

# Tina

Key 快速配置使用手册 V1.0

# 文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2018-4-16		初始版本

Yllwinnertec

# 目 录

1. 概述.....	4
1.1. 编写目的.....	4
1.2. 适用范围.....	4
2. Key 配置.....	5
2.1. GPIO-Key.....	5
2.1.1. 3.4 内核.....	5
2.1.2. 3.10 以及 4.4 内核.....	8
2.2. ADC Key.....	11
2.2.1. 3.4 内核.....	11
2.2.2. 3.10 以及 4.4 内核.....	13
2.3. AXP-Key.....	13
3. Declaration.....	15

Allwinner

# 1. 概述

## 1.1. 编写目的

介绍 Tina 平台 key 相关的快速配置和使用方法

## 1.2. 适用范围

该文档适用于 Allwinner Tina 平台, Allwinner R6, R11, R16, R18, R30, R40

Allwinner

## 2. Key 配置

R16,R18,R30,R40 平台均支持支持三种不同类型的 key: GPIO-Key, ADC-Key, AXP-Key  
R6,R11 没有使用我司 PMU, 没有对应的 AXP 按键  
按键相关配置根据平台不同内核会有部分差异, 下面作详细介绍

平台	内核	平台	内核	平台	内核
R6	3.10	R16	3.4	R30	4.4
R11	3.4	R18	4.4	R40	3.10

### 2.1. GPIO-Key

3.4 内核的平台, 需要改动源码文件; 而 3.10, 4.4 内核一般来说改动 dts 文件即可。

#### 2.1.1. 3.4 内核

下面以 R16 为例进行说明

支持 interrupt-key,poll-key 驱动文件如下:

```
linux-3.4/drivers/input/keyboard/gpio-keys-poll.c //gpio poll key
linux-3.4/drivers/input/keyboard/gpio-keys.c //interrupt key
linux-3.4/arch/arm/mach-sunxi/sun8i.c //提供 gpio_buttons[]为驱动文件提供 key 的定义和资源。
```

配置这种类型的 gpio-key 时, 请先查询, 当前的 gpio 是否可以用作中断功能, 如果该引脚可以用作中断功能, 则采用 interrupt 触发方式获取按键信息, 若不能用作中断功能, 则只能采用轮询的方式查询按键的状态信息。

##### 2.1.1.1. 普通 GPIO 采用 poll 方式

###### 1.修改 gpio\_buttons

根据原理图在 linux-3.4/arch/arm/mach-sunxi/sun8i.c 文件中添加 gpio\_buttons[],

例如下面按键部分原理图:

