Linux SPINAND
开发指南

版本号: 1.4
发布日期: 2024.10.23

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.02.01	AW1669	建立初始版本
1.1	2022.11.10	AW1669	添加 5.10 内核配置说明
1.2	2023.12.11	AW2080	添加 4.9、5.15 内核配置说明和内核源码目录
1.3	2024.08.14	AW2155	添加 uboot2023 配置说明
1.4	2024.10.23	AW1669	新增了开发功能、日志分析、调试方法和 FAQ 章节

版权所有 © 珠海全志科技股份有限公司。保留一切权利

目　录

1 概述 1
1.1 编写目的 . . . 1
1.2 适用范围 . . . 1
1.3 相关人员 . . . 1

2 术语、缩略语及概念 2

3 流程设计 3
3.1 体系结构 . . . 3
3.2 源码结构 . . . 4
3.2.1 Linux4.9/5.4 . . . 4
3.2.2 Linux5.4-ansc/5.10/5.15 . . . 4

4 模块配置 6
4.1 uboot 模块配置 . . . 6
4.1.1 uboot-2018 . . . 6
4.1.2 uboot-2023 . . . 7
4.2 kernel 模块配置 . . . 9
4.2.1 Linux4.9 . . . 9
4.2.2 Linux5.4 . . . 10
4.2.3 Linux5.4-ansc/5.10/5.15 . . . 13
4.2.4 env.cfg . . . 18

5 关键数据定义与接口说明 19
5.1 关键数据定义 . . . 19
5.1.1 flash 设备信息数据结构 . . . 19
5.1.2 flash chip 数据结构 . . . 20
5.1.3 aw_spinand_chip_request . . . 21
5.1.4 ubi_ec_hdr . . . 22
5.1.5 ubi_vid_hdr . . . 22
5.2 关键接口说明 . . . 25
5.2.1 MTD 层接口 . . . 25
5.2.1.1 aw_spinand_erase . . . 25
5.2.1.2 aw_spinand_read . . . 25
5.2.1.3 aw_spinand_read_oob . . . 26
5.2.1.4 aw_spinand_write . . . 26
5.2.1.5 aw_spinand_write_oob . . . 26
5.2.1.6 aw_spinand_block_isbad . . . 27
5.2.1.7 aw_spinand_block_markbad . . . 27

版权所有 © 珠海全志科技股份有限公司。保留一切权利

5.2.2 物理层接口 . . . 27
5.2.2.1 aw_spinand_chip_read_single_page . . . 27
5.2.2.2 aw_spinand_chip_write_single_page . . . 28
5.2.2.3 aw_spinand_chip_erase_single_block . . . 28
5.2.2.4 aw_spinand_chip_isbad_single_block . . . 28
5.2.2.5 aw_spinand_chip_markbad_single_block . . . 29
5.2.3 Uboot 应用接口 . . . 29
5.2.3.1 sunxi_flash_nand_probe . . . 29
5.2.3.2 sunxi_flash_nand_init . . . 29
5.2.3.3 sunxi_flash_nand_exit . . . 29
5.2.3.4 sunxi_flash_nand_write . . . 30
5.2.3.5 sunxi_flash_nand_read . . . 30
5.2.3.6 sunxi_flash_nand_erase . . . 30
5.2.3.7 sunxi_flash_nand_force_erase . . . 30
5.2.3.8 sunxi_flash_nand_flush . . . 31
5.2.3.9 sunxi_flash_nand_download_spl . . . 31
5.2.3.10 sunxi_flash_nand_download_toc . . . 31

6 功能开发 32
6.1 功能概述 . . . 32
6.2 开发流程 . . . 32
6.2.1 BOOT0 读取数据 . . . 32
6.2.2 uboot shell 使用 . . . 33
6.2.3 内核态访问 flash . . . 34
6.2.4 用户态访问 flash . . . 35
6.2.5 在 ubi 卷上模拟 mtdblock 设备，挂载块设备文件系统 . . . 36
6.3 注意事项 . . . 37

7 日志分析 38

8 调试方法 40

9 FAQ 41
9.1 dram 全盘扫描 . . . 41
9.2 启动失败 . . . 42
9.2.1 brom 启动 boot0 失败 . . . 42
9.2.2 boot0 读 uboot 失败 . . . 42
9.3 Ubifs scanning 时间过长 . . . 42
9.4 Flash 分区与使用说明大小不一致 . . . 43
9.5 fat 挂载在 mtdblock 设备上，sync 数据没有写下 . . . 45

版权所有 © 珠海全志科技股份有限公司。保留一切权利

插　图

图 3-1 UBI 架构 . . . 3
图 4-1 Support_sunxi_nand . . . 6
图 4-2 defconfig_nand_single_plane . . . 7
图 4-3 uboot2023_support_spinand . . . 8
图 4-4 uboot2023_spinand_twoplane . . . 8
图 4-5 MTD_SPINAND_UBI . . . 9
图 4-6 UBIFS . . . 10
图 4-7 sunxi-nand_UBI . . . 10
图 4-8 AWNAND_CHOICE . . . 11
图 4-9 SPINAND_Support . . . 11
图 4-10 SUNXI_SPI_Controller . . . 12
图 4-11 A31_DMA_support . . . 12
图 4-12 sid_support . . . 13
图 4-13 Support_UBIFS . . . 13
图 4-14 menuconfig_bsp_MTD . . . 14
图 4-15 menuconfig_bsp_single_plane . . . 14
图 4-16 SPI_Support . . . 15
图 4-17 SPI_Support . . . 15
图 4-18 DMA_Support . . . 16
图 4-19 menuconfig_MTD . . . 16
图 4-20 menuconfig_UBI . . . 17
图 4-21 UBIFS_Support . . . 17
图 4-22 menuconfig_gengeral_setup_ram . . . 18
图 5-1 PEB-LEB . . . 25
图 6-1 sunxi flash read . . . 33
图 6-2 hexdump . . . 33
图 6-3 mm - md . . . 34
图 6-4 sunxi flash write . . . 34
图 6-5 sunxi flash read2 . . . 34
图 9-1 DragonHD tool . . . 41
图 9-2 DragonHD tool . . . 41
图 9-3 分区信息 . . . 43
图 9-4 UBIFS 数量类型及损耗说明 . . . 44
图 9-5 PEB-LEB . . . 45

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1 概述

1.1 编写目的

介绍 Sunxi SPINand mtd/ubi 驱动设计, 方便相关驱动和应用开发人员

1.2 适用范围

本设计适用于所有 sunxi 平台

1.3 相关人员

Nand 模块开发人员，及应用开发人员等

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2 术语、缩略语及概念

MTD：（Memory Technology device）是用于访问存储设备的 linux 子系统。本模块是 MTD 子系统的 flash 驱动部分

UBI：UBI 子系统是基于 MTD 子系统的，在 MTD 上实现 nand 特性的管理逻辑，向上屏蔽 nand 的特性

坏块 (Bad Block)：制作工艺和 nand 本身的物理性质导致在出厂和正常使用过程中都会产生坏块

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3 流程设计

3.1 体系结构

NAND MTD/UBI 驱动主要包括 5 大组件，如下图：

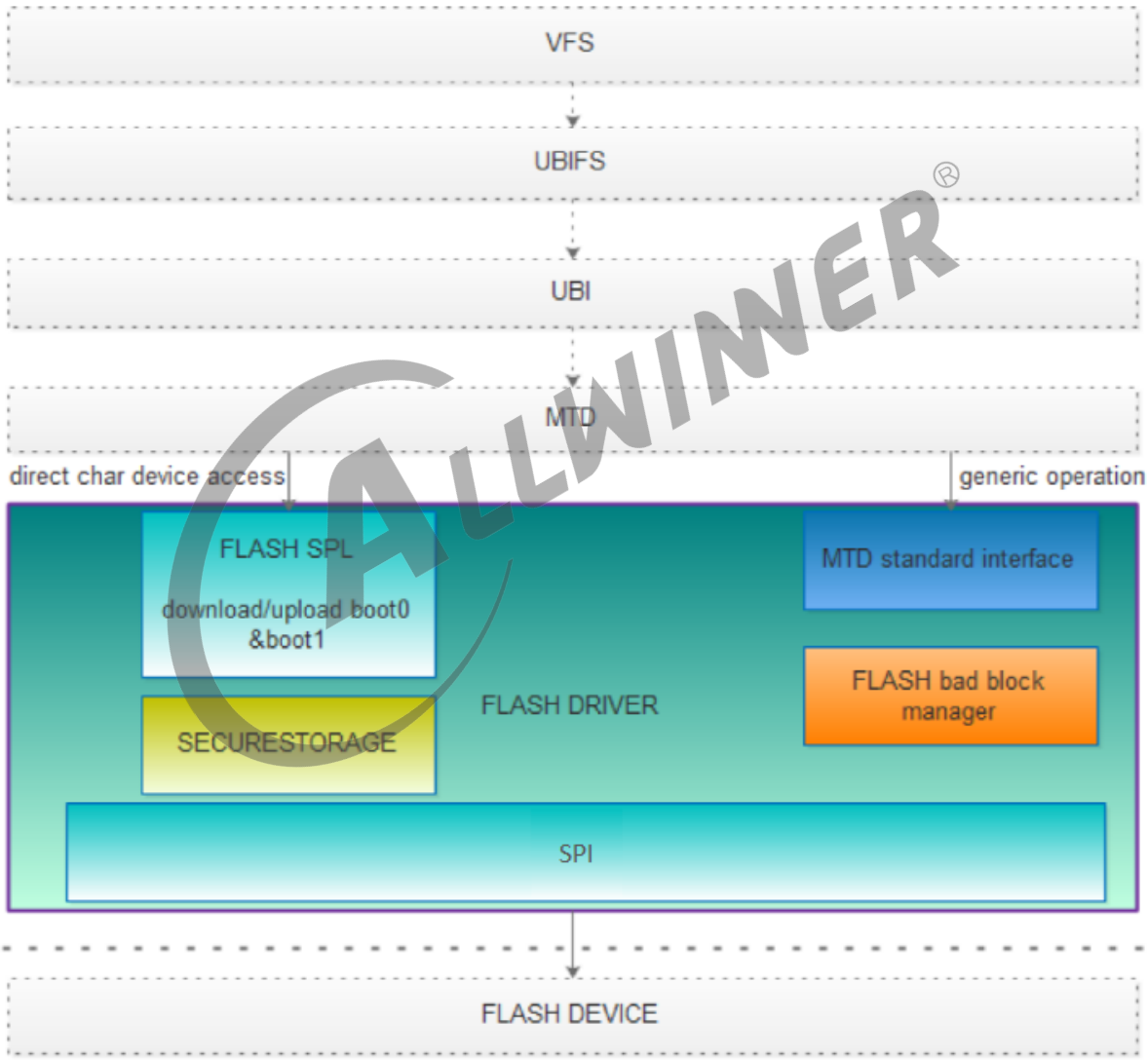

图 3-1: UBI 架构

说明：

MTD standard interface: 对接 MTD 层通用读写接口

版权所有 © 珠海全志科技股份有限公司。保留一切权利

FLASH bad block manager: 驱动层对 flash 坏块的管理

FLASH SPL: 主要是实现读写 boot0、boot1，可用于 ioctl 对 boot0、boot1 的升级

SECURESTORAGE：主要是给上层提供私有数据的管理

SPI：HOST 端控制器层的实现

3.2 源码结构

3.2.1 Linux4.9/5.4

kernel 源码目录：linux-4.9/drivers/mtd/awnand/spinand

```
.
├── Kconfig
├── Makefile
├── physic
│   ├── bbt.c
│   ├── cache.c
│   ├── core.c
│   ├── ecc.c
│   ├── id.c
│   ├── Makefile
│   ├── ops.c
│   └── physic.h
├── secure-storage.c
├── sunxi-common.c
├── sunxi-core.c
├── sunxi-debug.c
├── sunxi-nftl-core.c
└── sunxi-spinand.h
```

内核目录下

```
`-- include
  `-- linux
    `-- mtd
      |-- aw-spinand.h
```

3.2.2 Linux5.4-ansc/5.10/5.15

bsp 源码目录：bsp/drivers/mtd/awnand/spinand

```
├── Kconfig
├── Makefile
└── spinand
  ├── Kconfig
  ├── Makefile
  ├── physic
  │   ├── bbt.c
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
│   ├── cache.c
│   ├── core.c
│   ├── ecc.c
│   ├── id.c
│   ├── Makefile
│   ├── ops.c
│   └── physic.h
├── secure-storage.c
├── sunxi-common.c
├── sunxi-core.c
├── sunxi-debug.c
├── sunxi-nftl-core.c
└── sunxi-spinand.h
```

