修订记录

Date	Version	Author	Description
2023/12/05	V0.1	jamin.zhong	Init version

本文档(*.md)使用Typora工具编写，建议使用Typora工具阅读此文档。

Note: 此文档描述部分可能不准确，以实际调试为准。

最后更新时间： 202402291555

Overview

名称描述

cx3588-Linux： 表示包括 `cx3588-Linux5.10` `cx3588-Debian11` `cx3588-Xubuntu20.04`；

系统用户名、密码

cx3588-Linux5.10

系统默认有一个用户

- 用户名： `root`, 密码： `rockchip`；

cx3588-Debian11

系统默认有两个用户： `linaro`, `root`；

- 用户名： `linaro`, 密码： `linaro`；
- 用户名： `root`， 密码：默认没有设置密码，根据自己需求设置root密码： `sudo passwd root`

cx3588-Xubuntu20.04

系统默认有两个用户：`linaro`，`root`；

- 用户名：`linaro`，密码：`linaro`；
- 用户名：`root`， 密码：默认没有设置密码，根据自己需求设置root密码：`sudo passwd root`

Linux 系统注意的地方

- **屏幕尽量不要旋转，旋转很耗性能；**
  - 用原生的横屏或者竖屏；
- 双屏显示，使用相同的分辨率；

CX3568-Linux系统推荐

Linux 有三种系统：

- CX3588-Debian11： 带发行版桌面；
- CX3588-Linux5.10： 不带发行版桌面，Buildroot Linux系统，，默认是weston 的demo桌面，跑一些Qt应用；
- CX3588-Xbuntu20.04： 带发行版桌面；

系统推荐： CX3588-Debian11 **>** CX3588-Linux5.10 **>** CX3588-Xubuntu20.04;

原因：

- Debian 和 Buildroot Linux系统，RK原厂提供技术支持；
  - Ubuntu 系统，RK原厂不提供技术支持，不提供源码；
- Debian 是社区版操作系统，无版权，操作命令以及包管理几乎和ubuntu无异；
- Ubuntu 有版权，并不免费：如果做产品，是需要ubuntu官方授权的，需要注册和缴纳ubuntu core的费用；
- Buildroot 开发难度相对较大；

开发版操作系统的选择 (摘抄网上)

开发版操作系统的选择

有同学纠结于各种嵌入式操作系统如何选择，而且我们板子是双系统启动的，不知道哪个操作系统合适，我们简单分析下目前主流的支持ARM的操作系统：

1. Android
优点： UI开发非常简单，API丰富，接口通用，除了RKNN、RGA特殊的加速单元以外，其他所有的加速单元均可以通过android自带的API操作。兼容性稳定性都最佳。
缺点： 自身占用了较多的系统资源（例如GPU、RGA），并且编译链(android-gcc)的libc(bionic)特殊，移植一些第三方开源库难度较大。不支持python。
建议： 如果产品需要UI界面，强烈推荐使用Android作为基础平台，开发周期短，可移植性高。

2. Fedora

优点： 桌面操作系统，大部分人很熟悉，容易上手。源丰富，开源库可以rpm直接安装，不需要重新移植。也支持python。

缺点： 官方对ARM支持并不友好，对硬件平台几乎0优化，全部跑在CPU上（PC上也如此）。不适合做产品。UI开发非常麻烦，linux并没有统一的UI框架可用。不要想指望qt，qt的代码量不亚于一整个安卓系统，并且RK对QT是没有官方支持的。

建议： 如果只是科研学习，用fedora最好，但是不要指望fedora能优化到什么极限，并且python效率其实很低下，发挥不出平台优势。

3. Centos

优点： 最大优点就是稳定，无UI界面，系统资源最大程度的留给了你的应用。也可以rpm直接安装第三方库，支持python开发。

缺点： 因为他为了保持稳定，所以系统基础库版本是很低的，例如libc目前还停留在gcc4.8时代，所以如果需要用到c++14新特性，都必须用自己的libc库。

建议： 作无UI界面的产品非常合适，稳定性高，资源占用低，APP可以利用到3399Pro的全部资源。作为Arm服务器产品也非常合适。

4. Ubuntu

优点： 开发者多，官方支持丰富。优点同Fedora。

缺点： 同Fedora所有缺点。并且还多一个缺点，Ubuntu并不是免费的，如果你要做产品，是需要ubuntu官方授权的，需要注册和缴纳ubuntu core的费用。

建议： 自己科研学习玩玩可以，不建议做产品使用。

5. Debian (目前Toybrick仅提供debian作为linux开发系统)

优点：Ubuntu的前身，社区版操作系统，无版权，操作命令以及包管理几乎和ubuntu无异

缺点：同Fedora所有缺点。

建议：和Feodra相同，熟悉redhat的人可以选择Fedora，熟悉ubuntu的人可以选择debian

6. buildroot

优点： 自己组合想要的组件，灵活，可自己组装UI平台。资源占用最少，nand空间可以最大程度交给app使用。

缺点： 不适合新手，UI开发也极其复杂。

建议： 如果你的产品nand空间非常有限，可以考虑使用buildroot自己建立Linux系统。否则不建议用这个。

7. 其他发行版Linux系统

其他就不介绍了，看大家自己熟悉程度选择，也要看官方是否有arm的发行版系统。如果能下到arm/aarch64版的rootfs，就可以直接烧入3399Pro运行。

当然桌面版Linux最大问题就是都CPU渲染，对ARM GPU支持都很差，所以很多人会感觉拖动卡顿、闪屏等问题，都很正常。

Linux其实都大同小异，优缺点很类似，并无太大区别。当然如果要是找到一个官方支持ARM和Mali GPU非常好的发行版Linux，记得告诉我们哦。

Codes path

```
git clone git@172.0.0.249:cx3588_linux.git

# cx3588_linux5.10
git checkout -b cx3588_linux5.10 origin/cx3588_linux5.10

# cx3588_debian11
git checkout -b cx3588_debian11 origin/cx3588_debian11

# cx3588_xubuntu20.04
git checkout -b cx3588_xubuntu20.04 origin/cx3588_xubuntu20.04
```

How to compile

全部编译

```
source envsetup.sh rockchip_rk3588
./build.sh device/rockchip/rk3588/rockchip_rk3588_touchthink_lp4_v10_defconfig
./build.sh
```

分步编译

- 配置环境

  ```
  source envsetup.sh rockchip_rk3588
  ```

- 选择板级配置

  ```
  ./build.sh
  device/rockchip/rk3588/rockchip_rk3588_touchthink_lp4_v10_defconfig
  ```

- Uboot编译

  ```
  ./build.sh uboot
  ```

- Kernel编译

  ```
  ./build.sh kernel
  ```

- Recovery编译

  ```
  ./build.sh recovery
  ```

- 文件系统编译（Linux5.10）

  ```
  ./build.sh rootfs
  ```

- 文件系统编译(Debian11)

```
./build.sh debian
```

- 文件系统编译（Xubuntu20.04）

```
./build.sh xubuntu20
```

- 打包

```
./build.sh updateimg
    --打包生成的文件： rockdev/update.img
```

Clean with compile

Clean uboot

Clean kernel

Clean buildroot

Clean Debian

Clean Ubuntu

Clean all

加快编译速度

提前下载源码包

- `buildroot/dl` 这个目录存放编译所需要的源码包，可以提前下载好；

  编译前，把这个目录的文件拷贝到源码根目录，解压，然后编译，可以加快速度；

  `\\172.0.0.249\share\jamin3\cx3588-linux\modules\compile-with-speedup\`

多线程编译

编译可能遇到的问题

现象跟jws3399类似，这里以jws3399举例：

```
编译过程，可能会遇到两种情况：
1. 编译时，下载部分源码很慢；
    可以手动下载源码，然后拷贝到这个目录 jws3399_linux/buildroot/dl/;
    然后再单独编译这个模块。
```

```
   for example:
   例如下载coreutils-8.27.tar.xz很慢，
   咱们去官网下载：http://ftpmirror.gnu.org/coreutils/coreutils-8.27.tar.xz
   cp coreutils-8.27.tar.xz buildroot/dl/
   make coreutils-rebuild
   ./build.sh rootfs

2. 编译rootfs，编译失败；
    如果编译某个文件targets失败，需要单独多编译这个targets；
    targets编译成功，然后再继续编译文件系统rootfs
    make targets-rebuild
    ./build.sh rootfs
eg:
编译host-gcc-final-6.5.0失败，咱们需要多次编译host-gcc-final这个target，直到编译ok.
package/pkg-generic.mk:254: recipe for target '/home/zjm/workspace/jws3399/6-
12/jws3399_linux/buildroot/output/rockchip_rk3399_recovery/build/host-gcc-final-
6.5.0/.stamp_built' failed


make host-gcc-final-rebuild
./build.sh rootfs
```

