U-Boot next-dev和rkdevelop的差异化

发布版本：1.0

作者邮箱：chenjh@rock-chips.com

日期：2018.02

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。

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

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.