Rockchip
瑞芯微电子

RK Flash Storage Application Note

NVM: SPI NOR/ SPI NAND / SLC NAND/ MLC NAND
AP: RV1108/ PX3SE/ RK3036/ RK3229/RK3128/…
2017.Sept.6

Revision History

Confidential

Revision No.	Revised Details	Released Date	Remark
Rev.00	Initial Draft	2017.3.23	ZYF
Rev.01	add storage config in kernel	2017.3.28	ZJW
Rev.02	Add new device guide	2017.4.6	ZYF
Rev.03	Add miniloader and mlc nand	2017.9.6	ZYF

Contents

Confidential

Preview

Confidential

- Mini Loader/UBOOT:

  AP： RK3036, RK3288, RK3128/6, RK3126C, RK3228/9, RK3228-H, RK3328 , RK3366, RK3328, RK3368, RKPX3SE, RKPX5, RK3399…

  NVM: EMMC, MLC NAND, SLC NAND, SD CARD

- Loader:

  AP: RK3036, RK3229, RK3128, RKPX3SE, RV1108

  NVM: EMMC, SLC NAND, SD CARD, SPI NOR, SPI NAND

- rkflash :

  AP: RK3036, RK3229,RK3228, RK3128, RKPX3SE, RV1108

  NVM: SLC NAND, SPI NOR, SPI NAND

- rk_nand / nand ko:

  AP： RK3036, RK3288, RK3128/6, RK3126C, RK3228/9, RK3366, RK3328, RK3368, RKPX3SE, RKPX5…

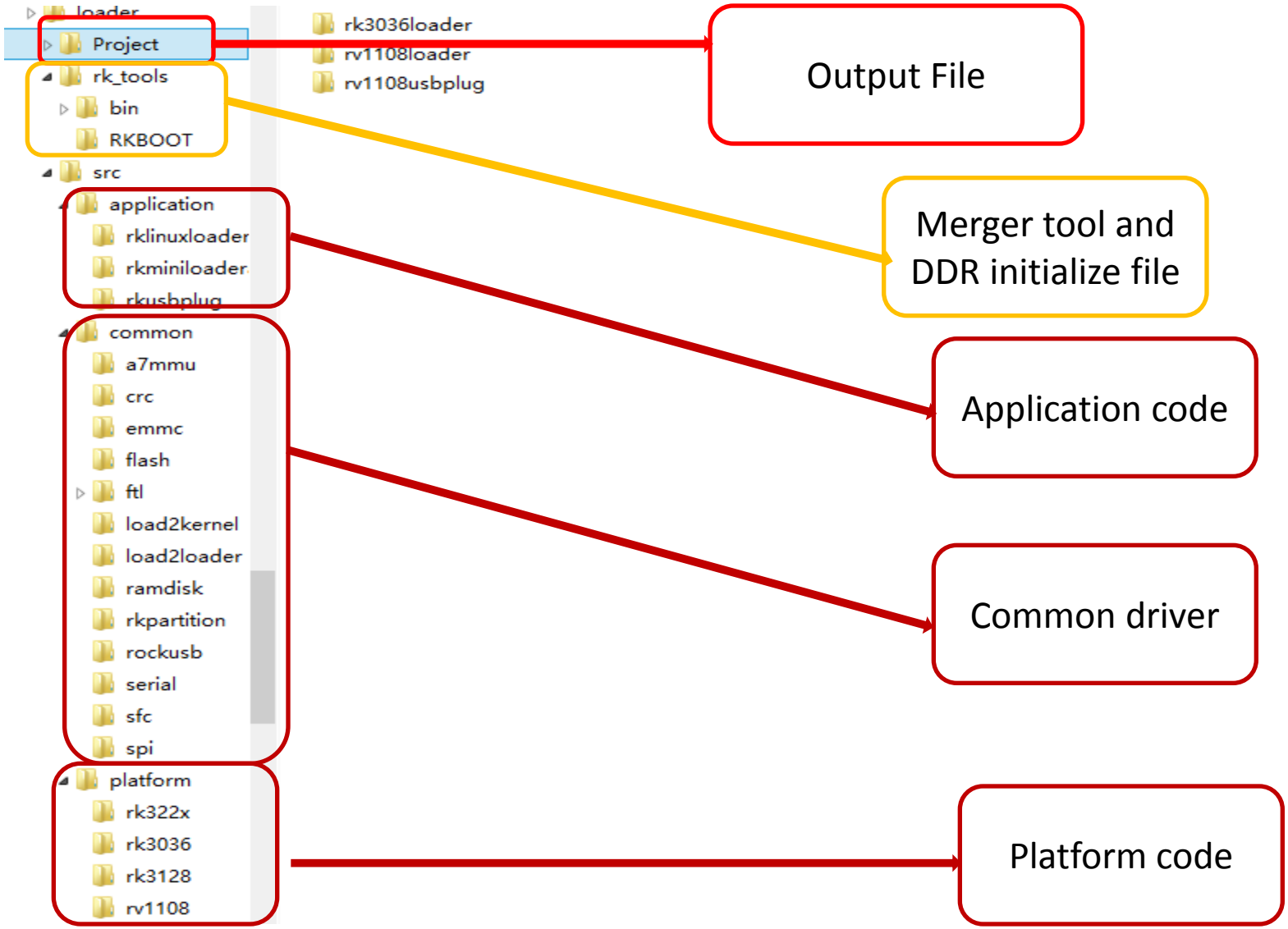
  NVM: MLC NAND, SLC NAND

Note: RK3399， 3228-H and RK3328 do not support NAND FLASH.

Loader Project

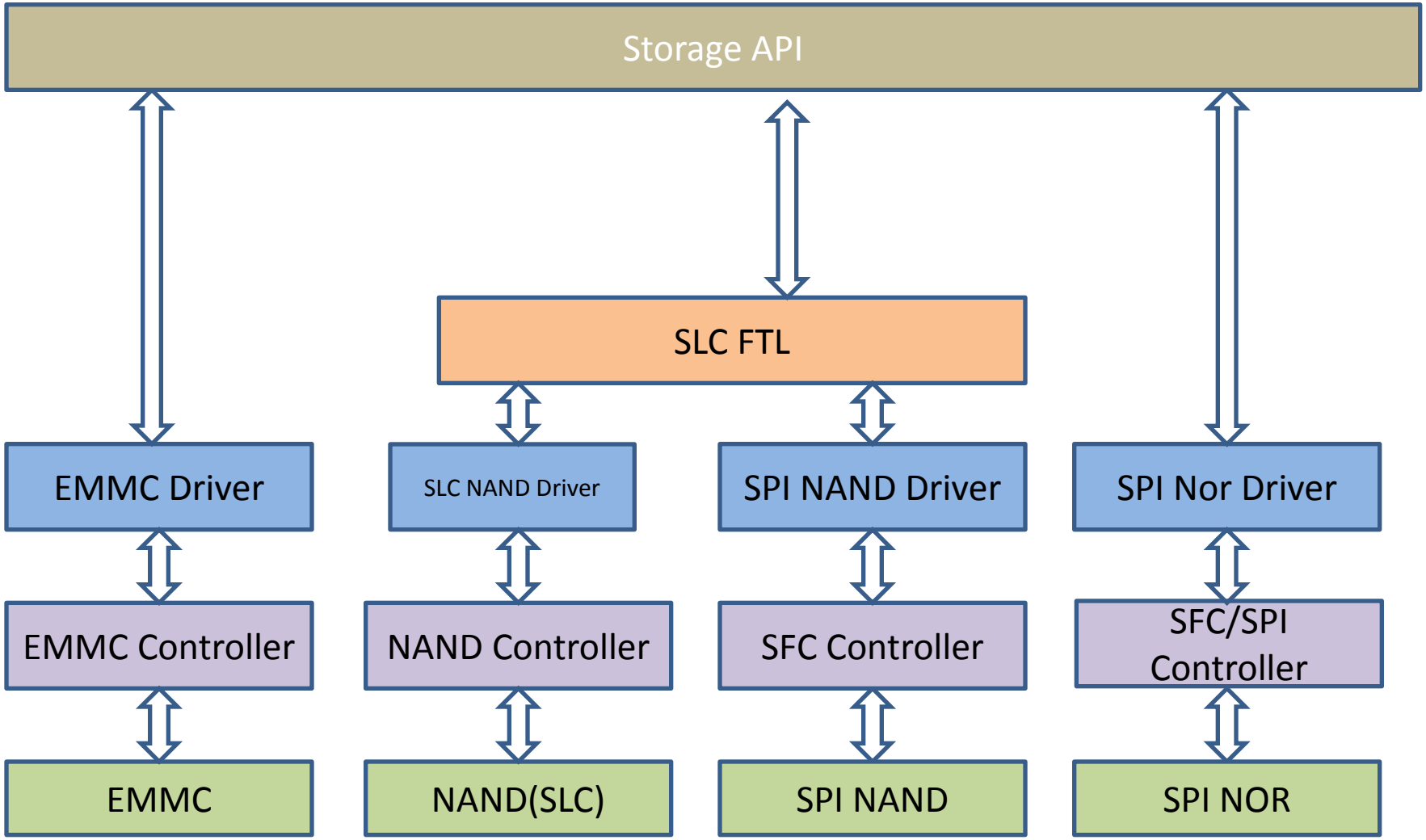
Loader project file overview:

