

A64

AXP803 Linux Power Driver 应用文档

Confidential

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2014.08.28	李明	
V1.1	2015.09.23	关健哲	适配 axp803; 修正语误
V1.2	2016.02.22	关健哲	删除不必要的节点说明

Confidential

目 录

A64.....	1
AXP81X Linux Power Driver 应用文档/V1.0.....	1
目 录	2
1. 概述.....	3
1.1. 编写目的.....	3
1.2. 适用范围.....	3
1.3. 相关人员.....	3
2. 模块源码结构.....	4
3. 模块配置.....	5
3.1. menuconfig 选项配置:	5
3.2. sys_config.fex 中的文件配置.....	5
3.2.1. 供电依赖关系.....	5
3.2.2. 电池管理部分.....	6
4. Regulator.....	13
4.1. regulator 使用 demo.....	13
4.1.1. LDO/DCDC 电源对应表.....	13
4.1.2. 使用示例.....	13
4.1.3. dump 节点使用方法.....	14
4.2. Regulator shell 命令使用示例.....	15
5. 电池管理.....	16
5.1. 电池信息.....	16
5.2. sysfs 文件节点.....	16
5.2.1. MFD.....	16
5.2.2. Power_supply (标准):	17
5.2.3. Power_supply (非标准):	18
6. 其它.....	19
6.1. GPIO.....	19
6.1.1. AXP GPIO 对应表.....	19
6.1.2. AXP GPIO 使用示例.....	19
6.2. Shell 命令读取或修改 AXP81X 寄存器.....	20
6.3. 打印控制.....	21

1. 概述

1.1. 编写目的

介绍 axp803 软件部分的使用方法，方便用户理解及调试 PMU。

1.2. 适用范围

软件：Linux-3.10 内核。

1.3. 相关人员

全志 A64 PMU 及驱动相关软件研发人员。

Confidential

2. 模块源码结构

```
linux-3.10
drivers\power\axp_power
|—— axp-irq.c
|—— axp-state.c
|—— axp-filenode.c
|—— axp-regu.h
|—— axp-regu.c
|—— axp-script.c
|—— axp-regu-script.c
|—— axp-mfd.c
|—— axp-cfg.h
|—— axp-rw.h
|—— Kconfig
|—— Makefile
drivers\power\axp_power\axp81x
|—— axp81x-board.c
|—— axp81x-regu.c
|—— axp81x-init.c
|—— axp81x-sply.c
|—— virtual81x.c
|—— virtural81x-dev.c
|—— axp81x-common.h
|—— axp81x-mfd.h
|—— axp81x-regu.h
|—— axp81x-sply.h
include\linux\mfd
|—— axp-mfd.h
|—— axp-mfd-81x.h
drivers\pinctrl\axp
|—— pinctrl-axp.c
|—— pinctrl-axp.h
```

axp_power 目录下为 AXP 驱动中的公共部分，各 AXP 驱动共用。

axp_power/axp81x/为 AXP803 驱动中的私有部分。

drivers\pinctrl\axp 目录下的 pinctrl-axp.c，pinctrl-axp.h 为 AXP803 的 GPIO 驱动。

3. 模块配置

3.1. menuconfig 选项配置:

进入 Linux3.10 目录，键入 make ARCH=arm64 menuconfig。需勾选如下选项：

```
Device Drivers
|——Voltage and Current Regulator Support
|——Power supply class support —> AXP Power drivers —> AXP PMU type ( AXP81X driver )
```

menuconfig 界面如下：

```
.config - Linux/arm 3.10.65 Kernel Configuration
> Device Drivers > Power supply class support > AXP Power drivers
  AXP Power drivers
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
  - AXP Power drivers
  [*]  AXP81X driver
  [ ]  AXP use twi as transfer channel
  [*]  AXP use NMI as IRQ SRC
  <Select>  < Exit >  < Help >  < Save >  < Load >
```

AXP81X driver 为 axp803 驱动编译选项。

AXP use twi as transfer channel 为 AXP i2c 通信接口编译选项（AXP 与主控芯片采用 I2C 接口通信时，选择此编译选项）

