Tina Linux Key 快速配置
使用指南

版本号: 1.1
发布日期: 2021.03.30

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.11.02	AWA1611	新建初始版本
1.1	2021.03.30	AWA1611	增加 Linux-5.4 和 R528/D1 的描述

版权所有 © 珠海全志科技股份有限公司。保留一切权利

目　录

1 前言 **1**
1.1 文档简介 1
1.2 目标读者 1
1.3 适用范围 1

2 模块介绍 **2**
2.1 Key 配置 2
2.2 相关术语介绍 2
2.2.1 软件术语 2

3 GPIO-Key **3**
3.1 4.4 以及 4.9 内核 3
3.1.1 普通 GPIO 采用 poll 方式 4
3.1.2 普通 GPIO 采用中断方式 5
3.2 5.4 内核 6
3.2.1 普通 GPIO 采用 poll 方式 7
3.2.2 普通 GPIO 采用中断方式 7
3.2.3 矩阵键盘 8

4 ADC-Key **12**
4.1 4.4 以及 4.9 内核 12
4.1.1 LRADC-Key 12
4.1.2 GPADC-Key 15
4.2 5.4 内核 19
4.2.1 LRADC-Key 19
4.2.2 GPADC-Key 21

5 AXP-Key **23**
5.1 4.9 内核 23
5.2 5.4 内核 25

版权所有 © 珠海全志科技股份有限公司。保留一切权利

插　图

3-1 GPIO-Key 配置图 3
3-2 linux4.4/4.9 轮询按键配置图 5
3-3 linux4.4/4.9 中断按键配置图 6
3-4 矩阵按键硬件原理图 9
3-5 矩阵键盘配置图 11
4-1 LRADC 按键原理图 12
4-2 linux4.4/4.9 LRADC 按键配置图 13
4-3 GPADC 按键硬件图 16
4-4 linux4.4/4.9 GPADC 按键配置图 16
4-5 Linux-5.4 LRADC 按键配置图 21
4-6 Linux-5.4 GPADC 配置图 22
5-1 AXP 按键节点图 23
5-2 AXPKEY 配置图 24

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1 前言

1.1 文档简介

本文介绍 Tina 平台 key 相关的快速配置和使用方法。

1.2 目标读者

Allwinner key 驱动驱动层/应用层的使用/开发/维护人员。

1.3 适用范围

表 1-1：适用产品列表

产品名称	内核版本	平台架构
R18	Linux-4.4	cortex-a53(64 位)
R30	Linux-4.4	cortex-a53(64 位)
R328	Linux-4.9	cortex-a7(32 位)
R329	Linux-4.9	cortex-a53(64 位)
R818	Linux-4.9	cortex-a53(64 位)
R818B	Linux-4.9	cortex-a53(64 位)
MR813	Linux-4.9	cortex-a53(64 位)
MR813B	Linux-4.9	cortex-a53(64 位)
R528	Linux-5.4	cortex-a7(32 位)
D1	Linux-5.4	risc-v(64 位)
H133	Linux-5.4	cortex-a7(32 位)

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2 模块介绍

2.1 Key 配置

Allwinner 平台支持三种不同类型的 Key：GPIO-Key，ADC-Key，AXP-Key。其中，GPIO-Key 又包括普通的 gpio 按键和矩阵键盘。

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。

说明

若板子上没有使用我司的带有按键功能的 *PMU*，则就没有对应的 *AXP* 按键。

2.2 相关术语介绍

2.2.1 软件术语

表 2-1：Key 软件术语列表

术语	解释说明
Key	按键
GPIO-Key	使用 GPIO 检测按键的设备
ADC	模数转换器
ADC-Key	通过 ADC 读取电压检测按键的设备
LRADC	精度为 6 位的单通道 ADC
GPADC	精度为 12 位的多通道 ADC
PMU	电源管理单元
AXP-Key	连接在电源芯片的按键

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3 GPIO-Key

3.1 4.4 以及 4.9 内核

4.4 以及 4.9 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：

其中，64 位平台和 32 位平台的 dts 文件位置是不一样的。

```
lichee/linux-*/arch/arm/boot/dts/平台代号.dtsi            //32位平台的dts文件位置
lichee/linux-*/arch/arm64/boot/dts/sunxi/平台代号.dtsi    //64位平台的dts文件位置
```

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 R328 的 dts 文件 sun8iw18p1.dtsi，下面以 R328 为例进行说明。

支持 interrupt-key,poll-key 驱动文件如下：

```
lichee/linux-*/drivers/input/keyboard/gpio-keys-polled.c   //gpio poll key
lichee/linux-*/drivers/input/keyboard/gpio-keys.c          //interrupt key
```

R328 的 dts 文件：

```
lichee/linux-4.9/arch/arm/boot/dts/sun8iw18p1.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。

GOIO-Key 的硬件配置图参考下图所示：

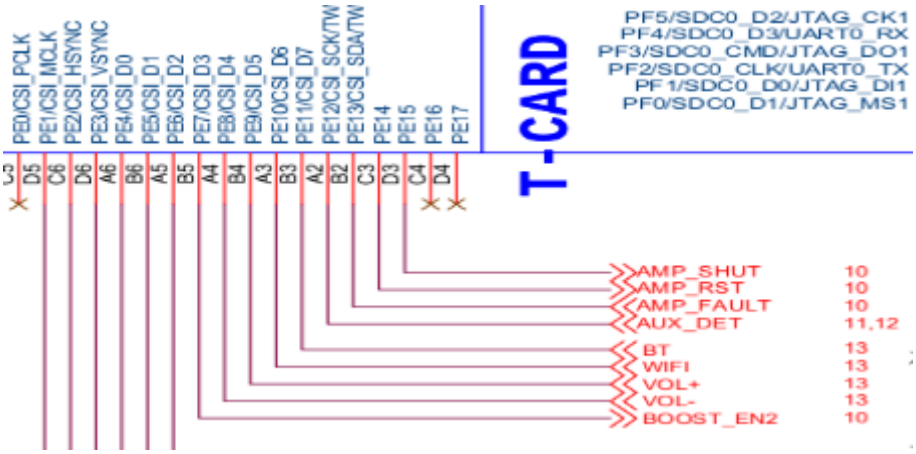

图 3-1：GPIO-Key 配置图

3.1.1 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/sun8iw18p1.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";
        status = "okay";
        vol-down-key {
             gpios = <&pio PH 5 1 2 2 1>;
             linux,code = <114>;
             label = "volume down";
             debounce-interval = <10>;
             wakeup-source = <0x1>;
        };
        vol-up-key {
             gpios = <&pio PH 6 1 2 2 1>;
             linux,code = <115>;
             label = "volume up";
             debounce-interval = <10>;
        };
};
```