Confidential



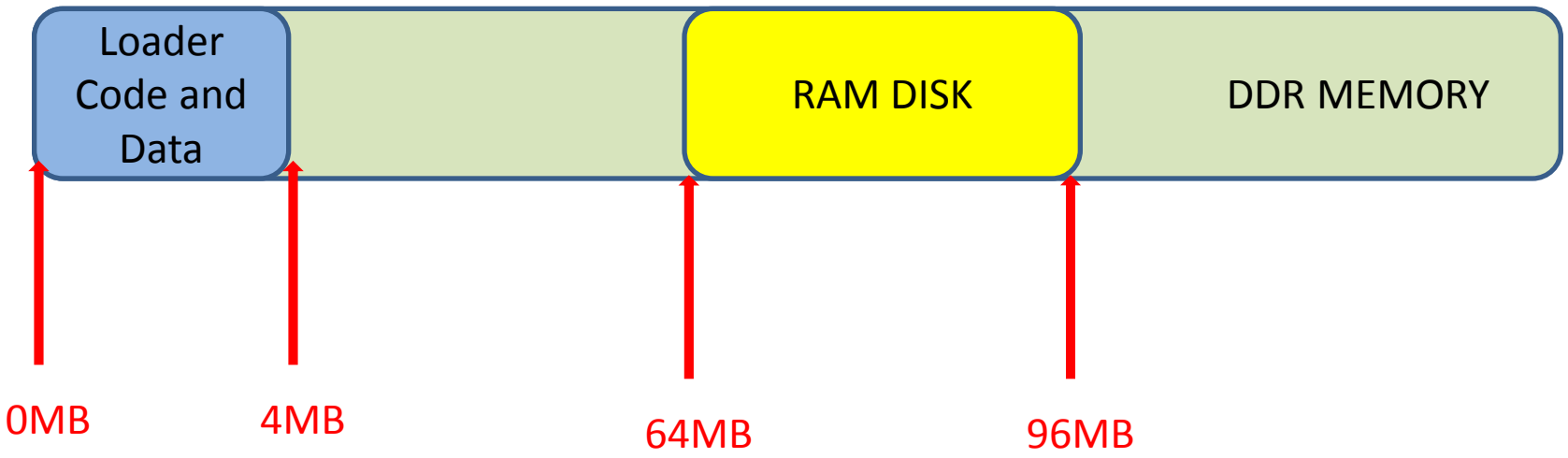
Loader: Storage Architecture

Confidential



Loader: Memory Layout

Confidential



Loader memory range: 0x0000-0000 --- 0x003F-FFFF

RAMDISK memory range: 0x0400-0000 --- 0x05FF-FFFF

Note: RAMDISK is used for fast ram boot via USB.

Storage configure for Loader

Confidential

Configure file: \src\platform\xxx(rv1108)\platconfig.h

platconfig.h

```
13  ******************************************
14
15  #define     PALTFORM    RK30XX
16
17  #define     RK_USB_EN
18  //#define     RK_SDMMC_BOOT_EN
19  #define     RK_SPI_BOOT_EN
20  #define     RK_FLASH_BOOT_EN
21  #define     RK_SPI_NAND_BOOT_EN
22
```

RK_SPI_BOOT_EN: enable spi nor boot

RK_FLASH_BOOT_EN: enable slc nand boot

RK_SPI_NAND_BOOT_EN: enable spi slc nand boot

RK_SDMMC_BOOT_EN: enable emmc boot

Loader: Makefile (1)

Confidential

Cross compiler download:

https://releases.linaro.org/archive/14.07/components/toolchain/binaries/gcc-linaro-arm-none-eabi-4.9-2014.07_linux.tar.bz2

Cross compiler config:

```
#CROSS_COMPILE := ../prebuilts/gcc/linux-x86/arm/arm-linux-androideabi-4.7/bin/arm-linux-androideabi-
CROSS_COMPILE := ../prebuilts/gcc/linux-x86/arm/gcc-linaro-arm-none-eabi-4.9-2014.07_linux/bin/arm-none-eabi-
# Build architecture
```

PLATFORMS confia:

```
PLATFORMS        := "rk322xusbplug,rk322xloader,rk3036usbplug,rk3036loader,rv1108usbplug,rv1108loader,rk
DEFAULT_PLAT     := rk3128usbplug
PLAT             ?= ${DEFAULT_PLAT}
HELP_PLATFORMS   := $(shell echo ${PLATFORMS} | sed 's/ /|/g')
```

Loader: Makefile (2)

Confidential

_USB_PLUG_: compile the code for usbplug.

LOADER: compile the code for loader.

_LOAD_KERNEL_: enable load kernel and run to kernel. Define in usbplug is used for fast ram boot.

MERGET_LOADER: enable to merger usb_boot_loader for download firmware.

```
# Build ARCH
ifeq (${PLAT},rv1108usbplug)
ARCH                    := cortex-a7
LOCAL_CFLAGS            += -D_RV1108_ -D_USB_PLUG_ -D_LOAD_KERNEL_
endif

ifeq (${PLAT},rv1108loader)
ARCH                    := cortex-a7
LOCAL_CFLAGS            += -D_RV1108_  -D_LOADER_ -D_LOAD_KERNEL_
MERGER_LOADER           := 1
RKCHIP                  := RV1108
endif
```

Loader: Build(1)

Confidential

Make help:

```
zyf@fs-server:~/rk30/rk3288_android4.4/loader$ make help
usage: make PLAT=<rk322xusbplug,rk322xloader,rk3036usbplug,rk3036loader,rv1108usbplug,

PLAT is used to specify which platform you wish to build.
If no platform is specified in first time, PLAT defaults to: rk3128usbplug

Supported Targets:
  all                           Build all the project
  clean                         Clean the current platform project
  distclean                     Clean the current project and delete .config
  version VER=[version num]     Add one new BIN for git version control

example: build the targets for the rk3128usbplug project:
  make PLAT=rk3128usbplug
```

make rv1108 loader:

make PLAT=rv1108usbplug

make clean

make

make PLAT=rv1108loader

make clean

make

Loader: Build(2)

Confidential

Output file:

1. USB boot loader

This file is merged with DDR initialization code, used as a NVM agent for upgrade firmware via USB.

```
 4096 Mar 28 11:34 ./
 4096 Mar 27 09:53 ../
   18 Mar 28 11:34 .config
 4096 Mar 24 18:07 .git/
14791 Mar  1 10:26 Makefile*
 4096 Mar 24 18:06 Project/
 4096 Mar  1 10:07 rk_tools/
35502 Mar 28 11:34 RV1108_usb_boot_V1.20.bin
 4096 Mar 10 17:38 src/
```

