

深入探究Linux的设备树

讲解时间：2017年8月14日晚8时
宋宝华 <21cnbao@gmail.com>

报名直播或者录播：
<http://edu.csdn.net/huiyiCourse/detail/465>

扫描二维码报名



麦当劳喜欢您来，喜欢您再来



扫描关注
Linuxer



设备树的终极目的

<https://www.kernel.org/doc/Documentation/devicetree/usage-model.txt>

The “Open Firmware Device Tree” , or simply Device Tree (DT), is a data structure and language for describing hardware. More specifically, it is a description of hardware that is readable by an operating system so that the operating system doesn't need to hard code details of the machine.

提供一种语言来解耦硬件
配置信息

历史和现在

✓ 最早：
2005 PowerPC Linux

✓ 现在：
arm, microblaze, mips, powerpc, sparc, x86
Openrisc, c6x

X86: arch/x86/platform/ce4100 (intel凌动处理器)

Device trees
everywhere

设备端：使用设备树之前

```
static struct resource dm9000_resource1[] = {
    {
        .start = 0x20100000,
        .end   = 0x20100000 + 1,
        .flags = IORESOURCE_MEM
        ...
        .start = IRQ_PF15,
        .end   = IRQ_PF15,
        .flags = IORESOURCE_IRQ | IORESOURCE_IRQ_HIGHEDGE
    }
};
```

```
static struct platform_device dm9000_device1 = {
    .name      = "dm9000",
    .id        = 0,
    .num_resources = ARRAY_SIZE(dm9000_resource1),
    .resource  = dm9000_resource1,
};
```

```
static struct platform_device *ip0x_devices[] __initdata = {
    &dm9000_device1,
    &dm9000_device2,
};
```

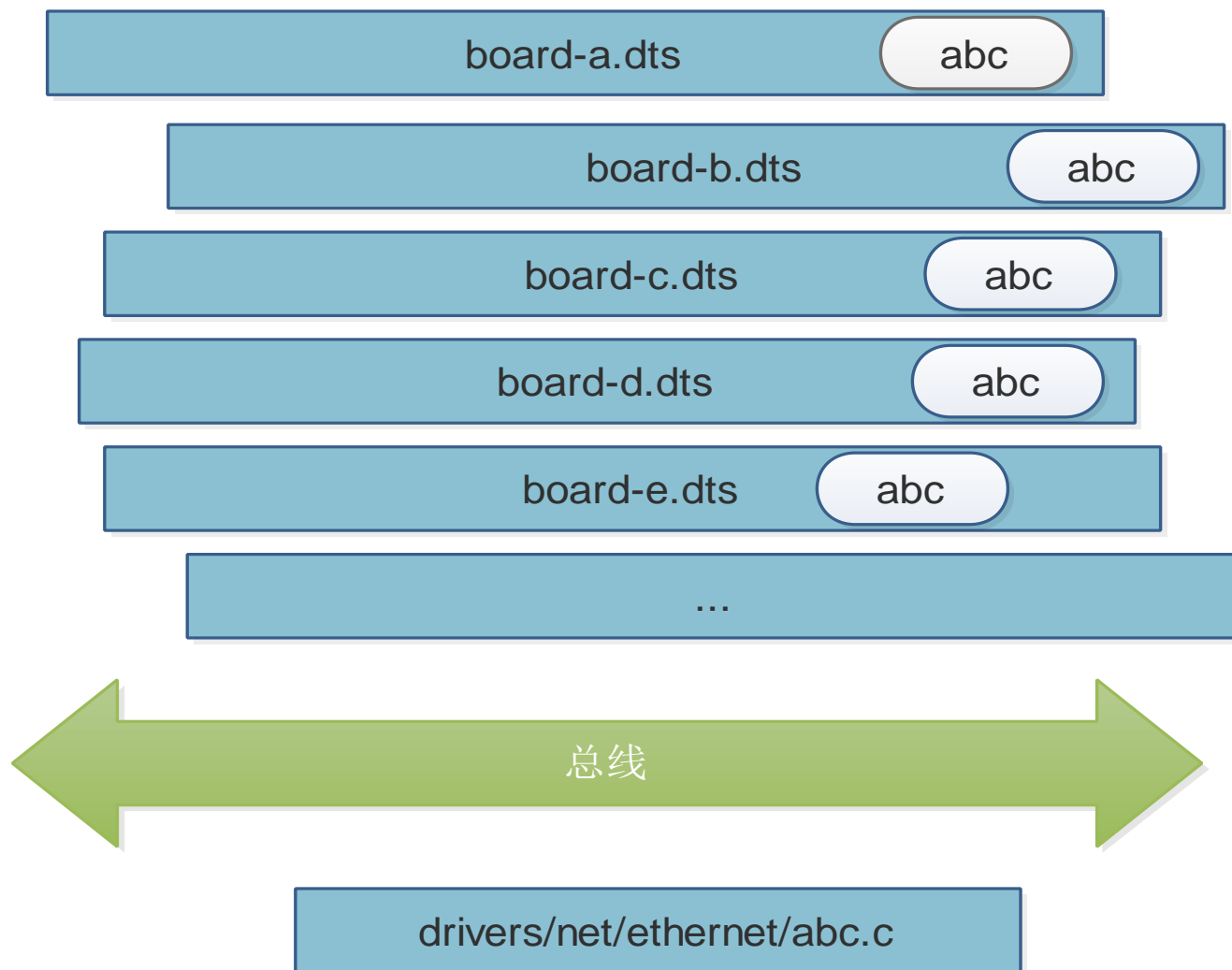
```
static int __init ip0x_init(void)
{
    platform_add_devices(ip0x_devices, ARRAY_SIZE(ip0x_devices));
}
```

设备端：使用设备树之后

开机过程中执行的类似语句会帮忙从dts节点生成platform_device:
of_platform_populate(NULL, of_default_bus_match_table,
NULL, NULL);

```
eth: eth@4,c00000 {
    compatible = "davicom,dm9000";
    reg = <
        4 0x00c00000 0x2
        4 0x00c00002 0x2
    >;
    interrupt-parent = <&gpio2>;
    interrupts = <14 IRQ_TYPE_LEVEL_LOW>;
    ...
};
```

设备在脚本，驱动在C里



驱动端:代码几乎不变: drivers/xxx/

```
static int dm9000_probe(struct platform_device *pdev)
{
    ...
    db->addr_res = platform_get_resource(pdev, IORESOURCE_MEM, 0);
    db->data_res = platform_get_resource(pdev, IORESOURCE_MEM, 1);
    db->irq_res = platform_get_resource(pdev, IORESOURCE_IRQ, 0);
    ...
}

static struct platform_driver dm9000_driver = {
    .driver = {
        .name = "dm9000",
        .pm = &dm9000_drv_pm_ops,
        .of_match_table = of_match_ptr(dm9000_of_matches),
    },
    .probe = dm9000_probe,
    .remove = dm9000_drv_remove,
};
```


ARM 设备树支持的相关补丁

Gaah. Guys, this whole ARM thing is a f*cking pain in the ass.

