**实验三：按键程序**

1. 实验目的
2. 熟悉linux系统，学会简单linux指令
3. 熟悉OK6410-A开发板的烧入步骤
4. 熟悉ARM寄存器，地址等。
5. 系统性的了解UBOOT和linux内核，yaffs2系统映像等知识
6. 掌握系统中断函数。
7. 掌握内部相关寄存器的操作方法，最终实现对外部设备的控制。
8. 实验环境

开发机环境

操作系统：ubuntu 12.04

交叉编译环境：arm-linux-gcc 4.3.2

6410板子内核源码：linux-3.0.1

目标板环境：OK6410-A linux-3.0.1

1. 实验原理

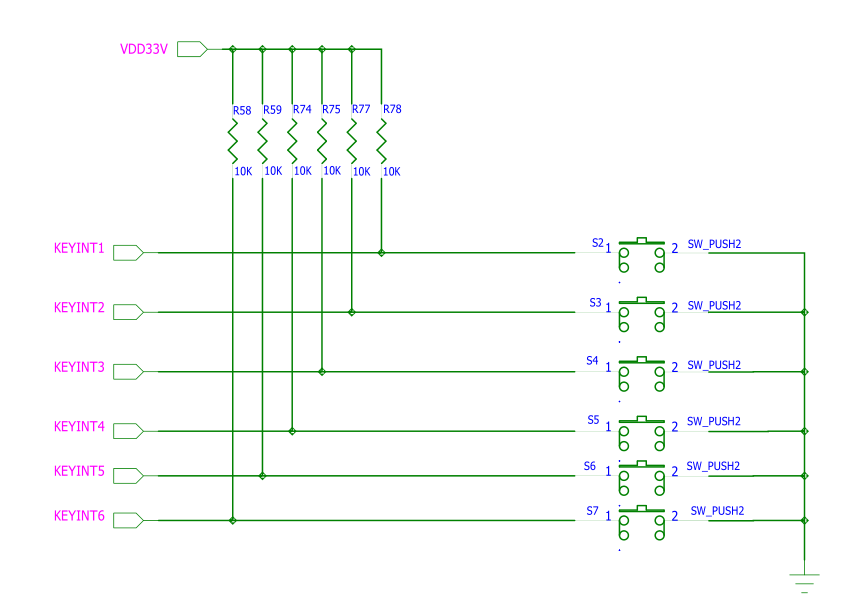


图1-OK6410按键原理图

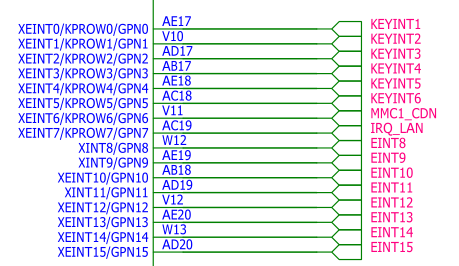
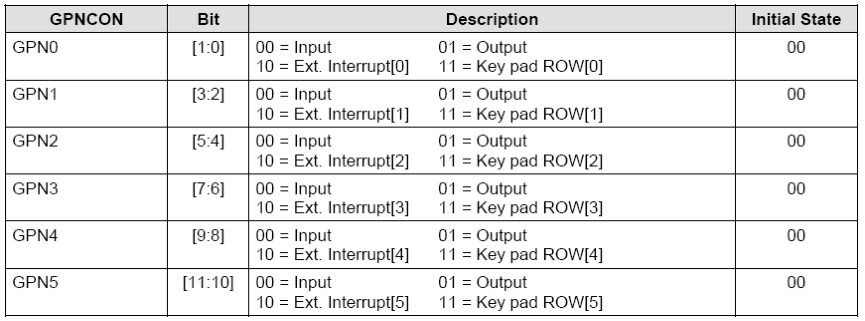
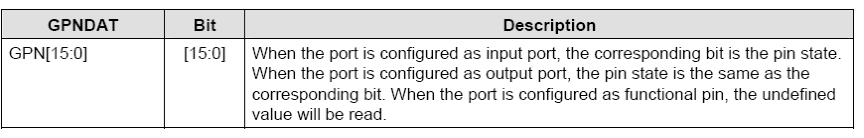


