# Linux驱动\_pinctrl子系统

## 1.pinctrl子系统初始化

### 1.1 dts树中关于pinctrl的定义

pinctrl@56000000 {

compatible = "samsung,s3c2440-pinctrl";

};

pinctrl\_0: pinctrl@56000000 {

reg = <0x56000000 0x1000>;

wakeup-interrupt-controller {

compatible = "samsung,s3c2410-wakeup-eint";

interrupts = <0 0 0 3>,

<0 0 1 3>,

<0 0 2 3>,

<0 0 3 3>,

<0 0 4 4>,

<0 0 5 4>;

};

};

&pinctrl\_0 {

/\*

\* Pin banks

\*/

gpa: gpa {

gpio-controller;

#gpio-cells = <2>;

};

gpb: gpb {

gpio-controller;

#gpio-cells = <2>;

};

gpc: gpc {

gpio-controller;

#gpio-cells = <2>;

};

gpd: gpd {

gpio-controller;

#gpio-cells = <2>;

};

gpe: gpe {

gpio-controller;

#gpio-cells = <2>;

};

gpf: gpf {

gpio-controller;

#gpio-cells = <2>;

interrupt-controller;

#interrupt-cells = <2>;

};

gpg: gpg {

gpio-controller;

#gpio-cells = <2>;

interrupt-controller;

#interrupt-cells = <2>;

};

gph: gph {

gpio-controller;

#gpio-cells = <2>;

};

gpj: gpj {

gpio-controller;

#gpio-cells = <2>;

};

/\*

\* Pin groups

\*/

uart0\_data: uart0-data {

samsung,pins = "gph-0", "gph-1";

samsung,pin-function = <2>;

};

i2c0\_bus: i2c0-bus {

samsung,pins = "gpe-14", "gpe-15";

samsung,pin-function = <EXYNOS\_PIN\_FUNC\_2>;

};

nand\_pinctrl: nand\_pinctrl {

samsung,pins = "gpa-17", "gpa-18", "gpa-19",

"gpa-20", "gpa-22";

samsung,pin-function = <1>;

};

};

### 1.2 2440 pinctrl子系统初始化

samsung,s3c2440-pinctrl的注册代码如下：

static const struct of\_device\_id samsung\_pinctrl\_dt\_match**[]** **=** **{**

**{** **.**compatible **=** "samsung,s3c2440-pinctrl"**,**

**.**data **=** s3c2440\_pin\_ctrl **},**

**};**

static struct platform\_driver samsung\_pinctrl\_driver **=** **{**

**.**probe **=** samsung\_pinctrl\_probe**,**

**.**driver **=** **{**

**.**name **=** "samsung-pinctrl"**,**

**.**of\_match\_table **=** samsung\_pinctrl\_dt\_match**,**

**.**suppress\_bind\_attrs **=** **true,**

**},**

**};**

static int \_\_init samsung\_pinctrl\_drv\_register**(**void**)**

**{**

register\_syscore\_ops**(&**samsung\_pinctrl\_syscore\_ops**);**

**return** platform\_driver\_register**(&**samsung\_pinctrl\_driver**);**

**}**

匹配后samsung\_pinctrl\_probe函数调用。

static int samsung\_pinctrl\_probe**(**struct platform\_device **\***pdev**)**

**{**

struct samsung\_pinctrl\_drv\_data **\***drvdata**;**

const struct samsung\_pin\_ctrl **\***ctrl**;**

struct device **\***dev **=** **&**pdev**->**dev**;**

struct resource **\***res**;**

int ret**;**

//分配samsung\_pinctrl\_drv\_data结构体空间

drvdata **=** devm\_kzalloc**(**dev**,** **sizeof(\***drvdata**),** GFP\_KERNEL**);**

//初始化samsung\_pinctrl\_drv\_data

ctrl **=** samsung\_pinctrl\_get\_soc\_data**(**drvdata**,** pdev**);**

//drvdata关联dev

drvdata**->**dev **=** dev**;**

//pinctrl操作对应的虚拟地址

res **=** platform\_get\_resource**(**pdev**,** IORESOURCE\_MEM**,** 0**);**samsung\_pinctrl\_parse\_dt

drvdata**->**virt\_base **=** devm\_ioremap\_resource**(&**pdev**->**dev**,** res**);**