3.2. sys_config.fex 中的文件配置

3.2.1. 供电依赖关系

```
[pmu1_regu]
ldo_count = 23
ldo1      = "axp81x_dcdc1 none sys vcc-emmc vcc-usb0-33 vcc-io vcc-io-gps vcc-io1 vcc-sensor"
ldo2      = "axp81x_dcdc2 none sys vdd-cpua"
ldo3      = "axp81x_dcdc3 none"
ldo4      = "axp81x_dcdc4 none"
ldo5      = "axp81x_dcdc5 none sys vcc-dram"
ldo6      = "axp81x_dcdc6 none sys vdd-sys vdd-usb0-09 vdd-hdmi-09"
ldo7      = "axp81x_dcdc7 none"
ldo8      = "axp81x_rtc none"
```

```

ldo9      = "axp81x_ald01 none sys vcc-dsi-18 vcc-csi2-18 vcc-lvds-18 vcc-efuse-18 vcc-hdmi-18
vcc-pd"
ldo10     = "axp81x_ald02 none sys vdd-drampll vdd-lpddr-18 vcc-pll-18 vcc-adc-18 vdd-codec-cp
vdd-codec-ldoin"
ldo11     = "axp81x_ald03 none sys vcc-avcc vcc-pl"
ldo12     = "axp81x_dldo1 none vcc-wifi-io vcc-io2"
ldo13     = "axp81x_dldo2 none vcc-lcd-0"
ldo14     = "axp81x_dldo3 none afvcc-csi iovdd-csi"
ldo15     = "axp81x_dldo4 none avdd-csi"
ldo16     = "axp81x_eldo1 none dvdd-csi-12"
ldo17     = "axp81x_eldo2 none vcc_dsi"
ldo18     = "axp81x_eldo3 none dvdd-csi-18"
ldo19     = "axp81x_fldo1 none vcc-hsic-12"
ldo20     = "axp81x_fldo2 none sys vdd-cpus"
ldo21     = "axp81x_gpio0ldo none vcc-ctp"
ldo22     = "axp81x_gpio1ldo none vcc-card"
ldo23     = "axp81x_dc1sw none"

```

此表，为电源拓扑结构的配置。

第一列，为 ldo/dcdc 的 name。如 axp81x_dc1。

第二列，为 ldo 之间依赖关系的配置项，如 eldo 的输入端为 dc1，则 eldo 此配置为 1，代表此路 ldo 依赖于 ldo1 对应的 axp81x_dc1。如果，不依赖于其他 ldo，则配置成 none。

从第三列，开始，为各个模块 supply id 的配置项，供各个驱动模块调用，regulator 获得句柄时使用。supply id 之间以空格隔开，每个 supply id 规定为 20 个字符以内，每个 ldo 最多支持 18 个 supply id。

3.2.2. 电池管理部分

```

-----
;pmu1 is axp81x
;
-----
[pmu1_para]
pmu_used          = 1
pmu_id            = 6
pmu_twi_addr      = 0x34
pmu_twi_id        = 0
pmu_irq_id        = 0
pmu_battery_rdc   = 100
pmu_battery_cap   = 0
pmu_batdeten      = 1
pmu_chg_ic_temp   = 0
pmu_runtime_chgcur = 1000
pmu_earlysuspend_chgcur = 1000
pmu_suspend_chgcur = 1600
pmu_shutdown_chgcur = 1600