图2-按键原理图

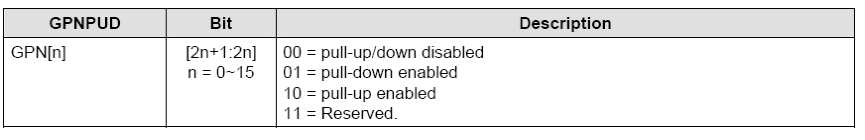
按键相关端口配置寄存器：



按键相关端口数据寄存器：



按键相关端口上拉电路使能寄存器：



1. 实验代码
2. 修改系统内核文件

在原有到内核中，按键的GPIO口被占用，需要进行相应到修改才能达到预期到效果，首先需要做的是安装libncurses 的相关软件，来实现对内核到编写。

内核编写过程：

找到内核：#make menuconfig

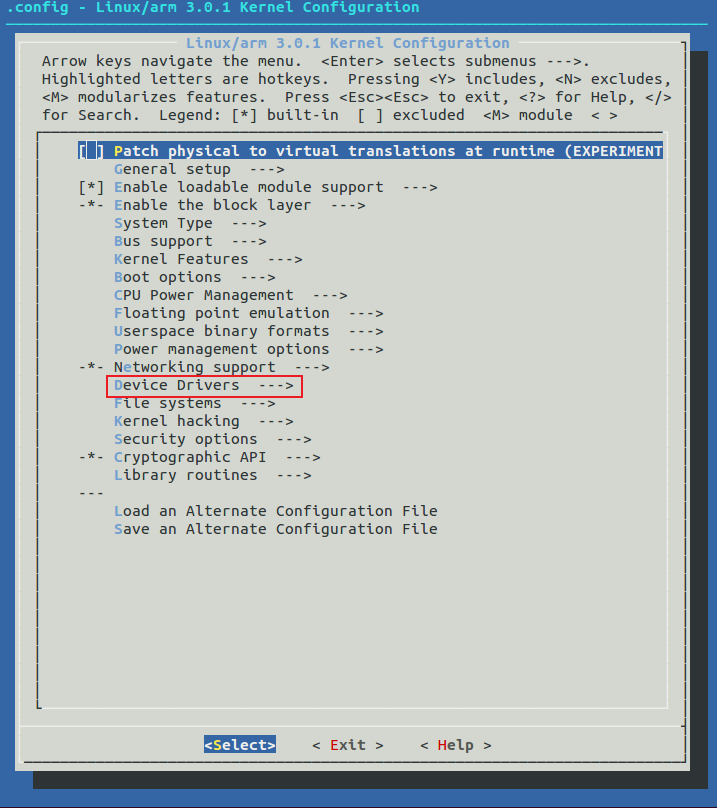
Device Drivers

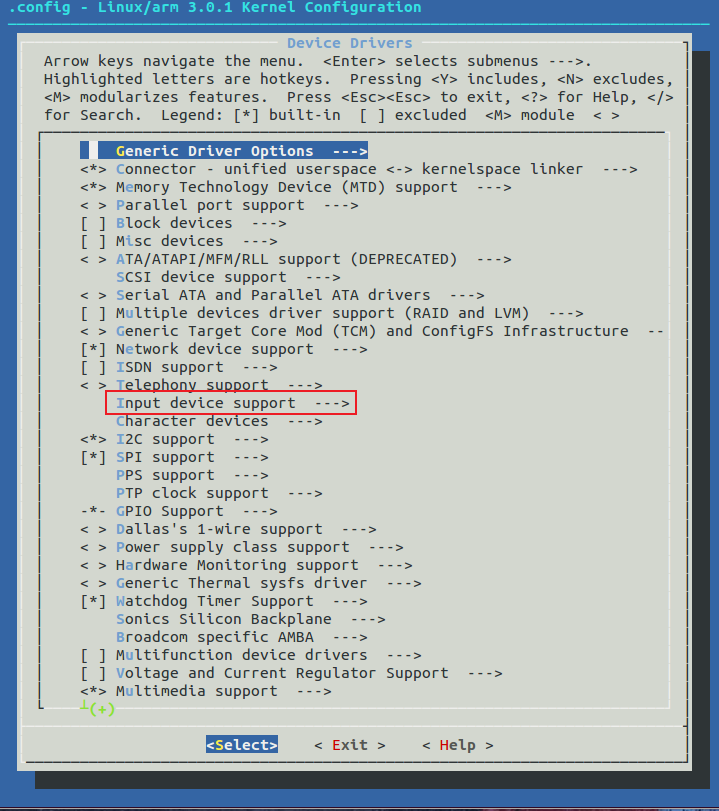
Input device support

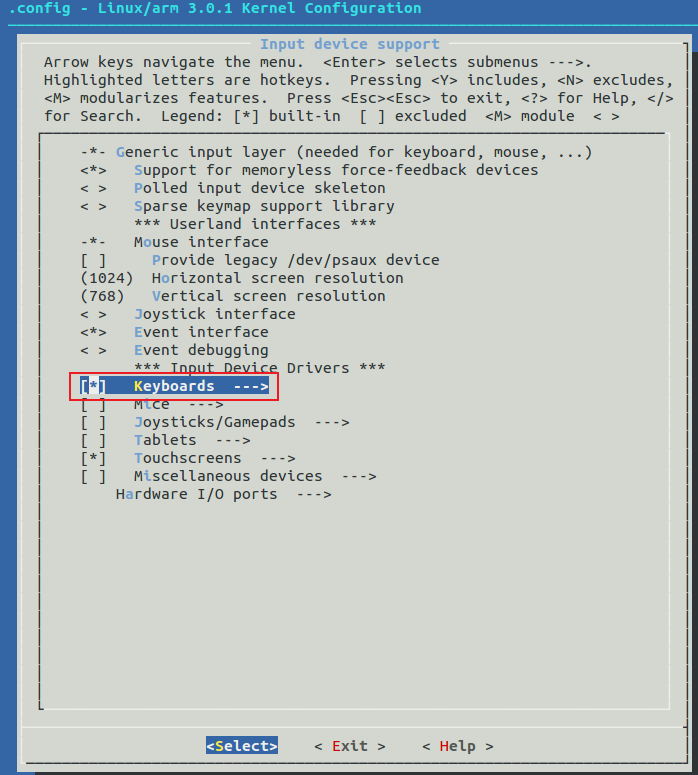
keyboards

把GPIO Buttons选项去掉

生成新的zImage镜像文件，重新烧写系统







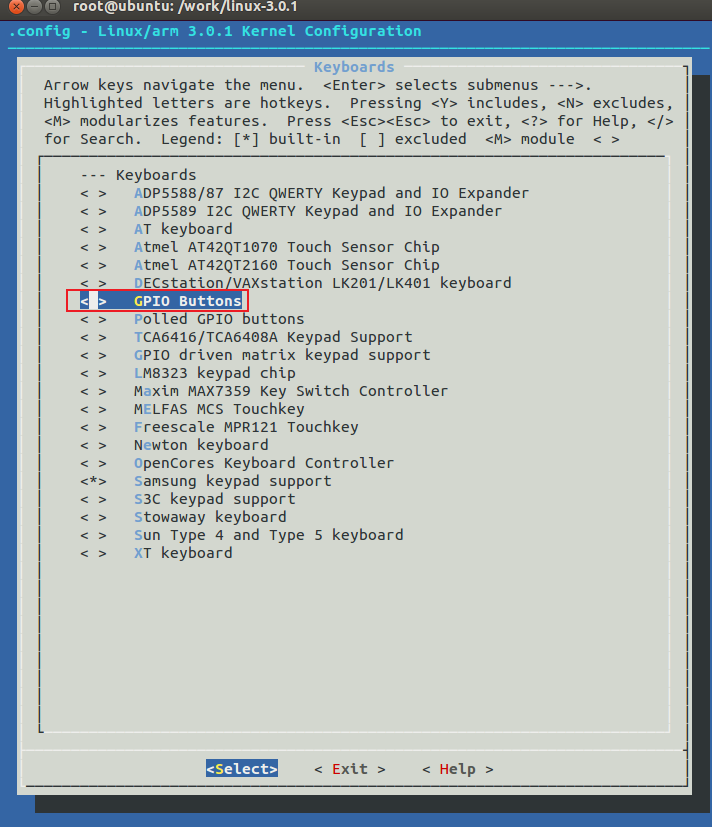


图3-内核修改

1. 编写驱动程序:driver\_key.c

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/init.h>

#include <linux/fs.h>

#include <linux/gpio.h>

#include <linux/types.h>

#include <linux/cdev.h>

#include <linux/interrupt.h>

#include <linux/sched.h>

#include <linux/device.h>

#include <linux/poll.h>

#include <linux/semaphore.h>

#include <linux/timer.h>

#include <asm/irq.h>

#include <asm/uaccess.h>

#include <mach/hardware.h>

#include <mach/irqs.h>

MODULE\_LICENSE**(**"GPL"**);**

#define DEVICE\_NAME "keyint"

#define KEYNUM 6

dev\_t devid**;**

//static DEFINE\_SEMAPHORE(key\_lock); //declare a mutex lock for keyint

//定义一个信号量

struct semaphore key\_lock**;**

