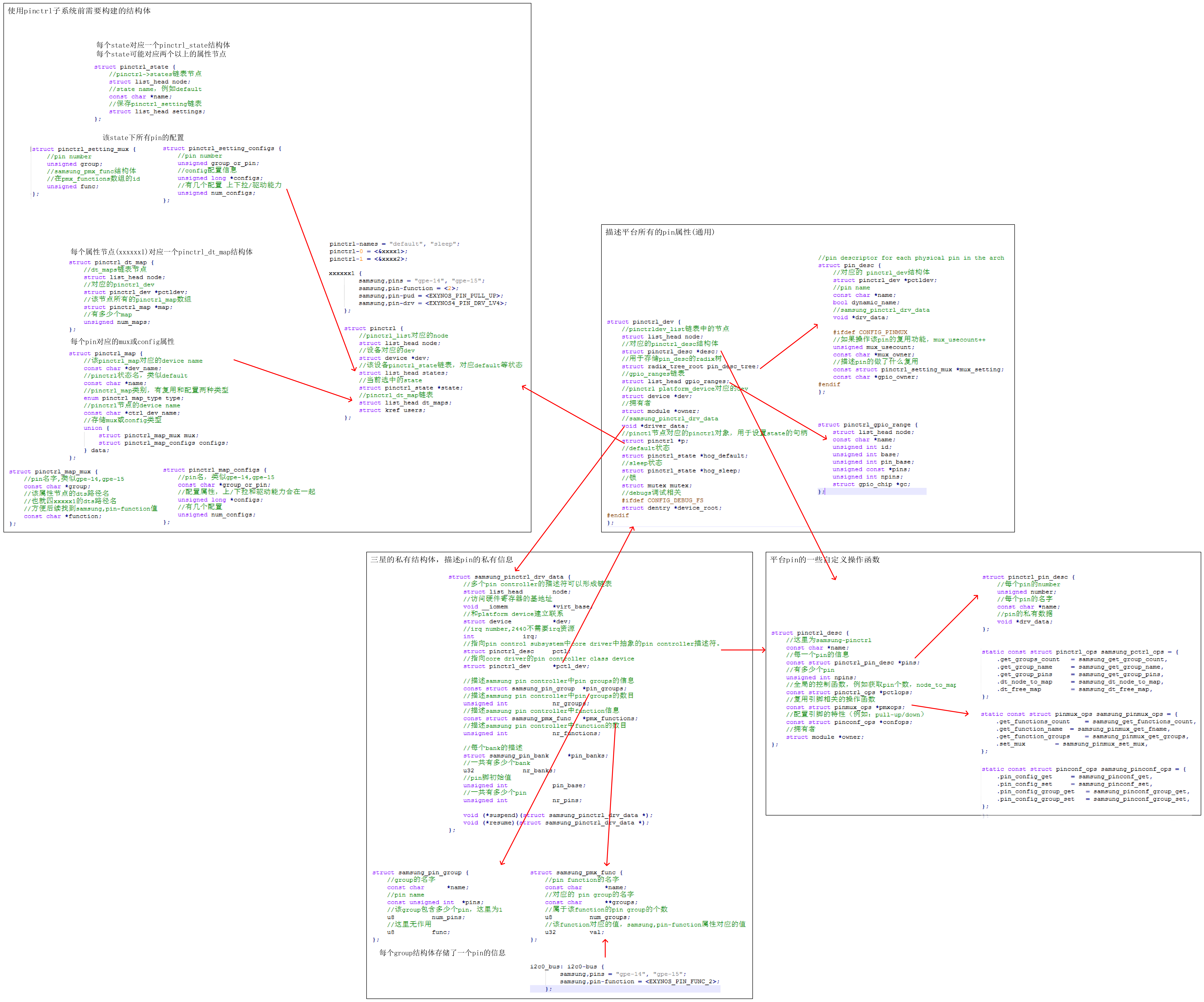

使用pinctrl子系统前需要构建的结构体

每个state对应一个pinctrl_state结构体
每个state可能对应两个以上的属性节点

```c
struct pinctrl_state {
    //pinctrl->states链表节点
    struct list_head node;
    //state name，例如default
    const char *name;
    //保存pinctrl_setting链表
    struct list_head settings;
};
```

