device_node的生成

在kernel的初始阶段，几乎是什么都没有开始初始化之前。

arch/arm/kernel/setup.c

```
void __init setup_arch(char **cmdline_p);
void __init setup_arch(char **cmdline_p)
{
        const struct machine_desc *mdesc;

        setup_processor();
        mdesc = setup_machine_fdt(__atags_pointer);
        if (!mdesc)
                mdesc = setup_machine_tags(__atags_pointer, __machine_arch_type);
        machine_desc = mdesc;
        machine_name = mdesc->name;

        if (mdesc->reboot_mode != REBOOT_HARD)
                reboot_mode = mdesc->reboot_mode;

        init_mm.start_code = (unsigned long) _text;
        init_mm.end_code   = (unsigned long) _etext;
        init_mm.end_data   = (unsigned long) _edata;
        init_mm.brk        = (unsigned long) _end;

        /* populate cmd_line too for later use, preserving boot_command_line */
        strlcpy(cmd_line, boot_command_line, COMMAND_LINE_SIZE);
        *cmdline_p = cmd_line;

        parse_early_param();

        early_paging_init(mdesc, lookup_processor_type(read_cpuid_id()));
        setup_dma_zone(mdesc);
        sanity_check_meminfo();
        arm_memblock_init(mdesc);

        paging_init(mdesc);
        request_standard_resources(mdesc);

        if (mdesc->restart)
                arm_pm_restart = mdesc->restart;

        unflatten_device_tree();

        arm_dt_init_cpu_maps();
        psci_init();
#ifdef CONFIG_SMP
        if (is_smp()) {
                if (!mdesc->smp_init || !mdesc->smp_init()) {
                        if (psci_smp_available())
                                smp_set_ops(&psci_smp_ops);
                        else if (mdesc->smp)
                                smp_set_ops(mdesc->smp);
                }
                smp_init_cpus();
                smp_build_mpidr_hash();
        }
#endif

```

```
        if (!is_smp())
                hyp_mode_check();

        reserve_crashkernel();

#ifdef CONFIG_MULTI_IRQ_HANDLER
        handle_arch_irq = mdesc->handle_irq;
#endif

#ifdef CONFIG_VT
#if defined(CONFIG_VGA_CONSOLE)
        conswitchp = &vga_con;
#elif defined(CONFIG_DUMMY_CONSOLE)
        conswitchp = &dummy_con;
#endif
#endif

        if (mdesc->init_early)
                mdesc->init_early();
}
```

unflatten_device_tree()根据传入的dtb生成一棵device tree。device_node依次被挂在该device tree上。

device的生成

```
static int __init customize_machine(void)
{
        /*
         * customizes platform devices, or adds new ones
         * On DT based machines, we fall back to populating the
         * machine from the device tree, if no callback is provided,
         * otherwise we would always need an init_machine callback.
         */
        if (machine_desc->init_machine)
                machine_desc->init_machine();
#ifdef CONFIG_OF
        else
                of_platform_populate(NULL, of_default_bus_match_table,
                                        NULL, NULL);
#endif
        return 0;
}
arch_initcall(customize_machine);
```

arch/arm/mach-pegmatite/pegmatite.c

```
DT_MACHINE_START(PEGMATITE_DT, "Marvell Pegmatite (Device Tree)")
#ifdef CONFIG_SMP
        .smp            = smp_ops(pegmatite_smp_ops),
#endif
        .init_machine   =pegmatite_dt_init,
        .map_io         = pegmatite_map_io,
        .init_early     = pegmatite_init_early,
        .init_irq       = pegmatite_init_irq,
        .init_time      = pegmatite_timer_and_clk_init,
        .restart        = pegmatite_restart,
        .dt_compat      = pegmatite_dt_compat,
#ifdef CONFIG_ZONE_DMA
        .dma_zone_size  = SZ_256M,
#endif
MACHINE_END
```

```
static void __init pegmatite_dt_init(void)
{
        /* Add devices not supported by device tree */
        platform_add_devices(platform_devices, ARRAY_SIZE(platform_devices));

of_platform_populate(NULL, of_default_bus_match_table, NULL, NULL);
}
```

这里platform_add_devices()用于生成不支持device tree的“device”，而of_platform_populate（ ）则根据unflatten_device_tree()生成的device tree,递归的enumerate device_node，然后依次create device node。

arch_initcall(customize_machine);

#define arch_initcall(fn)   __define_initcall(fn, 3)

也就是同built-in driver的initialization一样都是在kernel startup的后期（整个kernel几乎都已经就绪）

start_kernel() ---> rest_init() ---> kernel_init() ---> kernel_init_freeable() ---> do_basic_setup()

device creation 在 initcall_level 3.

driver的初始化
#define device_initcall(fn)   __define_initcall(fn, 6)
built-in driver initialziation在initcall_level 6.

而自动载入module的initialization则是在udevd service启动以后udevd扫描sysfs filesystem (也意味着sysfs file system的mount必须在udevd的运行之前)，根据hardware info生成hotplug event,udev根

据这些event，生成对应的硬件设备文件，并载入module。