static struct fasync\_struct **\***key\_async**;**

static struct timer\_list key\_timer**;**

struct key\_irq\_desc

**{**

int irq**;** //irq num

unsigned long flags**;** //irq flags,identified the way of irq here,eq.edge,level

char **\***name**;** //irq name

**};**

static struct key\_irq\_desc key\_irqs**[]** **=** **{**

//下降沿产生中断

**{**IRQ\_EINT**(**0**),** IRQF\_TRIGGER\_FALLING**,** "KEY1"**},**

**{**IRQ\_EINT**(**1**),** IRQF\_TRIGGER\_FALLING**,** "KEY2"**},**

**{**IRQ\_EINT**(**2**),** IRQF\_TRIGGER\_FALLING**,** "KEY3"**},**

**{**IRQ\_EINT**(**3**),** IRQF\_TRIGGER\_FALLING**,** "KEY4"**},**

**{**IRQ\_EINT**(**4**),** IRQF\_TRIGGER\_FALLING**,** "KEY5"**},**

**{**IRQ\_EINT**(**5**),** IRQF\_TRIGGER\_FALLING**,** "KEY6"**},**

**};**

/\*define a waiting queue here\*/

static DECLARE\_WAIT\_QUEUE\_HEAD**(**key\_waitq**);**

/\*define a event flag ev\_press\*/

static volatile int ev\_press **=** 0**;**

static volatile int press\_cnt**[**KEYNUM**]** **=** **{**0**,** 0**,** 0**,** 0**,** 0**,** 0**};**

/\*中断处理函数\*/

static irqreturn\_t keys\_interrupt**(**int irq**,** void **\***dev\_id**)**

**{**

volatile int **\***press\_cnt **=** **(**volatile int **\*)**dev\_id**;**

