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

Ubuntu在nautilus浏览器和terminal下面使用标签式窗口

1，文件浏览器下面 Ctrl+T ( ctrl + h 显示隐藏文件 )

2，terminal下面 Ctrl+Shift+T

卸载virtualbox： apt-get remove virtualbox\*

**安装flash插件**： sudo apt-get install flashplugin-installer

**ubuntu使用wine安装source insight4**

1. sudo apt-get install wine

2. wine …/sourceinsight4086-setup.exe

安装文件路径在/home/ch/.wine/drive\_c/Program Files (x86)

***注意：上班时间使用服务器，make -j8，中午或晚上再make -j16 和make clean***

Ubuntu1404 系统设置打不开 鼠标老是闪烁

sudo apt-get install Ubuntu-desktop或者

sudo apt-get install gnome-control-center

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

编译、打包、烧写步骤

(0) 选择内核模块：

进入到lichee/linux-3.x/目录，执行

make ARCH=arm menuconfig 选择编译指定模块

{

make clean仅仅是清除之前make命令产生的object文件及可执行文件。   
**make distclean**要清除所有生成的文件（包括configure生成的文件）。

}

(1) 编译内核：

xxx/lichee$ ./build.sh config

xxx/lichee$ ./build.sh

(2) 编译brandy（可选）u-boot

方法一：

cd lichee/brandy/

./build.sh –p sun8iw5p1

编译失败，没有生成目标文件：

先搜索 grep -rns “name” ./

再逐一更改或添加

方法二：

cd lichee/brandy/u-boot-2011.09

make distclean && make sun8iw5p1 –j8

(3) 编译Android ( 每打开一个终端都要执行source 和lunch，配置环境变量 )

source build/envsetup.sh

lunch

extract-bsp

**拷贝了bImage（kernel文件）和modules（modules文件夹）**

make –j8 ( 或**make installclean** )

(4) 打包

pack ( -d用作串口调试，查看信息 )

(5) 烧录固件：

方法一：启动烧录程序LiveSuit（只需USB连接，拔掉电源）

在~Bin/LiveSuit/目录 ./LiveSuit.sh　然后手动选择固件

方法二：adb下使用命令（USB+电源）

adb shell 进入调试，reboot efex执行烧录

1. 串口调试

minicom

1. USB调试

adb shell

getevent

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

调试LCD

*注意：（R16不能编译uboot）lcd相关代码包含在disp驱动模块中，在menuconfig中添加支持*

***[ lcd\_dclk\_freq = lcd\_ht \* lcd\_vt \* 60 ]***

(1) 配置sys\_config.fex中lcd屏参数

[disp\_init]

disp\_init\_enable = 1

disp\_mode = 0

screen0\_output\_type = 1

screen0\_output\_mode = 4

[lcd0\_para]

lcd\_used = 1

lcd\_driver\_name = "default\_lcd\_2"

lcd\_if = 0

lcd\_x = 1024

lcd\_y = 600

lcd\_width = 150

lcd\_height = 94

lcd\_dclk\_freq = 49

lcd\_pwm\_used = 1

lcd\_pwm\_ch = 0

lcd\_pwm\_freq = 50000

lcd\_pwm\_pol = 1

lcd\_hbp = 150

lcd\_ht = 1344

lcd\_hspw = 5

lcd\_vbp = 15

lcd\_vt = 635

lcd\_vspw = 5

(2) 添加新屏驱动支持

*注意：判断lcd驱动使用的是那一个，只需看编译后是否有.o文件生成*

在/lichee/linux-3.4/drivers/video/sunxi/lcd/panels/目录下

1> 仿照　"default\_panel.c"和“default\_panel.h”

编写 对应的　"default\_panel\_2.c"和“default\_panel\_2.h”

\_\_lcd\_panel\_t default\_panel\_2 = {

/\* panel driver name, must mach the name of lcd\_drv\_name in sys\_config.fex \*/

.name = "default\_lcd\_2",

2> 在panels.c中将新屏驱动添加到panel\_array中

… ...

extern \_\_lcd\_panel\_t default\_panel\_2;

... ...

\_\_lcd\_panel\_t\* panel\_array[] = {

&default\_panel,

… ...

&default\_panel\_2,

3> 修改Makefile

在上一级目录的Makefile中，添加default\_panel\_2.o

lcd-objs:=...panels/panels.o panels/default\_panel\_2.o

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

调试TP

*烧录到平板的固件源码在 android/out/target/product/astar-evb20/*

(1) 修改sys\_config.fex中tp参数

[ctp\_para]

ctp\_used = 1

ctp\_name = "gslX680"

ctp\_twi\_id = 0

ctp\_twi\_addr = 0x40

ctp\_screen\_max\_x = 1024

ctp\_screen\_max\_y = 600

… ...

ctp\_int\_port = port:PB05<4><default><default><default>

ctp\_wakeup = port:PH01<1><default><default><1>

ctp\_power\_ldo = "axp22\_ldoio1"

ctp\_power\_ldo\_vol = 3000

[ctp\_list\_para]

ctp\_det\_used = 1

… ...

gslX680 = 0

gslX680new = 1

(2) 添加驱动文件

在/lichee/linux-3.4/drivers/input/touchscreen/gslx680new目录下

1> 仿照　"gsl1680e\_p2.h"　编写　"gsl1680new.h"

2> 在文件gslX680.c中添加驱动文件"gslX680new"

struct gslX680\_fw\_array {

const char\* name;

unsigned int size;

const struct fw\_data \*fw;

} gslx680\_fw\_grp[] = {

{"gslX680\_inetd71", ARRAY\_SIZE(GSLX680\_FW\_INETD71),GSLX680\_FW\_INETD71},

{"gsl1680e\_p2" , ARRAY\_SIZE(GSL1680E\_FW\_P2),GSL1680E\_FW\_P2},

{"gslX680new" , ARRAY\_SIZE(GSL1680E\_FW\_P2),GSL1680E\_FW\_P2},　//自定义添加

};

"gslX680new" 用于匹配sys\_config.fex中[ctp\_para]选项的ctp\_name

3> 在android/device/softwinner/xxx/init.rc中添加驱动加载模块

insmod /system/vendor/modules/gslX680new.ko

与/lichee/linux-3.4/drivers/input/touchscreen/gslx680new目录下生成的.ko文件名保持一致

4> 在gslX680\_gesture.c（A83T中是gslX680.c）中做相应添加

… ...

