Utilizando Device Tree em Sistemas Embarcados

Linux Developer Conference Brazil
11/11/2017

Fabio Estevam
festevam@gmail.com

Fabio Estevam

- Trabalho na NXP Semicondutores com a linha de processadores ARM i.MX
- Contribuições: Linux Kernel, U-Boot, Buildroot
- U-Boot: Mantenedor i.MX
- Kernel: Reviewer arquitetura i.MX, drivers de áudio NXP

Agenda

- Introdução ao Device Tree / Motivação
- Exemplos de representação via DT
- Bootando um sistema com DT
- Dicas ao se fazer upstream de device trees

Device Tree

- "*Estrutura de dados em árvore que representa os dispositivos físicos de um sistema*"
- **Agnóstico** ao Sistema Operacional
- Utilizado em PowerArchitecture desde 2006.
- ARM, x86, MIPS, MicroBlaze, Sparc, PowerArchitecture
- Utilizado em diferentes projetos: **Linux kernel, U-Boot**, **Barebox**, **Zephyr**, **FreeBSD**
- https://www.devicetree.org/specifications/

Motivação do uso de DT no mundo ARM

- Bronca do Linus (abril, 2011)

“Hint for anybody on the arm list… People need to realize that the endless amounts of new pointless platform code is a problem, and since my only recourse is to say ‘if you don’t seem to try to make an effort to fix it, I won’t pull from you’, that is what I’ll eventually be doing.”

https://lwn.net/Articles/439326/

Comentário do Linus em 2017

"***I guess I can speak of one of the shining stars, which is ARM and which I used to hate and now it is one the shining starts of good behavior and I don't have any issues with them anymore***"

https://tinyurl.com/yct5dp8b @ 3:53

Representação de hardware antes do DT (board file)