/\*set the pressed key flag(must do here due to not be static value)\*/

**\***press\_cnt **=** **\***press\_cnt **+** 1**;**

//延时10ms后执行定时器处理函数

mod\_timer**(&**key\_timer**,** jiffies **+** HZ **/** 100**);** //start timer after 10ms

**return** IRQ\_RETVAL**(**IRQ\_HANDLED**);**

**}**

//定时器处理函数

static void key\_timer\_func**(**unsigned long data**)**

**{**

ev\_press **=** 1**;**

//唤醒等待队列

wake\_up\_interruptible**(&**key\_waitq**);**

kill\_fasync**(&**key\_async**,** SIGIO**,** POLL\_IN**);**

**}**

static int key\_fasync**(**int fd**,** struct file **\***filp**,** int on**)**

**{**

printk**(**"Function key\_fasync\n"**);**

**return** fasync\_helper**(**fd**,** filp**,** on**,** **&**key\_async**);**

**}**

static unsigned key\_poll**(**struct file **\***file**,** poll\_table **\***wait**)**

**{**

unsigned int mask **=** 0**;**

//指明要使用的等待队列

poll\_wait**(**file**,** **&**key\_waitq**,** wait**);**

//返回掩码

**if** **(**ev\_press**)**

mask **|=** POLL\_IN **|** POLLRDNORM**;**

printk**(**"poll wait\n"**);**

**return** mask**;**

**}**

static int key\_open**(**struct inode **\***inode**,** struct file **\***file**)**

