目录

[1 Rk3308 简介： 1](#_Toc4763001)

[2使用adb连接板子 1](#_Toc4763002)

[2.1 adb安装 1](#_Toc4763003)

[2.2 使用putty连接板子 1](#_Toc4763004)

[2.3 上传下载文件到开发板 1](#_Toc4763005)

[2.4 更新内核 2](#_Toc4763006)

[3编译下载内核 2](#_Toc4763007)

[3.1 编译uboot(32位) 2](#_Toc4763008)

[3.2 编译内核 2](#_Toc4763009)

[3.3 编译驱动的Makefile（32位内核） 2](#_Toc4763010)

[3.4 板子匹配的设备树 3](#_Toc4763011)

[3.5 添加lcd驱动到内核 3](#_Toc4763012)

[3.6 编译应用的编译器(32位编译器) 3](#_Toc4763013)

[3 GPIO口使用 3](#_Toc4763014)

# 1 Rk3308 简介：

四核64位核心处理器，整合高性能CODEC音频编解码器，VAD。支持国内外多种人工智能及物联网操作系统，可配置6个麦克风阵列及POE扩展板。可用于智能音频方案、语音识别等项目。

**处理器核心：**cortex-a35核心配置，主频1.3GHZ,

**开源开发板：**ROC-RK3308-CC 是一款集多种功能为一体的AI+IOT开源主板。

**vad:(静音抑制)：**语音活动检测(Voice Activity Detection,VAD)又称语音端点检测,语音边界检测。目的是从声音信号流里识别和消除长时间的静音期，以达到在不降低业务质量的情况下节省[话路](https://baike.baidu.com/item/%E8%AF%9D%E8%B7%AF/5922604)资源的作用，它是IP电话应用的重要组成部分。静音抑制可以节省宝贵的[带宽](https://baike.baidu.com/item/%E5%B8%A6%E5%AE%BD/266879)资源，可以有利于减少用户感觉到的端到端的时延。

# 2使用adb连接板子

## 2.1 adb安装

下载adb安装包，然后将system32里面的cmd命令行拷贝到adb文件夹里面。打开cmd工具执行adb shell登陆到板子的根目录中。如果能进入根目录证明配置成功

## 2.2 使用putty连接板子

下载putty的adb版本，放入adb文件夹里面，打开选择adb模式，Host name里面填写transport-usb,然后点击open进入到板子的根目录。

## 2.3 上传下载文件到开发板

打开adb目录下的cmd命令行工具，不要执行adb shell 登陆到板子。

adb push <本地路径> <远程路径>

adb push "E:\hello.txt" "userdata/mytest/" //板子data区可读写，注意/是linux下，\window

//下使用。

adb pull <远程路径> <本地路径>

adb push "userdata/mytest/ hello.txt " "E:\ "

## 2.4 更新内核

瑞芯微开发工具软件下载，首先按住reset 插入usb线，当软件提示发现一个ADB

设备后点击切换按钮进入loader模式，点击执行开始下载。

该软件可以单独更新某个固件。如果不知道下载地址，那么可以读取设备内部的分区表，分区表就是nandflash存放uboot，kernel，rootfs等的分区信息：

GPT part: 0, name: uboot, start:0x2000, size:0x800

GPT part: 1, name: trust, start:0x2800, size:0x800

GPT part: 2, name: misc, start:0x3000, size:0x800

GPT part: 3, name: recovery, start:0x3800, size:0x6000

GPT part: 4, name: boot, start:0x9800, size:0x3000

GPT part: 5, name: rootfs, start:0xc800, size:0x11800

GPT part: 6, name: oem, start:0x1e000, size:0xe000

GPT part: 7, name: userdata, start:0x2c000, size:0x97df

# 3编译下载内核

## 3.1 编译uboot(32位)

source buildroot/build/envsetup.sh //选择3,32位debug版本

Cd uboot/

./make.sh evb-aarch32-rk3308

## 3.2 编译内核

source buildroot/build/envsetup.sh //选择3,32位debug版本

cd kernel

make ARCH=arm rk3308\_linux\_aarch32\_debug\_defconfig

make ARCH=arm menuconfig

make ARCH=arm savedefconfig

cp defconfig arch/arm/configs/rk3308\_linux\_aarch32\_debug\_defconfig

//make ARCH=arm rk3308-voice-module-board-v10-aarch32.img

make ARCH=arm rk3308-voice-module-amic-mainboard-v10-aarch32.img//

## 3.3 编译驱动的Makefile（32位内核）

编译驱动的终端必须是配置内核的终端，而且使用root用户。

KERN\_DIR = /home/chao/rk3308\_linux/kernel

all:

make -C $(KERN\_DIR) M=`pwd` ARCH=arm modules //指定是32位

clean:

make -C $(KERN\_DIR) M=`pwd` ARCH=arm modules clean

rm -rf modules.order

obj-m += gpio\_keys.o

## 3.4 板子匹配的设备树

rk3308-voice-module-amic-mainboard-v10-aarch32.

#include "rk3308-voice-module-v10-aarch32.dtsi"

#include "rk3308-voice-module-mainboard-v10-aarch32.dtsi"

32位dts也包含arm64/boot/dts/rk3308.dtsi

## 3.5 添加lcd驱动到内核

以我编写的spilcd驱动为例

1. driver/char/目录下新建目录spilcd
2. 把spilcd.c拷贝到spilcd目录里面
3. 创建Makefile和Kconfig两个文件//Kconfig用于图像配置，Makefile用于编译
4. Kconfig内容如下：

menu "RK3308 SPILCD"

config RK3308\_SPILCD

tristate "the driver of spilcd for rk3308"

depends on CPU\_RK3308 //表示CPU\_RK3308选中才能配置该选项

default Y

help

this is a driver for spi lcd

endmenu

1. 修改Makefile添加如下行

obj-$(CONFIG\_RK3308\_SPILCD) += spilcd.o

// CONFIG\_RK3308\_SPILCD就是前面的config RK3308\_SPILCD 中的RK3308\_SPILCD加上//CONFIG\_

1. 修改Drivers/char目录下面的Kconfig

source "drivers/char/spilcd/Kconfig"//建立和子目录的联系

endmenu

1. 修改Drivers/char目录下面的Makefile,在最后添加

obj-$(CONFIG\_RK3308\_SPILCD) += spilcd/ //建立和子目录的联系

## 3.6 编译应用的编译器(32位编译器)

/home/chao/rk3308\_linux/buildroot/output/rockchip\_rk3308\_32\_release/host/bin/arm-rockchip-linux-gnueabihf-gcc

# 3 GPIO口使用

Firefly-RK3399的dts:

对引脚的描述与Firefly-RK3288有所区别，GPIO0\_B4被描述为：<&gpio0 12 GPIO\_ACTIVE\_HIGH>，这里的12来源于：8+4=12，其中8是因为GPIO0\_B4是属于GPIO0的B组，如果是A组的话则为0，如果是C组则为16，如果是D组则为24，以此递推，而4是因为B4后面的4。 GPIO\_ACTIVE\_HIGH表示高电平有效，如果想要低电平有效，可以改为：GPIO\_ACTIVE\_LOW，这个属性将被驱动所读取。

Rk3308 dts:

一般按键引脚配置：

gpio-keys {

compatible = "gpio-keys";

autorepeat;

pinctrl-names = "default";

pinctrl-0 = <&pwr\_key>;

power {

//gpio0 A组的第6个引脚

gpios = <&gpio0 RK\_PA6 GPIO\_ACTIVE\_LOW>;

linux,code = <KEY\_POWER>;

label = "GPIO Key Power";

wakeup-source;

debounce-interval = <100>;

};

};

&pinctrl {

pinctrl-names = "default";

pinctrl-0 = <&rtc\_32k>;

buttons {

pwr\_key: pwr-key {

rockchip,pins = <0 RK\_PA6 RK\_FUNC\_GPIO &pcfg\_pull\_up>;

};

};

};

};

# 4 spi 接口Lcd驱动

## 4.1 设计思路

使用帧缓冲区来操作，驱动中申请缓冲区，应用层映射该缓冲区，应用层像缓冲区中写入数据，驱动每隔30ms读取缓冲区数据发送到lcd设备。定时时间间隔发送采用延时工作队列delayed\_worker。先编程实现在硬件驱动中刷新lcd,然后实现支持framebuffer的驱动，从应用层测试。