//获取pinctrl对应的irq号，这里没有

res **=** platform\_get\_resource**(**pdev**,** IORESOURCE\_IRQ**,** 0**);**

**if** **(**res**)**

drvdata**->**irq **=** res**->**start**;**

//注册到gpio子系统中

ret **=** samsung\_gpiolib\_register**(**pdev**,** drvdata**);**

//注册到pinctrl子系统中

ret **=** samsung\_pinctrl\_register**(**pdev**,** drvdata**);**

//gpio相关中断注册到中断子系统中

**if** **(**ctrl**->**eint\_gpio\_init**)**

ctrl**->**eint\_gpio\_init**(**drvdata**);**

**if** **(**ctrl**->**eint\_wkup\_init**)**

ctrl**->**eint\_wkup\_init**(**drvdata**);**

//设置device的私有数据

platform\_set\_drvdata**(**pdev**,** drvdata**);**

/\* Add to the global list \*/

list\_add\_tail**(&**drvdata**->**node**,** **&**drvdata\_list**);**

**return** 0**;**

**}**

probe函数对pinctrl、中断、gpio子系统都进行了操作。struct pinctrl\_desc和struct pinctrl\_dev 都是pin control subsystem中core driver的概念。各个具体硬件的pin controller可能会各不相同，但是可以抽取其共同的部分来形成一个HW independent的数据结构，这个数据就是pin controller描述符。整个初始化过程的核心思想就是low level的driver定义一个pinctrl\_desc，设定pin相关的定义和callback函数，注册到pin control subsystem中。我们用引脚描述符（pin descriptor）来描述一个pin。在pin control subsystem中，struct pinctrl\_pin\_desc用来描述一个可以控制的引脚，我们称之引脚描述符。

#### 1.2.1 samsung\_pinctrl\_drv\_data

samsung\_pinctrl\_drv\_data结构体定义如下：

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 **\*);**

**};**

devm\_kzalloc函数是为struct samsung\_pinctrl\_drv\_data数据结构分配内存。每当driver probe一个具体的device实例的时候，都需要建立一些私有的数据结构来保存该device的一些具体的硬件信息（本场景中，这个数据结构就是struct samsung\_pinctrl\_drv\_data）。在过去，驱动工程师多半使用kmalloc或者kzalloc来分配内存，但这会带来一些潜在的问题。例如：在初始化过程中，有各种各样可能的失败情况，这时候就依靠driver工程师小心的撰写代码，释放之前分配的内存。当然，初始化过程中，除了memory，driver会为probe的device分配各种资源，例如IRQ号，io memory map、DMA等等。当初始化需要管理这么多的资源分配和释放的时候，很多驱动程序都出现了资源管理的issue。而且，由于这些issue是异常路径上的issue，不是那么容易测试出来，更加重了解决这个issue的必要性。内核解决这个问题的模式（所谓解决一类问题的设计方法就叫做设计模式）是Devres，即device resource management软件模块。更细节的内容就不介绍了，其核心思想就是资源是设备的资源，那么资源的管理归于device，也就是说不需要driver过多的参与。当device和driver detach的时候，device会自动的释放其所有的资源。

#### 1.2.2 samsung\_pinctrl\_get\_soc\_data

分配了struct samsung\_pinctrl\_drv\_data数据结构的内存，当然下一步就是初始化这个数据结构了。

static const struct samsung\_pin\_ctrl **\*** samsung\_pinctrl\_get\_soc\_data**(**struct samsung\_pinctrl\_drv\_data **\***d**,**

