New Android requirements for bootloaders

Sam Protsenko
<semen.protsenko@linaro.org>
Android 10

- Android 10 was released at 03 September 2019
- There are new requirements for new devices (launching with Android 10)
  - Checked by VTS
- This presentation covers:
  - Reboot reason (BCB)
  - Android Boot Image v2
  - Dynamic partitions
  - DTBO partition
  - fastboot changes
  - AVB 2.0
  - A/B
- U-Boot is used as a reference: the only one community driven Embedded bootloader
Android 10: Boot Flow

Start

Get A/B slot (from ‘misc’)

Check reboot reason (in BCB):

Normal boot? true

Choose ‘boot_x’ part. to load

Recovery boot? true

Choose ‘recovery’ part. to load

 AVB verification Cmdline: + A/B + AVB

Load chosen part.

Load DTB from chosen part.

Load DTBO from dtbo_x

Merge DTBO in DTB

Boot kernel from chosen part.

“system” is logical part:
- use ramdisk
- no system-as-root

false

Clear BCB

Enter Fastboot

End

false

false

false
### 1. BCB

**Start**

- Get A/B slot (from ‘misc’)

**Check reboot reason (in BCB):**

  - Normal boot?
    - True: Choose ‘boot_x’ part. to load
    - False: Recovery boot?
      - True: Choose ‘recovery’ part. to load
      - False: Bootloader boot?

**AVB verification**

- Cmdline: + A/B + AVB
- Load chosen part.
- Load DTB from chosen part.

**Load DTBO from dtbo_x**

**Merge DTBO in DTB**

**Boot kernel from chosen part.**

- “system” is logical part.:
  - Use ramdisk
  - No system-as-root

**End**
BCB: Overview

- Not all devices have non-volatile memory to store reboot reason
- BCB = Boot Control Block
- Part of ‘misc’ partition (first 2 KiB); BCB is a raw (binary), no FS
- Interface between bootloader and AOSP (main system and recovery)
  - reboot reason
  - messages to recovery
  - RescueParty messages
- Testing:
  - `adb reboot bootloader`
  - `adb reboot recovery`
  - `adb reboot fastboot`
  - Factory Reset
BCB: Format

- BCB format:
  - Repository: https://android.googlesource.com/platform/bootable/recovery
  - File: bootloader_message/include/bootloader_message/bootloader_message.h

```c
// 2 KiB structure
struct bootloader_message {
    char command[32];
    char status[32];
    char recovery[768];
    char stage[32];
    char reserved[1184];
};
```

Reboot reason:
- “bootonce-bootloader”
- “boot-recovery”
- empty

Message to recovery
BCB: U-Boot Implementation

- ‘bcb’ command:
  - CONFIG_CMD_BCB
  - cmd/bcb.c
  - doc/android/bcb.txt

`bcb` - Load/set/clear/test/dump/store Android BCB fields

Usage:

- `bcb load <dev> <part>` - load BCB from mmc `<dev>:` `<part>`
- `bcb set <field> <val>` - set BCB `<field>` to `<val>`
- `bcb clear [<field>]` - clear BCB `<field>` or all fields
- `bcb test <field> <op> <val>` - test BCB `<field>` against `<val>`
- `bcb dump <field>` - dump BCB `<field>`
- `bcb store` - store BCB back to mmc
**BCB: U-Boot Usage**

```c
#include/environment/ti/boot.h:
if bcb load 1 misc; then
    if bcb test command = bootonce-bootloader; then
        bcb clear command; bcb store
        # Boot fastboot
    elif bcb test command = boot-recovery; then
        # Boot recovery (don’t clear BCB!)
    else
        # Normal boot
    fi
else
    echo Warning: BCB is corrupted or does not exist
    # Normal boot
fi
```
2. A/B

Start

- Get A/B slot (from 'misc')

Check reboot reason (in BCB):

- Normal boot? (true/false)
  - true: Choose 'boot_x' part. to load
  - false: Recovery boot? (true/false)
    - true: Choose 'recovery' part. to load
    - false: Bootloader boot? (true)
      - Clear BCB
      - Enter Fastboot

- AVB verification

- Cmdline: + A/B + AVB

- Load chosen part.

