Porting U-Boot to a Modular Device

FOSDEM 2019

Marek Behún • marek.behun@nic.cz • 3. 2. 2019

Contents

About me

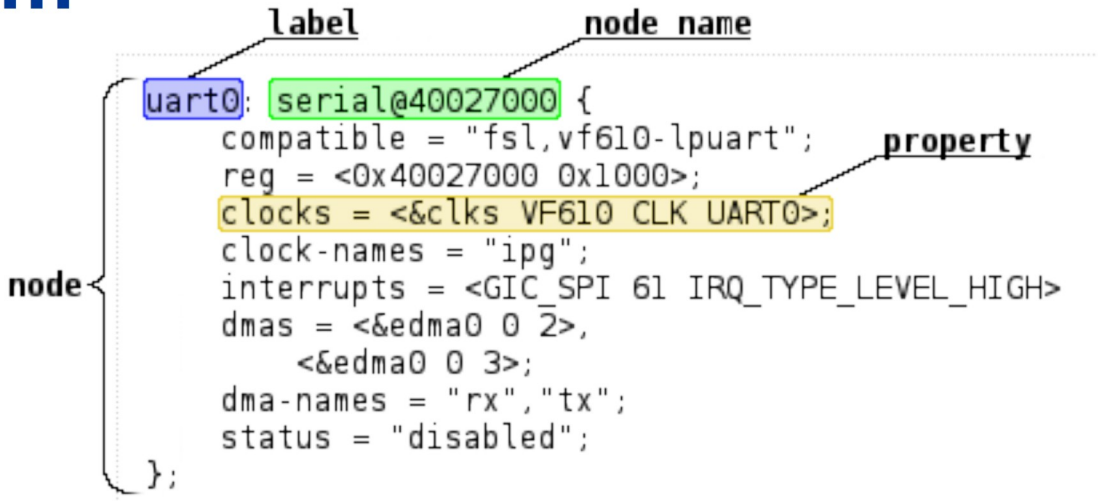
- kernel/U-Boot developer at CZ.NIC

  working on Turris{,Omnia,MOX}
- Student of CS at Charles University in Prague
- Linux user since 2005

Introducing the Problem

- What is DTS?

  Linux has to know how components on a board are connected to each other



Introducing the Problem

- What if some componets are pluggable?

  That is what buses like USB, PCIe and SDIO are for!

Introducing the Problem

- And if the components aren‘t plugged via such a bus?

  Then the device tree has to be changed.

Introducing the Problem

- Our device exposes several buses (USB, PCIe, SGMII) via one connector.
- There are several different modules which can be connected, some can be connected multiple times
- Topology of connected modules can be discovered via SPI shift register
- How to change the device tree in an elegant way?

Device tree overlays

- Device tree overlays are meant for this kind of thing!

```
kabel@dellmb /tmp/rpi-firmware/boot/overlays $ ls
3dlab-nano-player.dtbo                  gpio-key.dtbo
adau1977-adc.dtbo                       gpio-no-bank0-irq.dtbo
adau7002-simple.dtbo                    gpio-no-irq.dtbo
ads1015.dtbo                            gpio-poweroff.dtbo
ads1115.dtbo                            gpio-shutdown.dtbo
ads7846.dtbo                            hd44780-lcd.dtbo
adv7282m.dtbo                           hifiberry-amp.dtbo
```

Device tree overlays

- Can we have a device tree overlay for each module?

  NO!

- Why not?

  The parameters in the DTS nodes can depend on which modules are connected before them.

Device tree overlays

- But still, overlays can be used, or?

  Yes, they can:

  armada-3720-turris-mox.dts
  armada-3720-turris-mox-peridot.dts
  armada-3720-turris-mox-peridot-peridot.dts
  armada-3720-turris-mox-peridot-peridot-peridot.dts
  armada-3720-turris-mox-topaz.dts
  armada-3720-turris-mox-peridot-topaz.dts
  ...

ft_board_setup

- We can use the device tree setup feature of U-Boot
- It allows to call a special board function on the loaded DTB when booting Linux
- Are we going to build every node for each connected module in C?

  This is what I actually did at the beginning
- Result?

  Almost 1500 lines of ugly C code

Minimizing number of DTS changes

- Let‘s write nodes for each possible module into the main device tree
- Use `status = "disabled";` by default
- On boot time enable nodes for connected modules
- Configure the parameters for modules needning special configuration

Result

- Under 250 lines of additional C code
- +600 lines of DTS code in main DTS file
- Already reviewed on mailing list

```
res = enable_by_path(blob, SFP_GPIO_PATH);
if (res < 0)
        return res;

if (sfp_pos) {
        char newname[16];

        /* moxtet-sfp is on non-zero position, change default */
        node = node_by_path(blob, SFP_GPIO_PATH);
        if (node < 0)
                return node;

        res = fdt_setprop_u32(blob, node, "reg", sfp_pos);
        if (res < 0)
                return res;

        sprintf(newname, "gpio@%x", sfp_pos);

        res = fdt_set_name(blob, node, newname);
        if (res < 0)
                return res;
}
```

Result – Pros and Cons

- Pros:
  - Less C code, more readable, fitted to the problem
  - Changes in drivers are applied to DTS in mainline kernel (AFAIK)
- Cons:
  - We are not using overlays although they are meant for such problems

Bonus

- Currently SERDES initialization is done before SPI is probed
- Had to write own tiny implementation of SPI communication
- Fortunately only 35 LOC
- Why?

  Because chosen SERDES speed depends on module

  (SFP at 1.25 Ghz, switch at 3.125 Ghz)
- Once SERDES has a proper driver, this won‘t be needed

Thank you!

Marek Behún • marek.behun@nic.cz

CZ.NIC | SPRÁVCE DOMÉNY CZ