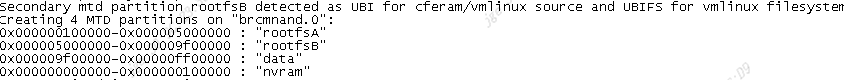
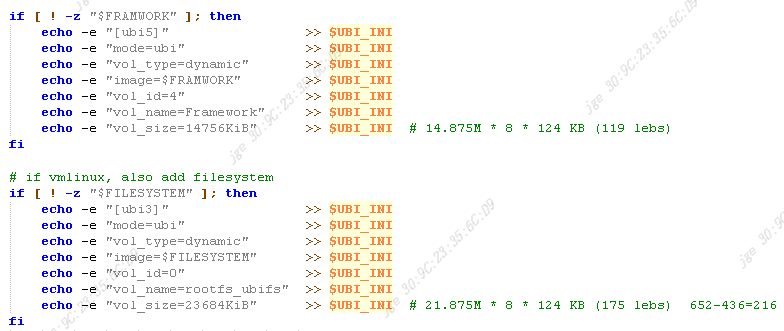
# UBI对应卷的设置

### 一、博通SDK502

1. 博通的项目的rootfs有2个分区，主备分区一样的大小,最终256M的nandflash被划分为 4个分区。对应一个mtd区(rootfsA/B)上分成7个ubi卷；
2. 看开机打印，看到mtd的分区，这个是在内核启动中代码写死的，可以搜打印“Creating 4 MTD partitions on”



1. 可以计算下rootfsA分区的实际大小：0x5000000-0x100000=0x4F00000=80896KB=79M
2. 再看下这个mtd分区被划分成7个卷，对应使用的配置脚本是sdk中/targets/buildUBI,这个脚本被调用后会生成ubi\_full.ini以及ubi\_full.ini\_A,截取buildUBI部分内容如下：



当前的的卷包括：

Ubi0 matadata0 1lebs 124kb

Ubi1 matadata0 1lebs 124kb

Ubi2:filestruct 31lebs 3844kb

Ubi3 filesystem 175lebs 21700kb //这个是rootfs卷容量

Ubi4 app\_ubifs 216lebs 26784kb

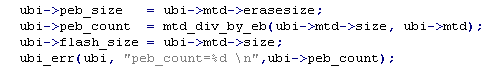
Ubi5 framwork 119lebs 14756kb

Ubi6 felix 68 lebs 8432kb

这里可以看到一个逻辑块（leb）是124kb，或者看开机打印也有。这个一般固定值是124kb。

这里简单计算下rootfsA分区总共lebs个数是80896/124=652. 但是实际可以按照652个leb分配吗？肯定是不行的

1. 在函数ubi\_init中调用ubi\_attach\_mtd\_dev-->io\_init-->mtd\_div\_by\_eb(ubi->mtd->size, ubi->mtd),在这个地方会根据mtd分区size计算得到总的实际的peb数peb\_count。



实际打印输出，只有632个peb，并不是计算得到的652。但即使是总的632个也不能全部拿来用，接着看为啥不能全用了这632个peb:

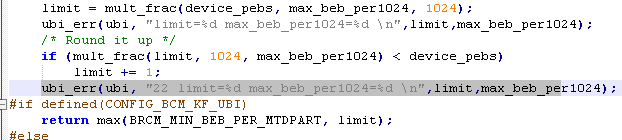


1. 在函数io\_init中还有个很重要的参数ubi->bad\_peb\_limit=get\_bad\_peb\_limit();这个参数会影响分区中有坏的peb块预留的peb数量。这个预留坏块的限制在BCM有宏控制，他们修改这个限制的大小：

CONFIG\_BCM\_KF\_UBI=y //当前定义了；bcm补丁采用的是device size是mtd分区的size，原始的内核代码size等于flash的整体size。