- Load DTB from chosen part.

- Load DTBO from dtbo_x

- Merge DTBO in DTB

- Boot kernel from chosen part.

“system” is logical part:
- use ramdisk
- no system-as-root

End
A/B: Overview

- A mechanism for seamless system updates
- Two sets of partitions: _a, _b
- Slots are stored in ‘misc’ partition
- Testing:
  - `bootctl set-active-boot-slot SLOT`

- Slotted partitions:
  - boot
  - dtbo
  - vbmeta
  - system (in super)
  - vendor (in super)

- Not slotted partitions:
  - misc
  - recovery
  - metadata
  - userdata
A/B: Format

- A/B format ('misc' partition):
  - Repository: https://android.googlesource.com/platform/bootable/recovery
  - File: bootloader_message/include/bootloader_message/bootloader_message.h

```c
// 4 KiB structure (A/B)
struct bootloader_message_ab {
    struct bootloader_message message;
    char slot_suffix[32];
    char update_channel[128];
    char reserved[1888];
};

// A/B metadata
struct bootloader_control {
    char slot_suffix[4];
    uint32_t magic;
    uint8_t version;
    uint8_t nb_slot : 3;
    uint8_t recovery_tries_remaining : 3;
    uint8_t reserved0[2];
    struct slot_metadata slot_info[4];
    uint8_t reserved1[8];
} __attribute__((packed));

// Slot data
struct slot_metadata {
    uint8_t priority : 4;
    uint8_t tries_remaining : 3;
    uint8_t successful_boot : 1;
    uint8_t verity_corrupted : 1;
    uint8_t reserved : 7;
} __attribute__((packed));
```

BCB (2 KiB)

COMPARE SLOTS => DEFINE CURRENT SLOT
A/B: U-Boot Implementation

- 'ab_select' command:
  - CONFIG_CMD_AB_SELECT
  - cmd/ab_select.c
  - doc/android/ab.txt
  - test/py/tests/test_android/test_ab.py

ab_select - Select the slot used to boot from and register the boot attempt.

Usage:
ab_select <slot_var_name> <interface> <dev[:part|#part_name]>

1. Load the slot metadata from the partition 'part' on device type 'interface' instance 'dev' and store the active slot in the 'slot_var_name' variable.
2. This also updates the Android slot metadata with a boot attempt, which can cause successive calls to this function to return a different result if the returned slot runs out of boot attempts.
A/B: U-Boot Usage

```c
#include/environment/ti/boot.h:

ab_select slot_name mmc 1#misc
setenv slot_suffix _$slot_name
# Now we can use $slot_suffix like: boot${slot_suffix}, dtbo${slot_suffix}
setenv bootargs $bootargs androidboot.slot_suffix=$slot_suffix
```
3. AVB

Start

Get A/B slot (from ‘misc’)

Check reboot reason (in BCB):

- Normal boot?
  - true: Choose 'boot_x' part. to load
  - false: Recovery boot?
    - true: Choose 'recovery' part. to load
    - false: Bootloader boot?
      - true: Clear BCB
      - false: Enter Fastboot

AVB verification

Cmdline: + A/B + AVB

Load chosen part.

Load DTB from chosen part.

Load DTBO from dtbo_x

Merge DTBO in DTB

Boot kernel from chosen part.

