RK3399 boot sequence

This article describes the boot sequence of the RK3399, the SoC on which the ROCKPro64 single board computer, Pinebook Pro laptop and the PinePhone Pro are based.

Contents

Boot ROM operation

After a hardware reset, the SoC's boot ROM (BROM) starts running on CPU0, one of the Cortex-A53s on the chip. This code is stored on a block of read-only memory, likely hard-wired into the chip.

It employs 5 strategies, in order, to load a bootloader from off-chip:

1. loading from NOR flash on SPI1
2. loading from NAND flash on SPI1
3. loading from eMMC
4. loading from SD on the SDMMC controller
5. bringing up the OTG0 USB controller in device mode and accepting control transfers to load programs

Loading a bootloader from storage is done in 4 steps:

1. Probing the boot device
2. Finding the ID block
3. Loading the first stage of the bootloader to main SRAM and running it
4. (if the first stage returned to BROM) Loading the second stage to low DRAM and running it

ID block

The ID block is a 512-byte block at one of several well-known locations (depending on boot device type) scrambled using RC4 with a well-known key, which begins with the bytes 0x3b 8c dc fc (ciphertext; plaintext: 0x55 aa f0 0f).

In addition to the magic number, it contains (all fields little-endian):

- a flag to RC4-scramble the bootloader images too (32b at offset 8)
- offset (in 512 B-sectors) of the first bootloader stage (16b at offset 12)
- the size (in 2 KiB blocks) of the first bootloader stage (16b at offset 506)
- the size (in 2 KiB blocks) of the second bootloader stage (16b at offset 508)

These fields can be seen in U-Boot **tools/rkcommon.c**.

SPI boot

The boot ROM tries to probe the device on SPI1 by issuing the SPI-NOR 0x9f (Read Identification) command.

If the type and capacity bytes are all-ones or all-zeroes, probe fails. If the first byte (manufacturer ID) is all-ones, the BROM uses NAND protocol, otherwise NOR protocol.

NOR protocol uses command 0x03 (low-frequency read without turnaround cycles) to read blocks, NAND protocol uses command 0x13.

The ID block is searched by loading the 32 bytes at the start of flash (TODO other load offsets) and testing for the magic number at any offset within those 32 bytes.

The SPI code has a bug that means that the 2 KiB blocks in which the bootloader is loaded have a stride of 4KiB, leaving the 2KiB inbetween as unused padding.

eMMC/SD boot

The boot ROM probes for eMMC on the eMMC controller using the normal MMC enumeration sequence (CMD0/1/2/3) at 375 kHz and runs the bus at 24 MHz, in 8-wide mode. The BROM does not support using the standard option for eMMC boot partitions.

SD is probed on the SDMMC (not the functionally equivalent SDIO) controller, using the normal SD enumeration sequence at 375 kHz and then runs the card at 12 MHz (TODO confirm) in 4-wide mode.

After initialization, eMMC and SD behave much the same. The ID block is searched by loading the blocks at sector 64 + 1024*n for n from 0 to 4 (inclusive) and checking for the magic number at the start of that sector. The BROM handles both byte- and block-addressed SD/MMC cards.

Hardware root of trust

The BROM theoretically supports SHA256/RSA2048 authentication with rollback protection.

The relevant data is stored in Secure eFuses: an enable flag, a SHA256 hash of the public key (which is stored on the boot medium) and a range of bits that are sequentially set for rollback protection.

In practice, the BROM does not verify the full length of the public key against the hash in eFuses, which means that it is most likely broken (expert review would be appreciated).

TOOD details

- FOSDEM 2023: Tomasz Żyjewski (3mdeb) - Overview of Secure Boot state in the ARM-based SoCs, 2nd edition (https://fosdem.org/2023/schedule/event/arm_secure_boot_2/)
- Rockchip Developer Guide Linux4.4 Secure Boot EN.pdf (https://github.com/96boards-tb-96aiot/docs/blob/4224417a08256cc97fb1e28c1ae1d35f8d55e89b/Linux%20reference%20documents/Secure/Rockchip_Developer_Guide_Linux4.4_SecureBoot_EN.pdf) (zh_CN (https://github.com/96boards-tb-96aiot/docs/blob/4224417a08256cc97fb1e28c1ae1d35f8d55e89b/Linux%20reference%20documents/Secure/Rockchip_Developer_Guide_Linux4.4_SecureBoot_CN.pdf))
- Artur Kowalski (3mdeb) - Enabling Secure Boot on RockChip SoCs - December 2021 (https://blog.3mdeb.com/2021/2021-12-03-rockchip-secure-boot/)
- RK3399 Efuse Operation Instructions V1.00 - 14 February 2019 (https://usermanual.wiki/Document/RK3399EfuseOperationInstructionsV10020190214.1448491699/view)