```

```
pmu_init_chgvol           = 4200
pmu_init_chgend_rate      = 15
pmu_init_chg_enabled      = 1
pmu_init_bc_en           = 0
pmu_init_adc_freq         = 800
pmu_init_adcts_freq       = 800
pmu_init_chg_pretime      = 70
pmu_init_chg_csttime     = 720
pmu_batt_cap_correct      = 1
pmu_bat_regu_en          = 0

pmu_bat_para1             = 0
pmu_bat_para2             = 0
pmu_bat_para3             = 0
pmu_bat_para4             = 0
pmu_bat_para5             = 0
pmu_bat_para6             = 0
pmu_bat_para7             = 0
pmu_bat_para8             = 0
pmu_bat_para9             = 5
pmu_bat_para10            = 8
pmu_bat_para11            = 9
pmu_bat_para12            = 10
pmu_bat_para13            = 13
pmu_bat_para14            = 16
pmu_bat_para15            = 20
pmu_bat_para16            = 33
pmu_bat_para17            = 41
pmu_bat_para18            = 46
pmu_bat_para19            = 50
pmu_bat_para20            = 53
pmu_bat_para21            = 57
pmu_bat_para22            = 61
pmu_bat_para23            = 67
pmu_bat_para24            = 73
pmu_bat_para25            = 78
pmu_bat_para26            = 84
pmu_bat_para27            = 88
pmu_bat_para28            = 92
pmu_bat_para29            = 93
pmu_bat_para30            = 94
pmu_bat_para31            = 95
pmu_bat_para32            = 100

pmu_usbvol_limit          = 0
pmu_usbcur_limit         = 0
```

```
pmu_usbvol           = 4000
pmu_usbcur           = 0
pmu_usbvol_pc        = 4400
pmu_usbcur_pc        = 500
pmu_pwroff_vol       = 3300
pmu_pwron_vol        = 2600
pmu_pekoff_time      = 6000
pmu_pekoff_func      = 0
pmu_pekoff_en        = 1
pmu_pekoff_delay_time = 0
pmu_peklong_time     = 1500
pmu_pekcon_time      = 1000
pmu_pwrok_time       = 64
pmu_pwrok_shutdown_en = 0
pmu_reset_shutdown_en = 1
pmu_battery_warning_level1 = 15
pmu_battery_warning_level2 = 0
pmu_restvol_adjust_time = 60
pmu_ocv_cou_adjust_time = 60
pmu_chgled_func      = 0
pmu_chgled_type      = 0
pmu_vbusen_func      = 1
pmu_reset            = 0
pmu_IRQ_wakeup       = 0
pmu_hot_shutdown     = 1
pmu_inshort          = 0
power_start          = 0

pmu_temp_enable      = 0
pmu_charge_1tf       = 2261
pmu_charge_hrf       = 388
pmu_discharge_1tf    = 3200
pmu_discharge_hrf    = 237
pmu_temp_para1       = 7466
pmu_temp_para2       = 4480
pmu_temp_para3       = 3518
pmu_temp_para4       = 2786
pmu_temp_para5       = 2223
pmu_temp_para6       = 1788
pmu_temp_para7       = 1448
pmu_temp_para8       = 969
pmu_temp_para9       = 664
pmu_temp_para10      = 466
pmu_temp_para11      = 393
pmu_temp_para12      = 333
pmu_temp_para13      = 283
```

```

pmu_temp_para14      = 242
pmu_temp_para15      = 179
pmu_temp_para16      = 134

```

配置项	相关说明
pmu_used	是否使用 AXPxx : 0:不使用,1:使用
pmu_id	0:axp19x,1:axp209,2:axp22x,3:axp806,4:axp808,5:axp809,6:axp803,7:axp813
pmu_twi_addr	AXPxx 通信 I2C 地址
pmu_twi_id	AXPxx 挂接在主控的哪个 I2C 控制口 (0, 1, 2 ...)
pmu_irq_id	irq 号 (0 irq0,1 irq1,……)
pmu_battery_rdc	电池通路内阻, 单位 mΩ
pmu_battery_cap	电池容量,单位 mAh, 如果配置, 计量方式为库仑计方式, 否则为电压方式
pmu_batdeten	电池检查使能控制: 0:使能 1:使能
pmu_chg_ic_temp	智能充电, PMIC 温度配置选项。此功能目前没有开, 配置为 0
pmu_runtime_chgcur	设置开机时充电电流大小, 单位 mA, 仅支持:200-2800,200mA/steps
pmu_earlysuspend_chgcur	设置关屏时充电电流大小, 单位 mA, 仅支持: 200-2800,200mA/steps
pmu_suspend_chgcur	设置待机时充电电流大小, 单位 mA, 仅支持: 200-2800,200mA/steps
pmu_shutdown_chgcur	设置关机时充电电流大小, 单位 mA, 仅支持: 200-2800,200mA/steps
pmu_init_chgvol	设置充电完成时电池目标电压, 仅支持: 4100/4150/4200/4350mV
pmu_init_chgend_rate	设置充电结束时电流占恒流值的百分比: 10/20
pmu_init_chg_enabled	开机后充电使能初始值: 0: 不开充电, 1: 开充电
pmu_init_adc_freq	ADC 采样频率设定值: 100/200/400/800 Hz
pmu_init_adcts_freq	TS ADC 采样频率设定值: 100/200/400/800 Hz
pmu_init_chg_pretime	涓流充电超时时间: 40/50/60/70 分钟
pmu_init_chg_cstime	恒流超时时间: 360/480/600/720 分钟
pmu_batt_cap_correct	满足电池容量校正条件后是否校正电池容量控制 0:不校正 1:校正
pmu_bat_regu_en	充电结束时, 充电开关是否关闭: 0: 关闭 1: 不关闭
pmu_bat_para1	电池空载电压为 3.1328V 对应的电量值
pmu_bat_para2	电池空载电压为 3.2736V 对应的电量值
pmu_bat_para3	电池空载电压为 3.3440V 对应的电量值
pmu_bat_para4	电池空载电压为 3.4144V 对应的电量值
pmu_bat_para5	电池空载电压为 3.4848V 对应的电量值
pmu_bat_para6	电池空载电压为 3.5552V 对应的电量值

