Linux内核中Nand flash 设备注册、初始化流程（AM4379）

**概述**

Linux 3.x的内核中，对设备驱动初始化的主要操作步骤分为两步：

1. 初始化设备相关的数据结构，申请设备空间，将设备注册到系统中，这部分的代码在对应ARM架构的内核代码中，如AM4379对应的代码放在linux\arch\arm\mach-omap2目录下，因为AM4379是ARMv7结构的，OMAP2平台；
2. 驱动的初始化。进行驱动的初始化时首先通过设备注册时对应的设备名称提取设备的私有数据，然后进行驱动层面的各个参数的读取、芯片配置、读写函数的实现和注册等工作，这部分的代码放在驱动目录下，如NAND Flash的驱动代码放在drivers\mtd\nand下；

**设备注册**

在AM4379上，NAND Flash在硬件上是挂接在GPMC总线上（在我们的AM4379核心板上，采用了一颗MICRON镁光的1GBytes的颗粒MT29F8G08A）,因此NAND设备是作为GPMC的一个子设备进行注册的。

Gpmc设备注册时（linux\arch\arm\mach-omap2\gpmc.c）系统调用gpmc\_probe函数，gpmc\_probe函数中完成GPMC各部分初始化后调用gpmc\_probe\_dt，读取dts（包括dtsi）文件中的参数；然后对gpmc下所有子节点进行遍历，包括nand\_child和fpga\_child（25所测控台项目中加入的fpga需求）。

Gpmc\_probe\_nand\_child函数对nand flash驱动进行初始化。读取dts数据，给gpmc\_nand\_data中的数据进行赋值，例如nand flash的ecc设置，数据宽度等。

然后将gpmc\_nand\_data传递给gpmc\_nand\_init函数（gpmc-nand.c），该函数主要完成nand的gpmc操作相关初始化，如读写时序（通过gpmc\_read\_settings\_dt函数从DTS文件中读取），寄存器设置，系统资源地址，ecc设置等。完成设置后将数据（gpmc\_nand\_data）保存在平台数据中（platform\_device\_add\_resources），然后进行设备注册（platrom\_device\_add）。

以上操作完成nand设备的注册（device），在进行设备分配（alloc）申请时，调用 platform\_device\_alloc函数，设备名参数定义为“omap2-nand”，这个需要和后续driver注册时用的设备名保持一致。

**Nand flash的初始化**

在Omap2.c中实现（linux/drivers/mtd/nand），当在系统中注册驱动时，omap\_nand\_probe函数首先进行nand设备相关数据的初始化，主要包括以下几个数据结构：

Info（omap\_nand\_info）

pdata（omap\_nand\_platform\_data）：这个数据从平台获取，就是之前注册设备时保存到平台的数据（gpmc\_nand\_data）

mtd（mtd\_info）

nand\_chip（nand\_chip）

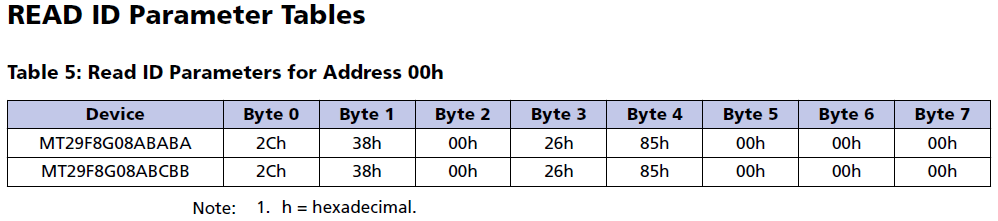
1. 首先读取设备注册时存入的数据（dev\_get\_platdata），放入Pdata；
2. 申请驱动内存空间，返回值赋给info；
3. 将info数据通过platform\_set\_drvdata与设备相关；
4. 对info内的数据进行赋值，初始化info/mtd/nand\_chip结构中的数据（mtd/nand\_chip作为info的成员进行指针赋值）；

调用nand\_scan\_ident，进行nand flash的识别和参数设置；

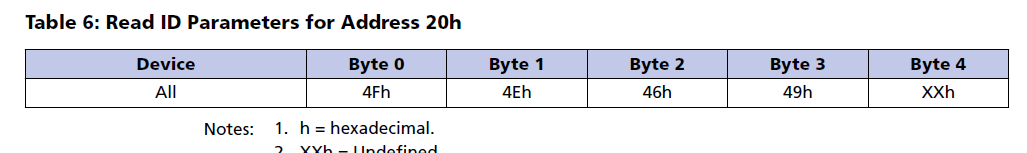
Nand\_get\_flash\_type函数解析（linux/drivers/mtd/nand/nand\_base.c，以MT29F8G08A为例）

1. chip->select\_chip，对应同一文件中的nand\_select\_chip函数，使能CE信号；
2. 复位芯片；
3. 读地址0的ID参数，制造商ID 0x2C，chip ID 0x38；

