SD卡分区详解

2018年03月08日

下午 06:19

**MBD主引导记录和DPT分区表**

参考资料:[MBR分区结构、DPT分区表、EBR扩展引导](http://www.blogfshare.com/mbr-dpt-ebr.html)

EasyARM IMX280A开发板的SD卡分区使用**MBR**格式记录分区信息.MBR信息位于SD卡第一个扇区中,

并且该扇区的最后两个字节必须是 **0x55,0xAA**才能被开发板识别为可启动的SD卡.

**MBR主引导记录格式:**



**DPT磁盘分区结构信息:**

DPT从MBR的第**0x01BE**字节偏移地址处开始,每个分区一个分区表,每个分区表占**16**个字节,MBR型分区结构最大支持4个磁盘分区.



**常见文件系统标志:**

[](https://www.cnblogs.com/aidonzhang/p/5295153.html)

**关于磁头,柱面,扇区的计算**

参考资料: [百度百科:硬盘分区表](https://baike.baidu.com/item/%E7%A1%AC%E7%9B%98%E5%88%86%E5%8C%BA%E8%A1%A8/108750?fr=aladdin)

**MBR的大小端模式:**

MBR格式采用的是大端模式,即数据的高字节保存在内存的低地址中，而数据的低字节保存在内存

的高地址中.比如：如果分区表的分区起始相对扇区号为(3F 00 00 00)，转换为十进制前要先反一下字节顺序，

*为(00 00 00 3F)然后在转换为十进制，即63.*

**逻辑扇区号与(柱面，磁头，扇区)的相互转换：**

令L=逻辑扇区号, C=柱面号, H=磁头号, S=扇区号.

每道最大扇区数 = 63.

每柱面最大磁头数 = 255.

每柱面扇区数 = 每道扇区数 \* 每柱面磁头数 = 63x255 = 16065.

柱面号下标从0开始, 磁头号[0-254], 扇区号[1-63], 逻辑扇区号下标也从0开始.

(柱面,磁头,扇区)转换成逻辑扇区号的公式为:

**L = C × 16065 + H × 63 + S - 1;**

**SD卡分区实例分析**

**如图所示为已经制作好的可引导SD卡的第一扇区,其DPT分区表中有三个DPT分区条目.**

第四个分区表条目数组全为0,其文件系统标志位为0,表示其不是一个正常的DPT条目.

计算机生成了可选文字:
可 移 磁 量 1 
Offset （ ， ， 
0 0 0 0 0 010 0 
0 0 0 0 0 0110 
0 0 0 0 0 012 0 
0 0 0 0 0 013 0 
0 0 0 0 0 014 0 
0 0 0 0 0 015 0 
0 0 0 0 0 0160 
0 0 0 0 0 0110 
0 0 0 0 0 01 E 0 
0 0 0 0 0 019 0 
0 0 0 0 0 01 0 
0 0 0 0 0 0130 
0 0 0 0 0 01 CO 
0 0 0 0 0 01 DO 
0 0 0 0 0 01 FO 
01 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
51 
51 
0 0 
0 2 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 三 
5 3 
10 
0 0 
0 3 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
弓 C 
€ D 
0 0 
0 4 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
DE 
E E 
0 0 
0 5 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
51 
51 
DS 
0 0 
0 6 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
01 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
01 
3 二 
3 
0 0 
0 E 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
DO 
DO 
0 0 
0 9 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 3 
0 0 
0 0 
03 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
3 0 
0 E 
0 0 
0 0 
0 C 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
DO 
0 0 
2 0 
0 0 
0 D 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
OE 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
三 0 
0 0 
0 0 
三 三 
0 F 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 弓 
0 0 
二 = 0 
= 03m30 
aQ 
IE 
第 
0 扇 区 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 0 
0 5 
0 0 