#include "gsl1680new.h"

… ...

struct gslX680\_fw\_array {

const char\* name;

unsigned int size;

const struct fw\_data \*fw;

} gslx680\_fw\_grp[] = {

{"gslX680\_inetd71" , ARRAY\_SIZE(GSLX680\_FW\_INETD71),GSLX680\_FW\_INETD71},

{"gsl1680e\_p2" , ARRAY\_SIZE(GSL1680E\_FW\_P2),GSL1680E\_FW\_P2},

{"gslX680" , ARRAY\_SIZE(GSL1680),GSL1680},

};

unsigned int \*gslX680\_config\_data[16] = {

… ...

gsl\_config\_data\_id,

};

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

调试WIFI ( make installclean )

*注意：按照R16 parrot SDK xr819 驱动移植文档说明*

(1) 修改sys\_config.fex中参数

;-----------------------------------------------------------

;wifi configuration

;wifi\_used: 0-not use, 1- use

;wifi\_sdc\_id: 0- SDC0, 1- SDC1, 2- SDC2, 3- SDC3

;wifi\_usbc\_id: 0- USB0, 1- USB1, 2- USB2

;wifi\_usbc\_type: 1- EHCI(speed 2.0), 2- OHCI(speed 1.0)

;wl\_reg\_on: wifi function enable io

;wl\_host\_wake: wifi device wake-up host

;wl\_host\_wake\_invert: whether wl\_host\_wake use inverter between ap and module

; 0: not used, 1: used

;------------------------------------------------------------

[wifi\_para]

wifi\_used = 1

wifi\_sdc\_id = 1

wifi\_usbc\_id = 1

wifi\_usbc\_type = 1

wl\_reg\_on = port:PL11<1><default><default><0>

wl\_host\_wake = port:PL10<4><default><default><0>

wl\_host\_wake\_invert = 0

wifi\_power = "axp22\_eldo1"

xr\_wl\_power\_vol = 1800000

wifi\_power\_ext1 = "axp22\_dldo1"

xr\_wl\_power\_vol1 = 3300000

wifi\_power\_ext2 = "axp22\_dldo1"

xr\_wl\_power\_vol2 = 3300000

xradio\_lpo\_use\_apclk ="losc\_out"

(2) 添加驱动文件

在lichee/linux-3.4/drivers/net/wireless/xradio/wlan/中

驱动代码wlan\_init.c

在/android/device/softwinner/astar-evb20目录下

**BoardConfig.mk**文件决定android加载哪一款wifi模组

# wifi and bt configuration

# 1. Wifi Configuration

# 1.1 realtek wifi support

# 1.2 broadcom wifi support

# 1.3 eag wifi config

BOARD\_WIFI\_VENDOR := xr\_wlan

# 2. Bluetooth Configuration

**astar\_evb20.mk**文件会在编译 时拷贝指定文件到特定目录下

# wifi features

PRODUCT\_COPY\_FILES += \

hardware/xradio/wlan/kernel-firmware/boot\_xr819.bin:/system/etc/firmware/boot\_xr819.bin \

... ...

device/softwinner/astar-evb20/modules/modules/xradio\_core.ko:obj/xradio\_mac.ko \

... ...

**init.sun8i.rc**是资源和服务配置相关的文件，加载wifi模块时，需开启wpa\_supplicant和p2p\_supplicant

# XRadio WLAN - wpa\_supplicant start

#1.3 xradio wifi sta p2p concurrent service

service wpa\_supplicant /system/bin/wpa\_supplicant -Dnl80211 -iwlan0 ... ...

service p2p\_supplicant /system/bin/wpa\_supplicant -Dnl80211 -ip2p0 ... ...

在.../astar-evb20/overlay/frameworks/base/core/res/res/values/中

config.xml文件，如ap6210等驱动代码支持softap功能，需配置

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

调试camera

GC0328串行通讯口采用标准I2C协议。

串口读出时的设备地址是0x43H;串口写入时的设备地址是0x42H。

(1) 修改sys\_config.fex中参数

根据camera型号及原理图设置reset,power,及standy引脚的gpio,控制逻辑和供电ldo和电压

[csi0]

vip\_csi\_xxx 对应的gpio口

vip\_dev0\_mname, \_pos, \_iovdd, \_avdd, \_dvdd, \_afvdd, \_reset, \_pwdn等参数

(2) android层修改

1> 在/android/device/softwinner/astar-evb20中init.sun8i.rc文件中加入装载驱动模块的语句

#csi module

insmod /system/vendor/modules/videobuf-core.ko

insmod /system/vendor/modules/videobuf-dma-contig.ko

insmod /system/vendor/modules/gc0328.ko

insmod /system/vendor/modules/sunxi\_csi0.ko

2> 在/android/device/softwinner/astar-evb20中ueventd.sun8i.rc文件中改变相关设备节点的权限/

/dev/video0 0666 media media

(3) camera参数配置

1> 在android/device/softwinner/astar-evb20/configs中camera.cfg文件中

number\_of\_camera , camera\_facing , camera\_orientation , camera\_device …

2> 在android/device/softwinner/astar-evb20/configs中media\_profiles.xml文件中

保存camera支持的摄像相关参数，包括摄像质量，音视频编码格式，帧率，比特率等，主要由camera厂商提供

<CamcorderProfiles cameraId="1">

//前置camera为1，后置为0

<EncoderProfile quality="480p" fileFormat="mp4" duration="30">

//正常录制模式

<EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="30">

timelapse480p //为延迟录制模式

(4) gc0328等摄像头驱动文件位置

在/lichee/linux-3.4/drivers/media/video/sunxi-vfe/device中gc0328.c文件

搜vip\_dev0\_twi\_id 使用的I2C通道序号，ctrl+f查找twi注意查看芯片原理图引脚

在R16原理图中，touch panel 使用I2C0，而camera使用I2C2

ctp\_twi\_id = 0 vip\_dev0\_twi\_id = 2

ctp\_twi\_addr = 0x40 vip\_dev0\_twi\_addr = 0x42

;----------------------------------------------------------------------------------

;i2c configuration

; twi\_used = twix enable

;----------------------------------------------------------------------------------

[twi0]

twi\_used = 1

twi\_scl = port:PH02<2><default><default><default>

twi\_sda = port:PH03<2><default><default><default>

[twi1]

twi\_used = 1

twi\_scl = port:PH04<2><default><default><default>

twi\_sda = port:PH05<2><default><default><default>

[twi2]

twi\_used = 1

twi\_scl = port:PE12<3><default><default><default>

twi\_sda = port:PE13<3><default><default><default>

验证GPIO口寄存器的值

R16 Port Register Base Address 0x01c20800

PB Data Register Offset 0x34 7:0 R/W 0

echo 0x01c20834 > /sys/class/sunxi\_dump/dump

cat /sys/class/sunxi\_dump/dump

echo “1111” > /proc/gpio\_extend/gpio\_str //依次将PB02 PB03 PL02 PL03拉高

echo “0000” > /proc/gpio\_extend/gpio\_str //依次将PB02 PB03 PL02 PL03拉低

cat /sys/class/sunxi\_dump/dump

Port Register List Base Address 0x01F02C00

PL Data Register Offset:0x10

调试RF射频模块（VC1SI-4432Y）

针对SPI总线

mosi master output slave input

miso master input slave outpu

在root@astar-evb20:/sys/class/m433 # rx tx

测试程序

1> 在\android\external\m433\目录，Android.mk和m433\_control.c

执行mmm ./生成m433\_dev文件

2> 当输入adb shell后直接把m433\_dev文件拖进来，然后加上目标目录

adb push

Y:\r16\_test\android\out\target\product\astar-evb20\system\bin\m433\_dev /system/bin

如果失败，可以adb remount

3> 进入/system/bin/目录，执行./m433\_dev

Fixme... ...

