Linux LRADC
开发指南

版本号: 1.1
发布日期: 2021.05.10

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.03.03	AWA1637	添加初版
1.1	2021.05.10	AWA1637	增加 R528 的说明

目　录

插　图

1 前言

1.1 文档简介

介绍 LRADC 模块的使用方法，方便开发人员使用。

1.2 目标读者

LRADC 模块的驱动开发/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	驱动文件
T509	Linux-4.9	sunxi-keyboard.c
MR813	Linux-4.9	sunxi-keyboard.c
R818	Linux-4.9	sunxi-keyboard.c
A100	Linux-4.9	sunxi-keyboard.c
A100	Linux-5.4	sunxi-keyboard.c
A133	Linux-4.9	sunxi-keyboard.c
A133	Linux-5.4	sunxi-keyboard.c
R528	Linux-5.4	sunxi-keyboard.c
H616	Linux-4.9	sunxi-keyboard.c
H616	Linux-5.4	sunxi-keyboard.c

2 模块介绍

2.1 模块功能

LRADC 模块属于 INPUT 输入设备，一般包括 VOL+、VOL-、HOME、MENU、ENTER 等等。Sunxi LRADC 模块的实际电路如下图所示：

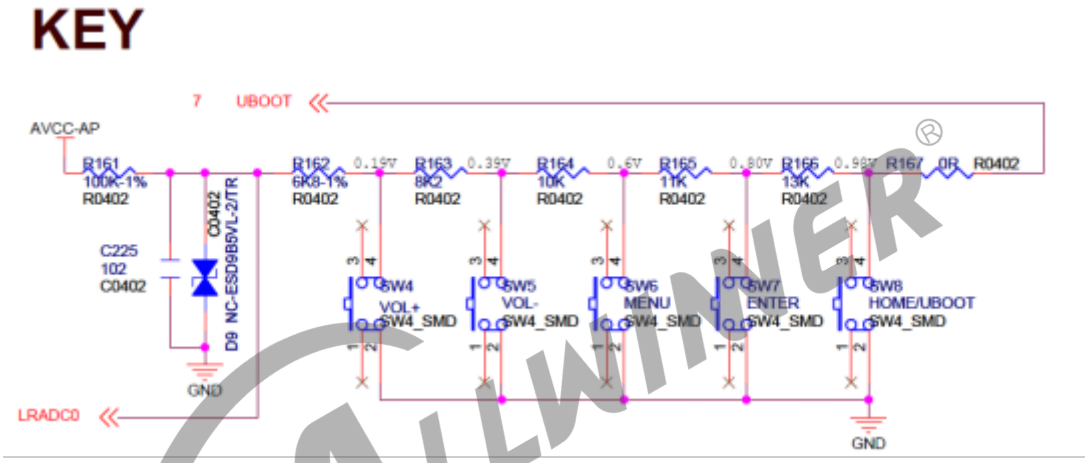

图 2-1: KEY 按键电路

AVCC-AP 为 1.8V 的供电，不同的按键按下，LRADC0 口的电压不同，CPU 通过对这个电压的采样来确定具体是哪一个按键按下。如上图，VOL+、VOL-、MENU、ENTER、HOME/U-BOOT 对应的电压分别为 0.19V、0.39V、0.6V、0.80V、0.98V。