Buildroot 开发 (可忽略)

Buildroot配置

```
# 配置buildroot:
source envsetup.sh rockchip_rk3588

cd buildroot
make menuconfig
make savedefconfig
```

Busybox 配置

```
make busybox-menuconfig
make busybox-update-config
```

建议: 第一次编译source code, 晚上下班用脚本循环编译

第一次编译文件系统(rootfs), 可能会出现多次编译失败，这时需多编译几次，就可以编译过，只要编译通过了，后面编译每次都会成功(正常)。

所以建议晚上用脚本循环编译:

```
#!/bin/sh

source envsetup.sh rockchip_rk3588
./build.sh device/rockchip/rk3588/rockchip_rk3588_touchthink_lp4_v10_defconfig

while true
do
    ./build.sh
done
```

出现编译失败可能的原因:

1. 网络问题，有时候编译时拉取代码失败(大概率);

> 第一次编译时，会检测本地是否有指定代码，没有则会先去拉取下载代码，然后再编译;
> 现在基本需要下载的代码，都已经放在`buildroot/dl/`目录，提交到服务器;

refer

rockchip-linux--github

http://linux-rockchip.org/

Welcome to Firefly Linux 开发指南

烧录

```
可以进入loader 或者 Maskrom 模式烧写，一般在loader模式下烧写;

进入loader模式，有以下方式:
1. reboot loader    // 这个方式简单快速
2. 进入uboot 命令行模式:  reboot loader
3. 按键:  先按复位键(reset)，然后按RECOVERY键，然后松开reset键;
```

====分割线: Debian====

以下部分，是Debian相关部分;

Debian 系统配置

用户-应用配置目录

- 路径: `/home/linaro/.config/xfce4`

- 平时一些应用的设置，保存在这个路径；
- 删除掉这些配置，会默认还原为出厂设置；

默认应用的配置目录

- 路径：`/etc/xdg/xfce4`

Panel config

eg:

```
+# Panel remove actions plugin;
+sed -i '/<value type="int" value="14"\/>/d' /etc/xdg/xfce4/panel/default.xml
+
```

Whiskermenu config

eg

```
+# whiskermenu disable lockscreen, enable hibernate/restart/shutdown;
+{
+       echo 'show-command-lockscreen=false'
+       echo 'command-hibernate=xset dpms force off'
+       echo 'show-command-hibernate=true'
+       echo 'command-restart=xfce4-session-logout --reboot --fast'
+       echo 'show-command-restart=true'
+       echo 'command-shutdown=xfce4-session-logout --halt --fast'
+       echo 'show-command-shutdown=true'
+} >> /etc/xdg/xfce4/whiskermenu/defaults.rc
```

Xfce4-session-logout config

eg

```
+# xfce4-session-logout remove suspend, because system/kernel does no support
suspend;
+sed -i '/<\/channel>/d' /etc/xdg/xfce4/xfconf/xfce-perchannel-xml/xfce4-
session.xml
+{
+       echo '  <property name="shutdown" type="empty"> '
+       echo '    <property name="ShowSuspend" type="bool" value="false"/>'
+       echo '  </property>'
+       echo '</channel>'
+} >> /etc/xdg/xfce4/xfconf/xfce-perchannel-xml/xfce4-session.xml
```

Thunar config

eg

```
/etc/xdg/xfce4/xfconf/xfce-perchannel-xml/thunar.xml

# Enable misc-single-click, avoid touch cann't open directories/files;
<?xml version="1.0" encoding="UTF-8"?>

<channel name="thunar" version="1.0">
  <property name="misc-single-click" type="bool" value="true"/>
</channel>
```

中英文切换

配置文件：`/etc/default/locale`

- 中文配置

  ```
  LANG=C.UTF-8
  LANG=zh_CN.UTF-8
  ```

- 英文配置

  ```
  LANG=C.UTF-8
  # LANG=zh_CN.UTF-8
  ```

中英文固件编译方法

中英文默认用同一套代码；

服务器代码默认是中文，所以只有英文需要手动修改打包；

英文固件，不需要全部重新编译，在上面中文的基础上，修改这个文件`/etc/default/locale`，重新打包rootfs，生成update.img;

- 代码默认全部编译完；
- 屏蔽中文：修改`binary/etc/default/locale`，改为如下：

  ```
  LANG=C.UTF-8
  # LANG=zh_CN.UTF-8
  ```

- 屏蔽编译文件系统rootfs，修改如下：

  ```
  zjm@jwast-rd3:~/wk/cx3568-n/debian/cx3568_linux$ git diff
  device/rockchip/common/scripts/mk-rootfs.sh
  diff --git a/device/rockchip/common/scripts/mk-rootfs.sh
  b/device/rockchip/common/scripts/mk-rootfs.sh
  index fa50db03e..9b35b81ad 100755
  ```

```
--- a/device/rockchip/common/scripts/mk-rootfs.sh
+++ b/device/rockchip/common/scripts/mk-rootfs.sh
@@ -111,7 +111,7 @@ build_debian()
                         linaro-$RK_DEBIAN_VERSION-$ARCH.tar.gz
         fi

-       VERSION=debug ARCH=$ARCH ./mk-rootfs-$RK_DEBIAN_VERSION.sh
+       # VERSION=debug ARCH=$ARCH ./mk-rootfs-$RK_DEBIAN_VERSION.sh
         ./mk-image.sh

         ln -rsf "$PWD/linaro-rootfs.img" $ROOTFS_DIR/rootfs.ext4
zjm@jwast-rd3:
```

- 重新打包rootfs, 编译执行如下：

```
./build.sh debian
```

- 生成update.img包，固件包默认是英文版；

```
./build.sh updateimg
```

- 完成；

中文输入法

通过`Ctrl+Space`切换输入法；

谷歌输入法

默认使用谷歌输入法；

具体安装，查看编译脚本；

```
# input method with googlepinyin;
\${APT_INSTALL} fcitx fcitx-bin fcitx-config-common fcitx-config-gtk fcitx-data
\
fcitx-frontend-all fcitx-frontend-fbterm fcitx-frontend-gtk2 fcitx-frontend-gtk3
\
fcitx-frontend-qt5:arm64  fcitx-keyboard fcitx-libs fcitx-libs-dev fcitx-m17n \
fcitx-module-dbus fcitx-module-kimpanel fcitx-module-lua fcitx-module-x11 \
fcitx-modules  fcitx-table fcitx-table-all fcitx-tools fcitx-ui-classic \
fcitx-ui-light fcitx-ui-qimpanel fcitx-googlepinyin:arm64

\${APT_INSTALL} im-config zenity
# autostart fcitx;
cp /usr/share/applications/fcitx.desktop /etc/xdg/autostart/

# im-config -n fcitx;
sed -i "s/IM_CONFIG_DEFAULT_MODE=auto/IM_CONFIG_DEFAULT_MODE=fcitx/g"
/etc/default/im-config
```

搜狗输入法

可以安装上，可以使用，稳定行没有验证；

具体安装，查看搜狗官网教程；

Fcitx5

没有验证，网上建议使用这个；

系统备份/还原

暂时没有比较好的方法，暂时可以使用`timeshift`去备份/还原；

```
apt install -y timeshift
```

顶部Panel大小设置

WhiskerMenu->设置->面板(Panel)

字体Font大小设置

WhiskerMenu->设置->外观(Appearance)

去掉Panel-2 (底部一行去掉, 旧的固件)

在顶部panel空白处，右键->select `Panel` -> `Panel Preferences...` -> 这里有`Panel 1`和`Panel 2`选择，下拉菜单选择`Panel 2`, 然后点击旁边的减号`-`; -> 成功去掉底部Panel-2;