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

**standby mode待机模式 sleep mode休眠模式**

**待机**是将当前处于运行状态的**数据保存在内存**中，**机器只对内存供电**，而硬盘、屏幕和CPU等部件则停止供电。由于数据存储在速度快的内存中，因此**进入等待状态和唤醒的速度比较快**。不过这些数据是保存在内存中，如果**断电则会使数据丢失**。

**休眠**是将当前处于运行状态的**数据保存在硬盘**中，**整机将完全停止供电**。因为数据存储在硬盘中，而**硬盘速度要比内存低得多**，所以**进入休眠状态和唤醒的速度都相对较慢**，在休眠时**可以完全断开电脑的电源**。

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

**查看Android源码的版本：**

1. 编译的时候在终端中一开始就会打印出来：  
 PLATFORM\_VERSION：2.3.1

2. 直接去make文件中去看：  
 build\core\version\_defaults.mk //搜索该文件中的 PLATFORM\_VERSION值

**关于UART协议**

如果UART只有RX、TX两个信号，要流控的话只能是软流控；如果有RX，TX，CTS，RTS四个信号，则多半是支持硬流控的UART；如果有RX，TX，CTS，RTS，DTR，DSR六个信号的话，RS232标准的可能性比较大。

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

关于git使用

*（替换了源码文件，如何改回原始状态；及在另一分支作业）*

git checkout master

git branch -a

git status

git checkout .

git status

git checkout a33\_xinyuan

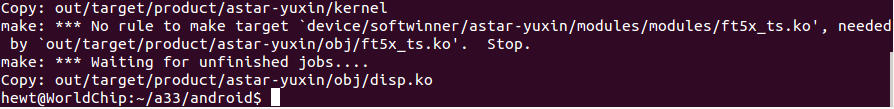
git add –p

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

在linux-3.4目录下，查看.config 即 make menuconfig 的内容

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

**编译android时出现No rule to make target**



在/android/device/softwinner/astar-yuxin目录，打开终端，grep -rns “ft5x\_ts.ko” ./

22

注释掉即可

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

**android固件烧录后，系统已经运行，但是：**

lsusb

33

**1> 解决：minicom无法正常打印信息**

修改 /a33/lichee/tools/pack/chips/sun8iw5p1/configs/default/env.cfg

loglevel=8

**2> 解决：**[**adb连接不上**](http://blog.csdn.net/srw11/article/details/8805760)

修改 /a33/android/device/softwinner/astar-yuxin/astar\_yuxin.mk

... ...

persist.sys.usb.config = mtp,adb \

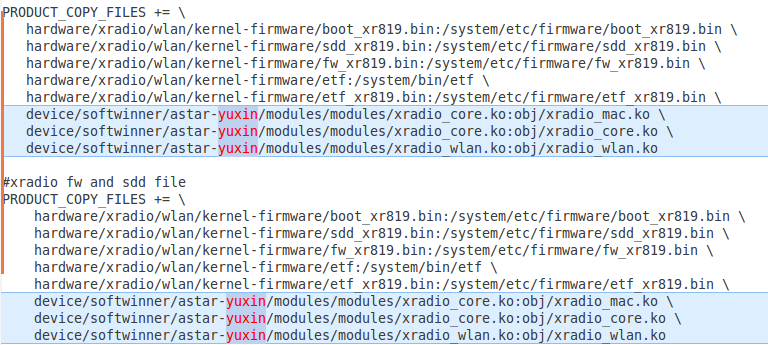
... ...

ro.adb.secure=0

*注意：调试时必须把接通电源（尴尬）*

*：另一种查看minicom信息方法：adb shell 进入后，cat proc/kmsg*

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



2222

在/a33/android/hardware/目录下添加 xradio文件夹

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

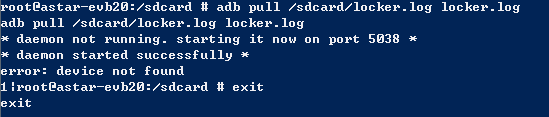
ADB使用的相关命令

Windows下D盘根目录下的file.txt文件拷贝到Android设备的/system/目录下：

**adb push D:\file.txt system/**

将Android设备的/system/目录下的file.txt文件拷贝到Windows下D盘根目录下：

**adb pull system/file.txt D:/**



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

PhoenixSuit无法烧录固件

ADB驱动没有安装好，用驱动精灵或驱动人生先卸载再安装，再卸载再安装；多试几次就OK

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

A83T二合一平板

**Bug1：mipi屏分屏问题总结：**

1. 按照公版的原理图，把每个外设的电源引脚都配上，务必！
2. 关于lcd部分，各个接口延时时间不要太长！

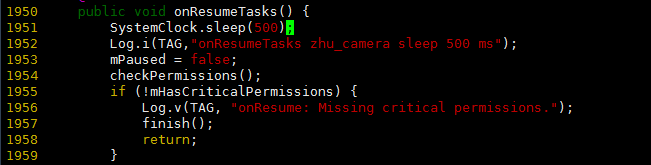
**Bug2：**相机报错（无法连接）

vim packages/apps/Camera2/src/com/android/camera/VideoModule.java

此处删除摄像头连接失败的弹出窗口

git add android/packages/apps/Camera2/src/com/android/camera/CameraActivity.java

此处修改相机应用的打开时间



git commit -m "MIPI分屏已解决，深度休眠后唤醒变灰已解决，CAMERA浅休眠唤醒后已连接，CameraActivity.java延时500ms"

git log 查看提交记录

**Bug3:音频调试**

**喇叭：**改配置参数时每次加1，不要一次加的太多。

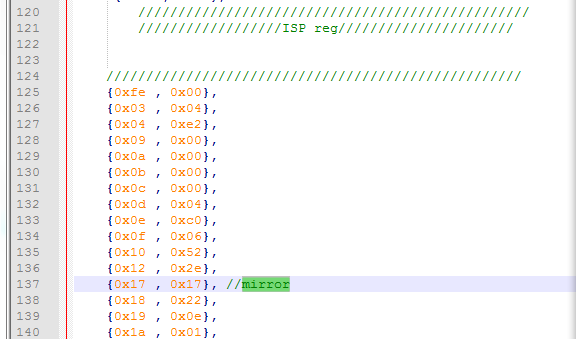
**耳机**：没有声音，可能是linux3-4目录下**.config**文件配置错了，把**下面两个禁掉**（前面**加 #** ）

**在sf\_A83T\lichee\linux-3.4\arch\arm\configs\sun8iw6p1smp\_android\_defconfig文件中**

CONFIG\_SUNXI\_SWITCH\_GPIO=y

CONFIG\_ANDROID\_SWITCH\_GPIO\_TS3A225=y

gc2145.c



**A83T\linux-3.4\include\linux\i2c.h**

#define i2c\_add\_driver(driver) \

i2c\_register\_driver(THIS\_MODULE, driver)

**A83T\linux-3.4\drivers\i2c\i2c-core.c**

int i2c\_register\_driver(struct module \*owner, struct i2c\_driver \*driver)

{

int res;

/\* Can't register until after driver model init \*/

if (unlikely(WARN\_ON(!i2c\_bus\_type.p)))

return -EAGAIN;

/\* add the driver to the list of i2c drivers in the driver core \*/

driver->driver.owner = owner;

driver->driver.bus = &i2c\_bus\_type;

/\* When registration returns, the driver core

\* will have called probe() for all matching-but-unbound devices.