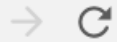
Linus, 2011,

<http://lkml.org/lkml/2011/3/17/492>



- 2011-07-25 [arm/dt: Add dtb make rule](#) Rob Herring2-0/+13
- 2011-07-25 [arm/dt: Add skeleton dtsi file](#) Grant Likely1-0/+13
- 2011-07-25 [arm/dt: Add dt machine definition](#) Grant Likely1-0/+7
- 2011-05-23 [arm/dt: probe for platforms via the device tree](#) Grant Likely6-4/+135
- 2011-05-23 [arm/dt: consolidate atags setup into setup machine atags](#) Grant Likely2-29/+47
- 2011-05-11 [arm/dt: Allow CONFIG_OF on ARM](#) Grant Likely7-1/+92
- 2011-05-11 [arm/dt: Make vet atags also accept a dtb image](#) Grant Likely2-10/+22

相似的东西-allwinner的fex



安全 | <https://github.com/hno/Allwinner-Info/blob/master/A20/A20-OLinuxIno-script.fex>

```
74  standby_mode = 1
75
76  [dram_para]
77  dram_baseaddr = 0x40000000
78  dram_clk = 384
79  dram_type = 3
80  dram_rank_num = 1
81  dram_chip_density = 4096
82  dram_io_width = 16
83  dram_bus_width = 32
84  dram_cas = 9
85  dram_zq = 0x7f
86  dram_odt_en = 0
87  dram_size = 1024
88  dram_tpr0 = 0x42d899b7
```

支持设备树的OS和平台

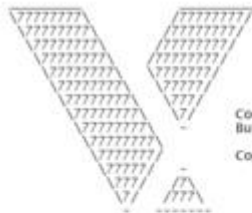


FreeBSD®

<https://wiki.freebsd.org/FlattenedDeviceTree>



```
## Starting application at 0x401010000 ...
```



```
Vxworks 7 SMP 64-bit  
Core Kernel version: 1.0.0.0  
Build date: May 30 2014 10:51:05  
Copyright Wind River Systems, Inc.  
1984-2014
```

```
Board: Wind River Dev Kit MPR  
CPU Count: 8  
OS Memory Size: 1899MB  
ED&R Policy Mode: Deployed
```

```
Adding 5290 symbols for standalone.
```

```
[vxworks]# i
```

NAME	TID	PRI	STATUS	PC	ERRNO	CPU #
tJobTask	40104cdbc0	0	PEND	401020c83c	0	-
tExcTask	40102a073c	0	PEND	401020c83c	0	-
tLogTask	40104d01d8	0	PEND	401020b0f0	0	-
tShell0	40105c1d30	1	READY	4010215e08	0	0
ipcom_tick>	401057a990	20	PEND	401020c83c	0	-
tvxdbgTask	401057dc20	25	PEND	401020c83c	0	-
tNet0	40104d3b78	50	PEND	401020c2b4	0	-
ipcom_sys1>	40104c9810	50	PEND	401020d3d4	0	-
tNetConf1	40105a6e40	50	PEND	401020c83c	0	-
m1BusMon1>	40104d5e08	252	DELAY	4010215e40	0	-
ipcom_egd	4010583c20	255	DELAY	4010215e40	0	-
tIdleTask0	40102a2fb0	287	READY	401020c004	0	-
tIdleTask1	40102a7220	287	READY	401020c00c	0	1
tIdleTask2	40102ab490	287	READY	401020c004	0	2
tIdleTask3	40102af620	287	READY	401020c004	0	3
tIdleTask4	40102b1700	287	READY	401020c004	0	4
tIdleTask5	40102b2440	287	READY	401020c004	0	5
tIdleTask6	40102a4e20	287	READY	401020c004	0	6
tIdleTask7	40102a4860	287	READY	401020c004	0	7

