linaro Connect 2025

# Generic BootLoader Library (GBL) on devboards

Amit Pundir

## About Me

- Senior Engineer at Linaro
- 10+ years of AOSP (Android Open Source Project) bringup and maintenance
- pundir on #linaro-android, #aosp-developers IRC channels at OFTC.net

## Agenda

- Android Bootloaders
- Generic Bootloader Library (GBL)
- Enabling GBL on Linaro supported devboards

## Android Bootloaders

<rant>

On almost every release, Android update the bootloader requirements

</rant>

# Android Bootloaders Fragmentation

- Unlike Linux kernel, no consensus on a generic bootloader among vendors

- Each SoC vendor has its own bootloader implementation
  - Implementing the new Android requirements every release
  - Worst case: Different bootloader versions for the same SoC (Android 15 vs Android 16)

- Little to no value in “differentiation” of Android bootloader implementation and functionality
  - No one buys a phone/tablet based on the bootloader

## Generic Bootloader Library (GBL)

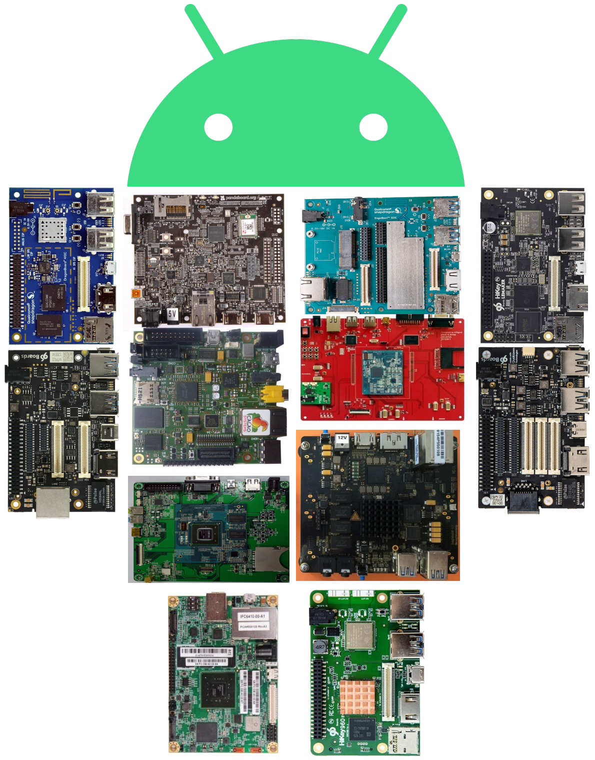
- GBL is a standardized Android bootloader interface to fix Android bootloader fragmentation problem
    - It provides a hardware-agnostic interface that abstracts the bootloader's platform-specific implementation

- It is an Android boot flow UEFI application provided by Google

- Vendor independent GBL updates
    - Reduces vendor's integration burden, resulting in faster uptake of Android Boot changes
    - Guaranteed using trusted components across ecosystem (ATF, libavb, libfdt, libufdt)

- Provide production ready open source Androidboot flow reference implementation

## Enabling GBL on Linaro supported devboards



## State of stock bootloaders on devboards

- Unlike high volume commercial devices, vendor bootloaders on devboards do not get any feature updates. Making Android updates nearly impossible.

- Luckily on Qualcomm devboards, the reference bootloader code is available on codelinaro
  - In the past cherry-picking relevant features from newer bootloader releases helped
  - But the process is tedious and error prone

- On other vendor devboards, a reasonable way out is to switch to other open-source bootloader alternatives if possible like U-Boot, coreboot etc

## GBL on Linaro supported devboards, why?

- Vendor independent Android bootflow implementation
  - Install new Android bootloader features as part of GBL updates/images
  - Leading to faster Android version upgrades

- Replicates production bootloader behavior
  - Helps in testing and validating bootloader-level features

