ANYKA
芯 随 你 动

版本: 1.0.2 2021 年 3 月

AnyCloud 平台

驱动对外接口说明文档

声　　明

联　系　方　式

广州安凯微电子股份有限公司

地址：广州市黄埔区知识城博文路 107 号安凯微电子 H 大厦
电话: (86)-20-3221 9000

传真: (86)-20-3221 9258

邮编: 510663

销售热线:

(86)-20-3221 9499

电子邮箱:

sales@anyka.com

主页:

http: //www. anyka.com

版 本 变 更 说 明

以下表格对于本文档的版本变更做一个简要的说明。版本变更仅限于技术内容的变更，不包括版式、格式、句法等的变更。

版本	说明	完成日期
V1.0.0	正式发布	2020 年 12 月
V1.0.1	删除 PCM 模块相关内容	2020 年 12 月
V1.0.2	添加 Efuse 模块相关内容	2021 年 3 月

目录

1 概述

本文档主要针对 AnyCloud PDK 上用户空间可访问的驱动模块，提供其相关操作接口的文档说明。另外，更为常见的模块诸如 input 子系统以及 V4L2 或者网络栈等模块本文档不做描述。

2 接口说明

2.1 GPIO

GPIO 模块控制设备上管脚的输入输出配置（输入输出的方向及电平变化、上下拉极性配置、驱动能力等），AnyCloud PDK 上 GPIO 模块已支持 Linux 上的 gpiolib 库，在用户空间中，如果想要操作对应的 GPIO 管脚的配置，可利用内核提供的节点进行操作，通过内核提供的文件节点写入具体配置来更改设备上管脚的配置，以 GPIO43 为例子,具体的操作说明如下：

- 设置使用的 gpio 号

echo 43 > /sys/class/gpio/export

- 设置方向

输出：echo out > /sys/class/gpio/gpio43/direction

输入：echo in > /sys/class/gpio/gpio43/direction

- 设置输出值

echo 1 > /sys/class/gpio/gpio43/value

echo 0 > /sys/class/gpio/gpio43/value

- 读取输入值

cat /sys/class/gpio/gpio43/value

- 设置驱动能力

echo 0 > /sys/class/gpio/gpio43/drive

设置档位范围：0~3，**注意实际的驱动能力需要参考芯片说明文档。**

- 设置上下拉极性

echo pullup > /sys/class/gpio/gpio43/pull_polarity

echo pulldown > /sys/class/gpio/gpio43/pull_polarity

- 设置上下拉使能

```
echo 1 > /sys/class/gpio/gpio43/pull_enable
echo 0 > /sys/class/gpio/gpio43/pull_enable
```

- 设置输入使能

```
echo 1 > /sys/class/gpio/gpio43/input_enable
echo 0 > /sys/class/gpio/gpio43/input_enable
```

- 设置带宽速率

```
echo fast > /sys/class/gpio/gpio43/slew_rate
echo slow > /sys/class/gpio/gpio43/slew_rate
```

示例代码可以参考驱动源码下 sample 中的 ak_drv_gpio_sample，该 sample 提供了代码方式操作 gpio 的示例，将内核 gpiolib 提供的 GPIO 控制接口用代码的方式进行调用。

该 sample 提供了配置 gpio 管脚相应的输出方向/输出值/驱动能力/上下拉配置/上下拉使能/slewrate 能力等，其参数说明如下所示：

参数(长短格式)	参数说明
-r/--read	读操作，注意读写操作只能选择一个。
-w/--write	写操作，注意读写操作只能选择一个。
-n/--gpio_no	GPIO pin 脚的标识号 从 0 开始，必须指明操作的 pin 脚。
-d/--direction	GPIO pin 脚的方向 out 表示输入 in 表示输出，如果是读操作不需要带参数，如果是写操作，必须带参数，而且参数只能是 out 或者 in。
-v/--value	GPIO pin 脚的输出，0 标识高电平，1 表示低电平，如果是读操作不需要带参数，如果是写操作，必须带参数，而且参数只能是 0 或者 1。
-c/--drive	GPIO pin 脚的驱动能力，可选 0-3 共 4 档，如果是读操作不需要带参数，如果是写操作，必须带参数，而且参数只能是 0～3。
-p/--pull_polarity	GPIO pin 脚的上下拉配置 ，pullup 标识上拉, pulldown 表示下拉，如果是读操作不需要带参数，如果是写操作，必须带参数，而且参数必须是 pullup 或者 pulldown。
-e/--pull_enable	GPIO pin 脚的上下拉使能，1 表示使能上下拉，0 表示不使能，如果是读操作不需要带参数，如果是写操作，必须带参数，而且参数必须是 0 或者 1。
-s/--slew_rate	GPIO pin 脚的 slew rate 能力，fast 表示使能，slow 表示关闭，如果是读操作不需要带参数，如果是写操作，必须带参数，而且参数必须是