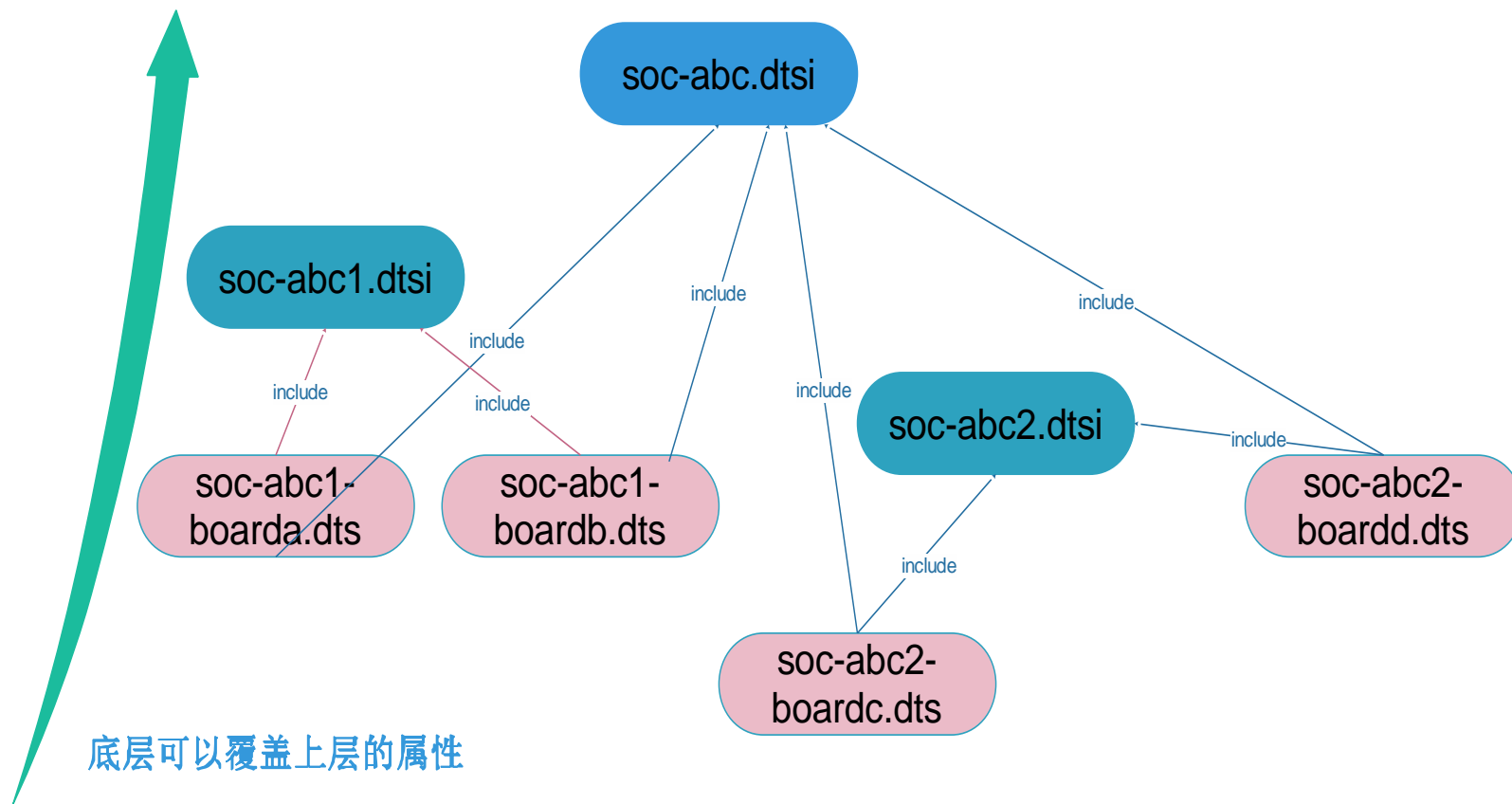
```
[vxworks]#
```



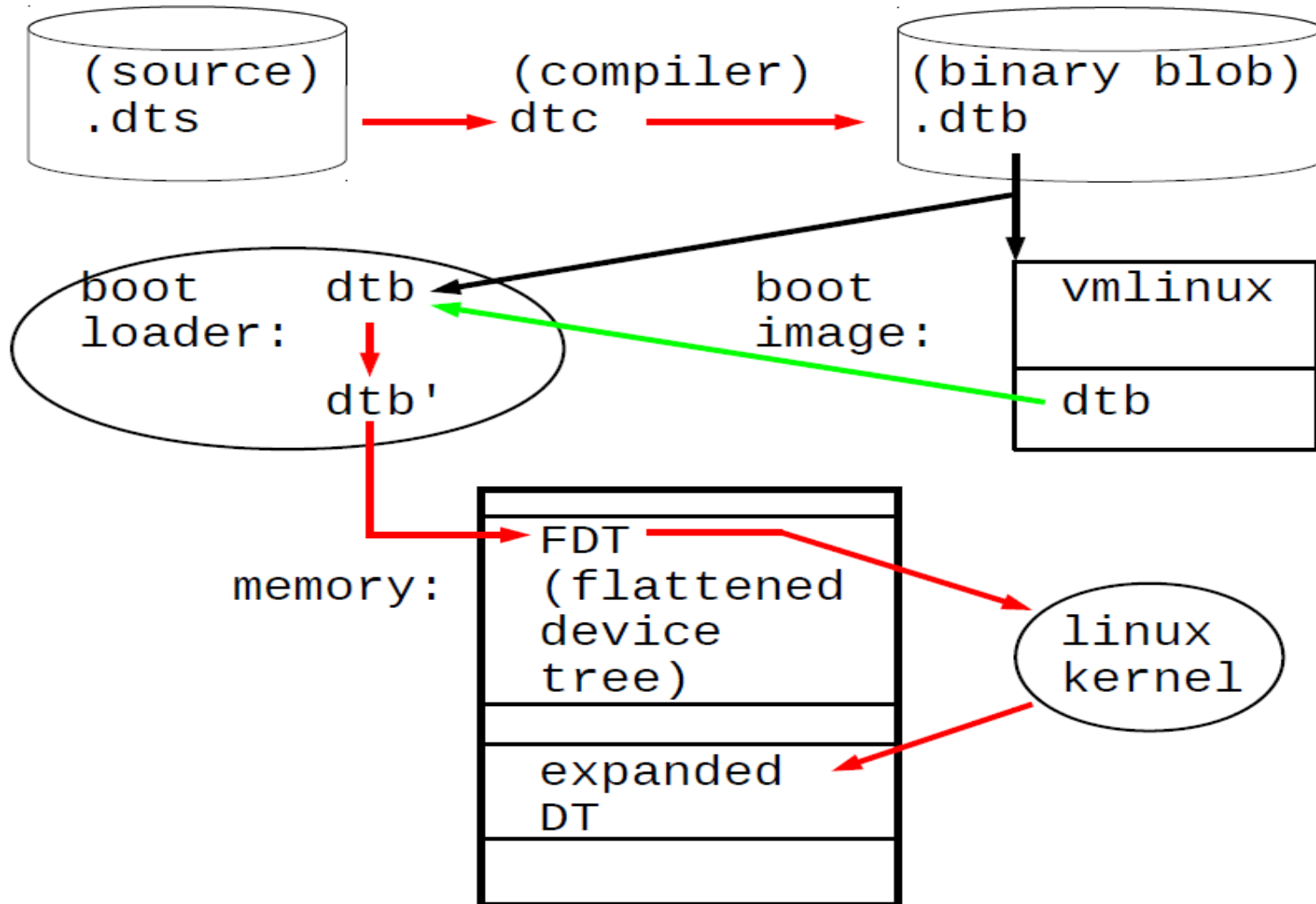
Das U-Boot –
the Universal Boot Loader



dtsi 和 dts



设备树的生命周期



脚本、代码、文档

.dts 节点

读取 

.c 代码



解释节点属性

.txt DT binding 文档

节点和属性

节点/子节点

```
i2c0: i2c@7f004000 {
```

```
compatible = "samsung,s3c2440-i2c";  
reg = <0x7f004000 0x1000>;  
interrupt-parent = <&vic1>;  
interrupts = <18>;  
clock-names = "i2c";  
clocks = <&clocks PCLK_IICo>;  
status = "disabled";  
#address-cells = <1>;  
#size-cells = <0>;
```

属性

```
g762@3e {
```

```
compatible = "gmt,g762";  
reg = <0x3e>;  
clocks = <&g762_clk>;
```

指向

```
};
```

```
vic1: interrupt-controller@64000 {  
    interrupt-controller;  
  
    ...  
};
```

一个bool节点属性的来龙去脉

omap5-sbc-t54.dts

```
&mmc1 {  
    ...  
    cd-inverted;  
    wp-inverted;  
    ...  
};
```

脚本

drivers/mmc/core/host.c

```
cd_cap_invert = of_property_read_bool(np, "cd-  
inverted");  
ro_cap_invert = of_property_read_bool(np, "wp-  
inverted");
```

代码

Documentation/devicetree/bindings/
mmc/mmc.txt

文档

- cd-inverted: when present, polarity on the CD line is inverted. See the note below for the case, when a GPIO is used for the CD line
- wp-inverted: when present, polarity on the WP line is inverted. See the note below for the case, when a GPIO is used for the WP line

一个u32数组节点属性的来龙去脉

tegra30.dtsi

```
arm,data-latency =  
<6 6 2>;
```

脚本

arch/arm/mm/cache-l2x0.c

```
of_property_read_u32_array(np, "arm,data-latency",  
data, ARRAY_SIZE(data));
```

代码

Documentation/devicetree/
bindings/arm/l2cc.txt

文档

- arm,data-latency : Cycles of latency for Data RAM accesses. Specifies 3 cells of read, write and setup latencies. Minimum valid values are 1. Controllers without setup latency control should use a value of 0.

