D1-H Tina Linux Key 快速配置使用指南

版本号: 1.0
发布日期: 2021.04.12

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.12	AWA1611	新建初始版本

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 普通 GPIO 采用 poll 方式 3
3.2 普通 GPIO 采用中断方式 4
3.3 矩阵键盘 5

4 LRADC-Key **9**

ALLWINNER®

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
D1-H	Linux-5.4	risc-v(64 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 平台支持两种不同类型的 Key：GPIO-Key，ADC-Key。

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

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

D1-H 方案的 dts 文件位置：

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

板级的 dts 文件位置为：

```
tina/device/config/chips/d1-h/configs/nezha/linux/board.dts
```

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
```

3.1 普通 GPIO 采用 poll 方式

1. 修改设备树文件

说明

如果是新增的 *gpio-keys* 设备配置，推荐在 *board.dts* 中进行修改。

```
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。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

- 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 轮询驱动 kernel_menuconfig 路径：

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

3.2 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。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2. 选上驱动

GPIO-Key 中断驱动 kernel_menuconfig 路径：

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

3.3 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：

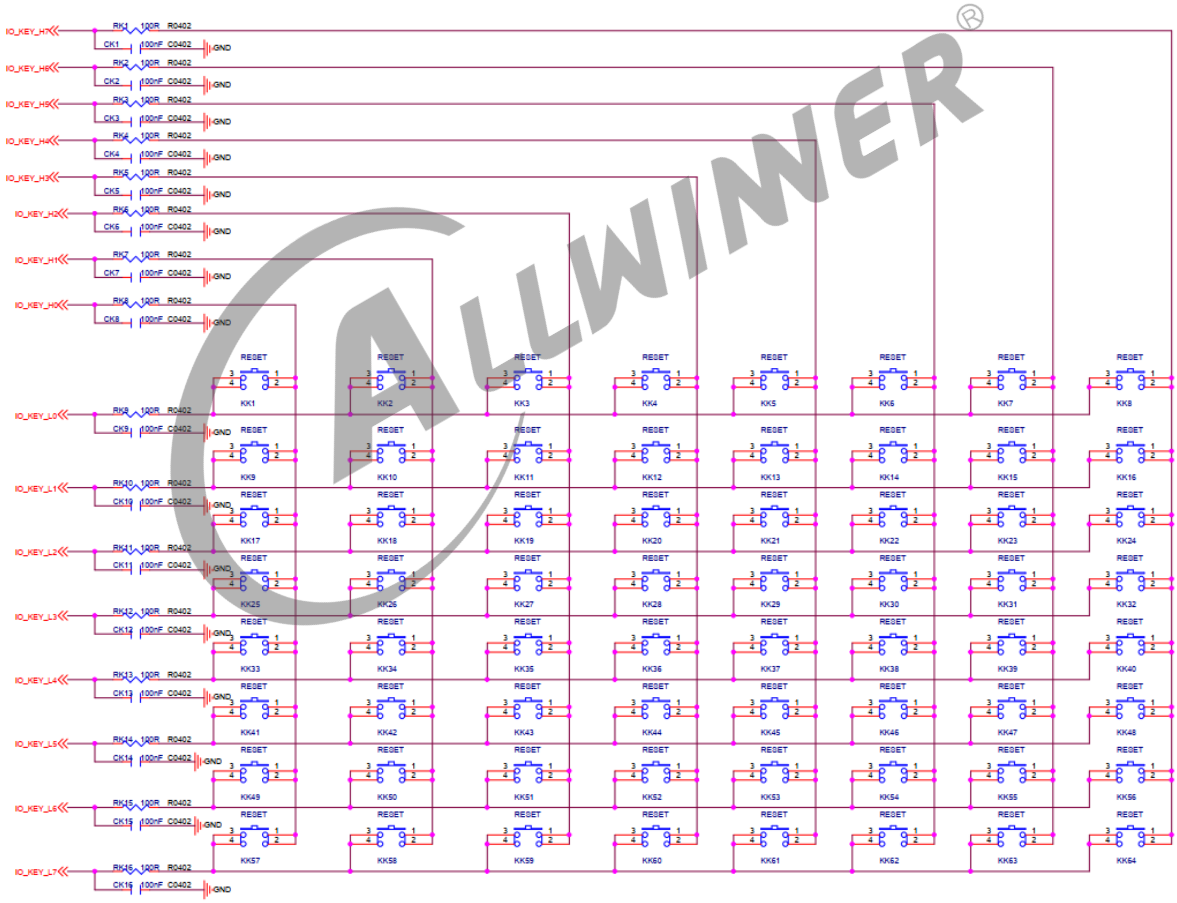

图 3-1：矩阵按键硬件原理图

驱动文件为：

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

版权所有 © 珠海全志科技股份有限公司。保留一切权利

⚠ 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。

1. 修改设备树文件

根据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
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
                        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
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

