

CSDN

首页

博客

学院

下载

GitChat

TinyMind

论坛

问答

商城

...

搜博文文章



写博客



发Chat

登

小贼哥的技术花园

原来写技术博客这么给劲



0



写评论



收藏



编写自己的SD/MMC Host驱动（二）:工作过程和大结局

微信

2010年11月16日 19:40:00

阅读数：7154



微博

上次说到，如果注册成功的话，Linux就会开始初始化SD/MMC了，SD/MMC的初始化都是通过drivers/mmc/core/core.c里面mmc_rescan来完成的：

```
void mmc_rescan(struct work_struct *work)
```

```
{
```

```
.....
```

```
if (host->ops->get_cd && host->ops->get_cd(host) == 0)
```

```
    goto out;
```

```
.....
```

```
mmc_power_up(host);
```

```
mmc_go_idle(host);
```

```
.....
```

```
mmc_send_if_cond(host, host->ocr_avail);
```

```
/*
```

```
 * First we search for SDIO...
```

```
*/
```

```
.....
```

```
err = mmc_send_io_op_cond(host, 0, &ocr);
```

```
.....
```

```
/*
```

```
 * ...then normal SD...
```

```
*/
```

```
err = mmc_send_app_op_cond(host, 0, &ocr);
```

```
.....
```

```
/*
```

```
 * ...and finally MMC.
```

```
*/
```

```
err = mmc_send_op_cond(host, 0, &ocr);
```

```
.....
```

```
}
```

首先调用`get_cd`方法，看看是不是有卡在里面，不然岂不是白忙活！

然后调用`mmc_power_up`来上电，该函数将调用驱动的`set_ios`操作函数，控制电源和时钟等。

电也上了，此时就需要调用`mmc_go_idle`让SD/MMC卡进入IDLE状态，如果你对协议比较熟的话，就是发送CMD0了。然后调用`mmc_send_if_cond`发送CMD8设置接口，这个命令只有SD2.0才会响应，对于1.0标准是没关系的。然后就开始判断插入的卡到底是SDIO、SD还是MMC，其实都是通过发送特定的命令，看看其是否响应来判断的，其实这种方法不是很好，估计协议的设计者开始的时候没有想到会有如此多类型，因此没有设计一条专门的命令来查询卡的类型。可以看到SDIO是通过CMD5来识别的，如果有回应则是SDIO卡。SD卡是通过ACMD41来实现的，由于ACMD需要先发切换命令CMD55，因此如果在`request`里面打log，将会发现发了两条opcode分别为55和41的命令。MMC的判断则是通过CMD1，即MMC的`send operation condition`命令。这些命令都是各个类型所独有的，因此可以用来判断卡的类型。

卡的类型确定以后就会调用`mmc_init_card`来初始化卡，关于发送命令的过程需要参考协议的具体描述。

说了半天，终于应该说SD/MMC驱动中最重要的函数`request`了，其实SD/MMC的操作还是很单纯的，典型的一问一答型：先发送`request`，然后等待`request`完成，如果`request`没有完成，则不会发起下一个`request`，如果程序有bug，超过120s都不回应，恭喜你，系统会崩溃的。发送`request`当然是通过之前注册的`mmc_host_ops`里面的函数来发起的了，回应则是通过调用`mmc_request_done`来完成的。`Request`函数的实现一般是这样的：

```
static void cbpmci_send_request(struct mmc_host *mmc)
{
    struct mmc_command *cmd = host->cmd_is_stop ? mrq->stop : mrq->cmd;

    if (cmd->data) {
        cbpmci_setup_data(host, mrq->data);

        host->dcnt++;

        cbpmci_prepare_pio(host, cmd->data);
    }

    cbpmci_send_command(host, cmd);
}
```