头文件在：bsp/include/linux/mtd/

```
├── aw-spinand.h
└── aw-spinand-nftl.h
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

4 模块配置

4.1 uboot 模块配置

4.1.1 uboot-2018

Device Drivers-->Sunxi flash support-->
[*]Support sunxi nand devices
[*]Support sunxi nand ubifs devices
[*]Support COMM NAND V1 interface

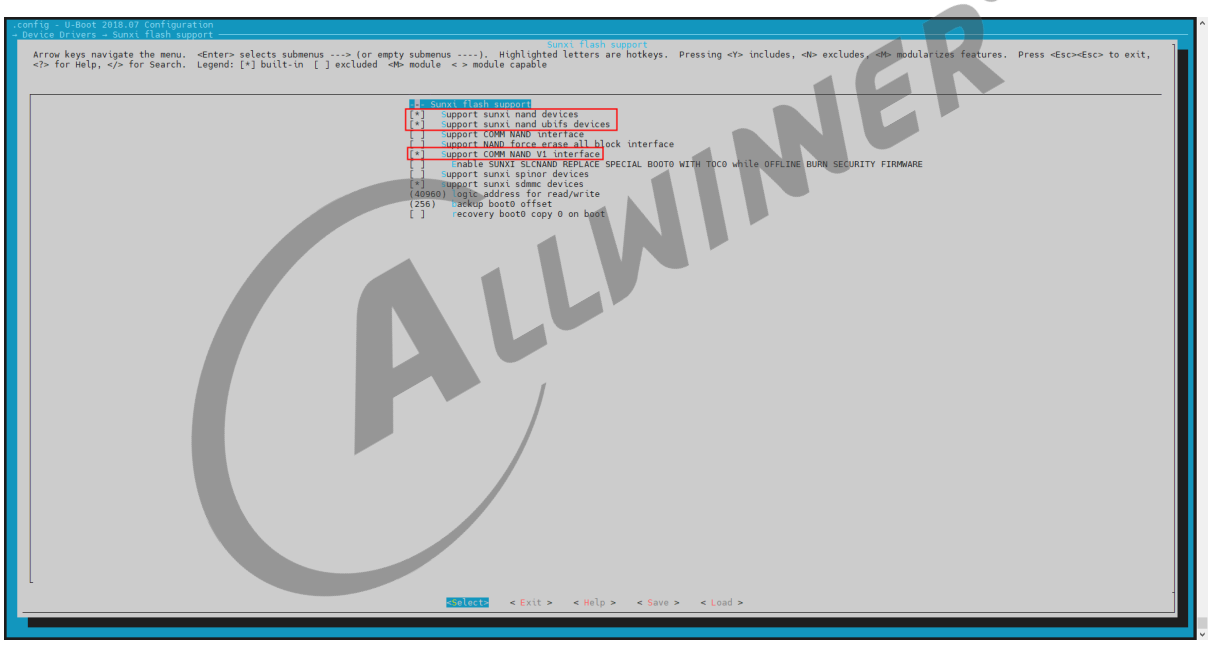

图 4-1: Support_sunxi_nand

如果内核是 Linux5.10 或 Linux 5.15，则应配置成 Single Plane（不打开 enable simulate multi-plane，之前的内核版本则需要打开），如下所示：

Device Drivers → MTD Support
[ ] enable simulate multiplane

版权所有 © 珠海全志科技股份有限公司。保留一切权利

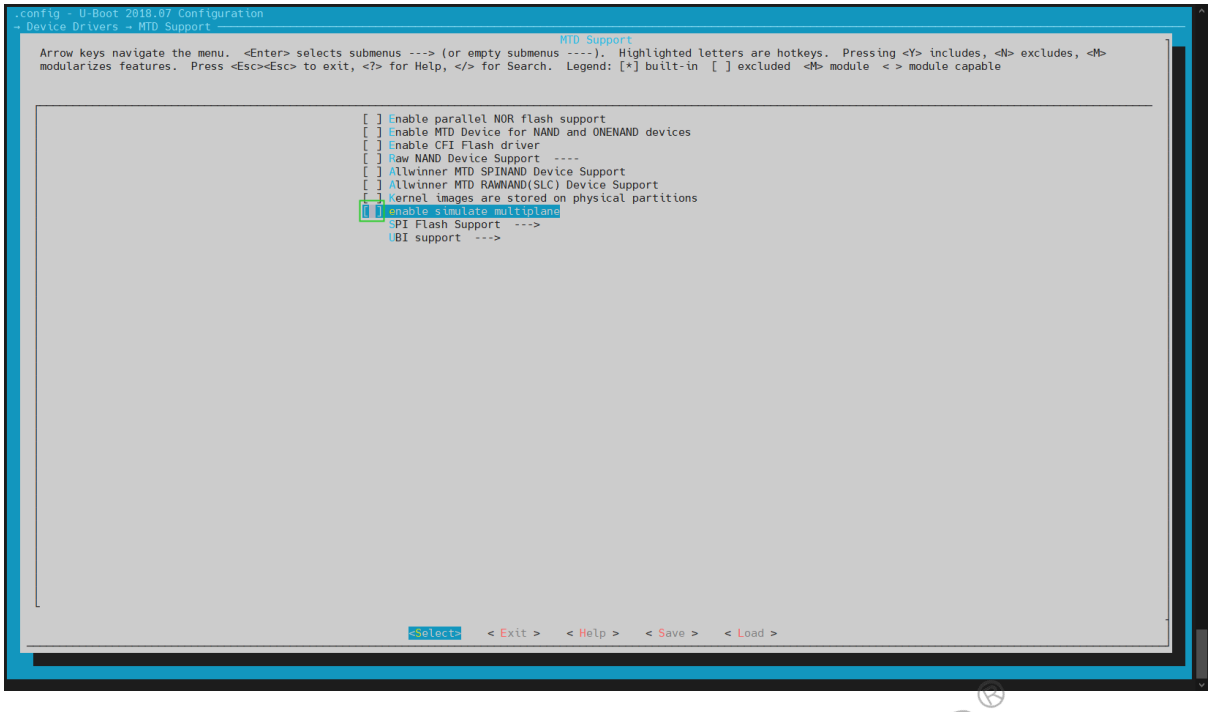

图 4-2: defconfig_nand_single_plane

4.1.2 uboot-2023

首先要打开 AW_SPI 和 SPI_MEM 配置

```
Allwinner uboot BSP ---> Device Drivers --->
[*] SUNXI SPI driver
```

```
Device Drivers  --->
  [*] SPI Support  --->
    [*]   SPI memory extension
```

接着打开 uboot-bsp 仓库中的 spinand 配置

```
Allwinner uboot BSP ---> Device Drivers --->
[*] SUNXI awnand driver
[*] Allwinner MTD SPINAND Device Support
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

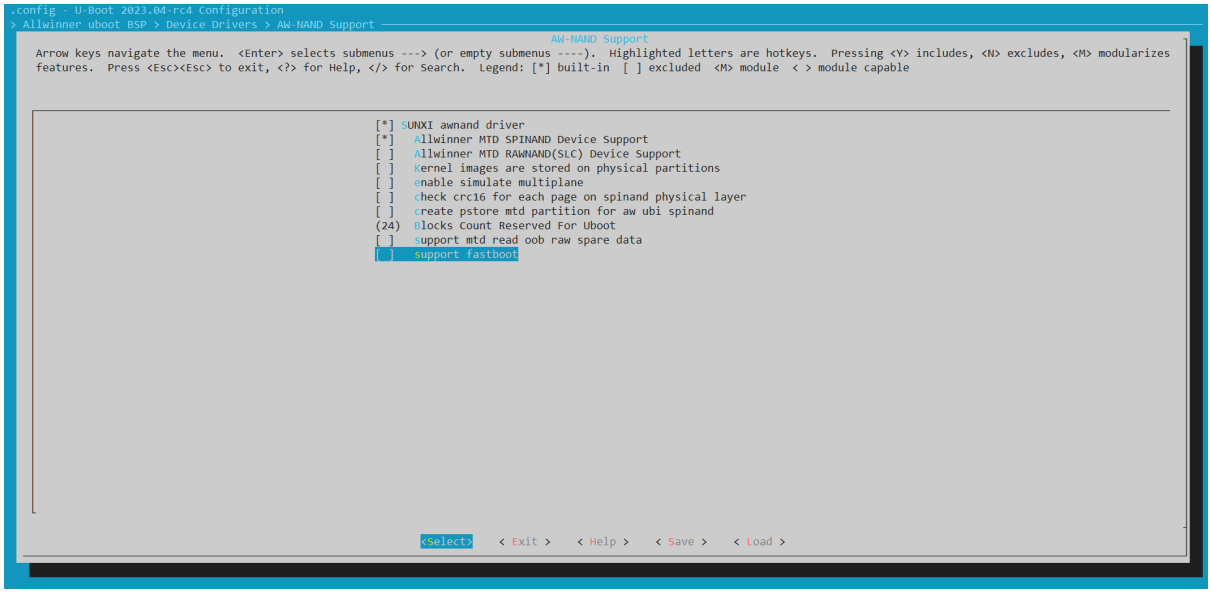

图 4-3: uboot2023_support_spinand

如果内核是 Linux5.10 或 Linux 5.15，则应配置成 Single Plane（不打开 enable simulate multi-plane，之前的内核版本则需要打开），如下所示：

```
Allwinner uboot BSP ---> Device Drivers --->
[*] enable simulate multiplane
```

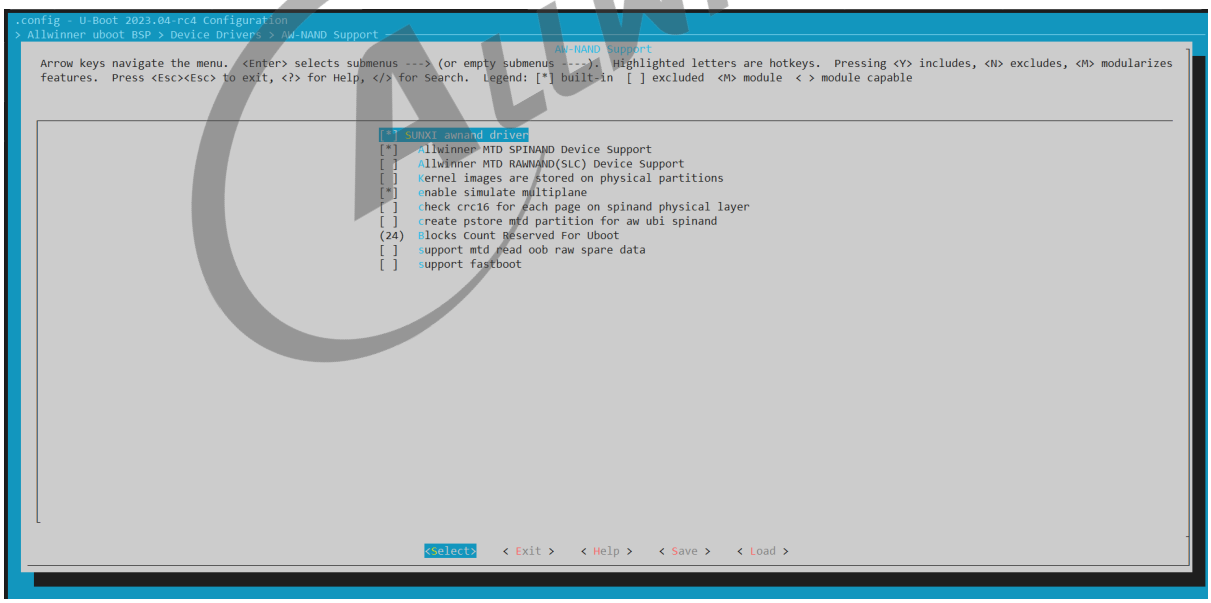

图 4-4: uboot2023_spinand_twoplane

版权所有 © 珠海全志科技股份有限公司。保留一切权利

4.2 kernel 模块配置

4.2.1 Linux4.9

Device Drivers->Memory Technology Device (AW_MTD) support
<*> Allwinner MTD SPINAND Device Support
-*- Enable UBI - Unsorted block images --->

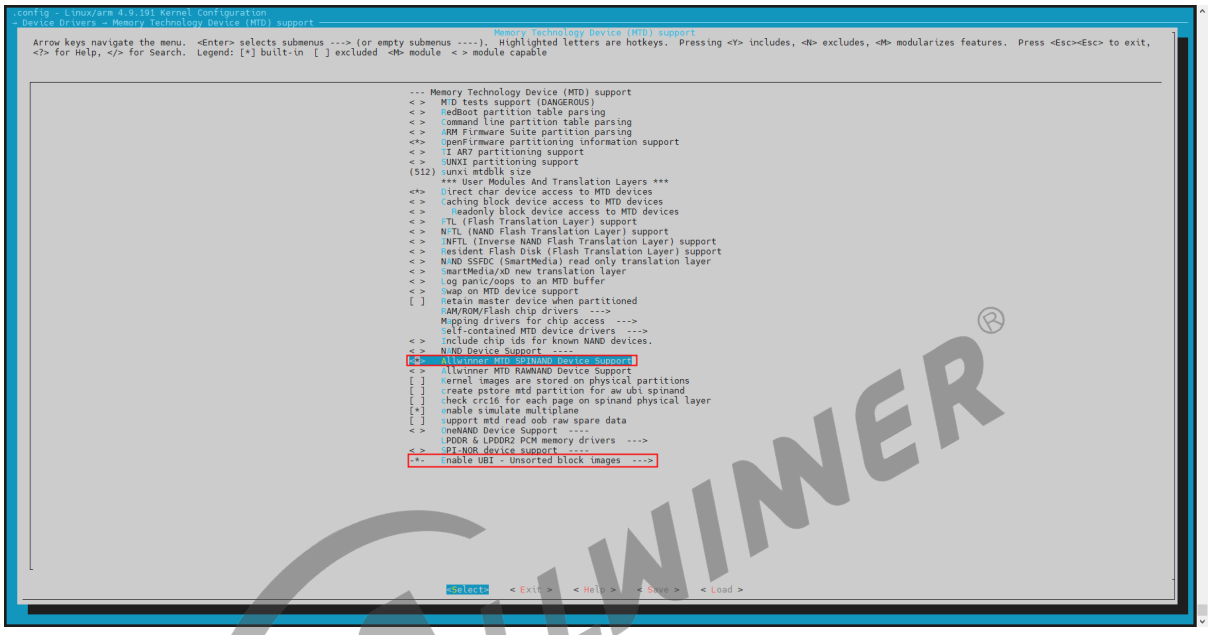

图 4-5: MTD_SPINAND_UBI

File systems->Miscellaneous filesystems->
<*> UBIFS fils system support

版权所有 © 珠海全志科技股份有限公司。保留一切权利

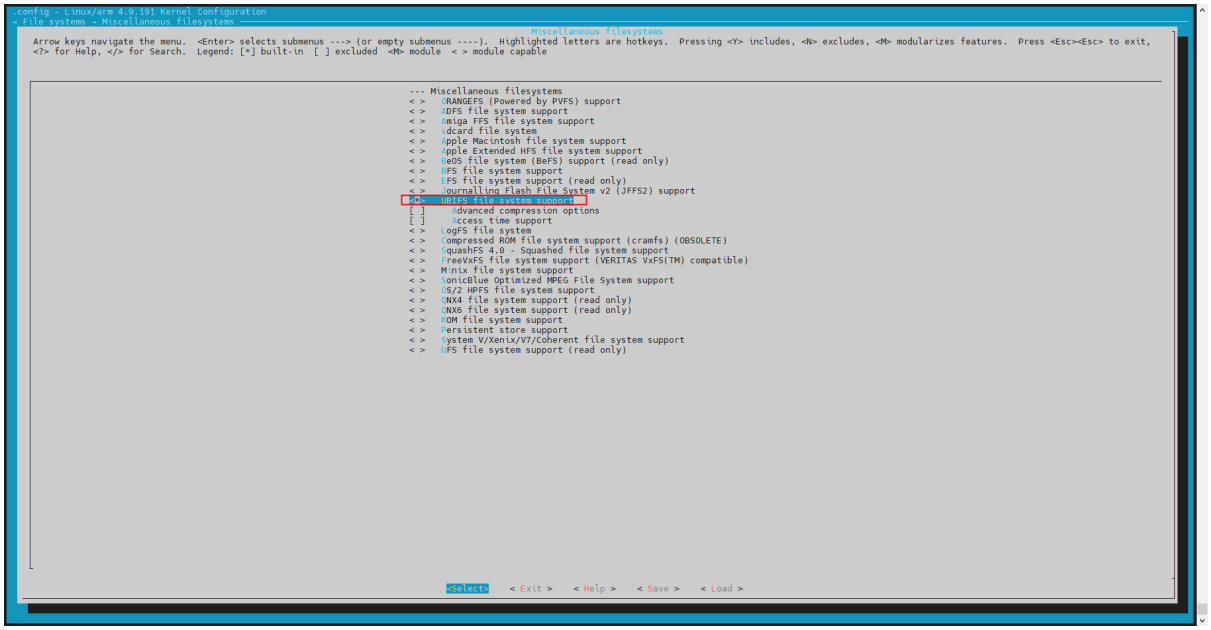

图 4-6: UBIFS

4.2.2 Linux5.4

Device Drivers->Memory Technology Device (AW_MTD) support
sunxi-nand --->
<*> AWNAND CHOICE (Allwinner MTD SPINAND Device Support) --->
(X) Allwinner MTD SPINAND Device Support
() Allwinner MTD RAWNAND Device Support
-*- Enable UBI - Unsorted block images --->