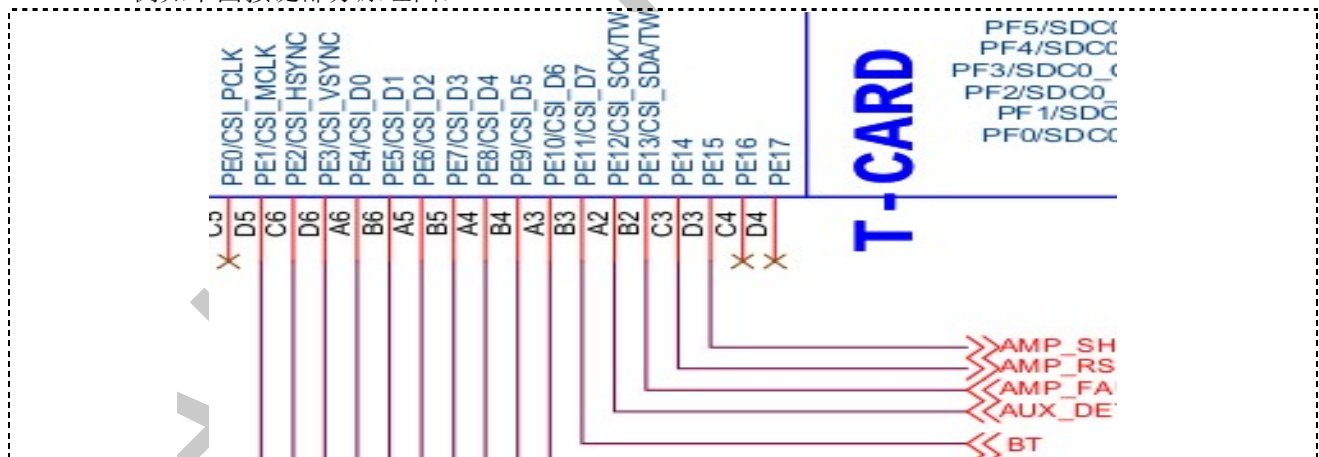


图 2-1

填写 gpio\_buttons[].gpio 这个成员变量。

```
bt key :gpio pe11
wifi key :gpio pe10
vol+ key:gpio pe9
vol- key :gpio pe8
```

将对应的 gpio 信息填写到 gpio\_buttons[].gpio.

###### 2.修改对应的键值

gpio\_buttons[].code 代表的是按键的键值, 按键的键值用户可以自行定义, 具体可以查询 linux-3.4/include/linux/input.h 文件, 并最终将 gpio\_buttons\_device 添加到 sunxi\_dev[]中。

如下所示:

```
static struct gpio_keys_button gpio_buttons[] = {
{
    .gpio          = GPIOE(11), //gpio
```

```

        .code          = KEY_UP, //键值
        .desc          = "BlueTooth",
        .active_low    = 1,
        .debounce_interval = 5,
        .type          = EV_KEY,
    },
    {
        .gpio          = GPIOE(10),
        .code          = KEY_DOWN,
        .desc          = "WiFi",
        .active_low    = 1,
        .debounce_interval = 5,
        .type          = EV_KEY,
    },
    {
        .gpio          = GPIOE(9),
        .code          = KEY_VOLUMEUP,
        .desc          = "VolumeUp",
        .active_low    = 1,
        .debounce_interval = 5,
        .type          = EV_KEY,
    },
    {
        .gpio          = GPIOE(8),
        .code          = KEY_VOLUMEDOWN,
        .desc          = "VolumeDown",
        .active_low    = 1,
        .debounce_interval = 5,
        .type          = EV_KEY,
    },
};

static struct gpio_keys_platform_data gpio_buttons_data = {
    .buttons          = gpio_buttons,
    .nbuttons         = ARRAY_SIZE(gpio_buttons),
};

static struct platform_device gpio_buttons_device = {
    .name = "gpio-keys-polled",
    .id = -1,
    .dev = {
        .platform_data = &gpio_buttons_data,
    }
};

