MBoot SOP

1. ubi part UBI常见错误范例
2. bootcmd相关的常见不开机情况
3. UBIFS中的数据损坏
4. 常见UBIFS中数据损坏打印的初步分析
5. 插著debug board 不接電腦開不了機(Rx接地)
6. 如何關掉L2 Cache
7. TFTP燒SN時出現UBI error: ubi\_create\_volume: cannot create
8. Make 時出現make: \*\*\* [include/config/MARKER] Error 255錯誤
9. **ubi part UBI常见错误范例**

下面的图是分别修改ENV mtdparts中UBI的分区size和分区名的结果。如果分区size或分区名不对，即使NAND上没有任何数据被破坏，系统肯定也是无法开机的。所以出现这类问题时可以首先printenv看一下mtdpart的内容是否正确。



ubi part UBI失败，虽然找到了UBI分区，但由于扫描的范围变小了，导致一些数据没有读取到而报错。



由于mtdparts中已经没有名字为UBI的分区了，因此ubi part UBI失败。

Hit any key to stop autoboot: 0

0x00220000-0x07240000 : "mtd=3"

UBI error: ubi\_read\_volume\_table: the layout volume was not found

UBI error: ubi\_init: cannot attach mtd0

UBI error: ubi\_init: UBI error: cannot initialize UBI, error -22

UBI init error -22

Error, no UBI device/partition selected!

## Booting image at 81000000 ...

Bad Magic Number

上面也是一个典型的错误打印，volume table没有找到导致ubi part UBI失败，从而导致从UBI中读取kernel失败，最终导致bootm时报bad magic number错误。

0x00000000-0x0fa00000 : "mtd=0"

UBI error: ubi\_read\_volume\_table: the layout volume was not found

UBI error: ubi\_init: cannot attach mtd0

UBI error: ubi\_init: UBI error: cannot initialize UBI, error -22

UBI init error -22

UBIFS error (pid 0): ubifs\_get\_sb: cannot open "ubi:FACTORY", error -19

Error reading superblock on volume 'ubi:FACTORY'!

ubifs\_mount FAIL !!

这也是一个类似的打印，由于UBI init的失败，自然导致后面ubifsmount FACORY的失败，从而导致开机的一些功能不正常。

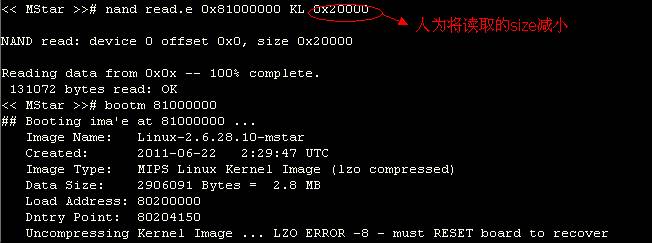
ENV不正确，NAND数据损坏，NAND几乎是一颗空片子（没有写入数据的oxff状态），都有可能导致这类开机异常。碰到这类问题，首先应该检查的是printenv看到的结果是否正常，这是最容易确认的一点。

1. **bootcmd相关的常见不开机情况**

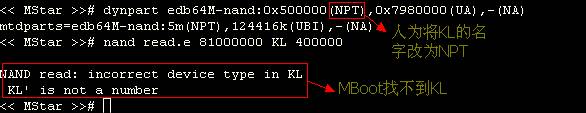
常见的不开机情况是解压失败，或是kernel的magic number不对。这种情况表明用nand read.e或是ubi read读出的kernel不对。

遇到这种情况有两个怀疑方向：一是NAND上存放的kernel真的被破坏了；二是kernel本身没有损坏，但nand read.e或ubi read读错了地方，或是读取的数据长度短了。比如使用了nand read；ENV中的分区设置不对；读取命令中的长度（比如0x400000）比kernel的实际大小要小等等，都会导致这类问题。

下面通过人为的破坏某些参数，重现了一些常见的不开机场景。



上图是人为的改小uImage.lzo的读取size，由于一个不完整的读取而导致kernel在解压的时候报错，系统有时候会因此反复reset。



再比如分区参数不对，mtdparts中没有KL分区，或是起了另外一个名字，导致nand read.e 81000000 KL 400000报错，找不到KL分区，也会导致开机失败。



uImage.lzo在编译完成后会在头部自带一个magic number，bootm如果找不到这个magic number也会报错而导致不开机。上图是人为制造了这样一个出错场景。

EEPROM @0x0 write: addr 874bfefc off 1fec count 1 ... cmd: md.b 0x874bfefc 0x1

874bfefc: 00 .

done

keypad\_pressed\_1 is:0

keypad\_pressed\_2 is:0

keypad\_pressed is [0]

<< MStar >>#

上面是一个典型的没有bootcmd而导致的不开机打印，mboot走完自己的流程后没有发现bootcmd，于是停在了命令行。一般来说ENV中没有bootcmd是ENV本身损坏而使用了默认ENV导致的（不仅仅是bootcmd的损坏），而如果默认ENV也是无法开机的设置就会出现这样的情况。

1. **UBIFS中的数据损坏**

kernel中最常见的出错情况是ubifs报出CRC error的错误和bad node type，但没有ECC错误。表现形式可能是某个partition mount失败（最严重的情况是rootfs mount失败直接导致不开机），某个文件读取失败（这个一般影响某个功能不能正常使用，但不影响开机）。