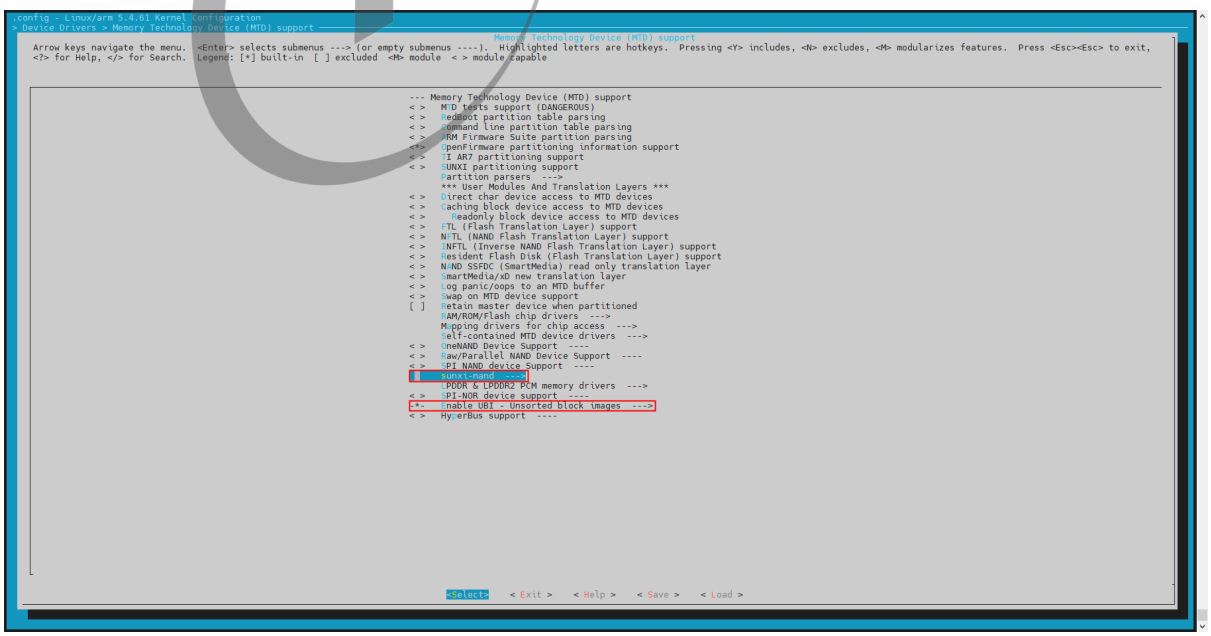

图 4-7: sunxi-nand_UBI

版权所有 © 珠海全志科技股份有限公司。保留一切权利

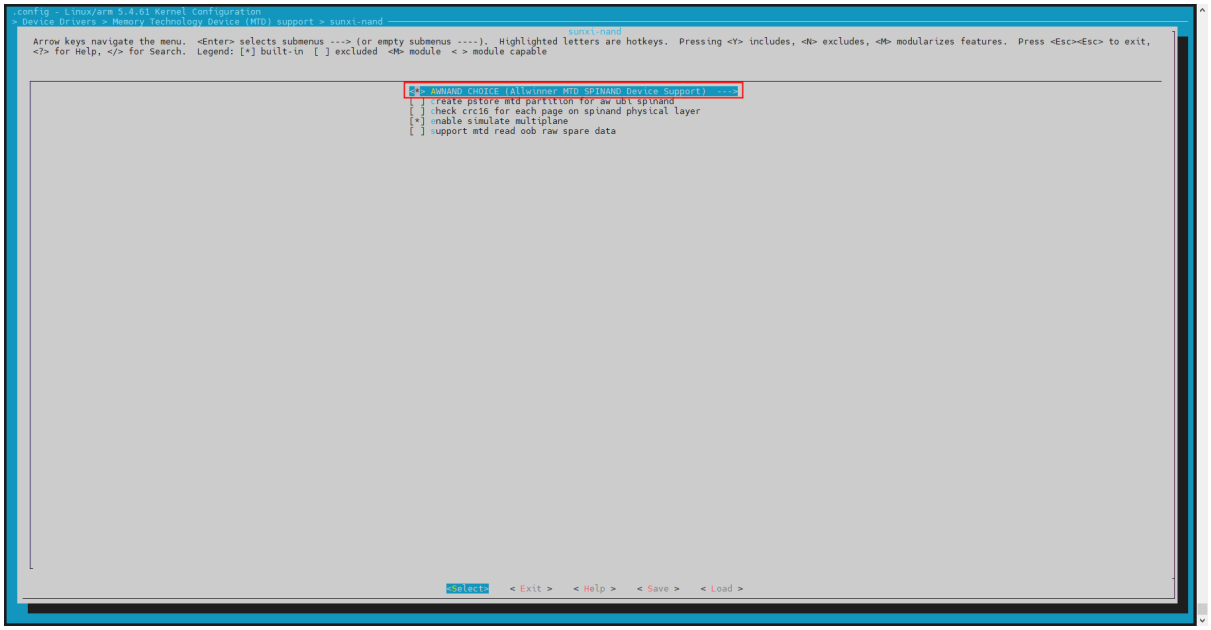

图 4-8: AWNAND_CHOICE

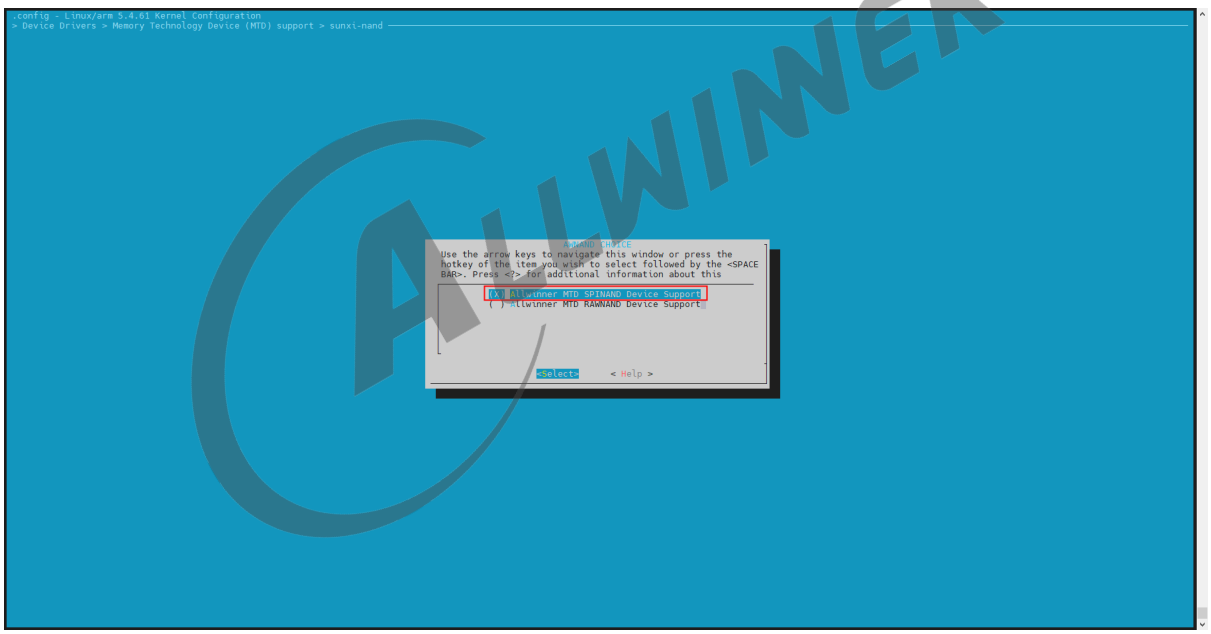

图 4-9: SPINAND_Support

```
Device Drivers->SPI support
[*] SPI support --->
  <*> SUNXI SPI Controller
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

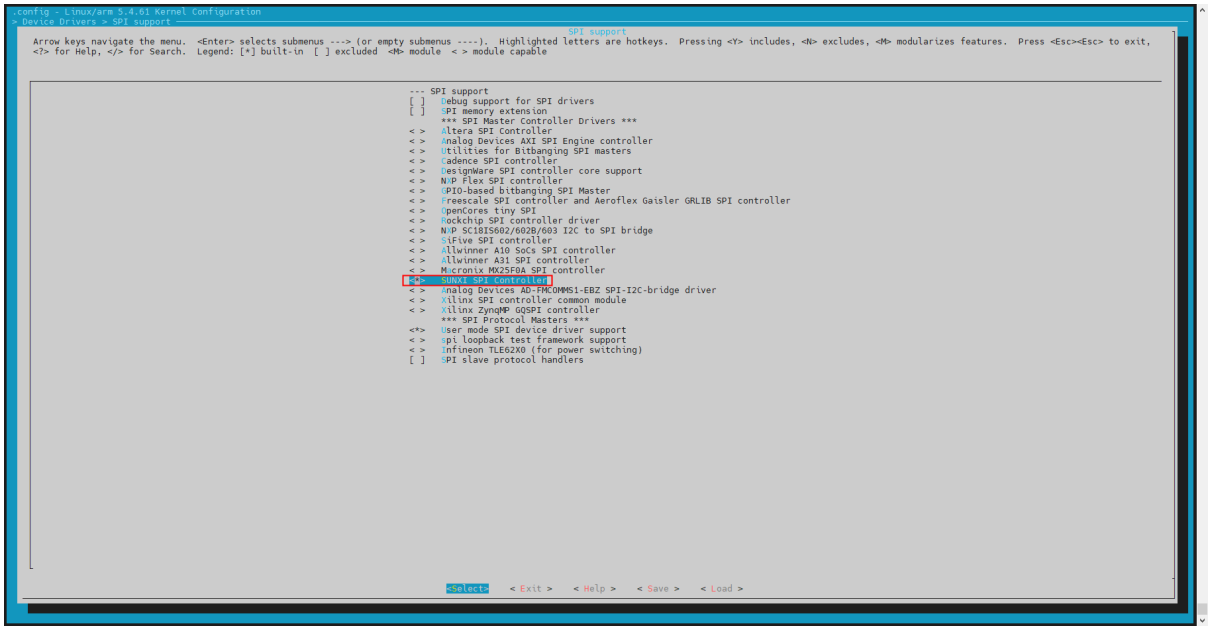

图 4-10: SUNXI_SPI_Controller

Device Drivers->DMA Engine support
[*] DMA Engine support --->
<*> Allwinner A31 SoCs DMA support

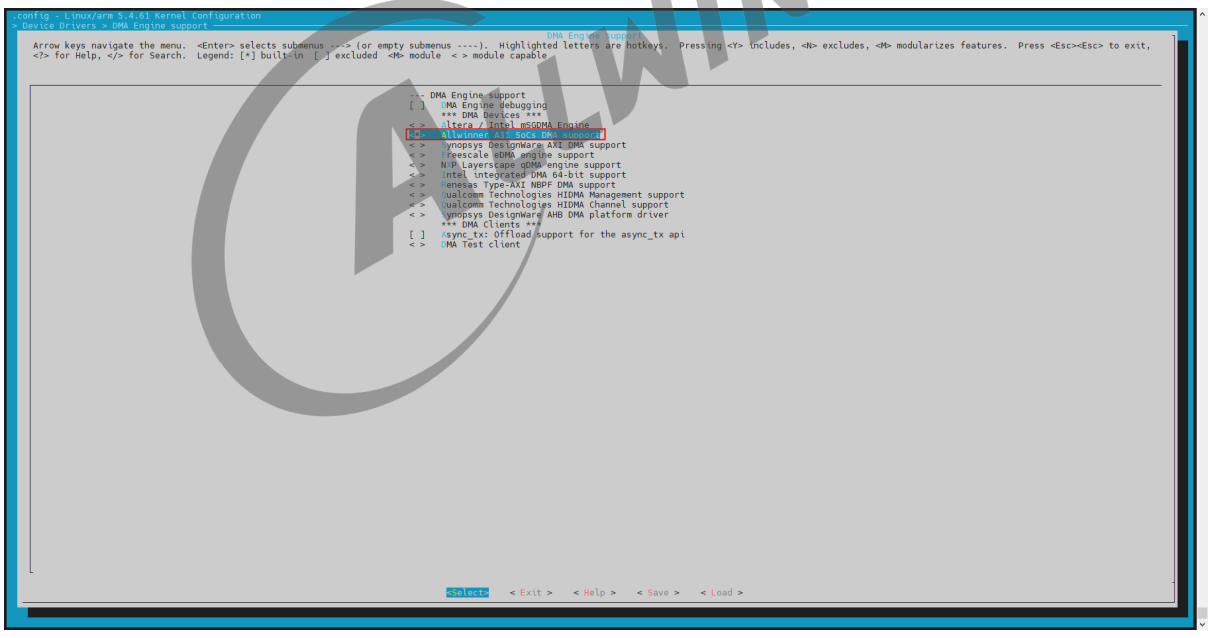

图 4-11: A31_DMA_support

Device Drivers->SOC (System On Chip)
<*> Allwinner sunxi sid support

版权所有 © 珠海全志科技股份有限公司。保留一切权利

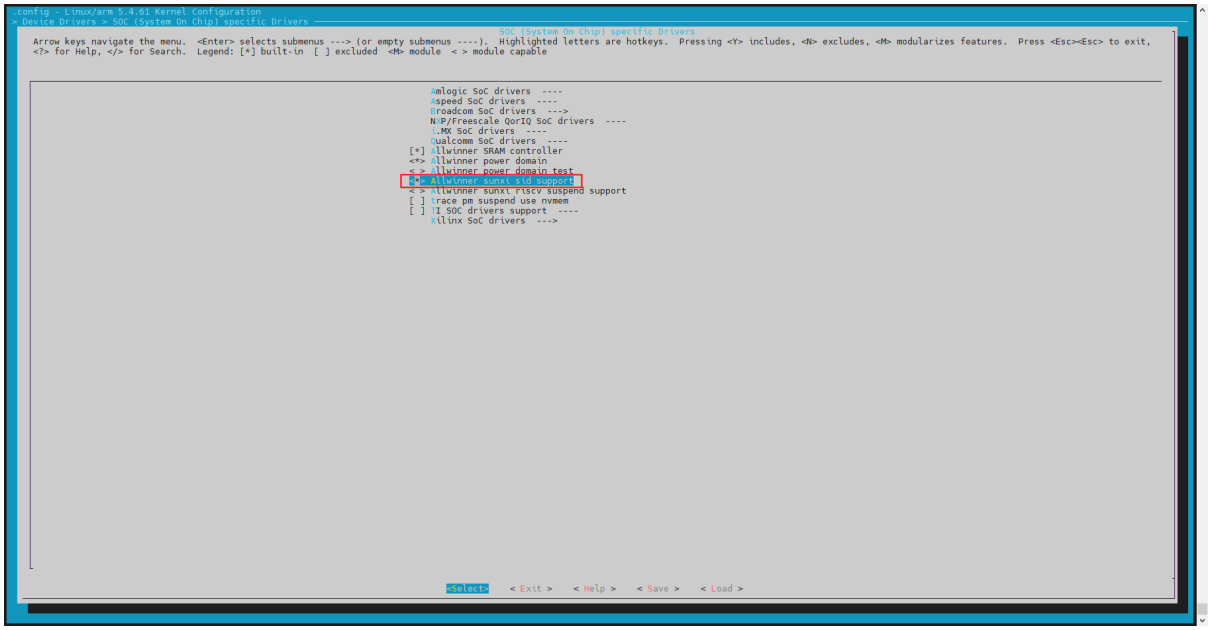

图 4-12: sid_support

File systems->Miscellaneous filesystems->
<*> UBIFS fils system support

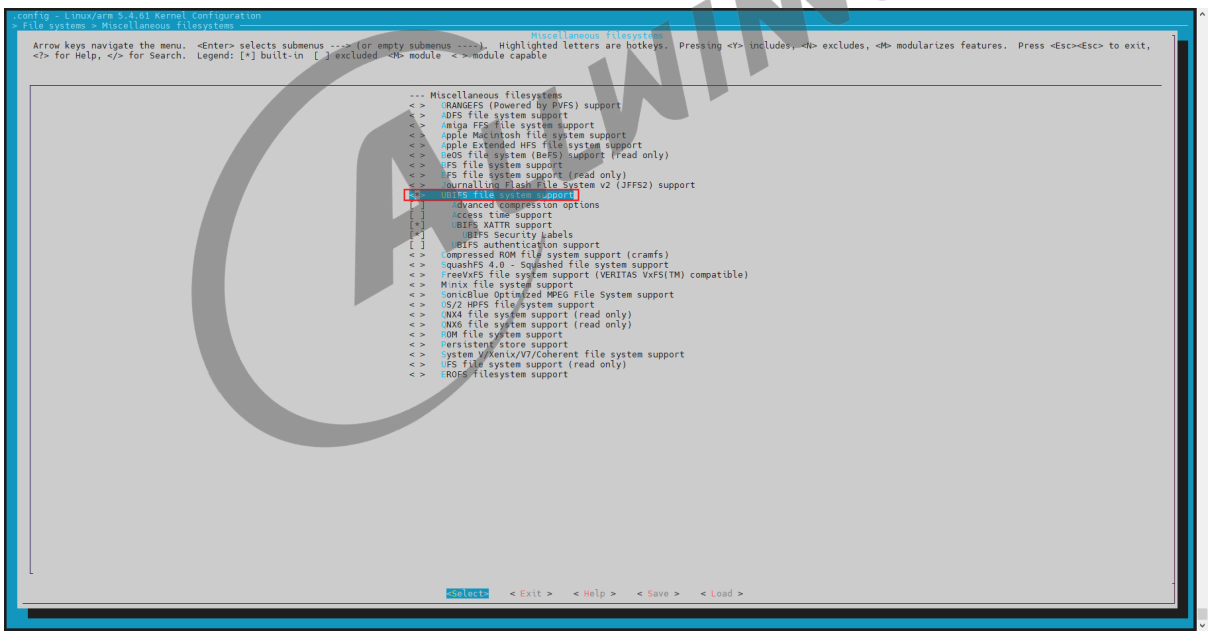

图 4-13: Support_UBIFS

4.2.3 Linux5.4-ansc/5.10/5.15

Allwinner BSP > Device Drivers > Memory Technology Device (AW_MTD) support > AW_MTD_NAND
<*> Allwinner MTD SPINAND Device Support --->

版权所有 © 珠海全志科技股份有限公司。保留一切权利

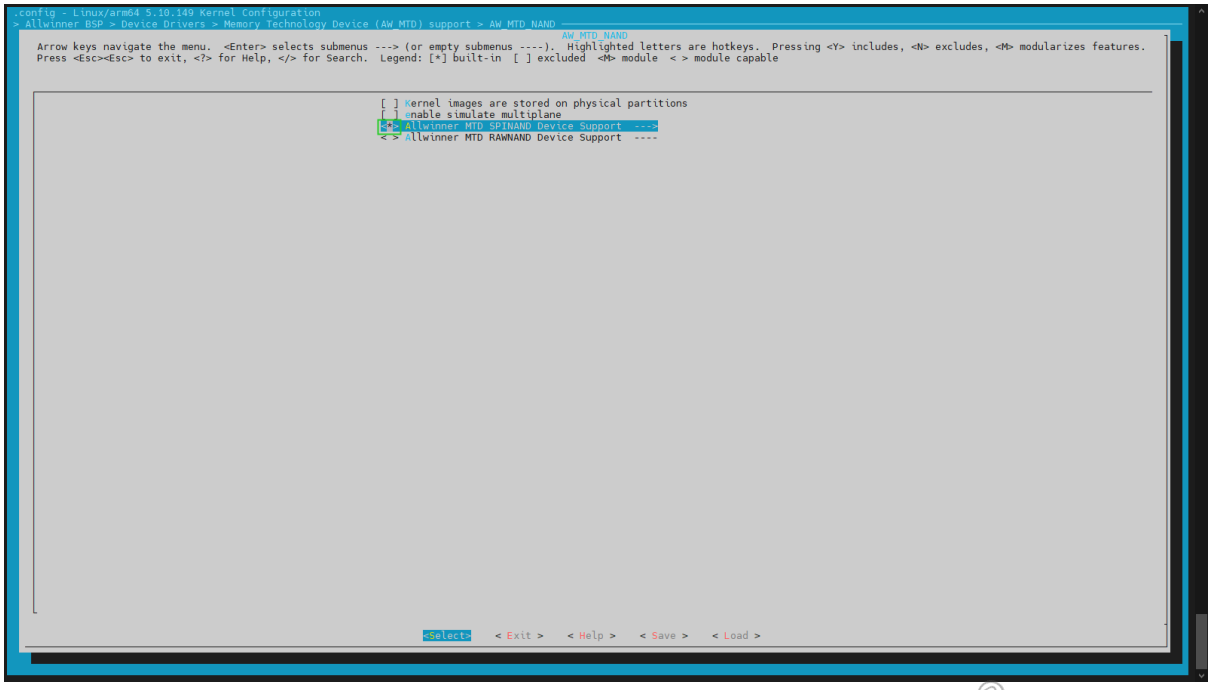

图 4-14: menuconfig_bsp_MTD

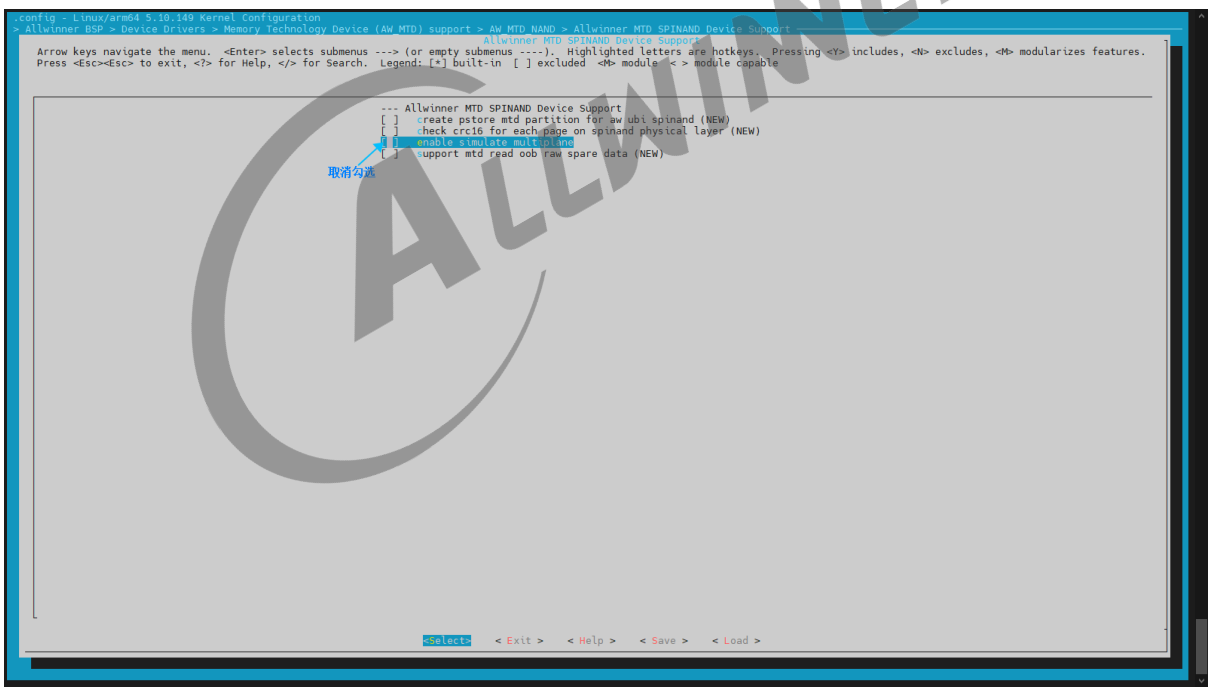

图 4-15: menuconfig_bsp_single_plane

Allwinner BSP->Device Drivers->SPI Drivers