2. Secondary boot loader

This file is secondary boot loader, it`s a raw binary file, which need be packed into image.

```
zyf 4096 Mar 24 18:06 obj/
k30/rk3288_android4.4/loader$ ll Project/rv1108loader/Debug/bin

zyf   4096 Mar 24 18:06 ./
zyf   4096 Mar 24 18:06 ../
zyf  59588 Mar 24 18:06 rv1108loader.bin*
zyf 605962 Mar 24 18:06 rv1108loader.dump
zyf 320151 Mar 24 18:06 rv1108loader.elf*
zyf 127499 Mar 24 18:06 rv1108loader.map
k30/rk3288_android4.4/loader$
```

New Firmware Image

Firmware image merge

Confidential

The "firmware.img" is merged by "firmwareMerger" tool, partitions are defined in the file "setting.ini", detail information see the file "setting.ini".

```
/rk3288_android4.4/kernel/Image$ ll

drwxr-xrwx  4096 Mar 28 11:44 ./
drwxr-xrwx  4096 Mar 24 14:29 ../
-rwxr-xrwx   451 Dec 23 11:29 3399.ini*
-rwxr-xrwx   624 Dec 23 11:41 3399spi.ini*
-rwxr-xrwx 4460544 Dec 23 11:30 Firmware.img
-rwxr-xrwx    16 Dec 23 11:30 Firmware.md5
-rwxr-xrwx  880923 Aug  1  2016 firmwareMerger*
-rwxr-xrwx   26522 Oct  9 11:02 kernelimage*
-rwxr-xrwx 7724844 Dec 23 11:12 kernel.img*
-rwxr-xrwx   4180 Aug  9  2016 rk1108ddr.bin*
-rwxr-xrwx  51788 Nov 18 16:30 rk1108loader.bin*
-rwxr-xrwx 1608145 Dec 23 11:08 rootfs.img*
-rwxr-xrwx    524 Aug  1  2016 setting.ini*
-rwxr-xrwx 4194304 Dec 23 10:54 trust.img*
-rwxr-xrwx 4194304 Dec 23 10:54 uboot.img*
/rk3288_android4.4/kernel/Image$
```

Merge command:

```
./firmwareMerger -P setting.ini firmware.img
```

Note: The firmware.img can be programed to an NVM device through a third-party programmer.

Setting.ini detail information

Confidential

```
#Flag目前只有两个值,1为分区需要下载,0为不需要下载
#type目前有5种值,0x1=Vendor分区 0x2=IDBlock分区 0x4=Kernel分区 0x8=boot分区 0x80
#PartSize和PartOffset字段的值都是以扇区为单位
[System]
FwVersion=16.12.23
#如果Nano=1,则生成nano的idblock
Nano=
#如果BLANK_GAP=1,则生成的idblock按每2k数据间隔2k空白保存
BLANK_GAP=0
#FILL_BYTE表示分区尾部空白用什么数据填充，默认为0
FILL_BYTE=
[IDBlock]
Flag=1
DDR_Bin=3128\RK3128_DDR3_300M_V2.05.bin
Loader_Bin=3128\rk3128loader.bin
PartOffset=0x40
PartSize=0x180
[UserPart1]
Name=kernel
Type=0x4
Flag=1
File=3128\kernel_new.img
PartOffset=0x200
PartSize=0x4000
```

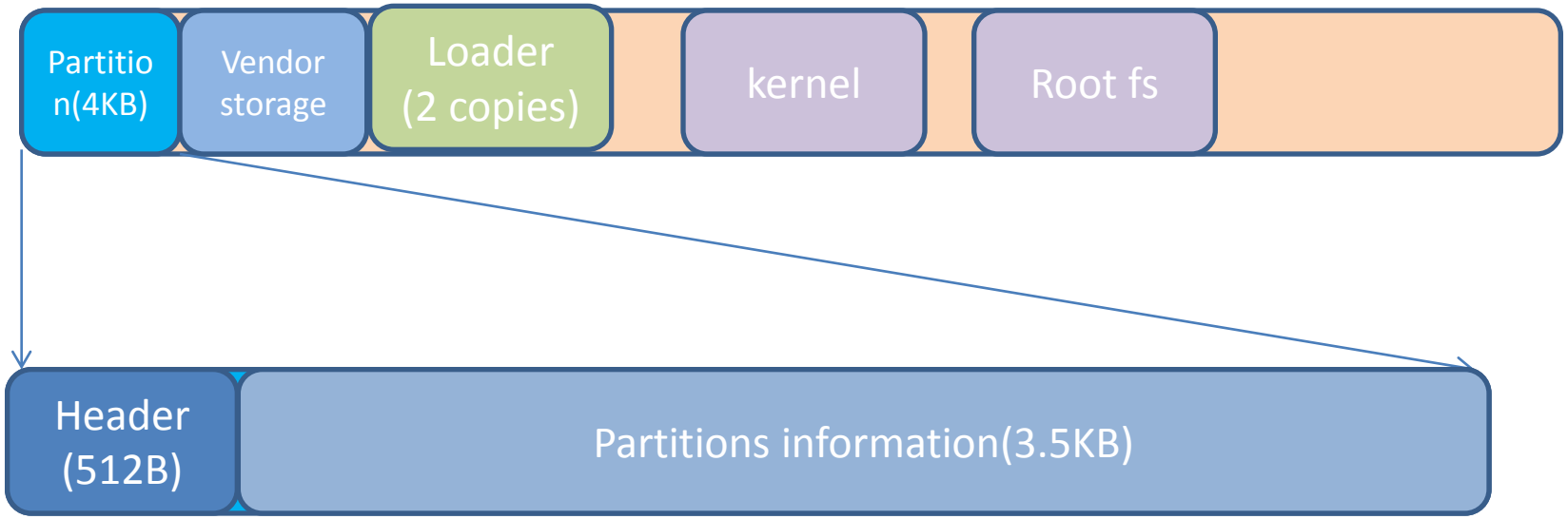
RK312X/RK3229/RK3036/RK3066/RK3188 + SPI NOR need config BLANK_CAP=1

PartOffset=0x40(Fixed),
NOR: (PartSize / 2 + 0x40) need equal to 0x80/0x100/0x200…
EMMC : PartSize/2 need equal to 0x400
NAND/SPI NAND: PartSize/2 >= 0x100