\*/

res = driver\_register(&driver->driver);

if (res)

return res;

/\* Drivers should switch to dev\_pm\_ops instead. \*/

if (driver->suspend)

pr\_warn("i2c-core: driver [%s] using legacy suspend method\n",

driver->driver.name);

if (driver->resume)

pr\_warn("i2c-core: driver [%s] using legacy resume method\n",

driver->driver.name);

pr\_debug("i2c-core: driver [%s] registered\n", driver->driver.name);

INIT\_LIST\_HEAD(&driver->clients);

/\* Walk the adapters that are already present \*/

i2c\_for\_each\_dev(driver, \_\_process\_new\_driver);

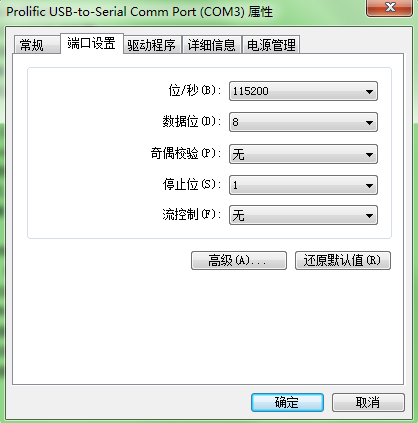
return 0;

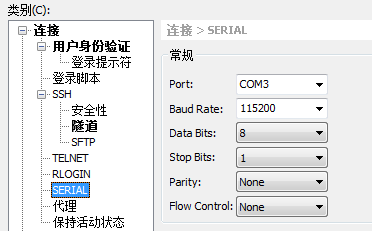
}

EXPORT\_SYMBOL(i2c\_register\_driver);

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

**关于在windows中使用X-shell软件，打开串口调试功能**



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

R16死机后抓取串口信息

**R16改io口**

ctp\_int\_port = port:PB05<4><default><default><default>

增加 休眠时双击唤醒

**PB05 ----> PL09**

1G = 1024M , 1M = 1024KBytes, 1KB = 1024 Byte

1G = 1 \* 1024 \* 1024 \* 1024Byte

**机器死机时，如何抓取LOG：**

执行adb shell登到设备，

**logcat -v time -f /sdcard/locker.log \*:W &**

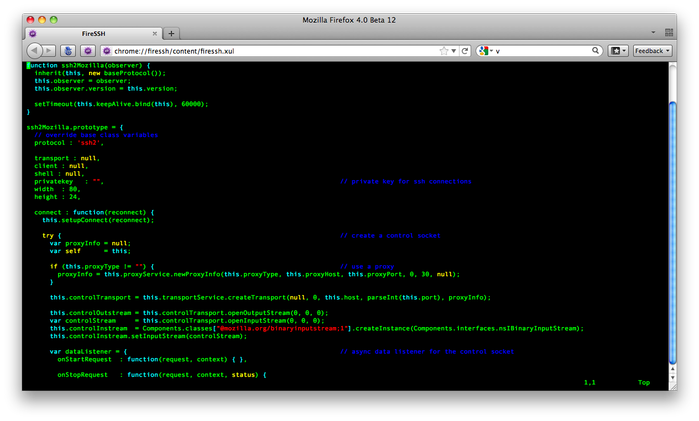
adb shell

cat /proc/kmsg > C:\Users\Administrator\Desktop1.txt

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

FireSSH 详细介绍

FireSSH 是一个免费跨平台支持的在Firefox浏览器上实现 SSH 远程终端客户的的插件，完全采用 JavaScript 编写。只需要在地址栏输入 **ssh://username@IP**地址 即可连接目标服务器。



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

Android编译时，报JACK端口错误：

android M引入了Jack (Java Android Compiler Kit),在编译时会启动jack server，优化编译速度等。

如果遇到jack编译错误，可以把$HOME/.jack中的配置文件中的SERVER=false设成false，不用jack，jack现在还不稳定。此外，M对应的是OpenJDK7

root@zhangyd-VirtualBox:~# cat .jack

Server settings

SERVER=false

SERVER\_PORT\_SERVICE=8072

SERVER\_PORT\_ADMIN=8073

SERVER\_COUNT=1

SERVER\_NB\_COMPILE=4

SERVER\_TIMEOUT=60

或者 更改 service 和 admin 的值

$HOME/.jack

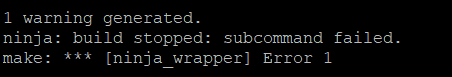
$HOME/.jack-server/config.properties

$HOME/.jack-settings

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

**易方项目**

安卓系统编译不过，报如下错误，是由于git的原因



修改**android/external/iw/version.sh**，禁掉

android\device\softwinner\tulip-p3\tulip\_p3.mk

对应 system\build.prop

android\device\softwinner\tulip-p3\configs\camera.cfg

对应 system\etc\camera.cfg

android\device\softwinner\tulip-p3\configs\gsensor.cfg

对应 system\usr\gsensor.cfg

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

Makefile

注释行的结尾如果存在反斜线（\），那么下一行也被作为注释行。

如果需要注释多行，在注释行的结尾加行反斜线(\),下一行也被注释，可以注释多行。

make: \*\*\* No rule to make target `system/etc/bluetooth/rtkbt.conf', needed by `device/softwinner/octopus-f1/bluetooth/rtkbt.conf'. Stop.

make: \*\*\* Waiting for unfinished jobs....

【总结】

Makefile在编译执行过程中，对于所依赖的条件，此处即一堆xml文件，最后一个是reference.xml，结果由于最后reference.xml后面，有多余的4个空格，导致Makefile将该处的4个空格，视为一个文件了，所以，必然找不到该“文件”，所以才报错的。

所以，如果Makefile出现：

make: \*\*\* No rule to make target ` ‘, needed by xxx. Stop.