```

### 3.确认驱动是否被选中

确认 `gpio_keys_polled.c` 是否编译进系统, 在 `tina` 目录下执行 `make kernel_menuconfig`

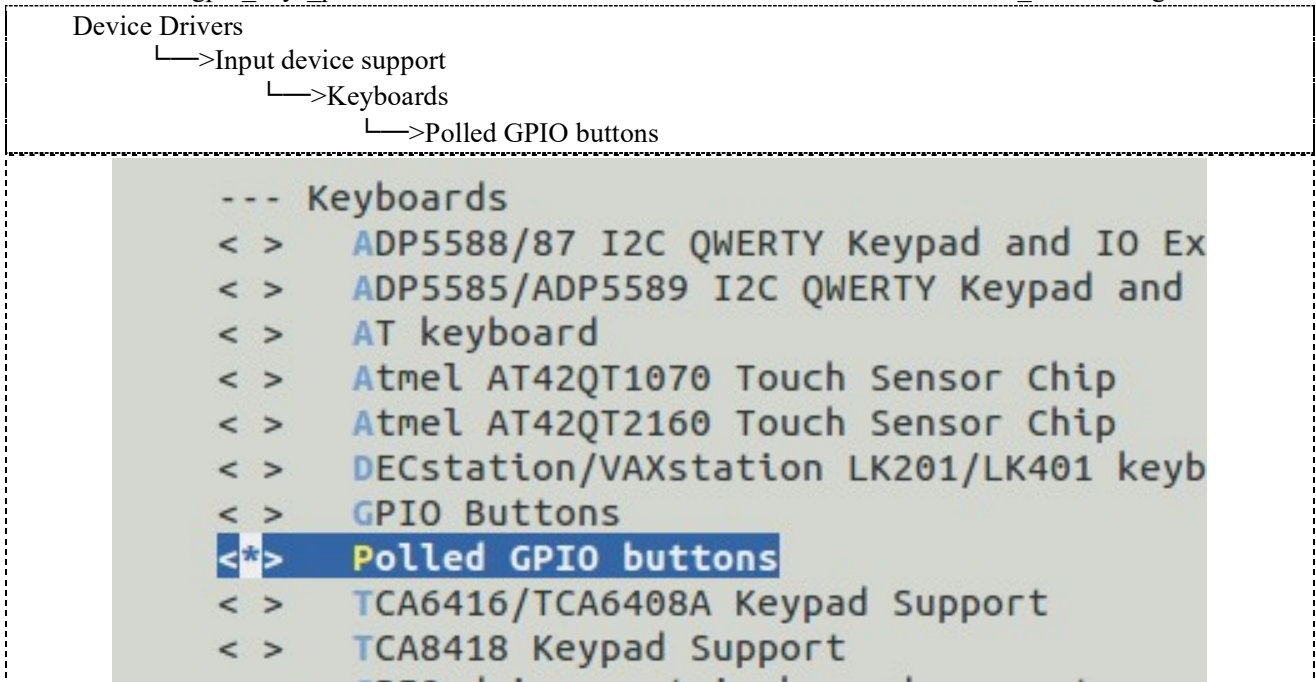


图 2-2

#### 2.1.1.2. 普通 GPIO 采用中断方式

跟采用 `poll` 方式一样, 也需要完成如下步骤:

##### 1.填写 `gpio_buttons[]`结构体

具体见普通 `gpio` 采用 `poll` 方式

##### 2.更改对应的键值

具体见普通 `gpio` 采用 `poll` 方式。

##### 3.确认 `device` 名称

修改 `gpio_buttons_device.name= "gpio-keys"`

这个名字必须跟 `gpio-keys.c` 中的 `gpio_keys_device_driver.driver.name` 一致, 否则驱动无法跟 `device` 正确匹配, 会导致驱动无法正常加载。

```
static struct platform_device gpio_buttons_device = {
    .name = "gpio-keys",
    .id = 0,
    .num_resources = 0,
    .dev = {
        .platform_data = &gpio_buttons_data,
    }
};
```

#### 4.确认驱动是否被选中

在 tina 目录下执行 make kernel menuconfig. 确保 Polled GPIO buttons 被选上.

```
Device Drivers
├──>Input device support
│   ├──>Keyboards
│       └──>GPIO Buttons
```

```
--- Keyboards
< > ADP5588/87 I2C QWERTY Keypad and IO Exp
< > ADP5585/ADP5589 I2C QWERTY Keypad and I
< > AT keyboard
< > Atmel AT42QT1070 Touch Sensor Chip
< > Atmel AT42QT2160 Touch Sensor Chip
< > DECstation/VAXstation LK201/LK401 keybo
< * > GPIO Buttons
< > Polled GPIO buttons
< > TCA6416/TCA6408A Keypad Support
< > TCA8418 Keypad Support
```

图 2-3

#### 2.1.1.3. Sunxi GPIO Key

这同样采用中断模式的 GPIO Key, 但并不需要改动太多代码, 仅仅需要在 sys\_config 中配置相关 gpio 信息, 具体驱动请参考:

```
linux-3.4/drivers/input/keyboard/sunxi-gpiokey.c
```

