Device Tree Overlays

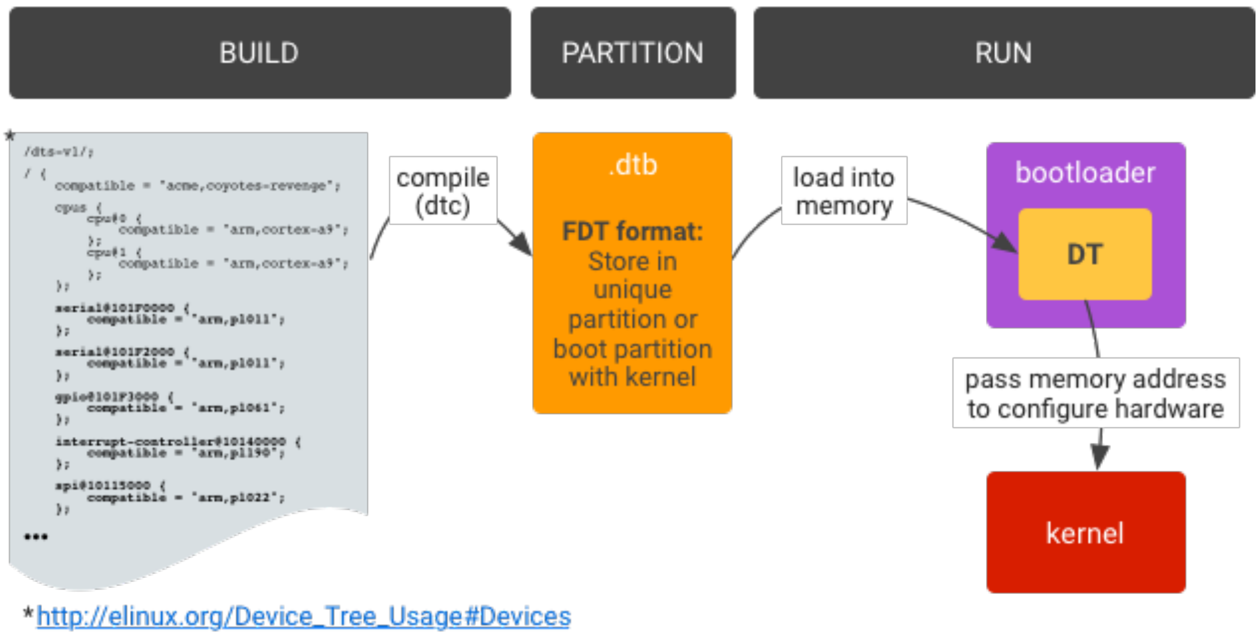
A device tree (DT) is a data structure of named nodes and properties that describe non-discoverable hardware. Operating systems, such as the Linux kernel used in Android, use DTs to support a wide range of hardware configurations used by Android-powered devices. Hardware vendors supply their own DT source files, which Linux then compiles into the Device Tree Blob (DTB) file used by the bootloader.

A device tree overlay (https://lkml.org/lkml/2012/11/5/615) (DTO) enables a central DTB to be overlaid on the device tree. A bootloader using DTO can maintain the system-on-chip (SoC) DT and dynamically overlay a device-specific DT, adding nodes to the tree and making changes to properties in the existing tree.

This page details a typical bootloader workflow for loading a DT and provides a list of common DT terms. Other pages in this section describe how to implement bootloader support for DTO (https://source.android.com/devices/architecture/dto/implement.html), how to compile (https://source.android.com/devices/architecture/dto/compile.html), verify, and optimize your DTO implementation (https://source.android.com/devices/architecture/dto/optimize.html), and how to use multiple DTs (https://source.android.com/devices/architecture/dto/multiple.html). You can also get details on DTO syntax (https://source.android.com/devices/architecture/dto/syntax.html) and recommended DTO/DTBO partition formatting (https://source.android.com/devices/architecture/dto/partition.html).

