1.编译代码

1.1.源代码全编译

source build/envsetup.sh

source gk\_patch/prebuild.sh //tvos

lunch -eng

make -j64

1.2.生成GokeUpgrade.bin

make -j64-->soure device/goke/kunpeng/build/script/createUsbUpgradeBin.sh

1.3.生成update.zip

make -j64-->make otapackage-->out目录下生成update.zip文件

2.编译版本脚本路径

device/goke/kunpeng/build/config/project\_cfg.mk 控制编译版本 aosp公版 cmcc中移物联 ip\_hunan湖南有线

3.版本号BUILD\_NUMBER脚本路径

Device/goke/kungpeng/customize/ip\_hunan/apps/system/Android.mk

4.内置属性字段

**Device/goke/Gk6323V100\device.mk**

#default upgrade address

PRODUCT\_PROPERTY\_OVERRIDES += \

persist.sys.ip.url=http://apkupdate.hunancatv.cn:90

**device/goke/kungpeng/customize/ip\_hunan/Android.mk**

ADDITIONAL\_BUILD\_PROPERTIES += \

persist.sys.ip.url=http://apkupdate.hunancatv.cn:90

5.账户密码

5.1.Gerrit账户

账户：qinzahngshuai

密码：qinzhangshuai123

5.2.内网邮箱

账户：qinzhangshuai@gkdesign.com

密码：qzs@2020

收发服务器：192.168.2.203

5.3.软通邮箱

账户：zsqinc@isoftstone.com

密码：ipsa@2020

5.4.外网FTP

Host：ftp://192.168.99.252

账户：ftp\_6323

密码：664cV2VC

5.5.内网Ubuntu.5

Host：192.168.105.5

账户：qinzhangshuai

密码：password

5.6.Samba虚拟磁盘.5

目录：\\192.168.105.5\samba\qinzhangshuai

账户：qinzhangshuai

密码：stb002

5.7.内网Ubuntu.3

Host：192.168.105.3

账户：qinzhangshuai

密码：abc@123

5.8.Samba虚拟磁盘.3

目录：\\192.168.105.3\samba\qinzhangshuai

账户：qinzhangshuai

密码：stb001

5.9.Redmine账户

账户：qinzhangshuai

密码：12345678

5.10.WIFI账户密码

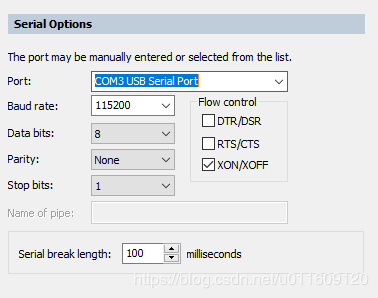
账户：zywl

密码：goke-stb123

6.连接serial

protocol serial

baud rate 115200



7.刷机

reboot 重启机顶盒

ugoke 刷机

reset 重置机顶盒

8.内网外网拷贝

filecheck /image/Gk6323V100/GoKeUpgrade.bin

9.命令行给 Android 输入框填充

input text "你好"

10.IPv6网络模块

framework/base/services/java/com/android/server/SystemServer.java 系统服务

framework/base/services/java/com/android/server/EthernetService.java

framework/base/core/java/android/net/EthernetDataTracker.java ->startIpv6()->runDhcpv6 有线网络

framework/base/ethernet/java/android/net/ethernet/EthernetManager.java

hardware/libhardware\_legacy/wifi/wifi.c

external/dhcpcd/

system/core/libnetutils/dhcpv6\_utils.c

device/goke/kunpeng/etc/init\_kunpeng.rc



11.adb 连接电视盒子

serial com3下执行

busybox ifconfig 查看盒子 ip

adb root

Windows下执行

adb remout

adb connect 192.168.98.35

adb remount

12.mm模块编译

mm -b 编译目标文件夹

看输出日志 Install 对应重新编译的文件

adb push 到 out输出的对应文件夹下

13.如何查看IPv6是否连接

busybox ifconfig

ipv6 link 本地连接地址

ipv6 global 网路连接地址

ps | grep dhclient

ps | grep dhcpcd

ps | grep dh\*

getprop | grep dh

14.赋予盒子目录读写权限

mount -o remount,rw /system 赋予读写权限 //终端需要先执行su

15.查看需要初始化开启的服务

cat init.kunpeng.rc

16.WIFI芯片驱动移植

hardware/libhardware\_legacy/wifi/wifi.c 108

hardware/libhardware\_legacy/wifi/wifi.c 343

hardware/libhardware\_legacy/include/hardware\_legacy/wifi.h 68

**hardware/libhardware\_legacy/wifi/wifi.c**

static wifi\_device\_s devices[] = {

设备ID需要去真实设备查看

{WIFI\_REALTEK\_RTL8189FTV.”024c:f129”},

}

#define DRIVER\_MODULE\_RTL8189FTV 2, \

{\

通用cfg80211驱动加载

{“cfg80211”,”/system/lib/modules/cfg80211.ko”,””,”cfg80211”},\

Wifi驱动8189ftv加载，黄色文字一般与.ko文件保持一致

{“8189ftv”,”/system/lib/modules/rtl8189ftv.ko”,”ifname=wlan0 if2name=p2p0”,”8189fs”}\