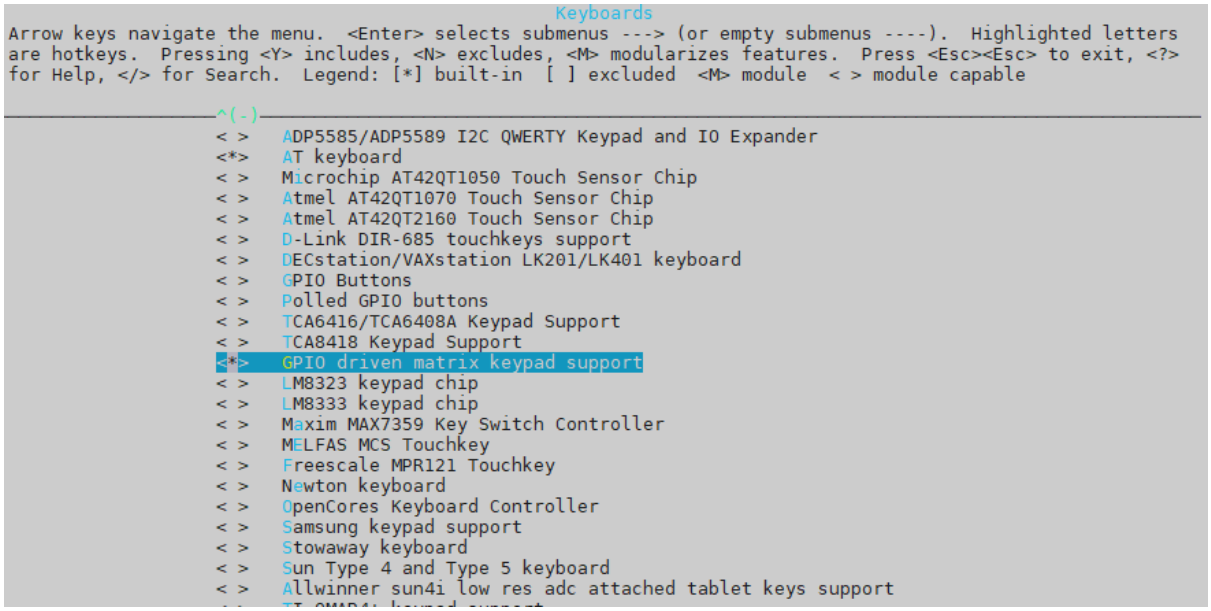

图 3-2: 矩阵键盘配置图

版权所有 © 珠海全志科技股份有限公司。保留一切权利

4 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,D1-H sdk 中默认配置为 500Hz。
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。

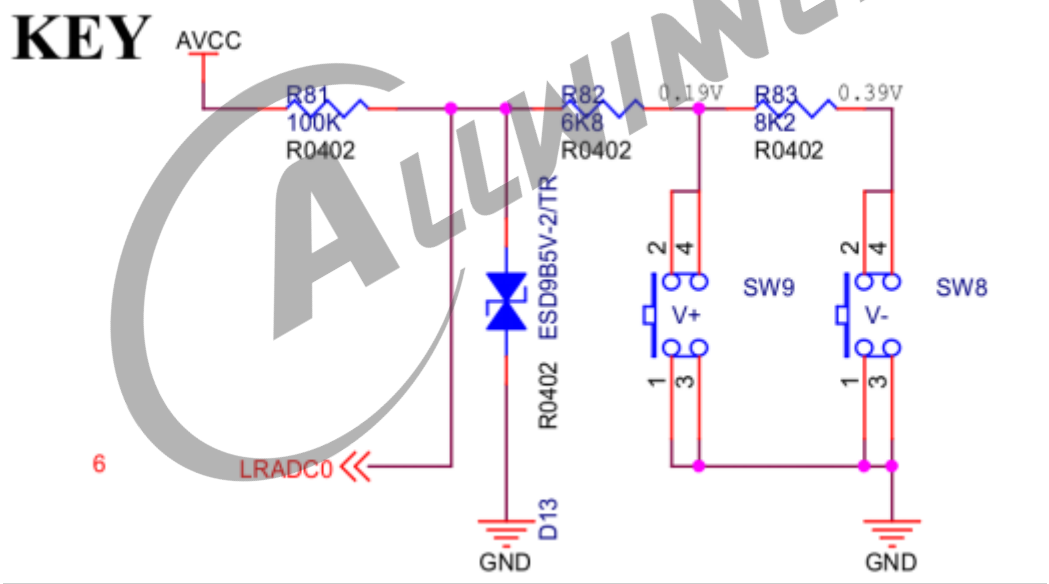

图 4-1: LRADC 按键原理图

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 */
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
    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,
};
```

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，该数组内无需任何更改，当个别情况下，如遇到 lradc-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 board.dts 进行配置：