2.2 结构框图

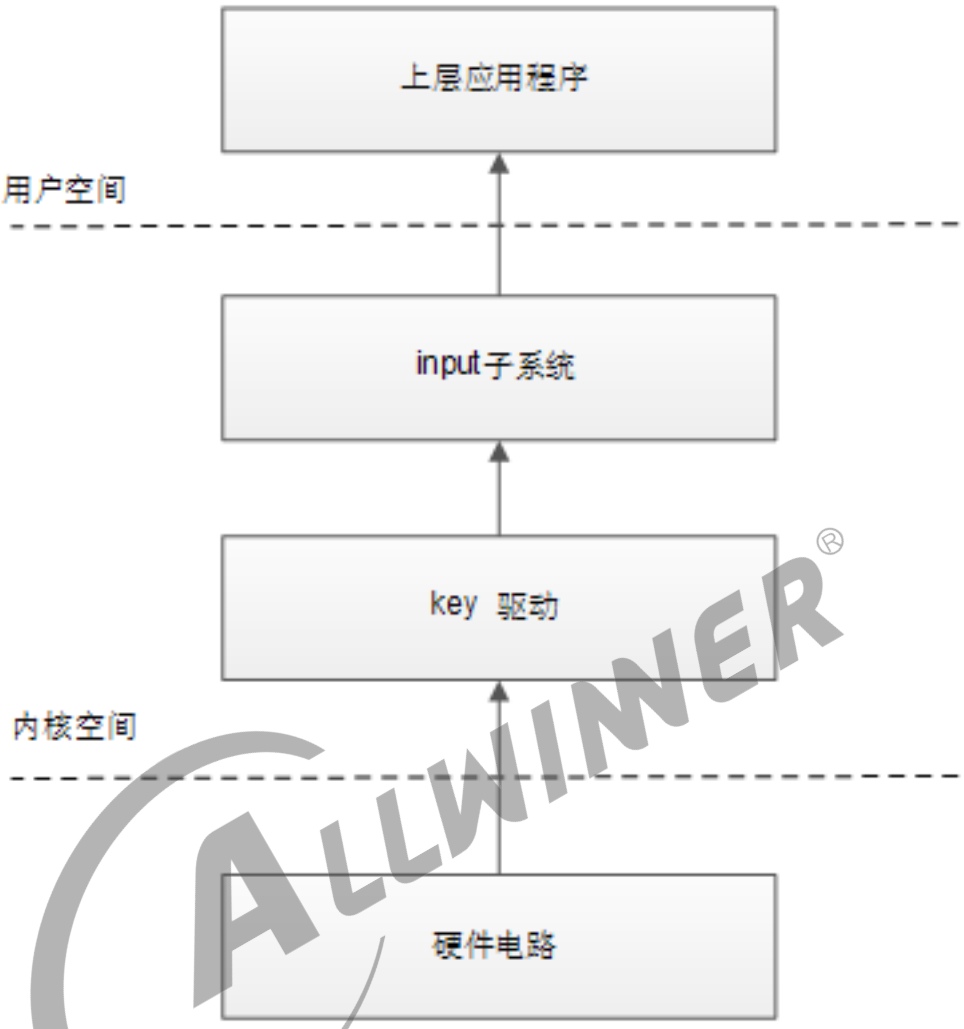

图 2-2: KEY 模块结构框图

整个系统框架流程如下所示：当用户按下按键的时候，会触发一个中断。这里的中断通过读取中断状态寄存器，判断是按键按下中断，还是数据中断，还是按键释放中断。KEY 按键驱动会进入中断，然后读取整个按键电路的电压值，然后对该电压值进行解码，然后将该事件上报给 INPUT 子系统。INPUT 子系统找到相应的事件处理程序之后，会将该按键事件上报给用户空间，等待用户程序对该按键信息的读取与处理。

表 2-1: 术语介绍

术语	解释说明
sunxi LRADC	指 Allwinner 的一系列 soc 硬件平台全志平台使用的按键模块

2.3 模块配置

2.3.1 设备树配置

LRADC 模块的设备树配置位于 longan 的内核目录，如 arch/arm/boot/dts/xxx.dtsi，64 位为：arch/arm64/boot/dts/sunxi/xxx.dtsi，下面为一个配置：

```
    sunxi_keyboard:keyboard{
        compatible = "allwinner,keyboard";
        reg = <0x0 0x01c21800 0x0 0x400>;        /* 寄存器地址 */
        interrupts = <GIC_SPI 30 IRQ_TYPE_NONE>;    /* 中断 */
        status = "okay";                 /* 使能该节点 */
        key_cnt = <5>;
        key0 = <115 115>;   /*根据实际情况，左边115是电压，单位为mV，右边115为该电压对应的键值*/
        key1 = <235 114>;
        key2 = <330 139>;
        key3 = <420 28>;
        key4 = <520 102>;
    };
```