**{**

int num**;**

**if** **(**file**->**f\_flags **&** O\_NONBLOCK**)**

**{**

**if** **(**down\_trylock**(&**key\_lock**))**

**return** **-**EBUSY**;**

**}**

**else**

**{**

down**(&**key\_lock**);**

**}**

//为每个按键注册中断处理程序

**for** **(**num **=** 0**;** num **<** KEYNUM**;** num**++)**

**{**

request\_irq**(**key\_irqs**[**num**].**irq**,** keys\_interrupt**,** key\_irqs**[**num**].**flags**,** key\_irqs**[**num**].**name**,** **(**void **\*)&**press\_cnt**[**num**]);**

**}**

**return** 0**;**

**}**

static int key\_close**(**struct inode **\***inode**,** struct file **\***file**)**

**{**

int num**;**

//释放中断号

**for** **(**num **=** 0**;** num **<** 6**;** num**++)**

**{**

free\_irq**(**key\_irqs**[**num**].**irq**,** **(**void **\*)&**press\_cnt**[**num**]);**

**}**

up**(&**key\_lock**);**

printk**(**"key\_close free irqs\n"**);**

**return** 0**;**

**}**

static int key\_read**(**struct file **\***filp**,** char \_\_user **\***buff**,** size\_t count**,** loff\_t **\***offp**)**

**{**

// unsigned int err;

//判断是阻塞读还是非阻塞读

**if** **(**filp**->**f\_flags **&** O\_NONBLOCK**)**

**{**

**if** **(!**ev\_press**)**

**return** **-**EAGAIN**;**

**}**

**else**

**{**

/\*if ev\_press==0,then sleep\*/

/\*阻塞，当有按键按下时(中断)被唤醒\*/

wait\_event\_interruptible**(**key\_waitq**,** ev\_press**);**

**}**

//阻塞结束，有键按下了

ev\_press **=** 0**;**

//拷贝数据到用户空间

copy\_to\_user**(**buff**,** **(**const void **\*)**press\_cnt**,** min**(sizeof(**press\_cnt**),** count**));**

memset**((**void **\*)**press\_cnt**,** 0**,** **sizeof(**press\_cnt**));**

// printk("read and clean press\_cnt\n");

**return** 1**;**

**}**

static struct file\_operations key\_ops **=** **{**

**.**owner **=** THIS\_MODULE**,**

**.**open **=** key\_open**,**

**.**release **=** key\_close**,**

**.**read **=** key\_read**,**

**.**poll **=** key\_poll**,**

**.**fasync **=** key\_fasync**,**

**};**

static struct cdev **\***cdev\_keyint**;**

static struct class **\***keyint\_class**;**

//模块初始化函数

static int \_\_init s3c6410\_keyint\_init**(**void**)**

**{**

int val**;**

/\*timer initial \*/

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer**(&**key\_timer**);**

/\*初始化信号量\*/

init\_MUTEX**(&**key\_lock**);**

/\*register device\*/

val **=** alloc\_chrdev\_region**(&**devid**,** 0**,** 1**,** DEVICE\_NAME**);**

**if** **(**val**)**

**{**

**return** **-**1**;**

printk**(**"register keyint error\n"**);**

**}**

cdev\_keyint **=** cdev\_alloc**();**

cdev\_init**(**cdev\_keyint**,** **&**key\_ops**);**

cdev\_keyint**->**owner **=** THIS\_MODULE**;**

cdev\_keyint**->**ops **=** **&**key\_ops**;**

val **=** cdev\_add**(**cdev\_keyint**,** devid**,** 1**);**

**if** **(**val**)**

**{**

**return** **-**1**;**

printk**(**"add device error\n"**);**

**}**

keyint\_class **=** class\_create**(**THIS\_MODULE**,** DEVICE\_NAME**);**

device\_create**(**keyint\_class**,** **NULL,** devid**,** **NULL,** "%s"**,** DEVICE\_NAME**);**