```
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 */
    3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,    /* key 4,  33-48 */
    4,4,4,4,4,4,4,4,4,4,                /* key 5,  49-58 */
    5,5,5,5,5,                      /* key 6,  59-63 */
};
```

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。

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 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。

下面介绍设备树文件与配置教程。

1. 修改设备树文件

这里以 D1-H 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-H 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。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

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
```

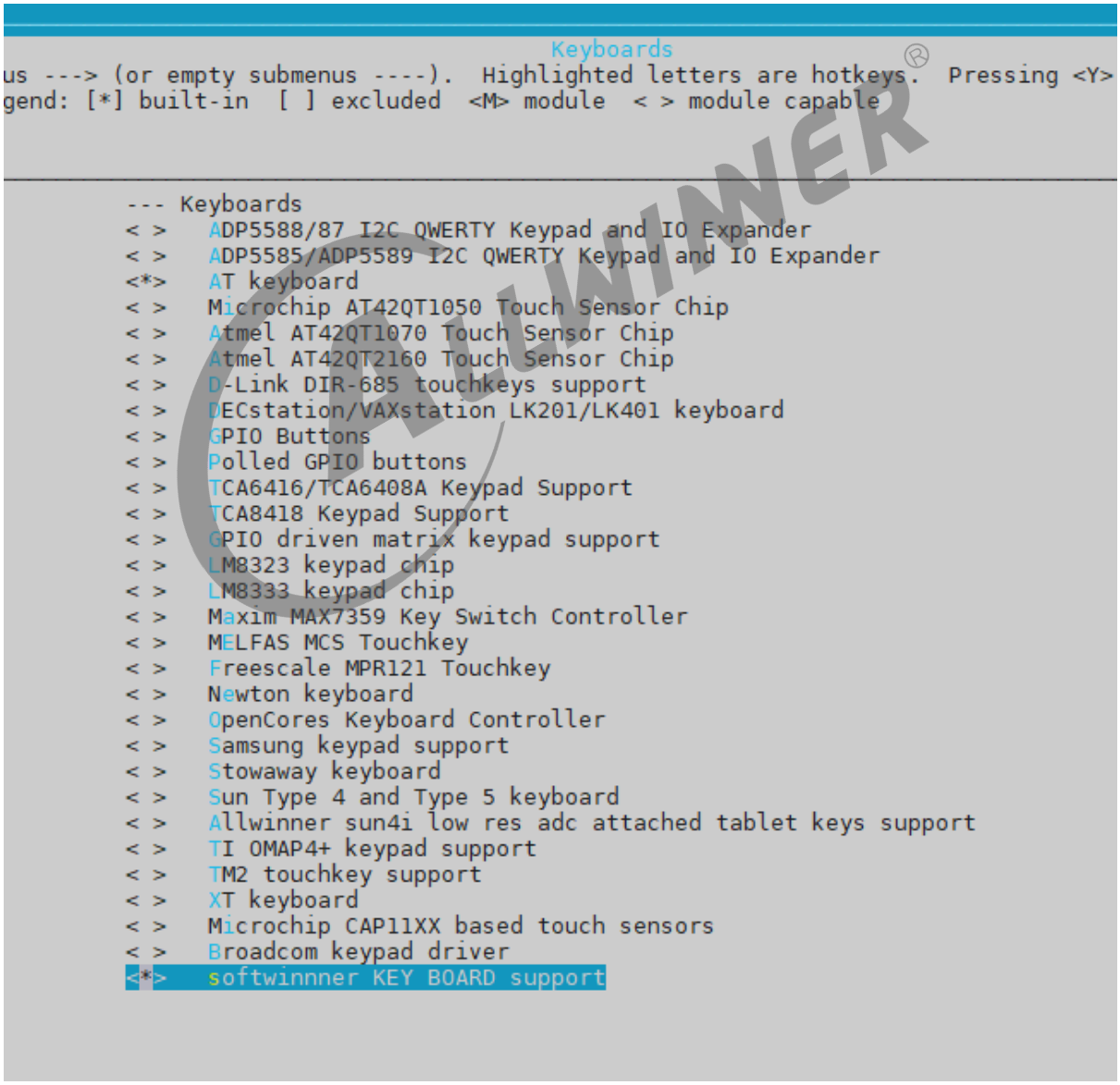

图 4-2: Linux-5.4 LRADC 按键配置图

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。