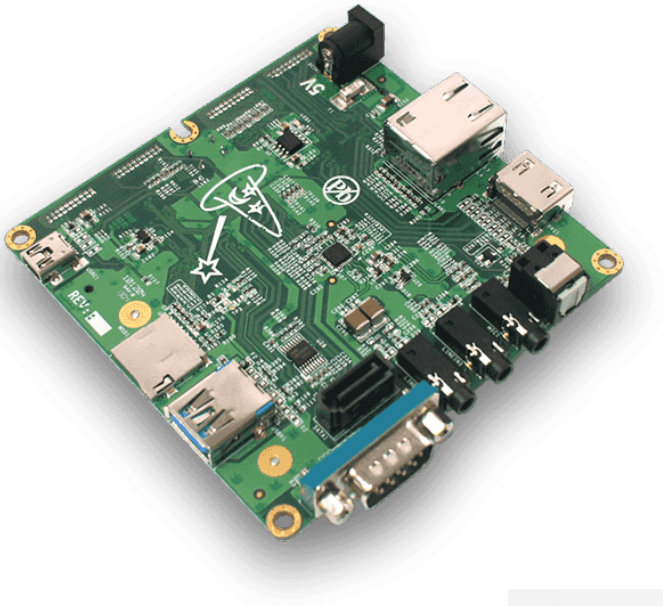
- arch/arm/mach-imx/mach-mx31_3ds.c

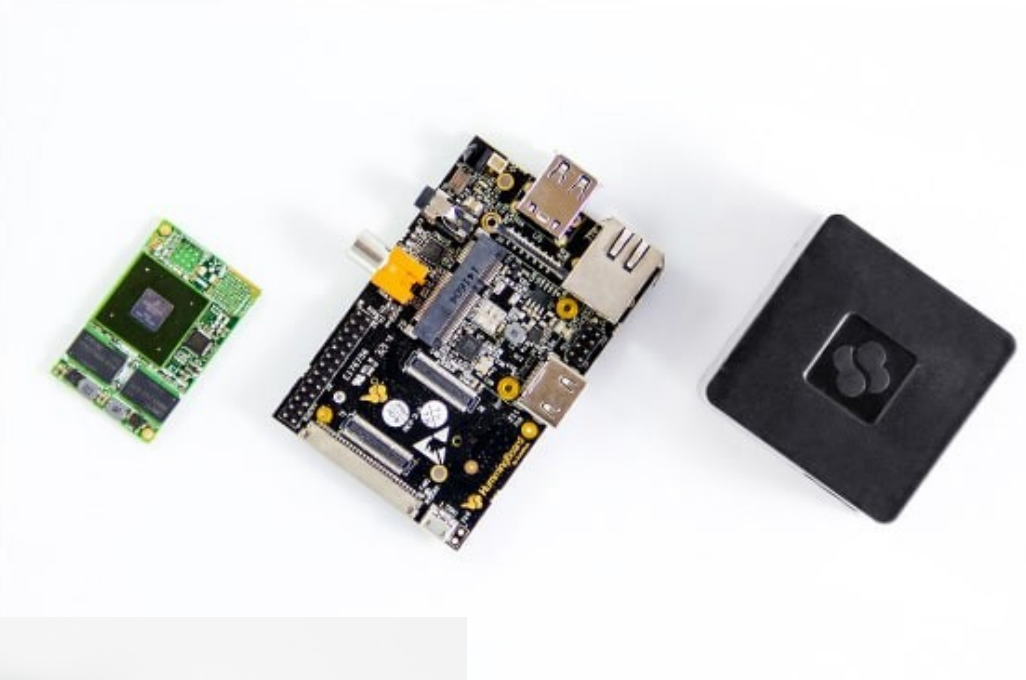
```
static const struct imxuart_platform_data uart_pdata __initconst = {
      .flags = IMXUART_HAVE_RTSCTS,
};

static const struct imxi2c_platform_data mx31_3ds_i2c0_data __initconst = {
      .bitrate = 100000,
};

static void __init mx31_3ds_init(void)
{
      imx31_soc_init();

      /* Configure SPI1 IOMUX */
      mxc_iomux_set_gpr(MUX_PGP_CSPI_BB, true);

      mxc_iomux_setup_multiple_pins(mx31_3ds_pins, ARRAY_SIZE(mx31_3ds_pins),
                              "mx31_3ds");

      imx31_add_imx_uart0(&uart_pdata);
      imx31_add_mxc_nand(&mx31_3ds_nand_board_info);

      imx31_add_spi_imx1(&spi1_pdata);

      imx31_add_imx_keypad(&mx31_3ds_keymap_data);

      imx31_add_imx2_wdt();
      imx31_add_imx_i2c0(&mx31_3ds_i2c0_data);

      imx31_add_spi_imx0(&spi0_pdata);
      imx31_add_ipu_core();
      imx31_add_mx3_sdc_fb(&mx3fb_pdata);

      imx31_add_imx_ssi(0, &mx31_3ds_ssi_pdata);

      imx_add_platform_device("imx_mc13783", 0, NULL, 0, NULL, 0);
}
```

Desvantagens do uso de board file

- Esquema não padronizado entre os diversos chips ARM
- Aumento do tamanho do kernel
- Não-escalável e difícil de manter
- Um dos motivos da bronca do Linus