“system” is logical part:
- use ramdisk
- no system-as-root

End
AVB: Overview

- Android Verified Boot 2.0
- Establishes a chain of trust from the bootloader to system images
- Provides integrity checking for:
  - Android Boot image. RAW hashing of the whole partition is done and the hash is compared with the one stored in the VBMeta image
  - system/vendor partitions: verifying root hash of dm-verity hashtrees
- Integrity of the bootloader (U-Boot BLOB and environment) is out of scope

Source: [https://android.googlesource.com/platform/external/avb/](https://android.googlesource.com/platform/external/avb/)
AVB: U-Boot Implementation

- libavb is ported from AOSP to U-Boot
  - Repository: https://android.googlesource.com/platform/external/avb/
  - Files: libavb/*
  - In U-Boot: lib/libavb/*

- 'avb' command:
  - CONFIG_CMD_AVB
  - cmd/avb.c
  - doc/android/avb2.txt
  - test/py/tests/test_android/test_avb.py

  avb - Provides commands for testing Android Verified Boot 2.0 functionality

Usage:
avb init <dev> - initialize avb2 for <dev>
avb verify [slot_suffix] - run verification process using hash data from vbmeta
...
AVB: U-Boot Usage

include/environment/ti/boot.h:

```bash
avb_verify=avb init 1; avb verify $slot_suffix;
if run avb_verify; then
    echo AVB verification OK
    set bootargs $bootargs $avb_bootargs
else
    echo AVB verification failed
    exit
fi
```
4. Android Boot Image

Start

Get A/B slot (from ‘misc’)

Check reboot reason (in BCB):

Normal boot?

true

Choose ‘boot_x’ part. to load

false

Recovery boot?

true

Choose ‘recovery’ part. to load

false

Bootloader boot?

true

Clear BCB

false

Enter Fastboot

Load chosen part.

Load DTB from chosen part.

AVB verification

Cmdline: + A/B + AVB

Load DTBO from dtbo_x

Merge DTBO in DTB

Boot kernel from chosen part.

“system” is logical part:
- use ramdisk
- no system-as-root

End
Android Boot Image v2: Overview

- Boot Image v0 (“legacy”, pre-Android 9):
  - kernel
  - ramdisk
- Boot Image v1 (Android 9):
  - + “recovery DTBO” field (for non-A/B devices only)
- Boot Image v2 (Android 10):
  - + “DTB” field
  - Keep DTB files in Boot Image (concatenated or in DTBO image format)
- FIT image format is superior
  - ...but we can’t use it, as it’s mandatory to use Android Boot Image (see VTS)
Android Boot Image v2: Format

+---------------------+
| boot header         | 1 page
+---------------------+
| kernel              | n pages
+---------------------+
| ramdisk             | m pages
+---------------------+
| second stage        | o pages
+---------------------+
| recovery dtbo/acpio | p pages
+---------------------+
| dtb                 | q pages
+---------------------+
Android Boot Image v2: Header Format

- Android Boot Image format:
  - Repository: https://android.googlesource.com/platform/system/tools/mkbootimg
  - File: include/bootimg/bootimg.h

```c
struct boot_img_hdr_v1 : public boot_img_hdr_v0 {
    uint32_t recovery_dtbo_size;
    uint64_t recovery_dtbo_offset;
    uint32_t header_size;
} __attribute__((packed));

struct boot_img_hdr_v2 : public boot_img_hdr_v1 {
    uint32_t dtb_size;
    uint64_t dtb_addr;
} __attribute__((packed));
```
Android Boot Image v2: U-Boot Implementation

- New command: ‘bootimg’ (planned for v2020.01)
- Add androidboot.dtb_idx=N to cmdline (needed for VTS)
- Boot example:

```plaintext
include/environment/ti/boot.h:

part start mmc 1 boot_a boot_start
part size mmc 1 boot_a boot_size
mmc read $loadaddr $boot_start $boot_size
# boot image must be in $loadaddr
bootimg get_dtb_file 0 dtb_start dtb_size
cp.b $dtb_start $fdtaddr $dtb_size
fdt addr $fdtaddr 0x100000
bootm $loadaddr $loadaddr $fdtaddr
```
Android Boot Image: v3 is coming

