地平线
Horizon Robotics

AN-2521-1-A-X3J3平台PMIC驱动修改指导

v1.0

2021-02

免责声明

本文档信息仅用于帮助系统和软件使用人员使用地平线产品。本文档信息未以明示或暗示方式授权他人基于本文档信息设计或制造任何集成电路。

本文档中的信息如有更改，恕不另行通知。尽管本文档已尽可能确保内容的准确性，本文档中的所有声明、信息和建议均不构成任何明示或暗示的保证、陈述或担保。

本文档中的所有信息均按“原样”提供。地平线不就其产品在任何特定用途的适销性、适用性以及不侵犯任何第三方知识产权方面做出任何明示或暗示保证、陈述或担保。地平线不承担产品使用所引起的任何责任，包括但不限于直接或间接损失赔偿。

买方和正在基于地平线产品进行开发的其他方（以下统称为“用户”）理解并同意，用户在设计产品应用时应承担独立分析、评估和判断的责任。用户应对其应用（以及用于其应用的所有地平线产品）的安全性承担全部责任，并保证符合所有适用法规、法律和其它规定的要求。地平线产品简介和产品规格中提供的“典型”参数在不同应用下可能会不同，实际性能也可能随时间而变化。所有工作参数，包括“典型”参数，都必须由用户自己针对每项用户应用进行验证。

用户同意如因用户未经授权使用地平线产品或因不遵守本说明中的条款，造成任何索赔、损害、成本、损失和（或）责任，用户将为地平线及其代表提供全额赔偿。

修订记录

修订记录列出了各文档版本间发生的主要更改。下表列出了每次文档更新的技术内容。

版本	修订日期	修订说明
1.0	2020-4-26	X3J3 平台 PMIC 驱动修改指导

目录

插图目录

1 概述

1.1 文档概述

本文档用于指导如何基于 kernel regulator 框架编写 PMIC 驱动。文档以 X-powers 公司生产的 AXP15060 电源管理模块为例进行说明。

AXP15060 支持 23 路电源输出（包括 6 路 DCDC）。为了保证电力系统的安全稳定，AXP15060 集成了过电压保护（OVP）、欠电压保护（UVP）和过温保护（OTP）等保护电路。

1.2 多电源输出

表 1-1 列出了 AXP15060 PMIC 的多电源输出。

	Output path	type	Default voltage	Startup sequence	Application suggestion	Load capacity(MAX)
1	DCDC1	BUCK	3.3v	2	IO/USB	2000mA
2	DCDC2	BUCK	0.9v	2	CPU	3500mA
3	DCDC3	BUCK	0.9v	1	GPU	3500mA
4	DCDC4	BUCK	0.9v	1	SYS	2000mA
5	DCDC5	BUCK	1.1v/DC5SET	1	DDR	2000mA
6	DCDC6	BUCK	OFF	OFF	LDO	2000mA
7	ALDO1	LDO	OFF	OFF	N/A	600mA
8	ALDO2	LDO	1.8V	2	N/A	300mA
9	ALDO3	LDO	1.8V	2	N/A	200mA
10	ALDO4	LDO	3.3V	2	N/A	300mA
11	ALDO5	LDO	2.5V	1	N/A	300mA
12	BLDO1	LDO	OFF	OFF	N/A	300mA
13	BLDO2	LDO	OFF	OFF	N/A	500mA
14	BLDO3	LDO	OFF	OFF	N/A	300mA
15	BLDO4	LDO	OFF	OFF	N/A	400mA
16	BLDO5	LDO	1.8V	2	N/A	600mA

17	CLDO1	LDO	OFF	OFF	N/A	200mA
18	CLDO2	LDO	3.3V	2	N/A	200mA
19	CLDO3	LDO	OFF	OFF	N/A	300mA
20	CLDO4	LDO	OFF	OFF	N/A	200mA
21	VCPUS	LDO	0.9V	1	CPUs/Reference of DDR	200mA
22	RTC-LDO	LDO	1.8V	Always on	RTC	100mA
23	DC1SW	Switch	OFF	OFF	N/A	1000mA