另外，bcm还定义了分区预留坏块peb门限最小值

BRCM\_MIN\_BEB\_PER\_MTDPART //目前是4，实际计算limit代码如下：



取的实际分区计算值和这个门限值的最大值，limit=13，参考实际打印。

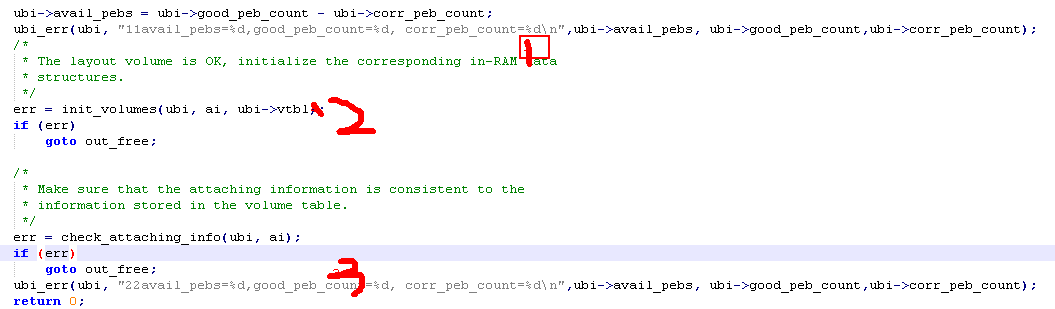


1. 在函数ubi\_attach中会调用scan\_all扫描整个mtd分区，然后得到对应的分区的坏块（bad\_peb\_count）以及崩溃块（corr\_peb\_count）。由此计算当前区的好块good\_peb\_count。

如下所示roofs对应ubi0区有1个bad peb：



1. 在ubi\_attach中再经过ubi\_read\_volume\_table计算获取可用的peb(avail\_pebs)，再经过ubi\_wl\_init与ubi\_eba\_init，这个可用的avail\_pebs逐渐减少，还要计算与bad peb相关的保留peb是否符合要求。
2. 先看ubi\_read\_volume\_table代码如下：在1处赋值得到总的可用avail\_pebs；



但是注意在2处init\_volumes中会根据各个卷用的peb来减少这个值。如下

init\_volumes（）

{  
 int i, reserved\_pebs = 0;

for (i = 0; i < ubi->vtbl\_slots; i++) {

vol->reserved\_pebs = be32\_to\_cpu(vtbl[i].reserved\_pebs);

reserved\_pebs += vol->reserved\_pebs;//累加，遍历每个卷，+镜像size用的peb个数

ubi\_err(ubi, "name %s, vol->reserved\_pebs %d reserved\_pebs=%d",

vol->name, vol->reserved\_pebs, reserved\_pebs);

}

vol->reserved\_pebs = UBI\_LAYOUT\_VOLUME\_EBS;//累加。宏定义预留+2

reserved\_pebs += vol->reserved\_pebs;

ubi\_err(ubi, "11 vol->reserved\_pebs %d reserved\_pebs=%d avail\_pebs=%d",

vol->reserved\_pebs, reserved\_pebs, ubi->avail\_pebs);

if (reserved\_pebs > ubi->avail\_pebs) {

ubi\_err(ubi, "not enough PEBs, required %d, available %d",

reserved\_pebs, ubi->avail\_pebs);//注意此处的判断打印说明：一个mtd分区总的可用avail\_pebs（=实际总peb\_count-bad\_peb\_count- corn peb）不能比各个卷分配zise用的peb总数+2要少。到此时，也就是说总avail\_pebs-2就是所有卷可以分配的peb数。但是这只是目前为止，还有限制，继续看

if (ubi->corr\_peb\_count)

ubi\_err(ubi, "%d PEBs are corrupted and not used",

ubi->corr\_peb\_count);

return -ENOSPC;

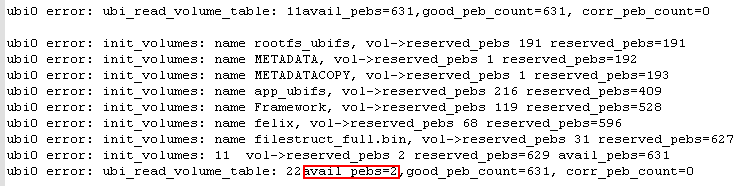
}

ubi->rsvd\_pebs += reserved\_pebs;

ubi->avail\_pebs -= reserved\_pebs; //此处减去各个卷所用的peb剩下的ubi->avail\_pebs

}

看实际输出打印：



1. 再看ubi\_wl\_init()中，其中涉及peb限制代码段如下：

{ reserved\_pebs = WL\_RESERVED\_PEBS;//WL模块约束预留宏WL\_RESERVED\_PEBS=1

ubi\_fastmap\_init(ubi, &reserved\_pebs);//此处fastmap有可能减少，受宏CONFIG\_MTD\_UBI\_FASTMAP控制，MTK有进入（再减2），bcm没有

ubi\_err(ubi, "00 ubi->avail\_pebs=%d, ubi->rsvd\_pebs= %d reserved\_pebs=%d",

ubi->avail\_pebs, ubi->rsvd\_pebs, reserved\_pebs);

if (ubi->avail\_pebs < reserved\_pebs) {

ubi\_err(ubi, "00 no enough physical eraseblocks (%d, need %d)",

ubi->avail\_pebs, reserved\_pebs);//此处说明，扣除各个卷后剩余的avail\_pebs不能小于1

if (ubi->corr\_peb\_count)

ubi\_err(ubi, "%d PEBs are corrupted and not used",

ubi->corr\_peb\_count);

err = -ENOSPC;

goto out\_free;