参数(长短格式)	参数说明
	fast 或者 slow。
-h/--help	显示帮助信息。

注意：管脚的电气特性需要借助硬件设备进行测量。

2.2 LED

LED 用于控制设备上的 LED 的显示状态（控制不同的 LED 的亮灭状态），LED 模块利用 Linux 的 LED 框架对上层提供操作接口，在驱动加载上后，在/sys/class/leds 下会生成相应的节点，如：

```
[root@anyka /sys/class/leds]$ ls
state_led
[root@anyka /sys/class/leds]$ ls state_led/
ak_led_info    device        subsystem     uevent
brightness     max_brightness  trigger
```

上述表示该设备上存在 state_led 的 LED 灯，LED 模块对 LED 灯的状态仅提供亮、灭两种状态，所以对于本模块而言，可以通过往 brightness 节点写入 1 来点亮 LED 灯，往 brightness 节点写入 0 来熄灭 LED 灯。

示例代码可以参考驱动源码下 sample 中的 ak_drv_led_sample，该 sample 通过传入打开的灯的类型并传入对应的状态（0：熄灭，1：点亮）即可操作对应的 LED 灯，sample 的参数说明如下所示：

参数(长短格式)	参数说明
-c/--type	LED 灯的类型，可选择的类型包括但不限于 state_led/状态灯，irled/红外灯等，根据实际设备而定。
-s/--status	LED 的状态，0 表示关闭 LED 灯，1 表示打开 LED 灯。
-h/--help	显示帮助信息。

2.3 MOTOR

MOTOR 模块用于驱动电机，驱动加载后，会生成/dev/motorX（以设备的实际情况可能会有 motor0、motor1 等多个设备节点），通过 ioctl 传入不同的命令实现用户空间与内核驱动的交互。对应的模块的操作接口说明如下：

提供功能	设计功能说明
open	打开设备，获得操作句柄；
close	关闭设备，释放操作句柄

ioctl	MOTOR_PARM：获取当前 MOTOR 的运行参数。
	struct motor_parm {
	int pos; //当前的位置
	int speed_step; //步进频率
	int steps_one_circle; //一圈的总步数
	int total_steps; //转动的总步数
	int boundary_steps; //边缘保留步数，
	};

	MOTOR_SPEED_STEP：配置转动的频率。
	MOTOR_SPEED_ANGLE：配置转动的角速度。
	MOTOR_MOVE_LIMIT：边界限定转动，到达边界时，会保留 boundary_steps 的步数。传入的正数代表顺时针转动，传入的负数代表逆时针转动，绝对值代表转动的步数。
	MOTOR_MOVE_NOLIMIT：无边界限定转动，传入的正数代表顺时针转动，传入的负数代表逆时针转动，绝对值代表转动的步数。
	MOTOR_STOP：停止转动。

	MOTOR_GET_STATUS：获取当前的状态。
	struct motor_message {
	enum motor_status status;//运行状态，参考 motor_status 定义，有 stop 跟 running 两种状态。
	int pos;//当前的位置
	int speed_step; //当前的速度
	int speed_angle;//当前的角速度
	int steps_one_circle;//一个的总步数
	int total_steps; //转动的总步数
	int boundary_steps; //边缘保留步数

提供功能	设计功能说明

	int attach_timer; // 使用的硬件 timer 的标识号
	};

	MOTOR_RESET：重置，顺时针到底后转到中间位置上，传入 motor_reset_data 用于获取 reset 后的状态。
	struct motor_reset_data {
	int total_steps; //总步数
	int cur_step; //当前步数
	};

	MOTOR_MIDDLE：转到中间位置上。
	MOTOR_CRUISE：来回反复的转动。
	MOTOR_BOUNDARY：转到边界位置。

示例代码可以参考驱动源码下 sample 中的 ak_drv_motor_sample，该 sample 提供以下命令动作：

0): MOTOR_MOVE_LIMIT 有限制转动
1): MOTOR_MOVE_NOLIMIT 无限制转动
2): MOTOR_RESET 转动到初始位置
3): MOTOR_MIDDLE 转动到中间位置
4): MOTOR_CRUISE 反复转动
5): MOTOR_BOUNDARY 转动到边界
6): MOTOR_STATUS 获取当前状态
7): CMD_MOTOR_CFG 更新配置

确认好设备上连接好电机马达后加载马达驱动即可，具体的指令格式参考 sample 的--help 说明即可，其参数说明如下：

参数(长短格式)	参数说明
-n/--devno	马达的设备号，从 0 开始，依具体设备而定。
-c/--current	马达当前的位置。
-s/--speed	频率，单位时间/1 秒内的步进总数。
-x/--step	一圈的步数总和。

