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

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:

标准 MBR 结构

地址			描述		长度（字节）
Hex	Oct	Dec			
0000	0000	0	代码区		**440**（最大 446）
01B8	0670	440	选用磁盘标志		4
01BC	0674	444	一般为空值；0x0000		2
01BE	0676	446	**标准 MBR 分区表规划**（四个16 byte的主分区表入口）		**64**
01FE	0776	510	55h	MBR 有效标志：0x55AA	**2**
01FF	0777	511	AAh		
MBR，总大小：446 + 64 + 2 =					**512**

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.

硬盘分区结构信息

偏移	长度(字节)	意义
00H	1	分区状态：00-->非活动分区；80--> 活动分区； 其它数值没有意义
01H	1	分区起始磁头号(HEAD)，用到全部8位
02H	2	分区起始扇区号(SECTOR)，占据02H的位0－5； 该分区的起始磁柱号(CYLINDER)，占据 02H的位6－7和03H的全部8位
04H	1	文件系统标志位
05H	1	分区结束磁头号(HEAD)，用到全部8位
06H	2	分区结束扇区号(SECTOR)，占据06H的位0－5； 该分区的结束磁柱号(CYLINDER)，占据 06H的位6－7和07H的全部8位
08H	4	分区起始相对扇区号
0CH	4	分区总的扇区数

常见文件系统标志:

分区类型标志：

00 空，mocrosoft不允许使用。	63 GNU HURD or Sys
01 FAT32	64 Novell Netware
02 XENIX root	65 Novell Netware
03 XENIX usr	70 Disk Secure Mult
04 FAT16 <32M	75 PC/IX
05 Extended	80 Old Minix
06 FAT16	81 Minix/Old Linux
07 HPFS/NTFS	82 Linux swap
08 AIX	83 Linux
09 AIX bootable	84 OS/2 hidden C:
0A OS/2 Boot Manage	85 Linux extended
0B Win95 FAT32	86 NTFS volume set
0C Win95 FAT32	87 NTFS volume set
0E Win95 FAT16	93 Amoeba
0F Win95 Extended(>8GB)	94 Amoeba BBT
10 OPUS	A0 IBM Thinkpad hidden
11 Hidden FAT12	A5 BSD/386
12 Compaq diagnost	A6 Open BSD
16 HiddenFAT16	A7 NextSTEP
14 Hidden FAT16<32GB	B7 BSDI fs
17 Hidden HPFS/NTFS	B8 BSDI swap
18 AST Windows swap	BE Solaris boot
1B Hidden FAT32	partition
1C Hidden FAT32 partition	C0 DR-DOS/Novell DOS
(using LBA-mode	secured partition
INT 13 extensions)	C1 DRDOS/sec
1E Hidden LBA VFAT partition	C4 DRDOS/sec
24 NEC DOS	C6 DRDOS/sec
3C Partition Magic	C7 Syrinx
40 Venix 80286	DB CP/M/CTOS
41 PPC PreP Boot	E1 DOS access
42 SFS	E3 DOS R/O
4D QNX4.x	E4 SpeedStor
4E QNX4.x 2nd part	EB BeOS fs
4F QNX4.x 3rd part	F1 SpeedStor
50 Ontrack DM	F2 DOS 3.3+ secondary
51 Ontrack DM6 Aux	partition
52 CP/M	F4 SpeedStor
53 oNtRACK DM6 Aux	FE LAN step
54 OnTrack DM6	FF BBT
55 EZ-Drive	
56 Golden Bow	
5C Priam Edisk	
61 Speed Stor	

图4

关于磁头,柱面,扇区的计算

参考资料: [百度百科:硬盘分区表]

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.

