

Re: [RFC] MTD driver for MMC cards

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-04/msg05375.html>

- *From:* Arnd Bergmann <arnd@xxxxxxxx>
 - *Date:* Mon, 16 Apr 2007 01:33:17 +0200
-

This is a new version of the driver I posted back in January. I now have hardware to test it and fixed a number of bugs, most of which are the ones that Pierre told me about in the first place.

It now seems to work fine with the mtddblock driver, which of course it entirely pointless.

I've tried using it with jffs2 once, but get an immediate oops, which still needs some investigation.

I'm also not sure what to do about SDHC media, but probably the easiest solutions is to disallow them with this driver — the mtd layer doesn't deal with media larger than 4GB anyway at this point.

There is also still some need for performance testing. Jörn brought up the point that if a specific card can't have multiple open erase block simulatiously, it's rather pointless for logfs. It might still be useful to use jffs2 on those cards, because IFAIK that only writes to one erase block at any time.

Signed-off-by: Arnd Bergmann <arnd@xxxxxxxx>

Index: [olpc-2.6/drivers/mmc/mmc.c](#)

```
=====
--- olpc-2.6.orig/drivers/mmc/mmc.c
+++ olpc-2.6/drivers/mmc/mmc.c
@@ -621,6 +621,7 @@ static void mmc_decode_csd(struct mmc_ca
csd->r2w_factor = UNSTUFF_BITS(resp, 26, 3);
csd->write_blkbits = UNSTUFF_BITS(resp, 22, 4);
csd->write_partial = UNSTUFF_BITS(resp, 21, 1);
+ csd->erase_blksize = UNSTUFF_BITS(resp, 39, 7);
break;
case 1:
/*
@@ -649,6 +650,8 @@ static void mmc_decode_csd(struct mmc_ca
```

Re: [RFC] MTD driver for MMC cards

```
csd->r2w_factor = 4; /* Unused */
csd->write_blkbits = 9;
csd->write_partial = 0;
+ #warning need to read au_size for sdhc
+ csd->erase_blksize = 0; // 8192 << au_size;
break;
default:
printk("%s: unrecognised CSD structure version %d\n",
@@ -691,6 +694,8 @@ static void mmc_decode_csd(struct mmc_ca
csd->r2w_factor = UNSTUFF_BITS(resp, 26, 3);
csd->write_blkbits = UNSTUFF_BITS(resp, 22, 4);
csd->write_partial = UNSTUFF_BITS(resp, 21, 1);
+ csd->erase_blksize = (UNSTUFF_BITS(resp, 37, 5) + 1) *
+ (UNSTUFF_BITS(resp, 42, 5) + 1);
}
}
```

Index: olpc-2.6/include/linux/mmc/card.h

```
=====
--- olpc-2.6.orig/include/linux/mmc/card.h
+++ olpc-2.6/include/linux/mmc/card.h
@@ -32,6 +32,7 @@ struct mmc_csd {
unsigned int max_dtr;
unsigned int read_blkbits;
unsigned int write_blkbits;
+ unsigned int erase_blksize;
unsigned int capacity;
unsigned int read_partial:1,
read_misalign:1,
Index: olpc-2.6/drivers/mmc/mmc_mtd.c
```

```
=====
--- /dev/null
+++ olpc-2.6/drivers/mmc/mmc_mtd.c
@@ -0,0 +1,366 @@
+/*
+ * MTD driver for MMC cards
+ */
+ #include <linux/init.h>
+ #include <linux/module.h>
+ #include <linux/mmc/card.h>
+ #include <linux/mmc/protocol.h>
+ #include <linux/mmc/host.h>
+ #include <linux/scatterlist.h>
+ #include <linux/mtd/mtd.h>
+
+ /*
+ * check if a write command was completed correctly, must be called
+ * with host claimed.
+ */
+ static int mmc_mtd_get_status(struct mmc_card *card)
+ {
```