设备树数据的三大作用

平台标识

platform identification

用DT来标识特定的machine;

root节点的compatible字段, 匹配machine_desc的dt_compat

比如:

```
compatible = "ti,omap3-beagleboard",  
"ti,omap3450", "ti,omap3";
```

运行时配置

runtime configuration

chosen节点的属性

```
chosen {  
    bootargs = "console=ttyS0,115200  
loglevel=8";  
    initrd-start = <0xc8000000>;  
    initrd-end = <0xc8200000>;  
};
```

设备信息集合

device population

```
serial@70006300 {  
    compatible = "nvidia,tegra20-uart";  
    reg = <0x70006300 0x100>;  
    interrupts = <122>;  
};
```

平台标识 - DT_MACHINE

```
mach-meson/meson.c:DT_MACHINE_START(MESON, "Amlogic Meson platform")
mach-mmp/mmp-dt.c:DT_MACHINE_START(PXA168_DT, "Marvell PXA168 (Device Tree Support)")
mach-mmp/mmp-dt.c:DT_MACHINE_START(PXA910_DT, "Marvell PXA910 (Device Tree Support)")
mach-mmp/mmp2-dt.c:DT_MACHINE_START(MMP2_DT, "Marvell MMP2 (Device Tree Support)")
mach-mvebu/board-v7.c:DT_MACHINE_START(ARMADA_370_XP_DT, "Marvell Armada 370/XP (Device Tree)")
mach-mvebu/board-v7.c:DT_MACHINE_START(ARMADA_375_DT, "Marvell Armada 375 (Device Tree)")
mach-mvebu/board-v7.c:DT_MACHINE_START(ARMADA_38X_DT, "Marvell Armada 380/385 (Device Tree)")
mach-mvebu/dove.c:DT_MACHINE_START(DOVE_DT, "Marvell Dove")
mach-mvebu/kirkwood.c:DT_MACHINE_START(KIRKWOOD_DT, "Marvell Kirkwood (Flattened Device Tree)")
mach-mxs/mach-mxs.c:DT_MACHINE_START(MXS, "Freescale MXS (Device Tree)")
mach-nomadik/cpu-8815.c:DT_MACHINE_START(NOMADIK_DT, "Nomadik STn8815")
mach-nspire/nspire.c:DT_MACHINE_START(NSPIRE, "TI-NSPIRE")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP242X_DT, "Generic OMAP2420 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP243X_DT, "Generic OMAP2430 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP3_N900_DT, "Nokia RX-51 board")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP3_DT, "Generic OMAP3 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP36XX_DT, "Generic OMAP36xx (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP3_GP_DT, "Generic OMAP3-GP (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(AM3517_DT, "Generic AM3517 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(TI81XX_DT, "Generic ti814x (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(TI816X_DT, "Generic ti816x (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(AM33XX_DT, "Generic AM33XX (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP4_DT, "Generic OMAP4 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(OMAP5_DT, "Generic OMAP5 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(AM43_DT, "Generic AM43 (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(DRA74X_DT, "Generic DRA74X (Flattened Device Tree)")
mach-omap2/board-generic.c:DT_MACHINE_START(DRA72X_DT, "Generic DRA72X (Flattened Device Tree)")
mach-orion5x/board-dt.c:DT_MACHINE_START(ORION5X_DT, "Marvell Orion5x (Flattened Device Tree)")
mach-picocell/common.c:DT_MACHINE_START(PICOXCELL, "Picochip picoXcell")
mach-prima2/common.c:DT_MACHINE_START(ATLAS6_DT, "Generic ATLAS6 (Flattened Device Tree)")
mach-prima2/common.c:DT_MACHINE_START(PRIMA2_DT, "Generic PRIMA2 (Flattened Device Tree)")
mach-prima2/common.c:DT_MACHINE_START(ATLAS7_DT, "Generic ATLAS7 (Flattened Device Tree)")
mach-pxa/pxa-dt.c:DT_MACHINE_START(PXA_DT, "Marvell PXA3xx (Device Tree Support)")
mach-pxa/pxa-dt.c:DT_MACHINE_START(PXA27X_DT, "Marvell PXA2xx (Device Tree Support)")
mach-qcom/board.c:DT_MACHINE_START(QCOM_DT, "Qualcomm (Flattened Device Tree)")
```

平台标识 - mach-omap2/board-generic.c

```
#ifdef CONFIG_SOC_OMAP2420
static const char *const omap242x_boards_compat[] __initconst = {
    "ti,omap2420",
    NULL,
};

DT_MACHINE_START(OMAP242X_DT, "Generic OMAP2420 (Flattened Device Tree)")
    .reserve           = omap_reserve,
    .map_io            = omap242x_map_io,
    .init_early        = omap2420_init_early,
    .init_machine      = omap_generic_init,
    .init_time         = omap2_sync32k_timer_init,
    .dt_compat         = omap242x_boards_compat,
    .restart           = omap2xxx_restart,
MACHINE_END
#endif

#ifdef CONFIG_SOC_OMAP2430
static const char *const omap243x_boards_compat[] __initconst = {
    "ti,omap2430",
    NULL,
};

DT_MACHINE_START(OMAP243X_DT, "Generic OMAP2430 (Flattened Device Tree)")
    .reserve           = omap_reserve,
    .map_io            = omap243x_map_io,
    .init_early        = omap2430_init_early,
    .init_machine      = omap_generic_init,
    .init_time         = omap2_sync32k_timer_init,
    .dt_compat         = omap243x_boards_compat,
    .restart           = omap2xxx_restart,
MACHINE_END
#endif
```

平台标识 - dts与machine匹配

omap2420-n800.dts

匹配common machine

```
/*dts-v1*/;

#include "omap2420-n8x0-common.dtsi"

/ {
    model = "Nokia N800";
    compatible = "nokia,n800", "nokia,n8x0", "ti,omap2420", "ti,omap2";
};
```