去掉plank dock (底部一行)

`rm /etc/xdg/autostart/plank.desktop`

Plank dock

鼠标移到dock位置，`Ctrl+鼠标右键`进入dock配置；

底部launchers配置

直接删除，或者添加即可；

```
/home/linaro/.config/plank/dock1/launchers
```

plank themes

`/usr/share/plank/`

multi plank dock配置

Xfce4 themes

`/usr/share/themes/`

设置主题后，不能还原主题

暂时没有找到好的方法；

规避方法：把添加的主题，全部删除掉，重启系统，即可以还原主题；

`rm -rf /usr/share/themes/Mc* /usr/share/themes/PRO* /usr/share/themes/Prof* /usr/share/themes/Grey* /usr/share/themes/Numix /usr/share/themes/White*`

Icons

`/usr/share/icons/, ~/.local/share/.icons/ or ~/.icons/`

桌面背景图 desktop image

`/usr/share/images/desktop-base/`

Docker

底层默认支持docker，应用默认安装docker.io,
使用docker功能，需要修改iptables version:

```
update-alternatives --set iptables /usr/sbin/iptables-legacy
update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
```

其它参考：

```
  124 commit 911fea1e28d6a57a8d6a408cccb80fc6c4e86d4b
```

```
125 Author: jamin <zhongjieming@sztouchtec.cn>
126 Date:   Fri Nov 24 15:21:23 2023 +0800
127
128     [Kernel] Enable docker configurations;
129 ....
130     1. Docker.io:
131     1.1 When use docker, configure the iptables version(legacy) manually:
132     // iptables-legacy
133     update-alternatives --set iptables /usr/sbin/iptables-legacy
134     update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
135 ....
136     1.2 Debian default version: iptables-nft
137     // iptables-nft
138     update-alternatives --set iptables /usr/sbin/iptables-nft
139     update-alternatives --set ip6tables /usr/sbin/ip6tables-nft
140 ....
141     2. how to test:
142     systemctl daemon-reload
143     systemctl restart docker.service
144     docker run hello-world
145 ....
146     modified:
device/rockchip/.chips/rk3588/rockchip_rk3588_touchthink_lp4_v10_defconfig
147
```

Docker-ce

```
# Unstall docker.io
apt-get purge -y docker.io

# Intall docker-ce
curl -fsSL get.docker.com | sh
```

====分割线: Xubuntu20.04===

Xubuntu20.04 系统配置

以下是Xubuntu20.04 相关部分；

用户-应用配置目录

- 路径：`/home/linaro/.config/xfce4`
    - 平时一些应用的设置，保存在这个路径；

- 删除掉这些配置，类似会默认还原为出厂设置；

默认应用的配置目录

- 路径：`/etc/xdg/xdg-xubuntu` 和 `/etc/xdg/xfce4`

系统备份/还原

暂时没有比较好的方法，可以使用`timeshift`去备份/还原；

```
apt install -y timeshift
```

中文输入法(谷歌中文输入法)

通过`Ctrl+Space`切换输入法；

```

# Install google input;
apt install -y fcitx-ui-light fcitx-ui-qimpanel fcitx-googlepinyin:arm64 fcitx
fcitx-table fcitx-config-gtk

# set default xinput for fcitx
sed -i 's/default/fcitx/g' /etc/X11/xinit/xinputrc

==================================

# Install Chinese fonts
apt install -y ttf-wqy-zenhei xfonts-intl-chinese
apt install -y language-pack-zh-hans language-pack-gnome-zh-hans

# show chinese
sed -i 's/^# *\(zh_CN.UTF-8\)/\1/' /etc/locale.gen
echo "LANG=zh_CN.UTF-8" >> /etc/default/locale
locale-gen zh_CN.UTF-8
```

中英文固件版本编译方法

中英文固件编译方法

中英文默认用同一套代码；

服务器代码默认是中文，所以只有英文需要手动修改打包；

英文固件，不需要全部重新编译，在上面中文的基础上，修改这个文件`/etc/default/locale`，重新打包rootfs，生成update.img;

- 代码默认全部编译完；
- 屏蔽中文：修改`binary/etc/default/locale`，改为如下：

```
LANG=C.UTF-8
# LANG=zh_CN.UTF-8
```

- 屏蔽编译文件系统rootfs，修改如下：

```
zjm@jwast-rd3:~/wk/cx3588-n/xubuntu20/cx3588_linux$ git diff
device/rockchip/common/scripts/mk-rootfs.sh
diff --git a/device/rockchip/common/scripts/mk-rootfs.sh
b/device/rockchip/common/scripts/mk-rootfs.sh
index 5cf8d4539..c83a7846e 100755
--- a/device/rockchip/common/scripts/mk-rootfs.sh
+++ b/device/rockchip/common/scripts/mk-rootfs.sh
@@ -149,7 +149,7 @@ build_xubuntu20()
        fi

        # VERSION=debug ARCH=$ARCH ./mk-rootfs-$RK_XUBUNTU20_VERSION.sh
-       RELEASE=$RK_XUBUNTU20_VERSION VERSION=debug ARCH=$ARCH ./mk-rootfs-
ubuntu.sh
+       # RELEASE=$RK_XUBUNTU20_VERSION VERSION=debug ARCH=$ARCH ./mk-
rootfs-ubuntu.sh
        ./mk-image.sh

        ln -rsf "$PWD/linaro-rootfs.img" $ROOTFS_DIR/rootfs.ext4
zjm@jwast-rd3:~/wk/cx3588-n/xubuntu20/cx3588_linux$
```

- 重新打包rootfs, 编译执行如下：

```
./build.sh xubuntu20
```

- 生成update.img包，固件包默认是英文版；

```
./build.sh updateimg
```

- 完成；

桌面配置单点触摸active Icons

```
sed -i '/property name="tooltip-size"/ a\    <property name="single-click"
type="bool" value="true"/>' /etc/xdg/xdg-xubuntu/xfce4/xfconf/xfce-perchannel-
xml/xfce4-desktop.xml
```

Thunar 文件管理器配置单点active Icons

```
sed -i '/shortcuts-icon-size/ a\  <property name="misc-single-click" type="bool"
value="true"/>' /etc/xdg/xdg-xubuntu/xfce4/xfconf/xfce-perchannel-xml/thunar.xml
```

Panel Item调整

```
/etc/xdg/xdg-xubuntu/xfce4/panel/default.xml
```

Docker

Install docker

```
sudo apt-get install docker.io
```

How to test

```
systemctl daemon-reload
systemctl restart docker.service
systemctl enable docker.service

docker run hello-world
```

硬件加速

有两种实现：

- 使用RK的deb包；
- 使用开源；

====分割线: Common===

以下部分，是通用部分；

Other configs

开机自启动

- 将应用软件加入启动器和桌面快捷方式
  - 会话与启动(Session and Startup)->Application Autostart;
- 可以通过`systemctl`方式启动服务；
- 可以把脚本放到`/etc/init.d`目录；
- 可以配置为应用桌面，放到`/etc/xdg/autostart`或者`/etc/xdg/xdg-xubuntu/autostart`；

开机启动自己应用，去掉桌面

例如：开机自动显示google浏览器，不显示桌面；

- 将谷歌浏览器加入到启动器中；

```
会话与启动(Session and Startup)->Application Autostart;

Name:           google chromium
Description:    todo
Command:        /usr/bin/chromium
Trigger:        on login
```

- 不显示桌面；

```
mv /usr/bin/xfdesktop /usr/bin/xfdesktop-bak
```

- 重启，设置完成；

性能相关

CPU GPU DDR NPU的频率电压表

```
cat /d/opp/opp_summary
```

cpu

cpu load

- `htop`
- cpu 当前频率：`cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_cur_freq`
- cpu 支持的频率：`cat /sys/devices/system/cpu/*/cpufreq/scaling_available_frequencies`