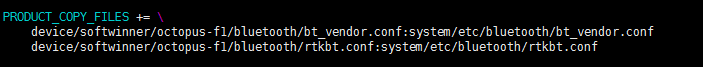
的错误，那么**基本上都是属于找不到所依赖的文件所导致**的，所以应该去：

**检测确保，所依赖的文件，是否真实存在**。

很可能，很常见的一个现象就是，此处的，误写了多余的空格，导致被视为依赖文件，导致找不到，导致报此错误。

解决办法就很简单，去掉多余的空格即可

在android/device/softwinner/octopus-f1/octopus\_f1.mk文件中



第一行末尾，缺少一个斜线 \

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

A64-D210项目

sys\_config.fex中测试DDR频率每次减少24的倍数，

A64 7.1版本 要先编译uboot-2014；并把lcd的驱动文件拷贝到u-boot的目录

参考文档**\lichee\brandy\SUNXI\_README** ，有详细说明

**./build.sh -p sun50iw1p1**

注：（内核里面不能用printk ，只能用printf）

A64 7.0编译android出现

make: \*\*\* [out/build-aosp\_arm.ninja] Error 1

安卓平板抓系统LOG：

**adb logcat -v time > C:\log\_email.txt**

**input keyevent POWER 软件模拟 按power键**

**Wifi打不开怎么搞？**

- aldo2\_vol = 1001800

+aldo2\_vol = 1003300

**AXP电源管理lpddr3的1.8v供电常开**

**目录lichee/linux-3.10/drivers/power/axp\_power\_legacy/axp81x/axp81x-regu.c**

[axp81x\_vcc\_ldo10] = {

{

.constraints = {

.name = "axp81x\_eldo2",

.min\_uV = 700000,

.max\_uV = 1900000,

**.always\_on = 1,**

.valid\_ops\_mask = REGULATOR\_CHANGE\_VOLTAGE | EGULATOR\_CHANGE\_STATUS,

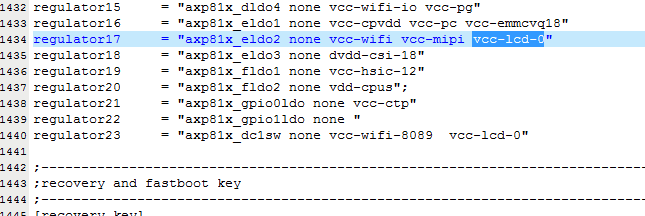
},

},

.info = &axp81x\_regulator\_info[axp81x\_vcc\_ldo10],

},

闪线条严重：打开“停用HW叠加层”选项，就OK！



**eldo2里面这个去掉vcc-lcd-0**

**配置不能随便配**

**echo N /sys/module/printk/parameters/**

s83t\_pci

- <feature name="android.hardware.sensor.compass" />

+ <!-- <feature name="android.hardware.sensor.compass" /> -->

root@octopus-f1:/sys/bus/iio/devices/iio:device0 # cat rawdata\_gyro

**在\lichee\linux-3.10\include\linux\power\aw\_pm.h 是小CPU唤醒配置**

/\* define cpus wakeup src \*/

#define CPUS\_MEM\_WAKEUP (CPUS\_WAKEUP\_LOWBATT |\

CPUS\_WAKEUP\_USB |\

CPUS\_WAKEUP\_AC |\

CPUS\_WAKEUP\_DESCEND |\

CPUS\_WAKEUP\_ASCEND |\

CPUS\_WAKEUP\_ALM0 |\

CPUS\_WAKEUP\_GPIO |\

CPUS\_WAKEUP\_IR)

#define CPUS\_BOOTFAST\_WAKEUP (CPUS\_WAKEUP\_LOWBATT |\

CPUS\_WAKEUP\_LONG\_KEY |\

CPUS\_WAKEUP\_ALM0 |\

CPUS\_WAKEUP\_USB |\

CPUS\_WAKEUP\_AC)

去掉 ro.sf.rotation=180

**MAC地址显示不可用**

adb shell后，ifconfig，getprop

**tulip-p3:/ # ifconfig**

ifconfig

wlan0 Link encap:Ethernet HWaddr 50:36:21:4c:16:02

inet addr:192.168.1.102 Bcast:192.168.1.255 Mask:255.255.255.0

inet6 addr: fe80::5236:21ff:fe4c:1602/64 Scope: Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:64 errors:0 dropped:0 overruns:0 frame:0

TX packets:12 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:21029 TX bytes:1812

p2p0 Link encap:Ethernet HWaddr 52:36:21:4c:16:02

UP BROADCAST MULTICAST MTU:1500 Metric:1

RX packets:0 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:0 TX bytes:0

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

inet6 addr: ::1/128 Scope: Host

UP LOOPBACK RUNNING MTU:65536 Metric:1

RX packets:0 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:0 TX bytes:0

**tulip-p3:/ # getprop**

getprop

… …

[wifi.interface]: [wlan0]

[wifi.supplicant\_scan\_interval]: [15]

[wifi.use\_customized\_macaddr]: [1]

[wlan.driver.status]: [ok]

[zygote.mount\_fs\_data\_done]: [1]

tulip-p3:/ #

**接HDMI输出后闪线条**

dram的参数使用如下的参数

[dram\_para]

dram\_clk = 576

dram\_type = 7

dram\_zq = 0x3B3BFB

dram\_odt\_en = 0x31

dram\_para1 = 0x10E210E2

dram\_para2 = 0x1001

dram\_mr0 = 0x1840

dram\_mr1 = 0x40

dram\_mr2 = 0x18

dram\_mr3 = 0x2

dram\_tpr0 = 0x0048A258

dram\_tpr1 = 0x01B1B28F

dram\_tpr2 = 0x00047028

dram\_tpr3 = 0x050005dc

dram\_tpr4 = 0x0

dram\_tpr5 = 0x0

dram\_tpr6 = 0x0

dram\_tpr7 = 0x2a066198

dram\_tpr8 = 0x0

dram\_tpr9 = 0x0

dram\_tpr10 = 0x4

dram\_tpr11 = 0x20000000

dram\_tpr12 = 0x00000000

dram\_tpr13 = 0x04000100

A83T-PCI项目

























**陀螺仪无法加载，不能上报数据：**

[ 145.383482] mpu6500: module is from the staging directory, the quality is unknown, you have been warned.

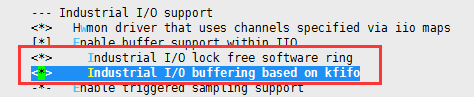
[ 145.394961] mpu6500: Unknown symbol iio\_kfifo\_free (err 0)

[ 145.402368] mpu6500: Unknown symbol iio\_kfifo\_allocate (err 0)

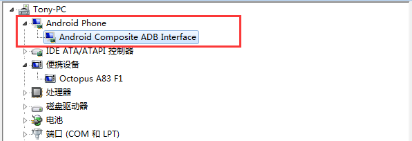
[ 145.402057] CPU1: Booted secondary processor

