

君正@
Linux X1000开发手册

Date: Sep.28 2015



北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

君正@Linux X1000开发手册

Release history

Date	Revision	Revision History
Sep.28, 2015	1.0	- First released

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Ingenic Headquarters, East Bldg. 14, Courtyard #10, Xibeiwang East Road, Haidian Dist., Beijing, China, 100193

Tel: 86-10-56345000

Fax: 86-10-56345001

Http: //www.ingenic.cn

1. 前言	1
1.1. 文档目的及开发背景.....	1
2. U-BOOT 的开发和使用	2
2.1. 配置环境变量.....	2
2.1.1. 修改引导参数.....	2
2.1.2. 修改 DDR 频率.....	2
2.1.3. 修改 IP 地址.....	2
2.1.4. 搭建网络文件系统.....	2
2.2. 编译 U-boot	3
3. LINUX 内核和驱动	3
3.1. GPIO.....	3
3.1.1. 配置文件.....	3
3.1.2. 使用方法.....	4
3.2. I2C	4
3.3. SPI nand	4
3.3.1. 分区划分.....	4
3.3.2. 分区变更.....	5
3.4. AUDIO.....	5
3.4.1. u-boot&kernel 驱动配置使用方法	6
3.4.1.1. u-boot 配置方法.....	6
3.4.1.2. kernel 配置方法.....	6
3.4.1.3. Audio 驱动功能验证方法及流程.....	8
3.5. TF Card.....	10
3.5.1. 板级配置.....	10
3.5.2. TF card 驱动程序.....	11
3.5.3. SD 卡挂载	12
3.6. USB.....	13
3.6.1. Host.....	13
3.6.1.1. Host 功能配置方法	13
3.6.1.2. USB功能验证.....	14
3.6.2. USB Device.....	15
3.6.2.1. USB Device 功能配置方法.....	15
3.6.2.2. USB Device 验证方法.....	15
3.6.2.3. USB Device 功能验证分析.....	15
3.7. LCD.....	16
3.7.1. 板级注册.....	16
3.7.2. drivers 文件描述.....	16
3.7.3. 驱动配置方法.....	16
3.8. Camera	18
3.8.1. 板级配置文件.....	18
3.8.2. drivers 文件描述.....	18
3.8.3. camera 驱动配置方法	18

3.8.3.1. cim 控制器配置	18
3.8.3.2. VPU 配置	20
3.8.3.3. 串口配置	21
3.8.4. 使用方法	21
3.9. USB Camera	21
3.9.1. USB Camera驱动配置方法	21
3.9.2. USB camera 验证、使用方法	22
3.10. 休眠唤醒	22
3.10.1. 休眠唤醒配置、使用方法	22
3.10.2. 休眠唤醒验证方法	22
3.11. Voice trigger	23
3.11.1. Voice trigger 简介	23
3.11.2. Voice trigger 驱动配置方法	23
3.11.3. 验证方法	24
3.12. AES-RSA	24
3.12.1. 驱动名称及路径	24
3.12.2. 驱动配置	25
3.12.3. IOCTL 命令定义	25
3.12.4. 驱动结构体描述	26
3.12.5. USER API	27
4. LINUX 根文件系统	28
4.1. 制作文件系统	28
4.1.1. Jffs2 文件系统	28
4.1.2. Ubi文件系统	28
4.2. 扩展文件系统	28
4.2.1. Jffs2 文件系统	29
4.2.2. Ubi文件系统	29
5. OTA	29
5.1. 环境准备	30
5.1.1. 烧录工具分区配置	30
5.1.2. 制作升级镜像	30
5.2. 编译	31
5.3. NV_RW 分区 bin 文件的制作	31
5.4. 升级包的制作	31
5.5. 升级	32
6. 测试用例	33
6.1. Camera 测试	33
6.2. USB Camera 测试	33
6.3. Wi-Fi 连接测试	34
6.4. Bluez 测试	36
7. 常见问题及解决办法	37

7.1. Ubuntu 下 Oracle VM VirtualBox 虚拟机烧录问题.....	37
7.1.1. 问题现象.....	37
7.1.2. 解决办法.....	38
7.2. Ubuntu 下 adb 问题	40
7.2.1. 问题现象.....	40
7.2.2. 解决办法.....	41
8. 源码编译	41
8.1. 源码目录结构.....	41
8.2. 整体编译	41
8.3. 部分编译	42