- Seems like more changes are coming in Android 11
- Device specific data will be extracted from boot.img to vendor_boot.img:
  - load the vendor ramdisk at ramdisk_addr
  - load the generic ramdisk immediately following the vendor ramdisk in memory
- Needed for GKI (Generic Kernel Image)
- See new ‘mkbootimg’ changes: https://android-review.googlesource.com/c/platform/system/tools/mkbootimg/+/1110470
5. DTBO

Start

Get A/B slot (from ‘misc’) →

Check reboot reason (in BCB):

- Normal boot? true
  - Choose ‘boot_x’ part. to load

- Normal boot? false →
  - Recovery boot? true
    - Choose ‘recovery’ part. to load

- Normal boot? false →
  - Boot-loader boot? true
    - Clear BCB
    - Enter Fastboot

- Normal boot? false →
  - Recovery boot? false →
    - Load DTB from dtbo_x

End

Load DTB from chosen part.

Load chosen part.

Cmdline: + A/B + AVB

AVB verification

“system” is logical part:
- use ramdisk
- no system-as-root

Merge DTBO in DTB

Boot kernel from chosen part.
DTBO: Overview

- All DTBO files must be stored in ‘dtbo’ partition
- Special image format
- Use it only for DTBO files
  (DTB must reside in boot.img)

Source:
https://source.android.com/devices/architecture/dto/partitions
DTBO: Header Format

- Android Boot Image format:
  - Repository: [https://android.googlesource.com/platform/system/libufdt](https://android.googlesource.com/platform/system/libufdt)
  - File: `utils/src/dt_table.h`

```c
struct dt_table_header {
    u32 magic;
    u32 total_size;
    u32 header_size;
    u32 dt_entry_size;
    u32 dt_entry_count;
    u32 dt_entries_offset;
    u32 page_size;
    u32 version;
};
```

```c
struct dt_table_entry {
    u32 dt_size;
    u32 dt_offset;
    u32 id;
    u32 rev;
    u32 custom[4];
};
```
DTBO: U-Boot Implementation

- New ‘dtimg’ command:
  - CONFIG_CMD_DTIMG
  - cmd/dtimg.c

**dtimg** - manipulate dtb/dtbo Android image

Usage:

- **dtimg dump <addr>** - parse specified image and print its struct
- **dtimg start <addr> <index> <varname>** - get address of FDT in the image, by index
- **dtimg size <addr> <index> <varname>** - get size of FDT in the image, by index
DTBO: Usage in U-Boot

include/environment/ti/boot.h:

```
part start mmc 1 dtbo_x p_dtbo_start
part size mmc 1 dtbo_x p_dtbo_size
mmc read $dtboaddr $p_dtbo_start $p_dtbo_size
...
fdt addr $fdtaddr 0x100000

dtimg start $dtboaddr 0 dtbo_addr  # get DTBO file with index 0
fdt apply $dtbo_addr                # merge it into DTB

dtimg start $dtboaddr 1 dtbo_addr  # get DTBO file with index 1
fdt apply $dtbo_addr                # merge it into DTB

bootm $loadaddr $loadaddr $fdtaddr
```
6. Dynamic Partitions

Start

Get A/B slot (from 'misc')

Check reboot reason (in BCB):

- Normal boot?
  - true: Choose 'boot_x' part. to load
  - false: Recovery boot?
    - true: Choose 'recovery' part. to load
    - false: Bootloader boot?
      - true: Clear BCB
      - false: Enter Fastboot

AVB verification

Cmdline:
- + A/B
- + AVB

Load chosen part.

Load DTB from chosen part.

Load DTBO from dtbo_x

Merge DTBO in DTB

Boot kernel from chosen part.