表 1-1 AXP15060 多电源输出

Table1-1 AXP15060 Multi-Power Outputs

具体需要哪些输出端口需要根据的硬件需求进行。

2 DTS 配置

根据硬件原理图来配置具体的 DTS。Hobot X3 平台使用了 AXP15060 PMIC 的 DCDC1 ~ DCDC6，BLDO1，CLDO2 等几个电压输出端口。

2.1 X3 平台 PMIC 配置

Hobot X3 平台关于 PMIC DTS 配置如下：

```
&i2c0 {
   axp15060@37 {
        compatible = "X-Powers,axp15060";
        reg = <0x37>;
        regulators {
             sys_pd1_3v3_reg: DCDC1 {
                  regulator-name = "VDD_3V3";
                  regulator-min-microvolt = <3300000>;
                  regulator-max-microvolt = <3300000>;
                  regulator-always-on;
             };

             cnn0_pd_reg: DCDC2 {
                  regulator-name = "VCC_CNN0";
                  regulator-min-microvolt = <800000>;
                  regulator-max-microvolt = <1000000>;
                  regulator-enable-ramp-delay = <3000>;
             };

             cnn1_pd_reg: DCDC3 {
                  regulator-name = "VCC_CNN1";
                  regulator-min-microvolt = <800000>;
                  regulator-max-microvolt = <1000000>;
                  regulator-enable-ramp-delay = <3000>;
             };

             cpu_pd_reg: DCDC4 {
                  regulator-name = "VCC_CPU";
                  regulator-min-microvolt = <800000>;
                  regulator-max-microvolt = <1000000>;
                  regulator-always-on;
             };
```

```
            ddr_ao_1v1: DCDC5 {
                regulator-name = "DDR_AO_1V1";
                regulator-min-microvolt = <1100000>;
                regulator-max-microvolt = <1100000>;
                regulator-always-on;
            };

            ddr_pd_reg: DCDC6 {
                regulator-name = "VDD_DDR_PD";
                regulator-min-microvolt = <800000>;
                regulator-max-microvolt = <800000>;
                regulator-always-on;
            };

            core_ao_reg: BLDO1 {
                regulator-name = "VDD_CORE_AO";
                regulator-min-microvolt = <800000>;
                regulator-max-microvolt = <800000>;
                regulator-always-on;
            };

            sys_pd_1v8_reg: CLDO2 {
                regulator-name = "VDD_1V8";
                regulator-min-microvolt = <1800000>;
                regulator-max-microvolt = <1800000>;
                regulator-always-on;
            };

        };
    };

};

&cpu0 {
    cpu-supply = <&cpu_pd_reg>;
};

&cpu1 {
    cpu-supply = <&cpu_pd_reg>;
};

&cpu2 {
    cpu-supply = <&cpu_pd_reg>;
};
```

```
&cpu3 {
   cpu-supply = <&cpu_pd_reg>;
};

&cnn0 {
   cnn-supply = <&cnn0_pd_reg>;
};

&cnn1 {
   cnn-supply = <&cnn1_pd_reg>;
};
```