<> SPI Support for Allwinner SoCs

版权所有 © 珠海全志科技股份有限公司。保留一切权利

图 4-16: SPI_Support

Allwinner BSP > Device Drivers > SPI NG Drivers
<*> SPI NG Driver Support for Allwinner SoCs

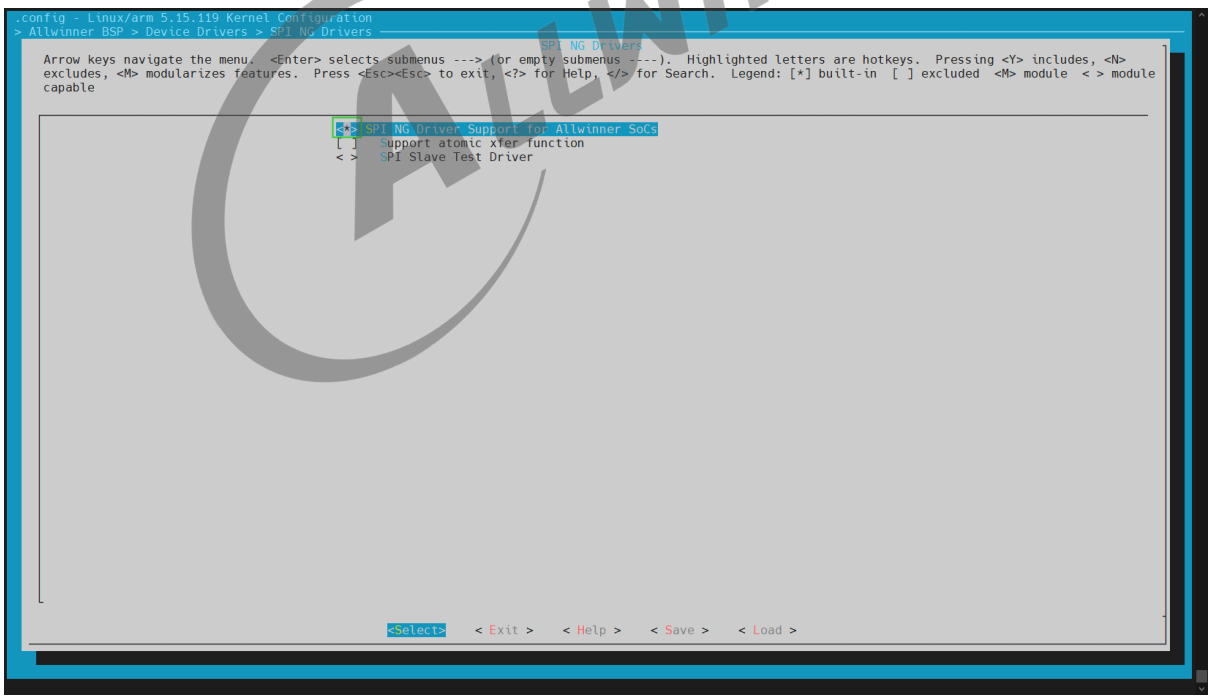

图 4-17: SPI_Support

Allwinner BSP->Device Drivers->DMA Drivers
<*> DMA Support for Allwinner SoCs

版权所有 © 珠海全志科技股份有限公司。保留一切权利

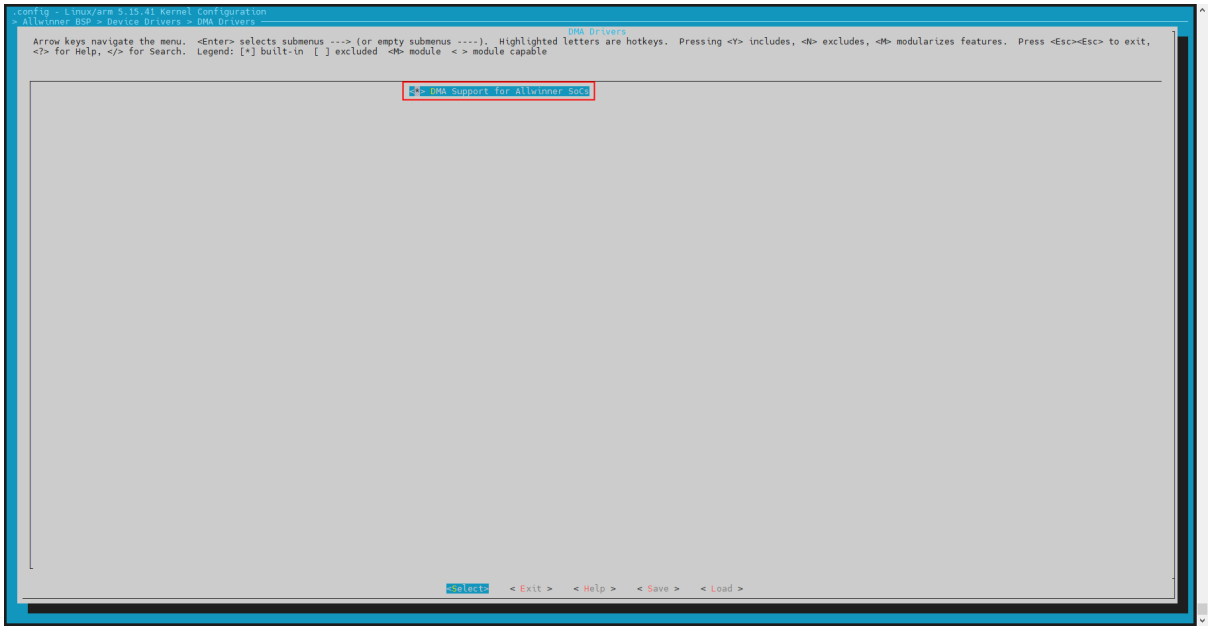

图 4-18: DMA_Support

Device Drivers -> Memory Technology Device (MTD) support
<*> Caching block device access to MTD devices

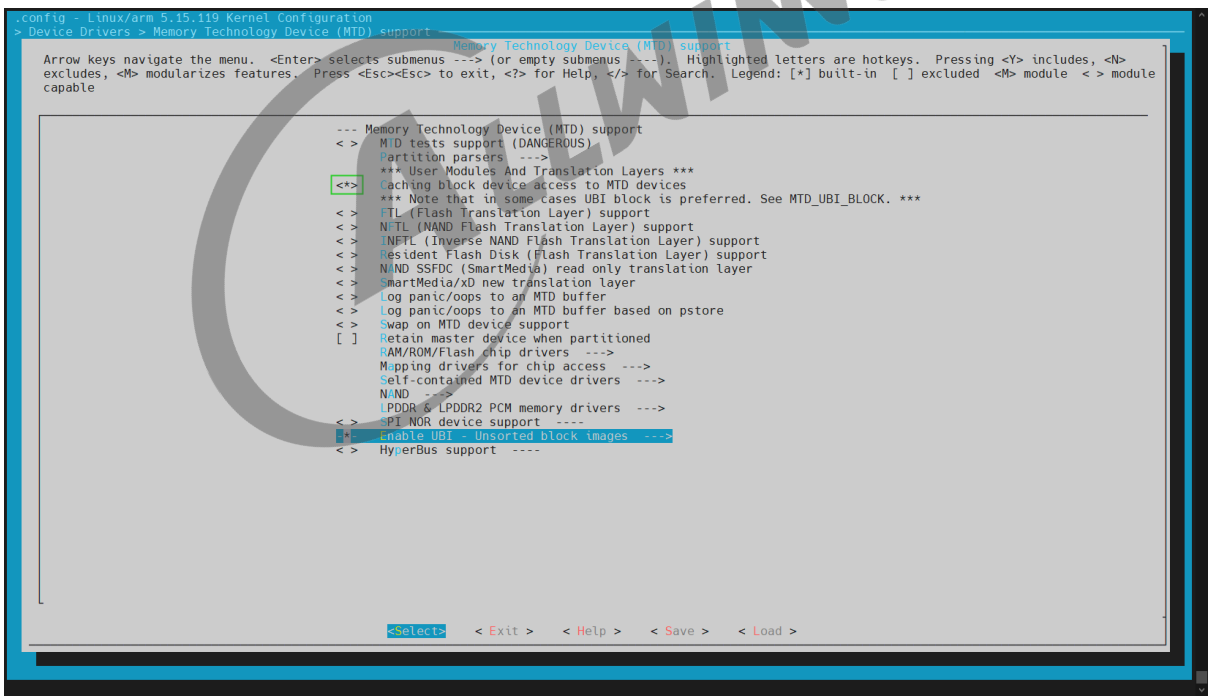

图 4-19: menuconfig_MTD

Device Drivers -> Memory Technology Device (MTD) support -> Enable UBI
<*> MTD devices emulation driver (gluebi)

版权所有 © 珠海全志科技股份有限公司。保留一切权利

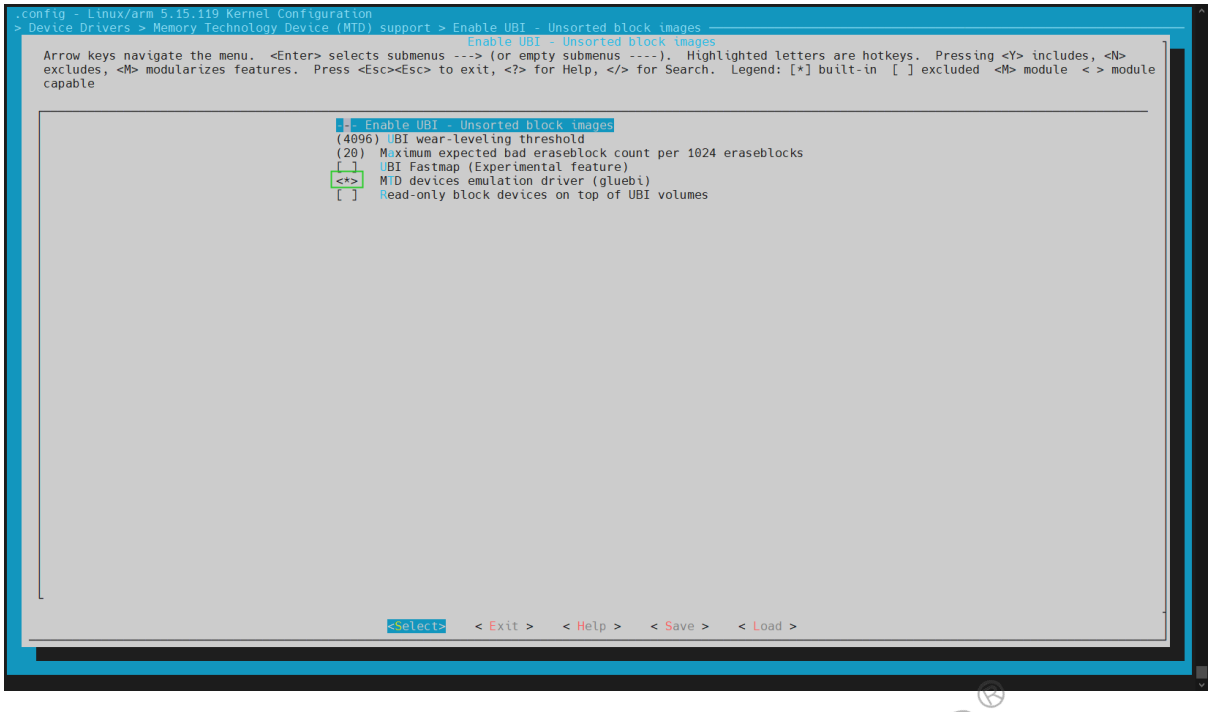

图 4-20: menuconfig_UBI

File systems->Miscellaneous filesystems
<*> UBIFS file system support

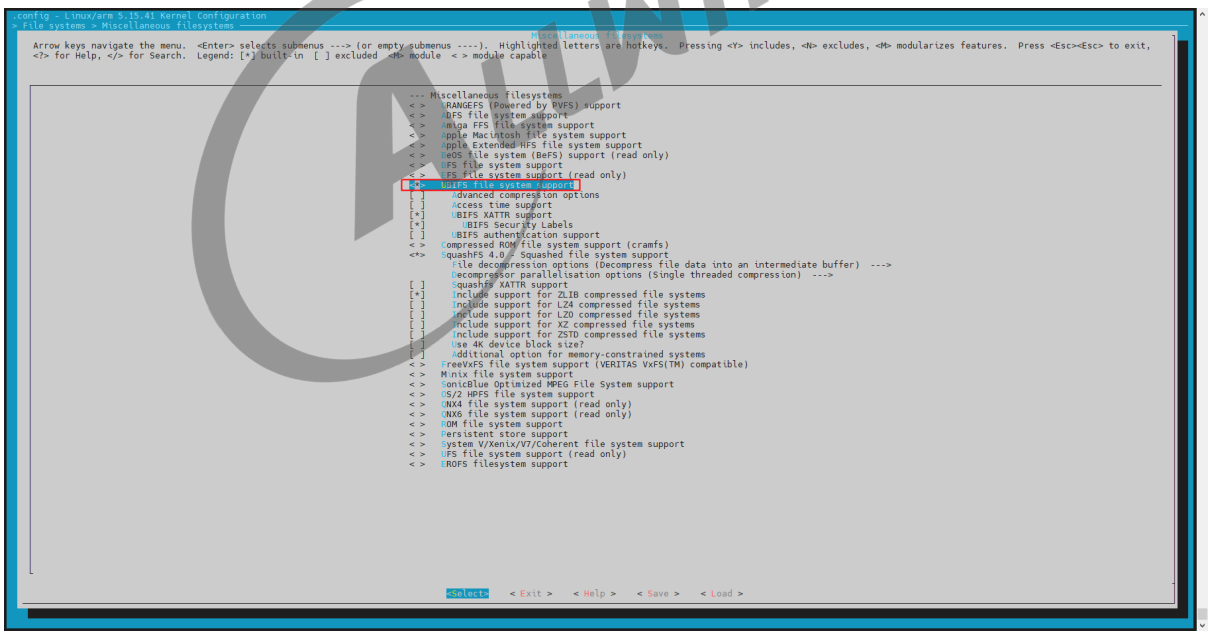

图 4-21: UBIFS_Support

General setup
[ ] Initial RAM filesystem and RAM disk (initramfs/initrd) support

版权所有 © 珠海全志科技股份有限公司。保留一切权利

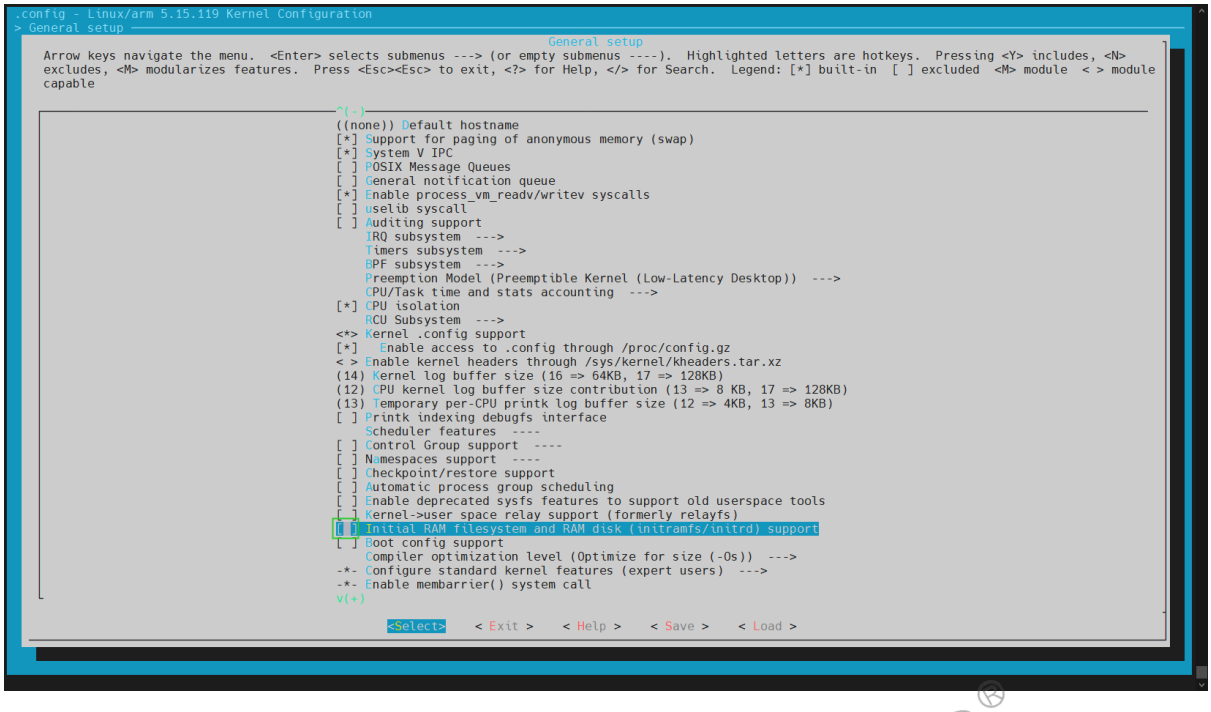

图 4-22: menuconfig_gengeral_setup_ram

4.2.4 env.cfg

在 env.cfg 中添加修改下值，setargs_nand_ubi 先 copy 一份 setargs_nand 再添加对应变量

路径：device/config/chips/平台（T113-i）/configs/evb1_auto_nand/bsp/env.cfg