- compatible：用于匹配驱动。
- status：是否加载设备。
- vol-down-key：每一个按键都是单独的一份配置，需要分别区分开来。
- gpios：GPIO 口配置。
- linux,code：这个按键对应的 input 键值。
- label：单个按键对应的标签。
- debounce-interval：消抖时间，单位为 us。
- wakeup-source：是否作为唤醒源，配置了这个项的按键可以作为唤醒源唤醒系统。

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，需要将 Polled GPIO buttons 选成 “[*]”。

```
Device Drivers
     └->Input device support
             └->Keyboards
                    └->Polled GPIO buttons
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
--- Keyboards
< >   ADP5588/87 I2C QWERTY Keypad and IO Expander
< >   ADP5585/ADP5589 I2C QWERTY Keypad and IO Expander
< >   AT keyboard
< >   Atmel AT42QT1070 Touch Sensor Chip
< >   Atmel AT42QT2160 Touch Sensor Chip
< >   DECstation/VAXstation LK201/LK401 keyboard
< >   GPIO Buttons
<*>   Polled GPIO buttons
< >   TCA6416/TCA6408A Keypad Support
< >   TCA8418 Keypad Support
< >   GPIO driven matrix keypad support
< >   Maxim MAX7359 Key Switch Controller
< >   MELFAS MCS Touchkey
```

图 3-2: linux4.4/4.9 轮询按键配置图

3.1.2 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. 修改设备树文件

可见普通 gpio 采用 poll 方式，不一样的是这里 gpio 采用中断的方式。

```
gpio-keys {
        compatible = "gpio-keys";
        status = "okay";
        vol-down-key {
             gpios = <&pio PH 5 6 2 2 1>;
             linux,code = <114>;
             label = "volume down";
             debounce-interval = <10>;
             wakeup-source = <0x1>;
        };
        vol-up-key {
             gpios = <&pio PH 6 6 2 2 1>;
             linux,code = <115>;
             label = "volume up";
             debounce-interval = <10>;
        };
};
```

- compatible：用于匹配驱动。
- status：是否加载设备。
- vol-down-key：每一个按键都是单独的一份配置，需要分别区分开来。
- gpios：GPIO 口配置。
- linux,code：这个按键对应的 input 键值。
- label：单个按键对应的标签。

- debounce-interval：消抖时间，单位为 us。
- wakeup-source：是否作为唤醒源，配置了这个项的按键可以作为唤醒源唤醒系统。

2. 确认驱动是否被选中

在 tina 目录下执行 make kernel_menuconfig，确保 GPIO Buttons 被选上。

```
Device Drivers
     └->Input device support
             └->Keyboards
                    └->GPIO Buttons
```

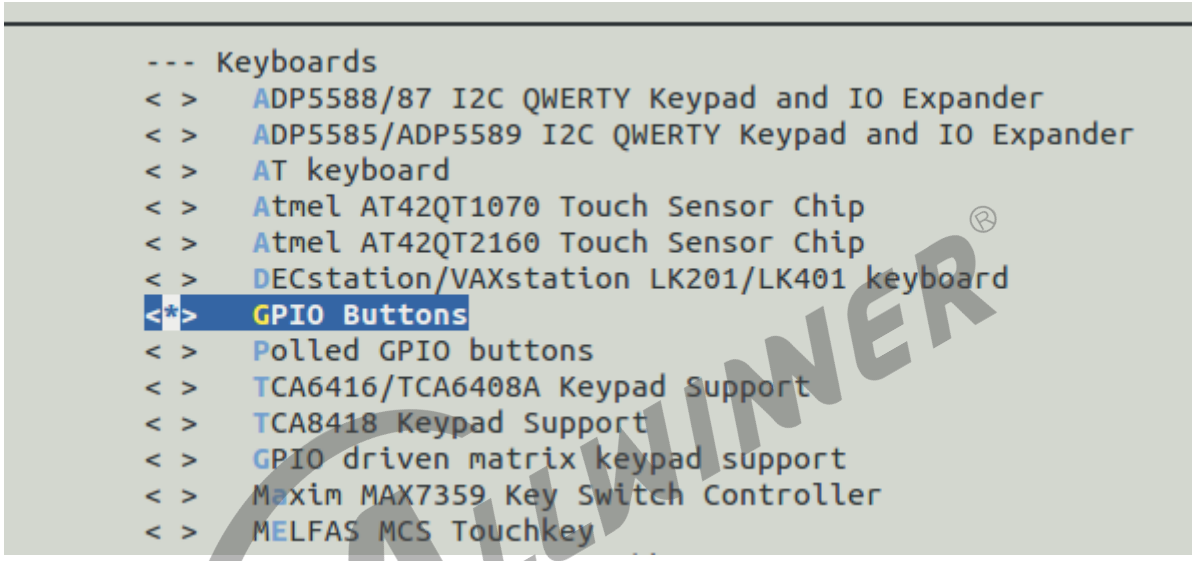

图 3-3: linux4.4/4.9 中断按键配置图

3.2 5.4 内核

Linux-5.4 内核相对 4.4/4.9 来说，GPIO 子系统有所变化，因此 dts 的配置也不大一样。

dts 文件位置：

```
lichee/linux-5.4/arch/arm/boot/dts/平台代号.dtsi          //32位平台的dts文件位置
lichee/linux-5.4/arch/riscv/boot/dts/sunxi/平台代号.dtsi   //risc-v架构平台的dts文件位置
```

其中 drivers/input/keyboard/目录下的相关文件为驱动文件。

支持 interrupt-key,poll-key 驱动文件如下：

```
lichee/linux-5.4/drivers/input/keyboard/gpio-keys-polled.c  //gpio poll key
lichee/linux-5.4/drivers/input/keyboard/gpio-keys.c         //interrupt key
```

