展讯通信
SPREADTRUM

Linux DTS 介绍

Chen long

目录

- DTS背景与概念
- DTS语法、工具与相关文件结构
- DTS实例介绍
- DTS相关代码分析

DTS背景与概念

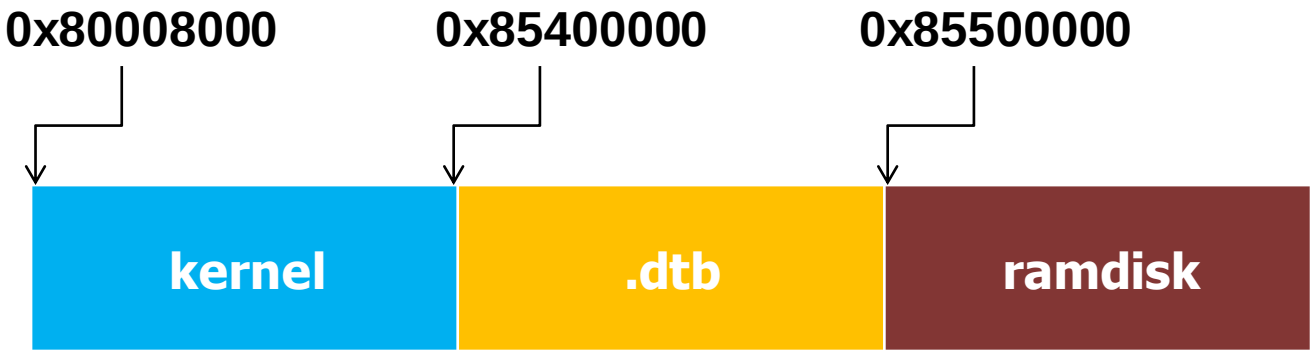
展讯通信 SPREADTRUM

- 引入DTS之前的状况
  - ARM平台使用hard code描述硬件资源。
  - 维护的痛苦，https://lkml.org/lkml/2011/3/17/492
  - 基于同一型号SoC的不同板子，需要不同的内核镜像。
- 如何解决上述问题？
  - 观察其它平台，X86与非X86系统。
  - EFI与Open Firmware。

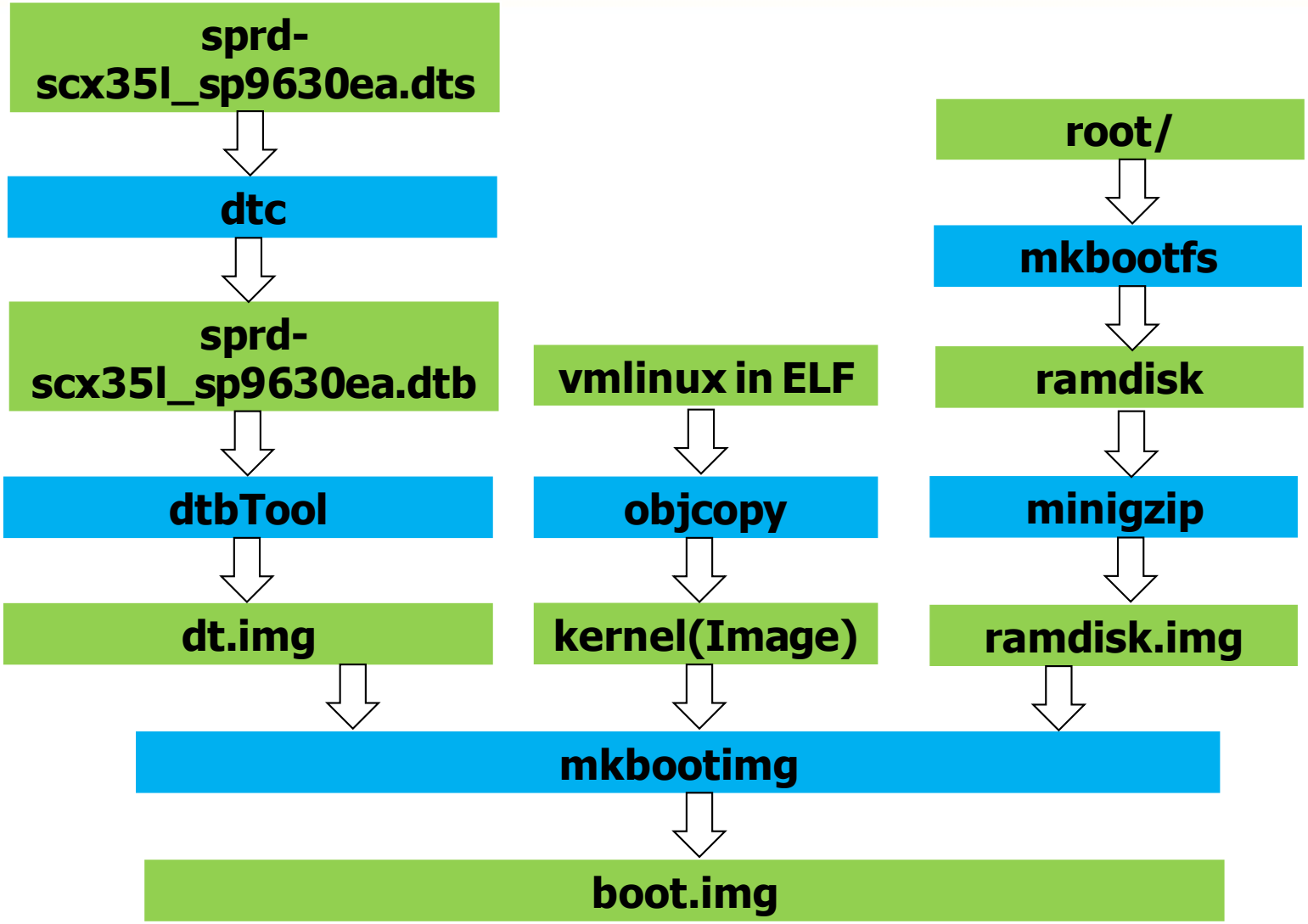
- 一些术语
  - **DTS ----- Device Tree Source，用于描述板级硬件资源的文本文件，扩展名是.dtsi或.dts 。**
  - **DTB ----- Device Tree Blob，用于描述板级硬件资源的二进制文件，扩展名是.dtb 。**
  - **DTC ----- Device Tree Compiler，用于将.dts文件编译转换成.dtb的编译器。**