```
nand_root=ubi0_4 #这里指定rootfs所对应的ubi卷号，卷号由rootfs分区在sys_partition.fex中的位置确定，mbr为ubi0_0，
      以此类推计算rootfs卷号
mtd_name=sys
rootfstype=ubifs,rw
setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} rootfstype=${rootfstype}
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

5 关键数据定义与接口说明

5.1 关键数据定义

5.1.1 flash 设备信息数据结构

```
struct aw_spinand_phy_info {
    const char *Model;
    unsigned char NandID[MAX_ID_LEN];
    unsigned int DieCntPerChip;
    unsigned int BlkCntPerDie;
    unsigned int PageCntPerBlk;
    unsigned int SectCntPerPage;
    unsigned int OobSizePerPage;
#define BAD_BLK_FLAG_MARK          0x03
#define BAD_BLK_FLAG_FRIST_1_PAGE      0x00
#define BAD_BLK_FLAG_FIRST_2_PAGE      0x01
#define BAD_BLK_FLAG_LAST_1_PAGE       0x02
#define BAD_BLK_FLAG_LAST_2_PAGE       0x03
    int BadBlockFlag;
#define SPINAND_DUAL_READ          BIT(0)
#define SPINAND_QUAD_READ          BIT(1)
#define SPINAND_QUAD_PROGRAM           BIT(2)
#define SPINAND_QUAD_NO_NEED_ENABLE    BIT(3)
#define SPINAND_ONEDUMMY_AFTER_RANDOMREAD  BIT(8)
    int OperationOpt;
    int MaxEraseTimes;
#define HAS_EXT_ECC_SE01           BIT(0)
#define HAS_EXT_ECC_STATUS         BIT(1)
    enum ecc_status_shift ecc_status_shift;
    int EccFlag;
    enum ecc_limit_err EccType;
    enum ecc_oob_protected EccProtectedType;
};
```

说明：

- Model：flash 的 model 名字
- NandID：flash 的 id 码
- DieCntPerChip：每 chip 的 die 个数
- BlkCntPerDie：每 die 有多少个 block
- PageCntPerBlk：每 block 有多少个 page
- SectCntPerPage：每 page 有多少个扇区
- OobSizePerPage：每 page 的 obb 大小
- BadBlockFlag：坏块标志存放在每个 block 的那个 page 中

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1. BAD_BLK_FLAG_FRIST_1_PAGE
2. BAD_BLK_FLAG_FIRST_2_PAGE
3. BAD_BLK_FLAG_LAST_1_PAGE
4. BAD_BLK_FLAG_LAST_2_PAGE

- OperationOpt：支持的操作

1. SPINAND_DUAL_READ
2. SPINAND_QUAD_READ
3. SPINAND_QUAD_PROGRAM
4. SPINAND_QUAD_NO_NEED_ENABLE
5. SPINAND_ONEDUMMY_AFTER_RANDOMREAD

- MaxEraseTimes：最大擦除数据
- EccFlag：特性物料读 ecc status 所需 EccFlag 不同
- GD5F1GQ4UCYIG 通过 0Fh + C0h 获取 ecc status，则无需配置 EccFlag
- MX35LF1GE4AB 通过 7Ch + one dummy byte 获取 ecc status，则配置 EccFlag = HAS_EXT_ECC_STATUS
- EccType：设置 ecc 值对应的状态关系
- EccProtectedType：在 spare 去选择受 ecc 保护的 16byte 作为 oob 区

例（MX35LF2GE4AD）：

```
{
    .Model       = "MX35LF2GE4AD",
    .NandID      = {0xc2, 0x26, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff},
    .DieCntPerChip  = 1,
    .SectCntPerPage = 4,
    .PageCntPerBlk  = 64,
    .BlkCntPerDie   = 2048,
    .OobSizePerPage = 64,
    .OperationOpt   = SPINAND_QUAD_READ | SPINAND_QUAD_PROGRAM |
        SPINAND_DUAL_READ,
    .MaxEraseTimes  = 65000,
    .EccFlag     = HAS_EXT_ECC_STATUS,
    .EccType     = BIT4_LIMIT5_TO_8_ERR9_TO_15,
    .EccProtectedType = SIZE16_OFF4_LEN4_OFF8,
    .BadBlockFlag = BAD_BLK_FLAG_FIRST_2_PAGE,
},
```

5.1.2 flash chip 数据结构

```
struct aw_spinand_chip {
    struct aw_spinand_chip_ops *ops;
    struct aw_spinand_ecc *ecc;
    struct aw_spinand_cache *cache;
    struct aw_spinand_info *info;
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
    struct aw_spinand_bbt *bbt;
    struct spi_device *spi;
    unsigned int rx_bit;
    unsigned int tx_bit;
    unsigned int freq;
    void *priv;
};
```

此结构定义了 flash chip 层的物理模型数据结构以及 chip 层对 flash 的操作接口。

- aw_spinand_chip_ops：flash 读、写、擦等操作接口
- aw_spinand_ecc：flash ecc 读、写和校验操作接口
- aw_spinand_cache：对缓存 page 的管理，提高读写效率
- aw_spinand_info：flash ID、page size 等信息及获取信息的操作接口
- aw_spinand_bbt：flash 坏块表及管理等操作接口
- spi_device：spi 父设备的操作结构体
- rx_bit：读状态操作标志
- tx_bit：写状态操作标志

5.1.3 aw_spinand_chip_request

```
struct aw_spinand_chip_request {
    unsigned int block;
    unsigned int page;
    unsigned int pageoff;
    unsigned int ooblen;
    unsigned int datalen;
    void *databuf;
    void *oobbuf;

    unsigned int oobleft;
    unsigned int dataleft;
};
```

操作目标结构体，改结构体填充我们待操作的 block 的那个 page 的多少偏移的数据 databuf/oobbuf

- block：待操作块
- page：待操作页
- pageoff：操作偏移
- ooblen：操作 oob 长度
- datalen：操作数据长度
- databuf：操作目标数据
- oobbuf：操作目标 oob

版权所有 © 珠海全志科技股份有限公司。保留一切权利

5.1.4 ubi_ec_hdr

```
struct ubi_ec_hdr {
    __be32  magic;
    __u8    version;
    __u8    padding1[3];
    __be64  ec; /* Warning: the current limit is 31-bit anyway! */
    __be32  vid_hdr_offset;
    __be32  data_offset;
    __be32  image_seq;
    __u8    padding2[32];
    __be32  hdr_crc;
} __packed;
```