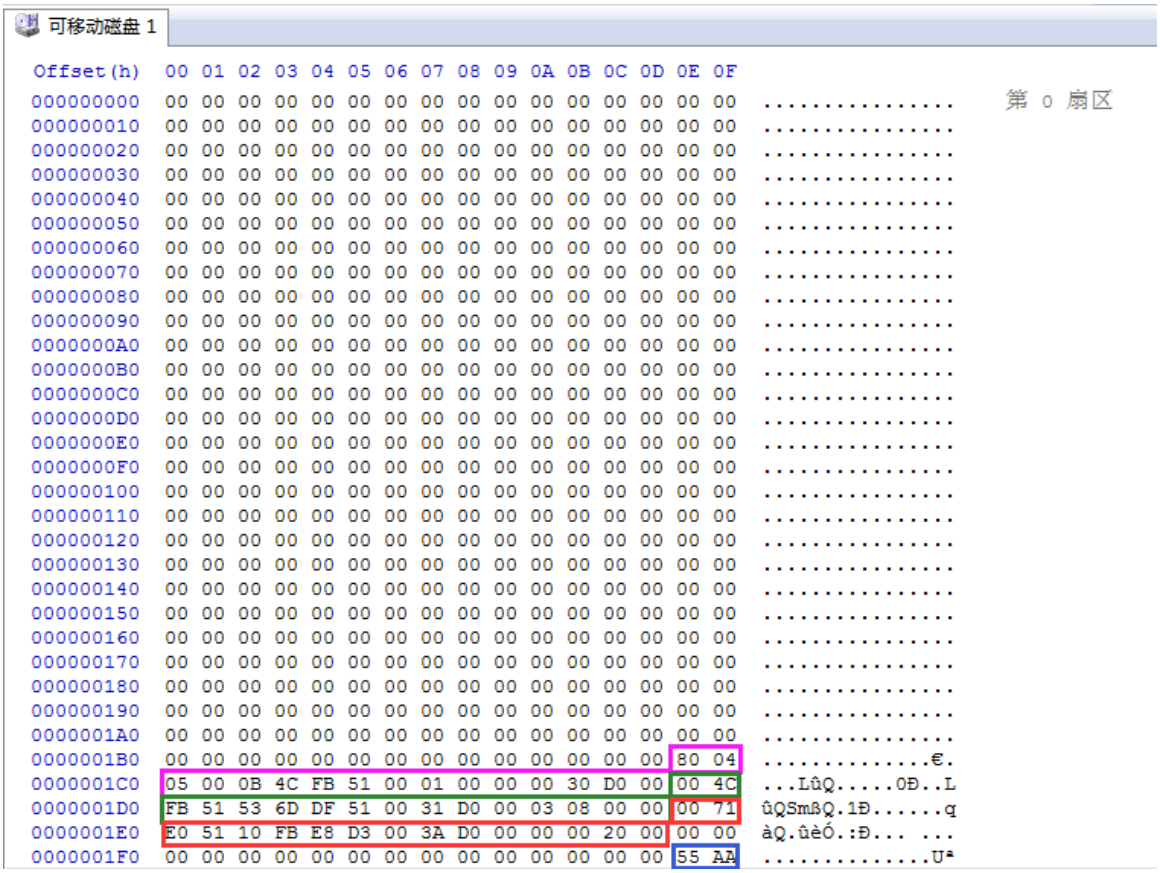
(柱面,磁头,扇区)转换成逻辑扇区号的公式为:

$$L = C \times 16065 + H \times 63 + S - 1;$$

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.



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:

```
root@wnavy-vpc:sd_boot# fdisk -l /dev/sdc

Disk /dev/sdc: 8068 MB, 8068792320 bytes
249 heads, 62 sectors/track, 1020 cylinders, total 15759360 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *         256    13644031     6821888    b  W95 FAT32
/dev/sdc2        13644032    13646082        1025+  53  OnTrack DM6 Aux3
/dev/sdc3        13646336    15743487     1048576   10  OPUS
root@wnavy-vpc:sd_boot#
```

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：

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 0x1FE of the first sector. The partition table is stored at address 0x1BE. 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 of firmware partition entry of partition table.

Field	Value
MBR Signature	0x55AA
MBR_SIGMATEL_ID	'S'

The first block of firmware partition contains BCB, allowing multiple copies of firmware 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. All 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