3.2.1 普通 GPIO 采用 poll 方式

1. 修改设备树文件

```
gpio-keys {
        compatible = "gpio-keys";
        status = "okay";
        vol-down-key {
             gpios = <&pio PH 5 GPIO_ACTIVE_LOW>;
             linux,code = <114>;
             label = "volume down";
             debounce-interval = <10>;
             wakeup-source = <0x1>;
        };
        vol-up-key {
             gpios = <&pio PH 6 GPIO_ACTIVE_LOW>;
             linux,code = <115>;
             label = "volume up";
             debounce-interval = <10>;
        };
};
```

- compatible：用于匹配驱动。
- status：是否加载设备。
- vol-down-key：每一个按键都是单独的一份配置，需要分别区分开来。
- gpios：GPIO 口配置。
- linux,code：这个按键对应的 input 键值。
- label：单个按键对应的标签。
- debounce-interval：消抖时间，单位为 us。
- wakeup-source：是否作为唤醒源，配置了这个项的按键可以作为唤醒源唤醒系统。

与 4.4/4.9 相比，主要是 gpios 配置方式变化了。

2. 选上驱动

GPIO-Key 轮询驱动 kernel_menuconfig 路径：

```
Device Drivers
    └>Input device support
        └>Keyboards
            └>Polled GPIO buttons
```

3.2.2 普通 GPIO 采用中断方式

1. 修改设备树文件

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

```
        vol-down-key {
             gpios = <&pio PH 5 GPIO_ACTIVE_LOW>;
             linux,code = <114>;
             label = "volume down";
             debounce-interval = <10>;
             wakeup-source = <0x1>;
        };
        vol-up-key {
             gpios = <&pio PH 6 GPIO_ACTIVE_LOW>;
             linux,code = <115>;
             label = "volume up";
             debounce-interval = <10>;
        };
};
```

- compatible：用于匹配驱动。
- status：是否加载设备。
- vol-down-key：每一个按键都是单独的一份配置，需要分别区分开来。
- gpios：GPIO 口配置。
- linux,code：这个按键对应的 input 键值。
- label：单个按键对应的标签。
- debounce-interval：消抖时间，单位为 us。
- wakeup-source：是否作为唤醒源，配置了这个项的按键可以作为唤醒源唤醒系统。

2. 选上驱动

GPIO-Key 中断驱动 kernel_menuconfig 路径：

```
Device Drivers
    └>Input device support
        └>Keyboards
            └>GPIO Button
```

3.2.3 矩阵键盘

当需要使用大量按键的时候，如果单独给每一个按键配一个 GPIO 的话，那 GPIO 是远远不够的。这时，可以使用矩阵键盘的方式，使用 N 个 GPIO，就可以最大支持 N*N 个按键。

矩阵按键的硬件原理图如下所示：

图 3-4: 矩阵按键硬件原理图

⚠ 警告

矩阵键盘的 GPIO 建议使用 SOC 自带的 IO 口，不使用扩展 IO 芯片的 IO 口。因为矩阵键盘扫描按键的时间比较短，而扩展 IO 芯片的 IO 是通过 I2C/UART 等等的总线去修改 IO 状态的，修改一次状态时间比较长，可能会导致矩阵按键扫描按键检测失败的。

这里以 R528 为示例，dts 为：

```
lichee/linux-5.4/arch/arm/boot/dts/sun8iw20p1.dtsi
```

驱动文件为：

```
lichee/linux-5.4/drivers/input/keyboard/matrix_keypad.c
```

1. 修改设备树文件

根据 R528 原理图来添加对应行和列的 gpio，分别写在 row-gpios 和 col-gpios，详细设备树文件为：

```
        matrix_keypad:matrix-keypad {
                compatible = "gpio-matrix-keypad";
                keypad,num-rows = <6>;
```

```
keypad,num-columns = <7>;
row-gpios = <&pio PG 17 GPIO_ACTIVE_LOW
        &pio PG 18 GPIO_ACTIVE_LOW
        &pio PG 1 GPIO_ACTIVE_LOW
        &pio PG 2 GPIO_ACTIVE_LOW
        &pio PG 3 GPIO_ACTIVE_LOW
        &pio PG 4 GPIO_ACTIVE_LOW>;
col-gpios = <&pio PG 0 GPIO_ACTIVE_LOW
        &pio PG 5 GPIO_ACTIVE_LOW
        &pio PG 12 GPIO_ACTIVE_LOW
        &pio PG 14 GPIO_ACTIVE_LOW
        &pio PG 6 GPIO_ACTIVE_LOW
        &pio PG 8 GPIO_ACTIVE_LOW
        &pio PG 10 GPIO_ACTIVE_LOW>;
linux,keymap = <
        0x00000001 //row 0 col0
        0x00010002
        0x00020003
        0x00030004
        0x00040005
        0x00050006
        0x00060007
        0x01000008 //row 1 col0
        0x01010009
        0x0102000a
        0x0103000b
        0x0104000c
        0x0105000d
        0x0106000e
        0x0200000f //row 2 col0
        0x02010010
        0x02020011
        0x02030012
        0x02040013
        0x02050014
        0x02060015
        0x03000016 //row 3 col0
        0x03010017
        0x03020018
        0x03030019
        0x0304001a
        0x0305001b
        0x0306001c
        0x0400001d //row 4 col0
        0x0401001e
        0x0402001f
        0x04030020
        0x04040021
        0x04050022
        0x04060023
        0x05000024 //row 5 col0
        0x05010025
        0x05020026
        0x05030027
        0x05040028
        0x05050029
        0x0506002a
>;
gpio-activelow;
debounce-delay-ms = <20>;
```

```
                col-scan-delay-us = <20>;
                linux,no-autorepeat;
                status = "okay";
        };
```

设备树文件参数描述如下：

- keypad,num-rows：行数
- keypad,num-columns：列数
- row-gpios：行对应的 gpio 口，从第一行开始
- col-gpios：列对应的 gpio 口，从第一列开始
- linux,keymap：每一个按键对应的键值
- gpio-activelow：按键按下时，行线是否为低电平。用于触发中断，必须配置。
- debounce-delay-ms：消抖时间。
- col-scan-delay-us：扫描延时，如果 IO 状态转换时间过长可能会导致按键扫描错误。
- linux,no-autorepeat：按键按下时是否重复提交按键, 设 1 就是不重复, 设 0 重复。

2. 确认驱动是否被选中在 tina 目录下执行运行 make kernel_menuconfig，确认以下配置：

```
Device Drivers
    └->Input device support
        └->Keyboards
            └->GPIO driven matrix keypad support
```