## 4.2 硬件信息

### 4.2.1 rk3308 SPI功能

摩托罗拉SPI协议

支持8位和16位。

时钟频率50MHZ

SPI 4中传输模式配置

### 4.2.2配置内核支持瑞芯微spi

Device Drivers ‐‐‐>  
[\*] SPI support ‐‐‐>  
<\*> Rockchip SPI controller driver

### 4.2.3 lcd 屏接口模式配置

接口类型总共有两种：8080 MCU接口和串口。

通过外部IM[3:0]引脚来控制。1110模式下配置为四线spi接口，SCL,SDI,D/CX,SDO,CS。 1101模式下配置为3线9位接口，SCL,SDI,SDO,CSX。 0101模式下三线9位串行接口，SCL,SDA,CSX, SDA线同时作为输入和输出口。0110四线8位模式，SCL,SDA,D/CX,CSX,SDA线同时作为输入和输出口。

原理图采用的是三线9位接口。Rk3308只能支持四线8位。

### 4.2.4 lcd屏spi接口引脚描述

CSX :片选信号，低电平有效。

WRX(D/CX) :四线系统命令数据选择。低电平表示命令，高电平表示数据。

D/CX(SCL):串行时钟

SDI/SDA:四线输入信号，三线输入输出信号

SDO: 四线输出信号。主机master输出数据端口

位传输顺序：最高位先传输。9位模式下，D/CX先传输。

主机通过SDA发送读命令后，芯片会在随后的时钟位发送数据到SDA。

## 4.2 dts配置

直接修改/arch/arm64/boot/dts/rk3308.dtsi

spi2: spi@ff140000 {

compatible = "rockchip,rk3308-spi", "rockchip,rk3066-spi";

reg = <0x0 0xff140000 0x0 0x1000>;

interrupts = <GIC\_SPI 17 IRQ\_TYPE\_LEVEL\_HIGH>;

#address-cells = <1>;

#size-cells = <0>;

clocks = <&cru SCLK\_SPI2>, <&cru PCLK\_SPI2>;

clock-names = "spiclk", "apb\_pclk";

max-freq = <80000000>;

dmas = <&dmac1 16>, <&dmac1 17>;

dma-names = "tx", "rx";

pinctrl-names = "default", "high\_speed";

pinctrl-0 = <&spi2\_clk &spi2\_csn0 &spi2\_miso &spi2\_mosi>;

pinctrl-1 = <&spi2\_clk\_hs &spi2\_csn0 &spi2\_miso\_hs &spi2\_mosi\_hs>;

status = "okay";

spilcd@0{

status="okay";

compatible="spilcd";

reg=<0>;

spi-max-frequency=<40000000>;

reset-gpios=<&gpio0 RK\_PC4 GPIO\_ACTIVE\_LOW>;

dc-gpios=<&gpio0 RK\_PC1 GPIO\_ACTIVE\_LOW>;

};

};

引用修改：

rk3308-voice-module-mainboard-v10-aarch32.dtsi中引用rk3308.dtsi

rk3308保持官方文件不变。只把需要更改的重新设置。这样就更改了rk3308.dtsi中的设定值。这样做的好处是，由于多个板子公用rk3308.dtsi文件，那么改变了rk3308.dtsi就改变了所有板子的配置。这不是我们想要的，我们只想更改我们自己的板子配置。

&spi2{

status = "okay";

max-freq = <48000000>;

spilcd@0{

status="okay";

compatible="spilcd";

reg=<0>;

spi-max-frequency=<12000000>;

dc-gpios=<&gpio0 RK\_PC1 GPIO\_ACTIVE\_LOW>;

reset-gpios=<&gpio0 RK\_PC4 GPIO\_ACTIVE\_LOW>;

};

};

## 4.3 驱动例程

完整例程查看rk3308驱动myfb.c ，里面有详细的注释。

## 4.4 测试

打开adb模式cmd命令行，在没有登录板子情况下执行：

adb push “myfb.ko” “tmp”

adb push “HZK16” “tmp”

adb push “show\_font” “tmp”

连接调试串口

adb shell 登录到板子

cd /tmp/

insmod myfb.ko

chmod 777 show\_font

./show\_font 发现板子上面显示字符A和汉字中