}

#define IS\_P2P\_SUPPORTED(id)(

|| WIFI\_REALTEK\_RTL8189FTV==id

)

Static wifi\_modules\_s sta\_drivers[] = {

{DRIVER\_MODULE\_RTL8189FTV},

}

Static wifi\_modules\_s ap\_drivers[] = {

{DRIVER\_MODULE\_RTL8189FTV},

}

**hardware/libhardware\_legacy/include/hardware\_legacy/wifi.h**

typedef enum{

数字按照文件内自动增加

WIFI\_REALTEK\_RTL8189FTV = 36，

}

Device/goke/Gk6323V100/BoardConfig.mk

# RTL8189FTV

BOARD\_BLUETOOTH\_WIFI\_DEVICE\_ RTL8189FTV := n

编译：Hardware\libhardware\_legacy mm -b

生成文件路径: out/…/system/lib/ libhardware\_legacy.so

17..ko文件路径

源码中：device/goke/kunpeng/customine/ip\_hunan/gkapi/modules

盒子中：/system/lib/modules # ls

18.WIFI硬件设备ID查看

Cat /sys/bus/sdio/devices/mmc1:001:1/uevent

DRIVER = rtl8189ftv

SDIO\_ID = 024c:f179

19.查看驱动是否加载正常

Lsmod

20.开启关闭指定服务

setprop ctl.start dhclientDns\_eth0 开启指定服务

setprop stl.stop dhclientDns\_eth0 停止指定服务

21.查看指定log

logcat -s EthernetDataTracker

22. 路由器无状态第一个选项，有状态第二个选项

23.adb命令启动应用

adb shell monkey -p com.android.settings -c android.intent.category.LAUNCHER 1

24.ko文件编译教程

105.3服务器/zhanghao/kernel/driver/GokeKernel\_20200331/rtl88x2CS\_WiFi\_.../ 下执行make

rtl88x2CS\_WiFi\_... 这个文件夹在以下地址取

105.3服务器/qinzhangshuai/work/tvos2/device/goke/kengpeng/wifi/rtl8822bs

注：里面的Makefile改过，更改内容，搜索6323 ；涉及知识点：交叉编译

rtl88x2CS\_WiFi\_... 原始厂商路径

105.5服务器/zhanghao/wifi

注：这里面的makefile没有改过

25.解决四层板死机

关闭动态电源管理

sete DVFS disable

sete DVFS\_V disable

savee

26.通过包名查看应用名

pm list package -f |grep 包名

27.拷贝apk到U盘

cp -rf /data/app /mnt/sda/sda4/

28.设置SN，MAC属性

DevicePropertiesService\_Test set stb\_id "0270050020160000109"

DevicePropertiesService\_Test set mac "00301BBA028B"



00:30:1B:BA:02:a0    15928163291   孙晓东  
00:30:1B:BA:02:a1    15802838143   陈恩洪  
00:30:1B:BA:02:a2    18081193860   刘贤  
00:30:1B:BA:02:a3     18980011613   林锐  
00:30:1B:BA:02:a4     13340984853   秦张帅  
00:30:1B:BA:02:a5     18525365739   任磊  
00:30:1B:BA:02:a6     18108458806   苏富英  
00:30:1B:BA:02:a7     18175160166  杨涛龙  
00:30:1B:BA:02:a8     17674107865  李旭东  
00:30:1B:BA:02:a9     18908072295   牛纪卫