EMMC: kernel partition offset need fix to 0x2000

Firmware image memory layout

Confidential

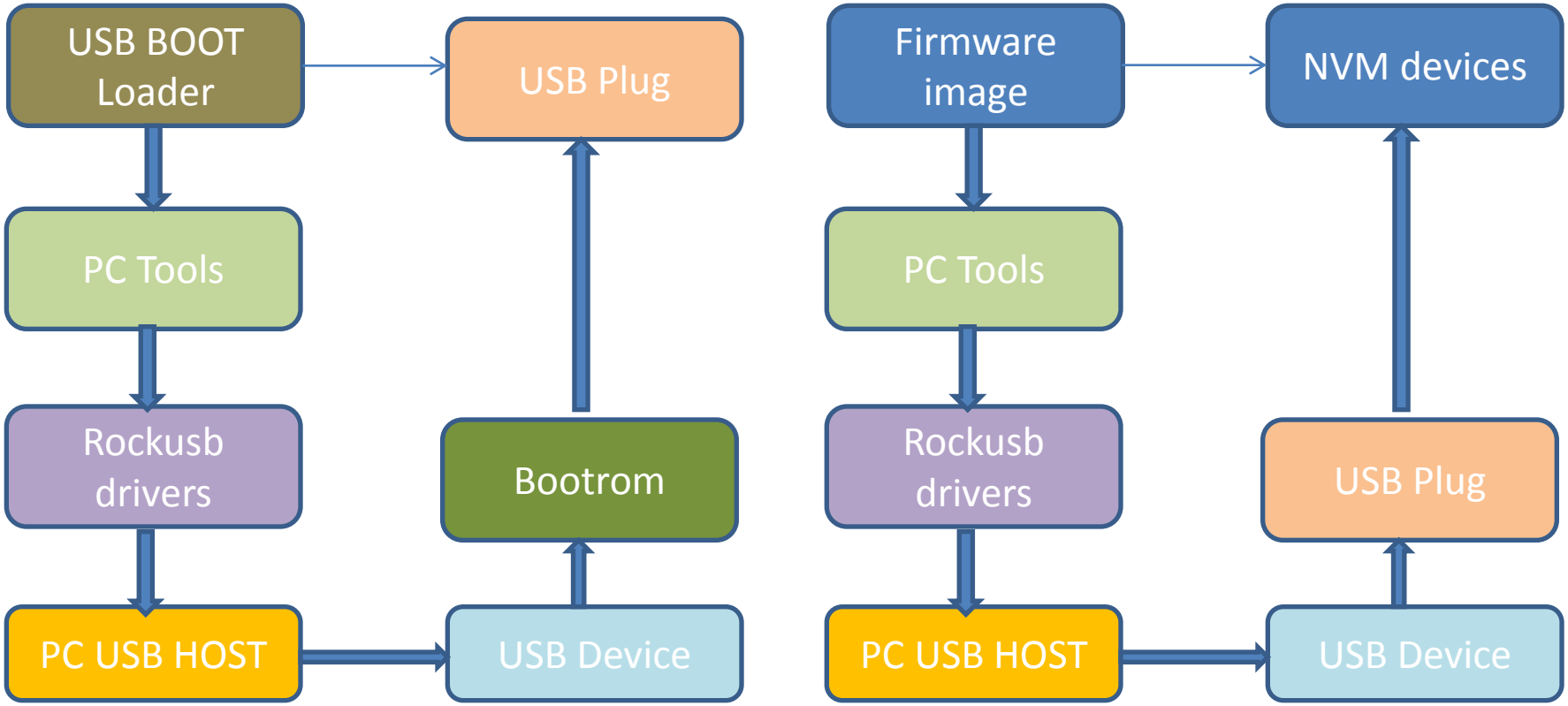


Note：

Detail information see file "src/common/rkpartition/rkpartition.h"

Firmware upgrade

Confidential



Add a new SPI NOR/NAND FLASH

Add a new SPI NOR FLASH(1)

Confidential

SPI NOR FLASH information define：

```
typedef  struct tag_flash_info
{
        uint32 id;

        uint8 block_size;
        uint8 sector_size;
        uint8 read_cmd;
        uint8 prog_cmd;

        uint8 read_cmd_4;
        uint8 prog_cmd_4;
        uint8 sector_erase_cmd;
        uint8 block_erase_cmd;

        uint8 feature;
        uint8 density;  /* (1 << density) sectors*/
        uint8 QE_bits;
        uint8 reserved2;
}flash_info, *pflash_info;
```

Support Nor flash id table:

```
flash_info spi_flash_tbl[] =
{
        {0xc84016, 128, 8, 0x03, 0x02, 0x6B, 0x32, 0x20, 0xD8, 0x0D, 13, 9, 0}, /* GD25Q32B */
        {0xc84017, 128, 8, 0x03, 0x02, 0x6B, 0x32, 0x20, 0xD8, 0x0D, 14, 9, 0}, /* GD25Q64B */
        {0xc84018, 128, 8, 0x03, 0x02, 0x6B, 0x32, 0x20, 0xD8, 0x0D, 15, 9, 0}, /* GD25Q128B */
        {0xc84019, 128, 8, 0x13, 0x12, 0x6C, 0x3E, 0x21, 0xDC, 0x1C, 16, 6, 0}, /* GD25Q256B */
        {0xef4018, 128, 8, 0x03, 0x02, 0x6B, 0x32, 0x20, 0xD8, 0x0C, 15, 9, 0}, /* 25Q128FV */
        {0xef4019, 128, 8, 0x13, 0x02, 0x6C, 0x32, 0x20, 0xD8, 0x3C, 16, 9, 0}, /* 25Q256FV */
};
```

Add a new SPI NOR FLASH(2)

Confidential

id: Nor flash id

block_size: Nor flash block size is 64KB, the value in this field is 128.

sector_size: sector size is 4KB,the value in this field is 8.

read_cmd: X1 read command

prog_cmd: X1 program command

read_cmd_4: X4 read command

prog_cmd_4: X4 program command

sector_erase_cmd: sector erase command

block_erase_cmd: block erase command

density: device size = (1 << density) sectors, 32MB NOR FLASH, density is 16.

QE_bits: QE bits in status register location.

Status Register

S15	S14	S13	S12	S11	S10	S9	S8
SUS	CMP	Reserved	Reserved	Reserved	LB	QE	SRP1

S7	S6	S5	S4	S3	S2	S1	S0
SRP0	BP4	BP3	BP2	BP1	BP0	WEL	WIP

Add a new SPI NOR FLASH(3)

Confidential

Feature:

```
#define FEA_READ_STATUE_MASK     (0x3<<0)
#define FEA_STATUE_MODE1         0
#define FEA_STATUE_MODE2         1
#define FEA_4BIT_READ            (1 << 2)
#define FEA_4BIT_PROG            (1 << 3)
#define FEA_4BYTE_ADDR           (1 << 4)
#define FEA_4BYTE_ADDR_MODE      (1 << 5)
```

FEA_STATUE_MODE1: Only one write status register command.

Write Status Register	01H	(S7-S0)	(S15-S8)

FEA_STATUE_MODE2: Two or three write status register command.

Write Status Register-1	3 & 4	01H	(S7-S0)
Write Status Register-2	3 & 4	31H	(S15-S8)
Write Status Register-3	3 & 4	11H	(S23-S16)

Add a new SPI NOR FLASH(4)

Confidential

Feature:

```
#define FEA_READ_STATUE_MASK     (0x3<<0)
#define FEA_STATUE_MODE1         0
#define FEA_STATUE_MODE2         1
#define FEA_4BIT_READ            (1 << 2)
#define FEA_4BIT_PROG            (1 << 3)
#define FEA_4BYTE_ADDR           (1 << 4)
#define FEA_4BYTE_ADDR_MODE      (1 << 5)
```

FEA_4BIT_READ: Support X4 read mode

FEA_4BIT_PROG: Support X4 program mode

FEA_4BYTE_ADDR: Support unique 4 bytes address command

FEA_4BYTE_ADDR_MODE: Support 4 bytes mode, 4bytes command and 3 bytes are the same command.

Add a new SPI NAND FLASH(1)

Confidential

SPI NAND FLASH information define：

```
typedef  struct tag_nand_info
{
        uint32 id;

        uint16 sec_per_page;
        uint16 page_per_blk;
        uint16 plane_per_die;
        uint16 blk_per_plane;

        uint8 page_read_cmd;
        uint8 page_prog_cmd;
        uint8 read_cache_cmd_1;
        uint8 prog_cache_cmd_1;

        uint8 read_cache_cmd_4;
        uint8 prog_cache_cmd_4;
        uint8 block_erase_cmd;
        uint8 feature;

        uint8 density;  /* (1 << density) sectors*/
        uint8 max_ecc_bits;
        uint8 QE_address;
        uint8 QE_bits;
} ? end tag_nand_info ? nand_info, *pnand_info;
```