在编写 PMIC 驱动的时候，需要注意的是，regulator-name 选项需要和 Hobot 平台保持一致，如果不一致的话，可能会导致某些模块无法正常工作。

配置 DTS 时候一般是配置 regulator 的最小，最大输出电压，以及是否一直保持输出等参数。

regulator 的其他属性的详细介绍可以参考 kernel 文档介绍，文档位于如下目录：

kernel/Documentation/devicetree/bindings/regulator/regulator.txt

3 PMIC 驱动编写

PMIC 的大部分工作，kernel regulator 子系统已经帮我们完成了，编写 PMIC 驱动的主要任务就是根据 PMIC Datasheet 来定义相关的宏或者结构体。定义的宏主要是 PMIC Register 以及一些操作位和 PMIC 输出电压的步进数等。结构体主要是 PMIC 线性输出电压的范围等。这个宏或者结构体主要是围绕下面的宏展开的。

```
#define AXP15060_REG(_name, _id, _linear, _step, _vset_mask, _enable)       \
    [ID_##_id] = {                  \
        .name           = _name,            \
        .id         = ID_##_id,     \
        .type           = REGULATOR_VOLTAGE,        \
        .ops            = &axp15060_ops,            \
        .n_voltages     = AXP15060_VOLTAGE_NUM##_step,      \
        .linear_ranges      = axp15060_voltage_ranges##_linear, \
        .n_linear_ranges    = ARRAY_SIZE(axp15060_voltage_ranges##_linear), \
        .vsel_reg       = AXP15060##_##_id##_VSET, \
        .vsel_mask      = AXP15060_VSET_MASK##_vset_mask,       \
        .enable_reg     = (AXP15060_ON_OFF_CTRL##_enable),  \
        .enable_mask        = AXP15060##_##_id##_ENA,           \
        .disable_val        = AXP15060_POWER_OFF,           \
        .owner          = THIS_MODULE,          \
    }

static const struct regulator_desc axp15060_regulators[] = {
    AXP15060_REG("DCDC1", DCDC1, 1, 20, 5, 1),
    AXP15060_REG("DCDC2", DCDC2, 2, 88, 7, 1),
    AXP15060_REG("DCDC3", DCDC3, 2, 88, 7, 1),
    AXP15060_REG("DCDC4", DCDC4, 2, 88, 7, 1),
    AXP15060_REG("DCDC5", DCDC5, 3, 69, 7, 1),
    AXP15060_REG("DCDC6", DCDC6, 4, 30, 5, 1),
    AXP15060_REG("BLDO1", BLDO1, 5, 27, 5, 2),
    AXP15060_REG("CLDO2", CLDO2, 5, 27, 5, 3),
};
```

这部分内容是 PMIC 驱动的核心部分。

axp15060_regulators 结构体中的成员就是刚才在 DTS 中配置的所有 regulator。

3.1 name 字段

AXP15060_REG 的 name 字段与 DTS 配置的对应关系为：

例如：

AXP15060_REG("DCDC1", DCDC1, 1, 20, 5, 1)对应 DTS“sys_pd1_3v3_reg: DCDC1”。

3.2 id 字段

id 字段的声明是在 kernel/include/linux/regulator 目录，如下：

```
enum {
    ID_DCDC1,
    ID_DCDC2,
    ID_DCDC3,
    ID_DCDC4,
    ID_DCDC5,
    ID_DCDC6,
    ID_BLDO1,
    ID_CLDO2,
};
```

3.3 type 字段

type 字段是固定的，类型就是 REGULATOR_VOLTAGE，也就是用于输出电压控制。

3.4 ops 字段

ops 字段是 PMIC 的操作集合，也是固定的，已被 regulator 子系统实现。这些操作函数主要是在 kernel/drivers/regulator 目录中实现，主要就是为了映射电压，设置以及读取输出电压。

```
static struct regulator_ops axp15060_ops = {
    .list_voltage       = regulator_list_voltage_linear_range,
    .map_voltage        = regulator_map_voltage_linear_range,
    .get_voltage_sel    = regulator_get_voltage_sel_regmap,
    .set_voltage_sel    = regulator_set_voltage_sel_regmap,
    .enable         = regulator_enable_regmap,
    .disable        = regulator_disable_regmap,
    .is_enabled     = regulator_is_enabled_regmap,
};
```