29. iperf参数含义与使用教程

1.Iperf参数介绍

|  |  |
| --- | --- |
| **命令行选项** | **描述** |
| 客户端与服务器共用选项 | |
| -f, --format [bkmaBKMA] | **eg:iperf -c 222.35.11.23 -f K**  格式化带宽数输出。支持的格式有： 'b' = bits/sec 'B' = Bytes/sec 'k' = Kbits/sec 'K' = KBytes/sec 'm' = Mbits/sec 'M' = MBytes/sec 'g' = Gbits/sec 'G' = GBytes/sec 'a' = adaptive bits/sec 'A' = adaptive Bytes/sec 自适应格式是kilo-和mega-二者之一。除了带宽之外的字段都输出为字节，除非指定输出的格式，默认的参数是a。 注意：在计算字节byte时，Kilo = 1024， Mega = 1024^2，Giga = 1024^3。通常，在网络中，Kilo = 1000， Mega = 1000^2， and Giga = 1000^3，所以，Iperf也按此来计算比特（位）。如果这些困扰了你，那么请使用-f b参数，然后亲自计算一下。 |
| -i, --interval # | **eg:iperf -c 222.35.11.23 -i 2**  设置每次报告之间的时间间隔，单位为秒。如果设置为非零值，就会按照此时间间隔输出测试报告。默认值为零。 |
| -l, --len #[KM] | **eg:iperf -c 222.35.11.23 -l 16 -m**  设置读写缓冲区的长度。TCP方式默认为8KB，UDP方式默认为1470字节。 |
| -m, --print\_mss | 输出TCP MSS值（通过TCP\_MAXSEG支持）。MSS值一般比MTU值小40字节。通常情况 |
| -p, --port # | **eg:iperf -s -p 9999;iperf -c 222.35.11.23 -p 9999**  设置端口，与服务器端的监听端口一致。默认是5001端口，与ttcp的一样。 |
| -u, --udp | 使用UDP方式而不是TCP方式。参看-b选项。 |
| -w, --window #[KM] | 设置套接字缓冲区为指定大小。对于TCP方式，此设置为TCP窗口大小。对于UDP方式，此设置为接受UDP数据包的缓冲区大小，限制可以接受数据包的最大值。 |
| -B, --bind host | 绑定到主机的多个地址中的一个。对于客户端来说，这个参数设置了出栈接口。对于服务器端来说，这个参数设置入栈接口。这个参数只用于具有多网络接口的主机。在Iperf的UDP模式下，此参数用于绑定和加入一个多播组。使用范围在224.0.0.0至239.255.255.255的多播地址。参考-T参数。 |
| -C, --compatibility | 与低版本的Iperf使用时，可以使用兼容模式。不需要两端同时使用兼容模式，但是强烈推荐两端同时使用兼容模式。某些情况下，使用某些数据流可以引起1.7版本的服务器端崩溃或引起非预期的连接尝试。 |
| -M, --mss #[KM} | 通过TCP\_MAXSEG选项尝试设置TCP最大信息段的值。MSS值的大小通常是TCP/IP头减去40字节。在以太网中，MSS值 为1460字节（MTU1500字节）。许多操作系统不支持此选项。 |
| -N, --nodelay | 设置TCP无延迟选项，禁用Nagle's运算法则。通常情况此选项对于交互程序，例如telnet，是禁用的。 |
| -V (from v1.6 or higher) | 绑定一个IPv6地址。 服务端：$ iperf -s –V 客户端：$ iperf -c <Server IPv6 Address> -V 注意：在1.6.3或更高版本中，指定IPv6地址不需要使用-B参数绑定，在1.6之前的版本则需要。在大多数操作系统中，将响应IPv4客户端映射的IPv4地址。 |
| 服务器端专用选项 | |
| -s, --server | Iperf服务器模式 |
| -D (v1.2或更高版本) | Unix平台下Iperf作为后台守护进程运行。在Win32平台下，Iperf将作为服务运行。 |
| -R(v1.2或更高版本，仅用于Windows) | 卸载Iperf服务（如果它在运行）。 |
| -o(v1.2或更高版本，仅用于Windows) | **eg:iperf -c 222.35.11.23 -o c:\iperflog.txt**  重定向输出到指定文件 |
| -c, --client host | 如果Iperf运行在服务器模式，并且用-c参数指定一个主机，那么Iperf将只接受指定主机的连接。此参数不能工作于UDP模式。 |
| -P, --parallel # | 服务器关闭之前保持的连接数。默认是0，这意味着永远接受连接。 |
| 客户端专用选项 | |
| -b, --bandwidth #[KM] | UDP模式使用的带宽，单位bits/sec。此选项与-u选项相关。默认值是1 Mbit/sec。 |
| -c, --client host | 运行Iperf的客户端模式，连接到指定的Iperf服务器端。 |
| -d, --dualtest | 运行双测试模式。这将使服务器端反向连接到客户端，使用-L 参数中指定的端口（或默认使用客户端连接到服务器端的端口）。这些在操作的同时就立即完成了。如果你想要一个交互的测试，请尝试-r参数。 |
| -n, --num #[KM] | **eg:iperf -c 222.35.11.23 -n 100000**  传送的缓冲器数量。通常情况，Iperf按照10秒钟发送数据。-n参数跨越此限制，按照指定次数发送指定长度的数据，而不论该操作耗费多少时间。参考-l与-t选项。 |
| -r, --tradeoff | 往复测试模式。当客户端到服务器端的测试结束时，服务器端通过-l选项指定的端口（或默认为客户端连接到服务器端的端口），反向连接至客户端。当客户端连接终止时，反向连接随即开始。如果需要同时进行双向测试，请尝试-d参数。 |
| -t, --time # | **eg:iperf -c 222.35.11.23 -t 5**  设置传输的总时间。Iperf在指定的时间内，重复的发送指定长度的数据包。默认是10秒钟。参考-l与-n选项。 |
| -L, --listenport # | 指定服务端反向连接到客户端时使用的端口。默认使用客户端连接至服务端的端口。 |
| -P, --parallel # | 线程数。指定客户端与服务端之间使用的线程数。默认是1线程。需要客户端与服务器端同时使用此参数。 |
| -S, --tos # | 出栈数据包的服务类型。许多路由器忽略TOS字段。你可以指定这个值，使用以"0x"开始的16进制数，或以"0"开始的8进制数或10进制数。 例如，16进制'0x10' = 8进制'020' = 十进制'16'。TOS值1349就是： IPTOS\_LOWDELAY minimize delay 0x10 IPTOS\_THROUGHPUT maximize throughput 0x08 IPTOS\_RELIABILITY maximize reliability 0x04 IPTOS\_LOWCOST minimize cost 0x02 |
| -T, --ttl # | 出栈多播数据包的TTL值。这本质上就是数据通过路由器的跳数。默认是1，链接本地。 |
| -F (from v1.2 or higher) | 使用特定的数据流测量带宽，例如指定的文件。 $ iperf -c <server address> -F <file-name> |
| -I (from v1.2 or higher) | 与-F一样，由标准输入输出文件输入数据。 |
| 杂项 | |
| -h, --help | 显示命令行参考并退出 。 |
| -v, --version | 显示版本信息和编译信息并退出。 |