CPU定频

RK3588 CPU GPU DDR NPU定频和性能模式设置

RK3588的cpu是4个A55+4个A76，分为3组单独管理，节点分别是：

```
/sys/devices/system/cpu/cpufreq/policy0:（对应4个A55：CPU0-3）
affected_cpus     cpuinfo_max_freq  cpuinfo_transition_latency
scaling_available_frequencies  scaling_cur_freq  scaling_governor
scaling_min_freq  stats
cpuinfo_cur_freq  cpuinfo_min_freq  related_cpus
scaling_available_governors    scaling_driver    scaling_max_freq
scaling_setspeed

/sys/devices/system/cpu/cpufreq/policy4:(对应2个A76：CPU4-5)
```

```
affected_cpus     cpuinfo_max_freq  cpuinfo_transition_latency
scaling_available_frequencies  scaling_cur_freq  scaling_governor
scaling_min_freq  stats
cpuinfo_cur_freq  cpuinfo_min_freq  related_cpus
scaling_available_governors    scaling_driver    scaling_max_freq
scaling_setspeed

/sys/devices/system/cpu/cpufreq/policy6:(对应2个A76：CPU6-7)
affected_cpus     cpuinfo_max_freq  cpuinfo_transition_latency
scaling_available_frequencies  scaling_cur_freq  scaling_governor
scaling_min_freq  stats
cpuinfo_cur_freq  cpuinfo_min_freq  related_cpus
scaling_available_governors    scaling_driver    scaling_max_freq
scaling_setspeed
rk3588_s:/ #

以上3个CPU是独立控制，下面以设置CPU6-7为例说明如何设置CPU6-7的频率
获取当前CPU支持的频点
rk3588_s:/ # cat
/sys/devices/system/cpu/cpufreq/policy6/scaling_available_frequencies

408000 600000 816000 1008000 1200000 1416000 1608000 1800000 2016000 2208000
2400000

获取cpu运行的模式
rk3588_s:/ # cat
/sys/devices/system/cpu/cpufreq/policy6/scaling_available_governors
conservative ondemand userspace powersave performance schedutil

默认是自动变频模式：schedutil（恢复的话设置为该模式即可）

设置手动定频模式：userspace
echo userspace > /sys/devices/system/cpu/cpufreq/policy6/scaling_governor

设置频率为2016000
echo 2016000 > /sys/devices/system/cpu/cpufreq/policy6/scaling_setspeed

确认是否设置成功
cat /sys/devices/system/cpu/cpufreq/policy6/cpuinfo_cur_freq
```

CPU性能模式

```
echo performance > /sys/devices/system/cpu/cpufreq/policy6/scaling_governor
```

gpu

gpu load

```
cat /sys/devices/platform/fb000000.gpu/utilisation
```

GPU定频

```
ls /sys/class/devfreq/fb000000.gpu/

获取GPU支持的频点
cat /sys/class/devfreq/fb000000.gpu/available_frequencies

获取GPU运行的模式
cat /sys/class/devfreq/fb000000.gpu/available_governors

默认是自动变频模式：simple_ondemand（恢复的话设置为该模式即可）
设置手动定频模式：userspace
echo userspace > /sys/class/devfreq/fb000000.gpu/governor

设置频率为1000000000
echo 1000000000 > /sys/class/devfreq/fb000000.gpu/userspace/set_freq

确认是否设置成功
cat /sys/class/devfreq/fb000000.gpu/cur_freq
```

GPU性能模式

```
echo performance > sys/class/devfreq/fb000000.gpu/governor
```

vpu load

```
fuser /dev/mpp_service
```

npu

npu load

```
cat /sys/kernel/debug/rknpu/load
```

NPU定频

```
NPU的节点路径
ls /sys/class/devfreq/fdab0000.npu/

获取NPU支持的频点
cat /sys/class/devfreq/fdab0000.npu/available_frequencies

获取NPU运行的模式
cat /sys/class/devfreq/fdab0000.npu/available_governors
```

```
默认是自动变频模式：simple_ondemand（恢复的话设置为该模式即可）

设置手动定频模式：userspace
echo userspace > /sys/class/devfreq/fdab0000.npu/governor

设置频率为1000000000
echo 1000000000 > /sys/class/devfreq/fdab0000.npu/userspace/set_freq

确认是否设置成功
cat /sys/class/devfreq/fdab0000.npu/cur_freq
```

NPU性能模式

```
 echo performance > /sys/class/devfreq/fdab0000.npu/governor
```

rga load

```
NPU的节点路径
cat /sys/kernel/debug/rkrga/load
```

ddr

ddr load

```
cat /sys/class/devfreq/dmc/load
or
htop
```

DDR定频

```
DDR的节点路径
ls /sys/class/devfreq/dmc/

获取DDR支持的频点
cat /sys/class/devfreq/dmc/available_frequencies

获取DDR运行的模式
cat /sys/class/devfreq/dmc/available_governors

默认是自动变频模式：dmc_ondemand（恢复的话设置为该模式即可）
设置手动定频模式：userspace
echo userspace > /sys/class/devfreq/dmc/governor

设置频率为2112000000
echo 2112000000 > /sys/class/devfreq/dmc/userspace/set_freq

确认是否设置成功
```

```
cat /sys/class/devfreq/dmc/cur_freq
```

DDR性能模式

```
echo performance > /sys/class/devfreq/dmc/governor
```

emmc load

```
df -h
```

cpu温度

```
cat /sys/class/thermal/thermal_zone0/temp
```

压力测试

ddr

emmc

ethernet

audio

video

camera

wifi / bt

登陆界面欢迎信息(motd)

修改路径:

```
/etc/update-motd.d/*
```

Led Ctl

drivers path: `kernel/drivers/jawest/jwsled.c`

设备节点: `/dev/jawest_led`

sys 控制led

turn on red led:

```
echo 1 > /sys/class/jawest_led/jawest_ledr
```

turn off red led:

```
echo 0 > /sys/class/jawest_led/jawest_ledr
```

turn on blue led:

```
echo 1 > /sys/class/jawest_led/jawest_ledb
```

turn off blue led:

```
echo 0 > /sys/class/jawest_led/jawest_ledb
```

read blue/red led status:

```
cat /sys/class/jawest_led/LEDB
cat /sys/class/jawest_led/LEDR
```

ioctl 控制led

```
通过ioctl调用,具体参考
`jws3399_linux/buildroot/package/rockchip/jws_test/jwsled_test`;
设备节点: /dev/jawest_led

#define LED_B_STAT_ON  (1)
#define LED_B_STAT_OFF (0)
#define LED_R_STAT_ON  (0)
#define LED_R_STAT_OFF (1)
#define LED_R_ON  (0x141)
#define LED_R_OFF  (0x142)
#define LED_B_ON  (0x143)
#define LED_B_OFF  (0x144)

eg:
fd = open("/dev/jawest_led", O_RDWR);
ioctl(fd, LED_R_ON);    //  turn on red led;
ioctl(fd, LED_R_OFF);   //  turn off red led;
```

LCD 屏

- 支持lvds/mipi/edp/hdmi显示；
- 烧录屏参，只支持屏参放在根目录；

lvds

屏参，参考屏参相关文档；

mipi

屏参，参考屏参相关文档；

edp

屏参，参考屏参相关文档；

hdmi

屏参，参考屏参相关文档；

backlight

调节范围： 0--255；

默认显示亮度(uboot, kernel阶段)，可以通过屏参修改，默认为：default-brightness-level = 128;

pwm默认输出1kHz.

