PX30 IO Domain Configuration Developer Guide

ID: RK-SM-YF-909

Release Version: V1.0.0

Release Date: 2021-05-15

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

The IO level of the controller's power domain must be matched with the IO level of the connected peripheral chip, and the voltage configuration of the software must be consistent with the voltage of the hardware. Otherwise, it may cause IO damage at worst.

This document mainly introduce the ways to configure IO power domain of PX30 Linux SDK platform, aiming to help developers to configure IO power domain correctly.

Product Version

Chipset	System Version
PX30	Linux 4.4

Intended Audience

This document (this guide) is mainly intended for:

- Technical support engineers
- Software development engineers
- Hardware development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	Caesar Wang	2021-05-15	Initial version

Contents

PX30 IO Domain Configuration Developer Guide

1. Step 1: Obtain the Hardware Schematic Diagram and Check the Design of the Hardware Power Supply
2. Step 2: Find the Corresponding Kernel dts Configuration File
3. Step 3: Modify the Power Domain Configuration Node pmu_io_domains of the Kernel dts
4. Step 4: Check the Current Firmware IO Domain Configuration from SDK
5. Step 5: Confirm Whether the Register Value is Correct after Flashing the Firmware

1. Step 1: Obtain the Hardware Schematic Diagram and Check the Design of the Hardware Power Supply

It will take px30_mini_evb_v11_20190507 EVB as an example to introduce in this document.

Hardware schematic diagram is: px30_mini_evb_v11_20190507.pdf.

Power solution: checking from the hardware schematic, the power solution of the **px30_mini_evb_v11_20190507 EVB** is with a PMU (RK809-1).

2. Step 2: Find the Corresponding Kernel dts Configuration File

From the first step, it can be seen that the hardware power supply design of the EVB is with a PMU , and the corresponding kernel dts configuration file is located in:

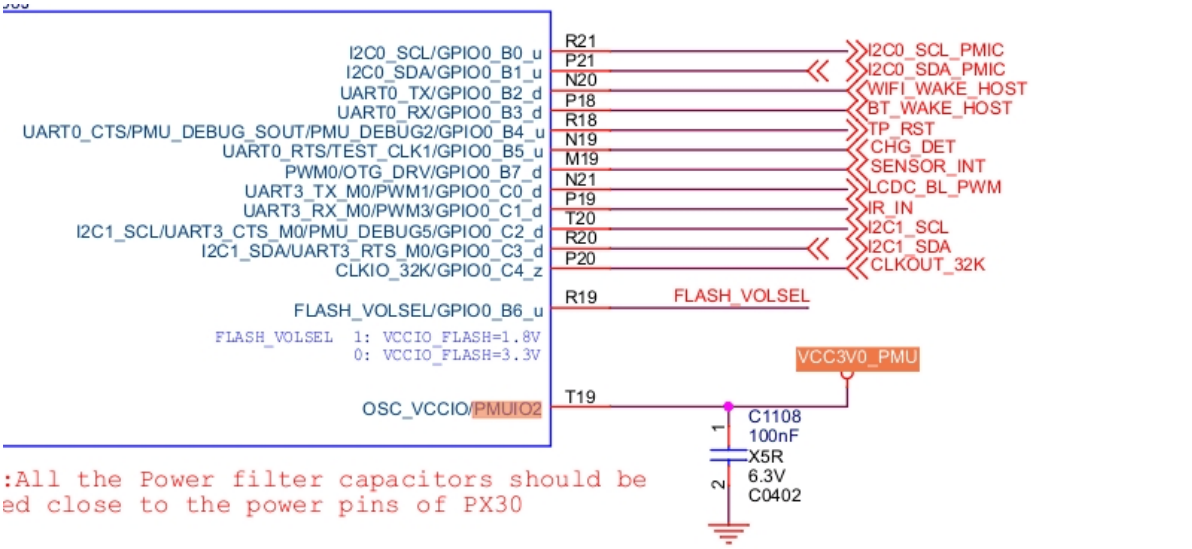
arch/arm64/boot/dts/rockchip/px30-evb-ddr3-v10-linux.dts (The solution discussed in this document)

3. Step 3: Modify the Power Domain Configuration Node pmu_io_domains of the Kernel dts

```
&io_domains {
        status = "okay";

        vccio1-supply = <&vcc_3v0>;
        vccio2-supply = <&vccio_sd>;
        vccio3-supply = <&vcc_3v0>;
        vccio4-supply = <&vcc3v0_pmu>;
        vccio5-supply = <&vcc_3v0>;
};

&pmu_io_domains {
        status = "okay";

        pmuio1-supply = <&vcc3v0_pmu>;
        pmuio2-supply = <&vcc3v0_pmu>;
};
```

Take **pmuio2-supply** for example, firstly, check the hardware schematic diagram to confirm the configuration of the pmuio2 power domain (PMUIO2) as shown in the below figure.