USB gadget boot

If no ID block is found on SPI, eMMC or SD, the BROM sets up the OTG0 USB controller to act as a USB high-speed (480 Mb/s) device with Vendor/Product ID of 2207:330c. In this state it supports most standard control requests, as well as 2 vendor control requests:

0x0471
The code is loaded to main SRAM. When the transfer is done, it does a function call to the load address.

0x0472
The code is loaded to low DRAM. When the transfer is done, USB is torn down and BROM jumps to the load address.

Transfer is performed by doing the requests on 4 KiB blocks. BROM considers the transfer complete as soon as a request supplies a block not 4096B long. (This means images a multiple of 4096 bytes long must be padded to achieve this condition.

U-Boot boot sequence

Bootloaders based on U-Boot (including Tow-Boot) run in 4 stages on the RK3399:

1. TPL, loaded by the BROM into main SRAM. Its job (https://www.denx.de/wiki/pub/U-Boot/MiniSummitELCE2013/tpl-presentation.pdf) is to initialize DRAM (main system memory). It returns to BROM. This job can alternatively be performed by proprietary Rockchip DDR blobs.
2. SPL, loaded by the BROM into low DRAM. It loads the respective parts of TF-A BL31 into DRAM, main and PMU SRAM, and U-Boot proper into DRAM.
3. TF-A BL31. It sets up EL2 to run U-Boot and stays resident until system shutdown.
4. U-Boot proper. It can load EFI binaries (Grub, systemd-boot, …) from a variety of block devices (SD, eMMC, NVMe, USB Mass Storage, …).

For mainline-based U-Boots, these stages usually come in 2 images:

idbloader.img
contains TPL and SPL.

u-boot.itb

contains TF-A and U-Boot.

Load offsets

This section applies to BSP U-Boot. Mainline-based U-Boots pack TF-A into **u-boot.itb**.

There are 3 sections for the boot loader. They are in order, without gap, though their is no need to use all the space in each section.
Here are the details:

Start in sectors	Size in sectors	Name	Description
64	16320	IDBLoader	SoC initialization code
16384	8192	OS loader	Generally U-Boot
24576	8192	TrustedFirmware-A	

General maintenance

If a new U-Boot is supplied, it is generally installed similar to this:

```
# dd if=/boot/idbloader.img conv=notrunc seek=64    of=/dev/mmcblkX
# dd if=/boot/u-boot.itb    conv=notrunc seek=16384 of=/dev/mmcblkX
```

Different devices

The RK3399 boots to multiple devices. Boot device selection is done in the following order, and it cannot be changed.
If a device is blank / unused, the SoC code moves on to the next device in the list.

- SPI
- eMMC
- SD card

However, whence the user boot code runs, it can then give priority to other devices, if available. The following devices are not directly bootable:

- NVMe
- USB 3
- WiFi

They can be made bootable by using one of the other devices as an initial bootloader. For example, several people have gotten their NVMe drives to be bootable with "/boot" and "/" on the NMVe. This either entails using the SPI or eMMC as the initial bootloader, with code to support PCIe NVMe devices.

Grub as the target of the bootloader

It is possible to use Grub as the target of U-Boot. This would allow;

- Selecting a different boot device

- Choosing a partition on a boot device for booting
- Different kernels
- Changes in kernel command line options

However, at present, Grub does not support the video & keyboard of the Pinebook Pro. So, any selection is done through the serial console.

Boot loader development

There are several projects that have their own versions of U-Boot, with different features. Here are some of the more common ones at present, 2020/06/14:

- Rockchip
- The original default Debian
- Manjaro
- U-Boot mainline

Bootloaders not based on U-Boot:

- coreboot runs on RK3399-based Chromebooks, it has not been ported to Pine64 boards yet.
- levinboot (https://gitlab.com/DeltaGem/levinboot) is a bootloader developed by CrystalGamma in the Pine64 community. It runs on RockPro64 and Pinebook Pro. Its development is on hiatus as of April 2022, but a fork porting it to the PinePhone Pro exists.

Retrieved from "https://wiki.pine64.org/index.php?title=RK3399_boot_sequence&oldid=21066"