DTS背景与概念


.dts文件
dtc
.dtb文件


0x80008000
0x85400000
0x85500000
kernel
.dtb
ramdisk

相关工具


sprd-scx35l_sp9630ea.dts
dtc
sprd-scx35l_sp9630ea.dtb
dtbTool
dt.img
vmlinux in ELF
objcopy
kernel(Image)
root/
mkbootfs
ramdisk
minigzip
ramdisk.img
mkbootimg
boot.img

DTB文件格式

struct boot_param_header
(alignment gap) (2)
memory reserve map
(alignment gap)
device-tree structure
(alignment gap)
device-tree strings

(1) Refer to setup.c for struct boot_param_header.
(2)alignment gaps并不是必须的，取决于各个数据块的对齐要求。

DTB文件中的节点格式

展讯通信 SPREADTRUM

Device-tree structure:

*** OF_DT_BEGIN_NODE (that is 0x00000001)**
*** for version 1 to 3, this is the node full path as a zero terminated string, starting with "/". For version 16 and later, this is the node unit name only (or an empty string for the root node)**
*** [align gap to next 4 bytes boundary]**
*** for each property:**
*** OF_DT_PROP (that is 0x00000003)**
*** 32-bit value of property value size in bytes (or 0 if no value)**
*** 32-bit value of offset in string block of property name**
*** property value data if any**
*** [align gap to next 4 bytes boundary]**
*** [child nodes if any]**
***OF_DT_END_NODE (that is 0x00000002)**

DTB初始化概览

1. u-boot从boot.img中装载.dtb到内存中,Refer to cmd_.cboot.c

2. kernel初始化时，扫描该.dtb数据，在内存中建立起相应的链表和树形结构，由of_allnodes 指向。Refer to unflatten_device_tree().

3. 内核为每一个节点，向platform bus注册一个platform_device结构。Refer to of_platform_populate().

4. 当平台驱动初始化的时候，向platform bus注册一个platform_driver结构，根据linux的驱动模型，该platform_driver结构将会与相应的platform_device结构关联。Refer to sprdfb_main.c.