struct platform\_device **\***pdev**)**

**{**

int id**;**

const struct of\_device\_id **\***match**;**

struct device\_node **\***node **=** pdev**->**dev**.**of\_node**;**

struct device\_node **\***np**;**

const struct samsung\_pin\_bank\_data **\***bdata**;**

const struct samsung\_pin\_ctrl **\***ctrl**;**

struct samsung\_pin\_bank **\***bank**;**

int i**;**

//获取device node对应的pinctrl alias id号，这里为0

id **=** of\_alias\_get\_id**(**node**,** "pinctrl"**);**

//match为如下数据

//{ .compatible = "samsung,s3c2440-pinctrl",

// .data = s3c2440\_pin\_ctrl }

match **=** of\_match\_node**(**samsung\_pinctrl\_dt\_match**,** node**);**

//获取对应的samsung\_pin\_ctrl s3c2440\_pin\_ctrl数组中的第一组数据

//struct samsung\_pin\_ctrl描述的具体samsung pin controller硬件相关的信息，

//比如说：pin bank的信息，不是所有的pin controller都是分bank的，

//因此pin bank的信息只能封装在low level的samsung pin controller driver中。

ctrl **=** **(**struct samsung\_pin\_ctrl **\*)**match**->**data **+** id**;**

//samsung\_pinctrl\_drv\_data设置对应的变量和函数

d**->**suspend **=** ctrl**->**suspend**;**

d**->**resume **=** ctrl**->**resume**;**

d**->**nr\_banks **=** ctrl**->**nr\_banks**;**

d**->**pin\_banks **=** devm\_kcalloc**(&**pdev**->**dev**,** d**->**nr\_banks**,sizeof(\***d**->**pin\_banks**),** GFP\_KERNEL**);**

//设置pin\_banks信息

bank **=** d**->**pin\_banks**;**

bdata **=** ctrl**->**pin\_banks**;**

//为每个bank设置基础数据

**for** **(**i **=** 0**;** i **<** ctrl**->**nr\_banks**;** **++**i**,** **++**bdata**,** **++**bank**)** **{**

//type按寄存器操作的bit数分类，

//例如BANK A只需一个bit完成pin功能的设置,其他的需要两个bit

bank**->**type **=** bdata**->**type**;**

//寄存器偏移

bank**->**pctl\_offset **=** bdata**->**pctl\_offset**;**

//每个BANK有多少个pin脚

bank**->**nr\_pins **=** bdata**->**nr\_pins**;**

//gpf gpg具备外部中断能力

bank**->**eint\_func **=** bdata**->**eint\_func**;**

//中断类型，这里为EINT\_TYPE\_WKUP

bank**->**eint\_type **=** bdata**->**eint\_type**;**

//那些gpio具备中断能力

bank**->**eint\_mask **=** bdata**->**eint\_mask**;**

//从第几个gpio具有中断能力

bank**->**eint\_offset **=** bdata**->**eint\_offset**;**

//gpio bank的名字,例如GPA

bank**->**name **=** bdata**->**name**;**

spin\_lock\_init**(&**bank**->**slock**);**

//根据bank能找到对应的samsung\_pinctrl\_drv\_data

bank**->**drvdata **=** d**;**

//设置bank的gpio数量偏移

bank**->**pin\_base **=** d**->**nr\_pins**;**

//更新整个2440具有的gpio数量

d**->**nr\_pins **+=** bank**->**nr\_pins**;**

**}**

//遍历device node下的gpio-controller子节点

for\_each\_child\_of\_node**(**node**,** np**)** **{**

//判断子节点是否是gpio-controller

**if** **(!**of\_find\_property**(**np**,** "gpio-controller"**,** **NULL))**

**continue;**

//轮询所有的samsung\_pin\_bank,名字相同的bank和gpio-controller做一个关联

bank **=** d**->**pin\_banks**;**

**for** **(**i **=** 0**;** i **<** d**->**nr\_banks**;** **++**i**,** **++**bank**)** **{**

**if** **(!**strcmp**(**bank**->**name**,** np**->**name**))** **{**

bank**->**of\_node **=** np**;**

**break;**

**}**

**}**

**}**

//更新全局变量pin\_base

d**->**pin\_base **=** pin\_base**;**

pin\_base **+=** d**->**nr\_pins**;**

//返回s3c2440\_pin\_ctrl

**return** ctrl**;**

**}**

struct samsung\_pinctrl\_drv\_data和硬件相关数据来自s3c2440\_pin\_ctrl：

struct samsung\_pin\_ctrl **{**

//定义具体的pin bank信息,例如操作寄存器/bank名称/有多少pin等

const struct samsung\_pin\_bank\_data **\***pin\_banks**;**

//number of pin bank

u32 nr\_banks**;**

//和中断相关的代码

int **(\***eint\_gpio\_init**)(**struct samsung\_pinctrl\_drv\_data **\*);**

int **(\***eint\_wkup\_init**)(**struct samsung\_pinctrl\_drv\_data **\*);**

void **(\***suspend**)(**struct samsung\_pinctrl\_drv\_data **\*);**

void **(\***resume**)(**struct samsung\_pinctrl\_drv\_data **\*);**

**};**

static const struct samsung\_pin\_bank\_data s3c2440\_pin\_banks**[]** \_\_initconst **=** **{**