Re: [RFC] MTD driver for MMC cards

```
+ int err;
+ struct mmc_command cmd;
+
+ do {
+ cmd = (struct mmc_command) {
+ .opcode = MMC_SEND_STATUS,
+ .arg = card->rca << 16,
+ .flags = MMC_RSP_R1 | MMC_CMD_AC,
+ };
+
+ err = mmc_wait_for_cmd(card->host, &cmd, 5);
+ if (err) {
+ dev_err(&card->dev, "error %d requesting status\n", err);
+ break;
+ }
+ } while (!(cmd.resp[0] & R1_READY_FOR_DATA));
+
+ return err;
+}
+
+/*
+ * erase a range of erase groups aligned to mtd->erase_size
+ */
+static int mmc_mtd_erase(struct mtd_info *mtd, struct erase_info *instr)
+{
+ struct mmc_card *card = mtd->priv;
+ struct mmc_command cmd[3] = { {
+ .opcode = MMC_ERASE_GROUP_START,
+ .arg = instr->addr,
+ .flags = MMC_RSP_R1 | MMC_CMD_AC,
+ }, {
+ .opcode = MMC_ERASE_GROUP_END,
+ .arg = instr->addr + instr->len,
+ .flags = MMC_RSP_R1 | MMC_CMD_AC,
+ }, {
+ .opcode = MMC_ERASE,
+ .flags = MMC_RSP_R1B | MMC_CMD_AC,
+ } },
+ };
+ int err, i;
+
+ dev_dbg(&card->dev, "%s: from %d len %d\n", __FUNCTION__,
+ instr->addr, instr->len);
+
+ instr->state = MTD_ERASING;
+ err = 0;
+ err = mmc_card_claim_host(card);
+ if (err)
+ goto error;
+
+ for (i=0; i<3; i++) {
```

Re: [RFC] MTD driver for MMC cards

```
+ err = mmc_wait_for_cmd(card->host, cmd, 5);
+ if (err) {
+ dev_err(&card->dev, "%s: error %d in stage %d\n",
+ __FUNCTION__, err, i);
+ break;
+ }
+ }
+ if (!err)
+ err = mmc_mtd_get_status(card);
+error:
+ mmc_card_release_host(card);
+ mtd_erase_callback(instr);
+ instr->state = err ? MTD_ERASE_FAILED : MTD_ERASE_DONE;
+ return err;
+}
+
+/*
+ * transfer a block to/from the card. The block needs to be aligned
+ * to mtd->writesize. If we want to implement an mtd_writev method,
+ * this needs to use stream operations with an appropriate stop
+ * command as well.
+ */
+static int mmc_mtd_transfer_low(struct mmc_card *card, loff_t off, size_t len,
+ size_t *retlen, u_char *buf, unsigned int blocksize,
+ int write)
+{
+ struct scatterlist sg;
+ struct mmc_data data = {
+ .blksz = blocksize,
+ .blocks = len / blocksize,
+ .flags = write ? MMC_DATA_WRITE : MMC_DATA_READ,
+ .sg = &sg,
+ .sg_len = 1,
+ };
+ struct mmc_command cmd = {
+ .arg = off,
+ .data = &data,
+ .flags = MMC_RSP_R1 | MMC_CMD_ADTC,
+ .opcode = write ? MMC_WRITE_BLOCK : MMC_READ_SINGLE_BLOCK,
+ };
+ struct mmc_command stop = {
+ .flags = MMC_RSP_R1B | MMC_CMD_AC,
+ .opcode = MMC_STOP_TRANSMISSION,
+ };
+ struct mmc_request mrq = {
+ .cmd = &cmd,
+ .data = &data,
+ };
+ int ret;
+
+ dev_dbg(&card->dev, "%s off %lld, len %zd\n",
```