pmu_bat_para7	电池空载电压为 3.5904V 对应的电量值
pmu_bat_para8	电池空载电压为 3.6080V 对应的电量值
pmu_bat_para9	电池空载电压为 3.6256V 对应的电量值
pmu_bat_para10	电池空载电压为 3.6432V 对应的电量值
pmu_bat_para11	电池空载电压为 3.6608V 对应的电量值
pmu_bat_para12	电池空载电压为 3.6960V 对应的电量值
pmu_bat_para13	电池空载电压为 3.7312V 对应的电量值
pmu_bat_para14	电池空载电压为 3.7664V 对应的电量值
pmu_bat_para15	电池空载电压为 3.8016V 对应的电量值
pmu_bat_para16	电池空载电压为 3.8192V 对应的电量值
pmu_bat_para17	电池空载电压为 3.8368V 对应的电量值
pmu_bat_para18	电池空载电压为 3.8544V 对应的电量值
pmu_bat_para19	电池空载电压为 3.8720V 对应的电量值
pmu_bat_para20	电池空载电压为 3.9072V 对应的电量值
pmu_bat_para21	电池空载电压为 3.9424V 对应的电量值
pmu_bat_para22	电池空载电压为 3.9776V 对应的电量值
pmu_bat_para23	电池空载电压为 4.0128V 对应的电量值
pmu_bat_para24	电池空载电压为 4.0480V 对应的电量值
pmu_bat_para25	电池空载电压为 4.0832V 对应的电量值
pmu_bat_para26	电池空载电压为 4.1184V 对应的电量值
pmu_bat_para27	电池空载电压为 4.1360V 对应的电量值
pmu_bat_para28	电池空载电压为 4.1536V 对应的电量值
pmu_bat_para29	电池空载电压为 4.1888V 对应的电量值
pmu_bat_para30	电池空载电压为 4.224V 对应的电量值
pmu_bat_para31	电池空载电压为 4.2592V 对应的电量值
pmu_bat_para32	电池空载电压为 4.2944V 对应的电量值
pmu_usbvol_limit	USB 适配器限压功能控制 0：不使能 1：使能
pmu_usbcur_limit	USB 适配器限流功能控制 0：不使能 1：使能
pmu_usbvol	设置 USB 适配器限压值： 4000/4100/4200/4300/4400/4500/4600 4700 mV，0-不限压
pmu_usbcur	设置 USB 适配器限流值：500/900mA，0-不限流
pmu_usbvol_pc	设置 USB 连接 PC 时限压值： 4000/4100/4200/4300/4400/4500 4600/4700 mV，0-不限压
pmu_usbcur_pc	设置 USB 连接 PC 时限流值：500/900mA，0-不限流
pmu_pwroff_vol	PMU 关机时，硬件低电保护电压设置值： 2600/2700/2800/2900 /3000/3100/3200/3300 mV
pmu_pwron_vol	PMU 开机后，硬件低电保护电压设置值： 2600/2700/2800/2900 /3000/3100/3200/3300 mV
pmu_pekoff_time	长按键关机时间设置值：4000/6000/8000/10000 ms
pmu_pekoff_func	长按键功能配置项：0：长按键后关机 1：长按键后重启