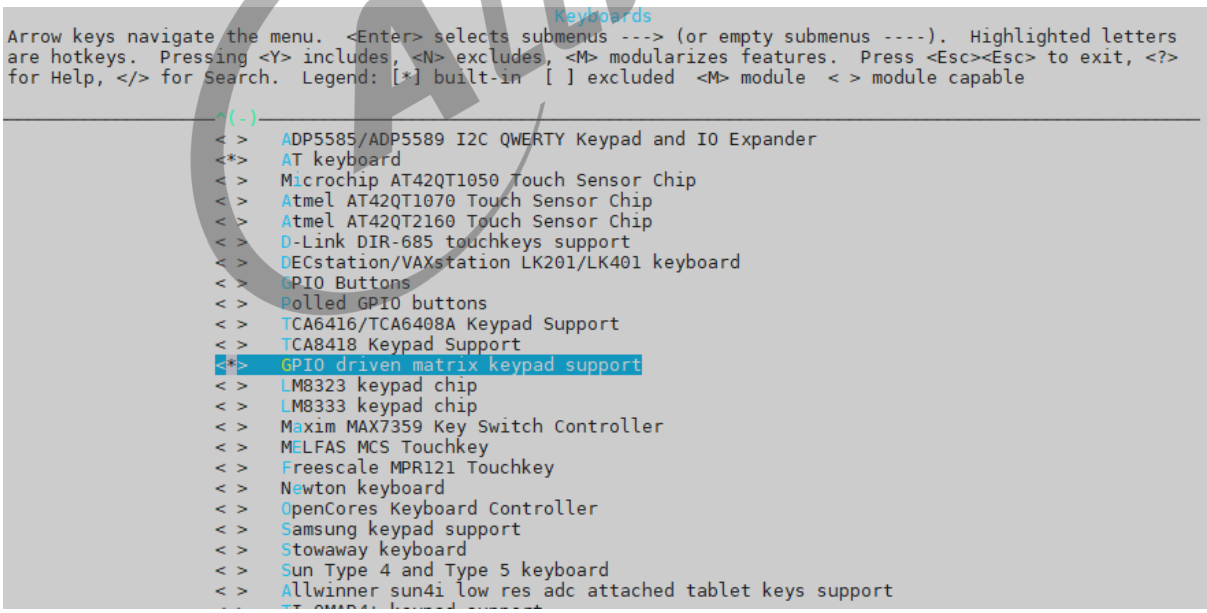

图 3-5: 矩阵键盘配置图

4 ADC-Key

ADC-Key 一共有两种 adc 检测方式，分别是 LRADC 和 GPADC。LRADC 分辨率为 6 位，GPADC 分辨率为 12 位，因此后者精度会更高。

下面将分别讲述两种不同的方式。

4.1 4.4 以及 4.9 内核

4.1.1 LRADC-Key

LRADC-Key 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,R328 sdk 中默认配置为 250Hz。
3. 当前 lradc key 最大可以支持到 13 个按键 (0.15V 为一个档)，通常情况下，建议 lradc key 最大不要超过 8 个 (0.2V 为一档)，否则由于采样误差、精度等因素存在，会很容易出现误判的情况。

如下图所示,lradc-key 检测原理是当有按键被按下或者抬起时，LRADC 控制器 (6bit) 检测到电压变化后，经过 LRADC 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 LRADC 内部的 data register 的值 (0~0x3f) 来体现。

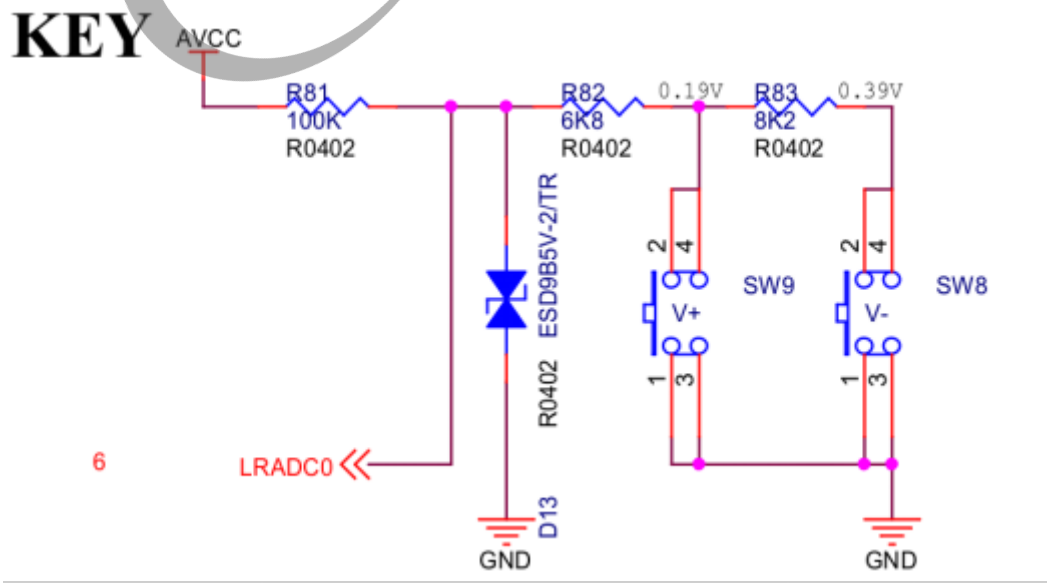

图 4-1: LRADC 按键原理图

下面以 R328 为例进行说明，驱动文件：

```
linux-4.9/drivers/input/keyboard/sunxi-keyboard.c
linux-4.9/arch/arm/boot/dts/sun8iw18p1.dtsi
```