3.5 n_voltages 字段

n_voltages 字段指的是 regulator 的输出的步进数，这个需要查看 PMIC Datasheet 来获取。

```
/*
 * PF5024 voltage number
 */
#define AXP15060_VOLTAGE_NUM20      20
#define AXP15060_VOLTAGE_NUM27      27
#define AXP15060_VOLTAGE_NUM30      30
```

```
#define AXP15060_VOLTAGE_NUM36      36
#define AXP15060_VOLTAGE_NUM69      69
#define AXP15060_VOLTAGE_NUM88      88
```

REG 13：DC/DC 1 voltage control

Default：12H

Reset：system reset

Bit	Description	R/W	Default
7-5	Reserved	RW	0
4-0	DCDC-1 voltage setting bit4-0, default is 3.3V: 1.5~3.4V，100mV/step，20steps	RW	10010

图 3-1 DCDC1 电压步进数

Figure3-1 Step number of DCDC1

3.6 linear_ranges 字段

linear_ranges 字段是 regulator 电压的输出范围，主要描述的是 regulator 的输出电压与步进数的关系，包括最小电压值，步进值等。这些参数也需要根据 Datasheet 来确认。

struct regulator_linear_range 中描述的电压值是以 uv 为单位的。500000 表示最低电压是 0.5v，0，70，71，87 表示的是步进，10000，20000 表示的是步进值。

```
static const struct regulator_linear_range axp15060_voltage_ranges2[] = {
    REGULATOR_LINEAR_RANGE(500000, 0, 70, 10000),
    REGULATOR_LINEAR_RANGE(1220000, 71, 87, 20000),
};
```

例如，DCDC2 就是对应上面结构体的实现，其 Datasheet 描述如图 3-2 所示：

REG 14：DC/DC 2 voltage control

Default：3CH

Reset：system reset

Bit	Description	R/W	Default
7	Reserved	RW	0
6-0	DCDC-2 voltage setting bit6-0, default is 1.1V: 0.5~1.2V，10mV/step，71steps 1.22~1.54V，20mV/step，17steps	RW	0111100

图 3-2 DCDC2 电压步进

Figure3-2 Step value of DCDC2

3.7 n_linear_ranges 字段

n_linear_ranges 字段描述的是 struct regulator_linear_range 结构体中的元素个数。

3.8 vsel_reg 字段

vsel_reg 字段是设置电压的 register，需要根据 Datasheet 来确认。例如 DCDC6 设置电压的 register 地址是 0x18。

#define AXP15060_DCDC6_VSET 0x18 //DCDC6 voltage set

REG 18：DC/DC 6 voltage control

Default：06H

Reset：system reset

Bit	Description	R/W	Default
7-5	Reserved	RW	0
4-0	DCDC-6 voltage setting bit4-0, default is 1.1V: 0.5~3.4V，100mV/step，30steps	RW	00110

图 3-3 设置 DCDC6 输出电压

Figure3-3 set voltage value on DCDC6

3.9 vsel_mask 字段

vsel_mask 字段指的是设置输出电压时的 mask。或者说是设置以及读取输出电压时 register 的哪几位有效，需要根据 Datasheet 来确认。例如 AXP15060 的 vsel_mask 有两种，低 5 位有效或者低 7 位有效。

```
1.  #define AXP15060_VSET_MASK5        0x1F /*VSET - [4:0]*/
2.  #define AXP15060_VSET_MASK7        0x7F /*VSET - [6:0]*/
```

REG 17：DC/DC 5 voltage control

Default：2CH

Reset：system reset

Bit	Description	R/W	Default
7	Reserved	RW	0
6-0	DCDC-5 voltage setting bit6-0, default is 1.36V: 0.8~1.12V，10mV/step，33steps 1.14~1.84V，20mV/step，36steps	RW	0101100

REG 18：DC/DC 6 voltage control

Default：06H

Reset：system reset

Bit	Description	R/W	Default
7-5	Reserved	RW	0
4-0	DCDC-6 voltage setting bit4-0, default is 1.1V: 0.5~3.4V，100mV/step，30steps	RW	00110