对应 sys\_config 中需要有如下配置:

```
[gpio_power_key]
key_used = 1
key_io    = port:PB03<0><1><default><default>
```

#### 2.1.2. 3.10 以及 4.4 内核

3.10 以及 4.4 内核的按键相关配置是一样的。

涉及到的驱动文件位于如下位置:

```
drivers/input/keyboard/
arch/arm/boot/dts/平台代号.dtsi
```

其中 drivers/input/keyboard/目录下的相关文件为驱动文件, 而平台名称.dtsi 为设备树文件, 例如 R40 的 dts 文件 sun8iw11p1.dtsi, 下面以 R40 为例进行说明

支持 interrupt-key,poll-key 驱动文件如下:

```
drivers/input/keyboard/gpio-keys-polled.c //gpio poll key
drivers/input/keyboard/gpio-keys.c //interrupt key
arch/arm/boot/dts/sun8iw11p1.dtsi
```

配置这种类型的 gpio-key 时, 请先查询, 当前的 gpio 是否可以用作中断功能, 如果该引脚可以用作中断功能, 则采用 interrupt 触发方式获取按键信息, 若不能用作中断功能, 则只能采用轮询的方式查询按键的状态信息。

#### 2.1.2.1. 普通 GPIO 采用 poll 方式

##### 1.修改设备树文件

根据原理图 arch/arm/boot/dts/sun8iw11p1.dtsi 文件中添加对应的 gpio  
例如音量加减键分别用到 PH5,PH6 这两个 GPIO, 则修改方法如下:

```
gpio-keys {
    compatible = "gpio-keys";
```



```

    vol-down-key {
        gpios = <&pio PH 5 1 2 2 1>;
        linux,code = <114>;
        label = "volume down";
        debounce-interval = <10>;
    };
    vol-up-key {
        gpios = <&pio PH 6 1 2 2 1>;
        linux,code = <115>;
        label = "volume up";
        debounce-interval = <10>;
    };
};

```

## 2. 确认驱动是否被选中

确认 `gpio_keys_polled.c` 是否编译进系统，在 `tina` 目录下执行 `make kernel_menuconfig`

```

Device Drivers
├──>Input device support
│   ├──>Keyboards
│   │   └──>Polled GPIO buttons

```

```

--- Keyboards
< > ADP5588/87 I2C QWERTY Keypad and IO Ex
< > ADP5585/ADP5589 I2C QWERTY Keypad and
< > AT keyboard
< > Atmel AT42QT1070 Touch Sensor Chip
< > Atmel AT42QT2160 Touch Sensor Chip
< > DECstation/VAXstation LK201/LK401 keyb
< > GPIO Buttons
< * > Polled GPIO buttons
< > TCA6416/TCA6408A Keypad Support
< > TCA8418 Keypad Support

```

图 2-4

### 2.1.2.2. 普通 GPIO 采用中断方式

跟采用 `poll` 方式一样，也需要完成如下步骤：

#### 1. 填写 `gpio_buttons[]` 结构体

具体见普通 `gpio` 采用 `poll` 方式

#### 2. 修改设备树文件

具体见普通 `gpio` 采用 `poll` 方式。

#### 3. 确认驱动是否被选中

在 `tina` 目录下执行 `make kernel_menuconfig`，确保 `Polled GPIO buttons` 被选上。

```

Device Drivers
├──>Input device support
│   ├──>Keyboards
│   │   └──>GPIO Buttons

```

```
--- Keyboards
< > ADP5588/87 I2C QWERTY Keypad and IO Exp
< > ADP5585/ADP5589 I2C QWERTY Keypad and I
< > AT keyboard
< > Atmel AT42QT1070 Touch Sensor Chip
< > Atmel AT42QT2160 Touch Sensor Chip
< > DECstation/VAXstation LK201/LK401 keybo
< * > GPIO Buttons
< > Polled GPIO buttons
< > TCA6416/TCA6408A Keypad Support
< > TCA8418 Keypad Support
```