Sistema de Linux Embarcado Típico





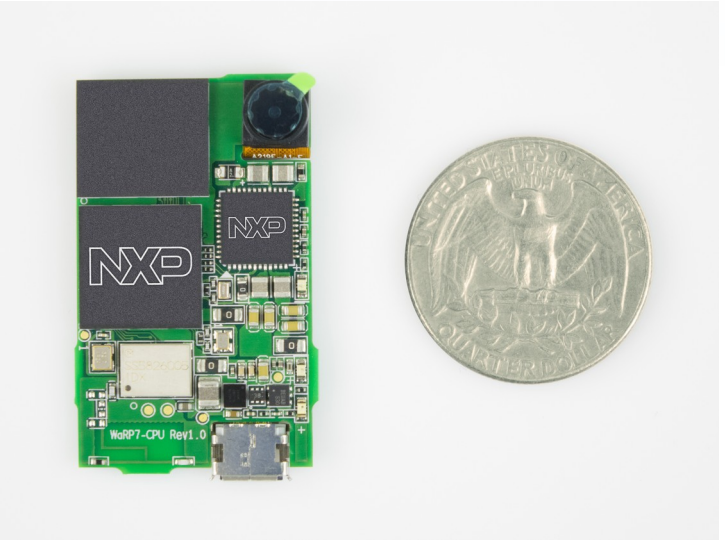
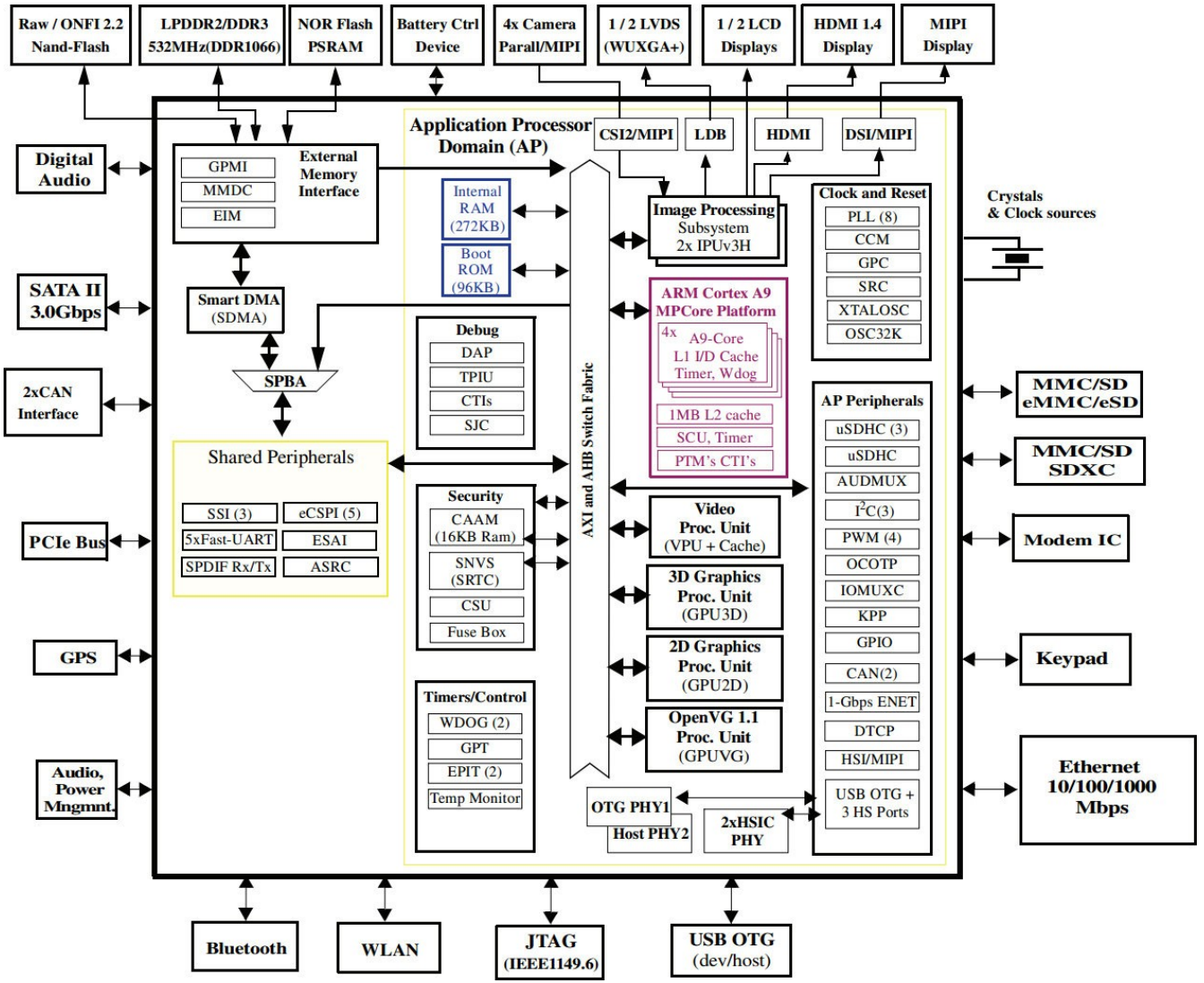