具体查看屏参描述：

```
\\172.0.0.249\share\LCD配置\RK3399-Linux\readme.md    (这部分暂时参考3399)

pwm-pol
pwm-bl
pwm-min
pwm-max
pwm-freq
default-brightness-level
```

how to test

```
// 这里仅是举例，实际一个屏，只会显示一个背光节点；
root@rk3568:/# ls /sys/class/backlight/
edp-backlight  lvds-backlight  mipi-backlight


LVDS:
/sys/class/backlight/lvds-backlight/brightness

cat /sys/class/backlight/lvds-backlight/brightness
echo 128 > /sys/class/backlight/lvds-backlight/brightness


MIPI:
/sys/class/backlight/mipi-backlight/brightness

cat /sys/class/backlight/mipi-backlight/brightness
echo 128 > /sys/class/backlight/mipi-backlight/brightness

EDP:
/sys/class/backlight/edp-backlight/brightness

cat /sys/class/backlight/edp-backlight/brightness
echo 128 > /sys/class/backlight/edp-backlight/brightness
```

休眠

Note:

1. cx3588-linux休眠功能，只做假休眠功能，即仅仅关闭背光，不会走真休眠流程；
2. 假休眠功能：只针对lvds/edp/mipi屏，hdmi默认不做假休眠；

   如果客户hdmi也需要做休眠功能，可以通过强制开关hdmi去实现；

```
lvds/edp/mipi 背光-假休眠控制：


1. lvds/edp/mipi控制节点：
/sys/class/backlight/*/sleep

/sys/class/backlight/lvds-backlight/sleep
/sys/class/backlight/edp-backlight/sleep
/sys/class/backlight/mipi-backlight/sleep


echo 1/0 > /sys/class/backlight/*/sleep
cat /sys/class/backlight/*/sleep
1：  打开休眠功能；
0：  关闭休眠功能；

示例：
// 打开休眠功能：
echo 1 > /sys/class/backlight/edp-backlight/sleep
// 关闭休眠功能：
```

```
echo 0 > /sys/class/backlight/edp-backlight/sleep
// 查看状态
cat /sys/class/backlight/edp-backlight/sleep

--------------------
2. HDMI开关控制节点:
/sys/class/drm/card0-HDMI-A-1/status

// 关闭HDMI
echo off > /sys/class/drm/card0-HDMI-A-1/status

// 打开HDMI
echo on > /sys/class/drm/card0-HDMI-A-1/status

// 查看HDMI状态
cat /sys/class/drm/card0-HDMI-A-1/status
```

uart

硬件对应系统的节点

详细参考硬件规格书;

```
UART0_TX    GPIO4_A3_d / UART0_TX_M2
UART0_RX    GPIO4_A4_d / UART0_RX_M2
    J5 TTL:   UART0_TX -> TX0 -> J5         （J13 J16 接 1，2脚）
              UART0_RX -> RX0 -> J5
    J9 232:   UART0_TX -> UART0-TX-1 -> (SP3232EEA-L) PC_TXD0 -> J9    (J13 J16
接 2,3脚）
              UART0_RX -> UART0-RX-1 -> (SP3232EEA-L) PC_RXD0 -> J9

UART2_TX_M0_DEBUG    GPIO0_B5_d / UART2_TX_M0        UART2_TX_DEBUG_PORT    J3
UART2_RX_M0_DEBUG    GPIO0_B6_d / UART2_RX_M0        UART2_RX_DEBUG_PORT

UART3_RX    GPIO1_C0_z / UART3_RX_M0 ->   UART3_RX_01(To 3.3V)    (SP3232EEA-
L)PC_RXD3    -> J10 232
UART3_TX    GPIO1_C1_z / UART3_TX_M0 ->    UART3_TX_01(To 3.3V)    (SP3232EEA-
L)PC_TXD3

UART4_TX    GPIO1_D2_d / UART4_TX_M0 ->    UART4_TX_01(To 3.3V)    -> J19 TTL
UART4_RX    GPIO1_D3_d / UART4_RX_M0 ->    UART4_RX_01(To 3.3V)    -> J19 TTL

UART6_RX    GPIO1_A0_d / UART6_RX_M1 ->    UART6_RX_01(To 3.3V)
(MAX3485)RS485_A    -> J2  RS485
UART6_TX    GPIO1_A1_d / UART6_TX_M1 ->    UART6_TX_01(To 3.3V)
(MAX3485)RS485_B

UART7_TX    GPIO3_C0_d / UART7_TX_M1
UART7_RX    GPIO3_C1_d / UART7_RX_M1
    J4 TTL:    UART7_TX -> TX7 -> J4    (J12 J15 接 2,1脚）
```

```
                UART7_RX -> RX7 -> J4
    J8 232:     UART7_TX -> UART7-TX-1 -> (SP3232EEA-L) PC_TXD7 -> J8    (J12 J15
接 2,3脚）
                UART7_RX -> UART7-RX-1 -> (SP3232EEA-L) PC_RXD7 -> J8

UART9_TX_M0_BT    GPIO2_C2_d / UART9_TX_M0
UART9_RX_M0_BT    GPIO2_C4_d / UART9_RX_M0
UART9_RTSn_M0_BT    GPIO4_C4_d / UART9_RTSN_M0
UART9_CTSn_M0_BT    GPIO4_C5_d / UART9_CTSN_M0
```

串口登陆添加密码配置(参考)

- 在设备上修改;

```
用户名：  root
密码：    rockchip

vi /etc/inittab

::respawn:-/bin/sh # ttyS0::respawn:/sbin/getty -L ttyS0 115200 vt100 # GENERI
改为
:respawn:/bin/login
```

- 修改登陆密码和登陆欢迎语句;
  - 在buildroot中配置;
  - `cd buildroot`, then `make menuconfig`;
  - Enter `System configuration` --> `Root password`;

gpio ctl

两种版型：cx3588-G and cx3588-GS;

对于底层，有一点差异;

但对于上层，没有差异;

sys 控制gpio

- 设置gpio输入或者输出;
- 读取gpio电平，或者设置gpio高低电平;

```
root@linaro-alip:/# ls /sys/external_gpio/
jwsioc_gpio0  jwsioc_gpio3  jwsioc_inout_gpio0  jwsioc_inout_gpio3
jwsioc_gpio1  jwsioc_gpio4  jwsioc_inout_gpio1  jwsioc_inout_gpio4
jwsioc_gpio2  jwsioc_gpio5  jwsioc_inout_gpio2  jwsioc_inout_gpio5
root@linaro-alip:/#
```

```
// 配置gpio输出
echo 1 >  /sys/external_gpio/jwsioc_inout_gpio0
echo 1 >  /sys/external_gpio/jwsioc_inout_gpio1
echo 1 >  /sys/external_gpio/jwsioc_inout_gpio2
echo 1 >  /sys/external_gpio/jwsioc_inout_gpio3
echo 1 >  /sys/external_gpio/jwsioc_inout_gpio4
echo 1 >  /sys/external_gpio/jwsioc_inout_gpio5
....

// gpio 设置高电平
echo 1 >  /sys/external_gpio/jwsioc_gpio0
echo 1 >  /sys/external_gpio/jwsioc_gpio1
echo 1 >  /sys/external_gpio/jwsioc_gpio2
echo 1 >  /sys/external_gpio/jwsioc_gpio3
echo 1 >  /sys/external_gpio/jwsioc_gpio4
echo 1 >  /sys/external_gpio/jwsioc_gpio5
....

// gpio 设置低电平
echo 0 >  /sys/external_gpio/jwsioc_gpio0
echo 0 >  /sys/external_gpio/jwsioc_gpio1
echo 0 >  /sys/external_gpio/jwsioc_gpio2
echo 0 >  /sys/external_gpio/jwsioc_gpio3
echo 0 >  /sys/external_gpio/jwsioc_gpio4
echo 0 >  /sys/external_gpio/jwsioc_gpio5
....

// 配置gpio输入
echo 0 >  /sys/external_gpio/jwsioc_inout_gpio0
echo 0 >  /sys/external_gpio/jwsioc_inout_gpio1
echo 0 >  /sys/external_gpio/jwsioc_inout_gpio2
echo 0 >  /sys/external_gpio/jwsioc_inout_gpio3
echo 0 >  /sys/external_gpio/jwsioc_inout_gpio4
echo 0 >  /sys/external_gpio/jwsioc_inout_gpio5
....

// 查看gpio电平
cat /sys/external_gpio/jwsioc_gpio0
cat /sys/external_gpio/jwsioc_gpio1
cat /sys/external_gpio/jwsioc_gpio2
cat /sys/external_gpio/jwsioc_gpio3
cat /sys/external_gpio/jwsioc_gpio4
cat /sys/external_gpio/jwsioc_gpio5
....
```