2.3.2 menuconfig 配置

longan 的 linux-4.9 环境中，在命令行进入内核根目录，执行 make ARCH=arm menuconfig（64 位系统为 make ARCH=arm64 menuconfig）进入配置主界面，并按以下步骤操作：(注，在 longan 的 linux-5.4 中，在需要在根目录执行./build.sh menuconfig 即可; 在 tina 环境中，可以在根目录执行 make kernel_menuconfig 进入)

- 首先，选择 Device Drivers 选项进入下一级配置，如下图所示：

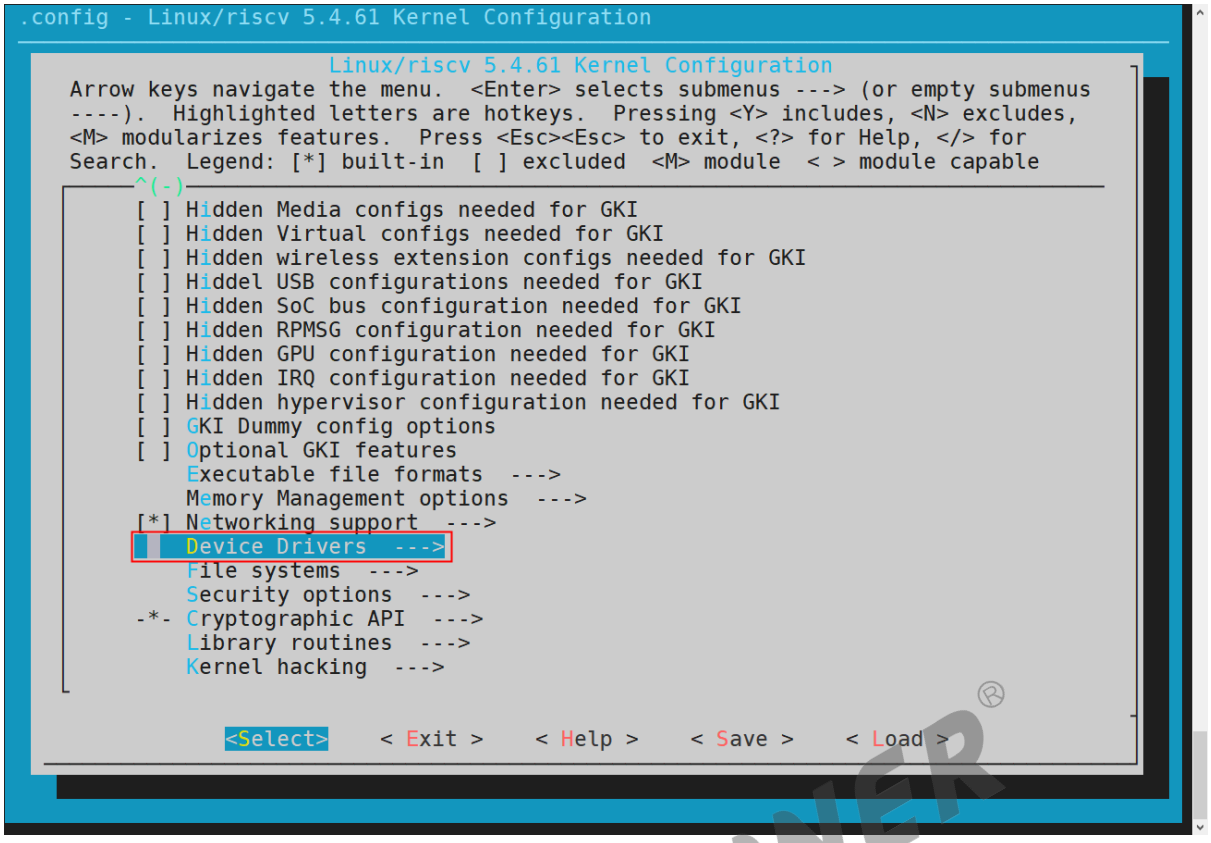

图 2-3: Device Drivers

- 选择 Input device support 选项进入下一级配置，如下图所示：

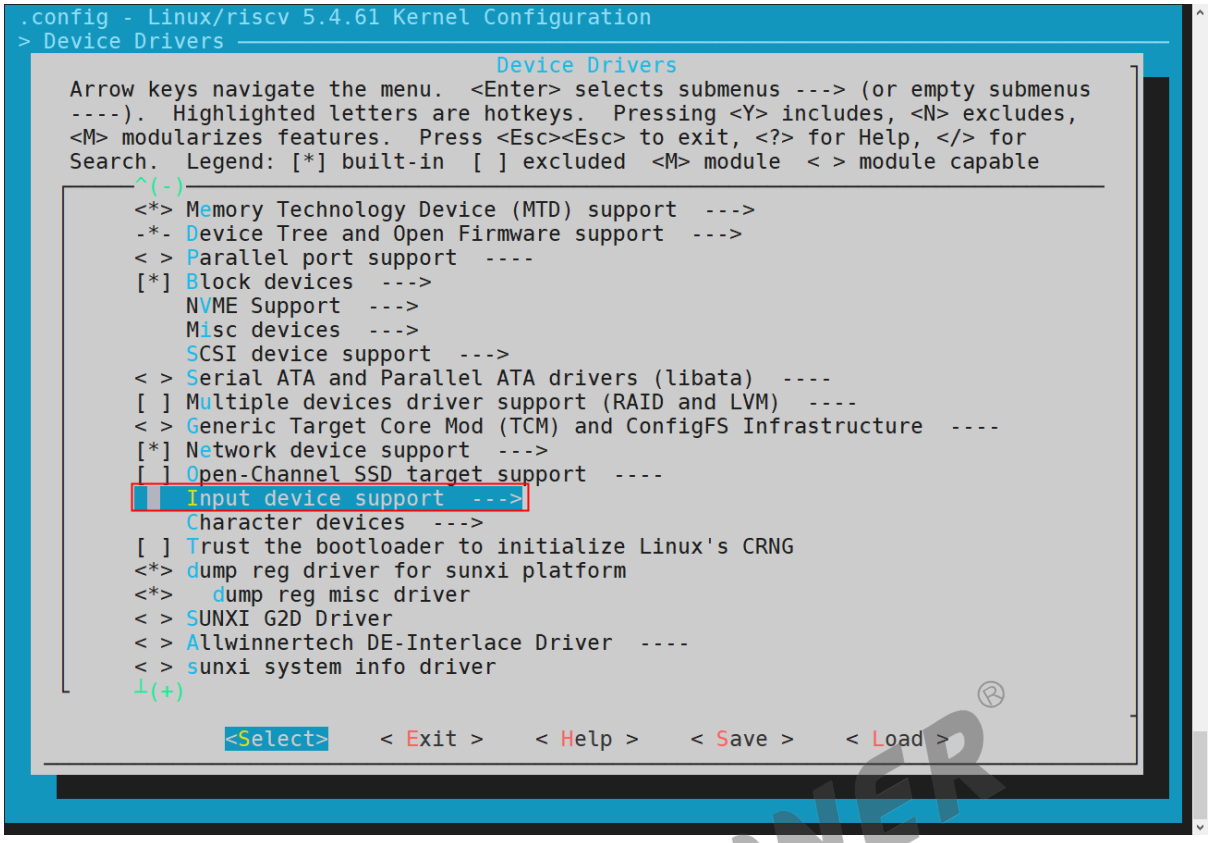