printk**(**"KEY initialezed ^\_^\n"**);**

**return** 0**;**

**}**

static void \_\_exit s3c6410\_keyint\_exit**(**void**)**

**{**

cdev\_del**(**cdev\_keyint**);**

device\_destroy**(**keyint\_class**,** devid**);**

class\_destroy**(**keyint\_class**);**

unregister\_chrdev\_region**(**devid**,** 1**);**

**}**

module\_init**(**s3c6410\_keyint\_init**);**

module\_exit**(**s3c6410\_keyint\_exit**);**

1. 编写Makefile文件

ifneq **(**$**(**KERNELRELEASE**),)**

obj**-**m **:=** driver\_key**.**o

**else**

KDIR **:=** **/work/linux-**3.0.1

all**:**

make **-**C $**(**KDIR**)** M**=**$**(**PWD**)** modules ARCH**=**arm CROSS\_COMPILE**=**arm**-**linux**-**

clean**:**

rm **-**f **\*.**ko **\*.**o **\*.**mod**.**o **\*.**mod**.**c **\*.**symvers

endif

1. 编写测试文件:app\_key.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/ioctl.h>

int main**(**int argc**,** char **\*\***argv**)**

**{**

int fd**;**

int val**;**

int i**;**

int key\_value**[**6**];**

fd **=** open**(**"/dev/keyint"**,** 0**);**

**if** **(**fd **<** 0**)**

**{**

printf**(**"open devie error\n"**);**