Support SPI NAND flash id table:

```
nand_info spi_nand_tbl[] = {
        {0x98C2, 4, 64, 1, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x02, 0xD8, 0x00, 18, 8, 0xB0, 0XFF},
        {0x98CB, 4, 64, 2, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x02, 0xD8, 0x00, 19, 8, 0xB0, 0XFF},
        {0xC212, 4, 64, 1, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x0C, 18, 4, 0xB0, 0}, /*
        {0xC222, 4, 64, 2, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x0C, 19, 4, 0xB0, 0}, /*
        {0xC8F1, 4, 64, 1, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x0C, 18, 4, 0xB0, 0}, /*
        {0x2C12, 4, 64, 1, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x00, 18, 4, 0xB0, 0}, /*
        {0xC8B1, 4, 64, 1, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x0C, 18, 4, 0xB0, 0}, /*
        {0xC8B2, 4, 64, 2, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x0C, 19, 4, 0xB0, 0}, /*
        {0xC8D1, 4, 64, 1, 1024, 0x13, 0x10, 0x03, 0x02, 0x6B, 0x32, 0xD8, 0x0C, 18, 4, 0xB0, 0}, /*
};
```

Add a new SPI NAND FLASH(2)

Confidential

id: SPI NAND FLASH ID

sec_per_page: SPI NAND Flash page size is 2KB, sector value is 4.

page_per_blk: SPI NAND Flash page pre block is 64.

plane_per_die: 128MB is 1, 256 is 2.

blk_per_plane : The block pre plane is 1024.

page_read_cmd: Page Read to cache command

page_prog_cmd: Program Execute command

read_cache_cmd_1: X1 Read From Cache command

prog_cache_cmd_1: X1 Program Load command

read_cache_cmd_4: X4 Read From Cache command

prog_cache_cmd_4: X4 Program Load command

block_erase_cmd: Block Erase command

density: device size = (1 << density) sectors, 128MB NAND FLASH, density is 18.

Add a new SPI NAND FLASH(3)

Confidential

Feature:

```
#define FEA_READ_STATUE_MASK      (0x3<<0)
#define FEA_STATUE_MODE1          0
#define FEA_STATUE_MODE2          1
#define FEA_4BIT_READ             (1 << 2)
#define FEA_4BIT_PROG             (1 << 3)
#define FEA_4BYTE_ADDR            (1 << 4)
#define FEA_4BYTE_ADDR_MODE       (1 << 5)
```

FEA_4BIT_READ: Support X4 read mode

FEA_4BIT_PROG: Support X4 program mode

max_ecc_bits: max support ecc bits.

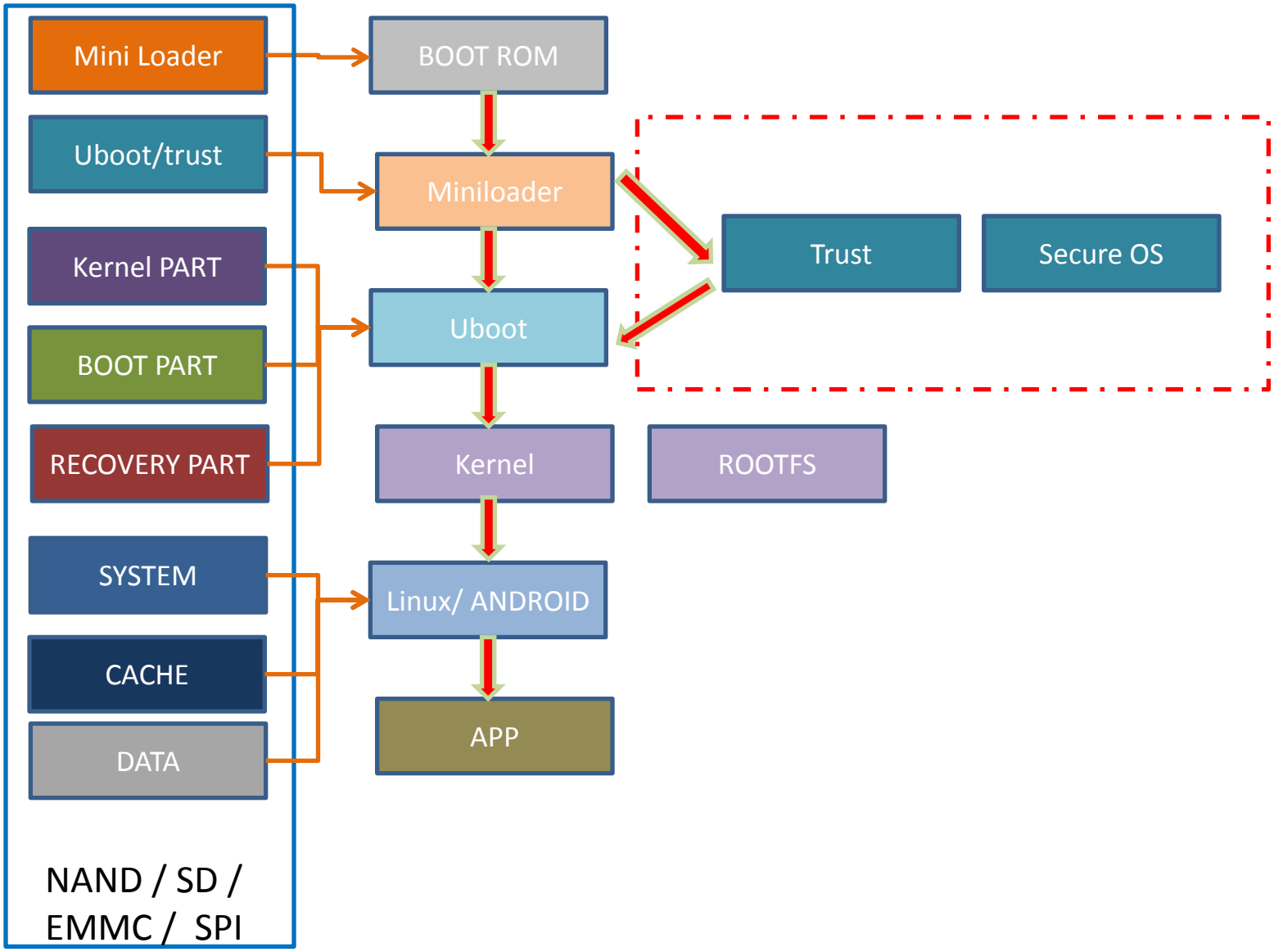
QE_address: QE bits status register address

QE_bits: QE bits location

Register	Addr.	7	6	5	4	3	2	1	0
Protection	A0H	BRWD	Reserved	BP2	BP1	BP0	INV	CMP	Reserved
Feature	B0H	OTP_PRT	OTP_EN	Reserved	ECC_EN	Reserved	Reserved	Reserved	QE
Status	C0H	Reserved	Reserved	ECCS1	ECCS0	P_FAIL	E_FAIL	WEL	OIP