PIN\_BANK\_A**(**25**,** 0x000**,** "gpa"**),**

PIN\_BANK\_2BIT**(**11**,** 0x010**,** "gpb"**),**

PIN\_BANK\_2BIT**(**16**,** 0x020**,** "gpc"**),**

PIN\_BANK\_2BIT**(**16**,** 0x030**,** "gpd"**),**

PIN\_BANK\_2BIT**(**16**,** 0x040**,** "gpe"**),**

PIN\_BANK\_2BIT\_EINTW**(**8**,** 0x050**,** "gpf"**,** 0**,** 0xff**),**

PIN\_BANK\_2BIT\_EINTW**(**16**,** 0x060**,** "gpg"**,** 8**,** 0xffff00**),**

PIN\_BANK\_2BIT**(**11**,** 0x070**,** "gph"**),**

PIN\_BANK\_2BIT**(**13**,** 0x0d0**,** "gpj"**),**

**};**

const struct samsung\_pin\_ctrl s3c2440\_pin\_ctrl**[]** \_\_initconst **=** **{**

**{**

**.**pin\_banks **=** s3c2440\_pin\_banks**,**

**.**nr\_banks **=** ARRAY\_SIZE**(**s3c2440\_pin\_banks**),**

**.**eint\_wkup\_init **=** s3c24xx\_eint\_init**,**

**},**

**};**

#### 1.2.3 samsung\_gpiolib\_register

samsung\_gpiolib\_register将gpio信息注册到了gpio子系统中。

static const struct gpio\_chip samsung\_gpiolib\_chip **=** **{**

**.**request **=** gpiochip\_generic\_request**,**

**.**free **=** gpiochip\_generic\_free**,**

**.**set **=** samsung\_gpio\_set**,**

**.**get **=** samsung\_gpio\_get**,**

**.**direction\_input **=** samsung\_gpio\_direction\_input**,**

**.**direction\_output **=** samsung\_gpio\_direction\_output**,**

**.**to\_irq **=** samsung\_gpio\_to\_irq**,**

**.**owner **=** THIS\_MODULE**,**

**};**

static int samsung\_gpiolib\_register**(**struct platform\_device **\***pdev**,** struct samsung\_pinctrl\_drv\_data **\***drvdata**)**

**{**

//得到pin\_bank描述结构体

struct samsung\_pin\_bank **\***bank **=** drvdata**->**pin\_banks**;**

//一个gpio\_chip代表一个bank

struct gpio\_chip **\***gc**;**

int ret**;**

int i**;**

**for** **(**i **=** 0**;** i **<** drvdata**->**nr\_banks**;** **++**i**,** **++**bank**)** **{**

//将三星通用的gpio chip先赋值给gpio\_chip

bank**->**gpio\_chip **=** samsung\_gpiolib\_chip**;**

//修改每个bank不同的属性

gc **=** **&**bank**->**gpio\_chip**;**

//每个bank对应的gpio起始值

gc**->**base **=** drvdata**->**pin\_base **+** bank**->**pin\_base**;**

//每个bank有多少个gpio

gc**->**ngpio **=** bank**->**nr\_pins**;**

//该bank的parent device

gc**->**parent **=** **&**pdev**->**dev**;**

//该bank对应的device node

gc**->**of\_node **=** bank**->**of\_node**;**

//该bank的名字

gc**->**label **=** bank**->**name**;**

//将该bank添加到gpio子系统中

ret **=** gpiochip\_add\_data**(**gc**,** bank**);**

**}**

**return** 0**;**

**}**

因为4.x的内核采用了设备树，所以gpio的申请采用设备树方式，下面将详细分析基于设备树的gpio子系统的使用方法。