"system" is logical part:
- use ramdisk
- no system-as-root

End
Dynamic Partitions: Overview

- Basically: ‘system’ and ‘vendor’ are now logical partitions inside of ‘super’ partition
- Allows sharing the space between partitions (‘system’ / ‘vendor’)
- Implemented via dm-linear KM (part of Device Mapper)
- Bootloader accessible partitions can’t be dynamic (‘boot’, ‘dtbo’, ‘vbmata’)
- Flashing can be done only via fastbootd
- super.img is pre-generated

‘super’ partition:

Source: https://www.linuxplumbersconf.org/event/2/contributions/225/
Dynamic Partitions: Boot Changes

- **Android 9 (Boot Image v1)**
  a. Early mount via dtb-fstab
  b. Don’t use recovery partition; use ramdisk from ‘boot’ image
  c. Use System-As-Root
  d. Don’t use ramdisk (pass skip_initramfs to kernel cmdline)

- **Android 10 (Boot Image v2)**
  a. Remove dtb-fstab; use regular fstab from ramdisk
  b. Use recovery partition again!
  c. Don’t use System-As-Root
  d. Use ramdisk again! (don’t pass skip_initramfs to kernel cmdline)
## Dynamic Partitions: Partition Table

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>uboot-spl</td>
<td>256K</td>
<td>- bootloader</td>
</tr>
<tr>
<td>uboot</td>
<td>2M</td>
<td></td>
</tr>
<tr>
<td>uboot-env</td>
<td>256K</td>
<td></td>
</tr>
<tr>
<td>misc</td>
<td>128K</td>
<td>- Android, not slotted</td>
</tr>
<tr>
<td>boot_a</td>
<td>20M</td>
<td></td>
</tr>
<tr>
<td>boot_b</td>
<td>20M</td>
<td></td>
</tr>
<tr>
<td>dtbo_a</td>
<td>8M</td>
<td></td>
</tr>
<tr>
<td>dtbo_b</td>
<td>8M</td>
<td></td>
</tr>
<tr>
<td>vbmeta_a</td>
<td>64K</td>
<td></td>
</tr>
<tr>
<td>vbmeta_b</td>
<td>64K</td>
<td></td>
</tr>
<tr>
<td>recovery</td>
<td>64M</td>
<td></td>
</tr>
<tr>
<td>super</td>
<td>2.5G</td>
<td>system_a</td>
</tr>
<tr>
<td>metadata</td>
<td>16M</td>
<td>system_b</td>
</tr>
<tr>
<td>userdata</td>
<td>till the end</td>
<td>vendor_a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>logical</td>
</tr>
</tbody>
</table>
Dynamic Partitions: Partition Table in U-Boot

```
include/environment/ti/boot.h:
partitions_android=uuid_disk=${uuid_gpt_disk};
name=xloader,start=128K,size=256K,uuid=${uuid_gpt_xloader};
name=bootloader,size=2048K,uuid=${uuid_gpt_bootloader};
name=uboot-env,start=2432K,size=256K,uuid=${uuid_gpt_reserved};
name=misc,size=128K,uuid=${uuid_gpt_misc};
name=boot_a,size=20M,uuid=${uuid_gpt_boot_a};
name=boot_b,size=20M,uuid=${uuid_gpt_boot_b};
name=dtbo_a,size=8M,uuid=${uuid_gpt_dtbo_a};
name=dtbo_b,size=8M,uuid=${uuid_gpt_dtbo_b};
name=vbmeta_a,size=64K,uuid=${uuid_gpt_vbmeta_a};
name=vbmeta_b,size=64K,uuid=${uuid_gpt_vbmeta_b};
name=recovery,size=64M,uuid=${uuid_gpt_recovery};
name=super,size=2560M,uuid=${uuid_gpt_super};
name=metadata,size=16M,uuid=${uuid_gpt_metadata};
name=userdata,size=-,uuid=${uuid_gpt_userdata};
=> gpt write mmc 1 $partitions_android
or
=> setenv partitions $partitions_android
$ fastboot oem format
```
Dynamic Partitions: Fastboot Changes