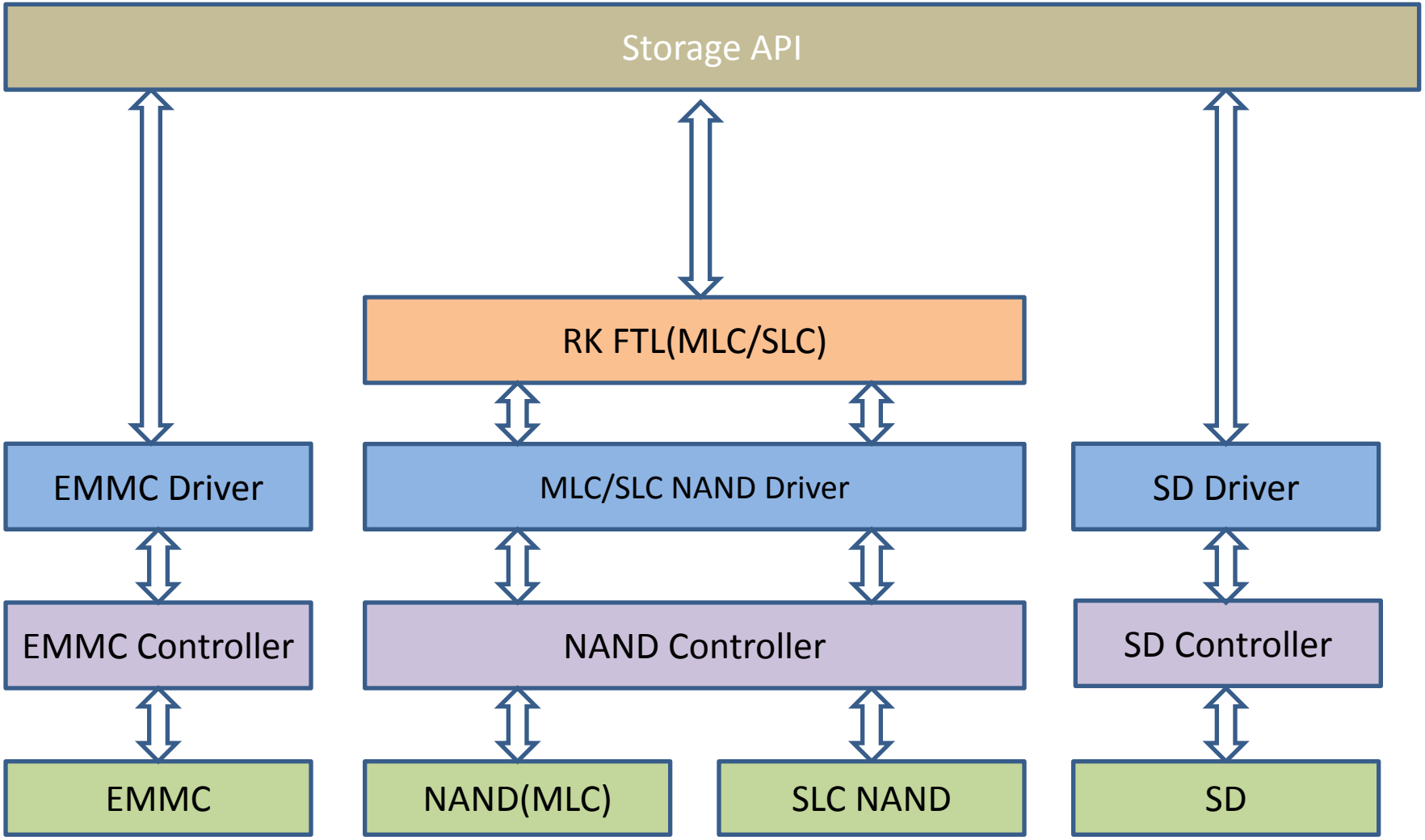
Miniloader and Uboot

System boot flow



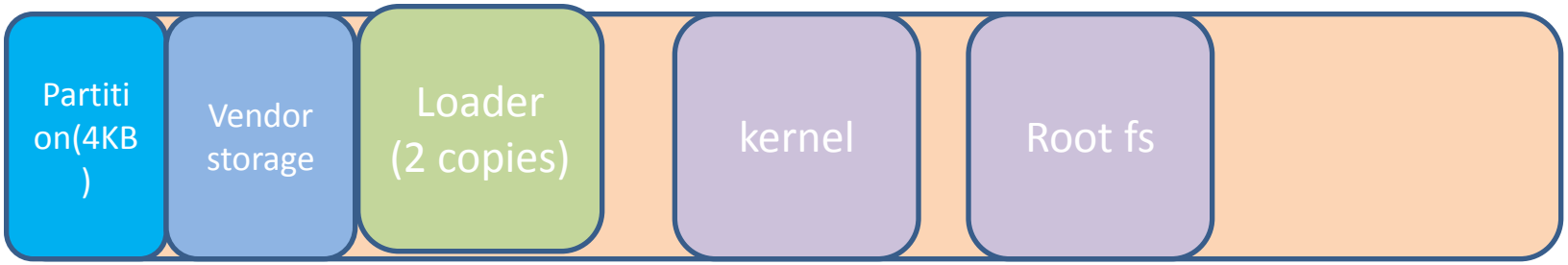
MiniLoader: Storage Architecture

Confidential

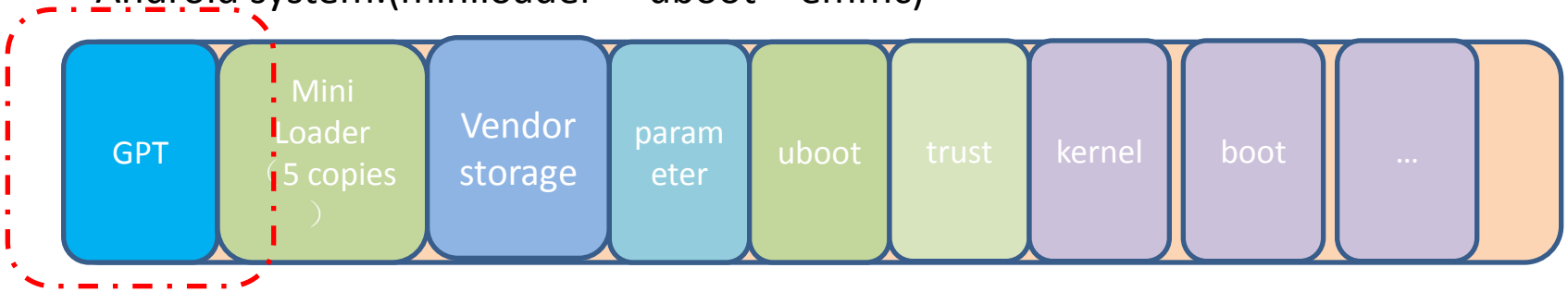


Storage Layout

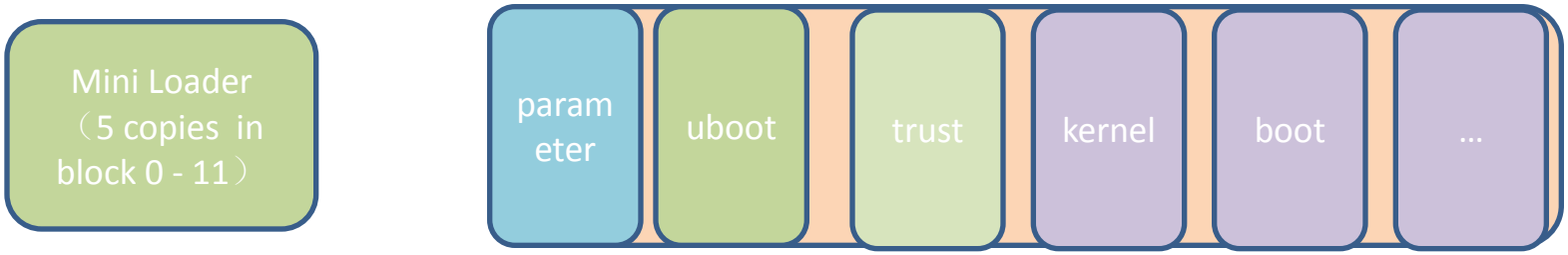
Linux system: (loader)



Android system:(miniloader + uboot + emmc)