2.TCP客户端和服务器

iperf需要两个系统，因为一个系统必须充当服务端，另外一个系统充当客户端，客户端连接到需要测试速度的服务端  
1.在需要测试的电脑上，以服务器模式启动iperf

iperf -s

可以看到类似于下图的输出

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

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

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

2.在第二台设备上，以客户端模式启动iperf连接到第一台电脑ip地址

iperf -c 198.51.100.5

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

Client connecting to 198.51.100.5, TCP port 5001

TCP window size: 45.0 KByte (default)

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

[ 3] local 198.51.100.6 port 50616 connected with 198.51.100.5 port 5001

[ ID] Interval Transfer Bandwidth

[ 3] 0.0-10.1 sec 1.27 GBytes 1.08 Gbits/sec

3.这时可以在第一步中的服务端终端看到连接和结果，类似下图

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

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

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

[ 4] local 198.51.100.5 port 5001 connected with 198.51.100.6 port 50616

[ ID] Interval Transfer Bandwidth

[ 4] 0.0-10.1 sec 1.27 GBytes 1.08 Gbits/sec

3.UDP客户端和服务器

使用iperf，还可以测试通过UDP连接实现的最大吞吐量  
1.启动UDP iperf服务

iperf -s -u

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

Server listening on UDP port 5001

Receiving 1470 byte datagrams

UDP buffer size: 208 KByte (default)

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

2.将客户端连接到iperf UDP服务器，以客户端模式启动iperf连接到第一台电脑ip地址

iperf -c 198.51.100.5 -u

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

Client connecting to 198.51.100.5, UDP port 5001

Sending 1470 byte datagrams

UDP buffer size: 208 KByte (default)

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

[ 3] local 198.51.100.6 port 58070 connected with 198.51.100.5 port 5001

[ ID] Interval Transfer Bandwidth

[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec

[ 3] Sent 893 datagrams

[ 3] Server Report:

[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec 0.084 ms 0/ 893 (0%)

* 1.05Mbits/sec远低于TCP测试中观察到的值，它也远远低于1GB 的最大出站贷款上限，这是因为默认情况下，iperf讲UDP客户端的贷款限制为每秒1Mbit。  
  3.可以用-b标志更改此值，讲数字替换为要测试的最大带宽速率。如果需要测试网络速度，可以将数字设置为高于网络提供商提供的最大带宽上线：

iperf -c 198.51.100.5 -u -b 1000m

* 这将告诉客户端我们希望尽可能达到每秒1000Mbits的最大值，该-b标志仅在使用UDP连接时有效，因为iperf未在TCP客户端上设置带宽限制。

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

Client connecting to 198.51.100.5, UDP port 5001

Sending 1470 byte datagrams

UDP buffer size: 208 KByte (default)

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

[ 3] local 198.51.100.5 port 52308 connected with 198.51.100.5 port 5001

[ ID] Interval Transfer Bandwidth

[ 3] 0.0-10.0 sec 966 MBytes 810 Mbits/sec

[ 3] Sent 688897 datagrams

[ 3] Server Report:

[ 3] 0.0-10.0 sec 966 MBytes 810 Mbits/sec 0.001 ms 0/688896 (0%)

[ 3] 0.0-10.0 sec 1 datagrams received out-of-order

4.双向测试

在某些情况下，可能希望测试两台服务器以获得最大吞吐量。使用iperf提供的内置双向测试功能可以轻松完成此测试。

1.要测试两个连接，从客户端运行一下命令,ip为服务端ip地址

iperf -c 198.51.100.5 -d

2.结果是iperf将在客户端服务器上启动服务器和客户端(198.51.100.6)连接。完成此操作后，iperf会将iperf服务器连接到客户端，该连接现在既充当服务器连接又充当客户端连接。