## GBL on Linaro supported devboards, how?

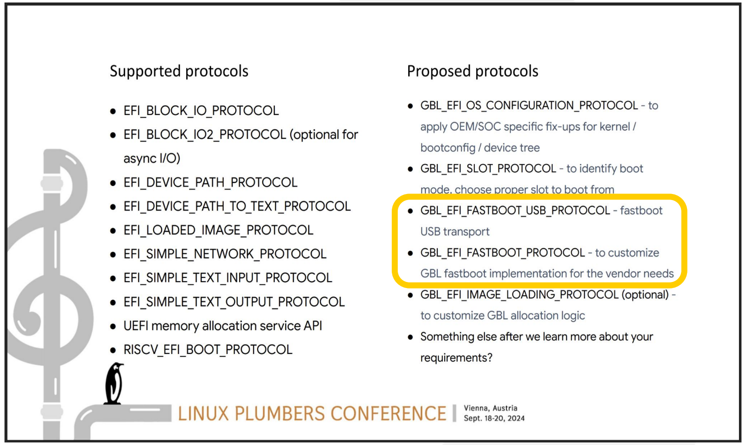
- GBL requirements
  - UEFI support in vendor bootloader
    - To reuse existing standardized interfaces such as block devices, network, etc

- U-Boot to the rescue
  - Already using U-Boot to boot AOSP from external storage (mmc-sdcard)
    - To avoid the wear and tear of internal storage due to prolonged usage in the lab
  - GBL is developed and tested with U-Boot internally
  - In house U-Boot expertise

# GBL and Custom EFI Protocols

- U-Boot integration with custom GBL protocols

- For example: GBL enables fastboot-over-USB over custom GBL_EFI_FASTBOOT_USB_PROTOCOL
  - GBL also support fastboot-over-TCP over the standard UEFI Simple Network Protocol



Supported protocols

- EFI_BLOCK_IO_PROTOCOL
- EFI_BLOCK_IO2_PROTOCOL (optional for async I/O)
- EFI_DEVICE_PATH_PROTOCOL
- EFI_DEVICE_PATH_TO_TEXT_PROTOCOL
- EFI_LOADED_IMAGE_PROTOCOL
- EFI_SIMPLE_NETWORK_PROTOCOL
- EFI_SIMPLE_TEXT_INPUT_PROTOCOL
- EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL
- UEFI memory allocation service API
- RISCV_EFI_BOOT_PROTOCOL

Proposed protocols

- GBL_EFI_OS_CONFIGURATION_PROTOCOL - to apply OEM/SOC specific fix-ups for kernel / bootconfig / device tree
- GBL_EFI_SLOT_PROTOCOL - to identify boot mode, choose proper slot to boot from
- GBL_EFI_FASTBOOT_USB_PROTOCOL - fastboot USB transport
- GBL_EFI_FASTBOOT_PROTOCOL - to customize GBL fastboot implementation for the vendor needs
- GBL_EFI_IMAGE_LOADING_PROTOCOL (optional) - to customize GBL allocation logic
- Something else after we learn more about your requirements?

LINUX PLUMBERS CONFERENCE | Vienna, Austria Sept. 18-20, 2024

## Build U-Boot and GBL

- Build U-Boot for the devboard with EFI loader support

- Build GBL

```
$ repo init -u https://android.googlesource.com/kernel/manifest -b uefi-gbl-mainline
$ repo sync -j`nproc`
$ ./tools/bazel run //bootable/libbootloader:gbl_efi_dist \
  --extra_toolchains=@gbl//toolchain:all --sandbox_debug --verbose_failures
$ dd if=/dev/zero of=gbl.img bs=1048576 count=2
$ mkfs.vfat esp.img
$ mcopy -i esp.img out/gbl_efi/gbl_aarch64.efi ::gbl_aarch64.efi
```

# Build AOSP

- Enable init_boot image support
  - GBL only supported boot header v4 images initially
  - Set BOARD_INIT_BOOT_HEADER_VERSION := 4 and BOARD_INIT_BOOT_IMAGE_PARTITION_SIZE in BoardConfig

- Prepare dummy vbmeta image to disable verity

```
$ external/avb/avbtool.py make_vbmeta_image --flag 2 --padding_size 4096 --output ./vbmeta_disabled.img
```

## Flash Images

- Prepare custom partition layout (A/B) on mmc-sdcard using GPT command from U-Boot

- Flash GBL and AOSP images using fastboot command from U-Boot
  - Make sure U-Boot's CONFIG_FASTBOOT_FLASH_MMC_DEV is set to correct device

```
$ fastboot flash esp esp.img flash boot boot.img flash init_boot init_boot.img \
  flash vendor_boot vendor_boot.img flash super super.img flash userdata userdata.img format:ext4 metadata
$ fastboot --disable-verity --disable-verification flash vbmeta vbmeta_disabled.img
```

- Detailed instructions to build, flash and boot U-Boot, GBL and AOSP on devboards are hosted on DevboardsForAndroid docs.

