# 一、6ULL的GPIO使用

1、设置PIN的复用和电气属性。

2、配置GPIO的

# 一、pinctrl子系统

 借助pinctrl来设置一个PIN的复用和电气属性。

 打开imx6ull.dtsi：

## 1.1 IOMUXC SNVS控制器

 iomuxc\_snvs: iomuxc-snvs@02290000 {

 compatible = "fsl,imx6ull-iomuxc-snvs";

 reg = <0x02290000 0x10000>;

 };

## 1.2 IOMUXC控制器

 iomuxc: iomuxc@020e0000 {

 compatible = "fsl,imx6ul-iomuxc";

reg = <0x020e0000 0x4000>;

 pinctrl-names = "default";

 pinctrl-0 = <&pinctrl\_hog\_1>;

 imx6ul-evk {

 pinctrl\_hog\_1: hoggrp-1 {

 fsl,pins = <

 MX6UL\_PAD\_UART1\_RTS\_B\_\_GPIO1\_IO19 0x17059 /\* SD1 CD \*/

 MX6UL\_PAD\_GPIO1\_IO05\_\_USDHC1\_VSELECT 0x17059 /\* SD1 VSELECT \*/

 MX6UL\_PAD\_GPIO1\_IO09\_\_GPIO1\_IO09 0x17059 /\* SD1 RESET \*/

 >;

 };

 ………

}

 };

 根据设备的类型，创建对应的子节点，然后设备所用PIN都放到此节点。

## 1.3 gpr控制器

 gpr: iomuxc-gpr@020e4000 {

 compatible = "fsl,imx6ul-iomuxc-gpr",

 "fsl,imx6q-iomuxc-gpr", "syscon";

 reg = <0x020e4000 0x4000>;

 };

## 1.4 如何添加一个PIN的信息

 pinctrl\_hog\_1: hoggrp-1 {

 fsl,pins = <

 **MX6UL\_PAD\_UART1\_RTS\_B\_\_GPIO1\_IO19 0x17059 /\* SD1 CD \*/**

 >;

 };

 我们在imx6ul-pinfunc.h中找到：

MX6UL\_PAD\_UART1\_RTS\_B\_\_GPIO1\_IO19 0x0090 0x031C 0x0000 0x5 0x0

<mux\_reg conf\_reg input\_reg mux\_mode input\_val>

0x0090 0x031C 0x0000 0x5 0x0

IOMUXC父节点首地址0x020e0000，因此UART1\_RTS\_B这个PIN的mux寄存器地址 就是：0x020e0000+0x0090=0x020e 0090。

conf\_reg：0x020e0000+0x031C=0x020e 031C，这个寄存器就是UART1\_RTS\_B的电气属性配置寄存器。

input\_reg，便宜为0，表示UART1\_RTS\_B这个PIN没有input功能。

mux\_mode：5表示复用为GPIO1\_IO19，将其写入0x020e 0090

input\_val：就是写入input\_reg寄存器的值。

**0x17059：为PIN的电气属性配置寄存器值。**

## 1.5 pincrtl驱动

 如何找到IMX6UL对应的pinctrl子系统驱动，设备树里面的设备节点是如何根驱动匹配的呢？通过compatible，此属性是字符串列表。驱动文件里面有一个描述驱动兼容性的东西，当设备树节点的compatible属性和驱动里面的兼容性字符串匹配，也就是一模一样的时候就表示设备和驱动匹配了。

 所以我们只需要全局搜索，设备节点里面的compatible属性的值，看看在哪个.c文件里面有，那么此.c文件就是驱动文件。

 找到pinctrl-imx6ul.c文件，那么此文件就是6UL/6ULL的pinctrl驱动文件。

 当驱动和设备匹配以后执行，probe函数。也就是

imx6ul\_pinctrl\_probe

 ->imx\_pinctrl\_probe

 -> 此函数会初始化imx\_pinctrl\_desc，为pinctrl\_desc类型的结构体。重点是：

 imx\_pinctrl\_desc->pctlops = &imx\_pctrl\_ops;

 imx\_pinctrl\_desc->pmxops = &imx\_pmx\_ops;

 imx\_pinctrl\_desc->confops = &imx\_pinconf\_ops;