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

Server listening on TCP port 5001

TCP window size: 85.3 KByte (default)

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

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

Client connecting to 198.51.100.5, TCP port 5001

TCP window size: 351 KByte (default)

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

[ 3] local 198.51.100.6 port 50618 connected with 198.51.100.5 port 5001

[ 5] local 198.51.100.6 port 5001 connected with 198.51.100.5 port 58650

[ ID] Interval Transfer Bandwidth

[ 5] 0.0-10.1 sec 1.27 GBytes 1.08 Gbits/sec

[ 3] 0.0-10.2 sec 1.28 GBytes 1.08 Gbits/sec

在服务器，可以看到：

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

Client connecting to 198.51.100.6, TCP port 5001

TCP window size: 153 KByte (default)

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

[ 6] local 198.51.100.5 port 58650 connected with 198.51.100.6 port 5001

[ 6] 0.0-10.1 sec 1.27 GBytes 1.08 Gbits/sec

[ 5] 0.0-10.2 sec 1.28 GBytes 1.08 Gbits/sec

30.编译四层板系统

打开 /device/goke/kunpeng/customize/ip\_hunan/gkapi/gboot/

编译四层板：用DDR3\_IPTV4/bootstrap\_e.bin替换bootstrap\_e.bin

out/下输出路径：out/target/product/Gk6323V100/prebuilts32/

image/下输出路径：image/ Gk6323V100/

**注意：确保所有的bootstrap\_e.bin一致**

31.ip\_hunan项目，编译打开wifi配置文件

源码配置文件路径：/device/goke/kunpeng/customize/ip\_hunan/gkapi/customerconfig/gkconf/sdio\_gmap.ini

系统配置文件路径：/stbconfig/gkconf/sdio\_gmap.ini

#define SDIO\_CD\_OPT 3 //有WIFI

#define SDIO\_CD\_OPT 0 //无WIFI

32.编译打开Bluetooth配置文件

源码配置文件路径：/device/goke/kunpeng/bluetooth/realtek8xxx//system/etc/bluetooth/uart/rtkbt.conf

系统配置文件路径：/system/etc/bluetooth/rtkbt.conf

BtDeviceNode=/dev/ttyS2 //打开uart bluetooth

BtDeviceNode=/dev/rtk\_btusb //打开usb bluetooth

33.打开，关闭红外

关红外：busybox devmem 0x1f007b00 16 0x01be

开红外：busybox devmem 0x1f007b00 16 0x01bf

34.打开kernel日志命令

echo 7 > /proc/sys/kernel/printk 调试级：调试级别的信息

内核printk的打印级别：<https://blog.csdn.net/sweetfather/article/details/81557730>

<https://blog.csdn.net/zgxzgxzg/article/details/46774709>

35.通过U盘烧录boot.img到盒子 uboot

fatload usb 0 0x20200000 boot.img 0xB00000  
mmc write.p 0x20200000 boot 0xB00000

uboot 常用命令：<https://blog.csdn.net/luckywang1103/article/details/77865476>

36.蓝牙配对流程分析

//需要过滤多个动作，则调用IntentFilter对象的addAction添加新动作

blueReceiver = new BluetoothReceiver();  
IntentFilter foundFilter = new IntentFilter();  
foundFilter.addAction(BluetoothDevice.*ACTION\_FOUND*);  
foundFilter.addAction(BluetoothAdapter.*ACTION\_DISCOVERY\_FINISHED*);  
foundFilter.addAction(BluetoothDevice.*ACTION\_BOND\_STATE\_CHANGED*); //配对状态发生改变广播  
foundFilter.addAction(BluetoothDevice.*ACTION\_PAIRING\_REQUEST*); //配对请求广播  
registerReceiver(blueReceiver, foundFilter);

private class BluetoothReceiver extends BroadcastReceiver {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 String action = intent.getAction();  
 //根据远程设备所携带的type信息判断是否可以获取到pairingkey，type值通过BluetoothDevice.EXTRA\_PAIRING\_VARIANT获取到  
 int type = intent.getIntExtra(BluetoothDevice.*EXTRA\_PAIRING\_VARIANT*,BluetoothDevice.*ERROR*);  
 //pairingkey通过BluetoothDevice.EXTRA\_PAIRING\_KEY获取到  
 int pairingKey = intent.getIntExtra(BluetoothDevice.*EXTRA\_PAIRING\_KEY*,BluetoothDevice.*ERROR*);

涉及类：

packages/apps/Settings/src/com/android/settings/bluetooth/BluetoothPairingRequest.java 监听action的类

packages/apps/Settings/src/com/android/settings/bluetooth/BluetoothPairingDialog.java 配对弹窗类

frameworks/base/core/java/android/bluetooth/BluetoothDevice.java 用于指代某个蓝牙设备，通常表示对方设备

frameworks/base/core/java/android/bluetooth/IBluetooth.aidl

packages/apps/Bluetooth/src/com/android/bluetooth/btservice/AdapterService.java IBluetooth.aidl实现类