Diagrama de blocos de um SoC (i.MX6)



Representação do SoC via DT

- Cada SoC é representado por um arquivo do tipo **dtsi** (arch/arm/boot/dts)
- Número de CPUs, interrupções, clocks, memory map
- Hierarquia de barramentos
- Modelo de programação dos periféricos (**compatible** strings)

Representação do SoC via DT (cont.)

```
uart4: serial@21f0000 {
    compatible = "fsl,imx6ul-uart", "fsl,imx6q-uart";
    reg = <0x021f0000 0x4000>;
    interrupts = <GIC_SPI 29 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&clks IMX6UL_CLK_UART4_IPG>,
        <&clks IMX6UL_CLK_UART4_SERIAL>;
    clock-names = "ipg", "per";
    status = "disabled";
};

i2c4: i2c@21f8000 {
    #address-cells = <1>;
    #size-cells = <0>;
    compatible = "fsl,imx6ul-i2c", "fsl,imx21-i2c";
    reg = <0x021f8000 0x4000>;
    interrupts = <GIC_SPI 35 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&clks IMX6UL_CLK_I2C4>;
    status = "disabled";
};
```

Representação de um sistema embarcado via DT

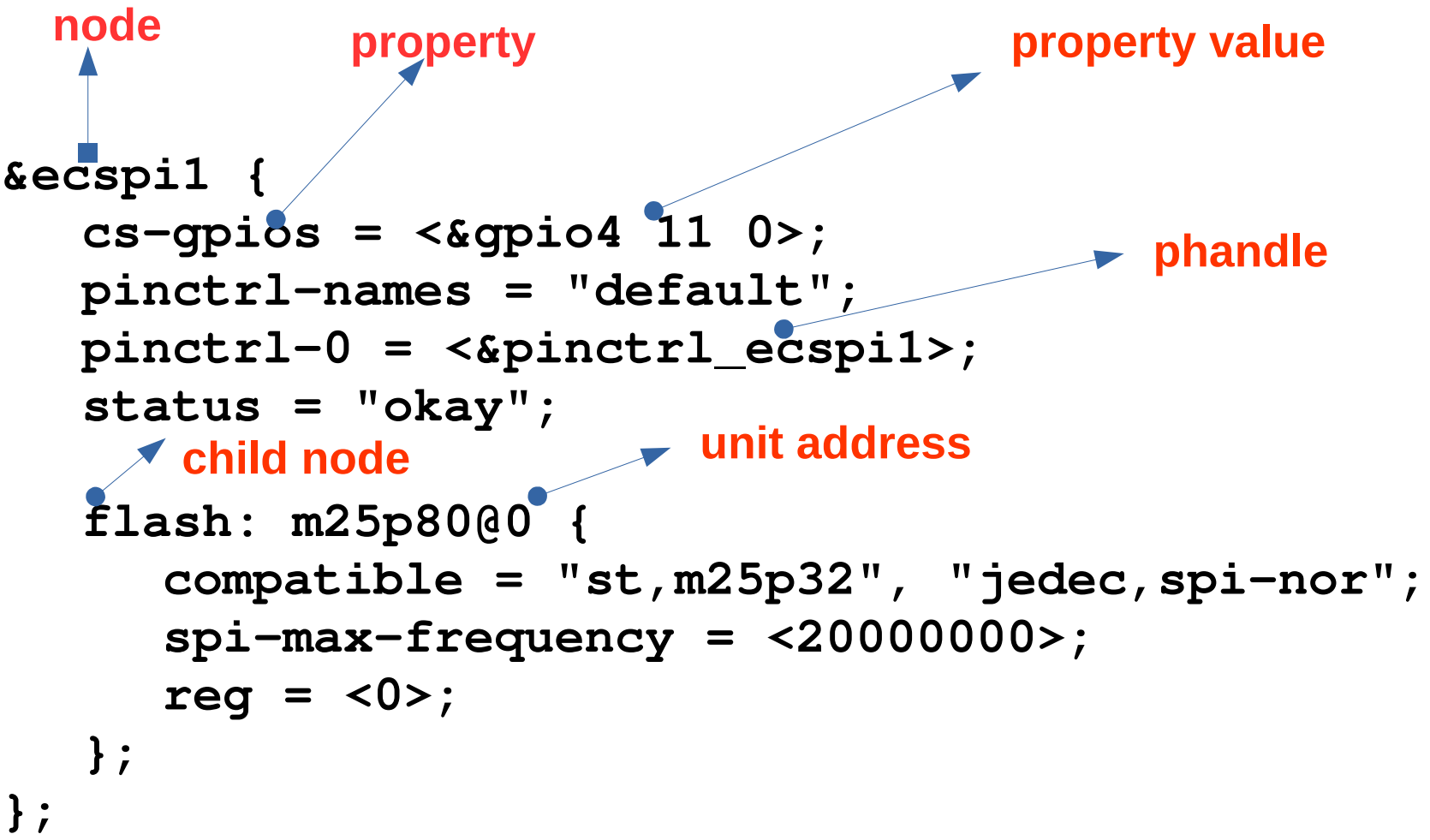
- arch/arm/boot/dts/imx6ul-pico-hobbit.dts

```
/dts-v1/;

#include "imx6ul.dtsi"

/ {
    model = "Technexion Pico i.MX6UL Board";
    compatible = "technexion,imx6ul-pico-hobbit", "fsl,imx6ul";

    memory {
        reg = <0x80000000 0x10000000>;
    };

    chosen {
        stdout-path = &uart6;
    };
```