pmu_pekoff_en	长按键后是否关闭 PMU：0：不关闭 1：关闭
pmu_pekoff_delay_time	长按键关机激活时间设置， 0/10/20/30/40/50/60/70 秒
pmu_peklong_time	报长按键消息时间设定值：1000/1500/2000/2500 ms
pmu_pekon_time	关机情况下按键多长时间后启动设置：128/1000/2000/3000 ms
pmu_pwrok_time	reset 启动延时时间设置值：8/16/32/64 ms
pmu_pwrok_shutdown_en	长按 reset 键 6s 是否关机，使能位(axp803 不支持此功能)
pmu_reset_shutdown_en	Reset 键重启时，是否关闭所有的 ldo/dcdc，0：不关闭，1：关闭。
pmu_battery_warning_level 1	低电报警门限 level 1 设置值百分比：5~20，每步设置 1%
pmu_battery_warning_level 2	低电报警门限 level 2 设置值百分比：0~15，每步设置 1%
pmu_restvol_time	电池电量更新时间设置值：30/60/120 s
pmu_ocv_cou_adjust_time	根据 OCV 校正电池电量更新时间值：30/60/120 s
pmu_chgled_func	CHGLED 功能控制：0：马达驱动 1：充电状态指示
pmu_chgled_type	CHGLED 作为充电状态指示时指示功能控制：0：方式 A 1：方式 B
pmu_vbusen_func	N_VBUSEN 工作方式控制：0：作为输入脚 1：作为输出脚
pmu_reset	长按键 16s 后 PMU 是否重启控制：0：不重启 1：重启
pmu_IRQ_wakeup	在关机和休眠状态下 IRQ 为低电平时是否触发开机和唤醒控制 0：不开机或不唤醒 1：开机或唤醒
pmu_hot_shutdownm	PMU 过温后是否关机 0：不关机 1：关机
pmu_inshort	是否手动设置 ACIN/VBUS 短路控制 0：PMU 自动检测 1：手动设置 ACIN 和 VBUS 为短路
pmu_temp_enable	电池温度检测使能控制：0：disable 1：enable
pmu_charge_ltf	充电下限电池温度对应的电压
pmu_charge_htf	充电上限电池温度对应的电压
pmu_discharge_ltf	关机下限电池温度对应的电压
pmu_discharge_htf	关机上限电池温度对应的电压
pmu_temp_para1	电池温度-25 度对应的电压
pmu_temp_para2	电池温度-15 度对应的电压
pmu_temp_para3	电池温度-10 度对应的电压
pmu_temp_para4	电池温度-5 度对应的电压
pmu_temp_para5	电池温度 0 度对应的电压
pmu_temp_para6	电池温度 5 度对应的电压
pmu_temp_para7	电池温度 10 度对应的电压
pmu_temp_para8	电池温度 20 度对应的电压
pmu_temp_para9	电池温度 30 度对应的电压
pmu_temp_para10	电池温度 40 度对应的电压
pmu_temp_para11	电池温度 45 度对应的电压
pmu_temp_para12	电池温度 50 度对应的电压
pmu_temp_para13	电池温度 55 度对应的电压

pmu_temp_para14	电池温度 60 度对应的电压
pmu_temp_para15	电池温度 70 度对应的电压
pmu_temp_para16	电池温度 80 度对应的电压

Confidential

4. Regulator

4.1. regulator 使用 demo

4.1.1. LDO/DCDC 电源对应表

AXP81X 电源对应表：

输出名称	ldo name	节点名称
DCDC1	axp81x_dcdc1	regulator.1
DCDC2	axp81x_dcdc2	regulator.2
DCDC3	axp81x_dcdc3	regulator.3
DCDC4	axp81x_dcdc4	regulator.4
DCDC5	axp81x_dcdc5	regulator.5
DCDC6	axp81x_dcdc6	regulator.6
DCDC7	axp81x_dcdc7	regulator.7
RTC-VCC	axp81x_rtc	regulator.8
ALDO1	axp81x_aldo1	regulator.9
ALDO2	axp81x_aldo2	regulator.10
ALDO3	axp81x_aldo3	regulator.11
DLDO1	axp81x_dldo1	regulator.12
DLDO2	axp81x_dldo2	regulator.13
DLDO3	axp81x_dldo3	regulator.14
DLDO4	axp81x_dldo4	regulator.15
ELDO1	axp81x_eldo1	regulator.16
ELDO2	axp81x_eldo2	regulator.17
ELDO3	axp81x_eldo3	regulator.18
FLDO1	axp81x_fldo1	regulator.19
FLDO2	axp81x_fldo2	regulator.20
GPIO0/LDO	axp81x_ldoio0	regulator.21
GPIO1/LDO	axp81x_ldoio1	regulator.22
SWOUT	axp81x_dc1sw	regulator.23

4.1.2. 使用示例

以 DCDC2 为例，假如 cpu cluster1 由 DCDC2 供电，并在 sysconfig 中 DCDC2 对应的 ldo2 中加入了 "vdd-cpu" 这个 supply id。

设置 DCDC2 最大输出电压值为 1.3V，需要设置目标电压值为 1V。

```
#include <linux/regulator/consumer.h>

struct regulator *regu= NULL;
int ret = 0;
regu= regulator_get(NULL, "vdd-cpu");
```

```

if (IS_ERR(regu)) {
    pr_err("%s: some error happen, fail to get regulator \n", __func__);
    goto exit;
}

//set output voltage to 1V
ret = regulator_set_voltage(regu, 1000000, 1300000);
if (0 != ret) {
    pr_err("%s: some error happen, fail to set regulator voltage!\n", __func__);
    goto exit;
}

//enalbe regulator
ret = regulator_enable(regu);
if (0 != ret) {
    pr_err("%s: some error happen, fail to enable regulator!\n", __func__);
    goto exit;
}

//disalbe regulator
ret = regulator_disable(regu);
if (0 != ret) {
    pr_err("%s: some error happen, fail to disable regulator!\n", __func__);
    goto exit;
}

//put regulator, when module exit
regulator_put(regu);