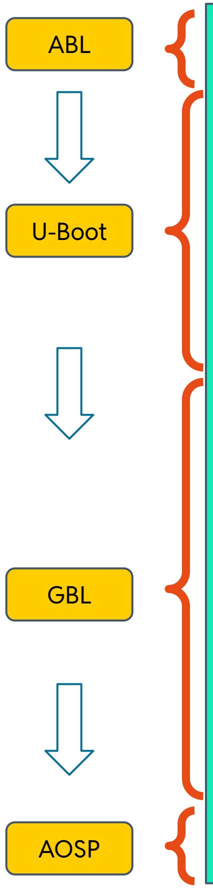
## Boot AOSP using GBL

- Launch GBL efi app from U-Boot and continue to boot AOSP, else interrupt the GBL bootflow by pressing <BACKSPACE> and jump to fastboot mode in GBL

```
=> load mmc 0:1 ${kernel_addr_r} gbl_aarch64.efi
=> bootefi ${kernel_addr_r}l
1664000 bytes read in 75 ms (21.2 MiB/s)
Cannot persist EFI variables without system partition
fw_name is not defined. Not generating capsule GUIDs
Booting /gbl_aarch64.efi
****Generic Bootloader Application****
Image path: /VenHw(e61d73b9-a384-4acc-aeab-82e828f3628b,0000000000000000)/VenHw(e61d73b9-a384-4acc-aeab-82e828f3628b,6d00000000000000)/
SD(0)/SD(0)/HD(1,GPT,bc0330eb-3410-4951-a617-03898dbe3377,0x100,0x1000)
Block #0 GPT sync result: Found valid GPT.
Proceeding as Android
Block #0 GPT sync result: Found valid GPT.
Press Backspace to enter fastboot
Backspace pressed, entering fastboot
Entering fastboot mode...
Started Fastboot over USB.
Failed to start EFI network. NotFound.
```

**Detailed instructions to build, flash and boot U-Boot, GBL and AOSP on DB845c, RB5 and SM8550-HDK devboards are hosted on [DevboardsForAndroid]() docs.*

