title

**加油**

1. android bsp cmd
2. Shell cmd
3. software
   1. Makefile编译
   2. lk以及cmdline传参
   3. kernel
      1. lcd
      2. led
      3. gpio
      4. pinctrl
      5. pm\_runtime
      6. i2c
      7. usb
   4. init 进程 节点权限
   5. HAL
   6. JNI
   7. framework service
   8. Device
   9. out
4. DEBUG
5. ISSUE

* android bsp cmd

环境安装

--------------------------------------------------------------------------------------------------------------------------------

android bsp编译

1. source mitac\_project\_env.sh
2. cd LINUX/android
3. source build/envsetup.sh
4. lunch msm8953\_64-userdebug
5. make –j8

make bootimage –j8

make dtboimage –j8

--------------------------------------------------------------------------------------------------------------------------------

命令回退提交

git reset --hard HEAD~2

--------------------------------------------------------------------------------------------------------------------------------

user image 烧录image

settings -> development option -> oem unlock.

fastboot oem unlock

fastboot flash boot boot.img

fastboot oem force-update强制烧录,但是可能有go.ABH问题

--------------------------------------------------------------------------------------------------------------------------------

BSP网址

<http://10.88.25.195/aiotbc/bsp/sc600_10/>

<http://10.88.26.87/aiotbc/bsp/sc600/>

<http://10.88.25.195/aiotbc/bsp/a9msm8909/>

--------------------------------------------------------------------------------------------------------------------------------

adb devices no permissions

error: insufficient permissions for device

add /etc/udev/rules.d/51-android.rules

SUBSYSTEM==”usb”, ENV{DEVTYPE}==”usb\_device”, MODE=”0666”

--------------------------------------------------------------------------------------------------------------------------------

/\*--------------提交---------------------------\*/

1. git add modify\_file

添加文件

git status

检查

2. git commit -m "message"

git commit --am 修改上次的提交历史

git commit --amend --no-edit 不编辑message

3. git push origin HEAD:refs/for/branch\_name

1，2可以使用git gui替代。

/\*-----------------------------------------\*/

* shell cmd

1.建立索引

ctags -R --exclude=out --exclude=folder2 --exclude=folder3

ctags -R --exclude={out, folder2, folder3}

2.$PATH 安装第三方软件添加环境变量

1. 一般安装在/opt或/user/local目录下
2. 将第三方软件命令添加到系统路径中
3. 将第三方软件软链接放到/usr/bin目录下
4. 将第三方软件路径添加到系统路径中

sudo vi ~/.bashrc

export PATH=$PATH:/opt/folder\_name/bin

source ~/.bashrc

echo $PATH

export 声明一个变量为环境变量，可以被其他脚本和当前程序调用。

类似于c中的extern.

3.shell 脚本

#!/bin/bash

1.shell中的function

function mm()

{

}

如何执行shell中的function

1.source \*.sh将sh中的function添加到环境变量中

mm执行sh中对应的function.

2. ./\*.sh 依靠内部逻辑调用

shell的参数

$1-$9,${10}，$0是file\_name

打开文件夹

nautilus .

1. SOFTWARE

--------------------------------------------------------------------------------------------------------------------------------

3.1 Makefile

:= = +=

立即赋值 mk结束后赋值 add结束后赋值

一次赋多值 用空格间隔开，常加\表示换行

makefile函数

foreach

$(foreach <temp\_val>,<list>,<text>)

eg：

names := a b c d

files := $(foreach n,$(names),$(n).o)

files的值”a.o b.o c.o d.o”

filter

$(filter pattern…,list)

保留符合pattern的list.

sources := foo.c bar.c baz.s ugh.h   
files:=  $(filter %.c %.s,$(sources))

files的值” foo.c bar.c baz.s”

eval

$(eval text)

是 text 的内容将作为makefile的一部分而被make解析和执行。

call

if and or

subst

dir notdir

basename

Makefile, Kconfig, .config, obj的编译关系

一般关系

Kconfig

depends on

A depends on B表依赖。CONFIG\_A 的前提是CONFIG\_B

select

A select C 表选择。CONFIG\_A就会CONFIG\_C.

--------------------------------------------------------------------------------------------------------------------------------

启动解析

cmdline、init进程

验证一个driver是否加载，以及加载时间。

1.查看out.../obj/KERNEL\_OBJ/drivers目录下相应的\*.o文件

2.查看对应的defconfig文件

以chiron\_pro\_a10为例

arch/arm64/configs/chiron\_pro\_a10-perf\_defconfig //for user image build config -b user

arch/arm64/configs/chiron\_pro\_a10\_defconfig //for user-debug image

arch/arm64/configs/msm8953-perf\_defconfig //no use

arch/arm64/configs/msm8953\_defconfig //no use

arch/arm/configs/msm8953-perf\_defconfig //no use

arch/arm/configs/msm8953\_defconfig //no use

选用对应的defconfig可以查看AndroidBoard.mk

device/mitac/chiron\_pro/AndroidBoard.mk

ifeq ($(TARGET\_BUILD\_VARIANT),user)

KERNEL\_DEFCONFIG := chiron\_pro\_a10-perf\_defconfig

else

KERNEL\_DEFCONFIG := chiron\_pro\_a10\_defconfig

defconfig没有合并覆盖的逻辑，只有一个文件决定driver的配置。

:=和=

=会在makefile文件展开后决定变量的值，即最后决定：a=1,b=$(a),a=2 ==> b=2;

:=使用当前的值。b=1;

3.查看System.map符号表

4.driver加载的顺序

1).与defconfig无关

2).与Makefile的顺序有关

3).与driver初始化段有关 module\_init call6.

* 1. kernel

[Linux内核专题 - 介绍\_beckdon的专栏-CSDN博客\_extcon](https://blog.csdn.net/beckdon/article/details/50424557)

* + 1. lcd

#### lcd

[msm8953平台LCD亮灭屏流程和LCD知识点总结\_移动开发\_jinron10的专栏-CSDN博客](https://blog.csdn.net/jinron10/article/details/103419337/?utm_medium=distribute.pc_relevant.none-task-blog-baidujs-2)

[高通MSM8916平台显示框架及lcd驱动概述\_lane](https://www.sohu.com/a/282532614_99956377)

[高通内核显示驱动MDSS解析(转) - Evsio0n](https://evsio0n.com/archives/286/)

[MIPI-DSI三种VideoMode理解\_Eliot\_shao的专栏-CSDN博客\_videoburstmode](https://blog.csdn.net/eliot_shao/article/details/52474348)

[驱动之LCD\_bigman\_123的专栏-CSDN博客\_](https://blog.csdn.net/bigman_123/article/details/19070579)

[android9.0上，实现双mipi屏\_xuhui\_7810的专栏-CSDN博客](https://blog.csdn.net/xuhui_7810/article/details/89213593)

