[Announce] megaraid_mbox 2.20.4.3 patch

Source: http://linux.derkeiler.com/Mailing−Lists/Kernel/2005−01/7839.html

---

From: Ju, Seokmann (*sju_at_lsil.com*)
Date: 01/26/05

```
To: "'linux−kernel@vger.kernel.org'" <linux−kernel@vger.kernel.org>, "'linux−scsi@vger.kernel.org
Date:   Tue, 25 Jan 2005 19:52:42 −0500
```

Hello,

Here is a patch for megaraid_mbox 2.20.4.3 and megaraid_mm 2.20.2.5.
The patch includes following changes/fixes
− sysfs support for drive addition/removal
− Tape drive timeout issue
− Made some code static

I am attaching and inlining the patch.

Thank you.

Seokmann
LSI Logic

```
−−−
diff −Naur linux_bk/Documentation/scsi/ChangeLog.megaraid
linux_bk.new/Documentation/scsi/ChangeLog.megaraid
−−− linux_bk/Documentation/scsi/ChangeLog.megaraid    2005−01−25
18:12:46.000000000 −0500
+++ linux_bk.new/Documentation/scsi/ChangeLog.megaraid      2005−01−25
18:21:48.525354048 −0500
@@ −1,3 +1,77 @@
+Release Date     : Fri Jan 21 00:01:03 EST 2005 − Atul Mukker
<atulm@lsil.com>
+Current Version  : 2.20.4.3 (scsi module), 2.20.2.5 (cmm module)
+Older Version    : 2.20.4.2 (scsi module), 2.20.2.4 (cmm module)
+
+i.   Remove driver ioctl for logical drive to scsi address translation and
+     replace with the sysfs attribute. To remove drives and change
+     capacity, application shall now use the device attribute to get the
+     logical drive number for a scsi device. For adding newly created
+     logical drives, class device attribute would be required to uniquely
+     identify each controller.
+           − Atul Mukker <atulm@lsil.com>
+
+     "James, I've been thinking about this a little more, and you may be on
+     to something here. Let each driver add files as such:"
+
+           − Matt Domsch <Matt_Domsch@dell.com>, 12.15.2004
+           linux−scsi mailing list
+
+
```

```
+     "Then, if you simply publish your LD number as an extra parameter of
+     the device, you can look through /sys to find it."
+
+           − James Bottomley <James.Bottomley@SteelEye.com>, 01.03.2005
+           linux−scsi mailing list
+
+
+     "I don't see why not ... it's your driver, you can publish whatever
+     extra information you need as scsi_device attributes; that was one of
+     the designs of the extensible attribute system."
+
+           − James Bottomley <James.Bottomley@SteelEye.com>, 01.06.2005
+           linux−scsi mailing list
+
+ii.  Add AMI megaraid support − Brian King <brking@charter.net>
+           PCI_VENDOR_ID_AMI, PCI_DEVICE_ID_AMI_MEGARAID3,
+           PCI_VENDOR_ID_AMI, PCI_SUBSYS_ID_PERC3_DC,
+
+iii. Make some code static
+           − Adrian Bunk <bunk@stusta.de>, 11.15.2004
+           linux−scsi mailing list
+
+iv.  Added NEC ROMB support − NEC MegaRAID PCI Express ROMB controller
+           PCI_VENDOR_ID_LSI_LOGIC, PCI_DEVICE_ID_MEGARAID_NEC_ROMB_2E,
+           PCI_SUBSYS_ID_NEC, PCI_SUBSYS_ID_MEGARAID_NEC_ROMB_2E,
+
+v.   Fixed Tape drive issue : For any Direct CDB command to physical device
+     including tape, timeout value set by driver was 10 minutes. With this
+     value, most of command will return within timeout. However, for those
+     command like ERASE or FORMAT, it takes more than an hour depends on
+     capacity of the device and the command could be terminated before it
+     completes.
+     To address this issue, the 'timeout' field in the DCDB command will
+     have NO TIMEOUT (i.e., 4) value as its timeout on DCDB command.
+
+
+Release Date     : Thu Dec  9 19:10:23 EST 2004
+     − Sreenivas Bagalkote <sreenib@lsil.com>
+
+Current Version  : 2.20.4.2 (scsi module), 2.20.2.4 (cmm module)
+Older Version    : 2.20.4.1 (scsi module), 2.20.2.3 (cmm module)
+
+i.   Introduced driver ioctl that returns scsi address for a given ld.