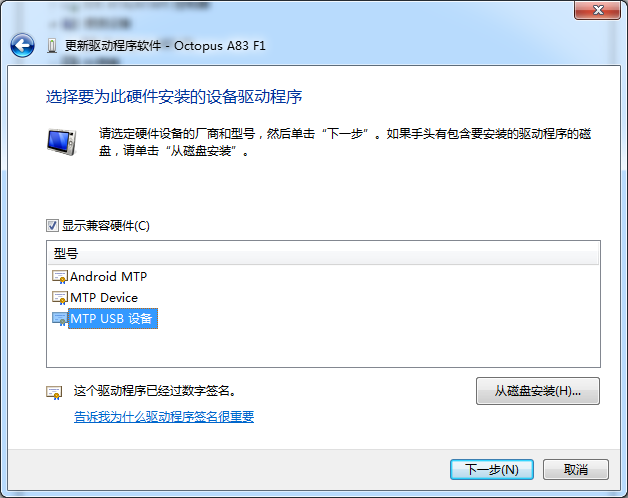
[ 146.922576] CPU1: shutdown

**如下两项要编译进内核：**



无法烧录固件，识别不到





**选择 MTP USB设备**

海蜘蛛路由



密码：168yx\_-@

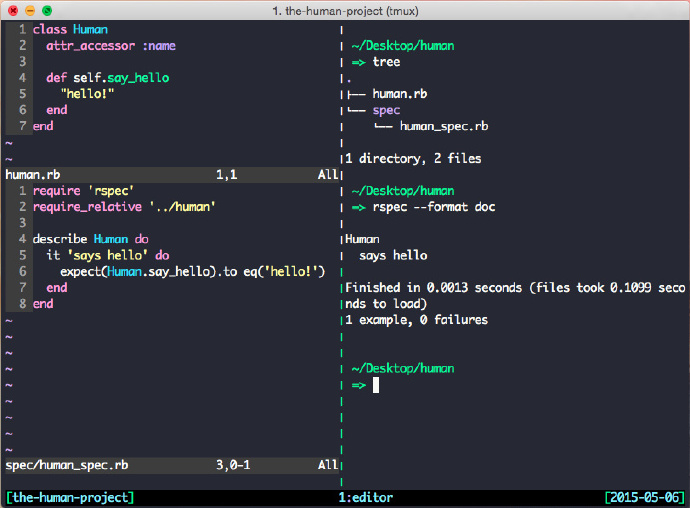
Tmux使用

## Tmux 中最棒的功能

提示：下面这截图也许与你使用 Tmux 时看到的界面略有不同。这是因为我修改了 Tmux 的状态栏配置，如果你也想修改成和截图中一样的效果，那么可以参照“美化 Tmux 的状态栏”这一节中的步骤。

### 窗格

我认为沿竖直方向分割屏幕是个不错的主意，这样我就可以在一边使用 Vim，而在另一边查看代码运行结果，如果需要的话，有时我还会再打开一个控制台。下面我就要讲解如何利用 Tmux 实现这一切。



从图中可以看出：

左侧：Vim（左上方是一个 Ruby 的类文件，左下方是针对这类编写的测试文件）

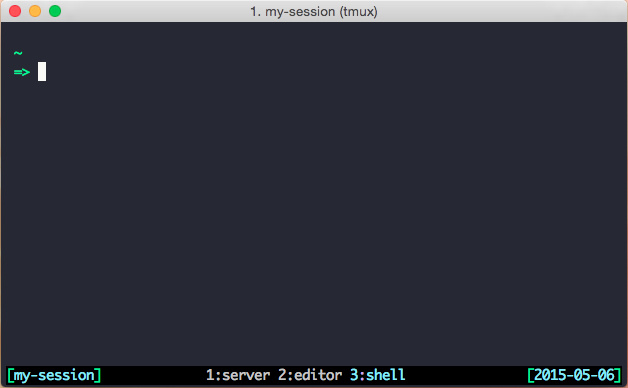
右侧：一个 Bash 的会话

要创建一个竖直放置的窗格很容易，**待开启了一个 Tmux 会话之后，只需再按下 Ctrl-b % ，一个竖直窗格就出现了**。另外，**若要把屏幕沿水平方向分割，则只需要按下 Ctrl-b "**。在 Tmux 的窗格间移动光标也很简单，只需要先按下 Tmux 的快捷键前缀，然后再按下对应的方向键就可以让光标进入到目标窗格了。

### 窗口

在Tmux中，窗口是个窗格容器，你可以将多个窗格放置在窗口中，并根据你的实际需要在窗口中排列多个窗格，也是完全取决于你的需要。例如，我经常是这样做，先开启一个叫作“server”的窗口用于运行应用程序的服务器（在这个窗口中可以看到服务器的日志），然后开启另一个叫作“editor”的窗口用于编写代码。在这个窗口中有两个窗格，一个用于 Vim，一个用于运行测试代码。最后再开启一个叫作“shell”的窗口用于通过 Bash shell 运行命令。Tmux 的窗口功能非常实用，因为在一个窗口中可以创建出多个窗格，这样在一个窗口中就能同时查看所有窗格内容，通过这种方法可以高效地利用有限的屏幕空间。

在 Tmux 的会话中，现有的窗口将会列在屏幕下方。下图所示的就是在默认情况下 Tmux 列出现有窗口的方式。这里一共有三个窗口，分别是“server”、“editor”和“shell”。



若要创建一个窗口，只需要按下Ctrl-b c；若要切换窗口，只需要先按下Ctrl-b，然后再按下想切换的窗口所对应的数字，该数字会紧挨着窗口的名字显示。

### 会话

一个 Tmux 会话中可以包含多个窗口。会话功能非常简单易用，例如可以为一个特定的项目创建一个专用的 Tmux 会话。若要创建一个新的会话，只需要在终端运行如下的命令：

**tmux new -s <name-of-my-session>**

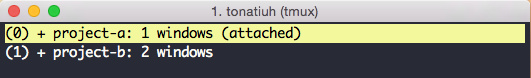
假设我还需要开发另一个项目，于是我就会为此再新建一个会话。虽然进入了新的会话，但是原来的会话并没有消失。所以我可以在稍后回到之前的会话继续工作。若要创建一个新的会话，只需要按下 Ctrl-b : ，然后输入如下的命令：

**new -s <name-of-my-new-session>**

除非显式地关闭会话，否则 Tmux 的会话在重启计算机之前都不会消失。只要还没有重启计算机，你都可以自由地从一个项目的会话跳转到另一个。

### 在 Tmux 的会话间切换

若要获取现有会话的列表，可以按下Ctrl-b s。下图所示的就是会话的列表：



列表中的每个会话都有一个 ID，该 ID 是从 0 开始的。按下对应的 ID 就可以进入会话。如果你已经创建了一个或多个会话，但是还没有运行 Tmux，那么可以输入如下命令以接入已开启的会话。

tmux attach

df命令 磁盘管理

**df命令**用于显示磁盘分区上的可使用的磁盘空间。默认显示单位为KB。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

**语法**

df(选项)(参数)