**return** **-**1**;**

**}**

**while** **(**1**)**

**{**

val **=** read**(**fd**,** key\_value**,** **sizeof(**key\_value**));**

**if** **(**val **<** 0**)**

**{**

printf**(**"read error\n"**);**

**continue;**

**}**

**for** **(**i **=** 0**;** i **<** 6**;** i**++)**

**{**

**if** **(**key\_value**[**i**])**

printf**(**"KEY%d pressed\n"**,** **(**i **+** 1**),** key\_value**[**i**]);**

**}**

**}**

close**(**fd**);**

**return** 0**;**

**}**

1. 实验步骤
2. 编译驱动程序和测试程序

在终端中运行：#make命令，编译成功生生下列文件



在终端中运行：#arm-linux-gcc app\_key.c -o app\_key，编译成功生成文件



1. 将文件拷贝到SD卡
2. 将SD卡插入到OK6410开发板中
3. 在OK6410终端中运行程序

加载驱动：#insmod sdcard/driver\_key.ko

运行测试文件：#./sdcard/app\_key

卸载驱动程序;#rmmod sdcard/driver\_key.ko

1. 运行结果：按下6个按键，终端中会显示按键的号位。

#./app\_key

KEY1 pressed

KEY2 pressed

KEY3 pressed

KEY4 pressed

KEY5 pressed

KEY6 pressed

1. linux批处理代码：文件命名为ll

#!/bin/sh

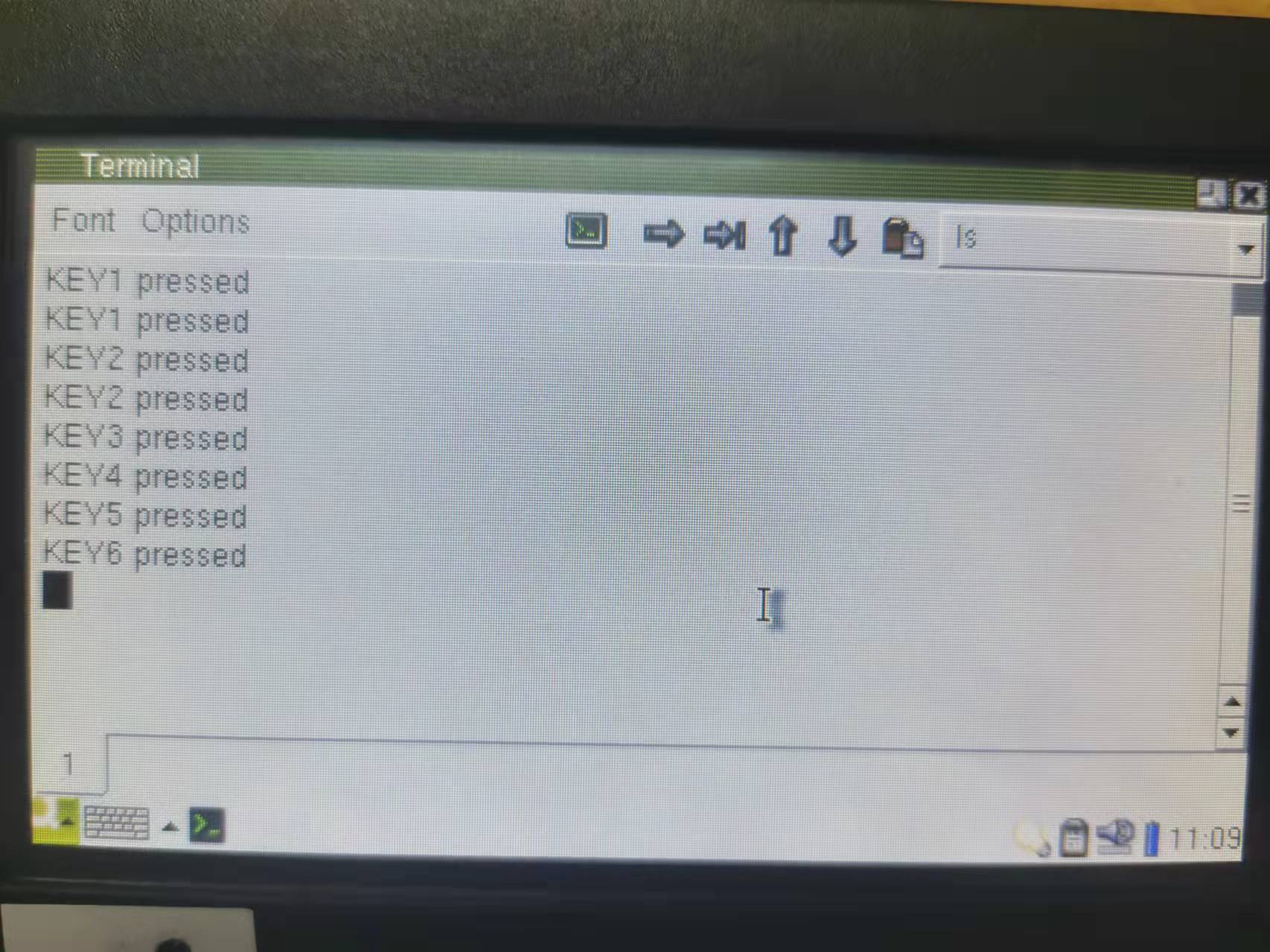
**insmod** **/**sdcard**/**test3**/**driver\_key.ko

**cd** sdcard**/**test3**/**

**./**app\_key

直接在OK6410中运行:#./sdcard/ll

可以观察到同样的测试结果



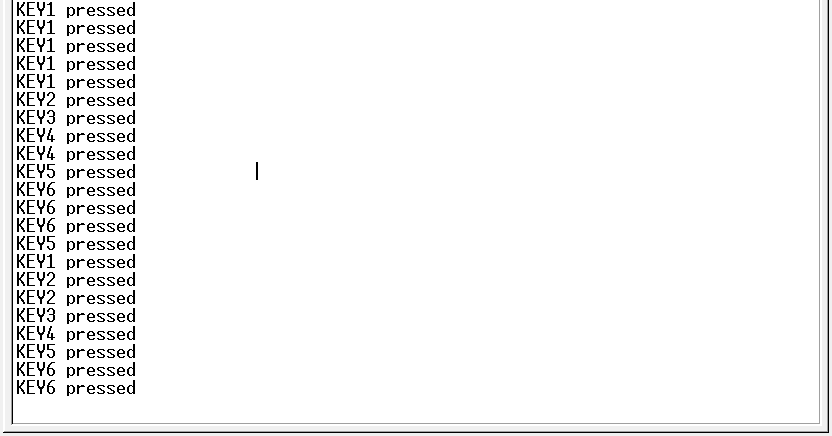


图5-按键运行结果

1. 实验总结

本次实验了解了Linux内核的修改过程，可以在后续的实验操作中删减或者添加内核文件。加深了对中断函数的理解，熟练掌握了系统的烧录，驱动程序加载，测试文件运行，驱动程序卸载。