参数(长短格式)	参数说明
-t/--total	总运动步数。
-b/--boundary	边界步数。
-m/--cmd	命令动作。具体定义见上文 MOTOR_MOVE_LIMIT 到 CMD_MOTOR_CFG 等定义。
-a/--args	命令参数。
-h/--help	显示帮助信息。

2.4 PWM

PWM 模块用于配置管脚输出不同占空比的脉冲信号，模块使用了内核的 pwm 框架，因此可以通过内核提供的通用接口进行配置。以 pwm0 为例，具体说明如下：

- 打开 pwm 节点，执行在命令后会生成 pwm0 的节点

```
echo 0 > /sys/class/pwm/pwmchip0/export
```

- 配置周期

```
echo xxx > /sys/class/pwm/pwmchip0/pwm0/period
```

- 配置占空比

```
echo xxx > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
```

- 使能 pwm

```
echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```

- 关闭 pwm

```
echo 0 > /sys/class/pwm/pwmchip0/pwm0/enable
```

- 关闭 pwm 节点

```
echo 0 > /sys/class/pwm/pwmchip0/unexport
```

说明：AnyCloud PDK 上可支持多组 PWM 脉冲输出，具体请参考实际设备上的 PWM 相关节点的配置信息。

示例代码可以参考驱动源码下 sample 中的 ak_drv_pwm_sample，该 sample 提供了代码方式操作 PWM 的示例，其参数说明说下：

参数(长短格式)	参数说明
-n/--pwm_no	PWM 设备标识号，可选从 0-5 等，依具体设备而定。

参数(长短格式)	参数说明
	需要留意的是目前的 pwmchip 跟初始化的顺序有关，那部分的 chip 对应那部分的管脚输出，需要依据初始化顺序来定。 后续驱动会优化此问题。
-d/--duty_ns	脉冲宽度 单位是纳秒。
-p/--period_ns	信号周期 单位是纳秒。 duty_ns/period_ns 即为占空比。
-t/--time	脉冲持续时间，单位为秒。
-h/--help	显示帮助信息。

2.5 SARADC

SAR ADC 用于将模拟信号转换为数字信号，常见的应用场景如电池电压的检测及模拟按键等。模块的实现借助 Linux 的 IIO 框架，驱动加载后，直接读取节点的数据便可获取最新的采样结果。

```
[root@anyka /sys/devices/platform/soc/8000000.saradc/iio:device0]$ cat in_voltage1_raw
654
```

示例代码可以参考驱动源码下 sample 中的 ak_drv_saradc_sample，该 sample 提供了代码方式读取指定通道数据的示例，需要注意的是，不同的芯片上支持的通道数量不一致，需根据实际设备选取适当的通道，其参数说明如下：

参数(长短格式)	参数说明
-c/--channel	指定的通道数，具体通道依设备而定。
-h/--help	显示帮助信息。

2.6 Watchdog

看门狗使得系统可以在出现故障自动重启系统以提高系统的可用性。看门狗模块的文件节点为/dev/watchdog，系统提供设定看门狗的超时时间及喂狗等指令。

功能	功能说明
open	获取操作句柄。
ioctl	WDIOC_KEEPALIVE：进行喂狗。

功能	功能说明
	WDIOC_SETTIMEOUT：设置看门狗的超时时间，传入-1 时关闭看门狗，单位为秒。
	WDIOC_GETTIMEOUT：获取超时时间，单位为秒。
close	释放操作句柄。

示例代码可以参考驱动源码下 sample 中的 ak_drv_wdt_sample，该 sample 提供了代码方式进行设置看门狗超时时间及喂狗的动作，其参数说明如下：

参数(长短格式)	参数说明
-t/--timeout	看门狗的超时时间设置，单位为秒。
-f/--feedtime	看门狗的喂狗时间设置，单位为秒
-c/--feedcircle	看门狗的喂狗次数
-h/--help	显示帮助信息。

2.7 Efuse

Efuse 是一种非易失性存储器件，一旦烧录，即使断电也不会导致数据丢失，该模块提供控制命令字用于获取 Efuse 内的相关数据，其操作接口定义如下：

功能	功能说明
open	获取操作句柄。
ioctl	IOC_READ_GLOBAL_ID：获取芯片的全球唯一 ID。 全球唯一 ID 定义如下： typedef struct ak_global_id { unsigned char chip_globle_id[AK_GLOBE_ID_LEN]; } AK_GLOBAL_ID; 该 ID 是一个 64-bit 长度的数据。
close	释放操作句柄。

注意：目前仅 AnyCloud37E 支持该控制命令字。

示例代码可以参考驱动源代码下的 sample 中的 ak_drv_efuse_sample，该 sample 提供了操作指令的示例代码，其参数说明说下：

参数(长短格式)	参数说明
-c/--cmd	操作的指令，目前提供的指令类型有： 1：显示芯片的全球唯一 ID。
-h/--help	显示帮助信息。