Representação de um sistema embarcado via DT (cont)

```
&usbotg2 {
    dr_mode = "host";
    disable-over-current;
    status = "okay";
};

&usdhc1 {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_usdhc1>;
    bus-width = <8>;
    no-1-8-v;
    non-removable;
    keep-power-in-suspend;
    status = "okay";
};

&usdhc2 {  /* Wifi SDIO */
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_usdhc2>;
    no-1-8-v;
    non-removable;
    keep-power-in-suspend;
    wakeup-source;
    vmmc-supply = <&reg_brcm>;
    status = "okay";
};
```

Nomenclatura em DT

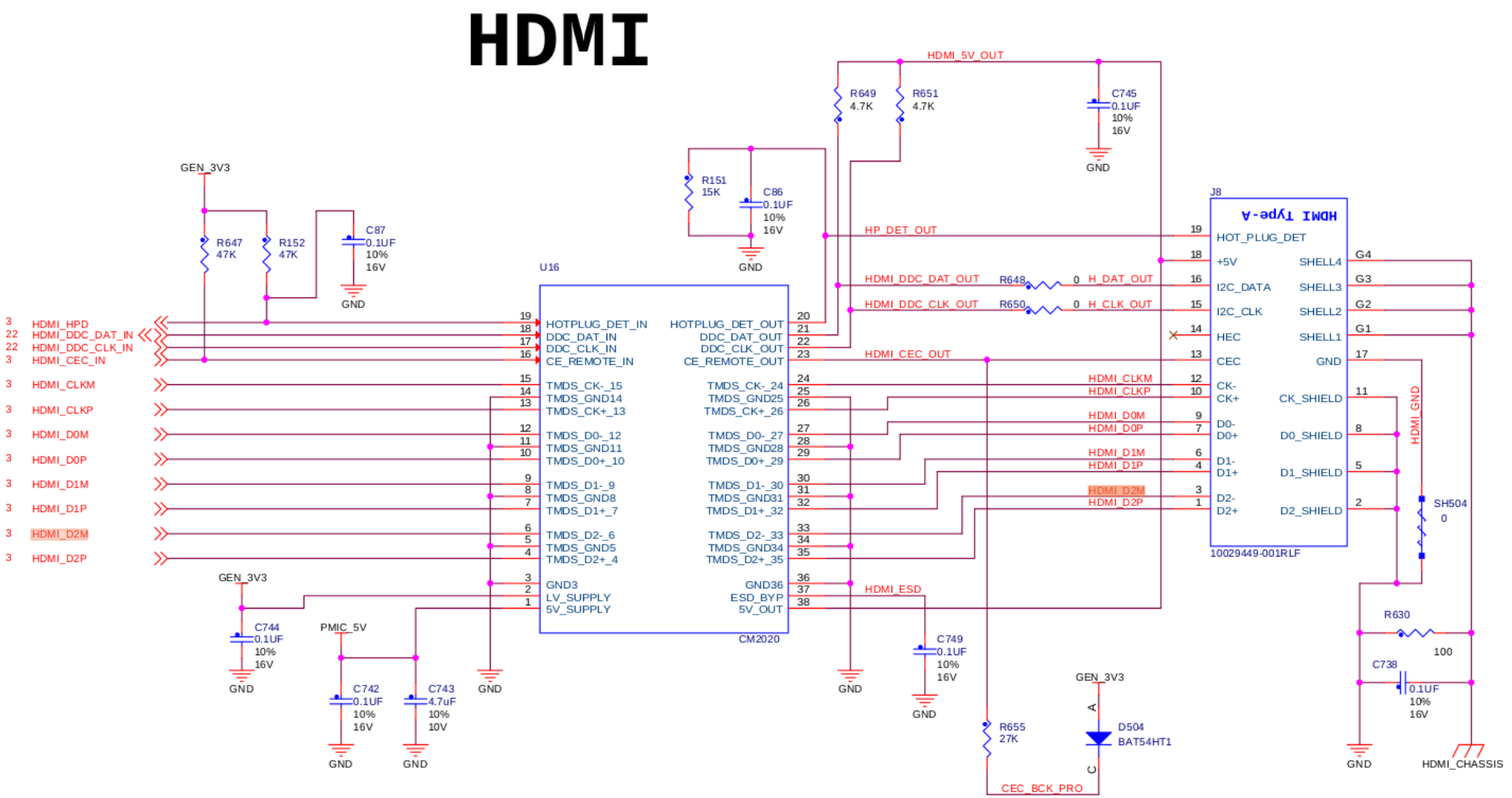


Bindings

- **Documentation/devicetree/bindings/**input/touchscreen/egalax-ts.txt

```
* EETI eGalax Multiple Touch Controller

Required properties:
- compatible: must be "eeti,egalax_ts"
- reg: i2c slave address
- interrupt-parent: the phandle for the interrupt controller
- interrupts: touch controller interrupt
- wakeup-gpios: the gpio pin to be used for waking up the controller
  and also used as irq pin

Example:

    egalax_ts@4 {
        compatible = "eeti,egalax_ts";
        reg = <0x04>;
        interrupt-parent = <&gpio1>;
        interrupts = <9 2>;
        wakeup-gpios = <&gpio1 9 0>;
    };
```