[高通 android平台LCD驱动分析 - LoongEmbedded - 博客园](https://www.cnblogs.com/LoongEmbedded/p/5298269.html)

[高通平台 lcd driver 调试小结\_guoguo295的专栏-CSDN博客\_xdpi ydpi 高通](https://blog.csdn.net/guoguo295/article/details/38555551?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-9.add_param_isCf&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-9.add_param_isCf)

[MIPI\_DSI协议简要介绍\_caihaitao2000的博客-CSDN博客](https://blog.csdn.net/caihaitao2000/article/details/80224976)

[高通SDM450+android9.0读显示屏ID\_LoongEmbedded的专栏-CSDN博客](https://loongembedded.blog.csdn.net/article/details/114988994#comments_15518445)

#### display

[Android系统启动流程分析\_麦田-CSDN博客\_android启动流程 sbin](https://blog.csdn.net/galensphang/article/details/13631929?utm_medium=distribute.pc_relevant.none-task-blog-baidujs-6#t18)

[android LCM启动流程——LK\_迟子涵925的博客-CSDN博客](https://blog.csdn.net/u010787514/article/details/82632684#t0)

[我所理解的高通平台Lcd驱动框架\_Croxd的博客-CSDN博客](https://blog.csdn.net/weixin_42432281/article/details/113246959)

[高通平台lcm 总结\_yctcgogo的专栏-CSDN博客](https://blog.csdn.net/yctcgogo/article/details/82655903)

#### rotation

[Android 7.1 竖屏转横屏全过程实现-基于高通平台\_Eliot\_shao的专栏-CSDN博客](https://blog.csdn.net/eliot_shao/article/details/70766283?utm_medium=distribute.pc_relevant.none-task-blog-title-1&spm=1001.2101.3001.4242)

[Android9.0 MTK 平板横屏方案修改(强制app横屏 + 开机logo/动画+关机充电横屏 + RecoveryUI 横屏)\_cczheng-CSDN博客](https://blog.csdn.net/u012932409/article/details/102585974?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.add_param_isCf&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.add_param_isCf)

[【转】高通平台android9.0设置开机默认横屏显示 - 简书](https://www.jianshu.com/p/e44512de8c53)

[Android 7.1 竖屏转横屏全过程实现-基于高通平台\_Eliot\_shao的专栏-CSDN博客\_高通横屏竖用](https://blog.csdn.net/seek_0380/article/details/70766283?utm_source=blogxgwz8)

#### bootAnimation

[android系统知识（8.0）---Android O 开关机动画流程\_zhangbijun1230的专栏-CSDN博客](https://blog.csdn.net/zhangbijun1230/article/details/79735530)

[MSM8909开机logo显示(1)---LCD背光的控制\_移动开发\_涛声依旧的专栏-CSDN博客](https://blog.csdn.net/armfpga123/article/details/79548827)

[Android开机动画总结\_卓越之路-CSDN博客](https://blog.csdn.net/mcsbary/article/details/89192508)

[android开机动画启动流程 - JavaShuo](http://www.javashuo.com/article/p-ajhxjlrl-hy.html)

[Android bootanim开机动画启动流程 - JavaShuo](http://www.javashuo.com/article/p-zhbzspum-co.html)

[Android开机动画的显示(一)\_SwallowJoe的博客-CSDN博客](https://blog.csdn.net/u014535072/article/details/82916548#t26)

[Android系统的开机画面显示过程分析\_thinkinwm的专栏-CSDN博客](https://blog.csdn.net/thinkinwm/article/details/17963781)

#### backlight

[msm8953背光\_jinron10的专栏-CSDN博客\_msm8953的mpp1](https://blog.csdn.net/jinron10/article/details/104710753/?utm_medium=distribute.pc_relevant.none-task-blog-baidulandingword-6&spm=1001.2101.3001.4242)

[高通LCD的pwm背光驱动 - yooooooo - 博客园](https://www.cnblogs.com/linhaostudy/p/9173117.html#autoid-1-0-0)

[android p 解锁 忽然亮屏后 恢复亮度 - soul丶嘟 - 博客园](https://www.cnblogs.com/chjgongzuo/p/10728402.html)

[android 9.0关于屏幕亮度的整理\_yyqx1128\_wuting的博客-CSDN博客](https://blog.csdn.net/yyqx1128_wuting/article/details/107784760?utm_medium=distribute.pc_relevant_right.none-task-blog-BlogCommendFromMachineLearnPai2-6.nonecase&depth_1-utm_source=distribute.pc_relevant_right.none-task-blog-BlogCommendFromMachineLearnPai2-6.nonecase#t1)

#### brightness

[Android frameworks去掉熄屏前先变暗的功能\_Just do IT.-CSDN博客](https://blog.csdn.net/zhoumushui/article/details/51443492?utm_medium=distribute.pc_relevant_right.none-task-blog-BlogCommendFromBaidu-6.nonecase&depth_1-utm_source=distribute.pc_relevant_right.none-task-blog-BlogCommendFromBaidu-6.nonecase)

[Android 9.0 (P版本) 亮度进度条变化等级更新\_苏法迪的专栏-CSDN博客](https://blog.csdn.net/su749520/article/details/84585672)

[Android 8.1 DisplayPowerController(四) 自动调节亮度(1)——流程\_sdkdlwk的博客-CSDN博客](https://blog.csdn.net/sdkdlwk/article/details/88368247?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_title-2&spm=1001.2101.3001.4242)

[android 8.1自动背光随笔\_mafei19870124的专栏-CSDN博客](https://blog.csdn.net/mafei19870124/article/details/80435377?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&dist_request_id=6b0dd3ed-22ec-475c-9fdf-34bfd87a7435&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control)

[自动背光调节\_lei7143的专栏-CSDN博客](https://blog.csdn.net/lei7143/article/details/73302287?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&dist_request_id=1328593.23209.16148289958593913&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control)

* + 1. led

#### led

[Android8.1系统Led的控制从底层到上层的实现\_只是一些暗恋而已的博客-CSDN博客](https://blog.csdn.net/qq_30624591/article/details/88216726?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-4.add_param_isCf&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-4.add_param_isCf)

[Android 7.1 从底层到上层分析 Led 例子\_qq\_35003588的博客-CSDN博客](https://blog.csdn.net/qq_35003588/article/details/99681852?utm_medium=distribute.pc_relevant.none-task-blog-title-4&spm=1001.2101.3001.4242)

[我的内核学习笔记11：linux leds-gpio驱动应用实例\_李迟的专栏-CSDN博客](https://blog.csdn.net/subfate/article/details/53494926?utm_source=blogxgwz6)

[Android硬件抽象层（HAL）概要介绍和学习计划\_老罗的Android之旅-CSDN博客](https://blog.csdn.net/luoshengyang/article/details/6567257)