+
+     "Why can't the existing sysfs interfaces be used to do this?"
+           − Brian King (brking@us.ibm.com)
+
+     "I've looked into solving this another way, but I cannot see how
+     to get this driver−private mapping of logical drive number−> HCTL
+     without putting code something like this into the driver."
+
+     "...and by providing a mapping a function to userspace, the driver
+     is free to change its mapping algorithm in the future if necessary .."
+           − Matt Domsch (Matt_Domsch@dell.com)
+
 Release Date     : Thu Dec  9 19:02:14 EST 2004 − Sreenivas Bagalkote
<sreenib@lsil.com>

 Current Version  : 2.20.4.1 (scsi module), 2.20.2.3 (cmm module)
diff −Naur linux_bk/drivers/scsi/megaraid/Kconfig.megaraid
linux_bk.new/drivers/scsi/megaraid/Kconfig.megaraid
−−− linux_bk/drivers/scsi/megaraid/Kconfig.megaraid   2005−01−25
```

```
18:13:37.000000000 −0500
+++ linux_bk.new/drivers/scsi/megaraid/Kconfig.megaraid     2005−01−25
18:22:19.382663024 −0500
@@ −59,6 +59,7 @@
      INTEL RAID Controller SRCU51L 1000:1960:8086:0520
      FSC MegaRAID PCI Express ROMB 1000:0408:1734:1065
      ACER MegaRAID ROMB−2E         1000:0408:1025:004D
+     NEC MegaRAID PCI Express ROMB 1000:0408:1033:8287

      To compile this driver as a module, choose M here: the
      module will be called megaraid_mbox
diff −Naur linux_bk/drivers/scsi/megaraid/mega_common.h
linux_bk.new/drivers/scsi/megaraid/mega_common.h
−−− linux_bk/drivers/scsi/megaraid/mega_common.h      2005−01−25
18:13:37.000000000 −0500
+++ linux_bk.new/drivers/scsi/megaraid/mega_common.h  2005−01−25
18:22:22.547181944 −0500
@@ −221,6 +221,9 @@
 #define MRAID_IS_LOGICAL(adp, scp) \
      (SCP2CHANNEL(scp) == (adp)−>max_channel) ? 1 : 0

+#define MRAID_IS_LOGICAL_SDEV(adp, sdev) \
+     (sdev−>channel == (adp)−>max_channel) ? 1 : 0
+
 #define MRAID_GET_DEVICE_MAP(adp, scp, p_chan, target, islogical)      \
      /*                                                        \
       * Is the request coming for the virtual channel            \
diff −Naur linux_bk/drivers/scsi/megaraid/megaraid_ioctl.h
linux_bk.new/drivers/scsi/megaraid/megaraid_ioctl.h
−−− linux_bk/drivers/scsi/megaraid/megaraid_ioctl.h   2005−01−25
18:13:37.000000000 −0500
+++ linux_bk.new/drivers/scsi/megaraid/megaraid_ioctl.h     2005−01−25
18:22:23.345060648 −0500
@@ −291,5 +291,6 @@

 int mraid_mm_register_adp(mraid_mmadp_t *);
 int mraid_mm_unregister_adp(uint32_t);
+uint32_t mraid_mm_adapter_app_handle(uint32_t);

 #endif /* _MEGARAID_IOCTL_H_ */
diff −Naur linux_bk/drivers/scsi/megaraid/megaraid_mbox.c
linux_bk.new/drivers/scsi/megaraid/megaraid_mbox.c
−−− linux_bk/drivers/scsi/megaraid/megaraid_mbox.c    2005−01−25
18:13:37.000000000 −0500
+++ linux_bk.new/drivers/scsi/megaraid/megaraid_mbox.c      2005−01−25
18:22:24.359906368 −0500
@@ −10,7 +10,7 @@
  *      2 of the License, or (at your option) any later version.
