Linaro Connect
MADRID 2024 | MAY 14-17 2024

Qualcomm & SystemReady IR

An overview of Qualcomm support in U-Boot

Caleb Connolly <caleb.connolly@linaro.org>

About me

- Caleb (they/them)
- Working on FOSS since ~2018
- Kernel engineer @ Linaro (Qualcomm Ecosystem Team)
  - Robotics board enablement
  - U-Boot Qualcomm maintainer
- postmarketOS core team member
  - Enabling upstream kernel support for the last decades worth of phones
  - Maintain Snapdragon 845 devices
- Maintainer of Qualcomm platform support in U-Boot
- @cas@social.treehouse.systems

SystemReady (the IoT Ready kind)!

- Not the most exciting topic in the world
- But that’s kinda the point :P
- SR/IR certification means the devices bootloader behaves how it should
  - EFI apps can run, and work as expected
  - No special requirements to get a distro booting (beyond normal hardware enablement)
  - A comprehensible devicetree
  - And you don’t have to go poking around the bootloader to Get Stuff Done

Qualcomm Quirks

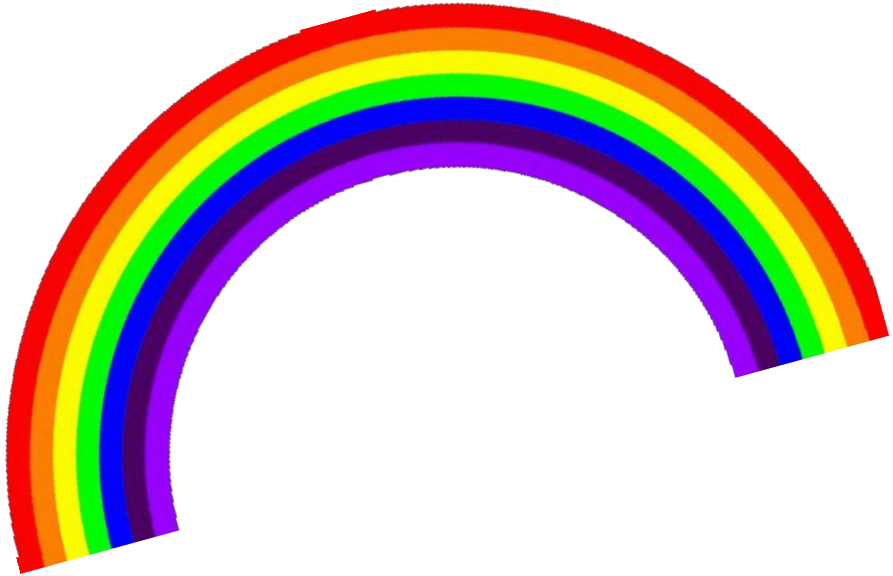
- IoT reference boards ship with Qualcomm's Android Bootloader by default
- Image.gz+dtb are stuffed into an Android boot image and flashed to the "boot" partition
  - Makes for a fun kernel upgrade experience
- Bootloader looks for downstream specific DT nodes and properties (nothing too egregious though)
  - Custom DTB overlay mechanism
- At least the bootloader app is open source [1]
- Needs adjustments to behave nicely, but has to be signed (with a random key on dev boards)
- … exact format of certs is not public, qcom signing tools under a proprietary licence

[1] https://git.codelinaro.org/linaro/qcomlt/abl/-/tree/release/LE.UM.5.4.1.r1-25200-QRB2210.0


That wasn't very
Standards compliant
of you X3

On the bright side

- Situation is expected to get much better with newer platforms
  - EFI runtime available ootb \o/
- In the meantime, all hope is not lost
- U-Boot to the rescue!



Welcome to GRUB!

Why now?

- Qualcomm placing more value in upstream for IoT
- Want to be more in-line with the rest of the industry (yay!)

=> EFI booting!