第一分区,主分区,W95 FAT32文件系统(**0x0B**):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 80 | 04 | 05 | 00 | 0B | 4C | FB | 51 | 00 | 01 | 00 | 00 | 00 | 30 | D0 | 00 |

|  |  |  |
| --- | --- | --- |
| 分区起始磁头号H | 0x04 | 4 |
| 分区起始扇区号S | 0x05&0x3f=0x05 | 5 |
| 分区起始磁柱号C | ((0x05&0xc0)\*0x04)+0x00 = 0x00 | 0 |
| 分区起始逻辑扇区号 | 0\*16065+4\*63+5-1=256 | 0x100 |
| 分区结束磁头号H | 0x4c | 76 |
| 分区结束扇区号S | 0xfb&0x3f=0x3b | 59 |
| 分区结束磁柱号C | ((0xfb&0xc0)\*0x04)+0x51 = 0x0351 | 849 |
| 分区结束逻辑扇区号 | 849\*16065+76\*63+59-1=13644031 | 0xd030ff |

第二分区,逻辑分区,OnTrack DM6 Aux文件系统(**0x53**):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 00 | 4C | FB | 51 | 53 | 6D | DF | 51 | 00 | 31 | D0 | 00 | 03 | 08 | 00 | 00 |

|  |  |  |
| --- | --- | --- |
| 分区起始磁头号H | 0x4c | 76 |
| 分区起始扇区号S | 0xfb&0x3f=0x3b | 59 |
| 分区起始磁柱号C | ((0xfb&0xc0)\*0x04)+0x51 = 0x0351 | 849 |
| 分区起始逻辑扇区号 | 849\*16065+76\*63+59-1=13644031 | 0xd030ff |
| 分区结束磁头号H | 0x6d | 109 |
| 分区结束扇区号S | 0xdf&0x3f=0x1f | 31 |
| 分区结束磁柱号C | ((0xdf&0xc0)\*0x04)+0x51 = 0x0351 | 849 |
| 分区结束逻辑扇区号 | 849\*16065+109\*63+31-1=13646082 | 0xd03902 |

第三分区,逻辑分区,OPUS文件系统类型(**0x10**):

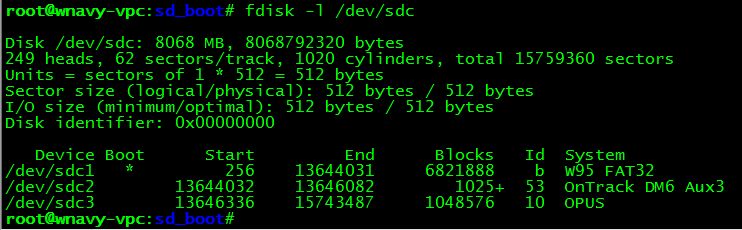
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 00 | 71 | E0 | 51 | 10 | FB | E8 | D3 | 00 | 3A | D0 | 00 | 00 | 00 | 20 | 00 |

|  |  |  |
| --- | --- | --- |
| 分区起始磁头号H | 0x71 | 113 |
| 分区起始扇区号S | 0xe0&0x3f=0x20 | 32 |
| 分区起始磁柱号C | ((0xe0&0xc0)\*0x04)+0x51 = 0x0351 | 849 |
| 分区起始逻辑扇区号 | 849\*16065+113\*63+32-1=13646335 | 0xd039ff |
| 分区结束磁头号H | 0xfb | 251 |
| 分区结束扇区号S | 0xe8&0x3f=0x28 | 40 |
| 分区结束磁柱号C | ((0xe8&0xc0)\*0x04)+0xd3 = 0x03d3 | 979 |
| 分区结束逻辑扇区号 | 979\*16065+251\*63+40-1=15743487 | 0xF039FF |

实际上我们手动计算出来的**分区起始逻辑扇区号**和**分区结束逻辑扇区号**与DPT条目中的最后8字节,

即**分区起始相对扇区号**和**分区总的扇区数**结果是一样的,这可以用于校验我们的计算结果.