首先通过`cmd->data`是否为空来判断这个命令是否有数据，然后做一些准备工作，比如`block size`、`block count`之类的东东，这些参数都在`cmd->data`里面。其实在这一段是不收发实际的数据的，只是准备而已，因为协议规定了，数据必须在命令发完了以后才能收发数据，因此其中一些准备工作也可以在命令完成中断里面来完成。不过标识`request`究竟对数据是进行读还是写的动作需要在这里完成，因为命令完成中断发生时需要利用这个信息。`Samsung`的那个`prepare`写的非常让人困惑，`PXA`的写得就清楚多了。

```
static int cbpmci_prepare_pio(struct cbpmci_host *host, struct mmc_data *data)
{
    host->pio_active=XFER_NONE;

    if(data->flags & MMC_DATA_READ)    host->pio_active=XFER_READ;

    if(data->flags & MMC_DATA_WRITE)    host->pio_active=XFER_WRITE;

}
```

紧接着就是调用`cbpmci_send_command`发送命令了。

命令发送完成后会产生命令完成中断，在处理该中断时应该判断当前的状态，即`host->pio_active`的值，如果该命令没有数据，则调用`mmc_request_done`直接返回，如果有则打开读写相关的中断，读写数据，在数据读写完成后在调用`mmc_request_done`通知上层。一般的读写过程是这样的，以读为例，在收到命令完成的中断后，判断此时处于读状态，然后打开发送FIFO的`almost full`、`full`以及数据完成中断，在这`almost full`和`full`中断的处理过程中读取FIFO的值，放入上层提供的`scatterlist`中，SD/MMC中的`scatterlist`操作在上上一篇博文中有详细的描写，当数据完成中断产生或者`scatterlist`已经填满时调用`mmc_request_done`通知上层。

在发送命令和数据的过程中都可能产生错误，这些错误都可以通过`host->mrq->cmd`以及`host->mrq->data`中的相关元素返回：

```
if (stat & HWD_MMC_CMD_RSPTOUT_ERR) cmd->error = -ETIMEDOUT;
```

```
else if (stat & HWD_MMC_CMD_RSP_CRC_ERR && cmd->flags & MMC_RSP_CRC)      cmd->error = -EILSEQ;
```

或者：

```
if (stat & HWD_MMC_DAT_TOUT_ERR)      data->error = -ETIMEDOUT;
else if (stat & (HWD_MMC_CMD_CRC_ERR|HWD_MMC_CRC_STS_ERR))      data->error = -EILSEQ;
```

至于发送命令的时候产生的response，则是根据返回是48bit或者136bit有所不同，host->mrq->cmd中的response只定义了4个word，也就是128位。在48bit返回值时cmd->resp[0]中是32bit的card status，cmd->resp[1]的高7位是回应的CRC值，第24位是停止位，就是“1”，没有研究代码，不知CRC和停止位不知是否有用。当为136bit时，其中的resp[0]到resp[3]中是128bit的card status。PXA和CBP的状态位都需要调整后才能写入resp[0]到resp[3]，而Samsung的s3cmci则可以直接从寄存器读出然后赋值，估计这也是三星的处理器之所以火的细节之一。

关于卡的热插拔，当卡插拔的时候，一般会产生中断，在中断中调用mmc_detect_change通知上层

```
if (isr_status & HWD_MMC_CARD_STS) mmc_detect_change(host->mmc, msecs_to_jiffies(500));
```

其实是启动了一个工作队列，看代码可知，该工作队列就是调用mmc_rescan函数，轰轰烈烈一番后，重新回到了原点，世界清静了。

如果各个过程都没有错误的话，在/dev下面就会出现两个设备节点mmcblk0和mmcblk0p1，如果没有的话，请问是不是忘敲了mdev -s了？

此时就可以mount了，不过mount之前由于一般SD/MMC卡都是FAT文件系统，因此还要在内核中加入dos 文件系统和FAT的支持。然后就可以mount了

```
Mount /dev/mmcblk0p1 /mnt
```

就可以到/mnt目录下看看你的成果了，如果写人小的文件，比如创建目录，一般要在sync或umount的时候才会产生实际的写入动作。

在mount的过程中如果出现错误：**FAT: codepage cp437 not found**。那是你的内核字符集不支持437，在内核中选中就OK了。

以下是各个配置的图片：

1) MMC配置

```
--- MMC/SD/SDIO card support
[*]   MMC debugging
[ ]   Allow unsafe resume (DANGEROUS)
*** MMC/SD/SDIO Card Drivers ***
<*>   MMC block device driver
[*]   Use bounce buffer for simple hosts
< >   SDIO UART/GPS class support
< >   MMC host test driver
*** MMC/SD/SDIO Host Controller Drivers ***
<*>   VTC CBP Multimedia Card Interface support
< >   Secure Digital Host Controller Interface support
<M>   Atmel SD/MMC Driver
```