图 3-4 AXP15060 输出电压的 mask 位

Figure3-4 mask bits of AXP15060 output voltage

3.10 enable_reg 字段

enable_reg 字段指是时控制 regulator 输出使能的寄存器。需要根据 Datasheet 来确认。例如控制 DCDC 输出使能的寄存器地址是 0x10。

```
1.  #define AXP15060_ON_OFF_CTRL1       0x10  //output power on-off control
2.  #define AXP15060_ON_OFF_CTRL2       0x11  //output power on-off control
3.  #define AXP15060_ON_OFF_CTRL3       0x12  //output power on-off control
```

REG 10：Output power on-off control 1

Default：37H

Reset：system reset

Bit	Description		R/W	Default
7-6	Reserved		RW	0
5	DCDC-6 on-off control	0-off; 1-on	RW	1
4	DCDC-5 on-off control	0-off; 1-on	RW	1
3	DCDC-4 on-off control	0-off; 1-on	RW	0
2	DCDC-3 on-off control	0-off; 1-on	RW	1
1	DCDC-2 on-off control	0-off; 1-on	RW	1
0	DCDC-1 on-off control	0-off; 1-on	RW	1

图 3-5 DCDC 输出电压使能

Figure3-5 enable output voltage on DCDC

3.11 enable_mask 字段

enable_mask 字段指的是输出电压的 mask。需要根据 Datasheet 来确认。其实也就是 enable 寄存器的哪一位来确定使能某一路 output 的使能。这个可以根据图 3-5 来看，例如 DCDC1 的使能位是 0x10 寄存器的 bit0。

```
#define AXP15060_DCDC6_ENA          0x20
#define AXP15060_DCDC5_ENA          0x10
#define AXP15060_DCDC4_ENA          0x08
#define AXP15060_DCDC3_ENA          0x04
#define AXP15060_DCDC2_ENA          0x02
#define AXP15060_DCDC1_ENA          0x01
```

3.12 disable_val 字段

disable_val 字段表示 disable 某一路电压的输出。如果不严格的话，该字段指定为 0 即可。

3.13 PMIC DTS 解析

以上的宏或者结构体完成后，pmic 驱动编写的工作已经完成了大部分。剩下的就是从 DTS 中解析数据，以及 probe 了。

解析 DTS 就是解析第 2 节中 DTS 中 regulators 节点下的数据：

```
np = of_get_child_by_name(dev->of_node, "regulators");
if (!np) {
    dev_err(dev, "missing 'regulators' subnode in DT, %d\n", -EINVAL);
    return -EINVAL;
}
```

具体的函数实现可以参考 axp15060-regulator.c 文件的 axp15060_pdata_from_dt 函数。

3.14 PMIC probe

pmic 驱动 probe 函数主要完成的任务就是完成 regulator 的注册。流程基本上也是固定的，可以参考 axp15060-regulator.c 文件的 axp15060_pmic_probe 函数。

```
/* Finally register devices */
for (i = 0; i < num_regulators; i++) {
```

```
3.      const struct regulator_desc *desc = &regulators[i];
4.      struct regulator_config config = { };
5.      struct axp15060_regulator_data *rdata;
6.      struct regulator_dev *rdev;
7.
8.      config.dev = dev;
9.      config.driver_data = axp15060;
10.     config.regmap = axp15060->regmap;
11.
12.     rdata = axp15060_get_regulator_data(desc->id, pdata);
13.     if (rdata) {
14.         config.init_data = rdata->init_data;
15.         config.of_node = rdata->of_node;
16.     }
17.
18.     rdev = devm_regulator_register(dev, desc, &config);
19.     if (IS_ERR(rdev)) {
20.         dev_err(dev, "failed to register %s\n", desc->name);
21.         return PTR_ERR(rdev);
22.     }
23. }
```