#### light

[Android Notification - 灯光提示\_chengchaooppo的专栏-CSDN博客](https://blog.csdn.net/chengchaooppo/article/details/105529783)

[2.5.8 Notification(状态栏通知)详解 | 菜鸟教程](https://www.runoob.com/w3cnote/android-tutorial-notification.html)

[傻瓜式Android APP开发入门教程 | 菜鸟教程](https://www.runoob.com/w3cnote/android-app-develop-learning.html)

* + 1. gpio

#### gpio

[高通平台中gpio简单操作和调试\_s\_jason的博客-CSDN博客\_android gpio调试](https://blog.csdn.net/s_jason/article/details/73864103)

[高通msm8909对gpio的使用（包括pinctrl的功能设置，gpio在节点中的引用，驱动程序获取gpio的方法）\_tasawen的博客-CSDN博客\_msm8909 gpio](https://blog.csdn.net/tasawen/article/details/89791344?utm_medium=distribute.pc_relevant_right.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant_right.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase)

[致驱动工程师的一封信](http://www.wowotech.net/device_model/429.html)

[Linux设备树语法详解 - Abnor - 博客园](https://www.cnblogs.com/xiaojiang1025/p/6131381.html)

[设备树之pinctrl编写 - ZYNQ/FPGA - 米联客uisrc](http://www.uisrc.com/forum.php?mod=viewthread&tid=2644)

<https://www.kernel.org/doc/Documentation/gpio/consumer.txt>

[Linux内核中的GPIO系统之（3）：pin controller driver代码分析](http://www.wowotech.net/gpio_subsystem/pin-controller-driver.html)

[(1条消息) Linux下的gpio,gpiod\_嵌入式Linux-CSDN博客](https://blog.csdn.net/weiqifa0/article/details/105085365/#t7)

* + 1. pinctrl
    2. pm\_runtime

<https://www.cnblogs.com/hellokitty2/p/9974273.html>

怎样动态地打开关闭设备的电源 ？ 最简单的方法：在驱动程序中，open时打开电源，在close时关闭电源。但是有一个缺点，当多个App使用该设备时可能造成干扰。  
解决方法：给驱动添加计数值，当该值大于0时打开电源，等于0时关闭电源。

runtime PM只是提供辅助函数，比如：  
(1).增加计数/减少计数  
(2).使能runtime pm

最好的资料是runtime\_pm.txt  TODO：翻译它

例子：\drivers\input\misc\bma150.c

pm\_runtime\_enable //bma150\_probe

pm\_runtime\_disable //bma150\_remove

pm\_runtime\_get\_sync //bma150\_open

pm\_runtime\_put\_sync //bma150\_close

* + 1. i2c

spi

https://blog.csdn.net/qq\_33790215/article/details/97614552#t3

* + 1. usb

usb 基础概念

usb state

usb spec 9.1.1

usb数据包解析

1. 编码方式

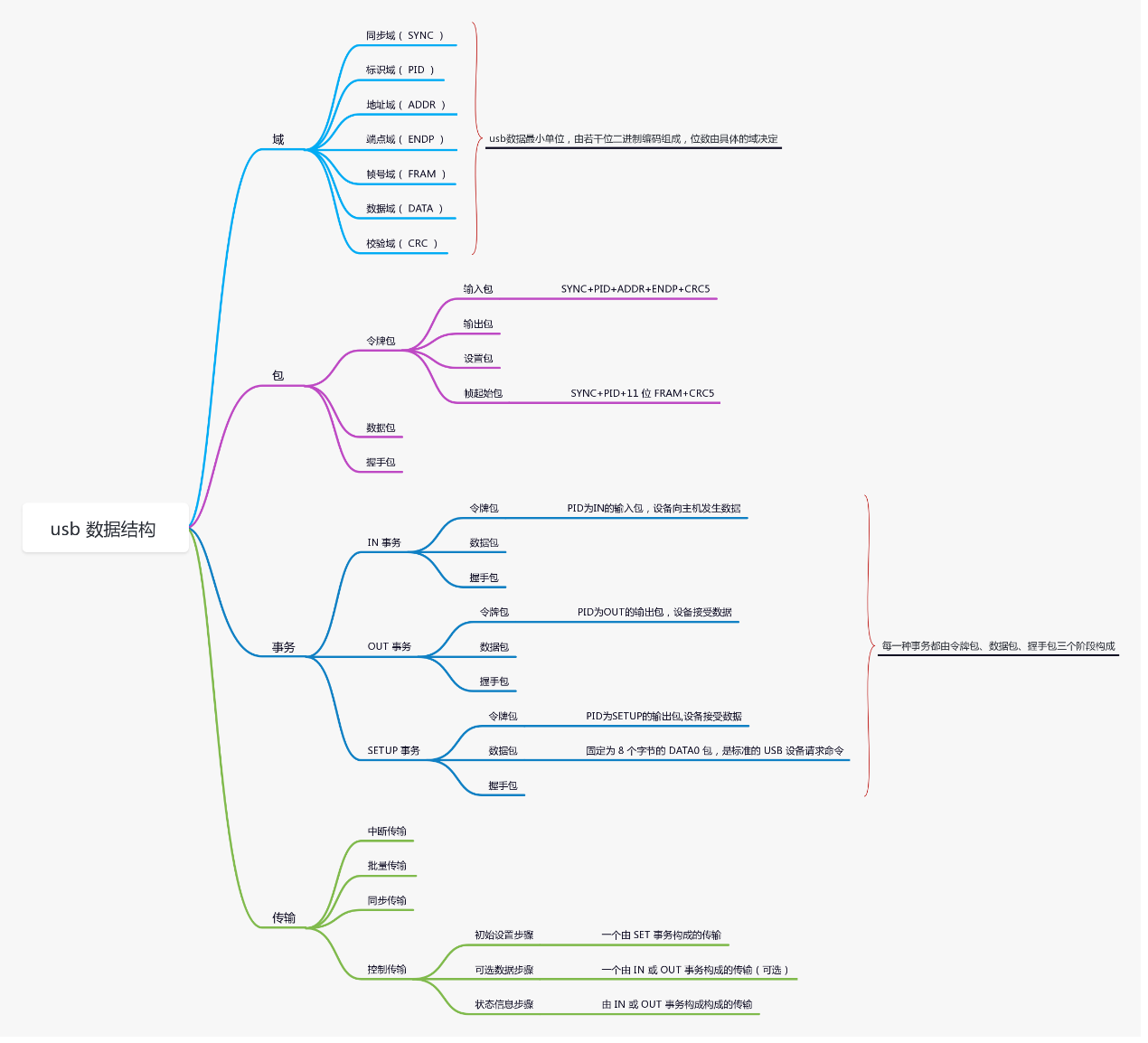
USB 采用不归零取反来传输数据，当传输线上的差分数据输入 0 时就取反，输入 1 时就保持原值。