2) FAT配置

```
<*> MSDOS fs support
<*> VFAT (Windows-95) fs support
(437) Default codepage for FAT
(iso8859-1) Default iocharset for FAT
< > NTFS file system support
```

3) Code Page 437配置

```
--- Native language support
(iso8859-1) Default NLS Option
<*>   Codepage 437 (United States, Canada)
< >   Codepage 737 (Greek)
< >   Codepage 775 (Baltic Rim)
< >   Codepage 850 (Europe)
< >   Codepage 852 (Central/Eastern Europe)
```

4) 附录: *set_ios* 范例

set_ios 设置一些控制参数, 比如时钟频率, 因为在初始化正常操作的时候时钟是不一样的, 初始化的时候一般小于400K, 正常操作的时候是26M, 52M等等高得多的频率。还有就是控制电源以及SD卡的总线宽度, CBP的SD总线宽度1,4,8都是支持的。*Set_ios*的代码基于文章后面的附录中。

写评论

收藏

微信

微博

QQ

```
static void cbpmci_set_ios(struct mmc_host *mmc, struct mmc_ios *ios)
{
    .....

    if (ios->power_mode==MMC_POWER_UP) {
        .....

    }

    else if (ios->power_mode==MMC_POWER_ON) {
        .....

    }

    else if (ios->power_mode==MMC_POWER_OFF) {
        .....

    }

    if (ios->clock)
    {
        /*此处的频率是以HZ为单位的*/

    }

    else
    {
        /*关闭时钟*/

    }

    if (ios->bus_width == MMC_BUS_WIDTH_4)
    {
        //4bit

    }

    else if (ios->bus_width == MMC_BUS_WIDTH_8)
    {
        //8bit

    }

    else
    {
        //1bit

    }
}
```

```
}

```



0

文章标签：[工作](#) [struct](#) [cmd](#) [command](#) [ios](#) [search](#)

个人分类：[嵌入式Linux](#) [写评论](#)

相关热词：[as编写](#) [编写如何编写文件](#) [编写录音软件编写](#) [xlat编写](#) [增删改查编写](#)

收藏

上一篇

编写自己的SD/MMC Host驱动（一）:注册

下一篇

用FPGA设计LCD 转 VGA



微博

大数据开发薪资多少？

随着时代的发展，大数据技术与运用越发的成熟，所应用的行业集中于互联网、金融、医疗、新能源、通信和房地产等行业，想要了解如何成为高薪的大数据开发工程师吗？需要.....

想对作者说点什么？

我来说两句

SD/MMC卡块设备驱动程序



9958

SD/MMC 卡组成的存储系统是很多嵌入设备的主要存储设备，相当于PC机的硬盘，在嵌入设备上的SD/MMC卡控制器通过MMC协...

Linux SD卡驱动开发(二) —— SD 卡驱动分析HOST篇



8423

回顾一下前面的知识，MMC 子系统范围三个部分：HOST 部分是针对不同主机的驱动程序，这一部分是驱动程序工程师需要根据自己...

linux2.6.28块设备mmc_sd卡初始化和识别流程及读写请求...CSDN博客

2018-6-26

```
//发送 CMD41 CMD55 读取 OCR 的值
#define SD_APP_OP_COND 41
/* bcr [31:0] OCR R3 */
err = mmc_send_app_op_cond(host, 0, &ocr);...
```

Linux SD卡驱动开发(三) —— SD 卡驱动分析CORE篇 - CSDN博客

2018-6-6

具体看看mmc_send_app_op_cond()的实现过程int mmc_send_app_op_cond(struct mmc_host *host, u32 ocr, u32 *rocr) { struct mmc_command cmd; cmd....

linux2.6.28块设备mmc_sd卡初始化和识别流程及读写请求流程



1958

```
////////////////////////////////////...
```

Linux SD卡驱动开发(三) —— SD 卡驱动分析CORE篇 - CSDN博客

2018-6-6

具体看看mmc_send_app_op_cond()的实现过程int mmc_send_app_op_cond(struct mmc_host *host, u32 ocr, u32 *rocr) { struct mmc_command cmd; cmd....