  *
  * FILE          : megaraid_mbox.c
− * Version : v2.20.4.1 (Nov 04 2004)
+ * Version : v2.20.4.3 (Jan 21 2005)
  *
  * Authors:
  *   Atul Mukker       <Atul.Mukker@lsil.com>
@@ −60,12 +60,11 @@
  * INTEL RAID Controller SROMBU42E 1000  0408  8086  3499
  * INTEL RAID Controller SRCU51L   1000  1960  8086  0520
  *
− *
  * FSC     MegaRAID PCI Express ROMB     1000  0408  1734  1065
  *
```

```
− *
  * ACER    MegaRAID ROMB−2E        1000  0408  1025  004D
  *
+ * NEC     MegaRAID PCI Express ROMB     1000  0408  1033  8287
  *
  * For history of changes, see Documentation/ChangeLog.megaraid
  */
@@ −91,6 +90,9 @@
 static int megaraid_mbox_setup_dma_pools(adapter_t *);
 static void megaraid_mbox_teardown_dma_pools(adapter_t *);

+static int megaraid_sysfs_alloc_resources(adapter_t *);
+static void megaraid_sysfs_free_resources(adapter_t *);
+
 static int megaraid_abort_handler(struct scsi_cmnd *);
 static int megaraid_reset_handler(struct scsi_cmnd *);

@@ −121,6 +123,9 @@

 static void megaraid_mbox_dpc(unsigned long);

+static ssize_t megaraid_sysfs_show_app_hndl(struct class_device *, char *);
+static ssize_t megaraid_sysfs_show_ldnum(struct device *, char *);
+
 static int megaraid_cmm_register(adapter_t *);
 static int megaraid_cmm_unregister(adapter_t *);
 static int megaraid_mbox_mm_handler(unsigned long, uioc_t *, uint32_t);
@@ −197,7 +202,7 @@
  * ### global data ###
  */
 static uint8_t megaraid_mbox_version[8] =
−     { 0x02, 0x20, 0x04, 0x00, 9, 27, 20, 4 };
+     { 0x02, 0x20, 0x04, 0x03, 1, 21, 20, 5 };


 /*
@@ −301,6 +306,12 @@
             PCI_SUBSYS_ID_PERC3_SC,
      },
      {
+            PCI_VENDOR_ID_AMI,
+            PCI_DEVICE_ID_AMI_MEGARAID3,
+            PCI_VENDOR_ID_AMI,
+            PCI_SUBSYS_ID_PERC3_DC,
+     },
+     {
             PCI_VENDOR_ID_LSI_LOGIC,
             PCI_DEVICE_ID_MEGARAID_SCSI_320_0,
             PCI_VENDOR_ID_LSI_LOGIC,
@@ −438,6 +449,12 @@
             PCI_VENDOR_ID_AI,
             PCI_SUBSYS_ID_MEGARAID_ACER_ROMB_2E,
      },
+     {
+            PCI_VENDOR_ID_LSI_LOGIC,
+            PCI_DEVICE_ID_MEGARAID_NEC_ROMB_2E,
+            PCI_VENDOR_ID_NEC,
+            PCI_SUBSYS_ID_MEGARAID_NEC_ROMB_2E,
+     },
      {0}   /* Terminating entry */
 };
 MODULE_DEVICE_TABLE(pci, pci_id_table_g);
```

```
@@ −454,6 +471,29 @@
 };


+
+// definitions for the device attributes for exporting logical drive number
+// for a scsi address (Host, Channel, Id, Lun)
+
+CLASS_DEVICE_ATTR(megaraid_mbox_app_hndl, S_IRUSR,
megaraid_sysfs_show_app_hndl,
+           NULL);
+
+// Host template initializer for megaraid mbox sysfs device attributes
+static struct class_device_attribute *megaraid_class_device_attrs[] = {
+     &class_device_attr_megaraid_mbox_app_hndl,
+     NULL,
+};
+
+
+DEVICE_ATTR(megaraid_mbox_ld, S_IRUSR, megaraid_sysfs_show_ldnum, NULL);
+
+// Host template initializer for megaraid mbox sysfs device attributes
+static struct device_attribute *megaraid_device_attrs[] = {
+     &dev_attr_megaraid_mbox_ld,
+     NULL,
+};
+
+
 /*
  * Scsi host template for megaraid unified driver
  */
@@ −467,6 +507,8 @@
      .eh_bus_reset_handler         = megaraid_reset_handler,
      .eh_host_reset_handler        = megaraid_reset_handler,
      .use_clustering               = ENABLE_CLUSTERING,
+     .sdev_attrs             = megaraid_device_attrs,
+     .shost_attrs                  = megaraid_class_device_attrs,
 };


@@ −953,6 +995,8 @@
            }
            adapter−>device_ids[adapter−>max_channel][adapter−>init_id] =
                  0xFF;
+
+           raid_dev−>random_del_supported = 1;