}

ubi->avail\_pebs -= reserved\_pebs; //剩余avail\_pebs再扣除1

ubi->rsvd\_pebs += reserved\_pebs;//保留rsvd\_pebs再+1

ubi\_err(ubi, "11 ubi->avail\_pebs=%d, ubi->rsvd\_pebs= %d reserved\_pebs=%d",

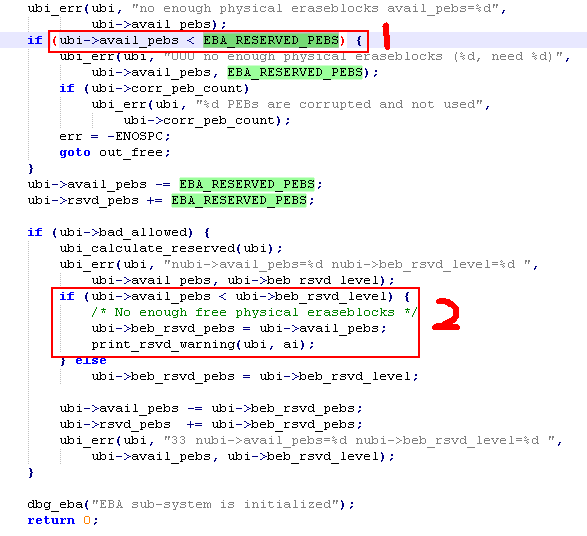
ubi->avail\_pebs, ubi->rsvd\_pebs, reserved\_pebs);

}

看实际打印：



1. 最后看函数ubi\_eba\_init()中的限制：



1处要求：剩余avail\_pebs不能小于1（EBA\_RESERVED\_PEBS=1），否则要报错。

然后剩余avail\_pebs再扣除1

保留rsvd\_pebs再+1

2处要求：剩余avail\_pebs不能小于bed\_rsvd\_level(13),否则有告警print\_rsvd\_warning().这个13是这样赋值的：

ubi\_calculate\_reserved()

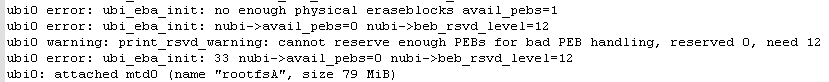
{

ubi->beb\_rsvd\_level=ubi->bad\_peb\_limit-ubi->bad\_peb\_count;

}

记得之前说过有个很重要的参数就是bad\_peb\_limit，计算值是13，当前bad\_peb是1，最终beb\_rsvd\_level=12；

看对应实际打印：



此处的判断是：当前实际的可用avail\_pebs（0）<beb\_rsvd\_level（12）的时候，此时赋值beb\_rsvd\_pebs=avail\_pebs，并打印告警‘cannot reserve enough PEBs for bad PEB handling’，此处仅仅是告警，不影响最终的attach。

要是实际可用的比12多，此时beb\_rsvd\_pebs=beb\_rsvd\_level（12）；

然后就是avail\_pebs再扣除beb\_rsvd\_pebs。

综合上面

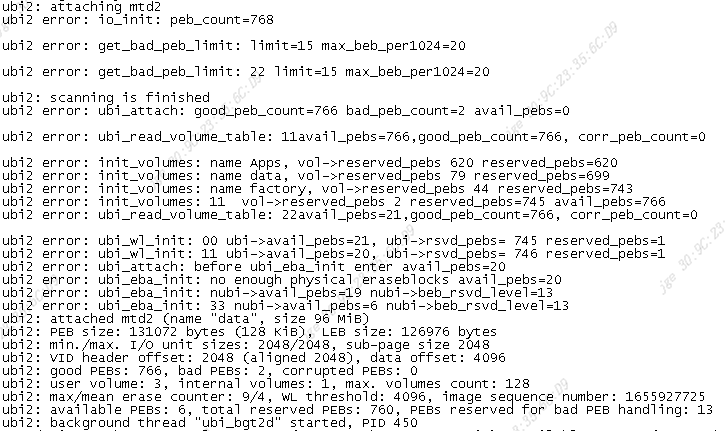
peb\_count(632) - bad\_peb\_count(1) - corn peb(0)- total-used-count(?) - UBI\_LAYOUT\_VOLUME\_EBS(2) - WL\_RESERVED\_PEBS(1) - EBA\_RESERVED\_PEBS(1) - beb\_rsvd\_level（12）>= 0 最后12 pebs可以用掉，但是会报警告，这个是给坏块处理的预留区域。且这个受bad\_peb\_limit计算的限制，最小是4。

这样反过来计算下我们可用的total-used-count=632-1-2-1-1-12=615

去掉已经给运营商划分分卷等其他卷metadata(1\*2)-filestruct(31)- app\_ubifs(216) - framwork(119) - felix(68) = 436;

615-436=179个，最大留个filesystem就是179个leb。

可以看下数据分区的对应打印：



该MTD分区实际是96M,内核计算是有768可用，bad peb 2个，Corr 0个，对应该分区的limit是15，再减去bad 2就是13，对应bad\_rsvd= 13。

Apps卷是620，data卷79，factory卷44，总计743.