@magic: erase counter header magic number (%UBI_EC_HDR_MAGIC)

@version: version of UBI implementation which is supposed to accept this UBI image

@padding1: reserved for future, zeroes

@ec: the erase counter

@vid_hdr_offset: where the VID header starts

@data_offset: where the user data start

@image_seq: image sequence number

@padding2: reserved for future, zeroes

@hdr_crc: erase counter header CRC checksum

EC: Erase Count，记录块的擦除次数，在 ubiattach 的时候指定一个 mtd，如果 PEB 上没有 EC，则用平均的 EC 值，写入 EC 值只有在擦除的时候才会增加 1

5.1.5 ubi_vid_hdr

```
struct ubi_vid_hdr {
    __be32  magic;
    __u8    version;
    __u8    vol_type;
    __u8    copy_flag;
    __u8    compat;
    __be32  vol_id;
    __be32  lnum;
    __u8    padding1[4];
    __be32  data_size;
    __be32  used_ebs;
    __be32  data_pad;
    __be32  data_crc;
    __u8    padding2[4];
    __be64  sqnum;
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
    __u8  padding3[12];
    __be32 hdr_crc;
} __packed;
```

@magic: volume identifier header magic number (%UBI_VID_HDR_MAGIC)

@version: UBI implementation version which is supposed to accept this UBI image (%UBI_VERSION)

@vol_type: volume type (%UBI_VID_DYNAMIC or %UBI_VID_STATIC)

@copy_flag: if this logical eraseblock was copied from another physical eraseblock (for wear-leveling reasons)

@compat: compatibility of this volume(%0, %UBI_COMPAT_DELETE, %UBI_COMPAT_IGNORE, %UBI_COMPAT_PRESERVE, or %UBI_COMPAT_REJECT)

@vol_id: ID of this volume

@lnum: logical eraseblock number

@padding1: reserved for future, zeroes

@data_size: how many bytes of data this logical eraseblock contains

@used_ebs: total number of used logical eraseblocks in this volume

@data_pad: how many bytes at the end of this physical eraseblock are not used

@data_crc: CRC checksum of the data stored in this logical eraseblock

@padding2: reserved for future, zeroes

@sqnum: sequence number

@padding3: reserved for future, zeroes

@hdr_crc: volume identifier header CRC checksum

参数说明

@sqnum 是创建此 VID 头时的全局序列计数器的值。每次 UBI 写一个新的 VID 头到 flash 时，全局序列计数器都会增加，比如当它将一个逻辑的 eraseblock 映射到一个新的物理的 eraseblock 时。全局序列计数器是一个无符号 64 位整数，我们假设它永远不会溢出。@sqnum(序列号) 用于区分新旧版本的逻辑擦除块。

有两种情况，可能有多个物理 eraseblock 对应同一个逻辑 eraseblock，即在卷标识头中有相同的 **@vol_id** 和 **@lnum** 值。假设我们有一个逻辑的擦除块 L，它被映射到物理的擦除块 P。

1. 因为 UBI 可以异步擦除物理上的擦除块，所以可能出现以下情况:L 被异步擦除，所以 P 被安排擦除，然后 L 被写入，即。映射到另一个物理的擦除块 P1，所以 P1 被写入，然后不干净的重

版权所有 © 珠海全志科技股份有限公司。保留一切权利

启发生。结果-有两个物理的 eraseblock P 和 P1 对应同一个逻辑的 eraseblock L。但是 P1 的序列号更大，所以 UBI 在连接 flash 时选择 P1。

2. UBI 不时地将逻辑擦除块移动到其他物理擦除块，以达到损耗均衡的目的。例如，如果 UBI 将 L 从 P 移动到 P1，在 P 被物理擦除之前会发生不干净的重启，有两个物理擦除块 P 和 P1 对应于 L, UBI 必须在 flash 连接时选择其中一个。**@sqnum** 字段表示哪个 PEB 是原始的 (显然 P 的 **@sqnum** 更低) 和副本。但是选择具有更高序列号的物理擦除块是不够的，因为不干净的重新引导可能发生在复制过程的中间，因此 P 中的数据被损坏（P->P1 没复制完）。仅仅选择序号较低的物理擦除块是不够的，因为那里的数据可能很旧 (考虑在复制之后向 P1 添加更多数据的情况)。此外，不干净的重启可能发生在擦除 P 刚刚开始的时候，所以它会导致不稳定的 P，“大部分” 是 OK 的，但仍然有不稳定的情况。

UBI 使用 **@copy_flag** 字段表示这个逻辑擦除块是一个副本。UBI 还计算数据的 CRC，当数据被移动时，并将其存储在副本 (P1) 的 **@data_crc** 字段。因此，当 UBI 需要从两个 (P 或 P1) 中选择一个物理擦除块时，会检查新块 (P1) 的 **@copy_flag**。如果它被清除，情况就简单了，新的就会被选中。如果设置了该值，则检查副本 (P1) 的数据 CRC。如果 CRC 校验和是正确的，这个物理擦除块被选中 (P1)。否则，将选择较老的 P。

如果是静态卷，**@data_crc** 字段包含逻辑擦除块内容的 CRC 校验和。对于动态卷，它不包含 CRC 校验和规则。唯一的例外情况是，当物理擦除块的数据被磨损均衡子系统移动时，磨损均衡子系统计算数据 CRC，并将其存储在 **@data_crc** 字段中。

@used_ebs 字段仅用于静态卷，它表示该卷的数据需要多少个擦除块。对于动态卷，这个字段不被使用并且总是包含 0。

@data_pad 在创建卷时使用对齐参数计算。因此，@data_pad 字段有效地减少了该卷的逻辑擦除块的大小。当一个人在 UBI 卷上使用面向块的软件 (比如，cramfs) 时，这是非常方便的。

LEB 与 PEB

MULTIPLANE 方案，data 区域为 504 个扇区，非 MULTIPLANE 方案，data 区域为 248 个扇区

版权所有 © 珠海全志科技股份有限公司。保留一切权利

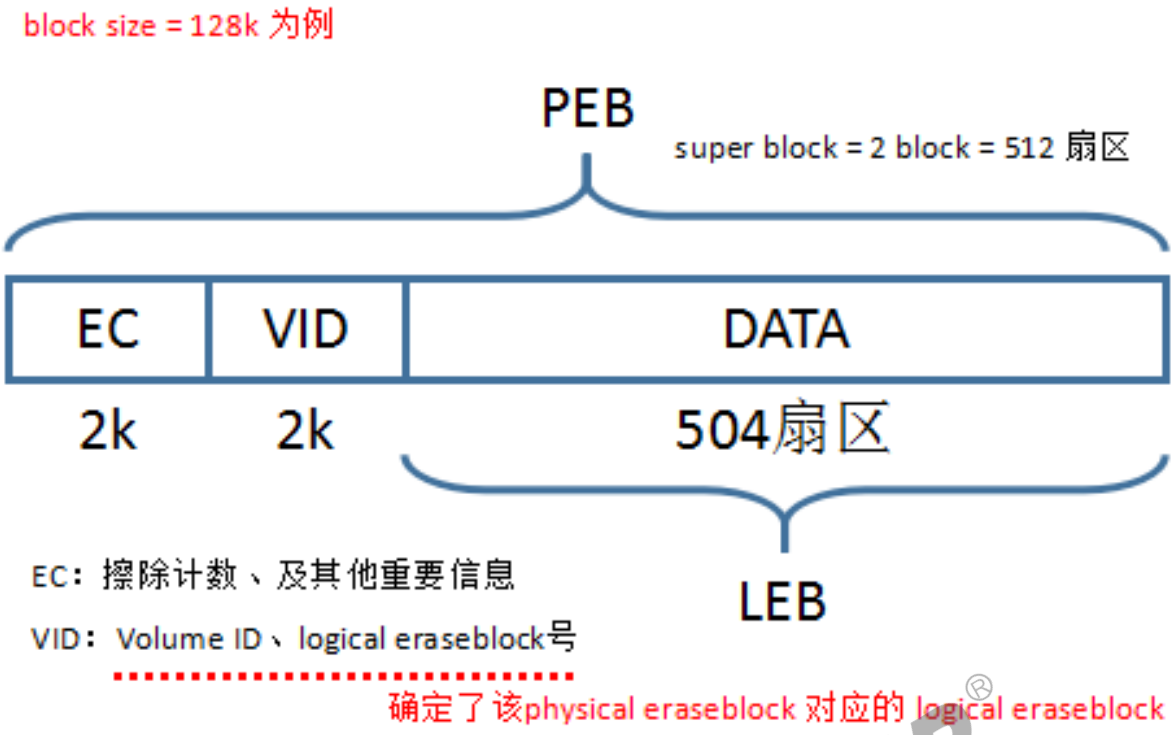

图 5-1: PEB-LEB

5.2 关键接口说明

5.2.1 MTD 层接口

5.2.1.1 aw_spinand_erase

```
static int aw_spinand_erase(struct mtd_info *mtd, struct erase_info *instr)
```

description： mtd erase interface

@mtd： MTD device structure

@instr： erase operation descrition structure

return： success return 0，fail return fail code

5.2.1.2 aw_spinand_read

```
static int aw_spinand_read(struct mtd_info *mtd, loff_t from, size_t len,size_t *retlen, u_char *buf)
```

description： mtd read interface

版权所有 © 珠海全志科技股份有限公司。保留一切权利

@mtd： MTD device structure

@from： offset to read from MTD device

@len： data len

@retlen： had read data len

@buf： data buffer

return： success return max_bitflips，fail return fail code

5.2.1.3 aw_spinand_read_oob

```
static int aw_spinand_read_oob(struct mtd_info *mtd, loff_t from, struct mtd_oob_ops *ops)
```

description： mtd read data with oob

@mtd： MTD device structure

@ops： oob eperation descrition structure

return： success return max_bitflips，fail return fail code

5.2.1.4 aw_spinand_write

```
static int aw_spinand_write(struct mtd_info *mtd, loff_t to, size_t len, size_t *retlen, const u_char *buf)
```

description： mtd write data interface

@to： offset to MTD device

@len： want write data len

@retlen： return the writen len

@buf： data buffer

return： success return 0, fail return code fail

5.2.1.5 aw_spinand_write_oob

```
static int aw_spinand_write_oob(struct mtd_info *mtd, loff_t to, struct mtd_oob_ops *ops)
```

description： write data with oob

版权所有 © 珠海全志科技股份有限公司。保留一切权利

@mtd： MTD device structure

@to： offset to MTD device

@ops： oob operation descrition structure

return： success return 0, fail return code fail

5.2.1.6 aw_spinand_block_isbad

```
static int aw_spinand_block_isbad(struct mtd_info *mtd, loff_t ofs)
```

description： check block is badblock or not

@mtd： MTD device structure

@ofs: offset the mtd device start (align to simu block size)

return： true if the block is bad, or false if the block is good

5.2.1.7 aw_spinand_block_markbad

```
static int aw_spinand_block_markbad(struct mtd_info *mtd, loff_t ofs)
```

description： mark block at the given offset as bad block

@mtd： MTD device structure

@ofs： offset the mtd device start

return： success to mark return 0, or fail return fail code.

5.2.2 物理层接口

5.2.2.1 aw_spinand_chip_read_single_page

```
static int aw_spinand_chip_read_single_page(struct aw_spinand_chip *chip,
        struct aw_spinand_chip_request *req)
```

description： Read physics on a page

@chip： See 3.3.2

@req： See 3.3.3

版权所有 © 珠海全志科技股份有限公司。保留一切权利

return： zero on success, else a negative error code.

5.2.2.2 aw_spinand_chip_write_single_page

```
static int aw_spinand_chip_write_single_page(struct aw_spinand_chip *chip,
        struct aw_spinand_chip_request *req)
```

description： Write physics on a page

@chip： See 3.3.2

@req： See 3.3.3

return： zero on success, else a negative error code.

5.2.2.3 aw_spinand_chip_erase_single_block

```
static int aw_spinand_chip_erase_single_block(struct aw_spinand_chip *chip,
        struct aw_spinand_chip_request *req)
```

description： Erase physics on a block

@chip： See 3.3.2

@req： See 3.3.3

return： zero on success, else a negative error code.

5.2.2.4 aw_spinand_chip_isbad_single_block

```
static int aw_spinand_chip_isbad_single_block(struct aw_spinand_chip *chip,
        struct aw_spinand_chip_request *req)
```

description： Set to bad block

@chip： See 3.3.2

@req： See 3.3.3

return： zero on success, else a negative error code.

版权所有 © 珠海全志科技股份有限公司。保留一切权利

5.2.2.5 aw_spinand_chip_markbad_single_block

```
static int aw_spinand_chip_markbad_single_block(struct aw_spinand_chip *chip,
        struct aw_spinand_chip_request *req)
```

description： Set to bad block

@chip： See 3.3.2

@req： See 3.3.3

return： zero on success, else a negative error code.

5.2.3 Uboot 应用接口

5.2.3.1 sunxi_flash_nand_probe

```
static int sunxi_flash_nand_probe(void)
```

description： MTD layer and SPINAND || RAWNAND initialization，Set the storage type。

return： zero on success, else a negative error code.

5.2.3.2 sunxi_flash_nand_init

```
static int sunxi_flash_nand_init(int boot_mode, int res)
```

description： MTD layer and SPINAND || RAWNAND initialization。

boot_mode： Working mode

res： The default is 0

return： zero on success, else a negative error code.

5.2.3.3 sunxi_flash_nand_exit

```
int spinand_mtd_exit(void)
```

description： Release registration is a resource for applications.

return： zero on success, else a negative error code.

版权所有 © 珠海全志科技股份有限公司。保留一切权利

5.2.3.4 sunxi_flash_nand_write

```
static int sunxi_flash_nand_write(uint start_block, uint nblock, void *buffer)
```

description： mtd write data interface.

start_block： want write start block

nblock： want write block count

buffer： data buffer

return： zero on success, else a negative error code.

5.2.3.5 sunxi_flash_nand_read

```
static int sunxi_flash_nand_read(uint start_block, uint nblock, void *buffer)
```

description： mtd readdata interface.

start_block： want read start block

nblock： want read block count

buffer： data buffer

return： zero on success, else a negative error code.

5.2.3.6 sunxi_flash_nand_erase

```
static int sunxi_flash_nand_erase(int erase, void *mbr_buffer)
```

description： erase boot || partition data.

erase： erase flag

buffer： The default is NULL

return： zero on success, else a negative error code.

5.2.3.7 sunxi_flash_nand_force_erase

```
int spinand_mtd_force_erase(void)
```

description： erase boot & partition data.

版权所有 © 珠海全志科技股份有限公司。保留一切权利

return： zero on success, else a negative error code.

5.2.3.8 sunxi_flash_nand_flush

```
int ubi_nand_flush(void)
```

description： Flush physical cache data to flash.

return： zero on success, else a negative error code.

5.2.3.9 sunxi_flash_nand_download_spl

```
static int sunxi_flash_nand_download_spl(unsigned char *buf, int len, unsigned int ext)
```

description： write boot0.

buf： boot0 data buffer

len： boot0 data len

ext： storage type

return： zero on success, else a negative error code.

5.2.3.10 sunxi_flash_nand_download_toc

```
static int sunxi_flash_nand_download_toc(unsigned char *buf, int len,  unsigned int ext)
```

description： write uboot.

buf： uboot data buffer

len： uboot data len

ext： storage type

return： zero on success, else a negative error code.