**选项**

-a或--all：包含全部的文件系统；

--block-size=<区块大小>：以指定的区块大小来显示区块数目；

-h或--human-readable：以可读性较高的方式来显示信息；

-H或--si：与-h参数相同，但在计算时是以1000 Bytes为换算单位而非1024 Bytes； -i或--inodes：显示inode的信息；

-k或--kilobytes：指定区块大小为1024字节；

-l或--local：仅显示本地端的文件系统；

-m或--megabytes：指定区块大小为1048576字节；

--no-sync：在取得磁盘使用信息前，不要执行sync指令，此为预设值；

-P或--portability：使用POSIX的输出格式；

--sync：在取得磁盘使用信息前，先执行sync指令；

-t<文件系统类型>或--type=<文件系统类型>：仅显示指定文件系统类型的磁盘信息；

-T或--print-type：显示文件系统的类型；

-x<文件系统类型>或--exclude-type=<文件系统类型>：不要显示指定文件系统类型的磁盘信息；

--help：显示帮助；

--version：显示版本信息。

**参数**

文件：指定文件系统上的文件。

**实例**

**查看系统磁盘设备，默认是KB为单位：**

[root@LinServ-1 ~]# **df**

文件系统 1K-块 已用 可用 已用% 挂载点

/dev/sda2 146294492 28244432 110498708 21% /

/dev/sda1 1019208 62360 904240 7% /boot

tmpfs 1032204 0 1032204 0% /dev/shm

/dev/sdb1 2884284108 218826068 2518944764 8% /data1

**使用-h选项以KB以上的单位来显示，可读性高：**

[root@LinServ-1 ~]# **df -h**

文件系统 容量 已用 可用 已用% 挂载点

/dev/sda2 140G 27G 106G 21% /

/dev/sda1 996M 61M 884M 7% /boot

tmpfs 1009M 0 1009M 0% /dev/shm

/dev/sdb1 2.7T 209G 2.4T 8% /data1

**查看全部文件系统：**

[root@LinServ-1 ~]# **df -a**

文件系统 1K-块 已用 可用 已用% 挂载点

/dev/sda2 146294492 28244432 110498708 21% /

proc 0 0 0 - /proc

sysfs 0 0 0 - /sys

devpts 0 0 0 - /dev/pts

/dev/sda1 1019208 62360 904240 7% /boot

tmpfs 1032204 0 1032204 0% /dev/shm

/dev/sdb1 2884284108 218826068 2518944764 8% /data1

none 0 0 0 - /proc/sys/fs/binfmt\_misc

来自: <http://man.linuxde.net/df>

getprop

**scp –r zhuqg@192****.168.2.228:/home/zhuqg/workspace/T3 .**

**scp /home/guozh/\*.tar.gz hewt@192.168.6.129:/hewt/swap/**

sudo cp –r sources.list /etc/apt/

Ubuntu关机重启命令简介

重启命令：  
1、reboot  
2、shutdown -r now 立刻重启(root用户使用)  
3、shutdown -r 10 过10分钟自动重启(root用户使用)  
4、shutdown -r 20:35 在时间为20:35时候重启(root用户使用)  
如果是通过shutdown命令设置重启的话，可以用shutdown -c命令取消重启

关机命令：  
1、halt   立刻关机  
2、poweroff 立刻关机  
3、shutdown -h now 立刻关机(root用户使用)  
4、shutdown -h 10 10分钟后自动关机

如果是通过shutdown命令设置关机的话，可以用shutdown -c命令取消重启

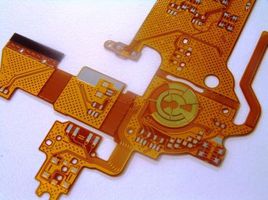
考试一

**一、LCD屏物理组成有哪些？**

LCD （液晶显示器 Liquid Crystal Display ） IC

背光（LED发光二极管作为液晶显示屏的背光源） FPC（柔性电路板Flexible Printed Circuit ）

**二、TP物理组成有哪些？**

不同材质的盖板（玻璃、塑料）、sensor、fpc

**三、LCD的调试过程**

（1）首先配置menuconfig，lcd的相关代码在disp驱动模块中，进入内核目录，执行make ARCH=arm(64) menuconfig 进行配置，Device Drivers -> Graphics support -> Support for frame buffer devices -> Video Support for sunxi -> DISP Driver Support(sunxi-disp2)

（2）其次配置sys\_config.fex

（3）添加新屏驱动支持

1.在源码目录下添加新的屏驱动，已default\_panel.c为模板，新建一个屏驱动，修改开关屏的流程，以及屏驱动的名称，名字须与ysy\_config中的lcd\_driver\_name对应。

2.在panel.c中，将新的屏驱动加到panel\_array中。

3.修改Makefile文件。

4.在sys\_config中，配置屏参数，lcd0，接口，时序等

5.在sys\_config中，配置屏的相关pin脚，power，pwm等。

6.编译，加载disp.ko，屏亮！

**四、C语言基础**

五、T3开发板

1GB DDR3 和16GB EMMC

LCD屏组成：

偏光板 Polarizer

彩色滤光片 CF (colours filter)

液晶盒 LC sell (liquid crystal cell)

TFT 薄膜晶体管thin film transistor

偏光板 Polarizer

灯管

导光管 Light Guide

PCBA 印刷电路板

LCD屏接口：

RGB： Red Green Blue

LVDS： Low Voltage Differential Signaling

MIPI： Mobile Industry Processor Interface

EDP： embedded Display Port

TP分类：

G+G 触控玻璃 + 钢化玻璃盖板

G+P 触控玻璃 + PET塑料盖板

G+F 玻璃 + 膜

OGS One glass solution

DDR分类：

DDR2、DDR3L、DDR4 / LPDDR2、LPDDR3

EMMC闪存分类：

SLC 32Gb(4GB) Single-Level Cell

MLC 64Gb(8GB) Multi-Level Cell

TLC 96Gb(12GB) Trinary-Level Cell

全志的CPU，90%是PBGA封装；

考试二

1、LCD屏的接口有哪些？

HV（SYNC+DE）、8080、TTL、LVDS、MIPI、EDP

2、调试WIFI模块硬件方面需要具备什么条件？

WIFI打不开：

首先检查模组是否有虚焊或连锡；

其次用万用表测量电压是否达到3.3V，且PMU供电电流至少大于400mA；

WIFI能打开，无法连接设备：

检查天线匹配，再检查外部晶振和上拉电阻是否正确；

WIFI休眠唤醒时重连：

检查电源控制PIN，将控制PIN接至主控在休眠不断电的GPIO；

3、Gsensor调试

首先，在sys\_config.fex中配置I2C总线号、地址和中断引脚；

其次，在init.rc中添加驱动；

最后，gsensor.cfg中调x、y、z轴方向：

首先调Z轴，当旋转超过90度才旋转，说明Z轴反了；确定Z轴

**确认硬件正确性：**

（1）确认各引脚与HOST正确连接；

（2）电源电压（VCC=3.3V，GND=0V）

（3）I2C引脚电平是否正常

（4）复位电路是否正常，WAKE\_UP引脚电压

（5）INT引脚电压是否正常

（6）确认I2C地址（设备可设置多个地址时）

**调试：**

1. 确保硬件连接正确
2. 上电，测试各引脚电压正常
3. 打开串口打印等级：通过adb shell进入目录

/proc/sys/kernel输入echo 8 4 1 7>printk

1. 查看probe是否成功
2. Probe成功之后，旋转机器，看界面是否旋转

使用adb shell getevent命令查看是否有数据上报

a有数据上报：

有明显变化：查看HAL层是否接收到数据

无明显变化：确认数据是否正常

b无数据上报:

进入/sys/class/input/inputX，X为设备编号，查看enable是否为1，若为0，将其设置为1，再getevent查看是否有数据

若有，说明HAL层无法使能打开，

4、WIFI调试

软件调试：在adb shell输入lsmod，确认bcmdhd.ko、8723bs.ko等驱动是否已加载；

5、C语言的基础知识

Static全局变量和普通全局变量的区别：static全局变量只被初始化一次，防止在其他文件单元中被引用；

即作用域不同：static全局变量只在定义了该变量的源文件中有效，在同一个源程序的其他源文件不能使用；而普通全局变量的作用域是整个源程序，当一个程序由多个源文件组成时，普通全局变量在每个源文件中都是有效的。

Static局部变量和普通局部变量的区别：static局部变量只被初始化一次，下一次被调用时依据上一次的结果值；

Static函数和普通函数的区别：static函数在内存中只有一份，普通函数在每一次被调用中维持一份拷贝；

程序的局部变量存在堆栈区，全局变量存在静态区，动态申请的数据存在堆中。

请找出下面代码中的所有错误。说明：以下代码是把一个字符串倒序，如“abcd”倒序后变为“dcba”。

int main()