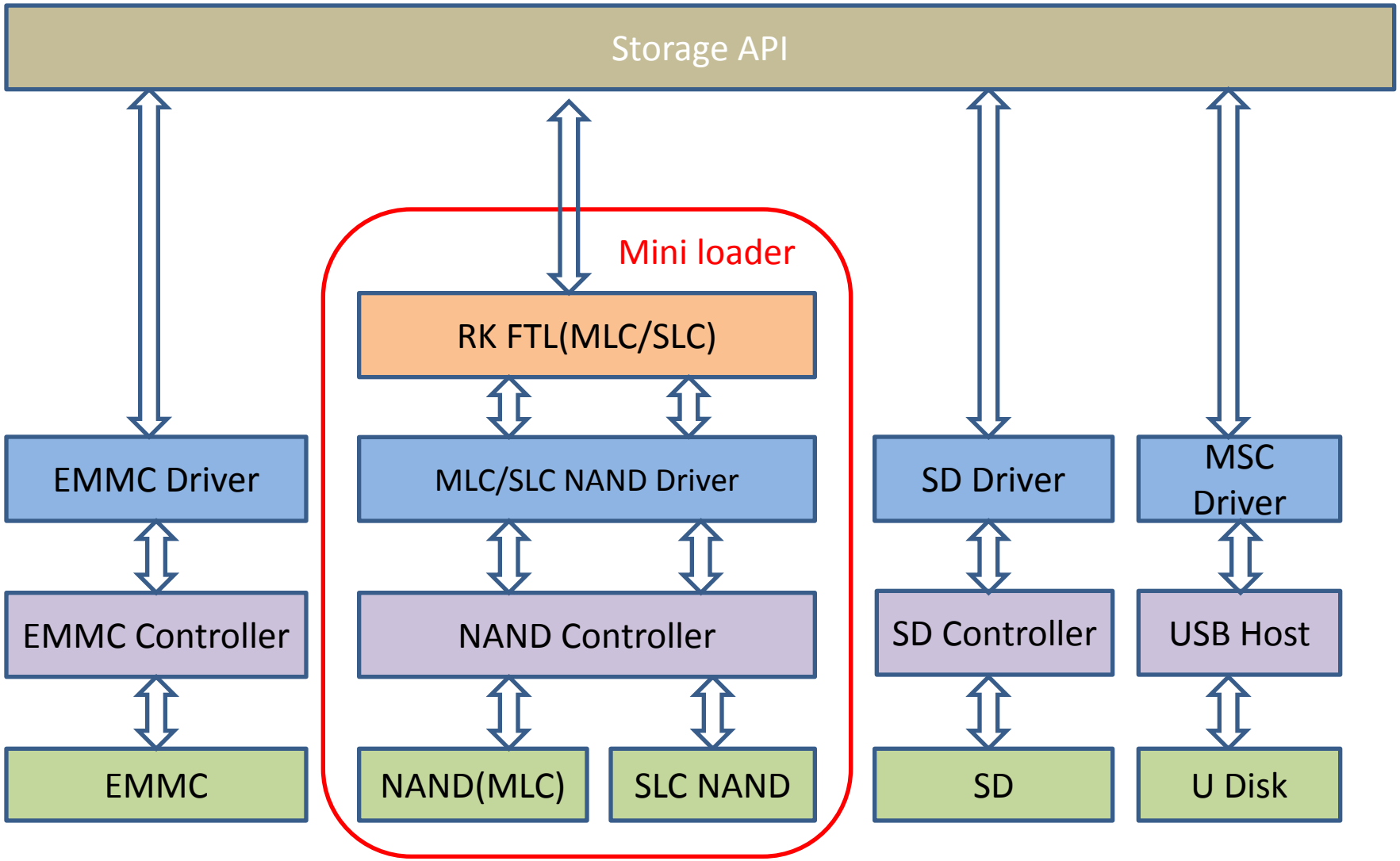


Android system:(miniloader + uboot + nand)



Uboot: Storage Architecture

Confidential



Uboot: storage driver overwiew

Confidential

u-boot > board > rockchip > common

名称	修改日期	类型	大小
emmc	2017/8/7 星期一 ...	文件夹	
mediaboot	2017/8/7 星期一 ...	文件夹	
nvme	2017/3/29 星期...	文件夹	
platform	2017/8/7 星期一 ...	文件夹	
rkboot	2017/8/7 星期一 ...	文件夹	
rkloader	2017/8/7 星期一 ...	文件夹	
sdhci	2017/8/3 星期四 ...	文件夹	
SecureBoot	2017/8/7 星期一 ...	文件夹	
storage	2017/8/7 星期一 ...	文件夹	
.built-in.o.cmd	2017/8/7 星期一 ...	Windows 命令脚本	2 KB
built-in.o	2017/8/7 星期一 ...	O 文件	708 KB
config.h	2017/3/29 星期...	H 文件	5 KB
Kconfig	2017/3/29 星期...	文件	1 KB
Makefile	2017/3/29 星期...	文件	2 KB

Emmc drivers

NVM API(NAND,EMMC...)

NVME drivers for SSD

Emmc drivers for RK3399

NVM storage API

Uboot: EMMC/SD timing config

Confidential

Z:\rk30\rk3288_android4.4\u-boot\board\rockchip\common\emmc\hw_SDConfig.h - N...

文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?

hw_SDConfig.h

```
#else
#define EN_EMMC_DDR_MODE        (0)
#endif /* CONFIG_RK_MMC_DDR_MODE */

#define EN_SD_PRINTF            (0)      /* 是否允许SD驱动内部调试信息打印，1:开启打印，0:关闭打印 */
#define DEBOUNCE_TIME           (25)     /* 卡拔插的消抖动时间,单位ms */

#define FOD_FREQ                (200)    /* 卡识别阶段使用的频率,单位KHz,协议规定最大400KHz */
/* 卡正常工作的最低频率为FREQ_HCLK_MAX / 8 */
#define SD_FPP_FREQ             (24000)  /* 标准SD卡正常工作频率，单位KHz，协议规定最大25MHz */
#define SDHC_FPP_FREQ           (40000)  /* SDHC卡在高速模式下的工作频率，单位KHz，协议规定最大50MHz
#define MMC_FPP_FREQ            (18000)  /* 标准MMC卡正常工作频率，单位KHz，协议规定最大20MHz */
#define MMCHS_26_FPP_FREQ       (25000)  /* 高速模式只支持最大26M的HS-MMC卡，在高速模式下的工作频率，

#if (EN_SD_DMA) || (EN_SDC_INTERAL_DMA)
#define MMCHS_52_FPP_FREQ       (50000)  /* 高速模式能支持最大52M的HS-MMC卡，在高速模式下的工作频率，
#else
#define MMCHS_52_FPP_FREQ       (40000)  /* 高速模式能支持最大52M的HS-MMC卡，在高速模式下的工作频率，
#endif

#if (!EN_SD_PRINTF)
#define SDOAM_Printf(...)
#else
```