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
                    └>softwinnner KEY BOARD support
```

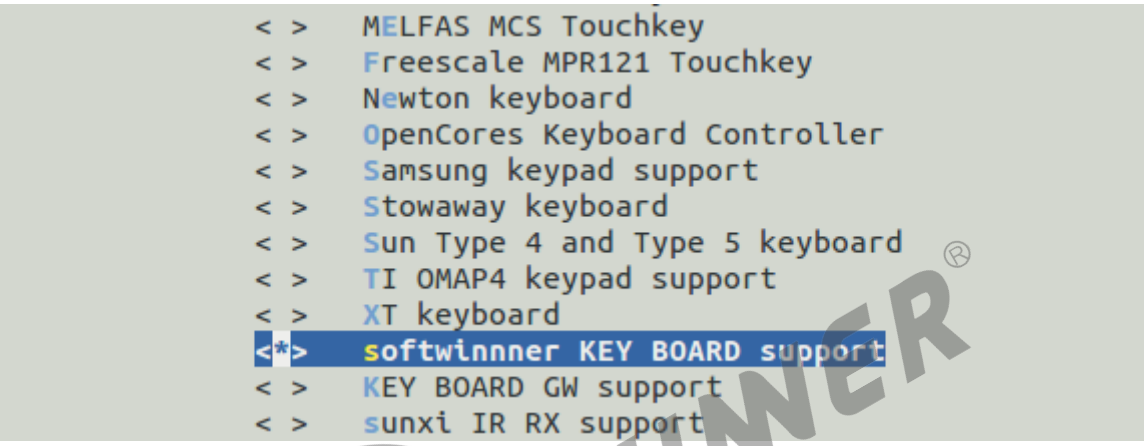

图 4-2: linux4.4/4.9 LRADC 按键配置图

2. 在 lradc-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,                  /* key 1,  0-7   */
    1,1,1,1,1,1,                    /* key 2,  8-14  */
    2,2,2,2,2,2,                    /* key 3,  15-21 */
    3,3,3,3,3,                      /* key 4,  22-27 */
    4,4,4,4,4,                      /* key 5,  28-33 */
    5,5,5,5,5,                      /* key 6,  34-39 */
    6,6,6,6,6,6,6,6,6,              /* key 7,  40-49 */
    7,7,7,7,7,7,7,7,7,7,7,7,7       /* 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,
};
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

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,                  /* key 2,  10-19  */
    2,2,2,2,2,2,2,2,2,2,2,2,            /* key 3,  20-32 */
    3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,      /* key 4,  33-48 */
    4,4,4,4,4,4,4,4,4,                  /* key 5,  49-58 */
    5,5,5,5,5,                       /* key 6,  59-63 */
};
```

3. 设备树文件 sun8iw18p1.dtsi

```
keyboard0:keyboard{
     compatible = "allwinner,keyboard_1350mv";
     reg = <0x0 0x05070800 0x0 0x400>;
     interrupts = <GIC_SPI 50 IRQ_TYPE_NONE>;
     status = "okay";
     key_cnt = <5>;
     key0 = <164 115>;
     key1 = <415 114>;
     key2 = <646 139>;
     key3 = <900 28>;
     key4 = <1157 102>;
     wakeup-source;
};
```

- compatible：匹配驱动。
- reg：LRADC 模块的基地址。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

- interrupts：LRADC 中断源。
- status：是否加载驱动。
- key_cnt：按键数量。
- key0：第一个按键，前面数字的是电压范围，后者是 input 系统的键值。
- wakeup-source：LRADC 作为唤醒源。

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。

4.1.2 GPADC-Key

GPADC 有多个通道，例如在 R328S3 中有 4 个通道。

通道有三种工作模式：
一、是在指定的通道完成一次转换，转换数据更新在对应通道的数据寄存器中；
二、在所有指定的通道连续转换直到软件停止，转换数据更新在对应通道的数据寄存器中；
三、可以在指定的通道进行 adc 的采样和转换，并将结果顺序地存入 FIFO，FIFO 深度为 64 并支持中断控制。

检测原理是当按键被按下或抬起时，指定的通道的控制器检测到电压变化，然后经过逻辑控制单元进行比较运算后，产生相应的中断给 CPU。R328S3 的 GPADC 的框图如下图所示。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

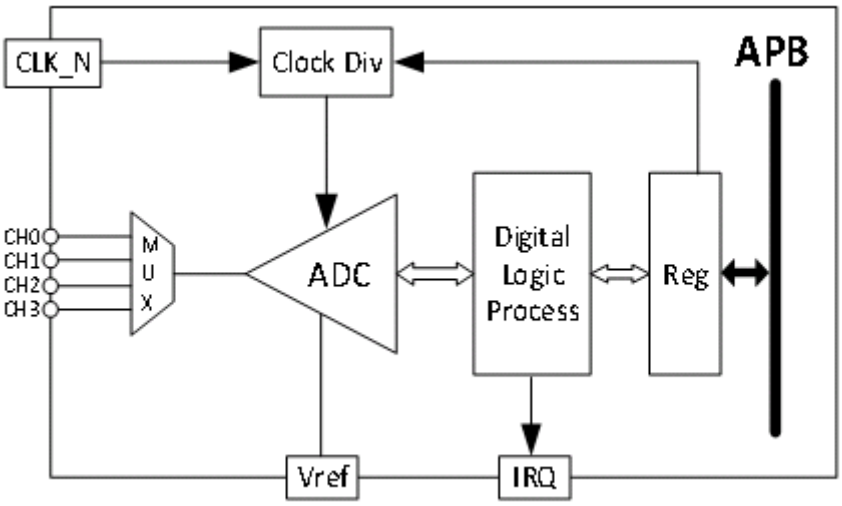

图 4-3: GPADC 按键硬件图

以 R328S3 为例，驱动文件：

```
linux-4.9/drivers/input/sensor/sunxi_gpadc.c
linux-4.9/drivers/input/sensor/sunxi_gpadc.h
```

1. 如果使用 gpadc-key，请先确保 SENSORS_GPADC 有被选上。

```
Device Drivers
    └->Device Drivers
        └->Input device support
            └->Sensors
                └->SUNXI GPADC
```

```
                                    Sensors
Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty 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  < > module capable

        --- Sensors
        <M>   BMA250 acceleration sensor support (NEW)
        <M>   SC7A30 3-Axis Orientation/Motion Detection Sensor Support (NEW)
        <M>   MMA7660 3-Axis Orientation/Motion Detection Sensor Support (NEW)
        <M>   MIR3DA 2-Axis Orientation/Motion Detection Sensor Support (NEW)
        <M>   MXC622X 2-Axis Orientation/Motion Detection Sensor Support (NEW)
        <M>   MMA8452 3-Axis Orientation/Motion Detection Sensor Support (NEW)
        <M>   MMA865X 3-Axis Orientation/Motion Detection Sensor Support (NEW)
        <M>   MC32X0 Orientation/Motion Detection Sensor Support (NEW)
        <*>   SUNXI GPADC
        < >   SUNXI GPADC test (NEW)
        [ ]   allwinnner KEY GPIO support (NEW)
```