该state下所有pin的配置

```c
struct pinctrl_setting_mux {
    //pin number
    unsigned group;
    //samsung_pmx_func结构体
    //在pmx_functions数组的id
    unsigned func;
};
```

```c
struct pinctrl_setting_configs {
    //pin number
    unsigned group_or_pin;
    //config配置信息
    unsigned long *configs;
    //有几个配置 上下拉/驱动能力
    unsigned num_configs;
};
```

每个属性节点(xxxxxx1)对应一个pinctrl_dt_map结构体

```c
struct pinctrl_dt_map {
    //dt_maps链表节点
    struct list_head node;
    //对应的pinctrl_dev
    struct pinctrl_dev *pctldev;
    //该节点所有的pinctrl_map数组
    struct pinctrl_map *map;
    //有多少个map
    unsigned num_maps;
};
```

每个pin对应的mux或config属性

```c
struct pinctrl_map {
    //该pinctrl_map对应的device name
    const char *dev_name;
    //pinctrl状态名，类似default
    const char *name;
    //pinctrl_map类别，有复用和配置两种类型
    enum pinctrl_map_type type;
    //pinctrl节点的device name
    const char *ctrl_dev_name;
    //存储mux或config类型
    union {
        struct pinctrl_map_mux mux;
        struct pinctrl_map_configs configs;
    } data;
};
```

```c
struct pinctrl_map_mux {
    //pin名字,类似gpe-14,gpe-15
    const char *group;
    //该属性节点的dts路径名
    //也就四xxxxx1的dts路径名
    //方便后续找到samsung,pin-function值
    const char *function;
};
```

```c
struct pinctrl_map_configs {
    //pin名，类似gpe-14,gpe-15
    const char *group_or_pin;
    //配置属性，上/下拉和驱动能力合在一起
    unsigned long *configs;
    //有几个配置
    unsigned num_configs;
};
```

```
pinctrl-names = "default", "sleep";
pinctrl-0 = <&xxxx1>;
pinctrl-1 = <&xxxx2>;

xxxxxx1 {
        samsung,pins = "gpe-14", "gpe-15";
        samsung,pin-function = <2>;
        samsung,pin-pud = <EXYNOS_PIN_PULL_UP>;
        samsung,pin-drv = <EXYNOS4_PIN_DRV_LV4>;
};
```

```c
struct pinctrl {
    //pinctrl_list对应的node
    struct list_head node;
    //设备对应的dev
    struct device *dev;
    //该设备pinctrl_state链表，对应default等状态
    struct list_head states;
    //当前选中的state
    struct pinctrl_state *state;
    //pinctrl_dt_map链表
    struct list_head dt_maps;
    struct kref users;
};
```

描述平台所有的pin属性(通用)

```c
struct pinctrl_dev {
    //pinctrldev_list链表中的节点
    struct list_head node;
    //对应的pinctrl_desc结构体
    struct pinctrl_desc *desc;
    //用于存储pin_desc的radix树
    struct radix_tree_root pin_desc_tree;
    //gpio_ranges链表
    struct list_head gpio_ranges;
    //pinctrl platform_device对应的dev
    struct device *dev;
    //拥有者
    struct module *owner;
    //samsung_pinctrl_drv_data
    void *driver_data;
    //pinct1节点对应的pinctrl对象，用于设置state的句柄
    struct pinctrl *p;
    //default状态
    struct pinctrl_state *hog_default;
    //sleep状态
    struct pinctrl_state *hog_sleep;
    //锁
    struct mutex mutex;
    //debugs调试相关
    #ifdef CONFIG_DEBUG_FS
    struct dentry *device_root;
#endif
};
```

```c
//pin descriptor for each physical pin in the arch
struct pin_desc {
    //对应的 pinctrl_dev结构体
    struct pinctrl_dev *pctldev;
    //pin name
    const char *name;
    bool dynamic_name;
    //samsung_pinctrl_drv_data
    void *drv_data;

    #ifdef CONFIG_PINMUX
    //如果操作该pin的复用功能，mux_usecount++
    unsigned mux_usecount;
    const char *mux_owner;
    //描述pin的做了什么复用
    const struct pinctrl_setting_mux *mux_setting;
    const char *gpio_owner;
#endif
};
```

```c
struct pinctrl_gpio_range {
    struct list_head node;
    const char *name;
    unsigned int id;
    unsigned int base;
    unsigned int pin_base;
    unsigned const *pins;
    unsigned int npins;
    struct gpio_chip *gc;
};
```

