Qualcomm

DT formatter – an elusive dream?

Konrad Dybcio

Senior Engineer,
Qualcomm Europe, Inc. S.A. Oddział w Polsce

@qualcomm

Agenda

Problem statement

High-level goals

Functional scope

Tooling currently in development

Problem statement

There's *a lot* of DTs

Every single machine SKU effectively has its own.

3405 files in arch/ ending in .dts as of v6.18

Codebase consistency

New code draws from existing code. Disjoint styles only grow more separate over time. This is naturally also the case for Device Trees.

See also: https://docs.kernel.org/devicetree/bindings/dts-coding-style.html

Ease of review

A unified style allows maintainers (who often review DTs for a wide array of platforms or even sub-architectures) to more easily catch logical errors that affect functionality.

All enforcement is currently manual

Not only is it inefficient, but it also lets inconsistencies through, gives room to disagreement, etc.

From a contributor's perspective, this makes the reviewers seem overly picky.

High-level goals

Improved UX/DX

Less time spent on manually looking for and addressing known issues

Reduced maintainer pushback - “please run the tool” is better than “please add a newline in these 20 places”

Lower barrier to entry for new contributors

An end to manual clean-ups

Those are tiresome to create by hand and may even cause issues if Git resolves the diff improperly

Removing variability

Maintainer A and maintainer B usually have differing opinions about a plethora of things.

Removing DTS code style from the equation leaves more room for productive discussions.

Functional scope

Whitespace, line wrapping

Wrap lines at ~100 characters, prefer one line per entry

Ensure 8-wide tabs

Align ‘<‘s / ‘[‘s in subsequent entries with spaces as necessary

Node/property order

Ensure compatible/reg/etc. are at the top, common properties precede vendor-prefixed ones, ‘status’ goes last

Order nodes by unit address then node name (with exceptions: e.g. GPIO number)

Newlines

Ensure a ‘\n’ between the last property and the following subnode, each subsequent subnode, each scoped block, logical property group

Avoid multiple stray ‘\n’s

File structure

License and #includes go at the top, then DTC directives if any, then the root node, followed by label-reference overrides to existing nodes below it

Find and merge multiple overrides to the same node

Functional scope (cont.)

LSP-like integration

The tool should be accessible and easy to invoke during development. Integration with popular text editors would be a huge User/Developer Experience win.

Idempotency

The output should be deterministic, and a single run should be good enough to make the file ready for submission.

Bringing the community to an agreement

The DT spec is rather liberal with all the points raised – if a broad agreement is reached, it may one day be updated to officially deprecate what we today see as anti-patterns.

Nice to haves

Multi-platform support

Some form of update check to alert the user about running an old release

Kernel build integration (“make dtbs_fmt”)?

Tooling currently in development

krzk/dt-schema

- Node sorting, whitespace, label & node naming rules
- regex-based
- Python script
- Almost no publicity, in-dev

axelkar/dt-tools

- Custom parser, lexer
- Linter
- LSP (VSCode + Nvim)
- Promising roadmap
- Set of Rust libraries/programs
- Almost no publicity, in-dev

Thank you

Follow us on:

For more information, visit us at qualcomm.com & qualcomm.com/blog