U-Boot next-dev 和 rkdevelop 的差异化

发布版本：1.1

作者邮箱：chenjh@rock-chips.com

日期：2019.11

文件密级：公开资料

前言

概述

Rockchip 平台上的 U-Boot 目前存在两个分支版本：rkdevelop（v2014-10）和 next-dev（v2017-09）。本文仅对这两个分支的状况和差异化进行概要说明，意在让读者能了解二者之间的概况。

产品版本

芯片名称	内核版本
RK3036/RK312X/RK322X/RK3288/RK3328/RK322XH/RK3368/RK3399/PX 系列	3.10、4.4

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

产品版本

修订记录

日期	版本	作者	修改说明
2018-02-28	V1.0	陈健洪	初始版本
2019-11-12	V1.1	陈健洪	增加产品版本列表

Rockchip U-Boot 概况

Rockchip 平台的 U-Boot 目前存在两个版本，旧的版本是 rkdevelop 分支，新的版本是 next-dev 分支。

1. rkdevelop 概况

rkdevelop 是从 U-boot 官方的 v2014.10 的正式版本中切出来进行开发的版本。目前大部分的芯片（2018 年之前）只要有使用 U-Boot，基本都是使用这个版本。目前这个版本支持的芯片包括：RK3036/RK312X/RK322X/RK3288/RK3328/RK322XH/RK3368/RK3399/部分 PX 系列。

目前，rkdevelop 支持的功能主要有：

- 支持 Android 平台的固件启动；
- 支持 RockUSB 和 Google Fastboot 两种方式烧写；
- 支持 Secure boot 固件签名加密保护机制；
- 支持 LVDS、EDP、MIPI、HDMI、CVBS 等显示设备；
- 支持 SDCard、Emmc、Nand Flash、U 盘等存储设备；
- 支持开机 logo 显示、充电动画显示，低电管理、电源管理；
- 支持 I2C、SPI、PMIC、CHARGE、GUAGE、USB、GPIO、PWM、DMA、GMAC、EMMC、NAND 中断等驱动；

对于上述的功能和使用方式有兴趣的读者，可以具体请参考《Rockchip U-Boot 开发指南 V3.8-20170214》。

2. next-dev 概况

next-dev 是从 U-Boot 官方的 v2017.09 正式版本中切出来进行开发的版本。随着 upstream 的 U-Boot 功能越来越完善，以及我们实际产品上对 U-Boot 的需求更加多样，因此 U-Boot 的版本升级也是势在必行，因此我们进行了 next-dev 开发。next-dev 作为 rkdevelop 的升级版本，目前在该平台上已经支持的芯片包括：RV1108/RK3188/RK3036/RK3066/RK312X/RK322X/RK3288/RK3328/RK322XH/RK3326/RK3368/RK3399/部分 PX 系列。

简而言之，rkdevelop 上支持的芯片在 next-dev 都同样有支持。在 rkdevelop 上支持的软件功能，在 next-dev 上大部分也同样会支持。

3. next-dev 和 rkdevelop 的差异

3.1 基础版本差异

如上述介绍，rkdevelop 分支基于 v2014.10 版本进行开发，不再更新 upstream 的代码；next-dev 分支基于 v2017.09 的版本进行开发，后续还会持续更新 upstream 的代码。

3.2 代码风格

rkdevelop 的整体代码风格偏“定制化”会多一些，coding style 以及功能实现上很多都不够标准化，更多走的是 rockchip 自己的一套流程或者框架。究其原因，主要还是 v2014.10 版本本身的系统框架还不够完善、第一次开发 U-Boot 也存在经验不足、工程师的标准化开发意识不够强、后续的各种产品需求越来越多样。

经过 rkdevelop 的开发经验积累，我们对 next-dev 有了更好的全局规划和认识，因此 next-dev 分支的代码严格遵循 upstream 的规范进行开发，所有驱动实现都尽量走现有的通用框架流程，方便以后持续更新和扩展功能。

3.3 DM(Driver Model)框架

DM 框架是 U-Boot 里现在所有驱动的框架模型统称，它主要包括 uclass、driver、device。U-Boot 的 doc 目录里可以找到官方详细的说明文档：

> Uclass - a group of devices which operate in the same way.
>
> A uclass provides a way of accessing individual devices within the group, but always using the same interface. For example a GPIO uclass provides operations for get/set value. An I2C uclass may have 10 I2C ports, 4 with one driver, and 6 with another.
>
> Driver - some code which talks to a peripheral and presents a higher-level interface to it.
>
> Device - an instance of a driver, tied to a particular port or peripheral.