{

    char \*src = "hello,world";

    int len = strlen(src);

    char \*dest = (char \*)malloc(len + 1); //要为\0分配一个空间

    char \*d = dest;

    char \*s = &src[len - 1]; //指向最后一个字符

    while ( len-- != 0 )

        \*d++ = \*s--;

    \*d = 0; //尾部要加\0

    printf("%s\n", dest);

    free(dest);// 使用完，应当释放空间，以免造成内存汇泄露

    return 0;

}

A83T\_PCI升级到android7.1系统

**无法使用adb shell**

在octopus\_n106.mk中打开USB调试

# usb

PRODUCT\_PROPERTY\_OVERRIDES += \

**persist.sys.usb.config=mtp,adb \**

**ro.adb.secure=0**

**更改显示大小**

PRODUCT\_PROPERTY\_OVERRIDES += \

ro.property.tabletUI=false \

ro.sf.lcd\_density=160

**更改显示方向**

1：在octopus\_n106.mk文件中增加以下属性：

persist.demo.hdmirotationlock=false \

persist.demo.hdmirotation=portrait \

ro.sf.rotation=90 \

2：还要修改HDMI的显示方向，android/frameworks/base/services目录下

--- a/services/core/java/com/android/server/display/LocalDisplayAdapter.java

+++ b/services/core/java/com/android/server/display/LocalDisplayAdapter.java

@@ -408,7 +408,7 @@ final class LocalDisplayAdapter extends DisplayAdapter {

// For demonstration purposes, allow rotation of the external display.

// In the future we might allow the user to configure this directly.

if ("portrait".equals(SystemProperties.get("persist.demo.hdmirotation"))) {

- mInfo.rotation = Surface.ROTATION\_270;

+ mInfo.rotation = Surface.ROTATION\_90;

}

**修改安卓系统“设置”中各选项的属性**

在 frameworks\base\packages\SettingsProvider\res\values\defaults.xml

**设置中开发者选项没有了**

和这个APK有关android/packages/apps/Provision.apk

**系统卡，修改CMA值位置：**

lichee\tools\pack\chips\sun8iw6p1\configs\default\env.cfg

cma=176M改为cma=200M

**开机慢，修改**

loglevel=8 改为loglevel=4

Repo

Echo和cat读取写入寄存器值

怎么用adb读取pmic6329寄存器的值

在mt6575上使用mt6329，可以通过如下方式控制pmic 寄存器。

1：进入adb shell

2：通过cd命令进入/sys/devices/platform/mt-pmic

3：读取操作，比如读取bank0的0x17这个寄存器

输入**echo 17>pmic\_access\_bank0**来读取寄存器的值

输入**cat pmic\_access\_bank0**来显示寄存器的值

4：写操作，比如向bank0的0x17这个寄存器写入0x5a

输入**echo 17 5a>pmic\_access\_bank0**

注意：

1：用cat显示的值是十进制的，而输入的寄存器的ID和输入的值都是十六进制的。

也就是说如果写入0b（十六进制的值），读出来是11（十进制的值）

2：每次读取都要输入echo和cat两条命令。

3：在/sys/devices/platform/mt-pmic目录下用ls命令，可以看到很多LDO的标志。

4：使用cat LDO\_VSIM\_VOLTAGE可以看到此LDO软件设置的输出电压。

5：使用cat LDO\_VSIM\_STATUS可以看到此LDO软件设置的输出电压。

uranus-t1:/sys/class/sunxi\_dump # **echo 0x0300B024 > dump**

echo 0x0300B024 > dump

uranus-t1:/sys/class/sunxi\_dump # **cat dump**

cat dump

**0x70671711**

uranus-t1:/sys/class/sunxi\_dump # **echo 0x0300B024 0x10671711 > write**

echo 0x0300B024 0x10671711 > write

uranus-t1:/sys/class/sunxi\_dump # **cat dump**

cat dump

**0x10671711**

A64云知声

调试触摸按键：

AW9163\_ts\_6Button.c文件打开**AW9163\_create\_sysfs(client);**

**cd sys/bus/i2c/drivers/AW9163\_ts/1-002c**

**cat rawdata**

**cat delta**

**echo 0x13 0x1515 > getreg**

**echo 0x14 0x1515 > getreg**

**echo 0x15 0x1515 > getreg**

A64\_6.0本亚门禁

调试摄像头黑白切换：

struct ctp\_config\_info{

enum input\_sensor\_type input\_type;

int ctp\_used;

\_\_u32 twi\_id;

const char \* name;

int screen\_max\_x;

int screen\_max\_y;

int revert\_x\_flag;

int revert\_y\_flag;

int exchange\_x\_y\_flag;

u32 int\_number;

unsigned char device\_detect;

const char \*ctp\_power;

u32 ctp\_power\_vol;

struct gpio\_config ctp\_power\_io;

struct regulator \*ctp\_power\_ldo;

struct gpio\_config irq\_gpio;

struct gpio\_config wakeup\_gpio;

#ifdef TOUCH\_KEY\_LIGHT\_SUPPORT

struct gpio\_config key\_light\_gpio;

#endif

struct device \*dev;

struct pinctrl \*pinctrl;

};

A63\_7.1开发板

**串口**里面使用：

**busybox vi /etc/camera.cfg**

在linux3.4内核中 #include<mach/sys\_config.h> 对应

在linux3.10内核中 #include<linux/sys\_config.h>

**光感、距感**

MG5691调试需要**打开使能**，getevent才会有数据上报

uranus-t1:/vendor/modules # insmod mg5691.ko //手动加载KO

uranus-t1:/sys/devices/platform/elan\_alsps # ls //进到目录

uranus-t1:/sys/devices/platform/elan\_alsps # echo 1 > als\_enable //打开使能

uranus-t1:/sys/devices/platform/elan\_alsps # echo 1 > ps\_enable //打开使能

**陀螺仪**

MPU6500打开节点

uranus-t1:/sys/bus/iio/devices/iio:

TP

**开机时喇叭有KAKA的声音**，可能开机声音输出超出喇叭的承受能力，软件应调小一格尝试，

**调试副屏显示时，备份文件，编译出错：**

修改SYSTEMUI相关代码时，备份不能使用 –copy文件，编译会出错，如：

packages/SettingsProvider/res/values/defaults.xml

packages/SettingsProvider/res/values/defaults **- copy**.xml

packages/SystemUI/AndroidManifest.xml

packages/SystemUI/src/com/android/systemui/SystemUIService.java

**尽量使用.bak备份**

**Android属性设置**

问题：

发现adb sehll setProp所设属性值在下次重起后被清除

adb shell setprop testing.mediascanner.skiplist /storage/sdcard1/test

结论：

必须采用persist.开头的属性名才能永久保存。

设置属性

ro.开头的属性将不能被更改属性值，persist.开头的属性会被永久纪录，其他属性值在重新开机后均将被丢弃

getprop setprop [key] [value] watchprops adb shell dumpsys window displays

文件

system/core/init/property\_service.c