# GBL Bootflow Overview on Devboards



```
Shutting Down UEFI Boot Services: 8557 ms
BDS: LogFs sync skipped, Unsupported
App Log Flush : 72 ms
Exit BS      [ 8783] UEFI End


U-Boot 2025.04-rc5-00761-gcf757313b8c9 (Apr 09 2025 - 16:29:53 +0530)

Model: Thundercomm Dragonboard 845c
DRAM:  4 GiB
Core:  221 devices, 31 uclasses, devicetree: board
MMC:   mmc@8804000: 0
Loading Environment from nowhere... OK
In:    serial,button-kbd
Out:   serial,vidconsole
Err:   serial,vidconsole
QCOM: U-Boot loaded from ABL
QCOM-FMP: Failed to find boot partition
Net:   No ethernet found.
GENI SE wrapper not found

scanning bus for devices...
starting USB...
Bus usb@a800000: Register 2000340 NbrPorts 2
Starting the controller
USB XHCI 1.10
scanning bus usb@a800000 for devices... 1 USB Device(s) found
       scanning usb for storage devices... 0 Storage Device(s) found
Hit any key to stop autoboot:  0
1664000 bytes read in 75 ms (21.2 MiB/s)
Cannot persist EFI variables without system partition
fw_name is not defined. Not generating capsule GUIDs
Booting /gbl_aarch64.efi
****Generic Bootloader Application****
Image path: /VenHw(e61d73b9-a384-4acc-aeab-82e828f3628b,0000000000000000)/VenHw(e61d73b9-a384-4acc-aeab-82e828f3628b,6d00000000000000)/SD(0)/SD(0)/HD(1,GPT,bc0330eb-3410-4951-a617-03898dbe3377,0x100,0x1000)
Block #0 GPT sync result: Found valid GPT.
Proceeding as Android
Block #0 GPT sync result: Found valid GPT.
Press Backspace to enter fastboot
Try booting as Android
boot mode from BCB: AndroidBootMode::Normal
boot image size: 14345450
boot image cmdline: ""
boot ramdisk size: 0
boot dtb size: 0
vendor ramdisk size: 11769074
vendor cmdline: "androidboot.boot_devices=soc@0/8804000.mmc androidboot.hardware=db845c androidboot.verifiedbootstate=orange androidboot.slot_suffix=_a earlycon firmware_class.path=
/vendor/firmware/ init=/init printk.devkmsg=on deferred_probe_timeout=30 pcie_pme=nomsi qcom_geni_serial.con_enabled=1 console=ttyMSM0,115200n8 bootconfig"
vendor dtb size: 1208513
init_boot image size: 35885165
WARNING: UEFI GblEfiAvbProtocol.read_is_device_unlocked implementation is missing. This will not be permitted in the future.
WARNING: UEFI GblEfiAvbProtocol.read_rollback_index implementation is missing. This will not be permitted in the future.
AVB verification failed with Verification failure. Device is unlocked: true. Color: orange. Continue current boot attempt.
final bootconfig: "androidboot.verifiedbootstate=orange\nandroidboot.slot_suffix=_a\nandroidboot.force_normal_boot=1\nandroidboot.boot_devices=soc@0/8804000.mmc\nandroidboot.hardware=db845c\nandroidboot.verifiedbootstate=orange\nandroidboot.slot_suffix=_a\n"
kernel is gzip compressed
kernel decompressed size 35625472
Applying 0 overlays
Overlays applied
linux,initrd-start: 0x138200000
linux,initrd-end: 0x13af7265a
final cmdline: "root=/dev/sda2 androidboot.bootdevice=1d84000.ufshc androidboot.serialno=c2d1480f androidboot.baseband=msm msm_drm.dsi_display0=dsi_lt9611_1080_video_display: androidbo
ot.slot_suffix=_a skip_initramfs rootwait ro init=/init androidboot.dtb_idx=-1347440721 androidboot.boot_devices=soc@0/8804000.mmc androidboot.hardware=db845c androidboot.verifiedboots
tate=orange androidboot.slot_suffix=_a earlycon firmware_class.path=/vendor/firmware/ init=/init printk.devkmsg=on deferred_probe_timeout=30 pcie_pme=nomsi qcom_geni_serial.con_enabled
=1 console=ttyMSM0,115200n8 bootconfig"

Booting kernel @ 0x13b000000, ramdisk @ 0x138200000, fdt @ 0x13af72660

[    0.000000][    T0] Booting Linux on physical CPU 0x0000000000 [0x517f803c]
[    0.000000][    T0] Linux version 6.12.5-android16-0-gc82917ebd289-ab12815448-4k (kleaf@build-host) (Android (12755234, +pgo, +bolt, +lto, +mlgo, based on r536225) clang version 19.0.1
(https://android.googlesource.com/toolchain/llvm-project b3a530ec6537146650e42be89f1089e9a3588460), LLD 19.0.1) #1 SMP PREEMPT Tue Dec 17 23:09:52 UTC 2024
[    0.000000][    T0] KASLR enabled
[    0.000000][    T0] Machine model: Thundercomm Dragonboard 845c
```

## Summary

- GBL standardizes the Android bootflow process, which will hopefully result in much faster Android updates

- Enabling it on devboards bridges the gap between prototype and production
  - Enables testing of Android boot flows with automated testing and CI on devboards

## References

- LPC 2020: Android Bootloader Consolidation
- LPC 2024: Android Generic Boot Loader
- DevboardsForAndroid: Flashing and booting AOSP on DB845c using GBL

Questions?

linaro Connect 2025

# Thank You!