37.河北IPBox遥控器按键集成



图一：遥控器按键码值匹配图

**源代码文件路径：/device/goke/kunpeng/customize/hebei\_ipbox/chuangpin/gkapi/customerconfig/gkconf/Irsetting.ini**

**系统文件路径：/stbconfig/gkconf /Irsetting.ini**

[Kernel]

0=KIR\_GK6323TV

1=KIR\_GK6323FJ

2=KIR\_JIUZHOU

3=KIR\_HeBei

4=KIR\_HBIPBox //添加自己的遥控器名称

[KIR\_HBIPBox] //遥控器名称

Enable=true #true or false

Protocol=0x01

Header=0x01fd //客户码(0x代表16进制) 查看遥控器按键码值匹配图。

Keymap= KIR\_HBIPBox\_Keymap //按键映射名称

[KIR\_HBIPBox\_Keymap] //按键映射名称

KEY\_POWER = 0xdc 查看遥控器按键码值匹配图。 (0x代表16进制)

KEY\_MUTE = 0x9c 同上

KEY\_0 = 0x87

KEY\_1 = 0x92

KEY\_2 = 0x93

KEY\_ESC = 0xc5

KEY\_UP = 0xca

KEY\_DOWN = 0xd2

KEY\_LEFT = 0x99

KEY\_RIGHT = 0xc1

KEY\_MENU = 0x98

KEY\_VOLUMEUP = 0xdd

KEY\_VOLUMEDOWN = 0x8c

KEY\_BACK = 0x82

KEY\_ENTER = 0xce

其他按键添加也遵循此方法，但是一定要key与value匹配。

38.终止Thread线程的方法。

终止线程的三种方法

1. 使用退出标志，使线程正常退出，也就是当方法完成后线程终止。

public volatile boolean exit = false;

myThread.exit = true;

一定要用volatile，这样程序就会一直检查volatile的值，且不会在多线程中复制，而是共享。

2. 使用stop方法强行终止线程（这个方法不推荐使用，因为stop和suspend、resume一样，也可能发生不可预料的结果）。

myThread.stop();

3. 使用interrupt方法中断线程。

myThread.interrupt();

39.TVOS客户目录与版本建立

1.cd device/goke/kunpeng/customize

2.复制一个客户项目如hunan

3.gkapi是驱动释放的

4.修改所有包含hunan字眼的product\_target

5. Gk6323V100/BoadConfigCommon.mk

Gk6323V100/device.mk

build/config/project\_cfg.mk

scripts/目录下所有差异

6.第一次编译没有release环境，只能先编译development环境

source build/envsetup.sh

lunch

userdebug

./device/goke/kungpeng/build/scripts/switch\_dev\_release.sh

先选择development

git diff ./device/goke/kungpeng/build/config/project\_cfg.mk 可以看到此文件已发生变化

lunch 因为变化了一定要lunch

make -j32

对比sdk下out目录输出与customize/定制目录/gkapis是否相同

编译完成

./device/goke/kungpeng/build/scripts/switch\_dev\_release.sh

install

这样就编译检查完成

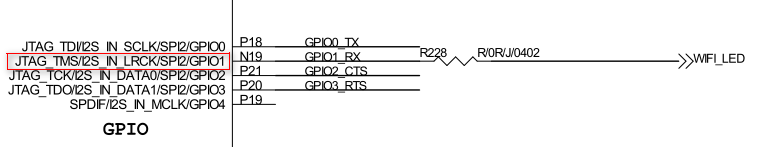
./device/goke/kungpeng/build/scripts/switch\_dev\_release.sh

release

lunch

userdebug

检查release版是否可以编译成功

40.wifi灯的驱动代码控制

WIFI\_LED针脚定义图

1.在wifi\_led针脚定义图中可以看到，这个wifi\_led是通过GPIO1控制的。

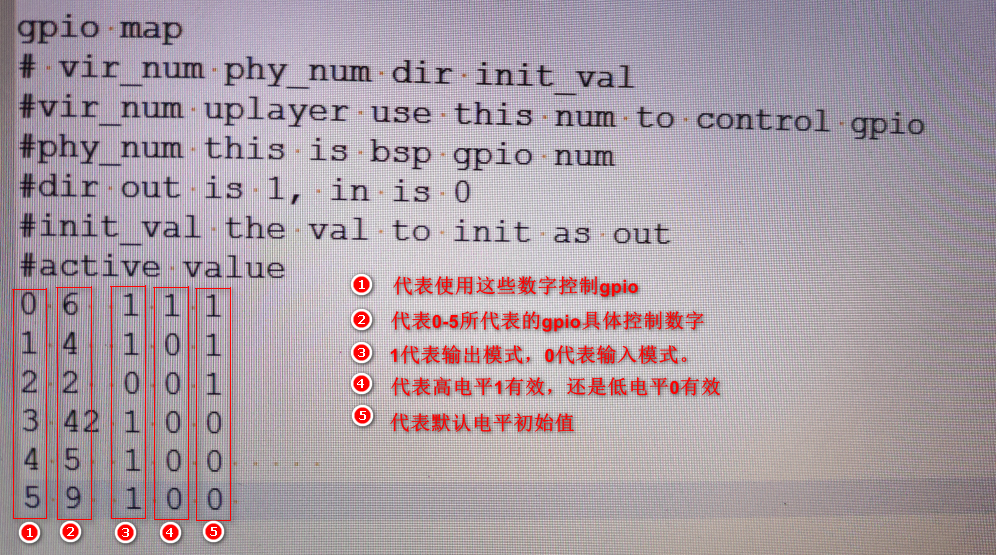
2. GPIO1的控制代码路径:

/device/goke/kunpeng/sdk/source/gkapi/source/customize/an\_gansu/include/gkbsp/gdrv\_gpio.h

#define PIN\_GPIOI 42

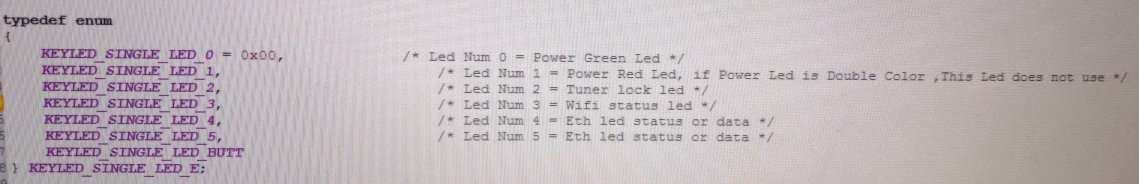
3.GPIO1的上层引用代码控制路径：

/device/goke/kunpeng/customize/ip\_gansu/gkapi/customerconfig/gkconf/front\_panel\_config



4.0-5所代表具体操作灯状态定义如下：

/bootable/recovery/keyled/keyled\_recovery.h



3.在盒子控制wifi\_led灯的位置如下：

stbconfig /gkconf/front\_panel\_config

总结：如果要修改其他盒子的wifi\_led灯控制，只需要修改KEYLED\_SINGLE\_LED\_3所操作的数字3所对应的数字（如：42）即可，而42所代表的含义就是wifi\_led所定义的针脚数字，具体是那个针脚需要在图纸上查看，然后在gdrv\_gpio.h查看定义数字，修改即可。

41.进入升级模式的几种方式

1.ok+0，开机上电进入recovery模式，先清除数据，再升级（同一系统，不区分高低版本升级）。

2.ok+2，开机上电进入GkUpgrade.bin升级，不区分系统和版本。

3.U盘根目录创建upgrade文件夹存入update.zip(同一系统，区分版本号，低升高)。

4.串口，开机上电不断按电脑enter，输入ugoke进入GkUpgrade.bin升级，不区分系统和版本

42.查看网络连接状态

有线：

logcat | grep "ConnectivityService"

logcat | grep "updateNetworkInfo"

无线：

logcat | grep "WifiHW"

logcat | grep "WifiStateMachine"

43.设置系统属性打开adb

setprop sys.usb.config adb

44.led灯控制文件

ghal\_api\_frontpanel.h

ghal\_api\_frontpanel.cpp

GHAL\_FRONTPANEL\_开头的接口，如GHAL\_FRONTPANEL\_SetSingleLed

编译更新libgk\_ssp.so

45.替换out/…/app直接编译GkUpgrade.bin

1.在out/…/app下替换需要不用系统签名的app

2.make snod

3.device/goke/kunpeng/build/scripts/releaseALLImage.sh

4. device/goke/kunpeng/build/scripts/createUsbUpgradeBin.sh

46.看编译版本信息

echo version>/proc/gkbsp

47.查看tombstone日志，crash地址所在行

如：

backtrace:

#00 pc 0001bc82 /system/lib/libDTVClient.so (android::InterDTVScanParamS::~InterDTVScanParamS()+71)

#01 pc 000181f1 /system/lib/libDTVClient.so (android::CHDTVClient::~CHDTVClient()+72)

#02 pc 00018759 /system/lib/libDTVClient.so (android::CHDTVClient::getInstance()+44)

#03 pc 0001e003 /system/lib/libDTVClient.so (DTVAL\_getTunerStatus(int)+6)

#04 pc 0001dbcc /system/lib/libdvm.so (dvmPlatformInvoke+112)

#05 pc 0004e123 /system/lib/libdvm.so (dvmCallJNIMethod(unsigned int const\*, JValue\*, Method const\*, Thread\*)+398)

#06 pc 00000214 /dev/ashmem/dalvik-jit-code-cache (deleted)

