Device tree format v18

Simon Glass, Wed 14/11/18
sjg@chromium.org

Today

- We have used v17 for >10 years
- There is a high cost to changing
- But there are quite a few features we'd like
- Maybe it is time for v18?
- If we did change the format, what would we want?

Wish list

- Type information
  - empty: diff **fred.dts** <(dtc **fred.dts** | dtc -O dts)
- More size-efficient
- Overlay symbols
- Import into node <<
- Delete nodes and properties
- Lists (anonymous nodes)
- Little-endian format
- External data (reference blobs outside DT)
- Comments / text
- Next/previous node pointers

Things we want to keep

- Basic structure
- Efficient to directly walk through
  - Needs no pre-parsing / temporary memory
- Not too much code needed
- Existing tools

- Ideally, can we just tweak flattree.c and libfdt a bit?

Proposal

- The tag cells are wasteful
  - We can use 28 of those 32 bits for something else
- Two types of tag
  - Properties
  - Everything else
- Encode the length, string pointer and type information in a single cell
  - With overflow into the next ones if needed
  - Also encode a one-byte value into the same tag
- Compatible strings -> 32-bit vendor/product encoding
- Leave everything else along

Property tag

Bits	**0**	**1-3**	**4**	**5**	**6**	**7**	**8-15**	**16-31**
Meaning	p (1)	type	l	s/e/i	c	d	len / val	str

- p - 1 means this is a property tag
- type - 0=string, 1=phandle, 2=multiple, 3=8bit, 4=16bit, 5=32bit, 6=64bit, 7=blob
- l - 0 for scalar, 1 for list
- s - 0 for unsigned, 1 for signed
- e - ('blob' type only) 0 for internal data, 1 for external (stored outside the device tree structure)
- i - ('byte' type only) 0 for data following tag, 1 for data encoded in val
- c - 1 to include a text comment
- d - 0 for hex, 1 for decimal
- len - length of property (0-0xfe), 0xff to store length in a separate cell
- str - string table offset for property name (0xffff to store in a separate cell)

Non-property tag

Bit	**0**	**1-4**	**5**	**6-15**	**16-31**
Meaning	p (0)	t	c	skip	str

- p - 0 means this is a non-property tag
- t - tag type - 1=begin_node, 2=end_node, 4=nop, 5=delete_node, 6=merge_node, 7=list element, 9=end
- c - 1 to include a text comment
- skip - stores the offset to related tags, in units of cells (0 = none):
    - For begin_node: stores the offset to the next begin_node tag at the same level
    - For end_node: stores the offset to end_node tag just before the previous begin_node tag at the same level
- str - string table offset for name (must fit in 16 bits) Store in-place

Size experiment

- Use Linux's device tree binaries for ARM devices (arm, arm64)
  - 1267 files totalling about 32MB of binary .dtb data
  - Range from 8KB (zynq-zybo-z7.dtb) to 83KB (dra7-evm.dtb)
- Compressing with xz results gives about 5.4MB
  - Source files (as produced by fdtdump) compress from 67MB to 5.8MB
  - Entropy in the overall data about 17%?

- ~40% size saving with the proposed measures

Code changes

- Only basic fdtget support so far, no type info
- New output method in flattree.c
    - 400-line diff in a 1200-line file (so far)
- Modifications to libfdt

```
○  libfdt/fdt.c                         |   72 +-
○  libfdt/fdt.h                         |   21 +
○  libfdt/fdt_ro.c                      |   86 +-
○  libfdt/libfdt.h                      |    5 +-
○  libfdt/libfdt_internal.h             |    5 +
```

Google

Backwards compatibility

- Source format mostly unchanged
  - Could add support for delete, lists, external data
- It is easy to make dtc generate all formats
- But for libfdt it is more annoying
  - It adds quite a bit of code to run-time size
  - Can we make it a build-time option?

Feedback / Questions

- Full notes https://goo.gl/4GCyXK