从具体到抽象

```
#define board_is_n800() (board_caps & NOKIA_N800)
#define board_is_n810() (board_caps & NOKIA_N810)
#define board_is_n810_wimax() (board_caps & NOKIA_N810_WIMAX)

static void board_check_revision(void)
{
    if (of_have_populated_dt()) {
        if (of_machine_is_compatible("nokia,n800"))
            board_caps = NOKIA_N800;
        else if (of_machine_is_compatible("nokia,n810"))
            board_caps = NOKIA_N810;
        else if (of_machine_is_compatible("nokia,n810-wimax"))
            board_caps = NOKIA_N810_WIMAX;
    }

    if (!board_caps)
        pr_err("Unknown board\n");
}
```

运行时配置-U-Boot修改dtb

用户设置bootargs

```
int fdt_chosen(void *fdt)
{
    int nodeoffset;
    int err;
    char *str;          /* used to set string properties */

    err = fdt_check_header(fdt);
    ...

    /* find or create "/chosen" node. */
    nodeoffset = fdt_find_or_add_subnode(fdt, 0, "chosen");
    ...

    str = getenv("bootargs");
    if (str) {
        err = fdt_setprop(fdt, nodeoffset, "bootargs", str,
                          strlen(str) + 1);
        ...
    }
    ...
}
```

运行时配置 - U-Boot设备树相关命令

```
#define CONFIG_OF_LIBFDT          /* Device Tree support */
```

Usage:

fdt addr <addr> [<length>] - Set the fdt location to <addr>

fdt move <fdt> <newaddr> <length> - Copy the fdt to <addr> and make it active

fdt resize - Resize fdt to size + padding to 4k addr

fdt print <path> [<prop>] - Recursive print starting at <path>

fdt list <path> [<prop>] - Print one level starting at <path>

fdt set <path> <prop> [<val>] - Set <property> [to <val>]

fdt mknnode <path> <node> - Create a new node after <path>

fdt rm <path> [<prop>] - Delete the node or <property>

fdt header - Display header info

fdt bootcpu <id> - Set boot cpuid

fdt memory <addr> <size> - Add/Update memory node

fdt rsvmem print - Show current mem reserves

fdt rsvmem add <addr> <size> - Add a mem reserve

fdt rsvmem delete <index> - Delete a mem reserves

fdt chosen [<start> <end>] - Add/update the /chosen branch in the tree

设备信息 - 展开 platform_device

customize_machine()或者init_machine()会调用of_platform_populate()函数会为“simple-bus”节点生成和展开platform_device

```
struct platform_device *of_device_alloc(struct device_node *np,
                                        const char *bus_id,
                                        struct device *parent)
{
    struct platform_device *dev;
    int rc, i, num_reg = 0, num_irq;
    struct resource *res, temp_res;

    dev = platform_device_alloc("", -1);
    if (!dev)
        return NULL;

    /* count the io and irq resources */
    while (of_address_to_resource(np, num_reg, &temp_res) == 0)
        num_reg++;
    num_irq = of_irq_count(np);

    /* Populate the resource table */
    if (num_irq || num_reg) {
        res = kzalloc(sizeof(*res) * (num_irq + num_reg), GFP_KERNEL);
        ...
        dev->num_resources = num_reg + num_irq;
        dev->resource = res;
        for (i = 0; i < num_reg; i++, res++) {
            rc = of_address_to_resource(np, i, res);
            WARN_ON(rc);
        }
        if (of_irq_to_resource_table(np, res, num_irq) != num_irq)
            ...
    }
    ...
}
```

设备驱动模型连本质都没有变!

设备信息 - 展开 i2c 子节点

i2c_register_adapter()函数会调用**of_i2c_register_devices()**生成和展开**i2c device**

```
static struct i2c_client *of_i2c_register_device(struct i2c_adapter *adap,
                                                struct device_node *node)
{
    ...
    if (of_modalias_node(node, info.type, sizeof(info.type)) < 0) {
        ...
    }

    addr = of_get_property(node, "reg", &len);
    ...
    info.addr = be32_to_cpup(addr);
    ...
    result = i2c_new_device(adap, &info);
    ...
    return result;
}
```

设备信息 - 展开spi子节点

spi_register_master()函数会调用of_register_spi_devices() 为子节点生成和展开spi device

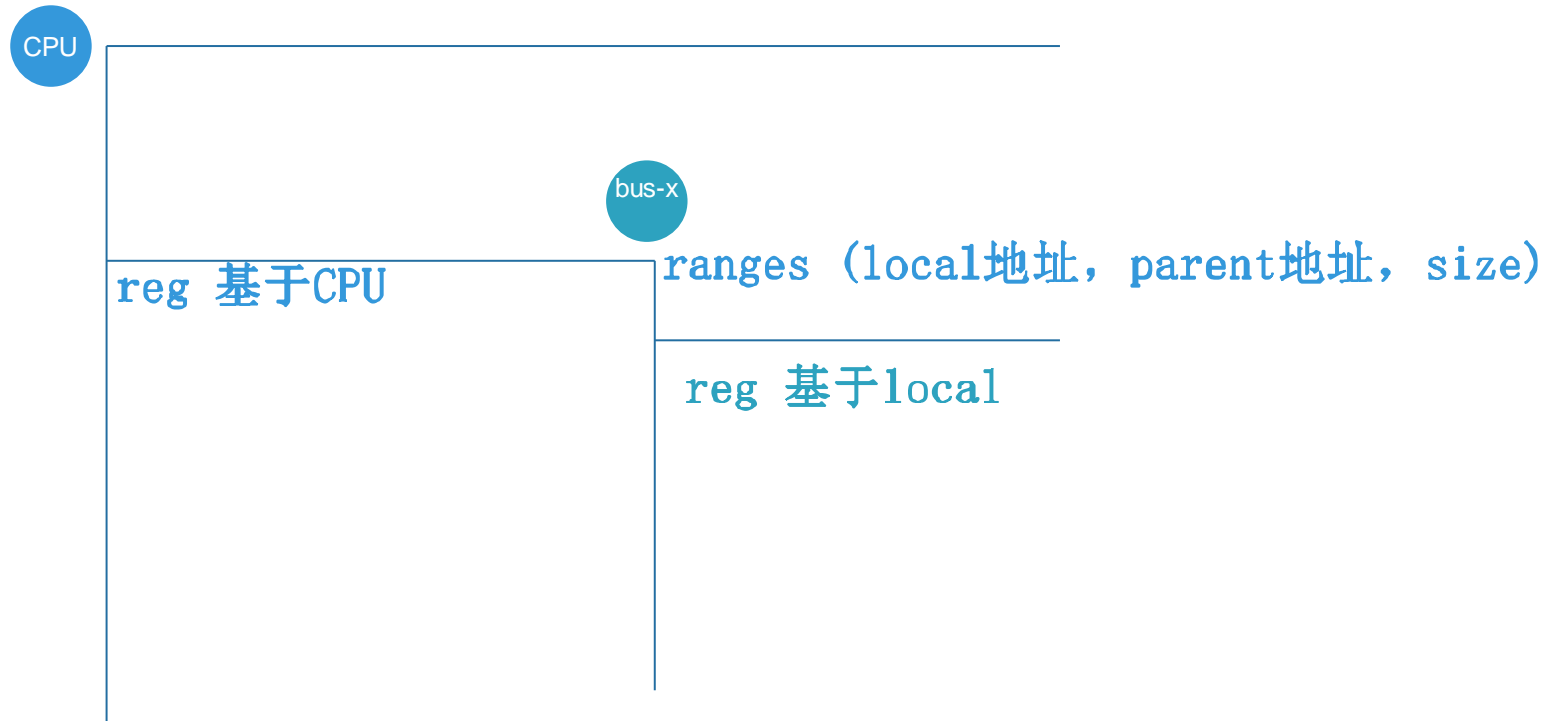
```
static void of_register_spi_devices(struct spi_master *master)
{
    ...

    for_each_available_child_of_node(master->dev.of_node, nc) {
        spi = of_register_spi_device(master, nc);
        if (IS_ERR(spi))
            dev_warn(&master->dev, "Failed to create SPI device for %s\n",
                    nc->full_name);
    }
}

static struct spi_device *
of_register_spi_device(struct spi_master *master, struct device_node *nc)
{
    rc = of_property_read_u32(nc, "reg", &value);
    if (rc) {
        dev_err(&master->dev, "%s has no valid 'reg' property (%d)\n",
                nc->full_name, rc);
        goto err_out;
    }
    spi->chip_select = value;
    rc = spi_add_device(spi);
    ...
}
```