出现这种情况说明数据的出错发生在这部分数据写入NAND之前（ECC是FCIE按已经出错的数据计算出来的，所以ECC不报错）。

如果损坏的是rootfs分区，则由于没有根目录会直接导致不开机；如果损坏的是其它分区或某个文件，则一般只是影响某些功能，有可能不会导致死机。也许出错的根源是一样的，但损坏的数据部位不同而导致现象不同。

根据之前的经验，CRC error的具体数据损坏形式是五花八门，只能具体问题具体分析。

ENV和BBT内部的地址都是绝对的，所以比较容易分析。UBI内部，文件系统是按逻辑地址来访问的，并不一定是物理的从前向后写NAND。另外，每次开机后，文件系统内部也会有各种写的动作，所以，每开一次机，NAND上的内容都会不同。如果问题机器已经开过机，则其image和原始image不一致不能说明任何问题（正常的机器也一定是不一致的）。

1. **常见UBIFS中数据损坏打印的初步分析**

UBIFS error (pid 1): ubifs\_check\_node: bad node length 8

UBIFS error (pid 1): ubifs\_check\_node: bad node at LEB 52:36864

UBIFS error (pid 1): ubifs\_scanned\_corruption: corruption at LEB 52:36864

UBIFS error (pid 1): ubifs\_recover\_leb: LEB 52 scanning failed

VFS: Cannot open root device "ubi:RFS" or unknown-block(0,0)

Please append a correct "root=" boot option; here are the available partitions:

1f00 128 mtdblock0 (driver?)

1f01 253952 mtdblock1 (driver?)

1f02 8064 mtdblock2 (driver?)

Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)

上面是一个典型的出错打印，最后一句kernel panic只是告诉我们rootfs mount失败，此外就没有任何有用的信息了。顺着往前找，我们可以看到几个UBIFS error。

**UBIFS error (pid 1): ubifs\_check\_node: bad node length 8**

这一句表明UBIFS读取某个node时发现这个node的length为8，很显然，UBIFS期望的length不是8.

**UBIFS error (pid 1): ubifs\_check\_node: bad node at LEB 52:36864**

这一句是最有用的信息，它直接告诉我们出错数据的位置，也就是LEB 52:36864，即损坏数据位于RFS分区的第52个逻辑块的第36864 byte开始的地方，这是进一步分析问题的入口。

至于怎么找到损坏的具体数据，一般有两种方法：

一是修改mboot，将u u-boot-1.1.6\include\ubi\_uboot.h中“CONFIG\_MTD\_UBI\_DEBUG”、“CONFIG\_MTD\_UBI\_DEBUG\_MSG”、“CONFIG\_MTD\_UBI\_DEBUG\_MSG\_BLD”三个宏前面的“#undef”改为“#define”，开机的时候，ubi part UBI就会给出大量的打印，主要是LEB和PEB的映射关系；

二是用前面提到的办法直接将整个nand内容dump出来，这是最彻底的方法，但需要时间，客户端传递几百MB的数据往往也比较吃力。不过后期的分析则会方便很多。

1. **插著debug board 不接電腦開不了機(Rx接地)**

基本上因為是共pin Rx不建議接地，designer的解釋如下

this pad DDCA\_CK should stay at 1’b1 in normal mode.

When it was tied to 1’b0, the following function are all broken.

assign DDCA\_SCLIn\_adc = PAD\_DDCA\_CK\_C; // ddca function

assign mcu\_iic\_scli = PAD\_DDCA\_CK\_C; // i2c for MSTV TOOL and ISP TOOL

assign pad2sbus\_csz = PAD\_DDCA\_CK\_C; // sbus for test purpose

assign pad2dbus\_ALE = PAD\_DDCA\_CK\_C; // dbus for test purpose

assign p30i = PAD\_DDCA\_CK\_C; // 51 GPIO

assign hk51\_rxd0 = reg\_hk51\_uart0\_en? PAD\_DDCA\_CK\_C: 1'b1; //51 UART RX

assign Uart\_Rx0\_IN = reg\_uart\_rx\_enable? PAD\_DDCA\_CK\_C: 1'b1; // PM UART RX, PM UART0

如有Rx接地情況發生只能用workaround方式解決，開不了機都是在會停在console init()，暫時解法是一進init function先把Enable UART Rx via PAD\_DDCA for dig\_mux 拉高，最後再把拉低。

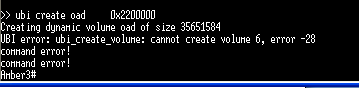
參考CL # 569167 CL# 569226 CL#570170

1. **如何關掉L2 Cache**

cid:image001.gif@01CC664C.639E05F0

可以參考以上項目：menuconfig  Platform Configuration  Extended CPU Configuration

1. **TFTP燒SN時出現UBI error: ubi\_create\_volume: cannot create**



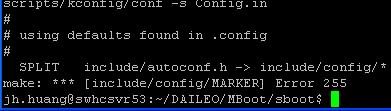
修改ubifs\mscript\ set\_partition

#34MB ubi create oad 0x2200000 改為 0x2000000

ubifs\mscript\ set\_partition [[oad

ubi write 80400000 oad 0x2200000改為 0x2000000

1. **Make 時出現make: \*\*\* [include/config/MARKER] Error 255錯誤**



移除DAILEO\MBoot\sboot\include下所有檔案在情新執行一次make menuconfig 有可能可以改善