Exemplo: Representação de HDMI via DT

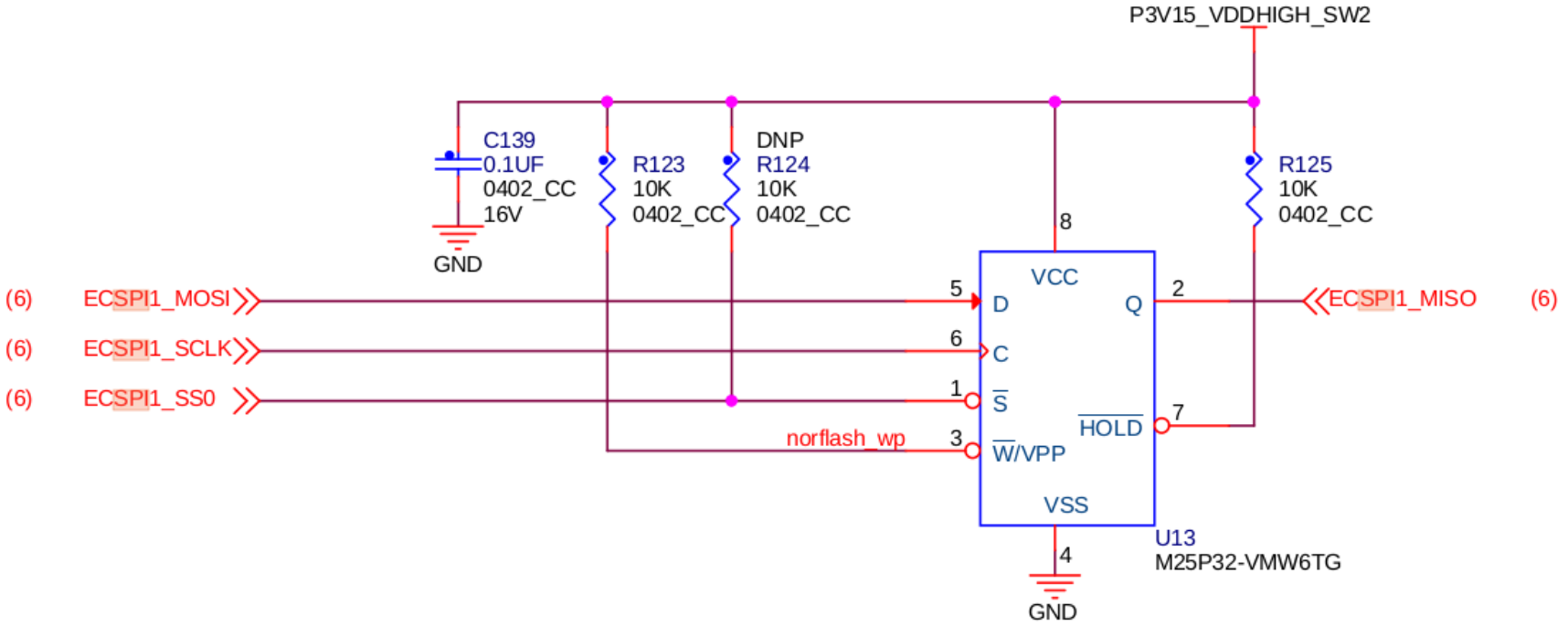


Exemplo: Representação de HDMI via DT (cont.)

- arch/arm/boot/dts/imx6qdl-sabresd.dtsi

```
&hdmi {
    ddc-i2c-bus = <&i2c2>;
    status = "okay";
};

&i2c2 {
    clock-frequency = <100000>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_i2c2>;
    status = "okay";

    ov5640: camera@3c {
    ...
    };

    pmic: pfuze100@08 {
    ....
    };
};
```

Exemplo: SPI Flash



Exemplo: SPI Flash (cont.)

- arch/arm/boot/dts/imx6sl-evk.dts

```
&ecspi1 {
    cs-gpios = <&gpio4 11 0>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_ecspi1>;
    status = "okay";

    flash: m25p80@0 {
        compatible = "st,m25p32", "jedec,spi-nor";
        spi-max-frequency = <20000000>;
        reg = <0>;
    };
};

...

        pinctrl_ecspi1: ecspi1grp {
            fsl,pins = <
                MX6SL_PAD_ECSPI1_MISO__ECSPI1_MISO   0x100b1
                MX6SL_PAD_ECSPI1_MOSI__ECSPI1_MOSI   0x100b1
                MX6SL_PAD_ECSPI1_SCLK__ECSPI1_SCLK   0x100b1
                MX6SL_PAD_ECSPI1_SS0__GPIO4_IO11 0x80000000
            >;
        };
```

Geração do dtb

- Para gerar o dtb a partir do dts:

  `make imx6q-sabresd.dtb`

- dtb’s são gerados automaticamente durante o build do kernel (arch/arm/boot/dts/Makefile)
- dtc compiler (./scripts/dtc/)

Device Tree ABI

- Um dtb atual não pode ‘quebrar’ em futuras versões de kernel
- Lembre-se que outros projetos também usam dtb’s
- Para novas propriedades, coloque-as como ‘optional’

Compilando kernel e dtb

- Compilar o kernel

```
export ARCH=arm
export CROSS_COMPILE=/usr/bin/arm-linux-gnueabi-
make imx_v6_v7_defconfig
make
```

(zImage é gerado)

- Compilar o dtb

```
make imx6q-sabresd.dtb
```

- Bootloader deve ler **dtb** e kernel (**zImage**) da media de boot e carregar na memória RAM