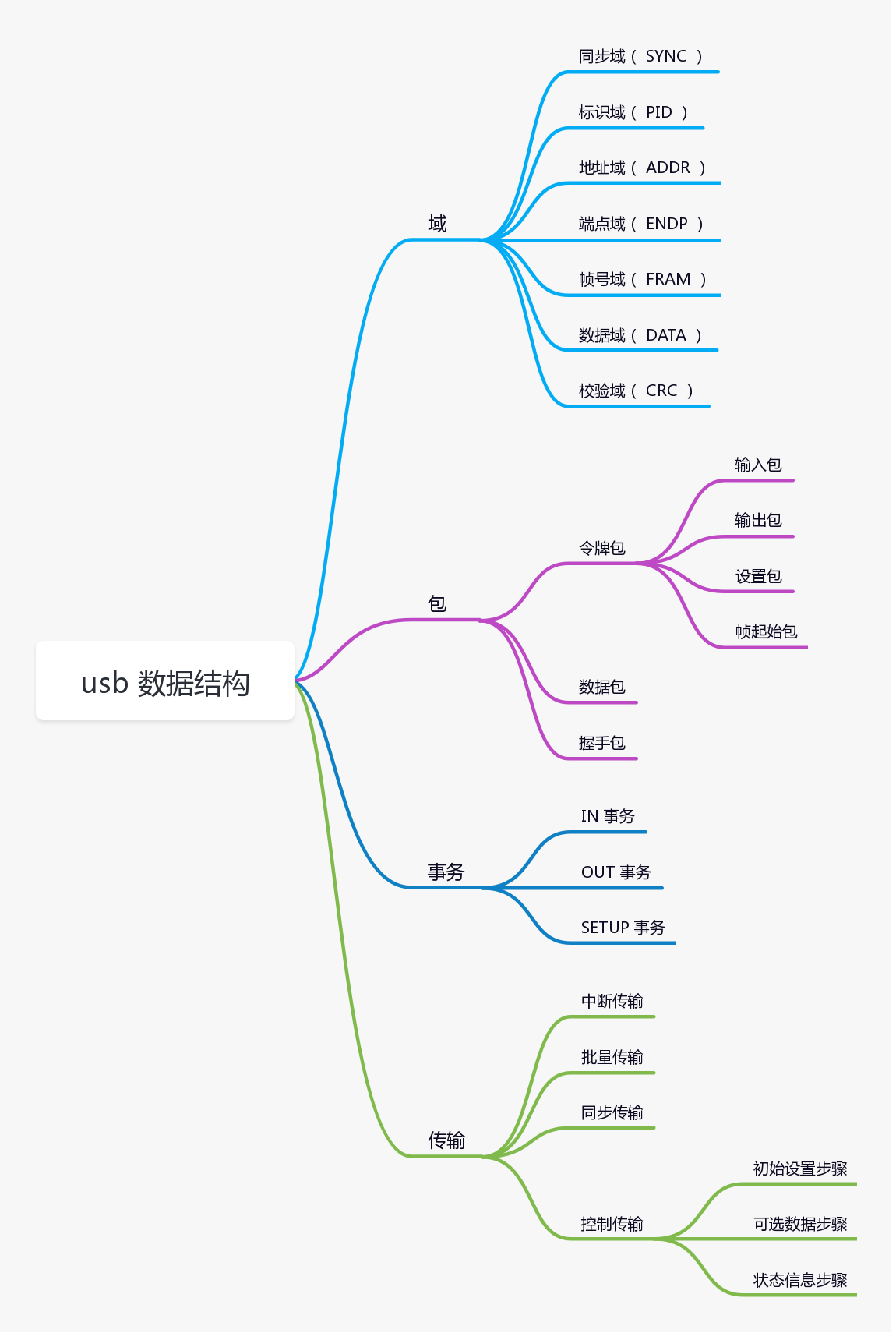
1. USB数据结构

USB 数据是由二进制数字串构成的，首先数字串构成域（有七种），域再构成包，包再构成事务（ IN 、 OUT 、 SETUP ），事务最后构成传输（中断传输、并行传输、批量传输和控制传输）。

<https://blog.csdn.net/yuanyou/article/details/5957609>

Chapter 8 Protocol Layer





USB枚举过程

* + 1. extcon driver

1. 什么是extcon driver

external connector class driver负责管理 外部连接器的检测。

通过注册相应的GPIO IRQ，去检测相应设备的拔插动作，并通知给相应的驱动。

<https://blog.csdn.net/qq_24622489/article/details/120436289?spm=1001.2014.3001.5501>

1. USB ID/VBUS IRQ是如何通知到对应的driver.

notifier head 是如何获取的.

1. 设备树，结构体，初始化，调用，以及例程
2. extcon-usb-gpio.c

extcon-usb-gpio.c是如何通知到usb otg driver?

extcon.c中注册notifier head，nh按照index区分不同的外部类型，如EXTCON\_NONE, EXTCON\_USB, EXTCON\_USB\_HOST

extcon-usb-gpio.c注册irq,irq触发notifier.

dwc3-msm.c 注册extcon\_register\_notifier.

* + 1. struct device/device tree
  1. node与property之间的关系
  2. node与函数之间的关系
  3. node转换成结构体的流程

kernel中相同的内容重复出现，

有整个文件替代，后续补充重复不覆盖，后续补充重复覆盖，

DT属于覆盖，但是哪些文件加载哪些文件不加载需要厘清

* + 1. 内核函数解析

notifier chain

1. 什么是notifier chain

一般来说，kernel module 是相互独立的，notifier chain用于kernel module 间通信.

具体查看kernel/notifier.c include/linux/notifier.h

1. 有几种类型

atomic blocking raw srcu

原子(spinlock) 阻塞(rw\_sem) 原始(锁provided by caller) 阻塞变式，

具体查看include/linux/notifier.h

blocking raw比较常用

1. 初始化，调用，以及实例

<https://www.cnblogs.com/3me-linux/p/6122444.html>

<https://blog.csdn.net/u014134180/article/details/86563754>

实例查看drivers/video/fbdev/core/fb\_notify.c

1. 自己的理解

register notifier,是将notifier handler注册在notify head的链表下.

当调用notify call chain(nh,val,void \*),会逐个调用nh链表下的handler.

handler根据val 和 void \*选择不执行或者执行相应的func(switch val).