三星的私有结构体，描述pin的私有信息

```c
struct samsung_pinctrl_drv_data {
    //多个pin controller的描述符可以形成链表
    struct list_head        node;
    //访问硬件寄存器的基地址
    void __iomem            *virt_base;
    //和platform device建立联系
    struct device           *dev;
    //irq number,2440不需要irq资源
    int                 irq;
    //指向pin control subsystem中core driver中抽象的pin controller描述符。
    struct pinctrl_desc     pctl;
    //指向core driver的pin controller class device
    struct pinctrl_dev      *pctl_dev;

    //描述samsung pin controller中pin groups的信息
    const struct samsung_pin_group  *pin_groups;
    //描述samsung pin controller中pin groups的数目
    unsigned int            nr_groups;
    //描述samsung pin controller中function信息
    const struct samsung_pmx_func   *pmx_functions;
    //描述samsung pin controller中function的数目
    unsigned int            nr_functions;

    //每个bank的描述
    struct samsung_pin_bank     *pin_banks;
    //一共有多少个bank
    u32                 nr_banks;
    //pin脚初始值
    unsigned int            pin_base;
    //一共有多少个pin
    unsigned int            nr_pins;

    void (*suspend)(struct samsung_pinctrl_drv_data *);
    void (*resume)(struct samsung_pinctrl_drv_data *);
};
```

```c
struct samsung_pin_group {
    //group的名字
    const char      *name;
    //pin name
    const unsigned int  *pins;
    //该group包含多少个pin，这里为1
    u8          num_pins;
    //这里无作用
    u8          func;
};
```

每个group结构体存储了一个pin的信息

```c
struct samsung_pmx_func {
    //pin function的名字
    const char      *name;
    //对应的 pin group的名字
    const char      **groups;
    //属于该function的pin group的个数
    u8          num_groups;
    //该function对应的值，samsung,pin-function属性对应的值
    u32         val;
};
```

```
i2c0_bus: i2c0-bus {
        samsung,pins = "gpe-14", "gpe-15";
        samsung,pin-function = <EXYNOS_PIN_FUNC_2>;
    };
```

平台pin的一些自定义操作函数

```c
struct pinctrl_desc {
    //这里为samsung-pinctrl
    const char *name;
    //每一个pin的信息
    const struct pinctrl_pin_desc *pins;
    //有多少个pin
    unsigned int npins;
    //全局的控制函数，例如获取pin个数，node_to_map
    const struct pinctrl_ops *pctlops;
    //复用引脚相关的操作函数
    const struct pinmux_ops *pmxops;
    //配置引脚的特性（例如：pull-up/down）
    const struct pinconf_ops *confops;
    //拥有者
    struct module *owner;
};
```

```c
struct pinctrl_pin_desc {
    //每个pin的number
    unsigned number;
    //每个pin的名字
    const char *name;
    //pin的私有数据
    void *drv_data;
};
```

```c
static const struct pinctrl_ops samsung_pctrl_ops = {
    .get_groups_count   = samsung_get_group_count,
    .get_group_name     = samsung_get_group_name,
    .get_group_pins     = samsung_get_group_pins,
    .dt_node_to_map     = samsung_dt_node_to_map,
    .dt_free_map        = samsung_dt_free_map,
};
```

```c
static const struct pinmux_ops samsung_pinmux_ops = {
    .get_functions_count    = samsung_get_functions_count,
    .get_function_name  = samsung_pinmux_get_fname,
    .get_function_groups    = samsung_pinmux_get_groups,
    .set_mux        = samsung_pinmux_set_mux,
};
```

```c
static const struct pinconf_ops samsung_pinconf_ops = {
    .pin_config_get     = samsung_pinconf_get,
    .pin_config_set     = samsung_pinconf_set,
    .pin_config_group_get   = samsung_pinconf_group_get,
    .pin_config_group_set   = samsung_pinconf_group_set,
};
```