Re: [RFC] MTD driver for MMC cards

```
+ write ? "write" : "read", off, len);
+
+ if (write &&
+ !(card->host->caps & MMC_CAP_MULTIWRITE) &&
+ data.blocks > 1 &&
+ !mmc_card_sd(card)) {
+ dev_dbg(&card->dev, "no multiwrite\n");
+ data.blocks = 1;
+ len = blocksize;
+ }
+
+ sg_init_one(&sg, buf, len);
+
+ if (data.blocks > 1) {
+ data.flags |= MMC_DATA_MULTI;
+ mrq.stop = &stop;
+ cmd.opcode = write ? MMC_WRITE_MULTIPLE_BLOCK
+ : MMC_READ_MULTIPLE_BLOCK;
+ }
+
+ mmc_set_data_timeout(&data, card, write);
+ mmc_wait_for_req(card->host, &mrq);
+
+ ret = 0;
+ if (cmd.error || data.error) {
+ dev_err(&card->dev, "error %d/%d sending read/write command\n",
+ cmd.error, data.error);
+ ret = -EIO;
+ }
+
+ /* wait for card to settle after write */
+ if (write && !ret)
+ ret = mmc_mtd_get_status(card);
+
+ if (!ret && retlen)
+ *retlen = len;
+ return ret;
+}
+
+/*
+ * do some common sanity checks and locking for the actual transfer
+ * function.
+ */
+static int mmc_mtd_transfer(struct mtd_info *mtd, loff_t off, size_t len,
+ size_t *retlen, u_char *buf, int write)
+{
+ struct mmc_card *card = mtd->priv;
+ unsigned long virt_addr = (unsigned long) buf;
+ int ret;
+
+ if (off > mtd->size)
```

```

+ return -EINVAL;
+ if (off + len > mtd->size)
+ len = mtd->size - off;
+
+ if (retlen)
+ *retlen = 0;
+
+ ret = mmc_card_claim_host(card);
+ if (ret) {
+ dev_warn(&card->dev, "%s: mmc_card_claim_host returned %d\n",
+ __FUNCTION__, ret);
+ ret = -EIO;
+ goto error;
+ }
+
+ /*
+ * Ugly: mtddblock passes in vmalloc addresses, but sdhci can't deal
+ * with actual scattered memory, so pass down individual pages.
+ */
+ if (virt_addr >= VMALLOC_START &&
+ virt_addr < VMALLOC_END) {
+ u_char *end = buf + len;
+ while (buf < end) {
+ unsigned long page_off = virt_addr & (PAGE_SIZE-1);
+ /* XXX this breaks with highmem */
+ u_char *bus_addr = pfn_to_kaddr(vmalloc_to_pfn(buf))
+ + page_off;
+
+ len = PAGE_SIZE - page_off;
+ if (len > (end - buf))
+ len = end - buf;
+
+ ret = mmc_mtd_transfer_low(card, off, len, &len,
+ bus_addr, mtd->writesize, write);
+ if (ret)
+ break;
+ if (retlen)
+ *retlen += len;
+ off += len;
+ buf += len;
+ }
+ } else {
+ ret = mmc_mtd_transfer_low(card, off, len, retlen,
+ buf, mtd->writesize, write);
+ }
+error:
+ mmc_card_release_host(card);
+
+ return ret;
+}
+

```

Re: [RFC] MTD driver for MMC cards

```
+static int mmc_mtd_read(struct mtd_info *mtd, loff_t from, size_t len,
+ size_t *retlen, u_char *buf)
+{
+ return mmc_mtd_transfer(mtd, from, len, retlen, buf, 0);
+}
+
+static int mmc_mtd_write(struct mtd_info *mtd, loff_t to, size_t len,
+ size_t *retlen, const u_char *buf)
+{
+ return mmc_mtd_transfer(mtd, to, len, retlen, (u_char *)buf, 1);
+}
+
+/*
+ * Set the block size used for this session */
+static int mmc_mtd_set_blksize(struct mmc_card *card, unsigned int size)
+{
+ struct mmc_command cmd;
+ int err;
+
+ /* Block-addressed cards ignore MMC_SET_BLOCKLEN. */
+ if (mmc_card_blockaddr(card))
+ return 0;
+
+ mmc_card_claim_host(card);
+ cmd.opcode = MMC_SET_BLOCKLEN;
+ cmd.arg = size;
+ cmd.flags = MMC_RSP_R1 | MMC_CMD_AC;
+ err = mmc_wait_for_cmd(card->host, &cmd, 5);
+ mmc_card_release_host(card);
+
+ return err;
+}
+
+/*
+ * Initialize an mmc card. We create a new MTD device for each
+ * MMC card we find. The operations are rather straightforward,
+ * so we don't even need our own data structure to contain the
+ * mtd_info.
+ */
+static int mmc_mtd_probe(struct mmc_card *card)
+{
+ struct mtd_info *mtd;
+ int ret;
+
+ if (!(card->csd.cmdclass & CCC_ERASE) ||
+ !(card->csd.cmdclass & CCC_BLOCK_READ))
+ return -ENODEV;
+
+ mtd = kzalloc(GFP_KERNEL, sizeof *mtd);
+ if (!mtd)
+ return -ENOMEM;
```