Loading a device tree

Loading a device tree in bootloader involves building, partitioning, and running.



```
/dts-v1/;
/ {
    compatible = "acme,coyotes-revenge";
    cpus {
        cpu@0 {
            compatible = "arm,cortex-a9";
        };
        cpu@1 {
            compatible = "arm,cortex-a9";
        };
    };
    serial@101F0000 {
        compatible = "arm,pl011";
    };
    serial@101F2000 {
        compatible = "arm,pl011";
    };
    gpio@101F3000 {
        compatible = "arm,pl061";
    };
    interrupt-controller@10140000 {
        compatible = "arm,pl190";
    };
    spi@10115000 {
        compatible = "arm,pl022";
    };
...
```

BUILD
PARTITION
RUN
*
compile (dtc)
.dtb
FDT format: Store in unique partition or boot partition with kernel
load into memory
bootloader
DT
pass memory address to configure hardware
kernel
*http://elinux.org/Device_Tree_Usage#Devices

Figure 1. Typical implementation for loading device tree in bootloader.

1. To build:
   - Use the device tree compiler (`dtc`) to compile device tree source (`.dts`) into a device tree blob (`.dtb`), formatted as a flattened device tree.
   - Flash the `.dtb` file into a bootloader runtime-accessible location (detailed below).
2. To partition, determine a bootloader runtime-accessible and trusted location in flash memory to put `.dtb`. Example locations:

Boot Partition

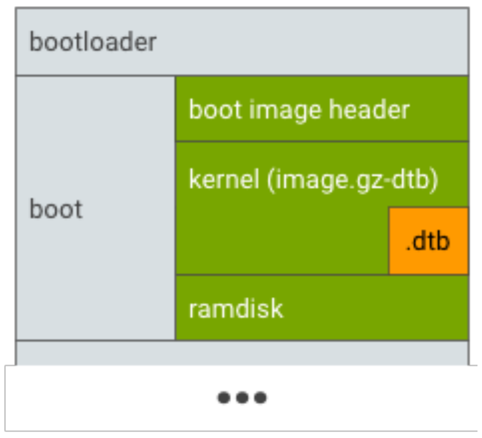


Figure 2. Put `.dtb` in boot partition by appending to `image.gz` and passing as "`kernel`" to `mkbootimg`.

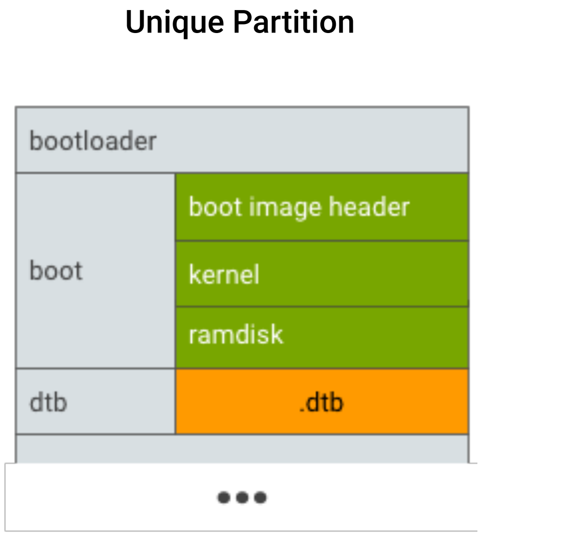


Figure 3. Put `.dtb` in an unique partition (e.g. `dtb` partition).

3. To run:
   - Load `.dtb` from storage into memory.
   - Start kernel given the memory address of the loaded DT.

Terminology

This section uses the following device tree terms:

DT	Device Tree
DTB	Device Tree Blob
DTBO	Device Tree Blob for Overlay
DTC	Device Tree Compiler
DTO	Device Tree Overlay
DTS	Device Tree Source
FDT	Flattened Device Tree, a binary format contained in a `.dtb` blob file

Last updated August 21, 2017.