图 2-5

## 2.2. ADC Key

R6 使用 TP 模块的 ADC；而 R11,R16,R18,R30,R40 均使用 LRADC 用于检测按键。key-adc 有如下特性：

1. Support voltage input range between 0 to 2V
2. Support sample rate up to 250Hz, 可以配置为 32Hz,62Hz,125Hz,250Hz,R40 sdk 中默认配置为 250Hz
3. 当前 key-adc 最大可以支持到 13 个按键(0.15V 为一个档), 通常情况下, 建议 adc key 最大不要超过 8 个(0.2V 为一档), 否则由于采样误差等因素存在, 会很容易出现误判的情况。

如下图所示,adc-key 检测原理是当有按键被按下或者抬起时, KEYADC 控制器(6bit)检测到电压变化后, 经过 KEYADC 内部的逻辑控制单元进行比较运算后, 产生相应的中断给 CPU,同时电压的变化值会通过 KEYADC 内部的数据寄存器 (0~0x3f)来体现

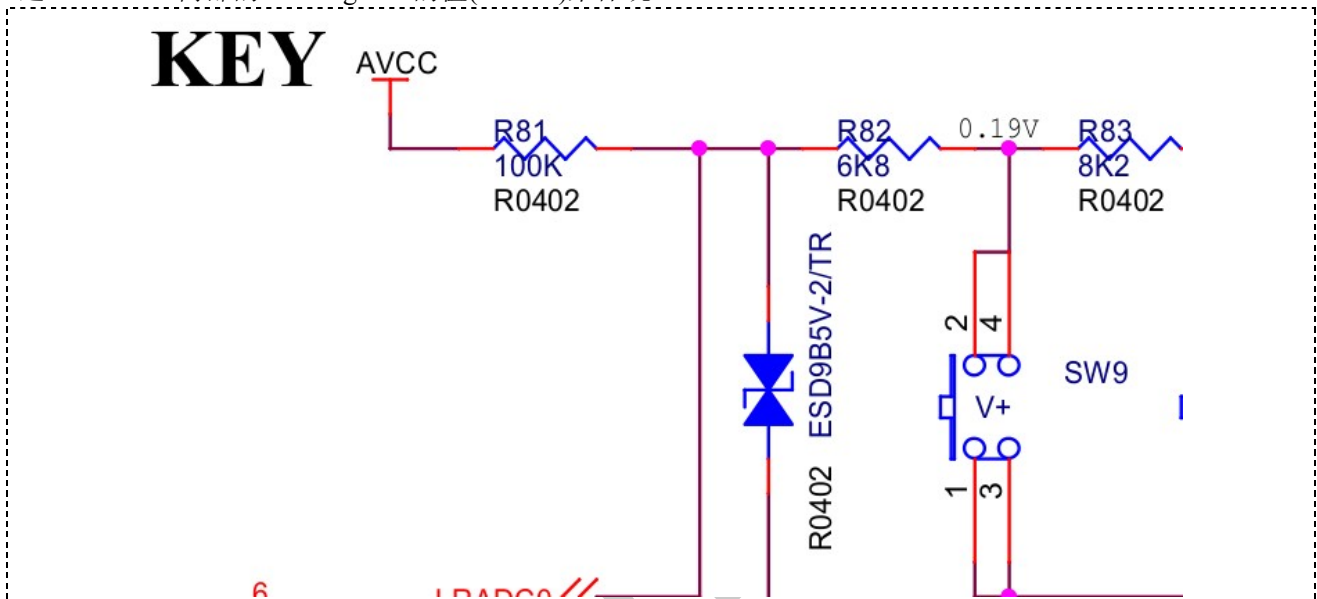


图 2-6

3.4 内核与 3.10, 4.4 内核修改的内容有差异, 下面分别作介绍:

### 2.2.1. 3.4 内核

同样以 R16 为例, 驱动文件:

linux-3.4/drivers/input/keyboard/sunxi-keyboard.c

linux-3.4/drivers/input/keyboard/sun8i-keyboard.h

1. 如果使用 adc-key, 请先确保 softwinner KEY BOARD support 有被选择上

```
Device Drivers
├──>Input device support
│   ├──>Keyboards
│       └──>softwinner KEY BOARD support
```

```
< > MELFAS MCS Touchkey
< > Freescale MPR121 Touchkey
< > Newton keyboard
< > OpenCores Keyboard Controller
< > Samsung keypad support
< > Stowaway keyboard
< > Sun Type 4 and Type 5 keyboard
< > TI OMAP4 keypad support
< > XT keyboard
* softwinner KEY BOARD support
```

图 2-7

## 2. 在 adc-key 驱动中涉及两个重要的结构体

```
static unsigned char keypad_mapindex[64] = {
    0,0,0,0,0,0,0,0,    /* key 1,  0-7   */
    1,1,1,1,1,1,1,1,    /* key 2,  8-14  */
    2,2,2,2,2,2,2,2,    /* key 3,  15-21 */
    3,3,3,3,3,3,3,3,    /* key 4,  22-27 */
    4,4,4,4,4,4,4,4,    /* key 5,  28-33 */
    5,5,5,5,5,5,5,5,    /* key 6,  34-39 */
    6,6,6,6,6,6,6,6,    /* key 7,  40-49 */
    7,7,7,7,7,7,7,7,7,7 /* key 8,  50-63 */
};
static unsigned int sunxi_scankeycodes[KEY_MAX_CNT] = {
    [0] = KEY_VOLUMEUP,
    [1] = KEY_VOLUMEDOWN,
    [2] = KEY_HOME,
    [3] = KEY_ENTER,
    [4] = KEY_MENU,
    [5] = KEY_RESERVED,
    [6] = KEY_RESERVED,
    [7] = KEY_RESERVED,
    [8] = KEY_RESERVED,
    [9] = KEY_RESERVED,
    [10] = KEY_RESERVED,
    [11] = KEY_RESERVED,
    [12] = KEY_RESERVED,
};
```

keypad\_mapindex[]数组的值跟 LRADC\_DATA0(0x0C)的值是对应的，表示的具体意义是：

key\_val(lradc\_data0 寄存器的值)在 0~7 范围内时，表示的是操作 key1，依此类推，key\_val 为 8~14 的范围之内时，表示的操作 key2。正常来说，按照 R16 推荐的硬件设计，该数组内无需任何更改，当个别情况下，如遇到 adc-key input 上报的 keycode 有异常时，需要依次按下每个按键同时读取 key\_val 的值来修正 keypad\_mapindex[]。

sunxi\_scankeycodes[]: 该数组的意义为 sunxi\_scankeycodes[\*] 标示的是具体的某一个 key，用户可以修改其中的 keycode。

例如，key\_val 等于 25，则根据 keypad\_mapindex 得到 scancode 为 4，再由 sunxi\_scankeycodes 得到键值为 KEY\_MENU

keypad\_mapindex 这个数组可以通过 sys\_config 进行配置：

```
[key_para]
key_used   = 1
key_cnt    = 5
key1_vol   = 300
key2_vol   = 600
key3_vol   = 1000
key4_vol   = 1500
key5_vol   = 1800
```

这个配置下转换得到的 keypad\_mapindex 数组是：

```
static unsigned char keypad_mapindex[64] = {
    0,0,0,0,0,0,0,0,0,0,    /* key 1,  0-9   */
    1,1,1,1,1,1,1,1,1,1,    /* key 2,  10-19  */
    2,2,2,2,2,2,2,2,2,2,2,2, /* key 3,  20-32 */
    3,3,3,3,3,3,3,3,3,3,3,3,3,3, /* key 4,  33-48 */
    4,4,4,4,4,4,4,4,4,4,4,4,4,4, /* key 5,  49-58 */
    5,5,5,5,5,5,5,5,5,5,5,5,5,5, /* key 6,  59-63 */
};
```