图 4-4: linux4.4/4.9 GPADC 按键配置图

2. 在 gpadc 中与 lradc 一样涉及两个重要的结构体

```
static unsigned char keypad_mapindex[128] = {
        0, 0, 0, 0, 0, 0, 0, 0, 0,              /* key 1, 0-8 */
        1, 1, 1, 1, 1,                      /* key 2, 9-13 */
        2, 2, 2, 2, 2, 2,                    /* key 3, 14-19 */
        3, 3, 3, 3, 3, 3,                    /* key 4, 20-25 */
        4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,          /* key 5, 26-36 */
        5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,          /* key 6, 37-39 */
        6, 6, 6, 6, 6, 6, 6, 6, 6,              /* key 7, 40-49 */
        7, 7, 7, 7, 7, 7, 7                   /* key 8, 50-63 */
};
u32 scankeycodes[KEY_MAX_CNT];
```

keypad_mapindex[] 数组的定义与 LRADC 的相似，但是要注意的是，GPADC 的数组大小为 128，而 LRADC 的是 64，所以说 GPADC 的精度更高。在 linux4.9 内核中已经不在数组里配置了，程序中该数组只是初始化的。后面会根据 sys_config 或者 dts 的配置来生成所要使用的 keypad_mapindex[]。

3. 设备树文件这里示例的是 R328S3 的方案级设备树文件：

```
lichee/linux-4.9/arch/arm/boot/dts/sun8iw18p1.dtsi
```

详细 GPADC 配置如下：

```
gpadc:gpadc{
        compatible = "allwinner,sunxi-gpadc";
        reg = <0x0 0x05070000 0x0 0x400>;
        interrupts = <GIC_SPI 48 IRQ_TYPE_NONE>;
        clocks = <&clk_gpadc>;
        status = "disable";
};
```

- compatible：匹配驱动。
- reg：GPADC 寄存器基地址。
- interrupts：GPADC 中断源。
- clocks：GPADC 依赖的时钟。
- status：是否加载设备。

4.sys_config.fex 文件

```
[gpadc]
gpadc_used                      = 1
channel_num                     = 1
channel_select                  = 0x01
channel_data_select             = 0
channel_compare_select          = 0x01
channel_cld_select              = 0x01
channel_chd_select              = 0
channel0_compare_lowdata        = 1700000
channel0_compare_higdata        = 1200000
key_cnt                         = 4
key0_vol                        = 164
key0_val                        = 115
```

```
key1_vol                        = 415
key1_val                        = 114
key2_vol                        = 700
key2_val                        = 139
key3_vol                        = 1000
key3_val                        = 28
```

在 sys_config 中，配置含义如下：

- channel_num：表示平台上支持的最大通道数；
- channel_select：表示通道启用设置，通道 0:0x01 通道 1:0x02 通道 2:0x04 通道 3:0x08；
- channel_data_select：表示通道数据启用，通道 0:0x01 通道 1:0x02 通道 2:0x04 通道 3:0x08；
- channel_compare_select；表示比较功能启用，通道写法跟前面一致；
- channel0_compare_lowdata：表示的低于该电压可以产生中断，这里是 1.7V；
- channel0_compare_higdata：则是高于该电压可以产生中断，这里是 1.2V；
- key_cnt：表示的是按键个数；
- key*_vol：表示第 * 个按键的电压值；
- key*_val：表示的是第 * 个按键的键值。

在软件上，当 key_vol 在 0~164 范围时，表示的是操作 key0，以此类推；在硬件上，164 表示的是 164mV，通过计算`keyn_vol = keyn_vol + ( key(n+1)_vol - keyn_vol) / 2`来得到范围。比如说第一个范围为 key0_vol = 164 + (415 - 164) / 2 = 289.5，即电压在 0~289.5 在，adc 检测到为操作的 key0，以此类推。然后通过 scankeycodes 得到键值 115。

5.board.dts 文件

方案级设备树文件只写这个模块的配置，而详细的按键配置一般需要写在板级的 board.dts 中。

```
/*
resistance gpadc configuration
channel_num:   Maxinum number of channels supported on the platform.
channel_select:   channel enable setection. channel0:0x01  channel1:0x02 channel2:0x04
    channel3:0x08
channel_data_select:  channel data enable.  channel0:0x01  channel1:0x02 channel2:0x04
    channel3:0x08.
channel_compare_select:   compare function enable channel0:0x01  channel1:0x02 channel2:0
    x04 channel3:0x08.
channel_cld_select:  compare function low data enable setection: channel0:0x01  channel1:0
    x02 channel2:0x04 channel3:0x08.
channel_chd_select:  compare function hig data enable setection: channel0:0x01  channel1:0
    x02 channel2:0x04 channel3:0x08.
*/
gpadc:gpadc{
            channel_num = <1>;
            channel_select = <0x01>;
            channel_data_select = <0>;
            channel_compare_select = <0x01>;
            channel_cld_select = <0x01>;
            channel_chd_select = <0>;
            channel0_compare_lowdata = <1700000>;
            channel0_compare_higdata = <1200000>;
```

```
            channel1_compare_lowdata = <460000>;
            channel1_compare_higdata = <1200000>;
            key_cnt = <5>;
            key0_vol = <210>;
            key0_val = <115>;
            key1_vol = <410>;
            key1_val = <114>;
            key2_vol = <590>;
            key2_val = <139>;
            key3_vol = <750>;
            key3_val = <28>;
            key4_vol = <880>;
            key4_val = <102>;
            status = "okay";
};
```

dts 文件参数的含义可以参考 sys_config.fex 的。

4.2 5.4 内核

4.2.1 LRADC-Key

5.4 内核的 LRADC 驱动功能与 4.4/4.9 内核的无明显变化，因此下面只介绍设备树文件与配置教程。

1. 修改设备树文件

这里以 D1 为例，设备树文件路径为：

```
lichee/linux-5.4/arch/riscv/boot/dts/sunxi/sun20iw1p1.dtsi
```

dtsi 一般默认已经写好 LRADC 的配置：

```
keyboard0: keyboard@2009800 {
        compatible = "allwinner,keyboard_1350mv";
        reg = <0x0 0x02009800 0x0 0x400>;
        interrupts-extended = <&plic0 77 IRQ_TYPE_EDGE_RISING>;
        clocks = <&ccu CLK_BUS_LRADC>;
        resets = <&ccu RST_BUS_LRADC>;
        key_cnt = <5>;
        key0 = <210 115>;
        key1 = <410 114>;
        key2 = <590 139>;
        key3 = <750 28>;
        key4 = <880 172>;
        wakeup-source;
        status = "okay";
};
```