版权所有 © 珠海全志科技股份有限公司。保留一切权利

6 功能开发

在 ubi 卷上模拟 mtdblock 设备，挂载块设备文件系统

1. 在 sys_partition*.fex 中添加分区（大小要求对齐到 504 扇区）；
2. 在内核配置中打开 CONFIG_MTD_BLOCK、CONFIG_MTD_UBI_GLUEBI；
3. 编译、打包、烧录固件；
4. 对应的块设备为/dev/mtdblock* ，具体序号可以从后往前对应 sys_partition*.fex 文件中的分区
5. 如果 sys_partition*.fex 中没有指定 downloadfile，挂载前需要格式化：mkfs.vfat /dev/mtdblock12
6. 挂载分区：mkdir /mnt/test1 & mount -t vfat /dev/mtd12 /mnt/test1；

6.1 功能概述

主要介绍 boot、内核态、用户态访问 flash 时的流程说明。

6.2 开发流程

6.2.1 BOOT0 读取数据

```
/* 头文件依赖 */
#include <spinand_boot0.h>

__s32 SpiNand_Read(__u32 sector_num, void *buffer, __u32 N)
```

参数说明：

sector_num：起始扇区，一个扇区等于 512byte

buffer：存放数据的缓存

N：要读取的 page 数

版权所有 © 珠海全志科技股份有限公司。保留一切权利

6.2.2 uboot shell 使用

mem_addr：内存地址，0x40000000 之后可以随便选取如：0x45000000，0x46000000

part_name：分区文件名，boot-resource、env、boot、rootfs

size：可以省略，默认读取整个分区文件

1. sunxi_flash read [size] 读取 flash 中的分区文件到内存中

例：使用 sunxi_flash read 命令将 boot 分区读入到 0x49000000 中，然后使用 md 命令读取 0x49000000 中的内容。

```
=> sunxi_flash read  0x49000000 boot
partinfo: name boot, start 0x2620, size 0x3c80
=> md 0x49000000
49000000: 52444e41 2144494f 003b52b0 40008000    ANDROID!.R;....@
49000010: 003cfac7 41000000 00000000 40f00000    ..<....A.......@
49000020: 40000100 00000800 00000000 00000000    ...@............
49000030: 386e7573 72615f69 0000006d 00000000    sun8i_arm.......
49000040: 00000000 00000000 00000000 00000000    ................
49000050: 00000000 00000000 00000000 00000000    ................
```

图 6-1: sunxi flash read

验证方法：

1. 0x49000000 读入前与读入后数据有没有发生变化
2. 在 **out/pack_out** 目录下找到对应的分区文件，使用 **hexdump -Cv boot.fex -n 500** 命令输出分区文件的数据，对比一致即读入成功。

```
guanyanfei@AwExdroid100:~/workspace/longanV853/out/pack_out$ hexdump -Cv boot.fex -n 500
00000000  41 4e 44 52 4f 49 44 21  b0 52 3b 00 00 80 00 40  |ANDROID!.R;....@|
00000010  c7 fa 3c 00 00 00 00 41  00 00 00 00 00 00 f0 40  |..<....A.......@|
00000020  00 01 00 40 00 08 00 00  00 00 00 00 00 00 00 00  |...@............|
00000030  73 75 6e 38 69 5f 61 72  6d 00 00 00 00 00 00 00  |sun8i_arm.......|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

图 6-2: hexdump

2. sunxi_flash write [size] 将内存中的数据，写入到分区中

例：

1）使用 mm 命令修改内存内容

版权所有 © 珠海全志科技股份有限公司。保留一切权利

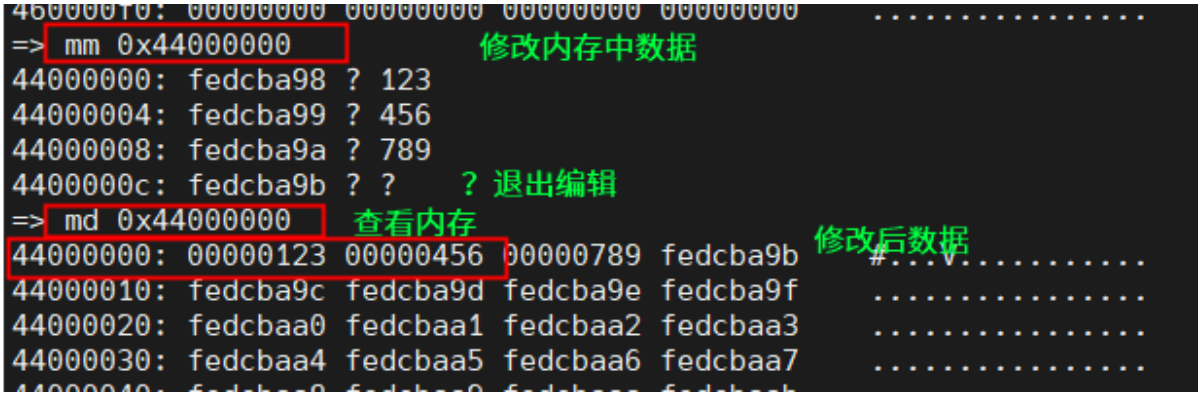

图 6-3: mm - md

2）使用 sunxi_flash write 0x44000000 env 将内存中的数据写入 env 分区

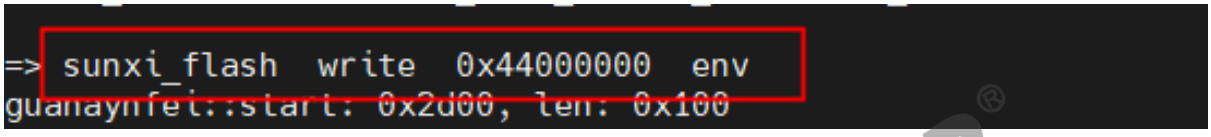

图 6-4: sunxi flash write

3）重新将 env 分区读入内存中，对比一致表示写入成功

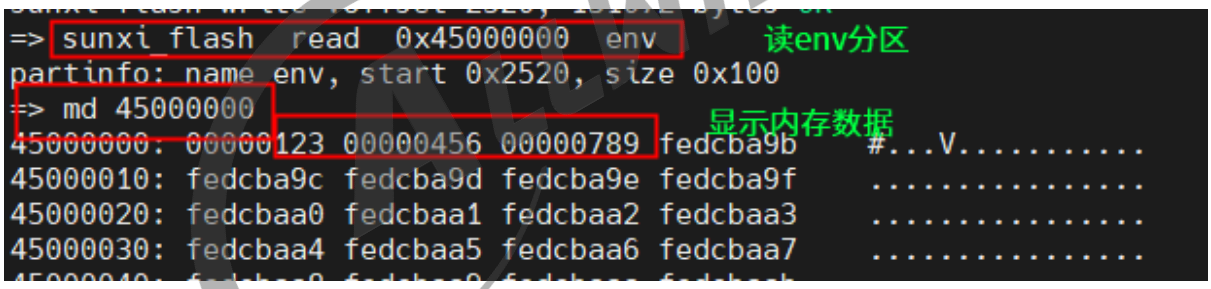

图 6-5: sunxi flash read2

6.2.3 内核态访问 flash

代码示例

```
/* 头文件依赖 */
#include <linux/fs.h>
#include <linux/uaccess.h>

char part_name = "/dev/ubo0_0";
struct file *fp;
mm_segment_t fs;

fp = filp_open(part_name, 0_RDONLT, 0444);
if (IS_ERR(fp)) {
   printk("open %s error\n", part_name);
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
    return -1;
}

fs =get_fs();
set_fs(KERNEL_DS);

ret = vfs_read(fp, buf, len, pos);
ret = vfs_write(fp, buf, len, pos);

filp_close(fp, NULL);
set_fs(fs);
```

下面注意说明一下 vfs_write 接口

```
vfs_write(struct file *file, const char __user *buf, size_t count, loff_t *pos)
```

file：传入 filp_open 的返回值

buf：要写入的数据 buf

count：要写入的数据大小，byte 为单位

pos：要写入的数据偏移，byte 为单位

6.2.4 用户态访问 flash

代码示例

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <linux/mtd/mtd-user.h>

#define DEVICE "/dev/mtdblock0" // 替换为实际的 mtdblock 设备节点
#define BUFFER_SIZE 4096

int main() {
    int fd;
    char buffer[BUFFER_SIZE];

    // 打开 mtdblock 设备
    fd = open(DEVICE, O_RDWR);
    if (fd < 0) {
        perror("Failed to open mtdblock device");
        return 1;
    }

    // 读取数据
    ssize_t bytes_read = read(fd, buffer, BUFFER_SIZE);
    if (bytes_read < 0) {
        perror("Failed to read from mtdblock device");
        close(fd);
        return 1;
    }
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
    printf("Read %zd bytes from %s\n", bytes_read, DEVICE);

    // 打印读取的数据
    for (ssize_t i = 0; i < bytes_read; i++) {
        printf("%02x ", (unsigned char)buffer[i]);
    }
    printf("\n");

    // 写入数据
    const char *data_to_write = "Hello MTD!";
    ssize_t bytes_written = write(fd, data_to_write, strlen(data_to_write));
    if (bytes_written < 0) {
        perror("Failed to write to mtdblock device");
        close(fd);
        return 1;
    }
    printf("Wrote %zd bytes to %s\n", bytes_written, DEVICE);

    // 关闭设备
    close(fd);
    return 0;
}
```

参数介绍

- **DEVICE**: 设备节点的路径，例如 /dev/mtdblock0。根据系统中的实际设备进行修改。
- **open()**: 用于打开设备文件。参数 O_RDWR 表示以读写方式打开设备。
- **read(fd, buffer, BUFFER_SIZE)**:
- **fd**: open() 返回的文件描述符。
- **buffer**: 存储读取数据的缓冲区。
- **BUFFER_SIZE**: 要读取的字节数。
- **write(fd, data_to_write, strlen(data_to_write))**:
- **fd**: open() 返回的文件描述符。
- **data_to_write**: 要写入的数据。
- **strlen(data_to_write)**: 要写入的字节数。

6.2.5 在 ubi 卷上模拟 mtdblock 设备，挂载块设备文件系统

1. 在 sys_partition*.fex 中添加分区（大小要求对齐到 504 扇区）；
2. 在内核配置中打开 CONFIG_MTD_BLOCK、CONFIG_MTD_UBI_GLUEBI；
3. 编译、打包、烧录固件；
4. 对应的块设备为/dev/mtdblock* ，具体序号可以从后往前对应 sys_partition*.fex 文件中的分区
5. 如果 sys_partition*.fex 中没有指定 downloadfile，挂载前需要格式化：mkfs.vfat /dev/mtdblock12
6. 挂载分区：mkdir /mnt/test1 & mount -t vfat /dev/mtd12 /mnt/test1；

版权所有 © 珠海全志科技股份有限公司。保留一切权利

6.3 注意事项

1. **设备节点**: 确保使用正确的设备节点路径，设备节点可能因系统配置而异。
2. **权限**: 确保你有足够的权限访问 mtdblock 设备，通常需要 root 权限。
3. **数据对齐**: MTD 设备的读写通常需要特定的对齐，具体取决于硬件特性。
4. **数据持久性**: 在写入数据之前，了解设备的擦除块和页面大小，以避免损坏数据。
5. **错误处理**: 在实际应用中，建议在每个系统调用后添加错误处理代码，以处理可能的失败情况。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

7 日志分析

```
sunxi:sunxi-spinand:[INFO]: AW SPINand MTD Layer Version: 2.7 20240110
sunxi:sunxi-spinand-phy:[INFO]: AW SPINand Phy Layer Version: 1.13 20231109
sunxi:sunxi-spinand-phy:[INFO]: detect munufacture from id table: GD
sunxi:sunxi-spinand-phy:[INFO]: detect spinand id: ffffd1c8 ffffffff
sunxi:sunxi-spinand-phy:[INFO]: ========== arch info ==========
sunxi:sunxi-spinand-phy:[INFO]: Model:            GD5F1GQ4UBYIG
sunxi:sunxi-spinand-phy:[INFO]: Munufacture:      GD
sunxi:sunxi-spinand-phy:[INFO]: DieCntPerChip:    1
sunxi:sunxi-spinand-phy:[INFO]: BlkCntPerDie:     1024
sunxi:sunxi-spinand-phy:[INFO]: PageCntPerBlk:    64
sunxi:sunxi-spinand-phy:[INFO]: SectCntPerPage:   4
sunxi:sunxi-spinand-phy:[INFO]: OobSizePerPage:   64
sunxi:sunxi-spinand-phy:[INFO]: BadBlockFlag:     0x0
sunxi:sunxi-spinand-phy:[INFO]: OperationOpt:     0x7
sunxi:sunxi-spinand-phy:[INFO]: MaxEraseTimes:    50000
sunxi:sunxi-spinand-phy:[INFO]: EccFlag:          0x1
sunxi:sunxi-spinand-phy:[INFO]: EccType:          10
sunxi:sunxi-spinand-phy:[INFO]: EccProtectedType: 4
sunxi:sunxi-spinand-phy:[INFO]: ====================================
sunxi:sunxi-spinand-phy:[INFO]:
sunxi:sunxi-spinand-phy:[INFO]: ========== physical info ==========
sunxi:sunxi-spinand-phy:[INFO]: TotalSize:   128 M
sunxi:sunxi-spinand-phy:[INFO]: SectorSize:  512 B
sunxi:sunxi-spinand-phy:[INFO]: PageSize:    2 K
sunxi:sunxi-spinand-phy:[INFO]: BlockSize:   128 K
sunxi:sunxi-spinand-phy:[INFO]: OOBSize:     64 B
sunxi:sunxi-spinand-phy:[INFO]: ====================================
sunxi:sunxi-spinand-phy:[INFO]:
sunxi:sunxi-spinand-phy:[INFO]: ========== logical info ==========
sunxi:sunxi-spinand-phy:[INFO]: TotalSize:   128 M
sunxi:sunxi-spinand-phy:[INFO]: SectorSize:  512 B
sunxi:sunxi-spinand-phy:[INFO]: PageSize:    2 K
sunxi:sunxi-spinand-phy:[INFO]: BlockSize:   128 K
sunxi:sunxi-spinand-phy:[INFO]: OOBSize:     64 B
sunxi:sunxi-spinand-phy:[INFO]: ====================================
sunxi:sunxi-spinand-phy:[INFO]: block lock register: 0x00
sunxi:sunxi-spinand-phy:[INFO]: feature register: 0x11
sunxi:sunxi-spinand-phy:[INFO]: sunxi physic nand init end
Creating 4 MTD partitions on "nand":
0x000000000000-0x000000100000 : "boot0"
0x000000100000-0x000000500000 : "uboot"
0x000000500000-0x000000600000 : "secure_storage"
0x000000600000-0x000008000000 : "sys"
sunxi:sunxi-spinand-phy:[INFO]: phy blk 1023 is bad
sunxi:sunxi-spinand:[INFO]: not find boot_param in flash
sunxi:sunxi-spinand:[INFO]: Try sample param is sample mode:1 right_delay:35
......
ubi0: attaching mtd3
ubi0: scanning is finished
ubi0: attached mtd3 (name "sys", size 122 MiB)
ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
ubi0: good PEBs: 975, bad PEBs: 1, corrupted PEBs: 0
ubi0: user volume: 9, internal volumes: 1, max. volumes count: 128
ubi0: max/mean erase counter: 2/1, WL threshold: 4096, image sequence number: 0
ubi0: available PEBs: 12, total reserved PEBs: 963, PEBs reserved for bad PEB handling: 19
ubi0: background thread "ubi_bgt0d" started, PID 268
ubi: mtd3 is already attached to ubi0
```