- Operations with logical partitions are done in fastboottd (in recovery)
- `libavb` in bootloader sets kernel cmdline for “`adb reboot fastboot/recovery`”:
  
  ```
  androidboot.verifiedbootstate=orange;
  ```

- Bootloader’s fastboot must implement next commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getvar is-userspace</code></td>
<td>Return “no”</td>
</tr>
<tr>
<td><code>reboot recovery</code></td>
<td>Reboot to recovery</td>
</tr>
<tr>
<td><code>reboot fastboot</code></td>
<td>Reboot to fastboot</td>
</tr>
<tr>
<td><code>flash system / flash vendor</code></td>
<td>Return error: “Partition should be flashed in fastboottd”</td>
</tr>
</tbody>
</table>

Source: [https://source.android.com/devices/bootloader/fastboottd](https://source.android.com/devices/bootloader/fastboottd)
Dynamic Partitions: Fastboot Changes in U-Boot

drivers/fastboot/fb_getvar.c:

```c
static const struct {
    const char *variable;
    void (*dispatch)(char *var_parameter, char *response);
} getvar_dispatch[] = {
    {
        .variable = "is-userspace",
        .dispatch = getvar_is_userspace
    },
    ...
};

static void getvar_is_userspace(
    char *var_parameter, char *response)
{
    fastboot_okay("no", response);
}
...
Get A/B slot (from ‘misc’)

Check reboot reason (in BCB):

Normal boot?

true

Choose 'boot_x' part. to load

false

Recovery boot?

true

Choose 'recovery' part. to load

false

Boot-loader boot?

true

Clear BCB

false

Enter Fastboot

Cmdline: + A/B + AVB

Load chosen part.

Load DTB from chosen part.

Load DTBO from dtbo_x

Merge DTBO in DTB

Boo kernel from chosen part.

“system” is logical part.:
- use ramdisk
- no system-as-root

End
...and now boot standards too

Source: https://xkcd.com/927/
Discussion

1. Single “android_boot” command for modern Android boot flow?
   ○ Spares scripting pain
   ○ Universal for all vendors (is it real?)

2. Shortcomings of DTBO + Android Boot Image approach
   ○ FIT image adoption?
   ○ Google vs community: communication matters

3. Is Android boot architecture mature now?
   ○ Changes are being made too fast; vendors can’t keep up...
   ○ Google vs vendors: communication matters
References

- **Bootloader requirements**: [https://source.android.com/devices/bootloader](https://source.android.com/devices/bootloader)
- **DTBO**: [https://source.android.com/devices/architecture/dto/partitions](https://source.android.com/devices/architecture/dto/partitions)
- **A/B**: [https://source.android.com/devices/tech/ota/ab](https://source.android.com/devices/tech/ota/ab)
- **AVB**:
  - [https://source.android.com/security/verifiedboot](https://source.android.com/security/verifiedboot)
  - [HKG18-124 presentation](https://www.example.com/hkg18-124)
- **Fastboot**:
  - [https://source.android.com/devices/bootloader/fastbootd](https://source.android.com/devices/bootloader/fastbootd)
  - [https://android.googlesource.com/platform/system/core/+/master/fastboot/README.md](https://android.googlesource.com/platform/system/core/+/master/fastboot/README.md)
- **CDD**: [https://source.android.com/compatibility/10/android-10-cdd.html](https://source.android.com/compatibility/10/android-10-cdd.html)
- **Pending U-Boot patches tree**: [https://github.com/joe-skb7/u-boot-misc/tree/android-next](https://github.com/joe-skb7/u-boot-misc/tree/android-next)
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org
Appendix A: AOSP Implementation of Features
AOSP: BCB

- Boot Control HAL must be implemented
- For a reference see:
  - `hardware/ti/am57x/bootctrl/*`
  - `device/ti/beagle-x15/{device.mk,manifest.xml}`
- Documentation: [https://source.android.com/devices/tech/ota/ab/ab_implement](https://source.android.com/devices/tech/ota/ab/ab_implement)
AOSP: A/B

- Boot Control HAL must be implemented

**BoardConfig.mk:**