      }

      /*
@@ −977,6 +1021,14 @@
       */
      adapter−>cmd_per_lun = megaraid_cmd_per_lun;

+     /*
+     * Allocate resources required to issue FW calls, when sysfs is
+     * accessed
+     */
+     if (megaraid_sysfs_alloc_resources(adapter) != 0) {
+           goto out_alloc_cmds;
+     }
+
      // Set the DMA mask to 64−bit. All supported controllers as capable of
```

```
      // DMA in this range
      if (pci_set_dma_mask(adapter->pdev, 0xFFFFFFFFFFFFFFFFULL) != 0) {
@@ -984,7 +1036,7 @@
            con_log(CL_ANN, (KERN_WARNING
                  "megaraid: could not set DMA mask for 64-bit.\n"));

-           goto out_alloc_cmds;
+           goto out_free_sysfs_res;
      }

      // setup tasklet for DPC
@@ -996,6 +1048,8 @@

      return 0;

+out_free_sysfs_res:
+     megaraid_sysfs_free_resources(adapter);
 out_alloc_cmds:
      megaraid_free_cmd_packets(adapter);
 out_free_irq:
@@ -1025,6 +1079,8 @@

      tasklet_kill(&adapter->dpc_h);

+     megaraid_sysfs_free_resources(adapter);
+
      megaraid_free_cmd_packets(adapter);

      free_irq(adapter->irq, adapter);
@@ -1559,12 +1615,14 @@

      if (scb->dma_direction == PCI_DMA_TODEVICE) {
            if (!scb->scp->use_sg) {      // sg list not used
-                 pci_dma_sync_single_for_device(adapter->pdev,
ccb->buf_dma_h,
+                 pci_dma_sync_single_for_device(adapter->pdev,
+                             ccb->buf_dma_h,
                              scb->scp->request_bufflen,
                              PCI_DMA_TODEVICE);
            }
            else {
-                 pci_dma_sync_sg_for_device(adapter->pdev,
scb->scp->request_buffer,
+                 pci_dma_sync_sg_for_device(adapter->pdev,
+                       scb->scp->request_buffer,
                        scb->scp->use_sg, PCI_DMA_TODEVICE);
            }
      }
@@ -2107,7 +2165,8 @@
      channel     = scb->dev_channel;
      target      = scb->dev_target;

-     pthru->timeout          = 1;  // 0=6sec, 1=60sec, 2=10min, 3=3hrs
+     // 0=6sec, 1=60sec, 2=10min, 3=3hrs, 4=NO timeout
+     pthru->timeout          = 4;
      pthru->ars        = 1;
      pthru->islogical  = 0;
      pthru->channel          = 0;
@@ -2155,7 +2214,8 @@
      channel     = scb->dev_channel;
      target      = scb->dev_target;
```

```
−     epthru−>timeout         = 1;  // 0=6sec, 1=60sec, 2=10min, 3=3hrs
+     // 0=6sec, 1=60sec, 2=10min, 3=3hrs, 4=NO timeout
+     epthru−>timeout         = 4;
      epthru−>ars       = 1;
      epthru−>islogical = 0;
      epthru−>channel         = 0;
@@ −3306,7 +3366,7 @@
      memset((caddr_t)raw_mbox, 0, sizeof(mbox_t));

      raw_mbox[0] = FC_DEL_LOGDRV;
−     raw_mbox[0] = OP_SUP_DEL_LOGDRV;
+     raw_mbox[2] = OP_SUP_DEL_LOGDRV;

      // Issue the command
      rval = 0;
@@ −3719,8 +3779,9 @@

      spin_unlock_irqrestore(USER_FREE_LIST_LOCK(adapter), flags);

−     scb−>state  = SCB_ACTIVE;
−     scb−>dma_type     = MRAID_DMA_NONE;
+     scb−>state        = SCB_ACTIVE;
+     scb−>dma_type           = MRAID_DMA_NONE;
+     scb−>dma_direction      = PCI_DMA_NONE;

      ccb         = (mbox_ccb_t *)scb−>ccb;
      mbox64            = (mbox64_t *)(unsigned long)kioc−>cmdbuf;
@@ −3888,6 +3949,324 @@
  */


+
+/**
+ * megaraid_sysfs_alloc_resources − allocate sysfs related resources
+ *
+ * Allocate packets required to issue FW calls whenever the sysfs
attributes
+ * are read. These attributes would require up−to−date information from the
+ * FW. Also set up resources for mutual exclusion to share these resources
and
+ * the wait queue.