Model：flash 型号

Munufacture: 厂家型号

DieCntPerChip：每个芯片有多少个 die

BlkCntPerDie：每个 die 有多少个 block

PageCntPerBlk：每个 block 有多少个 page

SectCntPerPage：每个 page 有多少个扇区

OobSizePerPage：每个 page oob 区域大小

MaxEraseTimes：最大擦除次数

TotalSize：flash size

SectorSize：每个扇区大小

PageSize：页大小

BlockSize：块大小

phy blk 1023 is bad：1023 为坏块

sample mode:1 right_delay:35：代码 spi 控制器的采样模式，由于电路的传输延时和 nor 设备接受命令后的响应延时，我们在读取数据时，发送完成命令后，需要 delay 一段时间，再发送 clock 进行数据读出，确保数据的准确性，启动对应的采样模式（sample_mode）和采样延时（sample_delay），要烧写时 try 出来，然后启动更新到设备树的。

4 sunxipart partitions：一个 4 个 mtd 分区，对应的起始地址、结束地址和分区名字

```
  起始地址      结束地址      分区名
0x000000000000-0x000000100000 : "boot0"
0x000000100000-0x000000500000 : "uboot"
0x000000500000-0x000000600000 : "secure_storage"
0x000000600000-0x000008000000 : "sys"
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

8 调试方法

挂载 debug 调试节点

```
mount -t debugfs none /sys/kernel/debug/
```

查看 spinand 工作频率

```
echo freq > /sys/kernel/debug/spinand/param
cat /sys/kernel/debug/spinand/status
100000000   //100M
```

查看 spinand 工作模式

```
echo mode > /sys/kernel/debug/spinand/param
cat /sys/kernel/debug/spinand/status
QUAD    //4线
```

查看 spinand flash 相关信息

```
echo info > /sys/kernel/debug/spinand/param
cat /sys/kernel/debug/spinand/status

========== arch info ==========
Model:           GD5F1GQ4UBYIG
DieCntPerChip:     1
BlkCntPerDie:      1024
PageCntPerBlk:     64
SectCntPerPage:    4
OobSizePerPage:    64
BadBlockFlag:      0x0
OperationOpt:      0x7
MaxEraseTimes:     50000
EccFlag:         0x1
EccType:         10
EccProtectedType:  4
==============================
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

9 FAQ

9.1 dram 全盘扫描

使用 DragonHD 工具全盘扫描 dram，排查是否为 dram 稳定性问题。

图 9-1: DragonHD tool

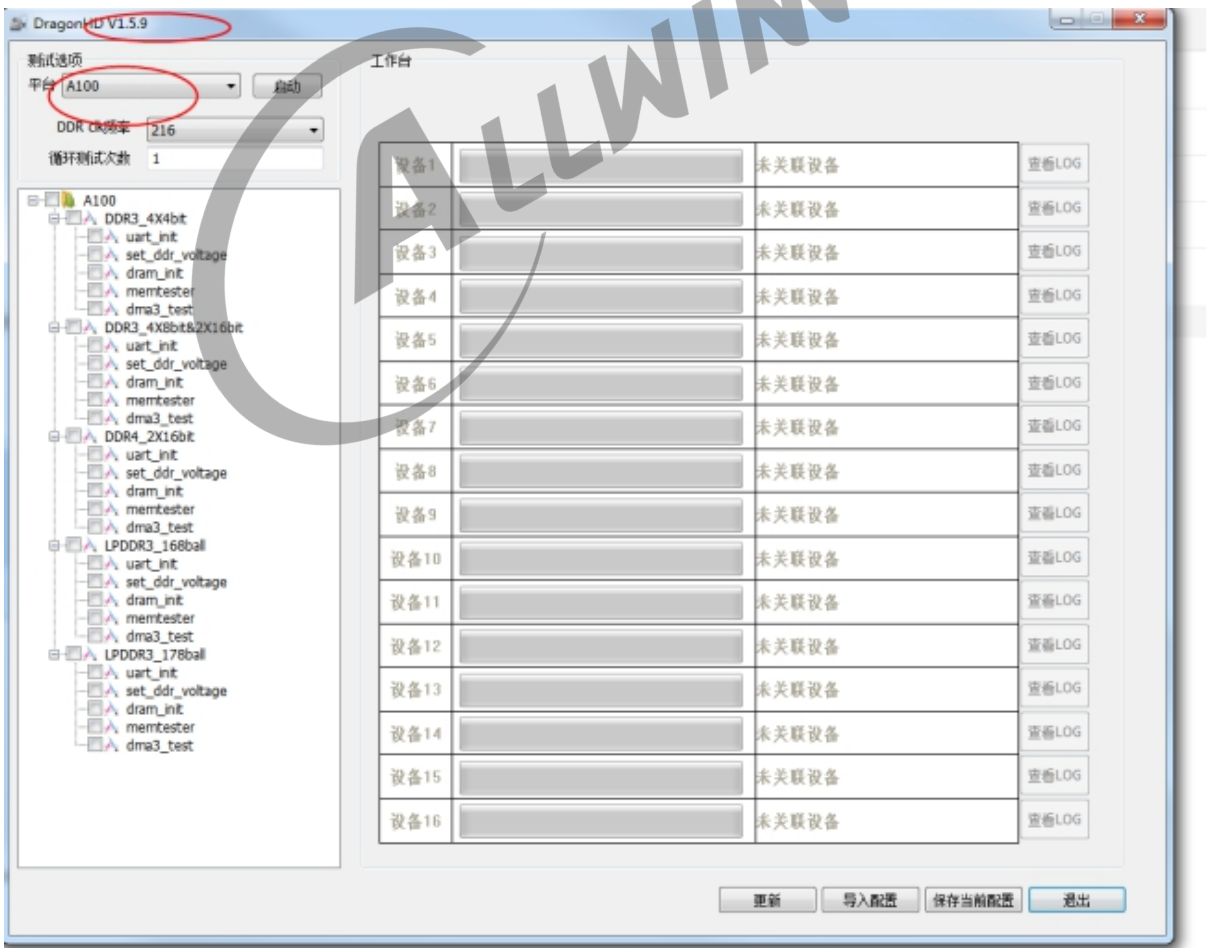

图 9-2: DragonHD tool

版权所有 © 珠海全志科技股份有限公司。保留一切权利

9.2 启动失败

根据启动阶段划分启动失败场景：

```
1.Brom启动boot0失败: 罕见，boot0烧写的有问题
2.boot0读uboot失败：比较少见，建议做dram全盘扫描实验
3.uboot启动失败
  * uboot nand驱动初始化失败
  * uboot加载、校验kernel, env等逻辑分区失败
4.kernel启动失败
  * kernel启动时异常崩溃
  * kernel挂载文件系统异常
```

9.2.1 brom 启动 boot0 失败

- 现象：串口无 boot0 运行标志打印 “HELLO! SBOOT is starting!” 或 “HELLO! BOOT0 is starting”
- 建议：

1. 做 dram 全盘扫描实验，排除烧写时 dram 稳定性问题
2. 请 FAE 检查 boot_select 配置
3. 有可能是 boot0 大小过大，做裁剪 boot0 大小实验
4. 若仍无法定位，交由全志工程师分析处理

9.2.2 boot0 读 uboot 失败

- 现象：串口出现打印 “Can’t find a good Boot1 copy in nand.” 或 “Can’t find a good Boot1 copy in spi nand.”
- 建议：做 dram 全盘扫描实验，排除 dram 稳定性问题，若仍无法定位，交由全志工程师分析处理。

9.3 Ubifs scanning 时间过长

1、优先考虑 spi 的工作频率，工作线宽，cpu 的频率，通过指令测试读性能是否正常。

```
#time dd if=/dev/mtd3 of=/dev/null bs=4096 count=10240
命令分析：
  time：     命令常用于测量一个命令的运行时间。
  dd：       用指定大小的块拷贝一个文件，并在拷贝的同时进行指定的转换。
  if=文件名：  输入文件名，缺省为标准输入。即指定源文件。< if=input file >
  /dev/mtd3：  物理分区，根据你实际使用情况选择，对它的读取会产生IO
  of=文件名：  输出文件名，缺省为标准输出。即指定目的文件。< of=output file >
  /dev/null：  是一个伪设备，相当于回收站，of到该设备不会产生IO（也可以指定存在的文件，但会产生IO）
  bs=bytes：   同时设置读入/输出的块大小为bytes个字节。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

count=blocks：仅拷贝blocks个块，块大小等于ibs指定的字节数。

2、检查内存适配器，导致 attach 较长的主要原因可能是是内存分配器的不同，我们内部默认配置是用 slub，slob 分配时容易引起内存碎片，导致性能问题。slub 分配器是 slab 分配器的进化版。

解决办法：改为用 slub 内存分配器，.config 文件修改 CONFIG_SLUB=y

9.4 Flash 分区与使用说明大小不一致

问题描述：

1. rootfs 分区设置为 96MB，按 512 扇区计算，size 设置为 196608，但是挂载后实际只有 86.8MB。为什么变小了？计算有问题吗？怎么计算的？
2. cat /proc/mtd 擦除块为什么是 0x3F000，但是 boot 分区是 0x40000

```
#
# df -h
Filesystem                Size      Used Available Use% Mounted on
ubi0_5                   86.6M     17.1M     69.5M  20% /
devtmpfs                118.0M         0    118.0M   0% /dev
tmpfs                   122.1M         0    122.1M   0% /dev/shm
tmpfs                   122.1M    776.0K    121.4M   1% /tmp
tmpfs                   122.1M     96.0K    122.0M   0% /run
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00100000 00040000 "boot0"
mtd1: 00400000 00040000 "uboot"
mtd2: 00100000 00040000 "secure_storage"
mtd3: 0fa00000 00040000 "sys"
mtd4: 0007e000 0003f000 "mbr"
mtd5: 0103e000 0003f000 "backup"
mtd6: 000bd000 0003f000 "env"
mtd7: 0081f000 0003f000 "boot"
mtd8: 0013b000 0003f000 "private"
mtd9: 06039000 0003f000 "rootfs"
mtd10: 0042f000 0003f000 "params"
mtd11: 06039000 0003f000 "data"
mtd12: 01332000 0003f000 "UDISK"
#
```

图 9-3: 分区信息

问题一：

df 显示的是文件系统大小，文件系统本身需要额外的空间存储元数据，导致实际空间会比风区大小少。

ubifs 文件系统本身会占用一些空间来管理数据。3 个 LEB + 5% 日志 + LPT (LPT 消耗是动态，按分区比例的）大概会少了 10% 左右，有损耗是正常的，文件系统 ubifs 格式要求的。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

编号	数据类型	说明	用途
1	superblock area	超级块域（LEB0）	占用一个LEB存储superblock node，一般来说，superblock node保存文件系统很少变化的参数。 superblock node仅仅占用LEB0的前4096个字节。
2	master area	主节点域（LEB1, LEB2）	存储主节点，包含了所有flash上没有固定逻辑位置的结构。占用LEB1 LEB2. 一般情况下，这两个LEBs保存着相同数据，master node尺寸为512 bytes，每次写入master node会顺序的使用LEB的空闲page，直到没有空闲page时，再从offset zero开始写master node，这时会重新unmapped LEBs为另一个erased LEB。 注意，master node不会同时unmapped两个LEBs，因为这会导致文件系统没有有效master node，如果此时掉电，系统无法找到有效master node。
3	log area	日志域(占leb_cnt的5%)	是ubifs日志(journal)的一部分，目的是为了减少对flash上的索引(index)的更新频率，log size在文件系统创建的时候被固定，log area永远不会耗尽空间。 It is the purpose of the log to record where the journal is.
4	LEB properties tree area	属性树域(LPT)	管理同种类型的LEBs。当前LPT area的尺寸是根据LEB size以及文件系统创建时的最大LEB数计算的。 和log area一样，LPT area永远不会耗尽空间；和log area不同的是LPT area不是顺序的，而是随机的。和index一样，LPT区仅仅在commit时更新。
5	orphan area	孤立域(1 leb)	是一个固定数量的擦除块，位于属性树域和主域之间，一般来说，占用一个LEB足以
6	main area	主域	包含了形成文件系统的数据和索引组成的节点

图 9-4: UBIFS 数量类型及损耗说明

问题二：

逻辑擦除块（LEB）与物料擦除块（PEB）

版权所有 © 珠海全志科技股份有限公司。保留一切权利

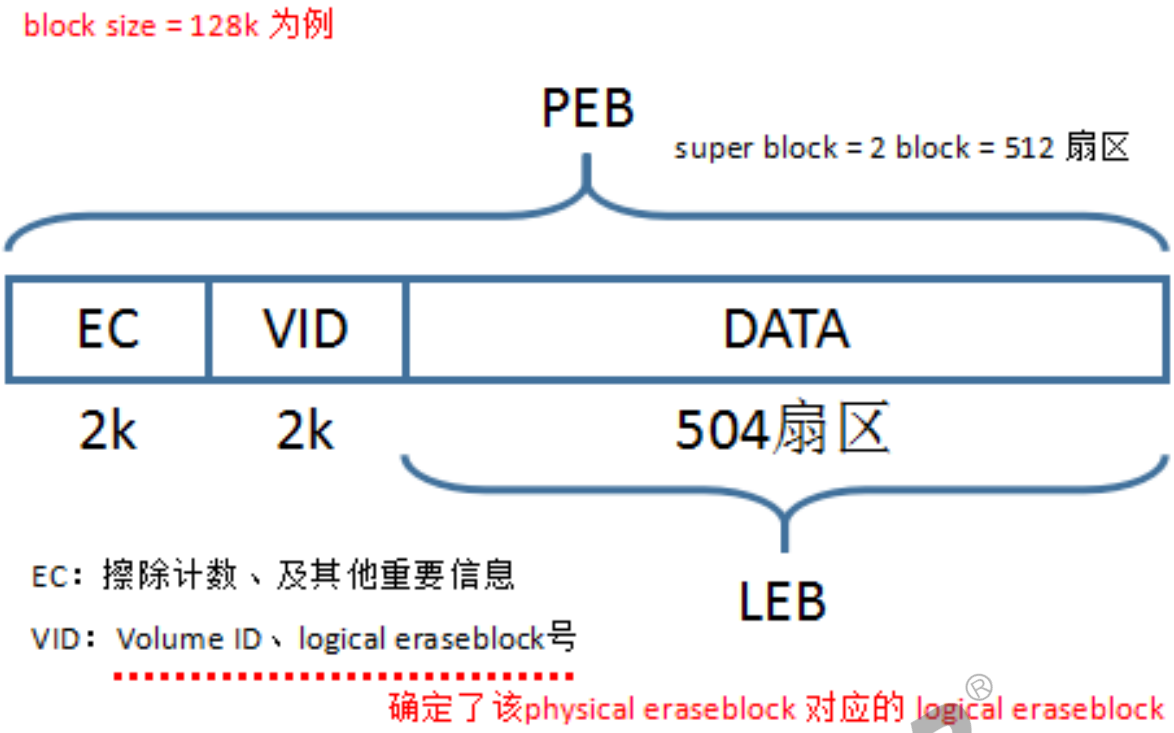

图 9-5: PEB-LEB

mtd4 - 12 走的是 UBI 架构，所以擦除单位是 LEB：0x3F000，mtd0 - 3 的擦除单位是 PEB：0x40000

9.5 fat 挂载在 mtdblock 设备上，sync 数据没有写下

问题描述：fat 挂载在 mtdblock 设备上，sync 数据没有写下（sync 后断电，重启后文件数据不存在）

原因：

1.sync 不保证数据落盘

2. 可以使用 fsync 保证数据落盘 s

版权所有 © 珠海全志科技股份有限公司。保留一切权利

著作权声明

版权所有 ©2024 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、、全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。