图 2-4: Input device support

- 选择 softwinner KEY BOARD support 加载按键支持，如下图所示：

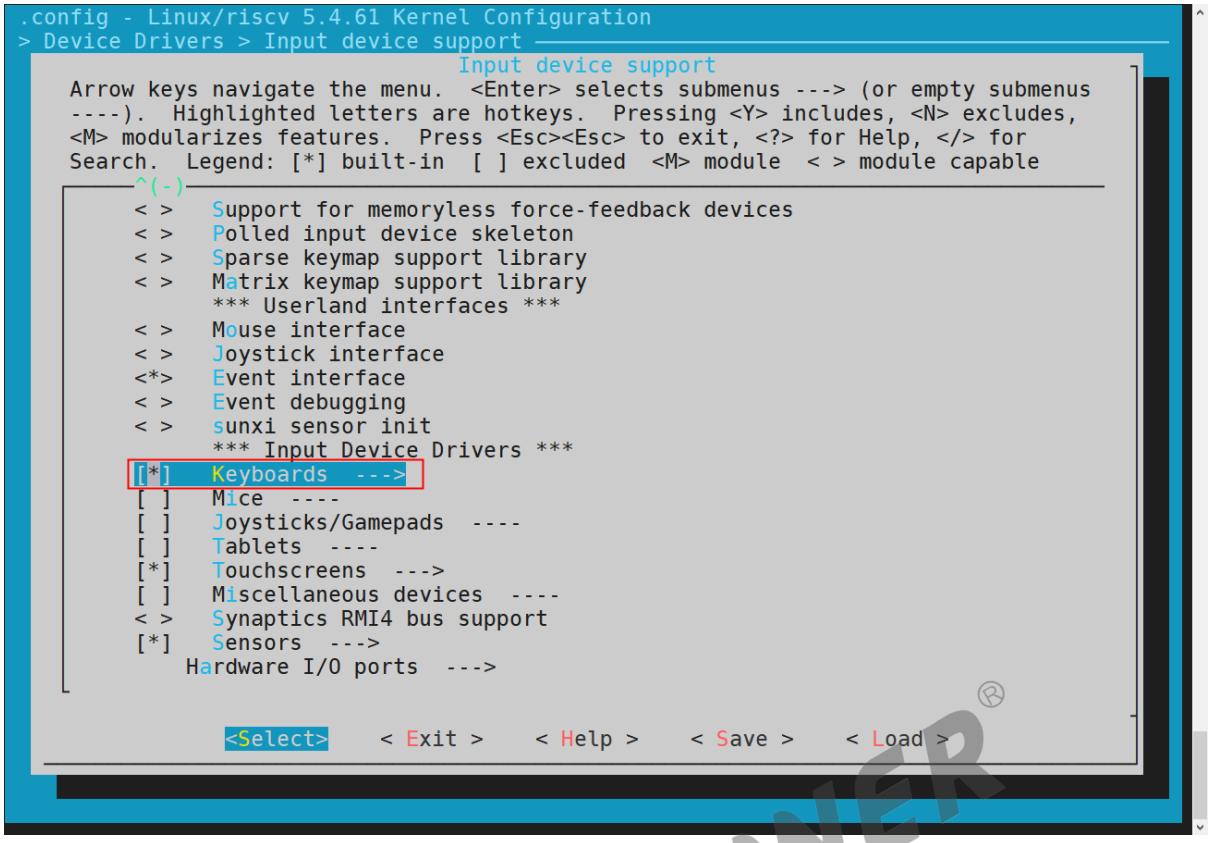

图 2-5: softwinner KEY BOARD support

- 返回 Input device support 选项，选择 Event Interface 选项，如下图所示：

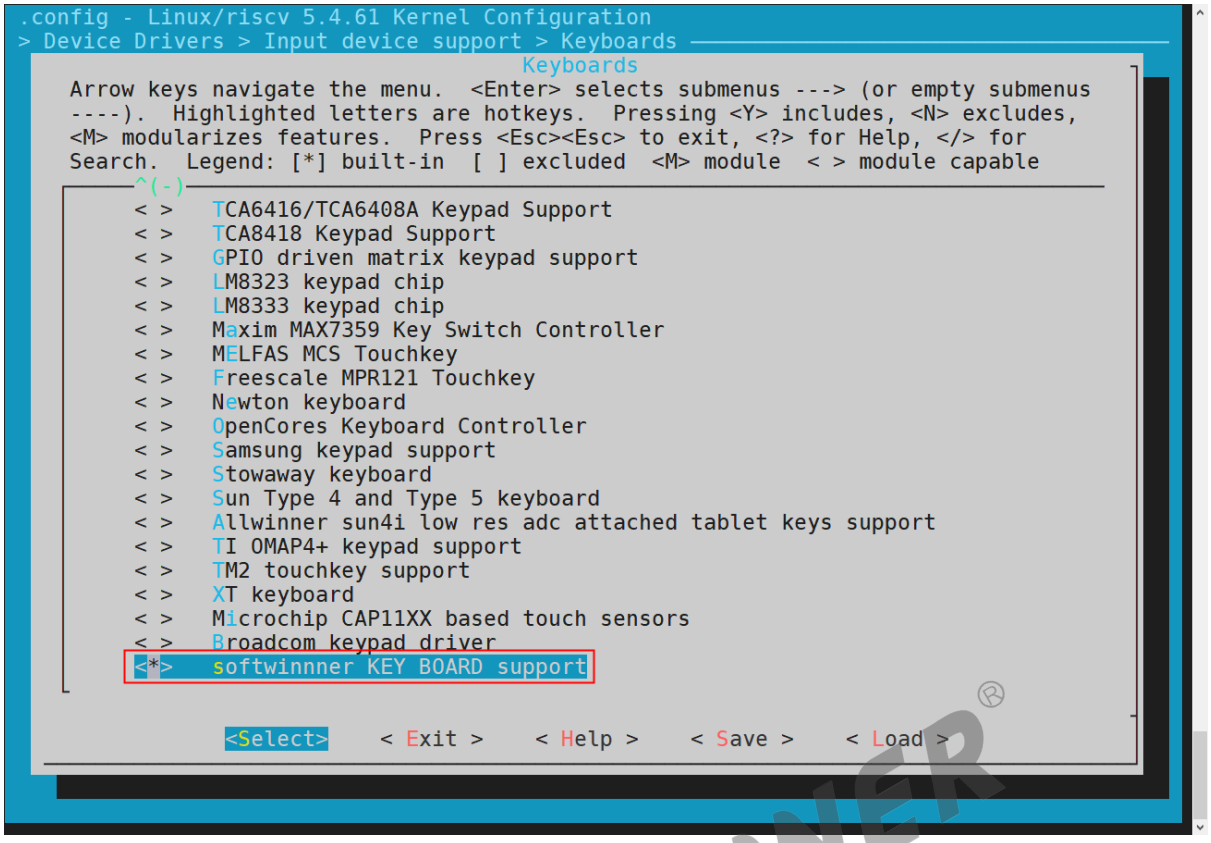

图 2-6: Event Interface

2.4 模块源码结构

KEY 模块的源码结构如下所示：

```
drivers/input/keyboard/
├── sunxi-keyboard.c
├── sunxi-keyboard.h
```

3 接口设计

3.1 内部接口