Re: [RFC] MTD driver for MMC cards

```
+
+ mtd->name = card->dev.bus_id;
+
+ /* 512 bytes is supposed to be supported by any card */
+ mtd->writesize = 512;
+ /* medium size in bytes */
+ mtd->size = card->csd.capacity << (card->csd.read_blkbits);
+ /* size of an erase unit in bytes */
+ mtd->erasesize = (1 << card->csd.write_blkbits) * card->csd.erase_blksize;
+
+ ret = mmc_mtd_set_blksize(card, mtd->writesize);
+ if (ret) {
+ dev_err(&card->dev, "Unable to set block size: %d\n", ret);
+ goto out;
+ }
+
+ dev_dbg(&card->dev, "size %d bytes, writesize %d, erasesize %d\n",
+ mtd->size, mtd->writesize, mtd->erasesize);
+
+ /*
+ * NAND is a relatively good approximation, but with the current
+ * MTD scheme, we might need to get our own type
+ */
+ mtd->type = MTD_NANDFLASH;
+ mtd->flags = MTD_CAP_NANDFLASH;
+
+ /* card might be readonly */
+ if (!(card->csd.cmdclass & CCC_BLOCK_WRITE) ||
+ mmc_card_readonly(card))
+ mtd->flags &= ~MTD_WRITEABLE;
+
+ /*
+ * operations that may be worthwhile implementing:
+ * writev, sync, lock, unlock, suspend, resume
+ * In the mean time, these should be sufficient.
+ */
+ mtd->erase = mmc_mtd_erase;
+ mtd->write = mmc_mtd_write;
+ mtd->writev = default_mtd_writev;
+ mtd->read = mmc_mtd_read;
+
+ mtd->owner = THIS_MODULE;
+ mtd->priv = card;
+ mmc_set_drvdata(card, mtd);
+
+ ret = add_mtd_device(mtd);
+out:
+ if (ret)
+ kfree(mtd);
+
+ return ret;
```

Re: [RFC] MTD driver for MMC cards

```
+}
+
+/*
+ * Not sure if this is safe. Can we call del_mtd_device while
+ * the device is still in use? Do we have a choice?
+ */
+static void mmc_mtd_remove(struct mmc_card *card)
+{
+ struct mtd_info *mtd;
+
+ mtd = mmc_get_drvdata(card);
+ del_mtd_device(mtd);
+ kfree(mtd);
+}
+
+/*
+ * This driver will match any card, so it conflicts with the
+ * mmc block driver. It's only possible to load one of them
+ * at a time.
+ */
+static struct mmc_driver mmc_mtd_driver = {
+ .drv = {
+ .name = "mmcmttd",
+ },
+ .probe = mmc_mtd_probe,
+ .remove = mmc_mtd_remove,
+};
+
+static int __init mmc_blk_init(void)
+{
+ return mmc_register_driver(&mmc_mtd_driver);
+}
+module_init(mmc_blk_init);
+
+static void __exit mmc_blk_exit(void)
+{
+ mmc_unregister_driver(&mmc_mtd_driver);
+}
+module_exit(mmc_blk_exit);
+
+MODULE_LICENSE("GPL");
+MODULE_DESCRIPTION("Multimedia Card (MMC) MTD device driver");
+MODULE_AUTHOR("Arnd Bergmann <arnd@xxxxxxxx>");
Index: olpc-2.6/drivers/mmc/Kconfig
```

```
=====
--- olpc-2.6.orig/drivers/mmc/Kconfig
+++ olpc-2.6/drivers/mmc/Kconfig
@@ -29,6+29,17 @@ config MMC_BLOCK
mount the filesystem. Almost everyone wishing MMC support
should say Y or M here.
```

Re: [RFC] MTD driver for MMC cards

```
+config MMC_MTD
+ tristate "MMC MTD device driver"
+ depends on MMC && MTD && (MMC_BLOCK = n || (MMC_BLOCK != y && MMC_BLOCK))
+ help
+ Enable this to as an alternative to the MMC block driver.
+ It makes it possible to use MMC memory cards with the
+ MTD layer and file systems based on it, e.g. jffs2.
+ Since both drivers operate on the same devices, you can't
+ build them both into the kernel and you need to say M here
+ if you want to build both.
```

```
+
config MMC_ARMMHCI
tristate "ARM AMBA Multimedia Card Interface support"
depends on ARM_AMBA && MMC
Index: olpc-2.6/drivers/mmc/Makefile
```

```
-----
--- olpc-2.6.orig/drivers/mmc/Makefile
+++ olpc-2.6/drivers/mmc/Makefile
@@ -11,6 +11,7 @@ obj-$(CONFIG_MMC) += mmc_core.o
# Media drivers
#
obj-$(CONFIG_MMC_BLOCK) += mmc_block.o
+obj-$(CONFIG_MMC_MTD) += mmc_mtd.o
```

```
#
# Host drivers
```

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