## 2.2.2. 3.10 以及 4.4 内核

下面以 R40 为例进行说明，驱动文件：

linux-3.10/drivers/input/keyboard/sunxi-keyboard.c

linux-3.10/arch/arm/boot/dts/sun8iw11p1.dtsi

3. 如果使用 adc-key, 请先确保 softwinner KEY BOARD support 有被选择上

```
Device Drivers
├──>Input device support
│   ├──>Keyboards
│       └──>softwinner KEY BOARD support
```

```
< > MELFAS MCS Touchkey
< > Freescale MPR121 Touchkey
< > Newton keyboard
< > OpenCores Keyboard Controller
< > Samsung keypad support
< > Stowaway keyboard
< > Sun Type 4 and Type 5 keyboard
< > TI OMAP4 keypad support
< > XT keyboard
* softwinner KEY BOARD support
```

图 2-8

4. 在 adc-key 驱动中涉及两个重要的结构体

```
static unsigned char keypad_mapindex[64];
sunxi_scankeycodes[KEY_MAX_CNT];
```

5. 设备树文件 sun8iw11p1.dtsi

```
keyboard0:keyboard{
    compatible = "allwinner,keyboard_2000mv";
    reg = <0x0 0x01c24400 0x0 0x400>;
    interrupts = <GIC_SPI 30 IRQ_TYPE_NONE>;
    status = "okay";
    key_cnt = <5>;
    key0 = <190 115>;
    key1 = <390 114>;
    key2 = <600 139>;
    key3 = <800 28>;
    key4 = <980 102>;
};
```

驱动初始化时，会读取设备树中相关属性，其中 key\_cnt 表示配置的按键个数，keyX 配置对应的 ADC 值以及键值。根据这些属性，会重新设置 keypad\_mapindex 数组，以及 sunxi\_scankeycodes 数组。

keypad\_mapindex[] 数组的值跟 ADC 值是对应的，6bitADC, 范围 0~63，表示的具体意义是：

key\_val 在 0~190 范围内时，表示的是操作 key0, 以此类推，key\_val 为 191~390 范围内时，表示的是操作 key1

sunxi\_scankeycodes[]: 该数组的意义为 sunxi\_scankeycodes[\*] 表示的是具体的某一个 key, 用户可以修改设备树来改变 keycode.

例如，默认配置下，key\_val 等于 300，则根据 keypad\_mapindex 得到 scancode 为 1，再由 sunxi\_scankeycodes 得到键值 114

## 2.3. AXP-Key

在 R16,R18,R30,R40 平台中, 均使用了我司 PMU, 虽然型号不一样, 但均有提供 power-key, 这是由 PMU 识别该按键, 并在驱动中上报相关事件, power-key 驱动源码在电源驱动源码同目录下。用户不需要做任何更改, 这部分是可以直接使用的。

例如 R40 对应的设备节点:

```
root@TinaLinux:/# ll /dev/input/event2
crw-r--r--  1 root  root   13, 66 Jan  1 12:15 /dev
root@TinaLinux:/# cat /sys/class/input/input2/name
axp22-powerkey
```

图 2-9

上报的键值为 KEY\_POWER 116

另外, PMU 还提供了 AXP GPIO, 根据不同型号的 PMU, GPIO 数量可能不一样, 但 gpio number 均以 1024 开始, 例如 AXP GPIO0 的 gpio number 为 1024, AXP GPIO1 的 gpio number 为 1025。它们均能当做普通 gpio 使用, 因此按照 GPIO-Key 章节的描述去操作它们即可

需要注意的是, sys\_config 中 axp gpio 的命名与普通 gpio 有些区别, 例如 AXP gpio0:

port:power0<1><0><default><0>



### **3. Declaration**

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgment to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.

Allwinner