LRADC 模块在 Linux 内核中是作为字符设备使用，所以可以使用相关字符设备接口来对 LRADC 模块进行相应的读写和配置操作。相关定义在 evdev.c 文件里面。下面介绍几个比较有用的函数：

3.1.1 evdev_open()

- 函数原型：static int evdev_open(struct inode *inode, struct file *file)
- 功能描述：程序（C 语言等）使用 open(file) 时调用的函数。打开一个 LRADC 设备，可以像文件读写的方式往 LRADC 设备中读写数据
- 参数说明：
  - inode：inode 节点；
  - file：file 结构体；
- 返回值：文件描述符。

3.1.2 evdev_read()

- 函数原型：static ssize_t evdev_read(struct file *file, char __user *buf, size_t count,loff_t *ppos);
- 功能描述：程序（C 语言等）调用 read() 时调用的函数。读取 LRADC 模块上报事件数据；
- 参数说明：
  - file，file 结构体；
  - buf，写数据 buf；
  - ppos，文件偏移。
- 返回值：成功返回读取的字节数，失败返回负数。

3.1.3 evdev_write()

- 函数原型：static ssize_t evdev_read(struct file *file, const char __user *buf,size_t count, loff_t *ppos);

- 功能描述：程序（C 语言等）调用 write() 时调用的函数。像 LRADC 模块里面写入上报事件;
- 参数说明：
  - file，file 结构体;
  - buf，读数据 buf;
  - ppos，文件偏移。
- 返回值：成功返回 0，失败返回负数。