硬件端口

```
J22  (rk3588-G)

PWRON_01
RESET_L
GPIO4_A6_1    GPIO4_A6_d
GPIO2_A7_1    GPIO2_A7_u        (1.8v to 3.3v)
GPIO4_A5_1    GPIO4_A5_d
GPIO2_B1_1    GPIO2_B1_u        (1.8v to 3.3v)
GPIO4_B4_1    GPIO4_B4_u
GPIO2_B0_1    GPIO2_B0_u        (1.8v to 3.3v)
GND
SARADC_IN7
```

eeprom at24c32

the path of at24c32 driver: `kernel/drivers/misc/eeprom/at24.c`

the path of eeprom test demo: `buildroot/package/rockchip/jws-test/eeprom_test`

Note:

At24c32 最大可以存储4096个字节。

how to test

```
/sys/devices/platform/fec80000.i2c/i2c-6/6-0050/eeprom
/sys/bus/i2c/devices/6-0050/eeprom

cat /sys/bus/i2c/devices/6-0050/eeprom
echo "this is eeprom test" > /sys/bus/i2c/devices/6-0050/eeprom
```

RTC

the path of drivers: `kernel/drivers/rtc/rtc-at8563.c`

the path of test demo: `buildroot/package/rockchip/jws-test/rtc_test`

IOCTL inteface: `/dev/rtc0`

其它参考demo：

`jws3399_android9.0\frameworks\opt\jws\jni\jws\com_android_jws.cpp`

`jws3399_linux/external/rtc_demo/rtc_demo.c`

Note: 一定 需要接上电池，rtc功能才能正常使用(这个待确认-20230421)。

how to test

```
cat /sys/class/rtc/rtc0/date
cat /sys/class/rtc/rtc0/time

date -s "2023-06-16 19:27:00"
date

hwclock：硬件时钟
-w：同步系统时间到硬件时间
-s：同步硬件时间到系统时间
hwclock -f /dev/rtc0 -w --utc
hwclock -f /dev/rtc0 -s --utc


定时开机：
 #120秒后定时开机
echo +120 >  /sys/class/rtc/rtc0/wakealarm
# 查看开机时间
cat /sys/class/rtc/rtc0/wakealarm
#关机
poweroff

Note:
    1. 开机的时候，系统自动会同步rtc的时间，即rtc时间同步到系统时间；
    2. 设置定时开机后，在定时时间到达前，一定要关机；


==================
# 时区设置
timedatectl


========================================
* 这里rtc硬件时间的年可以设置的范围是2000--2099；
* date系统设置时间的年的范围是：1970--2037，
  具体原因，请网上查询资料；

实际硬件可以设置的范围是(规格书的描述)：
* 具有世纪标志，可工作于 1900-2099 年（年份与世纪相关，主要用于计算闰年时二月的天数）



========================================
读寄存器：
read all registers, rd: echo rd > /sys/bsp_rtc/bsp_rtc
write register, wr: echo wr [reg] [value] > /sys/bsp_rtc/bsp_rtc
set time, set_time: echo set_time [year] [month] [week] [day] [hour] [min] [sec]
> /sys/bsp_rtc/bsp_rtc
read time, read_time: echo read_time > /sys/bsp_rtc/bsp_rtc
help: echo help > /sys/bsp_rtc/bsp_rtc
```

watchdog

the path of drivers: `kernel/drivers/watchdog/dw_wdt.c`

the path of test demo: `buildroot/package/rockchip/jawest/jwswdt_test/jwswdt_test.c`

`kernel-4.4/Documentation/watchdog/src/watchdog-test.c`

`kernel-4.4/Documentation/watchdog/src/watchdog-simple.c`

`external/rk_pcba_test/echo_pcbatest_server.c` -- (DEV_WDT_NAME 关闭wdt)

`jws3288_android10.0/frameworks/opt/jws/jni/jws/com_android_jws.cpp`

funcs

- 默认看门狗是关闭的;
- 超时时间，可以修改;
- 看门狗可以打开，关闭;

hwo to test

应用操作 `/dev/watchdog` 节点来控制watchdog，示例如下:

```
// /dev/watchdog

int main(void)
{
    int fd = open("/dev/watchdog", O_WRONLY);  // 通过open来启动watchdog
    int ret = 0;
    if (fd == -1) {
        perror("watchdog");
        exit(EXIT_FAILURE);
    }
    while (1) {
        ret = write(fd, "\0", 1); // 通过write来喂狗
        if (ret != 1) {
            ret = -1;
            break;
        }
        sleep(10);
    }
    close(fd);
    return ret;
}
```

关于 `close()`

- 正常情况下 close()，不再喂狗，watchdog会自动重启。

```
# 这里写入的是除大写V以外的任意字符：开启看门狗；
echo A > /dev/watchdog
```

- `write(fd, "V", 1)`；再 `close()`，写入大写`V`，内核继续喂狗，系统不会自动重启。

```
# 内核自动喂狗；
echo V > /dev/watchdog
```

暂停看门狗

暂停计数：

```
io -4 0xfd58c000 0x00010001
或者
busybox devmem 0xfd58c000 32 0x00010001
```

恢复计数：

```
io -4 0xfd58c000 0x00100000
或者
busybox devmem 0xfd58c000 32 0x00100000
```

g-sensor

the path of driver: `kernel/drivers/input/sensors/accel/bma2xx.c`

the path of test demo: `buildroot/package/rockchip/jawest/getevent`

how to test

```
Note: 功能没有测试；注意板子上是否贴了此芯片；

查看设备的名称
cat /proc/bus/input/devices

这里假设设备节点是/dev/input/event3

/opt/bin/getevent /dev/input/event3
或者
hexdump /dev/input/event3
```

红外遥控

pwrkey

- 短按休眠/唤醒功能，长按关机；

  关机后，短按开机；

touch

```
硬件位号： J14 (i2c触摸)；
```

ilitek--奕力

usb / i2c 支持；

goodix--汇顶

usb / i2c 支持；

eeti--禾瑞亚

usb 支持；

USB

硬件接口

查看硬件规格书；

otg/host 切换

```
方式1. [Legacy]
#1.Force host mode
echo host > /sys/devices/platform/fd5d0000.syscon/fd5d0000.syscon:usb2-
phy@0/otg_mode
#2.Force peripheral mode
echo peripheral > /sys/devices/platform/fd5d0000.syscon/fd5d0000.syscon:usb2-
phy@0/otg_mode
/usr/bin/usbdevice update
```

Audio / Sound

Overview

- cx3568-Linux, 有三个声卡;

```
root@linaro-alip:/# cat /proc/asound/cards
 0 [rockchiphdmiin ]: rockchip_hdmiin - rockchip,hdmiin
                      rockchip,hdmiin
 1 [rockchipes8388 ]: rockchip-es8388 - rockchip-es8388
                      rockchip-es8388
 2 [rockchiphdmi0  ]: rockchip-hdmi0 - rockchip-hdmi0
                      rockchip-hdmi0
root@linaro-alip:/#
```

aplay

喇叭/耳机 播放

```
aplay -Dhw:1,0 -f S16_LE -r 44100 -c 2 -t wav /aaa.wav
```

HDMI 播放

```
aplay -Dhw:2,0 -f S16_LE -r 44100 -c 2 -t wav /aaa.wav
```

arecord

```
# 录音
arecord -Dhw:1,0 -c 2 -r 44100 -f S16_LE /data/record.wav

# 播放录音的文件
aplay -Dhw:1,0 -f S16_LE -r 44100 -c 2 -t wav /data/record.wav
```

amixer

Ethernet

Copy from rk3399-linux, and there is not test.

ip/dns

```
配置IP
ifconfig eth0 172.0.6.183 netmask 255.255.248.0 up
route add default gw 172.0.0.1
(如果直接ping百度(www.baidu.com)，ping不通，可能DNS没有配置；这时可以直接ping 百度的IP)
------------------
或者自动分配IP:
udhcpc -i eth0
```

```
配置DNS:
vi /etc/resolv.conf
nameserver 8.8.8.8
nameserver 114.114.114.114
```

静态IP

```
1. 手动配置:
配置IP
ifconfig eth0 172.0.6.183 netmask 255.255.248.0 up
route add default gw 172.0.0.1
(如果直接ping百度(www.baidu.com)，ping不通，可能DNS没有配置；这时可以直接ping 百度的IP)

配置DNS:
vi /etc/resolv.conf
nameserver 8.8.8.8
nameserver 114.114.114.114


2. 在文件中配置:

在/etc/dhcpcd.conf文件末尾添加:

interface eth0
static ip_address=172.0.6.183/24
static routers=172.0.0.1
static domain_name_servers=202.96.128.86 114.114.114.114 120.196.165.24
```