sprd-scx35_sp7731gea.dts

```
gic: interrupt-controller@12001000 {
        compatible = "arm,cortex-a15-gic", "arm,cortex-a9-gic";
        #interrupt-cells = <3>;    /*  Determine if this can be a
                                       parent of INT controller  */
        #address-cells = <0>;  /*  Determine the length of address
                                   in reg property of this node only
                                   when there have no parent for
                                    this node. */
        interrupt-controller;  /*  Identify INT controller  */
        reg = <0x12001000 0x1000>,   /*  Index 0  */
              <0x12002000 0x    00>;   /*  Index 1  */
};
```

address

size

1. The linker put 'IRQCHIP_DECLARE(cortex_a15_gic, "arm,cortex-a15-gic", gic_of_init);' in irq-gic.c into the kernel according to the link script.
2. irqchip_init() in irqchip.c initializes the GIC according to the description in the DT.
3. [The callback irq_init_cb in of_irq_init()] < ===== > gic_of_init () in irq-gic.c.

sprd-scx35_sp7731gea.dts

```
gic: interrupt-controller@12001000 {
                compatible = "arm,cortex-a15-gic", "arm,cortex-a9-gic";
                #interrupt-cells = <3>;
                #address-cells = <0>;
                interrupt-controller;
                reg = <0x12001000 0x1000>,
                      <0x12002000 0x1000>;
        };

timer{
        compatible  =  "sprd,scx35-timer";
        ....
        interrupts = <0 118 0x0>,
                     ....
                     <0 121 0x0>;
};
```

SPI interrupts are in the range [0-987]. PPI interrupts are in the range [0-15].

不同于‘#address-cells’，‘#interrupt-cells’仅仅修饰它的孩子节点的‘interrupts’属性

Refer to kernel\Documentation\devicetree\bindings\arm\gic.txt. 最后一个字段是flag，0x0表示 不支持irq_set_type()，而是使用初始化时设定的缺省配置， level triggered, active low.

sprd-scx35_sp7731gea.dts

```
d_gpio_gpio: gpio@f5220000{
        compatible = "sprd,d-gpio-gpio";
        reg = <0Xf5220000 0x1000>;
        gpio-controller;
        interrupt-controller;
        #interrupt-cells = <2>;
        #gpio-cells = <2>;
        gpiobase = <0>;
        ngpios = <256>;
        interrupts = <0 35 0x0>;
    };
key_volumedown{
        label = "Volumedown Key";
        linux,code = <114>;
        gpios = <&d_gpio_gpio 124 1>;
        debounce-interval = <2>;
        gpio-key,wakeup;
    };
```

Label, optional, as phandle.

逻辑编号，用于sanity check以及可能的真实gpio编号计算，如 out_gpio = gpiobase + gpio

124是offset，1是flag，具体意义由使用该gpio的模块代码定义。Refer to gpiolib-of.c

sprd-scx35_sp7731gea.dts

```
i2c2: i2c@f5370000{
        compatible  = "sprd,i2c";
        interrupts = <0 13 0x0>;
        reg = <0xf5370000 0x1000>;
        #address-cells = <1>;
        #size-cells = <0>;
        …
        ltr_558als@23{
                compatible = "LIT     ,ltr_558als";
                reg = <0x23>;
                gpios = <&d_gpio_gpio 140 0>;
            };
};
```

蓝色部分是用于描述**i2c2 host.**

红色部分是用于描述连接到**i2c 2 host**的**client devices. reg=<0x23>**，定义了该**client**的地址。**Refer to of_i2c_register_devices().**

sprd-scx35_sp7731gea.dts

```
clk_gpu_axi: clk_gpu_axi {
        #clock-cells = <0>;
        clocks = <&clk_aon_apb>;
        clock-output-names = "clk_gpu_axi";
};

gpu{
        compatible  = "sprd,mali-ut...";
        clock-names= "clk_gpu_axi","clk_gpu","clk_153m6", ...;
        clocks = <&clk_gpu_axi>,<&clk_gpu>,<&clk_153m6>, ...;
};
```

clock-output-names属性描述了该节点的所有output cloks.

clock-cells属性描述了，该节点的child clocks中clocks属性的每个成员的参数个数。如#clock-cells = <1>; clocks = <&osc 0>, <&pll 1>;

clock-names属性描述了该节点的所有input clocks(parent clocks)。

clocks属性描述了该节点的所有parent clocks

1. of_clk_get_by_name(np, name):

根据name参数，在节点np的clocks-names属性中确定与name对应的index ===>根据index在clocks属性中找到相应的parent clock.

2. of_clk_get_parent_name(np, index):

根据index在节点np的clocks属性中找到相应的parent clock ，使用该parent clock成员中的参数作为index，在parent clock节点的clock-output-names中确定对应的clock name.

2. 也有例外，如fb0: fb@20800000 {};节点，其并未使用clocks和clock-names属性，而是使用了clock-src属性的值设置时钟。

展讯通信 SPREADTRUM

谢谢！