+ *
+ * @param adapter : controller's soft state
+ *
+ * @return 0 on success
+ * @return −ERROR_CODE on failure
+ */
+static int
+megaraid_sysfs_alloc_resources(adapter_t *adapter)
+{
+     mraid_device_t    *raid_dev = ADAP2RAIDDEV(adapter);
+     int         rval = 0;
+
+     raid_dev−>sysfs_uioc = kmalloc(sizeof(uioc_t), GFP_KERNEL);
+
+     raid_dev−>sysfs_mbox64 = kmalloc(sizeof(mbox64_t), GFP_KERNEL);
+
+     raid_dev−>sysfs_buffer = pci_alloc_consistent(adapter−>pdev,
+                PAGE_SIZE, &raid_dev−>sysfs_buffer_dma);
+
+     if (!raid_dev−>sysfs_uioc || !raid_dev−>sysfs_mbox64 ||
+           !raid_dev−>sysfs_buffer) {
```

```
+
+           con_log(CL_ANN, (KERN_WARNING
+                 "megaraid: out of memory, %s %d\n", __FUNCTION__,
+                 __LINE__));
+
+           rval = −ENOMEM;
+
+           megaraid_sysfs_free_resources(adapter);
+     }
+
+     sema_init(&raid_dev−>sysfs_sem, 1);
+
+     init_waitqueue_head(&raid_dev−>sysfs_wait_q);
+
+     return rval;
+}
+
+
+/**
+ * megaraid_sysfs_free_resources − free sysfs related resources
+ *
+ * Free packets allocated for sysfs FW commands
+ *
+ * @param adapter : controller's soft state
+ */
+static void
+megaraid_sysfs_free_resources(adapter_t *adapter)
+{
+     mraid_device_t    *raid_dev = ADAP2RAIDDEV(adapter);
+
+     if (raid_dev−>sysfs_uioc) kfree(raid_dev−>sysfs_uioc);
+
+     if (raid_dev−>sysfs_mbox64) kfree(raid_dev−>sysfs_mbox64);
+
+     if (raid_dev−>sysfs_buffer) {
+           pci_free_consistent(adapter−>pdev, PAGE_SIZE,
+                 raid_dev−>sysfs_buffer, raid_dev−>sysfs_buffer_dma);
+     }
+}
+
+
+/**
+ * megaraid_sysfs_get_ldmap_done − callback for get ldmap
+ *
+ * Callback routine called in the ISR/tasklet context for get ldmap call
+ *
+ * @param uioc : completed packet
+ */
+static void
+megaraid_sysfs_get_ldmap_done(uioc_t *uioc)
+{
+     adapter_t   *adapter = (adapter_t *)uioc−>buf_vaddr;
+     mraid_device_t    *raid_dev = ADAP2RAIDDEV(adapter);
+
+     uioc−>status = 0;
+
+     wake_up(&raid_dev−>sysfs_wait_q);
+}
+
+
+/**
+ * megaraid_sysfs_get_ldmap_timeout − timeout handling for get ldmap
```

```
+ *
+ * Timeout routine to recover and return to application, in case the
adapter
+ * has stopped responding. A timeout of 60 seconds for this command seem
like
+ * a good value
+ *
+ * @param uioc : timed out packet
+ */
+static void
+megaraid_sysfs_get_ldmap_timeout(unsigned long data)
+{
+     uioc_t            *uioc = (uioc_t *)data;
+     adapter_t   *adapter = (adapter_t *)uioc−>buf_vaddr;
+     mraid_device_t    *raid_dev = ADAP2RAIDDEV(adapter);
+
+     uioc−>status = −ETIME;
+
+     wake_up(&raid_dev−>sysfs_wait_q);
+}
+
+
+/**
+ * megaraid_sysfs_get_ldmap − get update logical drive map
+ *
+ * This routine will be called whenever user reads the logical drive
+ * attributes, go get the current logical drive mapping table from the
+ * firmware. We use the managment API's to issue commands to the
controller.
+ *
+ * NOTE: The commands issuance functionality is not generalized and
+ * implemented in context of "get ld map" command only. If required, the
+ * command issuance logical can be trivially pulled out and implemented as
a
+ * standalone libary. For now, this should suffice since there is no other
+ * user of this interface.
