G2 LSP中gpio device node的描述如下

```
        gpio0: gpio@d4019000 {
                compatible = "marvell,peg-gpio";
                #address-cells = <2>;
                #size-cells = <2>;
                reg = <0 0xd4019000 0 0x1000>;
                gpio-controller;
                #gpio-cells = <2>;
                interrupts = <0 36 4>, <0 119 4>, <0 120 4>, <0 121 4>, <0 122 4>, <0 123 4>, <0 221 4>, <0 222 4>;
                interrupt-names = "gpio_mux";
                interrupt-controller;
                #interrupt-cells = <2>;
                clocks = <&apbus_apb_clkgate>;
                ranges;

                gcb0: gpio@d4019000 {
                        reg = <0 0xd4019000 0 0x4>;
                };

                gcb1: gpio@d4019100 {
                        reg = <0 0xd4019100 0 0x4>;
                };

                gcb2: gpio@d4019200 {
                        reg = <0 0xd4019200 0 0x4>;
                };

                gcb3: gpio@d4019300 {
                        reg = <0 0xd4019300 0 0x4>;
                };

                gcb4: gpio@d4019400 {
                        reg = <0 0xd4019400 0 0x4>;
                };

                gcb5: gpio@d4019500 {
                        reg = <0 0xd4019500 0 0x4>;
                };

                gcb6: gpio@d4019600 {
                        reg = <0 0xd4019600 0 0x4>;
                };

                gcb7: gpio@d4019700 {
                        reg = <0 0xd4019700 0 0x4>;
                };
        };
```

但其实下面一段

```
                gcb0: gpio@d4019000 {
                        reg = <0 0xd4019000 0 0x4>;
                };

                gcb1: gpio@d4019100 {
                        reg = <0 0xd4019100 0 0x4>;
                };

                gcb2: gpio@d4019200 {
                        reg = <0 0xd4019200 0 0x4>;
                };

                gcb3: gpio@d4019300 {
                        reg = <0 0xd4019300 0 0x4>;
                };

                gcb4: gpio@d4019400 {
                        reg = <0 0xd4019400 0 0x4>;
                };

                gcb5: gpio@d4019500 {
                        reg = <0 0xd4019500 0 0x4>;
                };

                gcb6: gpio@d4019600 {
                        reg = <0 0xd4019600 0 0x4>;
                };

                gcb7: gpio@d4019700 {
                        reg = <0 0xd4019700 0 0x4>;
                };
```

是没用的。

1. reg = <0 0xd4019000 0 0x4>

只是gpio0的PLR register，怎么能代表整个gcb0 (gpio controller block 0)

2. gpio-pxa.c也没有使用到这些信息。

创建gcb0 to gcb7的gpio chip data structure如下：

in drivers/gpio/gpio-pxa.c

```
1.  static int pxa_init_gpio_chip(int gpio_end,
2.                                   int (*set_wake)(unsigned int, unsigned in
    t))
3.  {
4.          int i, gpio, nbanks = gpio_to_bank(gpio_end) + 1;
5.          struct pxa_gpio_chip *chips;
6.
7.          chips = kzalloc(nbanks * sizeof(struct pxa_gpio_chip), GFP_KERNEL);
8.          if (chips == NULL) {
9.                  pr_err("%s: failed to allocate GPIO chips\n", __func__);
10.                 return -ENOMEM;
11.         }
12.
13.         for (i = 0, gpio = 0; i < nbanks; i++, gpio += 32) {
14.                 struct gpio_chip *c = &chips[i].chip;
15.
16.                 sprintf(chips[i].label, "gpio-%d", i);
17.                 chips[i].regbase = gpio_reg_base + BANK_OFF(i);
18.                 chips[i].set_wake = set_wake;
19.
20.                 c->base  = gpio;
21.                 c->label = chips[i].label;
22.
23.                 c->direction_input  = pxa_gpio_direction_input;
24.                 c->direction_output = pxa_gpio_direction_output;
25.                 c->get = pxa_gpio_get;
26.                 c->set = pxa_gpio_set;
27.                 c->to_irq = pxa_gpio_to_irq;
28. #ifdef CONFIG_OF_GPIO
29.                 if (gpio_is_peg_type(gpio_type))
30.                         chips[i].irq = irq_of_parse_and_map(pxa_gpio_of_node, i);
31.                 c->of_node = pxa_gpio_of_node;
32.                 c->of_xlate = pxa_gpio_of_xlate;
33.                 c->of_gpio_n_cells = 2;
34. #endif
35.
36.                 /* number of GPIOs on last bank may be less than 32 */
37.                 c->ngpio = (gpio + 31 > gpio_end) ? (gpio_end - gpio + 1) : 32;
38.                 gpiochip_add(c);
39.         }
40.         pxa_gpio_chips = chips;
41.         return 0;
42. }
```

即gpio chip 是根据pxa_last_gpio来的。

in pxa_gpio_probe()

```
        /* Initialize GPIO chips */
        pxa_init_gpio_chip(pxa_last_gpio, info ? info->gpio_set_wake : NULL);
```

pxa_last_gpio的设定是在pxa_gpio_probe_dt()中

```
static int pxa_gpio_probe_dt(struct platform_device *pdev)
{
        int ret = 0, nr_gpios;
        struct device_node *np = pdev->dev.of_node;
        const struct of_device_id *of_id =
                                of_match_device(pxa_gpio_dt_ids, &pdev->dev);
        const struct pxa_gpio_id *gpio_id;

        if (!of_id || !of_id->data) {
                dev_err(&pdev->dev, "Failed to find gpio controller\n");
                return -EFAULT;
        }
        gpio_id = of_id->data;
        gpio_type = gpio_id->type;

        nr_gpios = gpio_id->gpio_nums;
        pxa_last_gpio = nr_gpios - 1;

        irq_base = irq_alloc_descs(-1, 0, nr_gpios, 0);
        if (irq_base < 0) {
                dev_err(&pdev->dev, "Failed to allocate IRQ numbers\n");
                ret = irq_base;
                goto err;
        }
        domain = irq_domain_add_legacy(np, nr_gpios, irq_base, 0,
                                       &pxa_irq_domain_ops, NULL);
        pxa_gpio_of_node = np;
        return 0;
err:
        iounmap(gpio_reg_base);
        return ret;
}
```

pxa_last_gpio <-- nr_gpio <-- gpio_id->gpio_nums <-- of_id->data

```
static const struct of_device_id pxa_gpio_dt_ids[] = {
        { .compatible = "intel,pxa25x-gpio",    .data = &pxa25x_id, },
        { .compatible = "intel,pxa26x-gpio",    .data = &pxa26x_id, },
        { .compatible = "intel,pxa27x-gpio",    .data = &pxa27x_id, },
        { .compatible = "intel,pxa3xx-gpio",    .data = &pxa3xx_id, },
        { .compatible = "marvell,pxa93x-gpio",  .data = &pxa93x_id, },
        { .compatible = "marvell,mmp-gpio",     .data = &mmp_id, },
        { .compatible = "marvell,mmp2-gpio",    .data = &mmp2_id, },
        { .compatible = "marvell,peg-gpio", .data = &peg_id, },
        {}
};
```

of_id->data = &peg_id;

```
static struct pxa_gpio_id peg_id = {
        .type           = PEG_GPIO,
        .gpio_nums      = 256,
};
```

即pxa_last_gpio = 255.

一个gpio bank manage 32 gpio pins，所以有8个gpio bank, from gpio0 to gpio7.