- #endif
- }
- break;
-
- case REQ_STATIN:
- status = DATA;
- break;
-
- case REQ_MSGOUT:
- /*
- * We can only have sent a MSG OUT if we
- * requested to do this by raising ATTN.
- * So, we must drop ATTN.
- */
- WRITE_CONTROL (BASE_CMD | CMD_DRVR_ENABLE);
- /*
- * If we are reconnecting, then we must
- * send an IDENTIFY message in response
- * to MSGOUT.
- */
- switch (reselect) {
- case CAN_RECONNECT:
- WRITE_DATA (IDENTIFY (1, lun));
- DPRINTK (PHASE_RESELECT | PHASE_MSGOUT, "scsi%d : sent IDENTIFY message.\n", hostno);
- break;
- #ifdef LINKED
- case LINKED_WRONG:
- WRITE_DATA (ABORT);
- linked_connected = 0;
- reselect = CAN_RECONNECT;
- goto connect_loop;
- #endif
- }
```

## [RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- DPRINTK (PHASE_MSGOUT | DEBUG_LINKED, "scsi%d : sent ABORT message to cancel incorrect
I_T_L nexus.\n", hostno);
-#endif /* LINKED */
- DPRINTK (DEBUG_LINKED, "correct\n");
- default:
- WRITE_DATA (NOP);
- printk("scsi%d : target %d requested MSGOUT, sent NOP message.\n", hostno, target);
- }
- break;
-
- case REQ_MSGIN:
- switch (message = DATA) {
- case DISCONNECT:
- DANY("seagate: deciding to disconnect\n");
- should_reconnect = 1;
- current_data = data; /* WDE add */
- current_buffer = buffer;
- current_bufflen = len; /* WDE add */
- current_nobuffs = nobuffs;
-#ifdef LINKED
- linked_connected = 0;
-#endif
- done = 1;
- DPRINTK ((PHASE_RESELECT | PHASE_MSGIN), "scsi%d : disconnected.\n", hostno);
- break;
-
-#ifdef LINKED
- case LINKED_CMD_COMPLETE:
- case LINKED_FLG_CMD_COMPLETE:
-#endif
- case COMMAND_COMPLETE:
- /*
- * Note : we should check for underflow here.
- */
- DPRINTK(PHASE_MSGIN, "scsi%d : command complete.\n", hostno);
- done = 1;
- break;
- case ABORT:
- DPRINTK(PHASE_MSGIN, "scsi%d : abort message.\n", hostno);
- done = 1;
- break;
- case SAVE_POINTERS:
- current_buffer = buffer;
- current_bufflen = len; /* WDE add */
- current_data = data; /* WDE mod */
- current_nobuffs = nobuffs;
- DPRINTK (PHASE_MSGIN, "scsi%d : pointers saved.\n", hostno);
- break;
- case RESTORE_POINTERS:
- buffer = current_buffer;
- cmdnd = current_cmdnd;
```

## [RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- data = current_data; /* WDE mod */
- len = current_bufflen;
- nobuffs = current_nobuffs;
- DPRINTK(PHASE_MSGIN, "scsi%d : pointers restored.\n", hostno);
- break;
- default:
-
- /*
- * IDENTIFY distinguishes itself
- * from the other messages by
- * setting the high bit.
- *
- * Note : we need to handle at
- * least one outstanding command
- * per LUN, and need to hash the
- * SCSI command for that I_T_L
- * nexus based on the known ID
- * (at this point) and LUN.
- */
-
- if (message & 0x80) {
- DPRINTK (PHASE_MSGIN, "scsi%d : IDENTIFY message received from id %d, lun %d.\n", hostno,
target, message & 7);
- } else {
- /*
- * We should go into a
- * MESSAGE OUT phase, and
- * send a MESSAGE_REJECT
- * if we run into a message
- * that we don't like. The
- * seagate driver needs
- * some serious
- * restructuring first
- * though.
- */
- DPRINTK (PHASE_MSGIN, "scsi%d : unknown message %d from target %d.\n", hostno, message, target);
- }
- }
- break;
- default:
- printk(KERN_ERR "scsi%d : unknown phase.\n", hostno);
- st0x_aborted = DID_ERROR;
- } /* end of switch (status_read & REQ_MASK) */
-#ifdef SLOW_RATE
- /*
- * I really don't care to deal with borken devices in
- * each single byte transfer case (ie, message in,
- * message out, status), so I'll do the wait here if
- * necessary.