+ *
+ * @param adapter : controller's soft state
+ *
+ * @return 0 on success
+ * @return −1 on failure
+ */
+static int
+megaraid_sysfs_get_ldmap(adapter_t *adapter)
+{
+     mraid_device_t          *raid_dev = ADAP2RAIDDEV(adapter);
+     uioc_t                  *uioc;
+     mbox64_t          *mbox64;
+     mbox_t                  *mbox;
+     char              *raw_mbox;
+     struct timer_list sysfs_timer;
+     struct timer_list *timerp;
+     caddr_t                 ldmap;
+     int               rval = 0;
+
+     /*
+     * Allow only one read at a time to go through the sysfs attributes
+     */
+     down(&raid_dev−>sysfs_sem);
+
+     uioc  = raid_dev−>sysfs_uioc;
+     mbox64      = raid_dev−>sysfs_mbox64;
```

```
+	ldmap = raid_dev->sysfs_buffer;
+
+	memset(uioc, sizeof(uioc_t), 0);
+	memset(mbox64, sizeof(mbox64_t), 0);
+	memset(ldmap, sizeof(raid_dev->curr_ldmap), 0);
+
+	mbox		= &mbox64->mbox32;
+	raw_mbox	= (char *)mbox;
+	uioc->cmdbuf	= (uint64_t)(unsigned long)mbox64;
+	uioc->buf_vaddr	= (caddr_t)adapter;
+	uioc->status	= -ENODATA;
+	uioc->done	= megaraid_sysfs_get_ldmap_done;
+
+	/*
+	* Prepare the mailbox packet to get the current logical drive mapping
+	* table
+	*/
+	mbox->xferaddr = (uint32_t)raid_dev->sysfs_buffer_dma;
+
+	raw_mbox[0] = FC_DEL_LOGDRV;
+	raw_mbox[2] = OP_GET_LDID_MAP;
+
+	/*
+	* Setup a timer to recover from a non-responding controller
+	*/
+	timerp	= &sysfs_timer;
+	init_timer(timerp);
+
+	timerp->function  = megaraid_sysfs_get_ldmap_timeout;
+	timerp->data		= (unsigned long)uioc;
+	timerp->expires		= jiffies + 60 * HZ;
+
+	add_timer(timerp);
+
+	/*
+	* Send the command to the firmware
+	*/
+	rval = megaraid_mbox_mm_command(adapter, uioc);
+
+	if (rval == 0) {  // command successfully issued
+		wait_event(raid_dev->sysfs_wait_q, (uioc->status != -ENODATA));
+
+		/*
+		* Check if the command timed out
+		*/
+		if (uioc->status == -ETIME) {
+			con_log(CL_ANN, (KERN_NOTICE
+				"megaraid: sysfs get ld map timed out\n"));
+
+			rval = -ETIME;
+		}
+		else {
+			rval = mbox->status;
+		}
+
+		if (rval == 0) {
+			memcpy(raid_dev->curr_ldmap, ldmap,
+				sizeof(raid_dev->curr_ldmap));
+		}
+		else {
+			con_log(CL_ANN, (KERN_NOTICE
+				"megaraid: get ld map failed with %x\n", rval));
```

```
+           }
+     }
+     else {
+           con_log(CL_ANN, (KERN_NOTICE
+                 "megaraid: could not issue ldmap command:%x\n", rval));
+     }
+
+
+     del_timer_sync(timerp);
+
+     up(&raid_dev−>sysfs_sem);
+
+     return rval;
+}
+
+
+/**
+ * megaraid_sysfs_show_app_hndl − display application handle for this
adapter
+ *
+ * Display the handle used by the applications while executing management
+ * tasks on the adapter. We invoke a management module API to get the
adapter
+ * handle, since we do not interface with applications directly.