From the figure, you will find that the power supply of **PMUIO2** is VCC3V0_PMU (that is 3.0V)



4. Step 4: Check the Current Firmware IO Domain Configuration from SDK

Command: `./build.sh info`

```
PLEASE CHECK BOARD GPIO POWER DOMAIN CONFIGURATION !!!!!
<<< ESPECIALLY Wi-Fi/Flash/Ethernet IO power domain >>> !!!!!
Check Node [pmu_io_domains] in the file: /home/wxt/linux-develop/px30/kernel/arch/arm64/boot/dts/rockchip/px30-evb-ddr3-v11-linux.dts

请再次确认板级的电源域配置！！！！！！
<<< 特别是Wi-Fi，FLASH，以太网这几路IO电源的配置 >>> ！！！！！
检查内核文件 /home/wxt/linux-develop/px30/kernel/arch/arm64/boot/dts/rockchip/px30-evb-ddr3-v11-linux.dts 的节点 [pmu_io_domains]


pmuio1-supply
regulator-min-microvolt =  3000mV
regulator-max-microvolt =  3000mV


pmuio2-supply
regulator-min-microvolt =  3000mV
regulator-max-microvolt =  3000mV


vccio1-supply
regulator-min-microvolt =  3000mV
regulator-max-microvolt =  3000mV


vccio2-supply
regulator-min-microvolt =  1800mV
regulator-max-microvolt =  3300mV
```

5. Step 5: Confirm Whether the Register Value is Correct after Flashing the Firmware

Take **PX30** as an example, get PMU_SOC_CON0 register (0xFF010100) and GRF_IO_VSEL register (0xFF140180) from the manual, they are shown as follows:

PMUGRF_SOC_CON0

Address: Operational Base + offset (0x0100)

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable When bit 16=1, bit0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; …… When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;

Bit	Attr	Reset Value	Description
15	RW	0x0	poc_pmuio2_sel18 PMU VCCIO2 voltage select 1'b0: 3.3V 1'b1: 1.8V
14	RW	0x0	poc_pmuio1_sel18 PMU VCCIO1 voltage select 1'b0: 3.3V 1'b1: 1.8V
13	RW	0x0	ddrphy_bufferen_core 1'b1: enable ddrphy_bufferen; 1'b0: disable ddrphy_bufferen
12	RW	0x0	ddrphy_bufferen_sel 1'b1: ddrphy_bufferen from ddrphy_bufferen_core; 1'b0: ddrphy_bufferen from pmu and ddr_fail_safe
11:7	RO	0x0	reserved
6	RW	0x0	uart0_cts_sel 1'b1: reverse polarity of cts;
5	RW	0x0	uart0_rts_sel 1'b1: reverse polarity of rts;

GRF_IO_VSEL

Address: Operational Base + offset (0x0180)

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable When bit16=1, bit0 can be written by software. When bit16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:8	RO	0x0	reserved
7	RW	0x0	grf_vccio_oscgpi_vsel IO voltage select 1'b0:3.3V 1'b1:1.8V
6	RW	0x0	grf_vccio5_vsel VCC IO5 voltage select 1'b0:3.3V 1'b1:1.8V
5	RW	0x0	grf_vccio4_vsel VCC IO4 voltage select 1'b0:3.3V 1'b1:1.8V
4	RW	0x0	grf_vccio3_vsel VCC IO3 voltage select 1'b0:3.3V 1'b1:1.8V
3	RW	0x0	grf_vccio2_vsel VCC IO2 voltage select 1'b0:3.3V 1'b1:1.8V

```
# io -r -4 0xFF010100
ff010100:  00002380

# io -r -4 0xFF140180
ff140180:  00000003
```