计算得到最终空闲ubi->avail\_pebs=768-2(bad)-743(total-used)-13(bad\_rsvd)-4(宏)=6，对应上打印。

### 二、MTK 5.4的内核

1. 对比下MTK 5.4的内核ubi\_attach函数，init\_volumes，UBI\_LAYOUT\_VOLUME\_EBS=2

ubi\_eba\_init函数以及 EBA\_RESERVED\_PEBS=1；

ubi\_wl\_init以及WL\_RESERVED\_PEBS=1，等都保持不变。

所以一个分区总的可用的peb数量一般是总数- bad(1)- 4- bad\_rsvd(12). 这个经验可以通用。

其中bad是根据分区实际情况确定，没有坏块就是0；

bad\_rsvd是根据算法确认的，默认是根据falsh整体size计算，Bcm打了补丁根据分区size计算，跟分区大小有关系，最小是4.

### 三、MTK的4.4内核

1. 对比mtk的4.4的电信版本，它唯一区别时定义了宏CONFIG\_MTD\_UBI\_FASTMAP，导致ubi\_wl\_init计算reserved\_pebs时进入会再次-2；

ubi\_fastmap\_init(ubi, &reserved\_pebs);//MTK有进入（再减2），bcm没有

static inline void ubi\_fastmap\_init(struct ubi\_device \*ubi, int \*count)

{

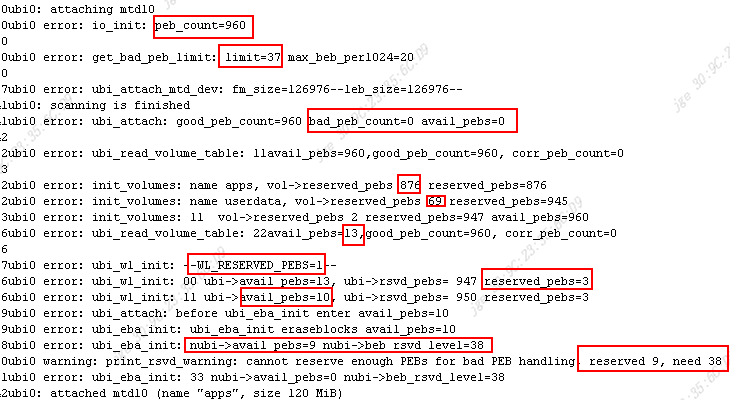
/\* Reserve enough LEBs to store two fastmaps. \*/

\*count += (ubi->fm\_size / ubi->leb\_size) \* 2;

INIT\_WORK(&ubi->fm\_work, update\_fastmap\_work\_fn);

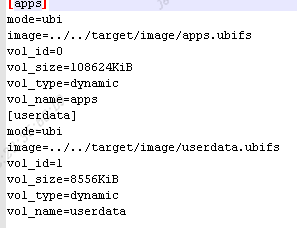
} //fm\_size=126976, leb\_size=126976(124kB)

具体打印参考如下：



解读一下：总可用peb：peb\_count=960,bad 和 corr都是0；good\_peb\_count=960;

apps分卷876，userdata分卷69，参考制作分卷镜像脚本ImageMaking.sh中mk\_ubi\_image使用配置文件ubinize\_conf\_partition.conf：



其中108624/124=876, 8556/124=69刚好跟上面匹配。

经过init\_volumes后960-876-69-UBI\_LAYOUT\_VOLUME\_EBS(2)=13;

然后经过ubi\_wl\_init后13-WL\_RESERVED\_PEBS-2(reserved\_pebs[ubi\_fastmap\_init])=10;

最周经过ubi\_eba\_init后10-EBA\_RESERVED\_PEBS(1)=9;该分区的limit是38，期望预留是38，所以这9个peb全部预留，同时还有告警‘cannot reserve enough PEBs for bad PEB handling’。相对bcm版本没有定义CONFIG\_MTD\_UBI\_FASTMAP，MTK版本要多减去2个peb，总计减去6个（宏定义4个）。