- Reworking custom proprietary EDK2 bootloader for generic EFI across existing platforms not feasible
- U-Boot work begins

Enter U-Boot

- Single threaded, polling only (no interrupts)
- Devicetree (the upstream kind)
- Driver model very reminiscent of Linux (with bonus compat headers)
- Lazy probing
- EFI framework and runtime!
- And a whole lot more

Caleb’s Cool list of Cool U-Boot features

OF_UPSTREAM for DT compatibility

- No more vendoring DT from Linux!
- Synced regularly from devicetree-rebasing [1]
- Ongoing community discussion about where device tree should live

[1] https://git.kernel.org/pub/scm/linux/kernel/git/devicetree/devicetree-rebasing.git/

OF_LIVE

- Parse the FDT just once and then have heap allocated data structures!
- Like an order of magnitude faster
- Super easy and fast to hot-patch (we use this to manage DT compatibility issues)

```
/* Remove all references to the power domain controller */
for_each_of_allnodes(prev: np) {
        if (!(prop = of_find_property(np, name: "power-domains", lenp: NULL)))
                continue;

        val = prop->value;
        if (val[0] == cpu_to_fdt32(pd->phandle))
                of_remove_property(np, prop);
}
```

Resource Power Manager? No thanks!

LMB allocator

- Aka memblock
- Buffer allocation made easy!
- Replaces a lot of hardcoded MESS in favour of dynamic runtime allocation
- Find room for your compressed kernel, uncompressed kernel, AND initramfs!
  - All without touching that oh-so-special trustzone memory

```
env_set_hex(varname: "kernel_addr_r", value: addr_alloc(lmb: &lmb, size: SZ_128M));
env_set_hex(varname: "ramdisk_addr_r", value: addr_alloc(lmb: &lmb, size: SZ_128M));
env_set_hex(varname: "kernel_comp_addr_r", value: addr_alloc(lmb: &lmb, size: KERNEL_COMP_SIZE));
env_set_hex(varname: "kernel_comp_size", value: KERNEL_COMP_SIZE);
env_set_hex(varname: "scriptaddr", value: addr_alloc(lmb: &lmb, size: SZ_4M));
env_set_hex(varname: "pxefile_addr_r", value: addr_alloc(lmb: &lmb, size: SZ_4M));
env_set_hex(varname: "fdt_addr_r", value: addr_alloc(lmb: &lmb, size: SZ_2M));
```

Qualcomm support 👌

- Needed some cleanup
- But in pretty great shape now
- Almost everything is handled generically at runtime
  - The silver lining of chainloading :D
  - Almost all Qualcomm devices are supported with a single build target
  - Just drop in a DT and you're good to go
- Bringup for new platforms is relatively straightforward
  - Bringup of vendor devices is a breeze
- Community adoption
  - EFI for the Android masses!
  - https://github.com/msm8916-mainline/gen-uboot-img

Upstream status report

- MSM8916, QCM2290, SM6115, SDM845, SM8250, SM8550, SM8650
  - Boot to U-Boot shell
- MSM8916, QCM2290, SM6115, SDM845, SM8250
  - Boot from USB
    - RB5 can’t turn on VBUS due to missing RPMh regulators
- MSM8916, SDM845
  - SDcard/MMC support

- Primary pain points are:
  - Missing stub drivers for rpmhcc (MMC drivers on QCM2290 & SM6115 fail due to this)
  - Missing UFS support
  - Missing rpmh regulators
  - Missing capsule update support for Qualcomm
- All are implemented, but not yet upstreamed.

What of SystemReady/IR?

- The RB2 platform is SR/IR compliant and in the process of being certified
  - Fedora rawhide workstation images can boot from USB!
  - Handful of U-Boot patches still necessary
- The RB1, 3, and 5 boards are all more or less on par
  - Some missing patches in U-Boot
  - Distro kernel enablement still TODO

The Future 

- Wider community adoption
  - Lots of folks running mainline Linux on Qualcomm Android devices
  - U-Boot solves a lot of problems there
  - Simplifies bringup and support for distros
- Provide community bootloader support via LVFS
  - If only the DTB needs to be different then scaling up is fairly easy
  - Ongoing discussions for avoiding the need to hardcode GUID image identifiers
    - https://lore.kernel.org/u-boot/20240426-b4-dynamic-uuid-v1-0-e8154e00ec44@linaro.org/
- Support more and more Qualcomm platforms, improve docs, and more!
  - https://docs.u-boot.org/en/latest/board/qualcomm/index.html

Slides: calebs.dev/u-boot24.pdf

Linaro Connect
MADRID 2024 | MAY 12-17 2024

Thank you

Caleb Connolly <caleb.connolly@linaro.org>