关键是通知者和被通知者需要获取同一个notifier head.

list head

work queue

* 1. init进程

property

android property机制

https://blog.csdn.net/stoic163/article/details/88815430

2.3 HAL

hardware 功能检测

* 1. device

overlay机制

1.overlays机制

1)

PRODUCT\_PACKAGE\_OVERLAYS

DEVICES\_PACKAGE\_OVERLAYS

优先级PRODUCT > DEVICE

2)

OVERLAYS := file1 file2

优先级file1 > file2

<https://blog.csdn.net/the_Sunshine_of_King/article/details/56488563?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-1.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-1.control>

AAPT android资源打包机制

aidl

PRODUCT\_COPY\_FILE

解析过程build/core/Makefile

2.10 OUT

* DEBUG

--------------------------------------------------------------------------------------------------------------------------------

I2C

使用tool读取I2C

i2cget tool

adb root

adb push i2cget /data

adb shell

cd data

./i2cget -y -f 03 0x2c 0x20 w

bus addr reg

./i2cget –help

--------------------------------------------------------------------------------------------------------------------------------

GPIO

--------------------------------------------------------------------------------------------------------------------------------

CTS测试

1. 安装java环境
2. 安装Android Studio
3. 安装SDK

https://blog.csdn.net/u012764358/article/details/107049333/

https://blog.csdn.net/weixin\_39529302/article/details/114606435

https://blog.csdn.net/qq\_22948593/article/details/109957099

* ISSUE
  1. driver no work?

c25 base line vibrator not worl.

1. check kernel node function.
2. check hardware lib 🡪 fail.
3. check service
4. check settings
   1. property get fail?

persist.sys.display.default.rotation

Bootanimation.cpp property\_get

PhoneWindowManage.java SystemProperties.getInt

init.target.rc setprop 0

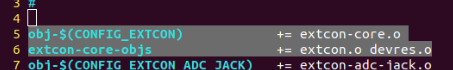
setprop 3

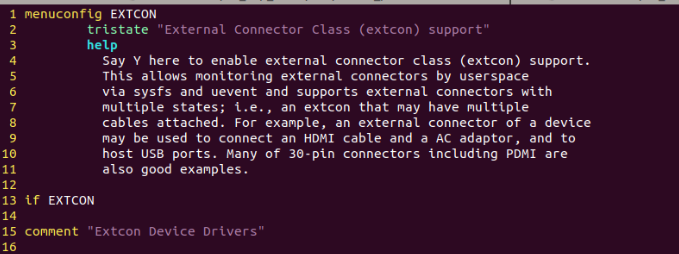
BootAnim get 0, JAVA get 3

* 1. 如何确认一个文件是否被编译?

以extcon driver为例.

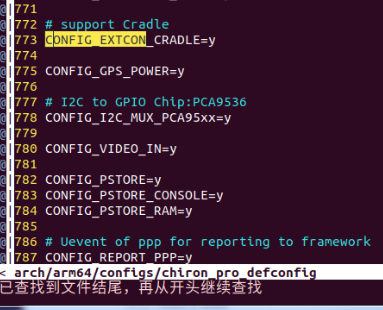
1. 查找Makefile以及Kconfig文件，确认编译文件对应的config项.





1. 确认编译对应的defconfig文件

arch/arm64/config/chiron\_pro\_defconfig



一般情况是可以找到CONFIG\_EXTCON=y,但是这里并没有

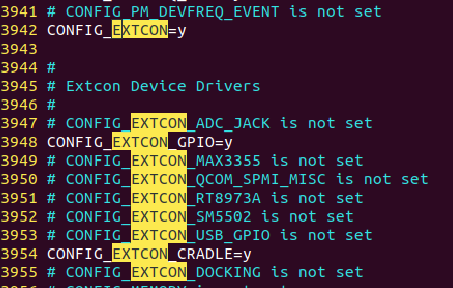
可以去确认其他driver Kconfig文件是否有select EXTCON.

grep -nR “select EXTCON” --include=Kconfig

1. 确认.config文件

obj/kernel/msm-4.9/.config

(obj/KERNEL\_OBJ link to obj/kernel/msm-4.9)



类似的文件

obj/kernel/msm-4.9/ include/config/auto.conf

obj/kernel/msm-4.9/ include/config/ tristate.conf

config转化成C语言可识别的宏:

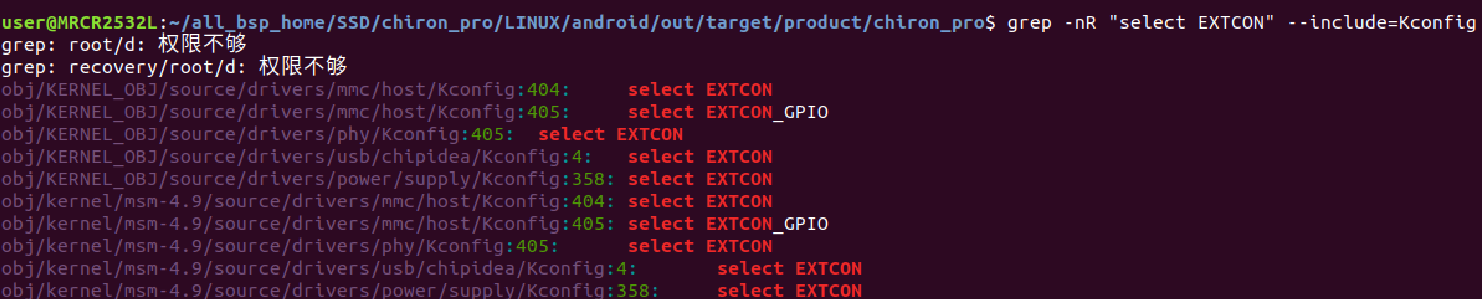
obj/kernel/msm-4.9/include/generated/autoconf.h