$ cd out/target/product/Gk6323v100/symbols/system/lib

$ addr2line -e libDTVClient.so 00018759

48.湖南DVB蜗牛TV进入luncher

rm /system/app/CATV\_CA\_2\_2.0\_200409-sign\_signed\_platform\_20200715112606.apk

rm /system/app/CATV\_AUTH\_20\_20.0\_200623.apk

write\_properties mac c0:13:2B:B4:F0:A7

write\_properties  qwer stb\_factorytest\_finish success

49. Android Framework添加自定义接口

1.定义Java接口

在 frameworks/base/core/java目录下，根据包名创建一个子目录，比如我创建的是com\gcable\system\upgrade，在该目录下，新建一个java文件，比如UpgradeBoot.java ，对应的代码如下：

public class UpgradeBoot {  
 private static final String TAG = "UpgradeBoot";  
 public static synchronized Boolean upgradeBootAnimation(UpgradeBoot.BootAnimationType bootAnimationType, File updateImgFile, UpgradeBoot.FileType fileType) {  
 try {  
 Log.d("UpgradeBoot", "upgradeBootAnimation: start");  
 if (updateImgFile != null) {  
 if (UpgradeBoot.BootAnimationType.BOOT\_PIC == bootAnimationType) {  
 Log.d("UpgradeBoot", "upgradeBootAnimation: 修改开机 boot 阶段 LOGO");  
 } else if (UpgradeBoot.BootAnimationType.BOOT\_ANIMATION == bootAnimationType) {  
 Log.d("UpgradeBoot", "upgradeBootAnimation: 修改开机Android 阶段动态图片帧");  
 }  
 return false;  
 } else {  
 Log.d("UpgradeBoot", "upgradeBootAnimation: 表示清空 A 类广告，包括 logo和 bootanimation");  
 return false;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 throw new RuntimeException("Stub");  
 }  
 }  
  
 public static enum FileType {  
 BOOT\_IMAGE,  
 BOOT\_VIDEO;  
 }  
  
 public static enum BootAnimationType {  
 BOOT\_PIC,  
 BOOT\_ANIMATION;  
 }  
}

2.编译接口

上面虽然定义了两个文件，但是默认情况下，编译Android的时候是不会编译到这两个文件的，如果要用编译到对应的文件，需要修改[frameworks/base/Android.mk](http://192.168.3.252:8080/rd-os/rk3288/commit/990c3abc7b1a62e66eafe0682dd9fd6589300f91#diff-1)文件，在对应的LOCAL\_SRC\_FILES宏中添加对应的文件即可，相应的patch文件如下：

LOCAL\_SRC\_FILES += \

packages/services/Proxy/com/android/net/IProxyPortListener.aidl \

core/java/android/os/IDisplayDeviceManagementService.aidl \

core/java/com/gcable/system/upgrade/UpgradeBoot.java

3.编译更新接口

编译源码，因为添加了接口，所以需要

make update-api

4.部分编译framework代码

$ cd frameworks/base/core/java /core/java/com/gcable/system/upgrade/

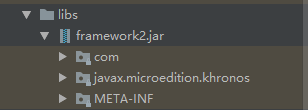
$ mm -B

5.拷贝编译生成的源码classes.jar

拷贝编译生成的源码class.jar到Android studio工程中看是否能使用

$ filecheck out/target/common/obj/JAVA\_LIBRARIES/framework2\_intermediates/classes.jar

以下是classes.jar重命名的framework2.jar



6.拷贝编译包含新接口的framework2.jar替换盒子/system/framework目录下文件

拷贝framework2.jar到盒子相应的目录下，目的是让使用新接口的app可以在盒子中调用到真正的framework接口。

参考链接：<https://blog.csdn.net/u013638184/article/details/88218600>

<https://www.cnblogs.com/rossoneri/p/9651782.html>

50.编译更新最新的驱动代码

1.编译最新gi驱动代码系统

$ source build/envsetup.sh

$ lunch

$ ./device/goke/kungpeng/build/scripts/switch\_dev\_release.sh

$ development 选择开发者模式编译会编译gi源码，这样就会编译打最新的驱动代码

$ lunch

$ make -j32 这样编译的系统使用的就是最新的驱动代码

2.发布最新的gi驱动代码，让release版同步到最新驱动代码

$ ./device/goke/kungpeng/build/scripts/switch\_dev\_release.sh

$ install 拷贝gkapi目录下编译的最新gi驱动相关库到customzise下，让release版也能用到最新的gi驱动代码

$ git status . 可以看到customzise/项目版本/gkapi下文件已经变为最新编译的gi驱动文件

$ 发布差异代码，这样以后编译release版，就会是这次编译提交的代码。

3.比较release版本是否已经更新到编译gi驱动

1.打开./device/goke/kungpeng/customize/aosp/gkapi

2.打开./device/goke/kungpeng/sdk/source/gkapi/source/out/release/gkapi

3.对比工具比较这两个目录文件是否完全相同。