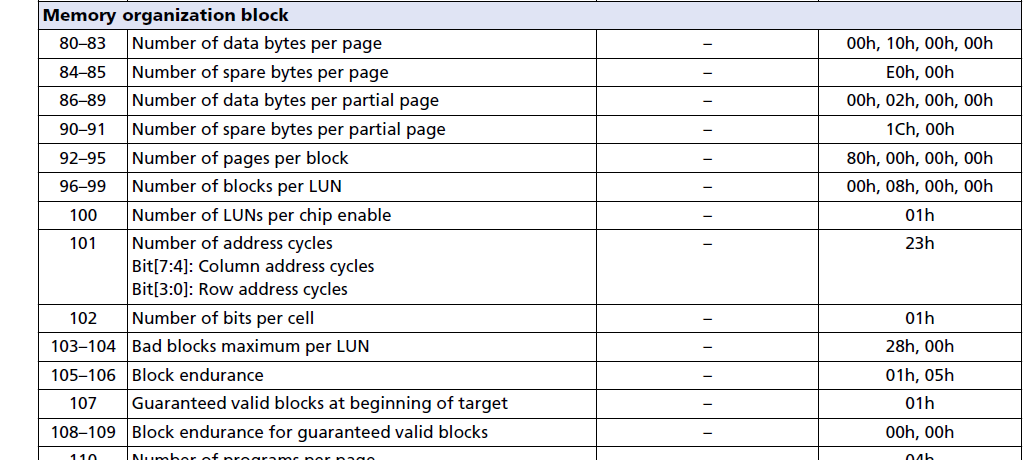
为了防止总线挂死，再进行一次读ID的操作，本次连续发起8次读操作，读取整个8Bytes的ID数据，读取完成后重新进行maf ID和chip ID的校验；



1. 如果设备指针type为空，则将缺省全局变量nand\_flash\_ids（结构数组）赋值给type；
2. 将读取的maf ID 和chip ID和nand\_flash\_ids中的数据进行比较；
3. 当在nand\_flash\_ids中没有找到对应flash的时（name或pagesize为空），调用nand\_flash\_detect\_onfi函数检查是否是支持ONFI（Open NAND Flash Interface）的芯片（MT29F8支持）；
4. 当检查到是支持ONFI的芯片时（读取ID指令，地址0x20），跳过ID decode 部分，跳转到ident\_done分支；



1. 如果是ONFI芯片，则读取芯片信息，进行芯片参数初始化：
   1. 读取parameter page（指令0xEC）
   2. 检查ONFI版本（MT29F8支持1.0，2.0，2.1）
   3. 芯片的block/page等信息是从parameter page 的第三部分“Memory organization block”中读取和计算出来的
   4. Mtd->writesize 即bytes per page，MT29F8的值为0x00100000



进行大小字序转换后为0x00001000，即4096；

mtd->writesize = le32\_to\_cpu(p->byte\_per\_page);

* 1. Erasesize和Oobsize的计算

/\*

\* pages\_per\_block and blocks\_per\_lun may not be a power-of-2 size

\* (don't ask me who thought of this...). MTD assumes that these

\* dimensions will be power-of-2, so just truncate the remaining area.

\*/

mtd->erasesize = 1 << (fls(le32\_to\_cpu(p->pages\_per\_block)) - 1);

mtd->erasesize \*= mtd->writesize;

mtd->oobsize = le16\_to\_cpu(p->spare\_bytes\_per\_page);

因为页数/块和块数/lane可能不是2的整数幂，所以先要进行转换；

Fls函数用于获取参数的最高有效bit位，在我们的应用中，pages per block=0x80，所以：

Erasesize = 0x80\*4096=0x80000=512K，即块擦除的最小单位是512Kbytes，因此在定义MTD分区的时候要注意最小的分区要>=512Kbytes；

OOBSize的大小在Micron的参数表中为spare bytes per page=0xE0=224字节；

1. 如之前所述，驱动进行类似的芯片参数读取和计算，完成flash芯片的各个参数的初始化。然后直接跳转‘ident\_done’标签处（如果是不支持ONFI的芯片，则要继续进行各个参数的计算；
2. 至此，nand\_get\_flash\_type函数基本完成了flash各个参数的读取和计算，在函数的最后部分，主要是对参数的检查和打印，然后返回上一级nand\_scan\_ident函数；
3. Nand\_scan\_ident函数继续进行芯片复位，读取ID的动作，完成芯片的初始化，然后返回omap\_nand\_probe函数；
4. Omap\_nand\_probe函数继续进行驱动层面的初始化，如dma xfer和ecc模式的设定，至此基本完成了nand flash的挂接和初始化工作。随后进行MTD分区挂接工作，不在本文的讨论范围内；

**Nand flash初始化及boot设置设计中的注意事项**

U-boot中需要修改的是对应的配置头文件（XXX.h，如u-boot\include\configs\am43xx\_evm.h）

1. 修改CONFIG\_SYS\_NAND\_PAGE\_SIZE宏定义，芯片的页大小，对应MT29F8为4096；
2. 修改CONFIG\_SYS\_NAND\_OOBSIZE宏定义，冗余信息区大小，对应MT29F8为224；
3. 确认CONFIG\_NAND\_OMAP\_ECCSCHEME和芯片中的一致，对应MT29F8是OMAP\_ECC\_BCH16\_CODE\_HW；
4. 修改CONFIG\_SYS\_NAND\_ECCPOS的定义，这部分可以从TI的wiki网站上查到，如[http://processors.wiki.ti.com/index.php/Linux\_Core\_NAND\_User's\_Guide#ECC\_schemes\_support](http://processors.wiki.ti.com/index.php/Linux_Core_NAND_User's_Guide%23ECC_schemes_support)
5. 修改MTDPARTS\_DEFAULT列表为实际linux分区，**注意最小分区大小为芯片的块擦除大小，MT29F8为512K**
6. 根据分区的地址修改CONFIG\_SYS\_NAND\_U\_BOOT\_OFFS的值，**这个是系统启动时UBOOT的入口**。

Kernel中需要修改dts文件（linux\arch\arm\boot\dts\am437x-gp-evm.dts）：

如前文所述，nand flash是挂接在gpmc总线下，因此在dts中要对应修改&gpmc节点中的内容：

1. 确保nand-ecc-opt设置正确“bch16”；
2. 总线宽度nand-bus\_width；
3. 各个时序参数设置正确；
4. Partition字段分区设置确保和上面u-boot中设置的一致；