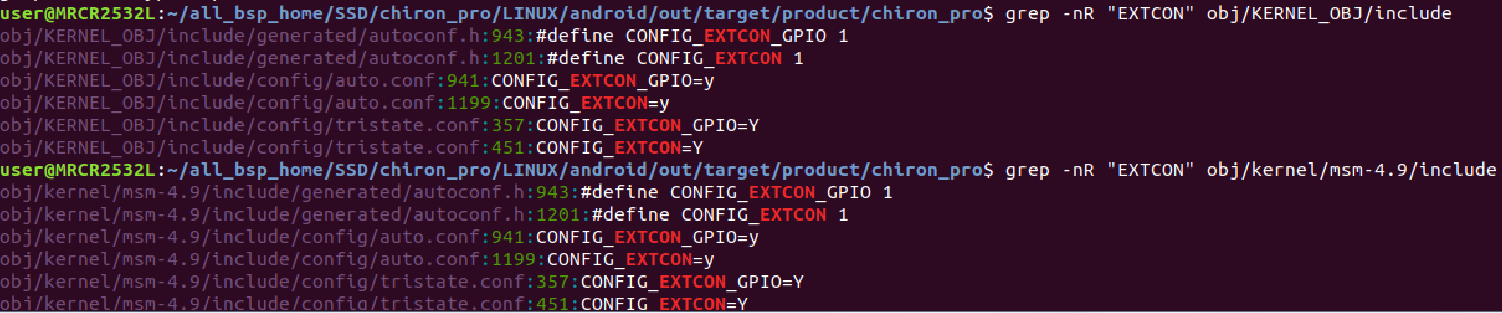
也可以对比旧的config文件看看是否有变化

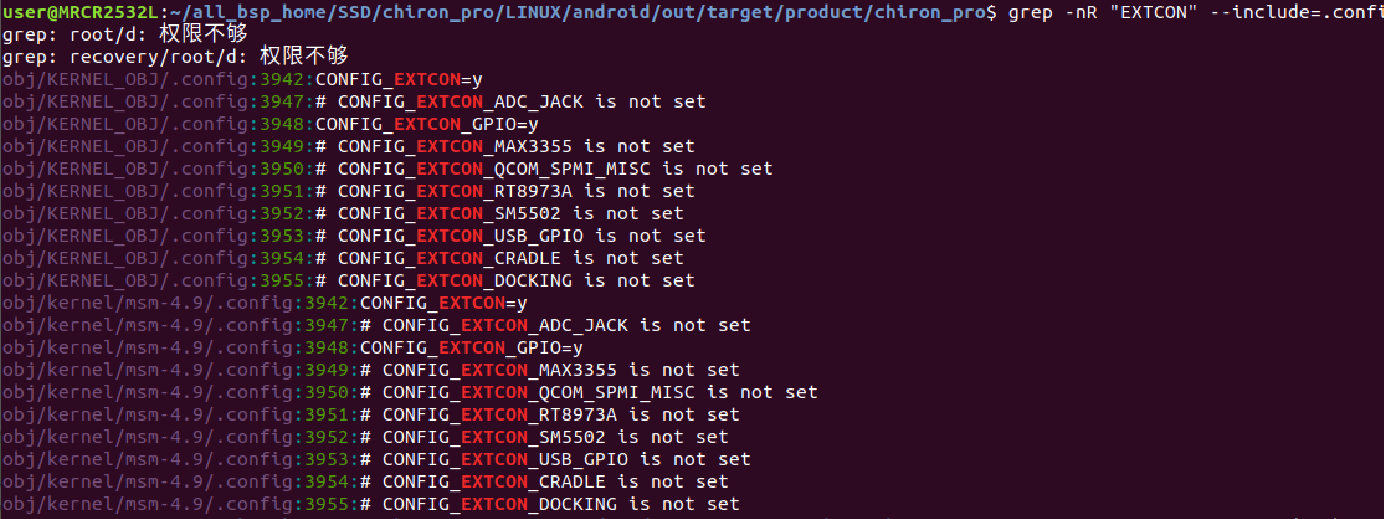
obj/kernel/msm-4.9/.config.old

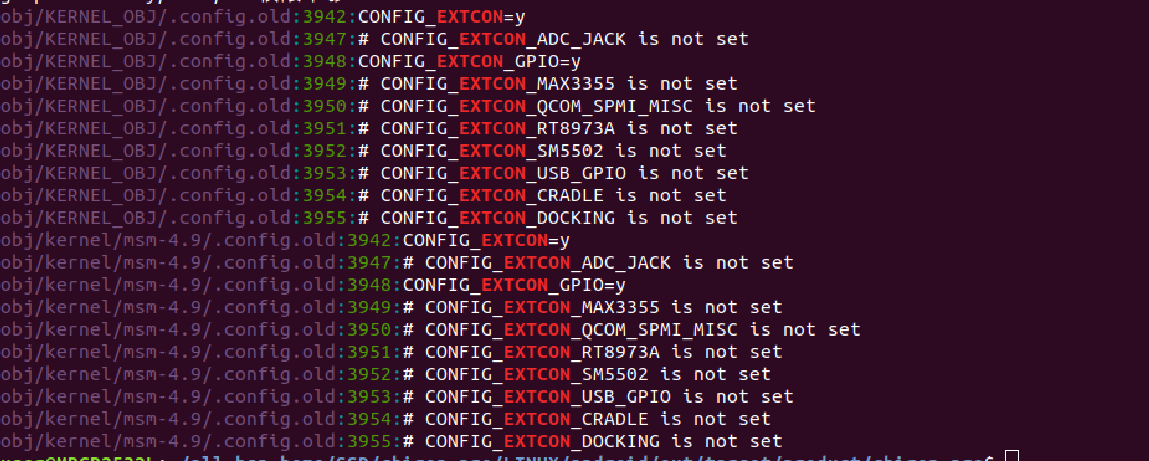
1. 确认编译生成相应的.o文件

ls obj/kernel/msm-4.9/driver/extcon/









* 1. 如何检查一个driver的加载时间?
  2. device切成host mode需要哪些流程？

IRQ🡪resuem work🡪sm\_work

为了避免文件的混乱，将每天的学习记录在daily.docx中，对于经常遇到的问题，二次整理在lcd\_issue.docx中

Import 2.8” panel

11.23—11.27

导入新屏，需要对屏的结构进行分析。driver ic是ILI9327是一块RGB驱动芯片，所以外接了bridge ic icn6211。根据模组厂提供的init code，厘清屏的初始化流程。

对屏初始化需要使用spi对driver ic 进行初始化，然后使用iic对bridge ic进行初始化。最后mipi发送图片至bridge ic，转化成RGB信号给driver ic输出显示。

11.24

第一步初始化spi,根据qcom BLSP文档添加spi总线节点，补充status = “ok”;

但是根据移远提供的patch,目前只有一部分spi总线初始化成功。还有一些总线没有挂载成功，不清楚是引脚复用的问题还是TZ source code的影响。

根据ILITEAK厂商提供的code,使用GPIO模拟SPI的方式进行cmd的传输。

spi总结：

cs为片选引脚，cs为低表示有效。spi接多个设备需要接多个cs引脚。所以spi从设备没有地址。同时spi没有应答。所以spi纠错方式比较少，波形无法确认。

/\*---------------------------------------------------------------------------------------------------------------------------\*/

11.27

i2c，高通dts使用的是7位地址。8位地址是7位后补1/0表示读写位。

I2c写数据时序：

ST + 地址（8位）+ACK（0有效）+CMD（8位）+ACK（0）+DATA+ACK（0）+END

读取波形时，使用clk高电平位对sda读取状态。I2c发送一个数据对应是9个clk.