```

4.1.3. dump 节点使用方法

在串口中，输入如下命令。

```
cat /sys/class/regulator/dump
```

会将系统所有 regulator 的使用信息打印出来。

```

axp81x_dclsw : disabled  0  1600000    supply_name:
axp81x_gpio1l1do : disabled  0  3300000    supply_name:
axp81x_gpio0l1do : enabled  1  3000000    supply_name:  vcc-ctp
axp81x_fldo2 : enabled  0  900000     supply_name:
axp81x_fldo1 : disabled  0  1250000    supply_name:
axp81x_eldo3 : disabled  0  1800000    supply_name:
axp81x_dldo4 : disabled  0  2800000    supply_name:
axp81x_eldo1 : disabled  0  1200000    supply_name:
axp81x_eldo2 : enabled  1  1800000    supply_name:  vcc_dsi

```

```

axp81x_dldo3 : disabled  0  2800000  supply_name:
axp81x_dldo2 : enabled  1  4200000  supply_name: vcc-lcd-0
axp81x_dldo1 : enabled  3  2800000  supply_name: vcc-io2 vcc-wifi-io vcc-wifi-io
axp81x_aldo3 : enabled  2  3000000  supply_name: vcc-avcc
axp81x_aldo2 : enabled  3  1800000  supply_name: vdd-codec-cp vdd-codec-ldoin
axp81x_aldo1 : enabled  0  1800000  supply_name:
axp81x_rtc   : disabled  0  3000000  supply_name:
axp81x_dcdc7 : disabled  0  1000000  supply_name:
axp81x_dcdc6 : enabled  0  900000   supply_name:
axp81x_dcdc5 : enabled  0  1200000  supply_name:
axp81x_dcdc4 : enabled  1  900000   supply_name: vdd-gpu
axp81x_dcdc3 : disabled  0  840000   supply_name:
axp81x_dcdc2 : enabled  0  840000   supply_name:
axp81x_dcdc1 : enabled  6  3000000  supply_name: vcc-io1 vcc-io vcc-io vcc-io vcc-emmc

```

第一列为 regulator 的 name。

第二列为 regulator 的 state，代表使能或不使能。

第三列为 regulator 的 use_count，代表了此路 regulator 被 enable 的次数。

第四列为 regulator 的电压值。

第五列的 supply_name, 会将系统中 enable 了此路 regulator 的所有 supply id 打印出来。每个模块都有自己的 supply id，这样，可以很方便的看出，那路 LDO 有那些模块使能了。

4.2. Regulator shell 命令使用示例

AXP regulator 可以通过 shell 命令控制和设置其开关以及输出电压，各路文件节点建立在在 /sys/bus/platform/devices 目录下，AXP81X 分别为 reg-81x-cs-dcdc1 reg-81x-cs-dcdc2 reg-81x-cs-dcdc3、reg-81x-cs-dcdc4 reg-81x-cs-dcdc5 reg-81x-cs-dcdc6 reg-81x-cs-dcdc7 reg-81x-cs-rtc reg-81x-cs-aldo1、reg-81x-cs-aldo2、reg-81x-cs-aldo3、reg-81x-cs-dldo1、reg-81x-cs-dldo2、reg-81x-cs-dldo3、reg-81x-cs-dldo4、reg-81x-cs-eldo1、reg-81x-cs-eldo2、reg-81x-cs-eldo3、reg-81x-cs-fldo1、reg-81x-cs-fldo2、reg-81x-cs-gpio0ldo、reg-81x-cs-gpio1ldo。

以设置 AXP81X ALDO1 输出最大电压为 3.3V，设置目标电压为 3.0V 为例做说明。

```
//使能并设置输出电压为 3.0V
```

```
echo 3300000 > /sys/bus/platform/devices/reg-81x-cs-aldo1/max_microvolts
echo 3000000 > /sys/bus/platform/devices/reg-81x-cs-aldo1/min_microvolts
```

```
//关闭输出
```

```
echo 3300000 > /sys/bus/platform/devices/reg-81x-cs-aldo1/max_microvolts
echo 3000000 > /sys/bus/platform/devices/reg-81x-cs-aldo1/min_microvolts
echo 0 > /sys/bus/platform/devices/reg-81x-cs-aldo1/min_microvolts
```