- compatible：匹配驱动。

- reg：LRADC 模块的基地址。
- interrupts：LRADC 中断源。
- status：是否加载驱动。
- key_cnt：按键数量。
- key0：第一个按键，前面数字的是电压范围，后者是 input 系统的键值。
- wakeup-source：LRADC 作为唤醒源。

但是 D1 方案电路板上实际 LRADC 只连接了一个按键，此时可以通过修改 board.dts 来进行配置。若实际的电路板上 LRADC 按键是符合以上配置的，则不需要修改。

说明

***board.dts* 的配置最终会覆盖了方案的 *dtsi* 文件配置**

board.dts 配置：

```
&keyboard0 {
        key_cnt = <1>;
        key0 = <210 115>;
        status = "okay";
};
```

2. 确认驱动是否被选中在 tina 目录下执行 make kernel_menuconfig，确认以下配置：

```
Device Drivers
     └->Input device support
             └->Keyboards
                   └->softwinnner KEY BOARD support
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
                                    Keyboards
us ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
gend: [*] built-in  [ ] excluded  <M> module  < > module capable

        --- Keyboards
        < >   ADP5588/87 I2C QWERTY Keypad and IO Expander
        < >   ADP5585/ADP5589 I2C QWERTY Keypad and IO Expander
        <*>   AT keyboard
        < >   Microchip AT42QT1050 Touch Sensor Chip
        < >   Atmel AT42QT1070 Touch Sensor Chip
        < >   Atmel AT42QT2160 Touch Sensor Chip
        < >   D-Link DIR-685 touchkeys support
        < >   DECstation/VAXstation LK201/LK401 keyboard
        < >   GPIO Buttons
        < >   Polled GPIO buttons
        < >   TCA6416/TCA6408A Keypad Support
        < >   TCA8418 Keypad Support
        < >   GPIO driven matrix keypad support
        < >   LM8323 keypad chip
        < >   LM8333 keypad chip
        < >   Maxim MAX7359 Key Switch Controller
        < >   MELFAS MCS Touchkey
        < >   Freescale MPR121 Touchkey
        < >   Newton keyboard
        < >   OpenCores Keyboard Controller
        < >   Samsung keypad support
        < >   Stowaway keyboard
        < >   Sun Type 4 and Type 5 keyboard
        < >   Allwinner sun4i low res adc attached tablet keys support
        < >   TI OMAP4+ keypad support
        < >   TM2 touchkey support
        < >   XT keyboard
        < >   Microchip CAP11XX based touch sensors
        < >   Broadcom keypad driver
        <*>   softwinnner KEY BOARD support
```

图 4-5: Linux-5.4 LRADC 按键配置图

4.2.2 GPADC-Key

5.4 内核的 GPADC 设备树配置主要是时钟和中断方面有所改动，board.dts 配置可参考 4.9 内核。

1. 修改设备树文件

这里以 D1 来作为示例，设备树文件为：

```
lichee/linux-5.4/arch/riscv/boot/dts/sunxi/sun20iw1p1.dtsi
```

详细配置为：

```
gpadc: gpadc@2009000 {
        compatible = "allwinner,sunxi-gpadc";
        reg = <0x0 0x02009000 0x0 0x400>;
        interrupts-extended = <&plic0 73 IRQ_TYPE_LEVEL_HIGH>;
        clocks = <&ccu CLK_BUS_GPADC>;
        clock-names = "bus";
        resets = <&ccu RST_BUS_GPADC>;
        status = "okay";
};
```

- compatible：匹配驱动。
- reg：GPADC 寄存器基地址。
- interrupts：GPADC 中断源。
- clocks：GPADC 依赖的时钟。
- status：是否加载设备。

2. 确认驱动是否被选中

在 tina 目录下执行 make kernel_menuconfig，确认以下配置：

```
Device Drivers
     └->Input device support
        └->Sensors
            └->sunxi gpadc driver support
```

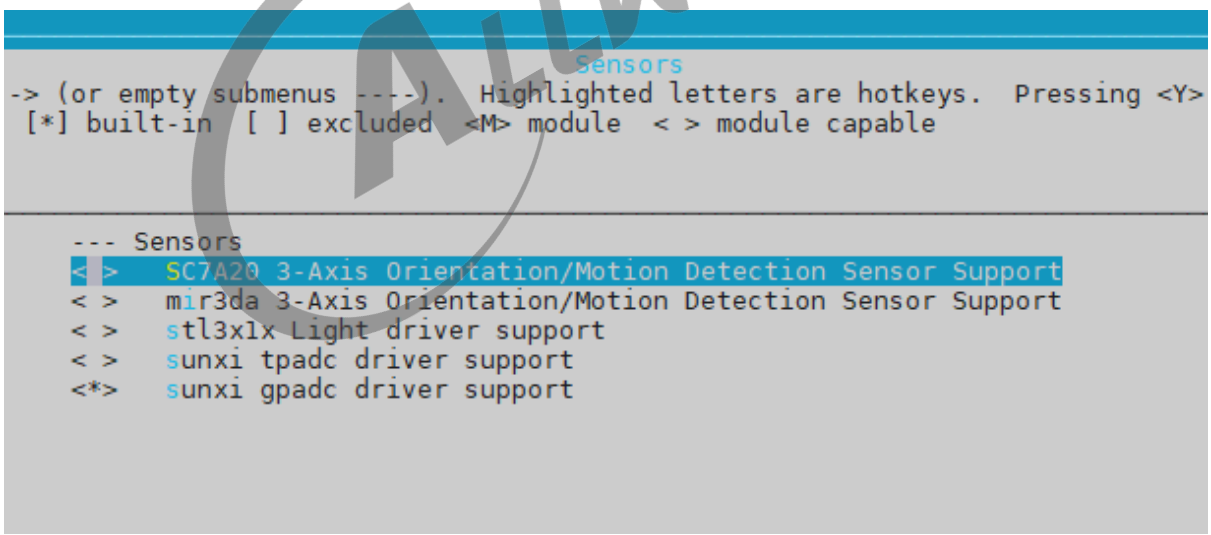

图 4-6: Linux-5.4 GPADC 配置图

5 AXP-Key

使用 AXP-Key 的话需要电路板上带上我司提供 power-key 功能的 PMU，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。

