Stable DeviceTree ABI

It's possible!

Embedded Linux Conference Europe 2017
Lucas Stach <l.stach@pengutronix.de>

Who's the guy in front?

- Kernel- and graphics developer at Pengutronix
- Providing customers with stable Linux based on mainline for their projects
- Helping customers to reduce long time maintanace cost by pushing things upstream

Agenda

- Why?
- How?

Why do (should) we care?

- Other projects using same DTs
  - Secure World Firmware
  - Bootloaders
  - Other OSes
- Bootloader / Firmware interaction
- User experience

Defining stable

- Frozen ABI infeasible
  - Don’t aim for impossible, it will lead to fail

- One way compatibility
  - Be able to run new OS kernel on old DT
  - Good enough for most use-cases (notable exception Enterprise Linux)

DT binding primer

Device-Tree bindings for i2c gpio driver

Required properties:

- compatible = "i2c-gpio";
- gpios: sda and scl gpio

Optional properties:

- i2c-gpio,sda-open-drain: sda as open drain
- i2c-gpio,scl-open-drain: scl as open drain
- i2c-gpio,scl-output-only: scl as output only
- i2c-gpio,delay-us: delay between GPIO operations (may depend one ach platform)
- i2c-gpio,timeout-ms: timeout to get data

Documentation/devicetree/bindings/i2c/i2c-gpio.txt

Defining stable bindings

- Provide exhaustive list of all properties
- Don’t cheap out because your system design doesn’t need some specifics!
- You can’t ever add more required properties
  - This has a maintenance cost in the long run!

Best practice

- Infer as much as possible from compatible or actual hardware

- More properties equates to higher chance of getting things wrong

Infer from compatible – the good

NVIDIA Tegra Secure Digital Host Controller

Required properties:
- compatible : should be one of:
- "nvidia,tegra20-sdhci": for Tegra20
- "nvidia,tegra30-sdhci": for Tegra30
[...]
- clocks : Must contain one entry, for the module clock.
[...]

Infer from compatible – the good

```
static const struct sdhci_pltfm_data sdhci_tegra20_pdata = {
        .quirks = SDHCI_QUIRK_BROKEN_TIMEOUT_VAL |
                  SDHCI_QUIRK_SINGLE_POWER_WRITE |
                  SDHCI_QUIRK_NO_HISPD_BIT |
                  SDHCI_QUIRK_BROKEN_ADMA_ZEROLEN_DESC |
                  SDHCI_QUIRK_CAP_CLOCK_BASE_BROKEN,
        .ops  = &tegra_sdhci_ops,
};

static const struct sdhci_tegra_soc_data soc_data_tegra20 = {
        .pdata = &sdhci_tegra20_pdata,
        .nvquirks = NVQUIRK_FORCE_SDHCI_SPEC_200 |
                    NVQUIRK_ENABLE_BLOCK_GAP_DET,
};
```

Infer from compatible – the bad

```
compatible = "ti,am335x-cpsw","ti,cpsw";
                [...]
                cpdma_channels = <8>;
                ale_entries = <1024>;
                bd_ram_size = <0x2000>;
                mac_control = <0x20>;
                slaves = <2>;
                active_slave = <0>;
                cpts_clock_mult = <0x80000000>;
                cpts_clock_shift = <29>;
```

Use new compatibles

- Always use new compatibles
  - Even if same IP block

```
ecspi1: ecspi@02008000 {
    compatible = "fsl,imx6ul-ecspi", "fsl,imx51-ecspi";
    [...]
}
```

Best practice (cont'd)

- What if you forgot to add a new compatible?
  - Maybe fix the DeviceTree
  - But first fix the code!

```
static int spi_imx_sdma_init([...])
{
    [...]
    /* use pio mode for i.mx6dl chip TKT238285 */
    if (of_machine_is_compatible("fsl,imx6dl"))
        return 0;
}
```

Best practice (cont'd)

- Need to break the binding?
  - Is you platform stable yet?
  - Number of users depending on the binding non-zero?

- Need to keep compatiblility in drivers
  - Yes, this is a maintenance burden – think twice!

Driver compatibility

- Split out parsing of old binding
  - Testing of the old binding will decline
  - Need to avoid code churn with possible regressions

Driver compatibility

```
static int imx_gpc_probe([...])
{
    [...]
    pgc_node = of_get_child_by_name(np, "pgc");
    [...]

    if (!pgc_node)
        ret = imx_gpc_old_dt_init([...]);
    else
        [... new DT parsing code]
}
```

Conclusions

- Stability can't be enforced by the DT maintainers
- It's up to the platform/driver maintainers to take responsibility
- Often it's actually easy to get things right