关闭输出，在设置完电压后，在将电压设成 0，就可以关闭此路 ldo/dcdc 的输出

注意：shell 命令设置电压，有时候会失效，原因是代码里驱动通过 regulator_set_voltage 函数设置了电压的范围。

5. 电池管理

5.1. 电池信息

参照 6.3 节，打开电池信息管理部分 debug 信息。会每隔 10 秒钟将电池的信息在串口打印出来。具体信息如下。

```

charger->ic_temp = 38 //PMIC 温度值，单位摄氏度
charger->bat_temp = 30 //电池温度值，单位摄氏度
charger->vbat = 3801 //电池电压值，单位 mV
charger->ibat = 553 //电池电流值，单位 mA
charger->ocv = 3874 //OCV 电压值，单位 mV
charger->disvbat = 3801
charger->disibat = 553
power_sply = 2101 mW //电池当前的功耗
charger->rest_vol = 59 //电池当前的电量值
Axp Rdc = 131
Axp batt_max_cap = 2961
Axp coulumb_counter = 1751
Axp REG_B8 = e0
Axp OCV_percentage = 61
Axp Coulumb_percentage = 59
charger->is_on = 0 //电池当前是否在充电，1：charge，0：not charge
charger->bat_current_direction = 0 //电池当前电流方向，1：充电，0：放电
charger->charge_on = 1 //电池充电功能是否使能，1：enable，0：disable
charger->ext_valid = 0
pmu_runtime_chgcur = 1000000 //运行状态，充电电流，单位 uA
pmu_earlysuspend_chgcur = 1000000 //earlysuspend 状态，充电电流，单位 uA
pmu_suspend_chgcur = 1600000 //suspend 状态，充电电流，单位 uA
pmu_shutdown_chgcur = 1600000 //关机状态，充电电流，单位 uA

```

5.2. sysfs 文件节点

5.2.1. MFD

功能	属性	文件路径	设置值
寄存器的值	rw	/sys/bus/platform/devices/axp81x_board/axp81x_reg	无
寄存器的值	rw	/sys/bus/platform/devices/axp81x_board/axp81x_regs(未实现)	无

可通过这两个节点，读取或设置 `axp803` 各寄存器的值。具体使用方法，参考 6.2 节。

5.2.2. Power_supply (标准):

功能	属性	文件路径	设置值
电池剩余电量	r	<code>/sys/class/power_supply/battery/capacity</code>	百分比，0\1\2\……\100
电池电流大小	r	<code>/sys/class/power_supply/battery/current_now</code>	单位 uA
电池状况	r	<code>/sys/class/power_supply/battery/health</code>	Unknown 未知, "Good"好, "Overheat"过温, "Dead"坏掉, "Over voltage"过压, "Unspecified failure"错误, "Cold"冷
供电状态	r	<code>/sys/class/power_supply/battery/online</code>	0\1 : 未在供电正在供电
电池存在	r	<code>/sys/class/power_supply/battery/present</code>	0\1 : 存在\不存在
电池当前状态	r	<code>/sys/class/power_supply/battery/status</code>	Unknown 未知, "Charging"正在充电, "Discharging"放电, "Not charging"未在充电, "Full"满
电池技术	r	<code>/sys/class/power_supply/battery/technology</code>	Unknown, "NiMH", "Li-ion", "Li-poly", "LiFe", "NiCd", "LiMn"
设备类别	r	<code>/sys/class/power_supply/battery/type</code>	battery 电池, "Mains"火牛, "USB"USB
电池最大设计电压	r	<code>/sys/class/power_supply/battery/voltage_max_design</code>	单位 uV
电池最小设计电压	r	<code>/sys/class/power_supply/battery/voltage_min_design</code>	单位 uV
电池电压大小	r	<code>/sys/class/power_supply/battery/voltage_now</code>	单位 uV
电池容量	r	<code>/sys/class/power_supply/battery/charge_full_design</code>	单位 mAh
DC 是否接上	r	<code>/sys/class/power_supply/ac/present</code>	0\1 : 插上没插上
DC 是否在使用	r	<code>/sys/class/power_supply/ac/online</code>	0\1 : 未使用正在使用
设备类别	r	<code>/sys/class/power_supply/ac/type</code>	"battery"电池, "Mains"火牛, "USB"USB
DC 供电时的电流大小	r	<code>/sys/class/power_supply/ac/current_now</code>	单位 uA
DC 供电时的电压大小	r	<code>/sys/class/power_supply/ac/voltage_now</code>	单位 uV
USB 是否接上	r	<code>/sys/class/power_supply/us</code>	0\1 : 插上没插上

		b/present	
USB 是否在使用	r	/sys/class/power_supply/usb/online	0\1 : 未使用\正在使用
设备类别	r	/sys/class/power_supply/usb/type	"battery"电池,"Mains"火牛,"USB"USB
USB 供电时的电流大小	r	/sys/class/power_supply/usb/current_now	单位 uA
USB 供电时的电压大小	r	/sys/class/power_supply/usb/voltage_now	单位 uV