- */
- if(borken)
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- borken_wait();
-#endif
-
- } /* if(status_read & STAT_REQ) ends */
- } /* while(((status_read = STATUS)...)) ends */
-
- DPRINTK(PHASE_DATAIN | PHASE_DATAOUT | PHASE_EXIT, "scsi%d : Transferred %d bytes\n",
hostno, transferred);
-
-#if (DEBUG & PHASE_EXIT)
-#if 0 /* Doesn't work for scatter/gather */
- printk("Buffer : \n");
- for(i = 0; i < 20; ++i)
- printk("%02x ", ((unsigned char *) data)[i]); /* WDE mod */
- printk("\n");
-#endif
- printk("scsi%d : status = ", hostno);
- scsi_print_status(status);
- printk(" message = %02x\n", message);
-#endif
-
- /* We shouldn't reach this until *after* BSY has been deasserted */
-
-#ifdef LINKED
- else
- {
- /*
- * Fix the message byte so that unsuspecting high level drivers
- * don't puke when they see a LINKED COMMAND message in place of
- * the COMMAND COMPLETE they may be expecting. Shouldn't be
- * necessary, but it's better to be on the safe side.
- *
- * A non LINKED* message byte will indicate that the command
- * completed, and we are now disconnected.
- */
-
- switch (message) {
- case LINKED_CMD_COMPLETE:
- case LINKED_FLG_CMD_COMPLETE:
- message = COMMAND_COMPLETE;
- linked_target = current_target;
- linked_lun = current_lun;
- linked_connected = 1;
- DPRINTK (DEBUG_LINKED, "scsi%d : keeping I_T_L nexus established for linked command.\n",
hostno);
- /* We also will need to adjust status to accommodate intermediate
- conditions. */
- if ((status == INTERMEDIATE_GOOD) || (status == INTERMEDIATE_C_GOOD))
- status = GOOD;
- break;
- }
- }
-#endif
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- * We should also handle what are "normal" termination
- * messages here (ABORT, BUS_DEVICE_RESET?, and
- * COMMAND_COMPLETE individually, and flake if things
- * aren't right.
- */
- default:
- DPRINTK (DEBUG_LINKED, "scsi%d : closing I_T_L nexus.\n", hostno);
- linked_connected = 0;
- }
- }
-#endif /* LINKED */
-
- if (should_reconnect) {
- DPRINTK (PHASE_RESELECT, "scsi%d : exiting seagate_st0x_queue_command() with reconnect
enabled.\n", hostno);
- WRITE_CONTROL (BASE_CMD | CMD_INTR);
- } else
- WRITE_CONTROL (BASE_CMD);
-
- return retcode (st0x_aborted);
-} /* end of internal_command */
-
-static int seagate_st0x_abort(struct scsi_cmnd * SCpnt)
-{
- st0x_aborted = DID_ABORT;
- return SUCCESS;
-}
-
-#undef ULOOP
-#undef TIMEOUT
-
-/*
- * the seagate_st0x_reset function resets the SCSI bus
- *
- * May be called with SCpnt = NULL
- */
-
-static int seagate_st0x_bus_reset(struct scsi_cmnd * SCpnt)
-{
- /* No timeouts – this command is going to fail because it was reset. */
- DANY ("scsi%d: Reseting bus... ", hostno);
-
- /* assert RESET signal on SCSI bus. */
- WRITE_CONTROL (BASE_CMD | CMD_RST);
-
- mdelay (20);
-
- WRITE_CONTROL (BASE_CMD);
- st0x_aborted = DID_RESET;
-
- DANY ("done.\n");
```

[RFC: 2.6 patch] remove the broken SCSI\_SEAGATE driver

```
- return SUCCESS;
-}
-
-static int seagate_st0x_release(struct Scsi_Host *shost)
-{
- if (shost->irq)
- free_irq(shost->irq, shost);
- release_region(shost->io_port, shost->n_io_port);
- return 0;
-}
-
-static struct scsi_host_template driver_template = {
- .detect = seagate_st0x_detect,
- .release = seagate_st0x_release,
- .info = seagate_st0x_info,
- .queuecommand = seagate_st0x_queue_command,
- .eh_abort_handler = seagate_st0x_abort,
- .eh_bus_reset_handler = seagate_st0x_bus_reset,
- .can_queue = 1,
- .this_id = 7,
- .sg_tablesize = SG_ALL,
- .cmd_per_lun = 1,
- .use_clustering = DISABLE_CLUSTERING,
-};
-#include "scsi_module.c"
```

-  
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in  
the body of a message to majordomo@xxxxxxxxxxxxxxxxx  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
Please read the FAQ at <http://www.tux.org/lkml/>