在Linux系统中查看该SD卡的分区信息,和刚才分析的结果一致:



**SD卡引导固件分区详细信息**

下图是官方芯片手册中SD卡主引导记录MBR的说明：

计算机生成了可选文字:
12 ． 11 ． 2 Master Boot Record (MBR) Media Format 
If the eFuse media format mode is MBR MEDIA FORMAT, then ROM expects a valid 
master boot record (MBR) to be present on the first block of media. The MBR IS Identified 
by Its signature located at offset Ox 1 FE ofthe first sector. The partition table IS stored at 
address OxlBE. The Freescale firmware partition IS Identified by MBR SIGMATEL ID 
at an offset 0x04 from partition table. The firmware partition's start block address IS located 
at offset 0x08 offirmware partition entry Of partition table. 
Field 
MBR Signature 
MBR SIGMATEL 旧 
Value 
Ox55AA 
The first block offirmware partition contains BCB allowmg multiple copes offirmware 
to reside Inside firmware partition and to support redundant boot feature of ROM. Refer to 
Boot Control Block (BCB) Data Structure for a detailed view of BCB data structure and its 
use. A11 firmware copies specified in BCB should be located Inside the firmware partition. 

官方芯片手册中介绍说,如果SD卡的MBR分区表条目中某一条的**文件系统标志位***(从分区表条目起址偏移0x04字节处)*为

**0x53('S')**,则表明该分区为**SD卡引导固件分区**, 并且该**引导固件分区**的**逻辑地址(单位:扇区)**存放在该分区表条目从起始

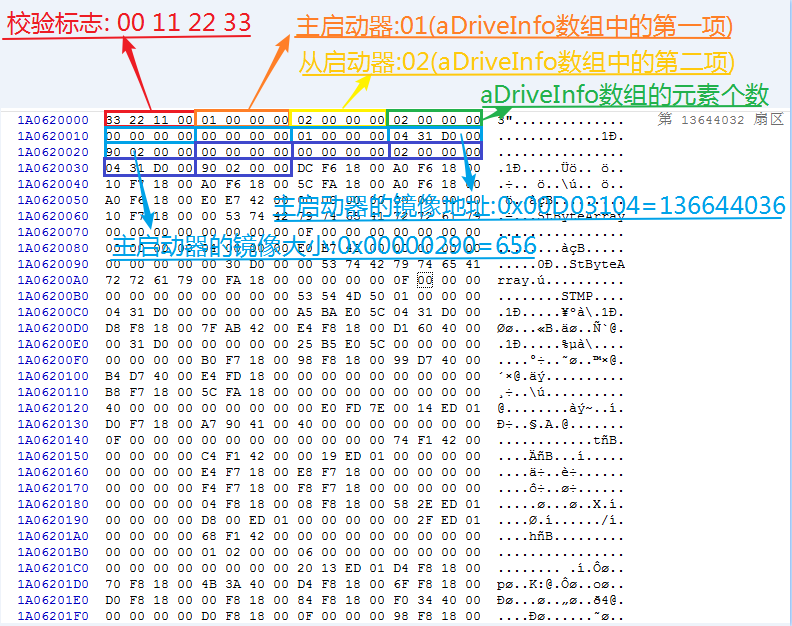
偏移**0x08**字节处, 长度4字节.因此可以看出,系统**引导固件分区**在该SD卡的第二分区上,并且该**引导固件分区**的逻辑扇区地址为:

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 31 | D0 | 00 |

由于是大端模式,所以实际为:

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | D0 | 31 | 00 |

即 在第**0x00d03100=13644032**个扇区处.跳到该扇区处看一看:



引导固件分区的第一个扇区用于存放系统**引导控制块(BCB)**,该BCB块的定义如下：

计算机生成了可选文字:
12 ． 11 ． 1 Boot Control Block (BCB) D ata Structure 
The design of BCB IS to allow multiple copes offirmware to be stored on media each 
Identified by Its umque tag. The tags can be defined either by the user or the firmware 
download application. The ROM is only Interested In user-defined primary and secondary 
boot tags. The ROM loads primary firmware, IfROM REDUNDANT BOOT persistent 
bit IS not set. otherwise It loads a secondary Image, providing support for a redundant boot. 
The config block has the followmg format: 
Chip Select, RO 卜 ， does not S e 工 t 
Always sys tem drive, ROM doe S not S e it 
Drive Tag 
Start sector/block address 0 f firmware. 
Not used by ROM 
Signature 0 x 0 0112233 
Primary boot drive 工 dent 工 fled by this tag 
Secondary boot drive Identified by this tag 
Num elements in aFWS1zeLOC array 
Let array aDriveInfo be last 工 n th 工 S data 
structure to be able to add mo r e dr 工 ves 工 n future 
wIthout changlng ROM code 
typedef struct 
工 nt32 t 
工 nt32 t 
工 nt32 t 
工 nt32 t 
u 工 nt32 t 
} Dr 工 velnfo t ： 
typedef struct 
工 nt32 t 
工 nt32 t 
工 nt32 t 
u 工 nt32 t 
DrIvelnfo 
} ConflgBIock 
t 
t ， 
Drivelnfo t 
u32Ch1pNum; 
u32Dr1veType; 
32Tag ： 
乙32F 工 rstSectorNumber; 
u32SectorCount; 
Conf 工 gBIock t 
乙32s 工 gnature; 
u32Pr1maryBootTag; 
u32SecondaryBootTag; 
u32NumCop 工 es; 
aDr 工 velnfoll ： 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 
/ / ！ < 

计算机生成了可选文字:
The driver first verifies the signature and version then searches all NumReg10ns for the 
appropriate tag. The following table shows the expected values for these parameters. 
Table 12 ． 29 ． Media Config Block Parameters 
Field 
Signature 
u32PrimaryBootTag 
u32SecondaryBootTag 
u32NumCopies 
aDrivelnfo 
Value 
0x00112233 
User-defined pnmary boot firmware tag 
User- defined secondary boot tag 
Number Of firmware copies present in array aDrivelnfo 
Each element in array describes the tag and start address 
for the image 

|  |
| --- |
| typedef struct \_DriveInfo\_t  {      uint32\_t u32ChipNum; //!< Chip Select, ROM does not use it      uint32\_t u32DriveType; //!< Always system drive, ROM does not use it      uint32\_t u32Tag; //!< Drive Tag      uint32\_t **u32FirstSectorNumber**; //!< Start sector/block address of firmware.      uint32\_t **u32SectorCount**; //!< Not used by ROM  } DriveInfo\_t;    typedef struct \_ConfigBlock\_t  {      uint32\_t u32Signature; //!< Signature 0x00112233      uint32\_t u32PrimaryBootTag; //!< Primary boot drive identified by this tag      uint32\_t u32SecondaryBootTag; //!< Secondary boot drive identified by this tag      uint32\_t u32NumCopies; //!< Num elements in aFWSizeLoc array      DriveInfo\_t **aDriveInfo**[]; //!< Let array aDriveInfo be last in this data      //!< structure to be able to add more drives in future      //!< without changing ROM code  } ConfigBlock\_t; |

由以上可知，BCB控制块从引导分区的第一个扇区的0地址处开始存放,

BCB块中的**aDriveInfo[0].u32FirstSectorNumber**用于存放主引导镜像文件(*imx28\_ivt\_uboot.sb*)的**逻辑偏移地址(单位:扇区)**,

BCB块中的**aDriveInfo[0].u32SectorCount**用于存放主引导镜像文件(*imx28\_ivt\_uboot.sb*)的**大小(单位:扇区)**.

BCB块中的**aDriveInfo**是一个数组,可以用于存放多个BCB块信息,此BCB块中包含了主启动器和第二启动器的BCB块信息,

因此支持多重启动,而且将来还可以扩展出更多.

**aDriveInfo[0].u32FirstSectorNumber**在BCB控制块中的偏移地址为**0x1C(28),长度4字节**.

**aDriveInfo[0].u32SectorCount**在BCB控制块中的偏移地址为**0x20(32),长度4字节**.如下:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 04 | 31 | D0 | 00 | 90 | 02 | 00 | 00 |

由于是大端模式,所以实际为:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 00 | D0 | 31 | 04 | 00 | 00 | 02 | 90 |

因此系统主引导镜像文件存放在第**0x00D03104=13644036(扇区)**,大小为**0x00000290=656(扇区)=328(KB)**.

<<EasyArm\_IMX280A SD卡分区详解.docx>>