基本上能够读取到一个完整的写时序就代表i2c通信正常。如果i2c没有应答，第一，从波形图中读取i2c的地址，看地址是否对。第二，分析i2c芯片的电源是否有电，使能引脚是否使能。

/\*---------------------------------------------------------------------------------------------------------------------------\*/

11.30

Bridge ic ICN6211集创FAE来协助，量测6211的硬件信号分析软件上的错误，6211没有使能导致ic不工作。修改后量测结果，输入电压正常，输出频率符合数据，mipi输出的RGB波形与彩条模式下的波形一致。确认bridge IC工作正常。

/\*---------------------------------------------------------------------------------------------------------------------------\*/

12.1

Lcd driver ic奕力FAE来协助，量测IOVCC,RESET,CS,波形。量测引脚的电平状态有三段式变化。量测二段三段，二段正常为1.8V，三段异常为2.2V。三段异常。在不接屏的情况下，波形正常。结合bridge ic的调试文档，注意到bridge ic的输出数据电压准位为3.3V,diver ic的电压准位为1.8V。电压准位不匹配导致数据分析不准确。

/\*---------------------------------------------------------------------------------------------------------------------------\*/

12.2

跳线修改bridge ic的电压准位，彩条模式与mipi模式均显示正常。

总结：

驱动ic,学会用硬件的方法分析问题。多个ic共同工作的情况下，首先要确认各个硬件的单独工作状态是否正常。在单独工作正常的情况下，再确认连接部分是否有问题。

分析硬件的逻辑：

1. 量测power

量测ic的输入电压波形是否正常，使能引脚是否正常。工作升压power是否正常。

有关power的部分可以去参考ic的spec.

1. 量测信号输入pin

量测ICN6211的i2c输入信号，如果正常可以从i2c中解析到正确的数据

1. 量测信号输出pin

量测VSYNC，HSYNC，PCLK等pin的输出。

这三步只是一种思路，实际过程不一定按顺序排除。一般是检测是否有输入信号，然后根据情况去判断工作电压是否正常，或者输出信号是否正常。对于量测波形中有异常的问题，一定要深入分析具体原因。比如这次三段式电压，三段电压分别是多少，中间电压状态存在多久，最高电压是否是准位电压？不要忽略细节。

硬件分析很重要，后续的工作中需要去培养这方面的能力。软件分析是解决逻辑错误，硬件分析是定位错误。结合硬件可以更快的发现问题。实际工作中，问题的原因的多方多面的，不仅仅是代码的问题，需要协调各方人员一起推动去解决问题，需要锻炼自己沟通的能力。在一些新技术上，有薄弱点很正常，更重要的是如何解决。

Analog PWM to driver buzzer

12/17

在pwm资源不够用时，需要使用其他方法去模拟一段有频率有占空比的波形。

1. 首先确认cpu端口是否有其他资源引脚可以作为pwm引脚。  
   高通的平台一般都是PMU输出pwm.也需要仔细查找。
2. 使用gpio模拟pwm,最简单可行的方法。但是频率不可控，占用cpu等。

使用线程控制gpio输出pwm信号。

<https://blog.csdn.net/u010632165/article/details/89722539#_11>

1. 使用GP\_CLK时钟信号模拟。

<https://blog.csdn.net/u010787514/article/details/82463356>

<https://blog.csdn.net/qq_35141454/article/details/101545510>

1. 使用hrtimer高精度定时器模拟。

<https://blog.csdn.net/feixiangtiakongn/article/details/103909288?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.control>

<https://blog.csdn.net/qq_35141454/article/details/101428415>

什么样的IO可以代替PWM，第一可以输出频率，是否可以修改duty；第二，输出电流是多少，是否符合要求。Cyclo的buzzer使用了一个放大电路，应该注意输入是否符合电路要求。

创建文件节点的几种方法

12/22

1. device\_creat
2. DEVICE\_ATTR

ssize\_t show(); 🡪cat

ssize\_t store(); 🡪echo

static DEVICE\_ATTR(name,0644,show,store); //权限不对,会编译错误

在probe中添加device\_create\_file(dev,dev\_attr\_name);

创建的文件节点在sys/devices/soc/soc:dev\_name/name

如何把文件创建在syc/class中？

设备树如何管理gpio function

12/24

drivers/pinctrl/qcom/pinctrl-msm8909.c中，

static const struct msm\_pingroup msm8909\_groups[]注册每个gpio的function table,

将每个功能列为enum,

enum msm8909\_functions {

...

msm\_mux\_gcc\_gp1\_clk\_b,

...

};

static const char \* const gcc\_gp1\_clk\_b\_groups[] = {

"gpio14",

};

static const struct msm\_function msm8909\_functions[] = {

…

FUNCTION(gcc\_gp1\_clk\_b),

…

};

devm架构

12/30

有一些以devm开头的申请资源的函数

devm\_kzalloc(dev,size);

alloc\_dr(release,size,gfp,node)

devres\_add

devm\_get\_gpiod\_from\_child(dev,con\_id,child);

1.devres\_alloc(release,size,flag)

2.devres\_add(dev,devres)

1.将申请的资源与device绑定，驱动注册失败时调用release函数

驱动注册失败会进入bus\_remove\_driver…

2.将资源添加到dev->devres链表中，释放时根据链表释放。

devm相关代码在各级目录的devres.c中，了解相应的部分最好是去追一遍代码，si有些函数追不了，用ctags.

of\_node/fwnode

12/31

源码注释:

\* @of\_node: Associated device tree node.

\* @fwnode: Associated device node supplied by platform firmware.

struct device\_node \*of\_node; /\* associated device tree node \*/

struct fwnode\_handle \*fwnode; /\* firmware device node \*/

这个注释太抽象了，设备树的相关节点。

但是struct devices\_node 中 有 struct fwnode\_handle

这一部分没有验证，打印两个地址就可以确认了。

但是从代码的角度确认是父子关系。

所以使用of函数搜node,或者是fwnode函数搜fw\_node都是一样的。

看个人喜好。

fwnode\_get\_named\_gpiod(fwnode,propname)

of\_get\_named\_gpiod\_flags(to\_of\_node(fwnode), propname, 0, &flags);

父节点子节点嵌套在of\_node结构体中

config.xml

1/12/2021

config.xml的加载顺序，以autoBrightnessLevels为例

首先注册符号

frameworks/base/core/res/res/values/symbols.xml:1828: <java-symbol type="array" name="config\_autoBrightnessLevels" />

1.framworks层

core/res/res/values/config.xml

2.device/common层device/qcom/common/device/overlay/frameworks/base/core/res/res/values/config.xml

3.device/company

device/mitac/chiron\_pro/overlay/frameworks/base/core/res/res/values/config.xml:105: <integer-array name="config\_autoBrightnessLcdBacklightValues">

3>2>1

加载顺序应该是类似设备树，覆盖补充加载。

还有两个目录也会有可能出现，应该都是common属性，客制化属性时不考虑

device/qcom/common/automotive/device/overlay/frameworks/base/core/res/res/values/config.xml

vendor/qcom/proprietary/resource-overlay/common/Frameworks/res/values/config.xml

和import android.os.Systemproperties有关系吗？

GP\_CLK

1/15/2021

1. dts 添加pinctrl,
2. dts 添加设备节点引用pinctrl

-clocks = <&clock\_gcc clk\_gp3\_clk\_src>;