编写自己的SD/MMC Host驱动(二):工作过程和大结局 - CSDN博客

2018-7-19

```
err = mmc_send_app_op_cond(host, 0, &ocr);...mmc_go_idle让SD/MMC卡进入IDLE状态,如果你对协议比较熟的话,就是发送CMD0了...
```

Linux内核之mmc子系统-sdio



1.2万

现在的Linux内核中，mmc不仅是一个驱动，而是一个子系统。这里通过分析Linux3.2.0内核，结合TI的arm335x平台及omap_hsmm...

WiFi驱动(4)SDIO驱动SDIO卡的扫描



1672

SDIO卡的扫描流程

Linux SD卡驱动开发(三) —— SD 卡驱动分析CORE篇 - CSDN博客

2018-4-17

```
int mmc_send_app_op_cond(struct mmc_host *host, u32 ocr, u32 *rocr) { struct mmc_command cmd; cmd.opcode = SD_APP_OP_COND; /* #define...
```

Linux SD卡驱动开发(三) —— SD 卡驱动分析CORE篇 - CSDN博客

2018-5-25

int mmc_send_app_op_cond(struct mmc_host *host, u32 ocr, u32 *rocr) { struct mmc_command cmd;...

mmc驱动的读写过程解析

mmc io的读写从mmc_queue_thread()的获取queue里面的request开始。 先列出调用栈，看下大概的调用顺序，下面的内容主要阐...

mmc_init函数流程分析 - CSDN博客

2018-6-27

发送CMD8读取SD卡版本 err = mmc_send_if_cond(host);发送CMD55 + CMD41检查card工作状态 (retry一次) err = mmc_send_app_op_cond(host);...

Linux下sdio设备扫描过程 - CSDN博客

2018-7-16

对于cmd52命令是不响应的,在这种情况下,这两个命令对sdio设备也是没有任何作用...mmc_send_app_op_cond->(cmd55,acmd41) mmc_sd_init_card-> ...

mmc子系统

209

drivers\mmc\host\rtx-icr.c module_platform_driver driver_register(&rtx_icr_driver); rtx_icr_probe...

uboot源码——mmc驱动分析

1305

以下内容源于朱有鹏《物联网大讲堂》课程的学习，以及博客<http://www.cnblogs.com/biaohc/p/6409197.html>的学习整理，如有侵权...

mmc子系统总结 - CSDN博客

2018-5-20

mmc_attach_sd mmc_send_app_op_cond(host, 0, &ocr); //cmd41, 读取ocr register, ocr和power有关 mmc_attach_bus(host, &mmc_sd_ops); //...

mmc_init函数流程分析

109

源码参考九鼎科技X210开发板捆绑的BSP中提取的ubootint mmc_init(struct mmc *host)函数定义在uboot/drivers/mmc/mmc.c中1. 调...

linux下MMC/SD/SDIO驱动系列之二 ---- host注册过程（一）

9359

上篇文章说到了MMC/SD/SDIO（以下简称MMC）的驱动从大的方面来说分为主设备驱动和从设备驱动，那本文就来详细的讲述主...

编写自己的SD/MMC Host驱动（一）：注册

5638

网上已经有很多文章写了Linux SD/MMC的驱动的分析了，尤其是SAMSUNG系列的，估计用汗牛充栋来描写都不过分。俺只能说点...

s5p4418-linux MMC驱动子系统分析

2273

平台说明：s5p4418 SD/MMC控制器驱动：Synopsys DesignWare Dw_mmc-pltfm.c (drivers\mmc\host) 和Dw_mmc.c (drivers...