动态IP

```
udhcpc -i eth0
```

ethtool

千兆

```
ethtool eth0

ethtool -s eth0 speed 1000 duplex full autoneg off
ethtool -s eth0 speed 1000 duplex full autoneg on

----------------
百兆，半双工
ethtool -s eth0 speed 100 duplex half autoneg on
```

配置以太网通用参数

打开或关闭自协商

```
ethtool -s port_name autoneg { on | off }
```

修改双工模式

```
ethtool -s port_name duplex { half | full }
```

注意：

- 当以太网接口工作在自协商模式时，缺省情况下双工模式是和对端接口协商得到的。
- 当以太网接口工作在非自协商模式时，缺省情况下双工模式为全双工模式。

修改速率

```
ethtool -s port_name speed { 10 | 100 | 1000 }
```

注意：

- 当以太网接口工作在自协商模式时，缺省情况下接口速率是和对端接口协商得到的。

- 当以太网接口工作在非自协商模式时，缺省情况下接口速率为接口支持的最大接口速率。

iperf

以下配置参数，可能不是最佳配置，需要查资料进一步确认。

丢包测试，最好两块板子直连，或者电脑跟板子直连。

缓存配置：

```
sysctl -w net.core.wmem_max=10485760
sysctl -w net.core.rmem_max=10485760
sysctl -w net.core.wmem_default=10485760
sysctl -w net.core.rmem_default=10485760
```

丢包测试：

```
iperf -u -s -i 1 -w 2M -p 30000
iperf -u -c 192.168.1.102 -i 1 -w 2M -p 30000 -t 60 -l 1300 -b 1000M
```

使用ip和netplan配置IP地址和路由

iptables NAT配置

iptables filter配置

以太网远程唤醒开机

Wireless

cx3588 wifi有三个型号，它们之间需要做到兼容；

- Wifi5/bt: Rtl8821cu;
- Wift6/bt: AP6275P;
- DS2.1

源码路径

```
// AP6275P driver;
cx3588_linux/external/rkwifibt/drivers/bcmdhd/

// // AP6275P firmware;
cx3588_linux/external/rkwifibt/firmware/broadcom/AP6275_PCIE/

// Rtl8821cu wifi driver;
cx3588_linux/external/rkwifibt/drivers/rtl8821cu/

// Rtl8821cu bluetooth driver;
cx3588_linux/external/rkwifibt/drivers/bluetooth_usb_driver/

// Rtl8821cu Firmware;
cx3588_linux/external/rkwifibt/firmware/realtek/RTL8821CU/

// Wifi AP demo;
linux5.10/realse/cx3588_linux/external/softapDemo/

// DS2.1
```

kernel 配置

```
# Support AP that the kernel requires the following configurations:
  1652     +CONFIG_NETFILTER=y
  1653     +CONFIG_NF_CONNTRACK=y
  1654     +CONFIG_NF_TABLES=y
  1655     +CONFIG_NF_TABLES_INET=y
  1656     +CONFIG_NF_CONNTRACK_IPV4=y
  1657     +CONFIG_IP_NF_IPTABLES=y
  1658     +CONFIG_IP_NF_NAT=y
  1659     +CONFIG_IP_NF_TARGET_MASQUERADE=y
  1660     +CONFIG_BRIDGE=y
```

STA

```
命令行连接示例：

wpa_supplicant -Dnl80211 -iwlan0 -c/etc/wpa_supplicant.conf  -B

这里假设有一个wifi热点：
SSID:   "360-2.4G"
passwd: 88888888

命令行连接wifi：
wpa_cli -i wlan0

scan
scan_results
add_network
```

```
set_net 1 ssid "360-2.4G"
set_network 1 key_mgmt WPA-PSK
set_network 1 psk "88888888"
select_network 1
enable_network 1
status

断开连接：
disconnect

重新连接：
reconnect

----------
连接成功后，可以使用udhcpc工具分配IP：
udhcpc -i wlan0

===========
关于更多wpa_cli命令，请查看官网。
status  查看连接状态
list_network
disable_network [network_id]：禁用网络。


=========
连接第二个wifi：
add_network
set_net 2 ssid "TP-LINK_5997"
set_network 2 key_mgmt WPA-PSK
set_network 2 psk "12345678"
select_network 2
enable_network 2
status
```

AP

AP6275P 和rtl8821cu 模组，使用的节点不一样：

AP6275P 使用wlan1，rtl8821cu 使用p2p0, 它们之间需要做兼容；

正基模组：AP6275P

AP: 不需要密码连接

连接脚本

```
#!/bin/sh
```

```
echo 0 > /proc/sys/net/ipv4/ip_forward
ifconfig wlan0 down
ifconfig wlan1 down
killall dnsmasq
killall hostapd

sleep .5

ifconfig wlan0 up
sleep .5
iw phy0 interface add wlan1 type managed

ifconfig wlan1 up
ifconfig wlan1 192.168.100.1 netmask 255.255.255.0 up
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE

sleep .5
hostapd /userdata/cfg/hostapd.conf -B

sleep .5
dnsmasq -C /userdata/bin/dnsmasq.conf --interface=wlan1
```

/userdata/cfg/hostapd.conf

```
interface=wlan1
ctrl_interface=/var/run/hostapd
driver=nl80211
ssid=zjm
channel=6
hw_mode=g
ieee80211n=1
ignore_broadcast_ssid=0
```

/userdata/bin/dnsmasq.conf

```
user=root
listen-address=192.168.100.1
dhcp-range=192.168.100.50,192.168.100.150
server=/google/8.8.8.8
```

AP: 需要密码连接

连接脚本

```
#!/bin/sh

echo 0 > /proc/sys/net/ipv4/ip_forward
ifconfig wlan0 down
ifconfig wlan1 down
killall dnsmasq
killall hostapd

sleep .5

ifconfig wlan0 up
sleep .5
iw phy0 interface add wlan1 type managed

ifconfig wlan1 up
ifconfig wlan1 192.168.100.1 netmask 255.255.255.0 up
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE

sleep .5
hostapd /userdata/cfg/hostapd.conf -B

sleep .5
dnsmasq -C /userdata/bin/dnsmasq.conf --interface=wlan1
```

/userdata/cfg/hostapd.conf

```
interface=wlan1
driver=nl80211
ssid=zjm
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GI-20]
ignore_broadcast_ssid=0
auth_algs=1
wpa=3
wpa_passphrase=123456789
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

/userdata/bin/dnsmasq.conf

```
user=root
listen-address=192.168.100.1
dhcp-range=192.168.100.50,192.168.100.150
server=/google/8.8.8.8
```

AP: 默认配置

/userdata/cfg/hostapd.conf

这部分settings创建;

```
interface=wlan0
driver=nl80211
ssid=HOTSPOT_TEST
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GI-20]
ignore_broadcast_ssid=0
auth_algs=1
wpa=3
wpa_passphrase=987654321
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

/userdata/bin/dnsmasq.conf

```
# 原生代码配置;

user=root
listen-address=10.201.126.1
dhcp-range=10.201.126.50,10.201.126.150
server=/google/8.8.8.8


====================

# Settings 创建;

user=root
listen-address=192.168.100.1
dhcp-range=192.168.100.50,192.168.100.150
server=/google/8.8.8.8
```

RTL wifi：RTL8821cu

AP: 不需要密码连接

连接脚本

```
#!/bin/sh

echo 0 > /proc/sys/net/ipv4/ip_forward
ifconfig wlan0 down
ifconfig p2p0 down
killall dnsmasq
killall hostapd

sleep .5

# iw phy0 interface add p2p0 type managed

ifconfig p2p0 up
ifconfig p2p0 192.168.100.1 netmask 255.255.255.0 up
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE

sleep .5
hostapd /userdata/cfg/hostapd.conf -B

sleep .5
dnsmasq -C /userdata/bin/dnsmasq.conf --interface=p2p0
```