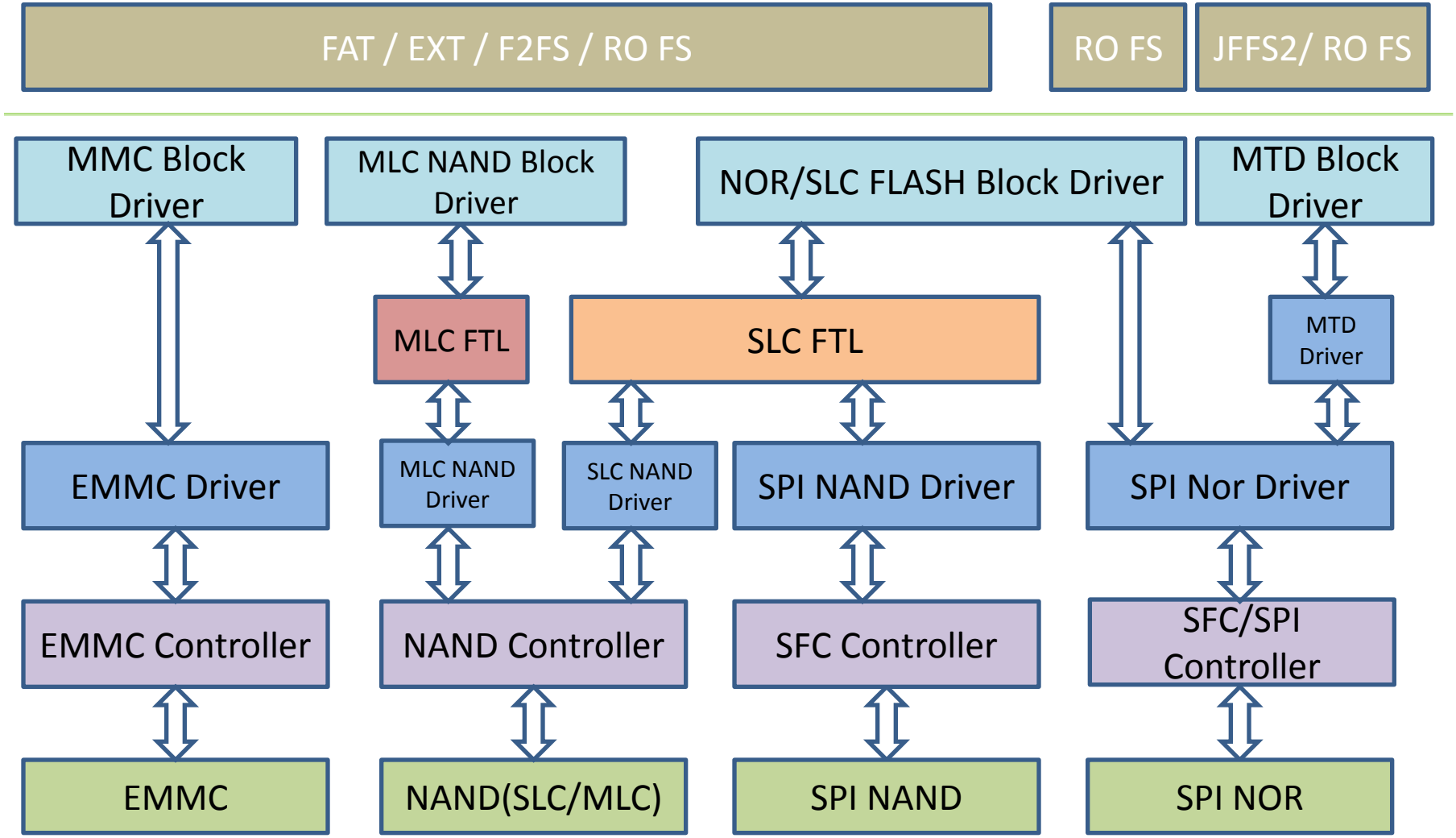
Config SD CARD max working clock

Config EMMC max working clock

Kernel

Kernel: Storage Architecture

Confidential



Kernel: Storage config: EMMC

Confidential

If the storage is EMMC, should config by:

1. make rkxxxx_defconfig, arch/arm/configs/rvxxxx_defconfig

```
CONFIG_MMC=y
CONFIG_MMC_PARANOID_SD_INIT=y
CONFIG_MMC_BLOCK_MINORS=32
# CONFIG_MMC_BLOCK_BOUNCE is not set
CONFIG_MMC_DW=y
CONFIG_MMC_DW_IDMAC=y
CONFIG_MMC_DW_ROCKCHIP=y
CONFIG_MMC_DW_SKIP_CACHE_OP=y
```

2. or make menuconfig

-> Device Drivers

-> MMC/SD/SDIO card support (MMC [=y])

->

```
[*]   Enable paranoid SD card initialization (EXPERIMENTAL)
      *** MMC/SD/SDIO Card Drivers ***
<*>   MMC block device driver
(32)    Number of minors per block device
<*>   Synopsys DesignWare Memory Card Interface
[*]     Internal DMAC interface
-*-     Synopsys Designware MCI Support as platform device
<*>     Rockchip specific extensions for Synopsys DW Memory Card Interface
```

Kernel: Storage config: rkflash

Confidential

The rkflash driver can support 3-type flash devices: SLC_NAND, SPI_NOR and SPI_NAND. The driver automatically selects the correct drivers by identifying the NVM devices that are connected.

1.make rvxxxxx_defconfig, arch/arm/configs/rkxxxx_defconfig

```
CONFIG_RK_FLASH=y
CONFIG_RK_NANDC_NAND=y  # SLC NAND #
CONFIG_RK_SFC_NOR=y     # SPI NOR #
CONFIG_RK_SFC_NAND=y    # SPI NAND #
# CONFIG_RK_NAND is not set    # MLC NAND #
```

2. or make menuconfig

-> Device Drivers

-> Rockchip Flash Devices Support (RK_FLASH [=y])

->

```
--- Rockchip Flash Devices Support
      *** Rockchip Flash Devices ***
<*>   RK NANDC NAND Device Support
<*>   RK SFC NOR Device Support
<*>   RK SFC NAND Device Support
[ ]   RK SFC NOR mtd Interface Support (NEW)
```

Kernel: Storage config: rkflash

Confidential

-> System Type

->RK NAND Devices Support(nand ko = [n])

```
[*]   Console write by thread
< >   RK NAND Device Support
[ ]   FPGA Board
```

-> Device Drivers

-> RK NAND Devices Support(rk_nand = [n])

```
      Android  --->
<*> Rockchip Flash Devices Support  --->
< > RK NAND Device Support
```

Kernel: Storage config: SPI NOR(MTD)

Confidential

If you want to use MTD block driver, depending on the default configuration of "rkflash", and also need to config MTD by:

step 1: make menuconfig

-> Device Drivers

-> Rockchip Flash Devices Support (RK_FLASH [=y])

->

```
--- Rockchip Flash Devices Support
      *** Rockchip Flash Devices ***
< >   RK NANDC NAND Device Support (NEW)
<*>   RK SFC NOR Device Support
< >   RK SFC NAND Device Support (NEW)
[*]   RK SFC NOR mtd Interface Support
```

-> Memory Technology Device (MTD) support (MTD [=y])

->

```
<*>   Caching block device access to MTD devices
```

Note: NAND and SPI NAND could not support MTD.

Kernel: Storage config: SPI NOR(MTD)

Confidential

step 2: modify rootfs device path in dts, such as:

```
diff --git a/arch/arm/boot/dts/rv1108-evb-v11.dts b/arch/arm/boot/dts/rv1108-evb-v11.dts
index c9900ea..a83b621 100644
--- a/arch/arm/boot/dts/rv1108-evb-v11.dts
+++ b/arch/arm/boot/dts/rv1108-evb-v11.dts
@@ -81,7 +81,8 @@
        };

        chosen {
-               bootargs = "rockchip_jtag noinitrd root=/dev/rootfs rootfstype=squashfs";
+               bootargs = "rockchip_jtag noinitrd root=/dev/mtdblock3 rootfstype=squashfs";
        };
```

Kernel: Storage config: rk_nand

Confidential

The rk_nand driver can support SLC and MLC NAND flash devices. The driver automatically selects the correct drivers by identifying the NVM devices that are connected.

Kernel/drivers/rk_nand:

```
-rw-rw-r--   1 zyf zyf    148 Aug 22 14:17 Kconfig
-rw-rw-r--   1 zyf zyf    144 Aug 22 14:17 Makefile
-rw-rw-r--   1 zyf zyf   1488 Aug 22 14:33 rk_ftl_api.h
-rw-rw-r--   1 zyf zyf 466886 Aug 24 18:37 rk_ftl_arm_v7.s
-rw-rw-r--   1 zyf zyf 430353 Aug 22 14:29 rk_ftl_arm_v8.s
-rw-rw-r--   1 zyf zyf  11105 Aug 22 18:03 rk_nand_base.c
-rw-rw-r--   1 zyf zyf  17982 Aug 22 14:19 rk_nand_blk.c
-rw-rw-r--   1 zyf zyf   1263 Aug 22 14:17 rk_nand_blk.h
```

make menuconfig

-> Device Drivers

-> RK NAND Devices Support(rk_nand = [y])

```
    Android  --->
< > Rockchip Flash Devices Support  --->
<*> RK NAND Device Support
```

-> System Type

->RK NAND Devices Support(nand ko = [n])

```
[*]   Console write by thread
< >   RK NAND Device Support
[ ]   FPGA Board
```

Kernel: Storage config: nand ko

Confidential

The nand ko driver can support SLC and MLC NAND flash devices. The driver automatically selects the correct drivers by identifying the NVM devices that are connected.

The "nand ko" need ismod by init in rootfs.

make menuconfig

-> Device Drivers

-> RK NAND Devices Support(rk_nand = [n])

```
< > Rockchip Flash Devices Support  --->
< > RK NAND Device Support
```

-> System Type

->RK NAND Devices Support(nand ko = [y])

```
[*]   Console write by thread
<*>   RK NAND Device Support
[ ]   FPGA Board
```