 最后通过pinctrl\_register函数向系统注册一个imx\_pinctrl\_desc

 ->imx\_pinctrl\_probe\_dt

 ->imx\_pinctrl\_parse\_functions

 ->imx\_pinctrl\_parse\_groups

 -> pin\_reg->mux\_reg = mux\_reg;

 pin\_reg->conf\_reg = conf\_reg;

 pin->input\_reg = be32\_to\_cpu(\*list++);

 pin->mux\_mode = be32\_to\_cpu(\*list++);

 pin->input\_val = be32\_to\_cpu(\*list++);

 config = be32\_to\_cpu(\*list++);

 if (config & IMX\_PAD\_SION)

 pin->mux\_mode |= IOMUXC\_CONFIG\_SION;

 pin->config = config & ~IMX\_PAD\_SION;

imx\_pinconf\_set函数设置PIN的电气属性

imx\_pmx\_set函数设置PIN的复用

# 二、gpio子系统

 使用gpio子系统来使用gpio。

## 2.1 gpio使用方式

1、gpio在设备树中的表示方法

&usdhc1 {

 pinctrl-names = "default", "state\_100mhz", "state\_200mhz";

 pinctrl-0 = <&pinctrl\_usdhc1>;

 pinctrl-1 = <&pinctrl\_usdhc1\_100mhz>;

 pinctrl-2 = <&pinctrl\_usdhc1\_200mhz>;

 **cd-gpios = <&gpio1 19 GPIO\_ACTIVE\_LOW>;**

 keep-power-in-suspend;

 enable-sdio-wakeup;

 vmmc-supply = <&reg\_sd1\_vmmc>;

 status = "okay";

};

 定义了一个cd-gpios属性。

 此处使用GPIO1\_IO19。

 gpio1: gpio@0209c000 {

 compatible = "fsl,imx6ul-gpio", "fsl,imx35-gpio";

 reg = <0x0209c000 0x4000>;

 interrupts = <GIC\_SPI 66 IRQ\_TYPE\_LEVEL\_HIGH>,

 <GIC\_SPI 67 IRQ\_TYPE\_LEVEL\_HIGH>;

 gpio-controller;

 #gpio-cells = <2>;

 interrupt-controller;

 #interrupt-cells = <2>;

 };

devicetree\bindings\gpio

如何从设备树中获取要使用的GPIO信息。of函数。

2、驱动中对gpio的操作函数

 1、首先，获取到GPIO所处的设备节点，比如of\_find\_node\_by\_path。

 2、获取GPIO编号, of\_get\_named\_gpio函数，返回值就是GPIO编号。

 3、请求此编号的GPIO，gpio\_request函数

 4、设置GPIO，输入或输出，gpio\_direction\_input或gpio\_direction\_output。

 5、如果是输入，那么通过gpio\_get\_value函数读取GPIO值，如果是输出，通过gpio\_set\_value设置GPIO值。

## 2.2 gpio驱动

1、gpiolib

 两部分，给原厂编写GPIO底层驱动的，给驱动开发人员使用GPIO操作函数的。

 使用gpiochip\_add向系统添加gpio\_chip，这些都是半导体原厂做的，这部分就是最底层的GPIO驱动。

2、gpio驱动

 在drivers/gpio目录下，gpio-xxx.c文件为具体芯片的驱动文件，

 static struct mxc\_gpio\_hwdata imx35\_gpio\_hwdata = {

 .dr\_reg = 0x00,

 .gdir\_reg = 0x04,

 .psr\_reg = 0x08,

 .icr1\_reg = 0x0c,

 .icr2\_reg = 0x10,

 .imr\_reg = 0x14,

 .isr\_reg = 0x18,

 .edge\_sel\_reg = 0x1c,

 .low\_level = 0x00,

 .high\_level = 0x01,

 .rise\_edge = 0x02,

 .fall\_edge = 0x03,

};

mxc\_gpio\_probe

 -> mxc\_gpio\_get\_hw 获取6ULL的GPIO寄存器组。

 -> bgpio\_init 重点初始化gpio\_chip

 ->gpiochip\_add 向系统添加gpio\_chip

# 三、编写驱动

3.1、修改设备树

3.2 编写驱动

 申请IO的时候失败，大部分原因是这个IO被其他外设占用了。检查设备树，查找有哪些使用同一IO的设备。

 1，检查复用，也就是pinctl设置。

 2，gpio使用

# 四、运行测试

五、总结

 1，添加pinctrl信息，

 2，检查当前设备树中要使用的IO有没有被其他设备使用，如果有的话要处理。

 3，添加设备节点，在设备节点中创建一个属性，此属性描述所使用的gpio。

 4，编写驱动，获取对应的gpio编号，并申请IO，成功以后即可使用此Io