+ *
+ * @param cdev   : class device object representation for the host
+ * @param buf    : buffer to send data to
+ */
+static ssize_t
+megaraid_sysfs_show_app_hndl(struct class_device *cdev, char *buf)
+{
+     struct Scsi_Host *shost = class_to_shost(cdev);
+     adapter_t   *adapter = (adapter_t *)SCSIHOST2ADAP(shost);
+     uint32_t    app_hndl;
+
+     app_hndl = mraid_mm_adapter_app_handle(adapter−>unique_id);
+
+     return snprintf(buf, 8, "%u\n", app_hndl);
+}
+
+
+/**
+ * megaraid_sysfs_show_ldnum − display the logical drive number for this
device
+ *
+ * Display the logical drive number for the device in question, if it a
valid
+ * logical drive. For physical devices, "−1" is returned
+ * The logical drive number is displayed in following format
+ *
+ * <SCSI ID> <LD NUM> <LD STICKY ID> <APP ADAPTER HANDLE>
+ *   <int>     <int>       <int>            <int>
+ *
+ * @param dev    : device object representation for the scsi device
+ * @param buf    : buffer to send data to
+ */
+static ssize_t
+megaraid_sysfs_show_ldnum(struct device *dev, char *buf)
+{
+     struct scsi_device *sdev = to_scsi_device(dev);
+     adapter_t   *adapter = (adapter_t *)SCSIHOST2ADAP(sdev−>host);
+     mraid_device_t    *raid_dev = ADAP2RAIDDEV(adapter);
```

```
+     int         scsi_id = −1;
+     int         logical_drv = −1;
+     int         ldid_map = −1;
+     uint32_t    app_hndl = 0;
+     int         mapped_sdev_id;
+     int         rval;
+     int         i;
+
+     if (raid_dev−>random_del_supported &&
+                 MRAID_IS_LOGICAL_SDEV(adapter, sdev)) {
+
+           rval = megaraid_sysfs_get_ldmap(adapter);
+           if (rval == 0) {
+
+                 for (i = 0; i < MAX_LOGICAL_DRIVES_40LD; i++) {
+
+                       mapped_sdev_id = sdev−>id;
+
+                       if (sdev−>id > adapter−>init_id) {
+                             mapped_sdev_id −= 1;
+                       }
+
+                       if (raid_dev−>curr_ldmap[i] == mapped_sdev_id) {
+
+                             scsi_id = sdev−>id;
+
+                             logical_drv = i;
+
+                             ldid_map = raid_dev−>curr_ldmap[i];
+
+                             app_hndl = mraid_mm_adapter_app_handle(
+                                         adapter−>unique_id);
+
+                             break;
+                       }
+                 }
+           }
+           else {
+                 con_log(CL_ANN, (KERN_NOTICE
+                       "megaraid: sysfs get ld map failed: %x\n",
+                       rval));
+           }
+     }
+
+     return snprintf(buf, 36, "%d %d %d %d\n", scsi_id, logical_drv,
+                 ldid_map, app_hndl);
+}
+
+
 /*
  * END: Mailbox Low Level Driver
  */
diff −Naur linux_bk/drivers/scsi/megaraid/megaraid_mbox.h
linux_bk.new/drivers/scsi/megaraid/megaraid_mbox.h
−−− linux_bk/drivers/scsi/megaraid/megaraid_mbox.h    2005−01−25
18:13:37.000000000 −0500
+++ linux_bk.new/drivers/scsi/megaraid/megaraid_mbox.h      2005−01−25
18:22:25.343756800 −0500
@@ −21,8 +21,8 @@
 #include "megaraid_ioctl.h"
```

```
−#define MEGARAID_VERSION     "2.20.4.1"
−#define MEGARAID_EXT_VERSION "(Release Date: Thu Nov  4 17:44:59 EST 2004)"
+#define MEGARAID_VERSION     "2.20.4.3"
+#define MEGARAID_EXT_VERSION "(Release Date: Fri Jan 21 00:01:03 EST 2005)"


 /*
@@ −137,6 +137,9 @@
 #define PCI_SUBSYS_ID_PERC3_DC                      0x0493
 #define PCI_SUBSYS_ID_PERC3_SC                      0x0475

+#define PCI_DEVICE_ID_MEGARAID_NEC_ROMB_2E          0x0408
+#define PCI_SUBSYS_ID_MEGARAID_NEC_ROMB_2E          0x8287
+
 #ifndef PCI_SUBSYS_ID_FSC
 #define PCI_SUBSYS_ID_FSC                     0x1734
 #endif
@@ −216,6 +219,14 @@
  * @param hw_error           : set if FW not responding
  * @param fast_load          : If set, skip physical device scanning
  * @channel_class            : channel class, RAID or SCSI
+ * @sysfs_sem                : semaphore to serialize access to sysfs res.