其实 DM 模型和内核的 device-driver 框架模型是类似的，它为各类驱动模块指定了一套统一的标准框架。工程师只需要把硬件相关的部分嵌套进这个框架里即可。目前 U-Boot 支持的框架类型已经比较完整，next-dev 分支的驱动基本都是遵循下述已有的框架流程进行实现。例如：

```
./drivers/block/blk-uclass.c
./drivers/power/pmic/pmic-uclass.c
./drivers/power/regulator/regulator-uclass.c
./drivers/power/domain/power-domain-uclass.c
./drivers/thermal/thermal-uclass.c
./drivers/pinctrl/pinctrl-uclass.c
./drivers/gpio/gpio-uclass.c
./drivers/core/syscon-uclass.c
./drivers/core/uclass.c
./drivers/rtc/rtc-uclass.c
./drivers/reset/reset-uclass.c
./drivers/cpu/cpu-uclass.c
./drivers/clk/clk-uclass.c
......
```

3.4 代码开源

目前各芯片平台的基础代码都已经合并到官方的分支里，可以满足客户自己下载源代码进行编译下载、开源的需求。我们的工程师还会继续开展各芯片平台的 upstream 工作。

3.5 Pre-loader 支持

```
Maskrom -> Pre-loader -> Trust -> U-Boot -> kernel -> Android
```

上述是使用 rkdevelop 分支时整个系统的启动流程：Pre-loader 负责加载 Trust 和 U-Boot，然后 U-Boot 负责加载 kernel。我们可以把上述的 Pre-loader 称为为一级 loader，U-Boot 称为二级 loader。在 Rockchip 平台上 Pre-loader 是不开源的，仅提供 bin 文件对外使用。

如果客户的产品有代码全开源的需求怎么办？next-dev 分支的 U-Boot 支持 SPL/TPL 作为 Pre-loader 引导系统的功能，它可以负载加载 Trust 和 U-Boot。SPL/TPL 本身就是 U-Boot 自身提供的一个功能，只是在 rkdevelop 没有被使用。

其中 TPL 主要功能是 DDR 初始化, SPL 主要功能是加载和引导 trust/U-Boot 两个模块.rkdevelop 仅支持 Rockchip miniloader 作为 pre-loader；

next-dev 上各芯片平台都支持两种启动方式：SPL/TPL 和 Rockchip miniloader。

3.6. 分区支持

1. rkdevelop 仅支持 rk 格式 parameter.txt 分区, 不支持其他分区；
   rkdevelop 同时解析 parameter.txt 的 CMDLINE 信息
2. next-dev 支持 GPT 分区和 rk parameter 分区；
   为了保持 GPT 与 rkparameter 一致性, next-dev 推荐使用 kernel dts 的 bootargs 来自定义 cmdline, 不使用 parameter 的 CMDLINE.

3.7. 支持的固件类型

从固件打包格式上看：

1. next-dev 支持 RK 独立分区的固件启动方式，我们称为 boot rockchip；
2. next-dev 支持 AOSP 格式的启动方式，我们称之为 boot android；
3. next-dev 支持 FIT 格式打包的固件，rkdevelop 不支持；

rkdevelop 支持上述 1. 2. 两种命令启动固件，但是两种功能的代码被杂糅在一起，耦合性太大。next-dev 上进行了二者的剥离，它们是两个独立的启动命令。

从固件 boot 途径看：

1. next-dev 支持 tftpboot 和 pxe boot（基于 net）；
2. next-dev 支持 linux 的 Distro Boot（Linux Distribution 的 EFI boot）；
3. next-dev 支持 AOSP 固件(Android AVB 校验和 A/B 分区)；

3.8 文件系统

1. next-dev 文件系统支持，rkdevelop 不支持文件系统；
2. next-dev 支持 fat、ext2/4 文件系统；

3.9 rkbin 仓库

rkdevelop 的 U-Boot 工程里存放了许多 bin 文件和脚本工具。存放路径如下：

```
tools/rk_tools/bin/
tools/rk_tools/RKBOOT/
tools/rk_tools/RKTRUST/
tools/resource_tool/
tools/boot_merger.c
tools/trust_merger.c
......
```

在 next-dev 分支上，U-Boot 工程里不再存放这些 bin 文件和脚本工具，这些统一放到“rkbin”仓库里，因此用户需要再单独下载一个 rkbin 仓库配合 next-dev 使用。编译 uboot.img 的时候编译脚本会从 rkbin 仓库里索引对应平台的 bin 文件和工具，然后编译打包它们。最终和 rkdevelop 分支一样，也还是会在 U-Boot 工程下生成相关的 uboot.img、trust.img、loader 等固件。

3.10 Secure Boot

rkdevelop 分支支持 rk secure boot, next-dev 不支持；
next-dev 分支支持 AVB(Android), FIT 签名校验, rkdevelop 不支持；

3.11 Rockusb 和烧写

rkdevelop 支持所有的 Rockusb 命令, next-dev 支持的 WL 命令和所需的信息交互命令,
next-dev 不支持 ul, write pba, write vendor storage 等命令；
之前的固件烧写主要由 usbplug+miniloader 组成, usbplug 负责用 ul 命令烧写 miniloader,然后重启(或直接进入这个步骤)进入 miniloader, 在此环境完成剩余固件烧写；
新的烧写过程可以一次性在 usbplug 阶段完成, 省去重启步骤, 包括写 vendor storage 在内, 都直接在 usbplug 完成, 不需要 miniloader 辅助.

3.12 存储介质烧写地址

旧的方式:
NAND: IDB 存放在在特别的 boot 分区,系统阶段不可见, 后续的数据按 parameter 定义的地址按实际物理地址存放；
EMMC: 由于没有使用 boot 分区, 而是使用 normal 分区,所以在软件上(driver 或 partition 层)保留了前面 4MB(0x2000 block)作为 idb loader 数据；后续的数据按 parameter 定义的地址加上 4MB 后写到 eMMC;
新的方式:
EMMC 和 NAND 均按实际的物理可用空间提供 block 接口, 烧写过程 rockusb 可直接通过 WL 命令烧写所有空间.