3.1.4 evdev_ioctl()

- 函数原型：static long evdev_read(struct file *file, unsigned int cmd, unsigned long arg);
- 功能描述：程序（C 语言等）调用 ioctl() 时调用的函数。可以控制相关的 LRADC 模块;
- 参数说明：
  - file，file 结构体;
  - cmd，指令;
  - arg，其他参数。
- 返回值：成功返回 0，失败返回负数。

找到 LRADC 模块对应的 eventX(如 dev/input/event0) 文件，就可以使用 C 语言的文件读写，控制函数来调用上述的接口。

3.2 外部接口

3.2.1 确认 LRADC 模块的 event 节点

在内核中，查看 /proc/bus/input/devices，确认 LRADC 的数据上报节点。

```
/ # cat /proc/bus/input/devices
I: Bus=0019 Vendor=0001 Product=0001 Version=0100
N: Name="sunxi-keyboard"
P: Phys=sunxikbd/input0
S: Sysfs=/devices/virtual/input/input0
U: Uniq=
H: Handlers=kbd event0
B: PROP=0
B: EV=3
B: KEY=800 c0040 0 0 10000000
```

3.2.2 读取 LRADC 模块的上报数据

直接在内核中 hexdump 相应的 event 节点，当按下按键后 LRADC 模块采集到数据的时候，可以看到 event 节点上报的数据。

```
/ # hexdump /dev/input/event0
0000000 bcc6 0000 3dbd 0009 0001 008b 0001 0000
0000010 bcc6 0000 3dbd 0009 0000 0000 0000 0000
0000020 bcc6 0000 0e1d 000b 0001 008b 0000 0000
0000030 bcc6 0000 0e1d 000b 0000 0000 0000 0000
```

其中，在读取到 event 节点的数据后，我们可以进行分析这些数据：每行的开头 4 个字节是 hexdump 打印的长度信息，后面跟着的是 16 字节的数据，struct timeval 占了 8 个字节，后面是 2 个字节的 type，2 个字节的 code，4 个字节的 value。具体实现也可以在内核代码中查看 input_event 结构体查看。

3.2.3 查看 LRADC 模块的中断次数

查看 /proc/interrupts 可以查看相关的 LRADC 模块中断次数。

```
/ # cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
 67:        280          0          0          0  wakeupgen  34 Level     sunxi-mmc2
 68:          0          0          0          0  wakeupgen  32 Level     sunxi-mmc0
 69:         12          0          0          0  wakeupgen  33 Level     sunxi-mmc1
 85:          0          0          0          0  wakeupgen  31 Edge      sunxikbd   //
    sunxi keyboard
```

4 模块使用范例

为了演示 LRADC 模块的使用，下面将演示用 C 语言对 LRADC 模块上报的数据进行读写：

```
#include <stdio.h>
#include <linux/input.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/time.h>
#include <limits.h>
#include <unistd.h>
#include <signal.h>

#define DEV_PATH "/dev/input/event0"   //difference is possible
const int key_exit = 102;
static int keys_fd = 0;

unsigned int test_keyboard(const char * event_file)
{
    int code = 0,i;

    struct input_event data;

    keys_fd = open(DEV_PATH, O_RDONLY);

    if(keys_fd <= 0)
    {
        printf("open %s error!\n", DEV_PATH);
        return -1;
    }

    for(i = 0;i < 10;i++)
    {
        read(keys_fd, &data, sizeof(data));

        if(data.type == EV_KEY && data.value == 1)
        {
            printf("key %d pressed\n", data.code);
        }
        else if(data.type == EV_KEY && data.value == 0)
        {
            printf("key %d releaseed\n", data.code);
        }
    }

    close(keys_fd);
    return 0;
}

int main(int argc,const char *argv[])
{
```

```
        int rang_low = 0, rang_high = 0;
        return test_keyboard(DEV_PATH);
    }
```

该 Demo 用来读取 LRADC 模块用于 KEY 的按键上报事件（其他类似）。其循环 10 次读取按键上报事件输入，并且显示出相应按键的值。

5 FAQ

无

著作权声明

商标声明

免责声明