Booting

```
reading zImage
6378728 bytes read in 319 ms (19.1 MiB/s)
Booting from mmc ...
reading imx6q-sabresd.dtb
42122 bytes read in 19 ms (2.1 MiB/s)
## Flattened Device Tree blob at 18000000
   Booting using the fdt blob at 0x18000000
   Using Device Tree in place at 18000000, end 1800d489

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.13.3 (fabio@r49496) (gcc
version 6.4.0 (Buildroot 2017.11-rc1-00005-g97f054b)) #1
SMP Thu Nov 9 23:24:32 BRST 2017
[    0.000000] CPU: ARMv7 Processor [412fc09a] revision
10 (ARMv7), cr=10c5387d
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache,
VIPT aliasing instruction cache
[    0.000000] OF: fdt: Machine model: Freescale i.MX6
Quad SABRE Smart Device Board
```

Driver parsing

- arch/arm/boot/dts/imx51.dtsi

```
fec: ethernet@83fec000 {
    compatible = "fsl,imx51-fec", "fsl,imx27-fec";
    reg = <0x83fec000 0x4000>;
    interrupts = <87>;
    clocks = <&clks IMX5_CLK_FEC_GATE>,
         <&clks IMX5_CLK_FEC_GATE>,
         <&clks IMX5_CLK_FEC_GATE>;
    clock-names = "ipg", "ahb", "ptp";
    status = "disabled";
};
```

- arch/arm/boot/dts/imx51-babbage.dts

```
&fec {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_fec>;
    phy-mode = "mii";
    phy-reset-gpios = <&gpio2 14 GPIO_ACTIVE_LOW>;
    phy-reset-duration = <1>;
    status = "okay";
};
```

Driver parsing (cont.)

- drivers/net/ethernet/freescale/fec_main.c

```
static const struct of_device_id fec_dt_ids[] = {
    { .compatible = "fsl,imx25-fec", .data = &fec_devtype[IMX25_FEC], },
    { .compatible = "fsl,imx27-fec", .data = &fec_devtype[IMX27_FEC], },
    { .compatible = "fsl,imx28-fec", .data = &fec_devtype[IMX28_FEC], },
    { .compatible = "fsl,imx6q-fec", .data = &fec_devtype[IMX6Q_FEC], },
    { .compatible = "fsl,mvf600-fec", .data = &fec_devtype[MVF600_FEC], },
    { .compatible = "fsl,imx6sx-fec", .data = &fec_devtype[IMX6SX_FEC], },
    { .compatible = "fsl,imx6ul-fec", .data = &fec_devtype[IMX6UL_FEC], },
    { /* sentinel */ }
};
MODULE_DEVICE_TABLE(of, fec_dt_ids);
```

```
err = of_property_read_u32(np, "phy-reset-duration", &msec);
phy_reset = of_get_named_gpio(np, "phy-reset-gpios", 0);
fep->clk_ipg = devm_clk_get(&pdev->dev, "ipg");
fep->clk_ahb = devm_clk_get(&pdev->dev, "ahb");
irq = platform_get_irq(pdev, i);
r = platform_get_resource(pdev, IORESOURCE_MEM, 0);
fep->hwp = devm_ioremap_resource(&pdev->dev, r);
```

Dicas ao se fazer upstream de DT

- Compatibilidade ABI (dtb antigo precisa continuar funcionando em kernels mais novos)
- Sempre inclua nos patches:

  `Rob Herring <robh+dt@kernel.org>`
  `devicetree@vger.kernel.org`
- Compile com W=1 (make W=1 dtbs) e evite introduzir novos warnings
- Confira a implementacao com Documentation/devicetree/bindings

Dicas para fazer upstream de DT (cont.)

- Cuidado ao portar dts de vendor tree, pois pode conter bindings que não são padrões
- Pesquise em **Documentation/devicetree/bindings**
- Ao criar novos bindings, tente ser o mais genérico possível em relação à descrição do hardware e não fique preso ao suporte do driver
- Lembre-se: uma vez que o binding é aceito, ele deve ser suportado para sempre, portanto é necessário bastante cautela ao se criar novos bindings

Obrigado!