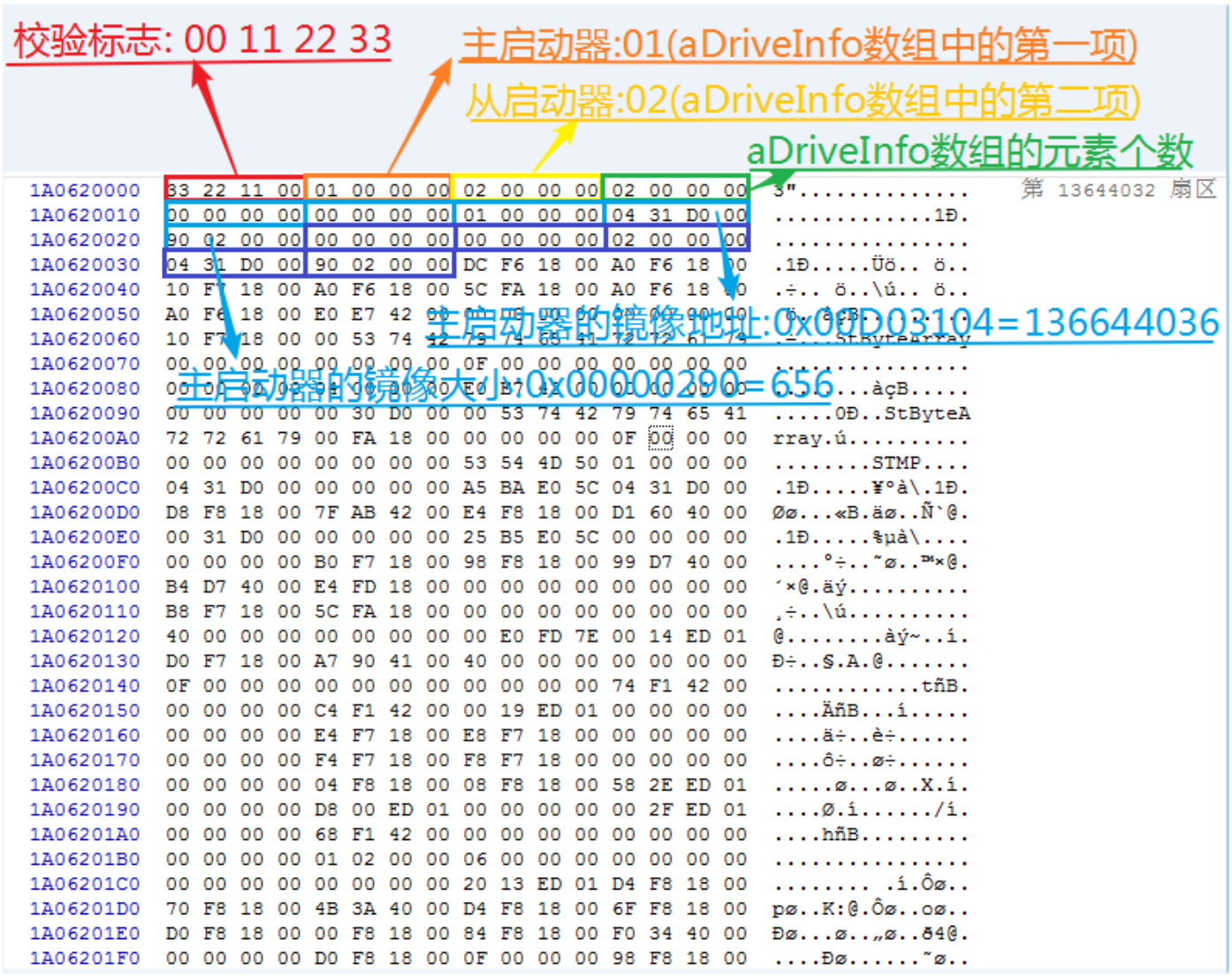
偏移 **0x08** 字节处, 长度 4 字节.因此可以看出,系统**引导固件分区**在该 SD 卡的第二分区上,并且该**引导固件分区**的逻辑扇区地址为:

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：

12.11.1 Boot Control Block (BCB) Data Structure

The design of BCB is to allow multiple copies of firmware to be stored on media each identified by its unique 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, if ROM_REDUNDANT_BOOT persistent bit is not set; otherwise it loads a secondary image, providing support for a redundant boot. The config block has the following format:

```
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;
```

The driver first verifies the signature and version, then searches all NumRegions for the appropriate tag. The following table shows the expected values for these parameters.

Table 12-29. Media Config Block Parameters

Field	Value
Signature	0x00112233
u32PrimaryBootTag	User-defined primary boot firmware tag
u32SecondaryBootTag	User- defined secondary boot tag
u32NumCopies	Number of firmware copies present in array aDriveInfo
aDriveInfo	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>>