+clcoks = <&clock\_gcc clk\_gcc\_gp3\_clk>;

1. clock-gcc-target.c配置clk

F(freq,s,div,M,N);

F(5000, xo, 16, 1, 240)

N的参数是高通使用tool计算出来的，上面的引用节点也是高通给的。

debug clk

dir

/sys/kernel/debug/clk/gcc\_gp2\_clk

enable

/sys/kernel/debug/clk/gcc\_gp2\_clk/enable

rate

/sys/kernel/debug/clk/gcc\_gp2\_clk/rate

对于自己不懂得东西还是多问问比较好，对于clock部分还可以追指针函数的源码

节点权限问题总结

1/19/2021

1. kernel层创建节点

static DEVICE\_ATTR(node\_name,0664,show,store);

show函数对应cat,read函数,

得到buf的内容

store函数对应echo,write函数,

buf写入的参数，count是buf的长度。

返回值如果小于count,show函数会重新读count-ret字节大小的剩下buf.

1. device层节点加权限

在init.target.rc中添加相应的权限，一般都是

on boot

chown system system /sys/devices/soc/soc:buzzer/buzzer

chmod 0660 /sys/devices/soc/soc:buzzer/buzzer

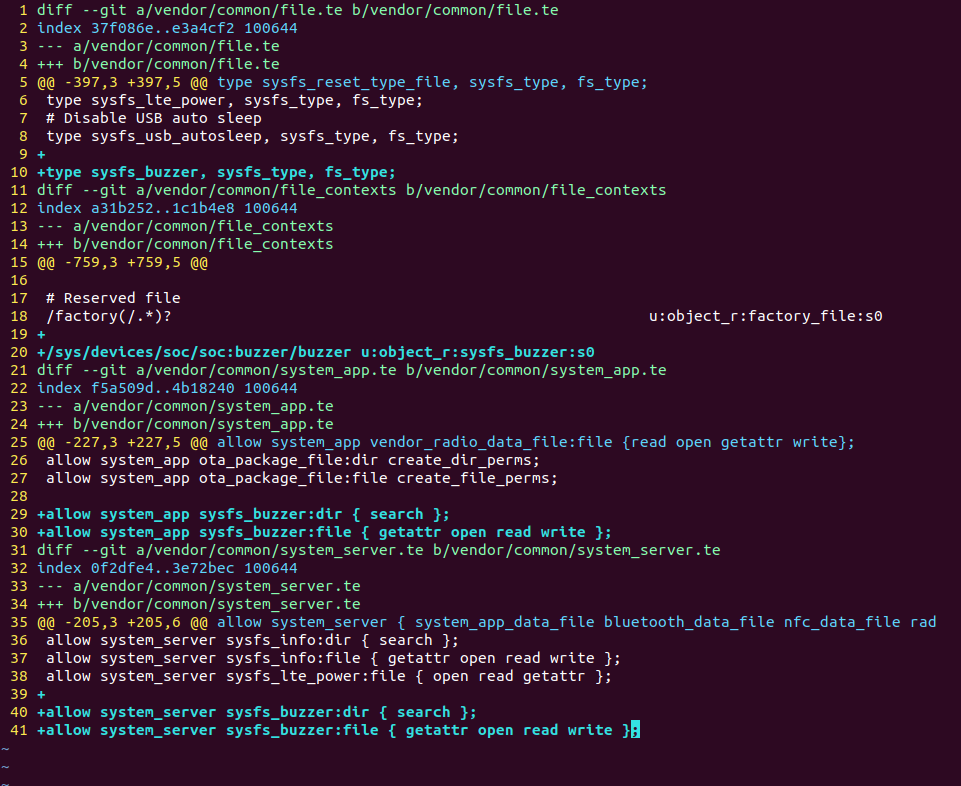
init.target.rc一般修改/device/company\_name/project\_name/init.target.rc

init.tarfet.rc区分project和soc.

on boot是init.target.rc的执行条件

1. sepolicy添加system\_app权限

sepolicy可以添加很多权限，但是可能会引起GMS认证，never\_allow等错误

一般是给节点加system\_server,system\_app权限，再给app添加系统权限。

1. android stuio给apk添加系统权限

platform.pk8和platform.x509.pem生成keystore系统签名文件，编译时自动签名。

1. 下载keytool-importkeypair工具

[https://github.com/getfatday/keytool-importkeypair](https://link.jianshu.com/?t=https%3A%2F%2Fgithub.com%2Fgetfatday%2Fkeytool-importkeypair)

1. ./keytool-importkeypair -k ./release.keystore -p android -pk8 platform.pk8 -cert platform.x509.pem -alias platform

-k 表示要生成的 keystore 文件的名字，这里命名为 release.keystore

-p 表示要生成的 keystore 的密码，一般就用android

-pk8 表示要导入的 platform.pk8 文件

-cert 表示要导入的platform.x509.pem

-alias 表示给生成的 release.keystore 取一个别名，这是命名为 platform

将项目中的platform.pk8和platform.x509.pem copy到keytool目录下，

文件路径在bsp/LINUX/android/build/target/product/security/

<https://www.mobibrw.com/2015/2826>

<https://blog.csdn.net/weixin_33804990/article/details/89617190>

1. 在android studio中使用签名，

在项目目录右键open module setting ->signing configs导入keystore

也可以在Gradle Scripts->build.gradle(app)添加代码

<https://blog.csdn.net/u011904605/article/details/53148683>

NavigationBarPosition

issue:triton + region image navigation在display size为large时，

getResources().getDisplayMetrics()判断navgiation在右侧

nvgationbarPosition

services/core/java/com/android/server/policy/PhoneWindowManager.java

navigationBarPosition函数中判断position位置是mNavigationBarCanMove，

CanMove也会决定getDisplayMetrics获取到屏幕尺寸。

mNavigationBarCanMove = width != height && shortSizeDp < 600;

还是需要多看源码，多动手。

getResources().getDisplayMetrics()

getRealMetrics获取屏幕的分辨率

getMetrics获取屏幕分辨率-navigationbarPosition

可以在android.util.DisplayMetrics中查看相关源码，如果去追这一部分代码，可以看到最后只是返回了R.height,R.width。具体这个类是怎么填充的看不到。

canMove决定Metrics应该还是从PhoneWindowManage处去追。

LCD driver ic(ILI9881C)

command

写command需要page + address

auto brightness自动亮度调节

1. 分析流程

settings代码目录 package/apps

<https://blog.csdn.net/lilidejing/article/details/45022697>

<https://blog.csdn.net/decisiveness/article/details/51899173>

GPIO

KBA-190803031133

performance

list of URL

<https://www.kernel.org/>

<https://www.kernel.org/doc/Documentation/>

<http://ilinuxkernel.com/?cat=3>