+ * @sysfs_uioc               : management packet to issue FW calls from
sysfs
+ * @sysfs_mbox64       : mailbox packet to issue FW calls from sysfs
+ * @sysfs_buffer       : data buffer for FW commands issued from sysfs
+ * @sysfs_buffer_dma         : DMA buffer for FW commands issued from sysfs
+ * @sysfs_wait_q       : wait queue for sysfs operations
+ * @random_del_supported     : set if the random deletion is supported
+ * @curr_ldmap               : current LDID map
  *
  * Initialization structure for mailbox controllers: memory based and IO
based
  * All the fields in this structure are LLD specific and may be discovered
at
@@ −223,6 +234,7 @@
  *
  * NOTE: The fields of this structures are placed to minimize cache misses
  */
+#define MAX_LD_EXTENDED64    64
 typedef struct {
      mbox64_t               *una_mbox64;
      dma_addr_t             una_mbox64_dma;
@@ −247,6 +259,14 @@
      int                    hw_error;
      int                    fast_load;
      uint8_t                      channel_class;
+     struct semaphore       sysfs_sem;
+     uioc_t                       *sysfs_uioc;
+     mbox64_t               *sysfs_mbox64;
+     caddr_t                      sysfs_buffer;
+     dma_addr_t             sysfs_buffer_dma;
+     wait_queue_head_t      sysfs_wait_q;
+     int                    random_del_supported;
+     uint16_t               curr_ldmap[MAX_LD_EXTENDED64];
 } mraid_device_t;

 // route to raid device from adapter
diff −Naur linux_bk/drivers/scsi/megaraid/megaraid_mm.c
linux_bk.new/drivers/scsi/megaraid/megaraid_mm.c
−−− linux_bk/drivers/scsi/megaraid/megaraid_mm.c      2005−01−25
18:13:37.000000000 −0500
```

```
+++ linux_bk.new/drivers/scsi/megaraid/megaraid_mm.c  2005−01−25
18:22:26.421592944 −0500
@@ −10,7 +10,7 @@
  *      2 of the License, or (at your option) any later version.
  *
  * FILE          : megaraid_mm.c
− * Version : v2.20.2.3 (Dec 09 2004)
+ * Version : v2.20.2.5 (Jan 21 2005)
  *
  * Common management module
  */
@@ −58,6 +58,7 @@

 EXPORT_SYMBOL(mraid_mm_register_adp);
 EXPORT_SYMBOL(mraid_mm_unregister_adp);
+EXPORT_SYMBOL(mraid_mm_adapter_app_handle);

 static int majorno;
 static uint32_t drvr_ver     = 0x02200201;
@@ −65,7 +66,7 @@
 static int adapters_count_g;
 static struct list_head adapters_list_g;

−wait_queue_head_t wait_q;
+static wait_queue_head_t wait_q;

 static struct file_operations lsi_fops = {
      .open = mraid_mm_open,
@@ −1007,6 +1008,40 @@
      return rval;
 }

+
+/**
+ * mraid_mm_adapter_app_handle − return the application handle for this
adapter
+ *
+ * For the given driver data, locate the adadpter in our global list and
+ * return the corresponding handle, which is also used by applications to
+ * uniquely identify an adapter.
+ *
+ * @param unique_id : adapter unique identifier
+ *
+ * @return adapter handle if found in the list
+ * @return 0 if adapter could not be located, should never happen though
+ */
+uint32_t
+mraid_mm_adapter_app_handle(uint32_t unique_id)
+{
+     mraid_mmadp_t     *adapter;
+     mraid_mmadp_t     *tmp;
+     int         index = 0;
+
+     list_for_each_entry_safe(adapter, tmp, &adapters_list_g, list) {
+
+          if (adapter−>unique_id == unique_id) {
+
+               return MKADAP(index);
+          }
+
+          index++;
+     }
```

```
+
+     return 0;
+}
+
+
 /**
  * mraid_mm_setup_dma_pools − Set up dma buffer pools per adapter
  *
diff −Naur linux_bk/drivers/scsi/megaraid/megaraid_mm.h
linux_bk.new/drivers/scsi/megaraid/megaraid_mm.h
−−− linux_bk/drivers/scsi/megaraid/megaraid_mm.h      2005−01−25
18:13:37.000000000 −0500
+++ linux_bk.new/drivers/scsi/megaraid/megaraid_mm.h  2005−01−25
18:22:27.692399752 −0500
@@ −29,9 +29,9 @@
 #include "megaraid_ioctl.h"


−#define LSI_COMMON_MOD_VERSION     "2.20.2.3"
+#define LSI_COMMON_MOD_VERSION     "2.20.2.5"
 #define LSI_COMMON_MOD_EXT_VERSION \
−           "(Release Date: Thu Dec  9 19:02:14 EST 2004)"
+           "(Release Date: Fri Jan 21 00:01:03 EST 2005)"

 #define LSI_DBGLVL                 dbglevel
−
To unsubscribe from this list: send the line "unsubscribe linux−kernel" in
the body of a message to majordomo@vger.kernel.org
More majordomo info at  http://vger.kernel.org/majordomo−info.html
Please read the FAQ at  http://www.tux.org/lkml/
```