ranges

ranges代表了**local**地址向**parent**地址的转换;
ranges为空代表**1:1**映射;
无**range**代表不是**memory map**区域



ranges(cont.)

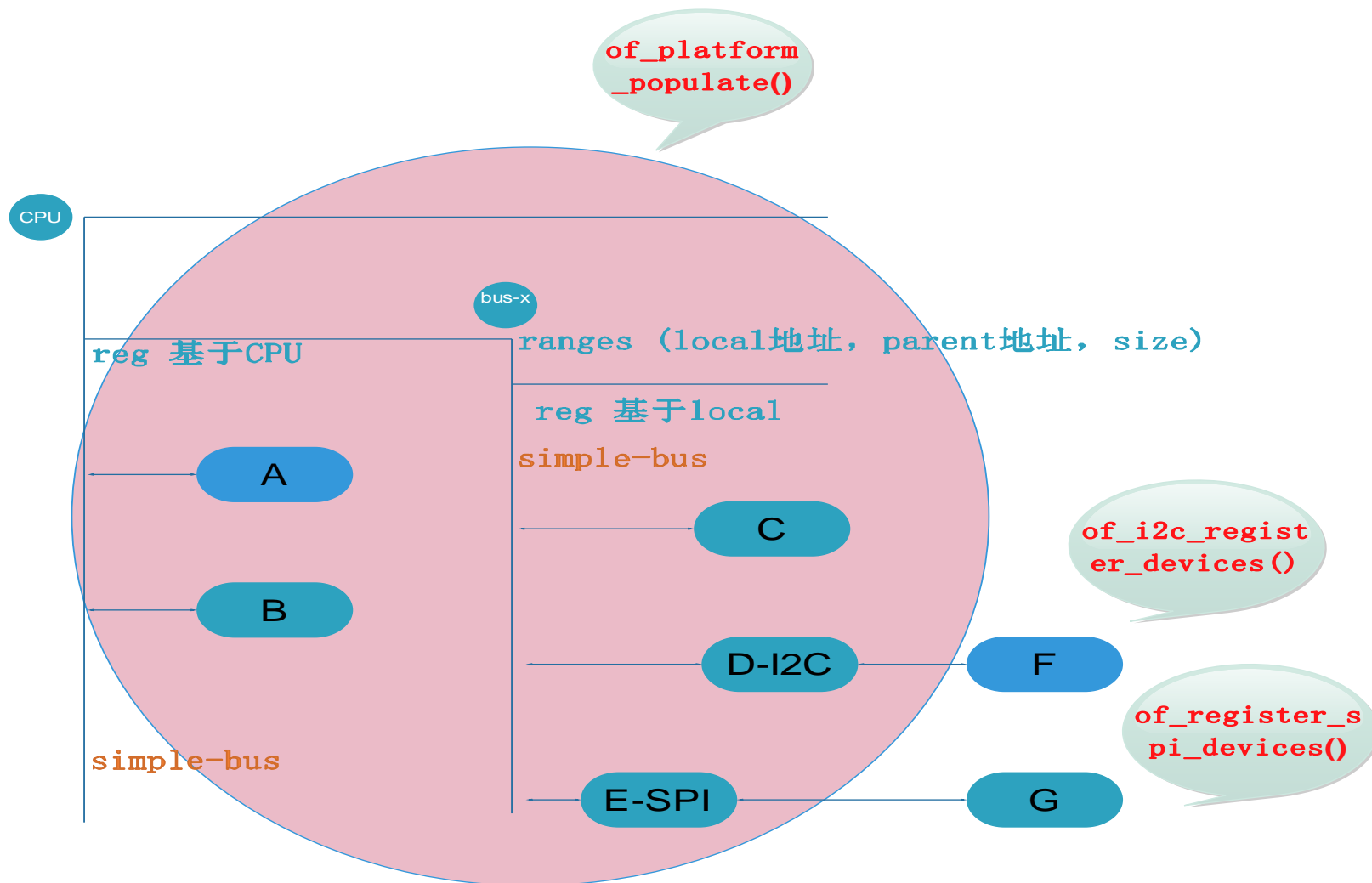
```
mpcore {  
    compatible = "simple-bus";  
    ranges = <0x00000000 0x19020000 0x00003000>;  
    #address-cells = <1>;  
    #size-cells = <1>;  
  
    scu@0000 {  
        compatible = "arm,cortex-a9-scu";  
        reg = <0x0000 0x100>;  
    };  
  
    timer@0200 {  
        compatible = "arm,cortex-a9-global-timer";  
        reg = <0x0200 0x100>;  
        interrupts = <GIC_PPI 11 IRQ_TYPE_LEVEL_HIGH>;  
        clocks = <8 clk_periph>;  
    };  
}
```