```makefile
AB_OOTA_UPDATER := true
AB_OOTA_PARTITIONS += 
  boot 
  system 
  vbmeta 
  dtbo 
  vendor 

PRODUCT_PACKAGES += 
  update_engine 
  update_verifier

PRODUCT_PACKAGES += 
  update_engine_sideload

PRODUCT_PACKAGES_DEBUG += 
  update_engine_client
```
AOSP: AVB

**BoardConfig.mk:**

```makefile
BOARD_AVB_ENABLE := true
BOARD_BOOTIMAGE_PARTITION_SIZE := 20971520  # 20 MiB
```
AOSP: Android Boot Image v2

BoardConfig.mk:

BOARD_INCLUDE_DTB_IN_BOOTIMG := true
BOARD_PREBUILT_DTBIMAGE_DIR := $(DTB_DIR) # or generate dtb.img manually
BOARD_MKBOOTIMG_ARGS := --header_version 2

AOSP build system will concatenate all *.dtb files one after another and put it into boot image

TIP: keep DTB files list fixed, as those files are accessed by index in bootloader.
AOSP: DTBO

- Generate `dtbo.img` (unsigned):

  ```
  $ prebuilts/misc/linux-x86/libufdt/mkdtimg create <image_file> <dtbo_file ...>
  ```

  **TIP**: Either prepare the image beforehand, or generate it e.g. on droidcore target.

- Automatically sign it for AVB:

  **BoardConfig.mk**:

  ```
  BOARD_DTBOIMG_PARTITION_SIZE := 8388608 # 8 MiB
  BOARD_PREBUILT_DTBOIMAGE = $(PRODUCT_OUT)/$(DTBO_UNSIGNED)
  ```
AOSP: Dynamic Partitions (1)

BoardConfig.mk:

+BOARD_SUPER_PARTITION_SIZE := 2684354560 # 2560 MiB
+BOARD_SUPER_PARTITION_GROUPS := group_oem
+BOARD_GROUP_OEM_SIZE := 1341652992 # 1279.5 MiB
+BOARD_GROUP_OEM_PARTITION_LIST := system vendor
+BOARD_BUILD_SUPER_IMAGE_BY_DEFAULT := true
+BOARD_SUPER_IMAGE_IN_UPDATE_PACKAGE := true

-TARGET_NO_RECOVERY := true
-BOARD_USES_RECOVERY_AS_BOOT := true
-BOARD_BUILD_SYSTEM_ROOT_IMAGE := true
AOSP: Dynamic Partitions (2)

fstab.beagle_x15board:

-/dev/block/platform/44000000.ocp/480b4000.mmc/by-name/system  / ext4 ro,barrier=1 wait,slotselect
+system  /system ext4 ro,barrier=1 wait,slotselect,logical,first_stage_mount

+# Add all non-dynamic partitions except system, after this comment
...

+# Add all dynamic partitions except system, after this comment
+vendor /vendor ext4 ro,barrier=1 wait,slotselect,logical,first_stage_mount
AOSP: Dynamic Partitions (3)

- arch/arm/boot/dts/dra7xx-android-fstab.dtsi (remove dtb-fstab):

  ```
  - { firmware { android {
    - compatible = "android,firmware";
    - fstab {
      - compatible = "android,fstab";
      - vendor {
        - compatible = "android,vendor";
        - dev = "/dev/block/platform/44000000.ocp/480b4000.mmc/by-name/vendor";
        - type = "ext4";
        - mnt_flags = "ro,barrier=1";
        - fsmgr_flags = "wait,slotselect";
      - }
    - }
  - }
  ```