Confidential

6. 其它

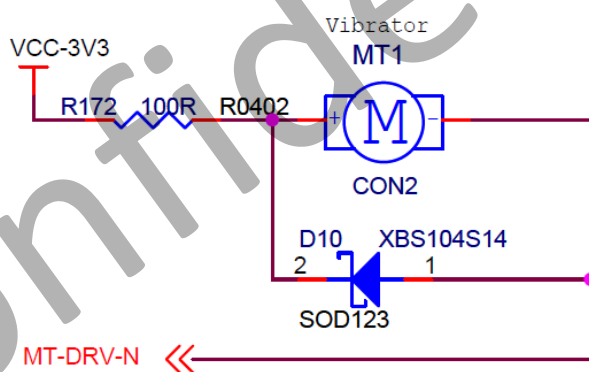
6.1. GPIO

6.1.1. AXP GPIO 对应表

原理图名称	配置表名称	GPIO 组名称	axp81x PIN No.	IO status
axp81x GPIO0	power0	GPIO_AXP(0)		IO
axp81x GPIO1	power1	GPIO_AXP(1)		IO
axp81x DC1SW	power2	GPIO_AXP(2)		O
axp81x CHGLED	power3	GPIO_AXP(3)		O
axp81x N_VBUSEN	power4	GPIO_AXP(4)		O

6.1.2. AXP GPIO 使用示例

以 power3 CHGLED 作为 moto 的控制脚为例，电路如下图。



MT-DRV-N 脚连接到了 power3 CHGLED 脚，拉低则马达震动，拉高则关闭。
首先，在 sysconfig 中 moto 的配置如下：

```

-----
;
;motor configuration
;
-----
[motor_para]
motor_used          = 1
motor_shake        = port:power3<1><<default><default><1>

```

第一列，power3 为 CHGLED 脚的配置名称。

第二列，1 代表设置为输出，0 代表输入。

第五列，1 代表默认输出高，0 代表默认输出低。

在 moto 驱动中，代码如下。

```

struct gpio_config motor_gpio;
int vibe_off;
script_item_u val;
script_item_value_type_e type;

/* 脚本解析 */
type = script_get_item("motor_para", "motor_shake", &val);
if(SCIRPT_ITEM_VALUE_TYPE_PIO != type) {
    printk(KERN_ERR "no motor_shake, ignore it!");
} else {
    motor_gpio = val.gpio;
    vibe_off = val.gpio.data;
}

/* 申请 GPIO，并设置为默认值 */
if (0 != motor_gpio.gpio) {
    if(0 != gpio_request(motor_gpio.gpio, "vibe")) {
        printk(KERN_ERR "ERROR: vibe Gpio_request is failed\n");
    }
    gpio_direction_output(motor_gpio.gpio, vibe_off);
}

/* 根据变量 on，进行输出高，输出低设置 */
if (0 != motor_gpio.gpio) {
    if(on) {
        __gpio_set_value(motor_gpio.gpio, !vibe_off);
    } else {
        __gpio_set_value(motor_gpio.gpio, vibe_off);
    }
}

/* 模块退出时，释放 GPIO */
if (0 != motor_gpio.gpio) {
    gpio_free(motor_gpio.gpio);
}

```

6.2. Shell 命令读取或修改 AXP803 寄存器

往 axp803 寄存器 0f 写入值 0x55：

```
echo 0f55 > /sys/bus/platform/devices/axp81x_board/axp81x_reg
```

读出 axp803 寄存器 0f 的值：

```
echo 0f > /sys/bus/platform/devices/axp81x_board/axp81x_reg
cat /sys/bus/platform/devices/axp81x_board/axp81x_reg
```

一次读取多个寄存器的值（未实现）：

```
echo 0f > /sys/bus/platform/devices/axp81x_board/axp81x_regs
cat /sys/bus/platform/devices/axp81x_board/axp81x_regs
```

可一次读取从 0x0f 开始的连续 20 个寄存器的值。

6.3. 打印控制

```
echo 1 > /sys/class/axppower/axpdebug //打开电池信息管理部分 debug 信息
echo 2 > /sys/class/axppower/axpdebug //打开 regulator 部分 debug 信息
echo 4 > /sys/class/axppower/axpdebug //打开 axp 中断部分 debug 信息
echo 8 > /sys/class/axppower/axpdebug //打开充电部分 debug 信息
echo 0 > /sys/class/axppower/axpdebug //关闭 debug 信息
```

此开关，实现对 debug 信息的动态控制。

Confidential