0 映射到 0x19020000



timer 映射到 0x19020200

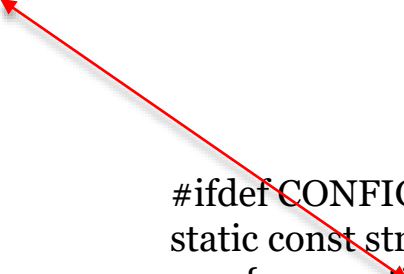
各级设备的展开



dts和driver的匹配

```
eth: eth@4,c00000 {  
    compatible = "davicom,dm9000";  
    ...  
};
```

```
#ifdef CONFIG_OF  
static const struct of_device_id dm9000_of_matches[] = {  
    { .compatible = "davicom,dm9000", },  
    { /* sentinel */ }  
};  
MODULE_DEVICE_TABLE(of, dm9000_of_matches);  
#endif  
  
static struct platform_driver dm9000_driver = {  
    .driver = {  
        .name = "dm9000",  
        .pm = &dm9000_drv_pm_ops,  
        .of_match_table = of_match_ptr(dm9000_of_matches),  
    },  
    .probe = dm9000_probe,  
    .remove = dm9000_drv_remove,  
};
```



总线match函数

```
static int platform_match(struct device *dev, struct device_driver *drv)
{
    struct platform_device *pdev = to_platform_device(dev);
    struct platform_driver *pdrv = to_platform_driver(drv);

    /* When driver_override is set, only bind to the matching driver */
    if (pdev->driver_override)
        return !strcmp(pdev->driver_override, drv->name);

    /* Attempt an OF style match first */
    if (of_driver_match_device(dev, drv))
        return 1;

    /* Then try ACPI style match */
    if (acpi_driver_match_device(dev, drv))
        return 1;

    /* Then try to match against the id table */
    if (pdrv->id_table)
        return platform_match_id(pdrv->id_table, pdev) != NULL;

    /* fall-back to driver name match */
    return (strcmp(pdev->name, drv->name) == 0);
}
```

硬件描述数据

drivers/dma/sun6i-dma.c

```
static struct sun6i_dma_config sun8i_a23_dma_cfg = {
    .nr_max_channels = 8,
    .nr_max_requests = 24,
    .nr_max_vchans = 37,
};

static struct of_device_id sun6i_dma_match[] = {
    { .compatible = "allwinner,sun6i-a31-dma", .data = &sun6i_a31_dma_cfg },
    { .compatible = "allwinner,sun8i-a23-dma", .data = &sun8i_a23_dma_cfg },
    { /* sentinel */ }
};

static int sun6i_dma_probe(struct platform_device *pdev)
{
    ...

    device = of_match_device(sun6i_dma_match, &pdev->dev);
    if (!device)
        return -ENODEV;
    sdc->cfg = device->data;
}
```

sun8i-a23.dtsi

```
dma: dma-controller@01c02000 {
    compatible = "allwinner,sun8i-a23-dma";
}
```

sun6i-a31.dtsi

```
dma: dma-controller@01c02000 {
    compatible = "allwinner,sun6i-a31-dma";
    reg = <0x01c02000 0x1000>;
}
```


reg(寄存器等)

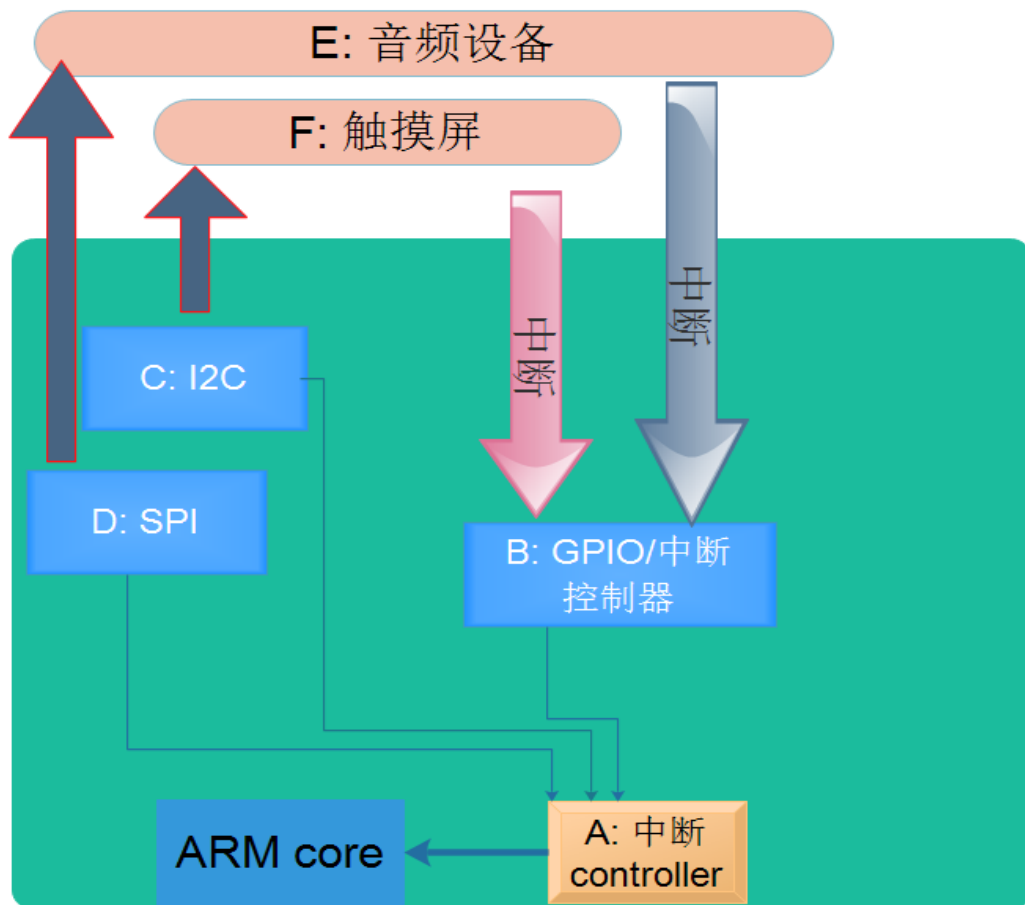
```
soc@0 {  
    #address-cells = <1>;  
    #size-cells = <1>;  
    compatible = "intel,ce4100-cp";  
    ranges;  
    ioapic1: interrupt-controller@fec00000 {  
        #interrupt-cells = <2>;  
        compatible = "intel,ce4100-ioapic";  
        interrupt-controller;  
        reg = <0xfec00000 0x1000>;  
    };  
};
```

```
cpus {  
    #address-cells = <1>;  
    #size-cells = <0>;  
    cpu@0 {  
        device_type = "cpu";  
        compatible = "intel,ce4100";  
        reg = <0>;  
        lapic = <&lapico>;  
    };  
};
```

```
i2c-controller@b,2 {  
    #address-cells = <2>;  
    #size-cells = <1>;  
    ...  
    i2c@0 {  
        reg = <0 0 0x100>;  
    };  
};
```