Linux设备驱动程序架构分析之MMC/SD（二）

1.3万

作者：刘昊昱 博客：<http://blog.csdn.net/liuhaoyut> 内核版本：3.10.1 一、s3cmci_ops分析 在上一篇文章中我们分析了Mini2440...



[下载CAMBLY]随时随地学英语
随时随地学习，把欧美外教装在口袋里！现在注册可领取15分钟免费课程！

linux下MMC/SD/SDIO驱动系列之三 ---- host注册过程（二）

1万

上篇文章说到了探测函数sdhci_s3c_probe，现在就来仔细分析这个函数的作用 在分析代码之前，先简要的概括一下这个函数的功...

Linux SD卡驱动开发(三) —— SD 卡驱动分析CORE篇

3752

废话不多说，直接切进主题： Linux在内核源码的drivers/mmc/core文件夹下为我们的提供了一系列SD卡的接口服务函数。可以...

SD卡驱动调试经验

3663

做SD/MMC卡驱动的项目也有大半年了，总觉得该总结些什么了。在这里，主要就是记录一些比较常见的问题及解决方法，以免再...

[sd card] mmc硬件总线扫描流程（以sd card为例）

1981

一、扫描mmc硬件总线扫描mmc硬件总线，也就是检测mmc硬件总线上是否有挂载card。更加通俗的，就是卡槽上是否有插入card...

SD/MMC 初始化及热插拔检测机制

1082

SD卡的技术是基于MultiMedia卡（MMC）格式上发展而来，SD卡与MMC卡保持着向上兼容。SD卡的内部结构主要分两部分：SD...

弥合信息鸿沟，共享知识社会
打造公益平台，传播公益资讯
gongyi.baidu.com

微信

微博

Linux设备驱动程序架构分析之MMC/SD（一）

2.5万

作者：刘昊昱 博客：http://blog.csdn.net/liuhaoyutz 内核版本：3.10.1 MMC MMC全称MultiMedia Card，由西门子公司和SanDisk...

SD卡驱动(详细介绍,不明白的人可以仔细看看了.有流程图)

2567

转载自：http://bbs.elecfans.com/infocenter.php?mod=space&uid=701087&do=blog&id=224468 一.SD/MMC卡...

个人资料



关注

原创	粉丝	喜欢	评论
26	86	11	44

等级： 博客 4 访问：12万+
积分：1456 排名：3万+



最新文章

- FPDLINK中I2C通信的巧妙设计以及I2C Str etch
- Robot：一个记录和回放Android input eve nt(touch)的工具
- Android Camera HAL V3 Vendor Tag及V 1，V3参数转换
- Android下增加service和对应的AIDL
- Android Uevent 分析，从kernel到framewor k

个人分类

Android 7篇



0



写评论



收藏



微信



微博



QQ

FPGA	1篇
其它嵌入式	5篇
嵌入式Linux	11篇
嵌入式小工具	1篇
...	...

展开

归档

2017年1月	1篇
2015年3月	1篇
2015年1月	1篇
2012年3月	2篇
2011年3月	1篇

展开

热门文章

- Android Uevent 分析，从kernel到framework
阅读量：20563
- SD/MMC 中的scatterlist
阅读量：7829
- 关于platform_device和platform_driver的匹配
阅读量：7531
- 编写自己的SD/MMC Host驱动（二）:工作过程和大结局
阅读量：7148
- OV7670 的SCCB (I2C)波形记录
阅读量：7143

最新评论

- Android Camera HA...
hbw1992322：看了之后还是没有懂怎么实现vendor_tag_ops中的方法。要在那里实现？一头雾水啊
- OV7670 的SCCB (I2C...
qinhu6431：哈哈 目测示波器是安捷伦的MDO系列的
- FPDLINK中I2C通信的巧妙设...
zhouyuanwei01：给贼哥点个赞！
- Android Uevent 分析...
Qidi_Huang：有收获，感谢分享！
- Robot：一个记录和回放And...
sksweet：哦，知道怎么用了，原来使用方法是在代码里边。生成可执行程序后，robot --help也可以看到...





0



写评论



收藏



微信



微博



QQ

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

👤 QQ客服 🗣 客服论坛

[关于](#) [招聘](#) [广告服务](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🐼 百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心