这里使用 R329 来作为例子，其对应的设备节点为：

```
root@TinaLinux:/# ls
bin       lib       overlay   rom       sys       var
dev       lib64     proc      root      tmp       www
etc       mnt       rdinit    sbin      usr
root@TinaLinux:/# ll /dev/input/event1
crw-------    1 root     root       13,  65 Jan  1 00:00 /dev/input/event1
root@TinaLinux:/# cat /sys/class/input/input1/name
axp2101-pek
root@TinaLinux:/#
```

图 5-1: AXP 按键节点图

上报的键值为 KEY_POWER 116

5.1 4.9 内核

1. 修改设备树文件这里以 R329-evb1 为例，设备树文件为 board.dts，路径：

```
device/config/chips/r329/configs/evb1/board.dts
```

详细配置为：

```
&twi2{
        no_suspend = <1>;
        status = "okay";

        pmu0: pmu@34 {
                compatible = "x-powers,axp2585";

............省略其他无关配置

                powerkey0: powerkey@0 {
                        status = "okay";
                        compatible = "x-powers,axp2585-pek";
                        pmu_powkey_off_time = <6000>;
                        pmu_powkey_off_func = <0>;
                        pmu_powkey_off_en = <1>;
```

```
                        pmu_powkey_long_time = <1500>;
                        pmu_powkey_on_time = <1000>;
                        wakeup_rising;
                        /* wakeup_falling; */
                };
        };

};
```

2. 确认驱动是否被选中

tina 目录下执行 make kernel_menuconfig，确认以下配置：

```
Device Drivers
    └->Input device support
        └->Miscellaneous devices
            └->X-Powers AXP2101 power button driver
```

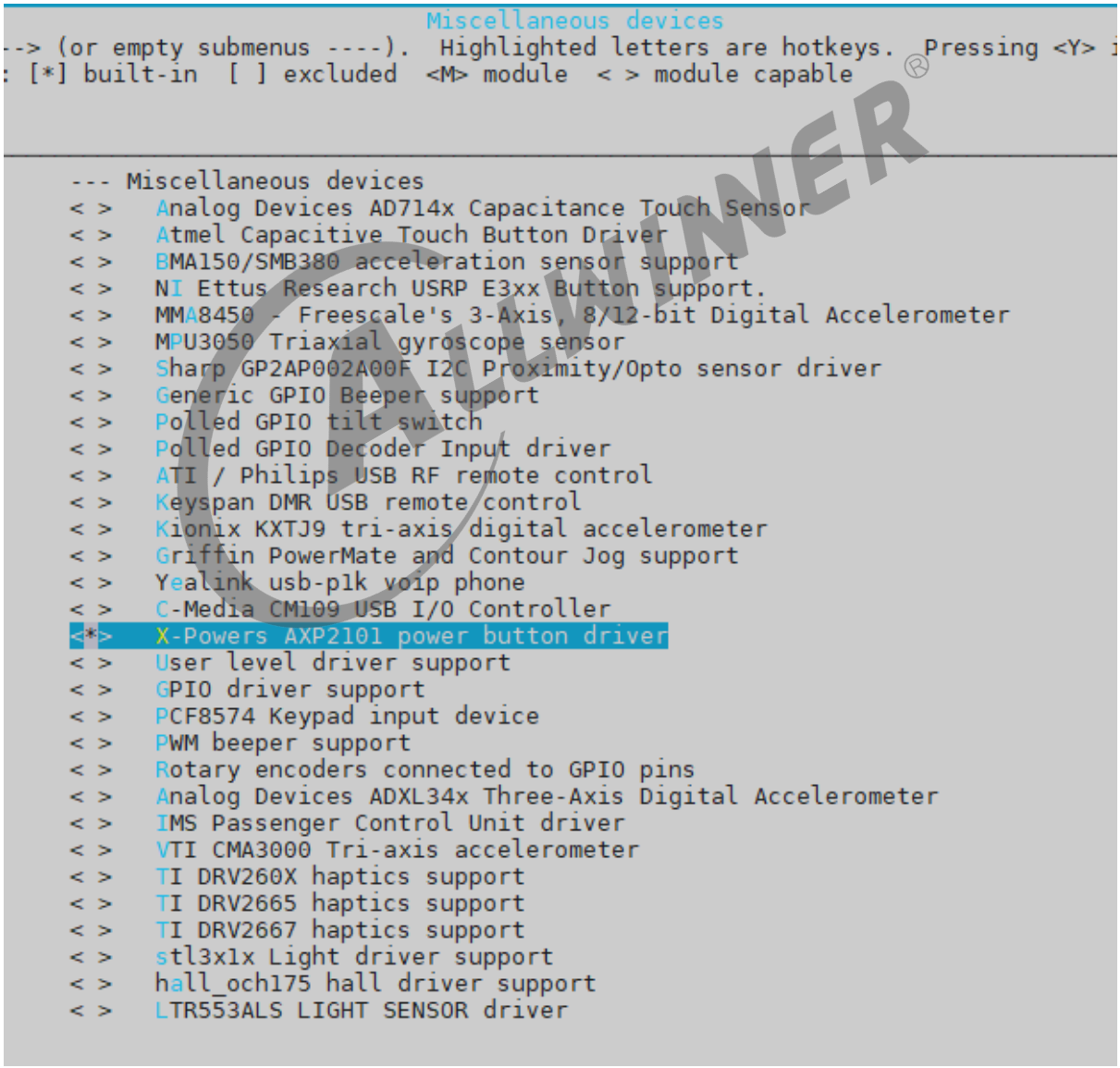

图 5-2: AXPKEY 配置图

警告

不同方案使用的 PMU 可能会不一样，因此 AXP-KEY 的配置也会不一致，详细查看各个方案使用的 PMU 型号。

这里另外，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>
```

5.2 5.4 内核

说明

***Linux-5.4* 中暂时没有平台使用我司的 *PMU*，本章节暂不介绍 *Linux-5.4* 内核。**

版权所有 © 珠海全志科技股份有限公司。保留一切权利

著作权声明

版权所有 ©2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、、全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。