中断

- ✓ B、C、D的interrupt-parent是A;
- ✓ E、F的interrupt-parent是B



```
/ {  
    #address-cells = <1>;  
    #size-cells = <1>;  
  
    compatible = "ti,dra7xx";  
    interrupt-parent = <&gic>;  
    ...  
}
```

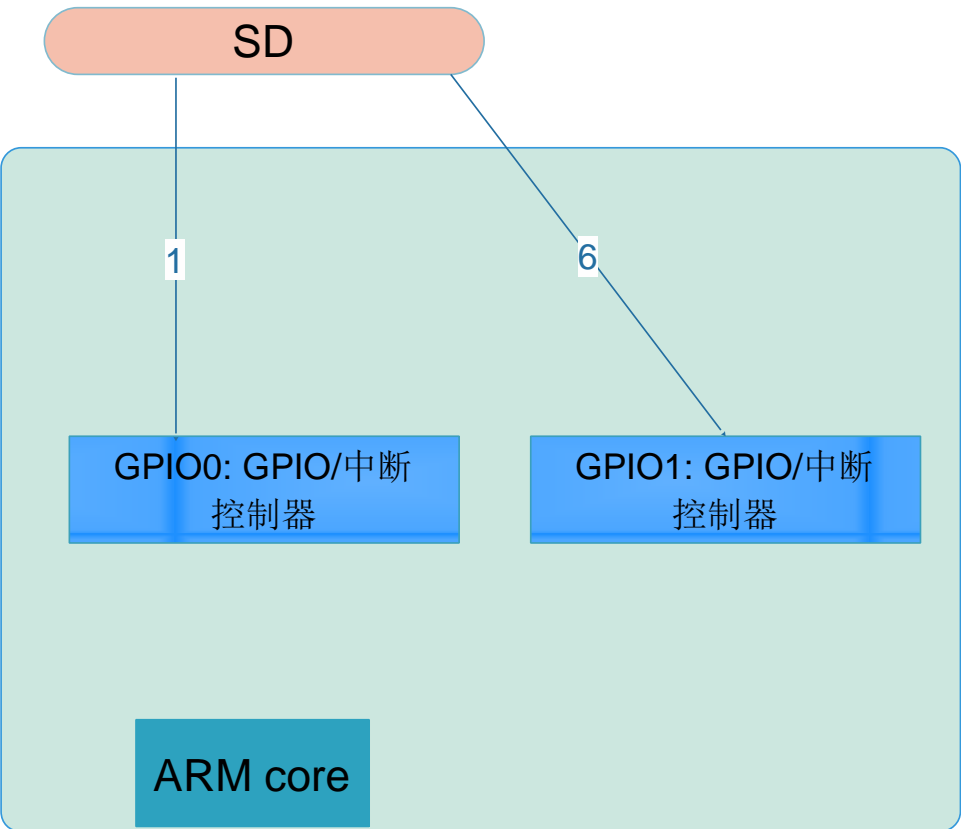
```
gpio1: gpio@4ae10000 {  
    compatible = "ti,omap4-gpio";  
    reg = <0x4ae10000 0x200>;  
    interrupts = <GIC_SPI 24  
    IRQ_TYPE_LEVEL_HIGH>;  
    gpio-controller;  
    #gpio-cells = <2>;  
    interrupt-controller;  
    #interrupt-cells = <2>;  
};
```

```
tps659038: tps659038@58 {  
    compatible = "ti,tps659038";  
    reg = <0x58>;  
    interrupt-parent = <&gpio1>;  
    interrupts = <0 IRQ_TYPE_LEVEL_LOW>;  
}
```

GPIO, DMA, CLK, pinctrl 描述方式

硬件

驱动



```
of_get_named_gpio(np, "cd-gpios", 0);  
of_get_named_gpio(np, "wp-gpios", 0);
```

dts

```
wp-gpios = <&gpio0 1  
           GPIO_ACTIVE_HIGH>;  
wp-gpios = <&gpio1 6  
           GPIO_ACTIVE_HIGH>;  
gpio0: gpio@e0050000 {  
    ...  
    gpio-controller;  
    #gpio-cells = <2>;  
    ngpios = <32>;  
    ...  
};  
gpio1: gpio@e0050080 {  
    ...  
    gpio-controller;  
    #gpio-cells = <2>;  
    ngpios = <32>;  
    ...  
};
```

DEMO 和 案例

加一个新的SoC和DTS

- ✓ 新建一个目录：arch/arm/mach-demosoc
- ✓ 加arch/arm/mach-demosoc/Kconfig、Makefile

```
config ARCH_DEMOSOC
```

```
bool "Linuxer demo soc(made by baohua)"
```

```
help
```

```
Support for Linuxer demo soc(made by baohua)
```

- ✓ 加arch/arm/mach-demosoc/common.c

```
static void __init demosoc_init_late(void)
```

```
{
```

```
}
```

```
#ifdef CONFIG_ARCH_DEMOSOC
```

```
static const char *const demosoc_dt_match[] __initconst = {
```

```
    "linuxer,demosoc",
```

```
    NULL
```

```
};
```

```
DT_MACHINE_START(DEMOSOC_DT, "Linuxer DEMOSOC (Flattened Device Tree)")
```

```
/* Maintainer: Barry Song <baohua@kernel.org> */
```

```
.init_late    = demosoc_init_late,
```

```
.dt_compat    = demosoc_dt_match,
```

```
MACHINE_END
```

```
#endif
```

加一个新的SoC和DTS(cont.)

- ✓ 加dtsi和dts

linuxer-demosoc.dtsi

linuxer-demosoc-evb.dts

- ✓ 把dts编译

修改:

arch/arm/boot/dts/Makefile

```
dtb-$(CONFIG_MACH_DEMOSOOC) += \  
    linuxer-demosoc-evb.dtb
```

- ✓ 反编译dtb

```
fdtdump linuxer-demosoc-evb.dtb
```

或者

```
dtc -I dtb -O dts ....
```

阅读与其他参考资料

[《Linux总线、设备、驱动模型》直播PPT分享](#)

[让天堂的归天堂，让尘土的归尘土——谈Linux的总线、设备、驱动模型](#)

[http://www.devicetree.org/Device Tree Usage](http://www.devicetree.org/Device_Tree_Usage)

<http://events.linuxfoundation.org/sites/events/files/slides/petazzoni-device-tree-dummies.pdf>

《Linux总线、设备、驱动模型》录播：

<http://edu.csdn.net/course/detail/5329>

谢谢!