/userdata/cfg/hostapd.conf

```
interface=p2p0
ctrl_interface=/var/run/hostapd
driver=nl80211
ssid=zjm
channel=6
hw_mode=g
ieee80211n=1
ignore_broadcast_ssid=0
```

/userdata/bin/dnsmasq.conf

```
user=root
listen-address=192.168.100.1
dhcp-range=192.168.100.50,192.168.100.150
server=/google/8.8.8.8
```

AP: 需要密码连接

连接脚本

```
#!/bin/sh

echo 0 > /proc/sys/net/ipv4/ip_forward
ifconfig wlan0 down
ifconfig p2p0 down
killall dnsmasq
killall hostapd

sleep .5

# iw phy0 interface add p2p0 type managed

ifconfig p2p0 up
ifconfig p2p0 192.168.100.1 netmask 255.255.255.0 up
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE

sleep .5
hostapd /userdata/cfg/hostapd.conf -B

sleep .5
dnsmasq -C /userdata/bin/dnsmasq.conf --interface=p2p0
```

/userdata/cfg/hostapd.conf

```
interface=p2p0
driver=nl80211
ssid=zjm
channel=6
hw_mode=g
ieee80211n=1
ht_capab=[SHORT-GI-20]
ignore_broadcast_ssid=0
```

```
auth_algs=1
wpa=3
wpa_passphrase=123456789
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

/userdata/bin/dnsmasq.conf

```
user=root
listen-address=192.168.100.1
dhcp-range=192.168.100.50,192.168.100.150
server=/google/8.8.8.8
```

Bluetooth

使用bluez协议；

FAE配置

```
hciattach -s 1500000 /dev/ttyS1 any 1500000 flow nosleep
hciconfig hci0 up
hcitool scan


// 祝工的配置：
1.  echo 1 > /sys/class/rfkill/rfkill0/state  上电
2.  hcitool dev

3.  hciconfig hci0 up
bluealsa --profile=a2dp-source &    拉起服务

4.  bluetoothctl                                命令行进入蓝牙
5.  power on
6.  agent on                                    代理

7.  default-agent                   默认代理

8.  scan on                       扫描开启
hcitool scan                        扫描
9.  scan off                        扫描关闭
10. discoverable on
11. pairable on
12. unblock F4:63:1F:D6:82:DD
13. pair F4:63:1F:D6:82:DD          匹对蓝牙
14. trust F4:63:1F:D6:82:DD         添加信任
15. connect F4:63:1F:D6:82:DD       连接
```

Modem

支持4G和5G： EC20, EC200N-CN, RM500U；

how to test

```
# 使用quectel-CM工具去连接网络， 支持移动、电信、联通；
quectel-CM &
```

硬件上下电, 复位操作

```
open,write:
/dev/jawestctrl

具体操作，参考jws3399-android9.0，没有详细测试。


write: 需要传下面两个参数下来：
pwrctrls <value>    // type:upper 8bit ,value: lower 8bit
switch (type) {
    case 9:// 3/4 G power
        jawest_4g_power(state);
        ctrl_data->bb_power_state = state;
        break;
    case 10:  // 3/4 G reset
        jawest_4g_reset(state);
        ctrl_data->bb_reset_state = state;
        break;
}

read:
284 static ssize_t jawest_read(struct file *filp, char __user *buf, size_t
count, loff_t *f_pos)
  285 {
  286     int  len = 0;
  287     char tmp[512] = {0};
  288
  289     len = sprintf(tmp,
  290                 "wlanpwr    : %d\n"

  291                 "wlanrst    : %d\n"
  292                 "ledstate   : %d\n"
  293                 "spkstate   : %d\n"
  294                 "bbpwr      : %d\n"

  295                 "bbrst      : %d\n"
```

```
296                 /* "checkeeti    : %d\n" */
297 //              "modem      : %d\n"

298                 ,
299                 ctrl_data->wlan_power_state,
300                 ctrl_data->wlan_reset_state,

301                 ctrl_data->led_control_state,

302                 ctrl_data->speaker_power_state,
303                 ctrl_data->bb_power_state,

304                 ctrl_data->bb_reset_state

305 //               jawest_get_modem_mold()
306             ) + 1;
307
308     if (len < count && *f_pos == 0) {
309         if (copy_to_user(buf, tmp, len)) {

310             return -EFAULT;
311         }
312         else {
313             *f_pos += len;

314             return len;
315         }
316     }
317
318     return 0;
319 }
```

CAN

硬件端口： J24

两块板子收发

- 两块板子，用线接上(AABB解法)；
- 接收(receive)板子的命令配置：

```
1. 查询当前网络设备：
ifconfig -a

2. CAN启动：
// 关闭CAN：
ip link set can0 down
// 设置比特率500KHz：
ip link set can0 type can bitrate 500000
```

```
// 打印can0信息:
ip -details -statistics link show can0
// 启动CAN:
ip link set can0 up

3. CAN接收:
// 开启打印，等待接收:
candump can0
```

- 发送(Send)板子的命令配置:

```
1. 查询当前网络设备:
ifconfig -a

2. CAN启动:
// 关闭CAN:
ip link set can0 down
// 设置比特率500KHz:
ip link set can0 type can bitrate 500000
// 打印can0信息:
ip -details -statistics link show can0
// 启动CAN:
ip link set can0 up

3. CAN发送:
// 发送（标准帧,数据帧,ID:123,date:DEADBEEF）:
cansend can0 123#DEADBEEF
```

一块板子, 自发自收

- 不用接线;
- 命令配置，需要打开两个窗口:
  - 初始化配置:

```
1.
// 查询当前网络设备:
ifconfig -a

2. CAN启动:
//关闭CAN:
ip link set can0 down
//设置比特率500KHz:
ip link set can0 type can bitrate 500000
//打印can0信息:
ip -details -statistics link show can0
//启动CAN:
ip link set can0 up
```

```
// (rk3588 can2)启动can后，io输入命令开启回环自测（基地址根据实际dts启动的can配置）：
io -4 0xfea70000  0x8415
```

- 一个窗口，做CAN接收

```
// 开启打印，等待接收:
candump can0
```

- 一个窗口，做CAN发送

```
// 发送（标准帧,数据帧,ID:123,date:DEADBEEF）:
cansend can0 123#DEADBEEF
```

refer

参考 《docs\cn\Common\CAN\Rockchip_Developer_Guide_Can_CN.pdf》

或者:

`\\172.0.0.249\share\LCD配置\CX3568-Linux\000-CX3568-Linux-相关资料\0003-RK文档\CAN`

rknpu

refer

参考《cx3568_linux\docs\cn\Common\NPU》

或者:

`\\172.0.0.249\share\LCD配置\CX3568-Linux\000-CX3568-Linux-相关资料\0003-RK文档\NPU\`

HDMI-IN

Qt

针对Debian11 and Xubuntu20.04;

- Debian11/Xubuntu20.04共用一套Qt配置，兼容rk3568;
- 支持Qt5和Qt6: Qt5.12.8， Qt5.15.2, Qt6.2.4;

其它详细，请参考：`\\172.0.0.249\share\个人资料\jamin3\cx3588-linux\modules\debian11\qt`

动态替换静态logo

系统支持动态替换开机静态logo，把开机静态logo放到这个目录即可：`/oem/custom_logo/`；

- 如果没有`custom_logo`目录，手动创建: `mkdir -p /oem/custom_logo`；
- 静态logo，必须提供两张：一张是在uboot中显示，一张在kernel中显示；

  （开机 logo 一般分为两个阶段：显示 U-Boot logo 和 显示 Kernel logo. ）
  - logo的名称是固定的，分别为：`logo.bmp logo_kernel.bmp`；
  - 图片要求：
    - 只支持 8bit，16bit，24bit、32bit 的 bmp 图片；
    - U-Boot logo(`logo.bmp`) 和 Linux Kernel logo(`logo_kernel.bmp`) 分辨率相同，而且分辨率必须是偶数；

其它详细，请参考：`\\172.0.0.249\share\个人资料\jamin3\cx3588-linux\modules\linux-repack`；

导出文件系统

支持文件系统导出：支持导出静态logo，屏参和文件系统，同时可以二次打成整包；

请参考：`\\172.0.0.249\share\个人资料\jamin3\cx3588-linux\modules\linux-repack`；