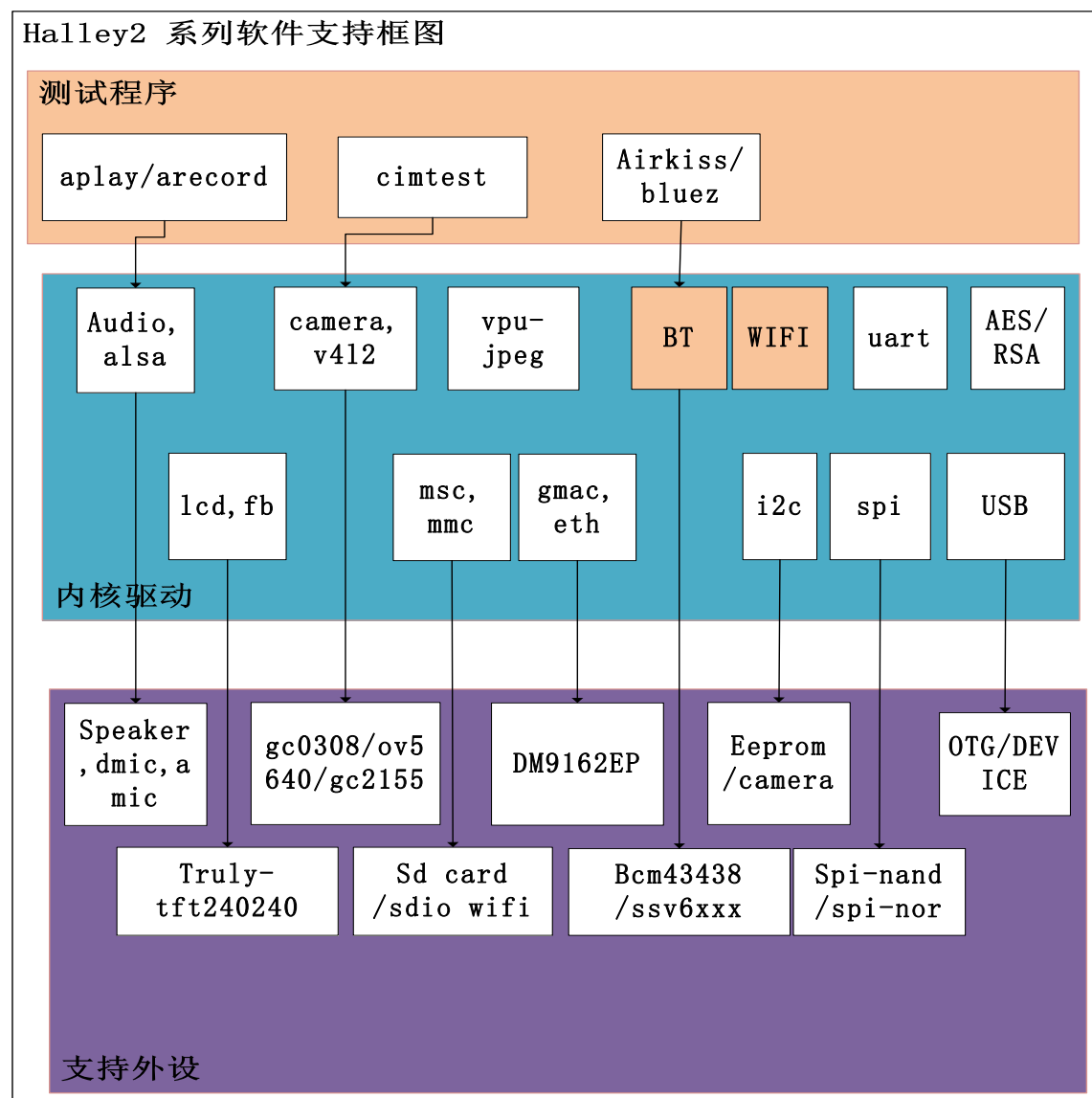
1. 前言

1.1. 文档目的及开发背景

君正处理器是高集成度、高性能和低功耗的 32位 RISC 处理器，带有 MMU 和数据及指令 Cache，以及丰富的外围设备，可以运行 Linux 操作系统。本文将向读者介绍基于君正处理器平台进行Linux 内核的配置方法和开发过程,引导开发人员快速进行 Linux 开发。包括引导程序u-boot的配置，Linux3.10内核和驱动，文件系统的制作及拓展，OTA的配置及应用开发等。

(注：此文档为halley2系列开发板的文档，halley2系列开发板主要包括halley2， halley2_mini，存储设备分SPI-nor 和SPI-nand版本。)

以下为：halley2 系列软件支持框图



2. U-Boot 的开发和使用

Linux 内核需要 U-Boot 来引导。U-Boot 是为嵌入式平台提供的开放源代码的引导程序, 它提供串行口、以太网等多种下载方式, 提供 NOR 和 NAND 闪存和环境变量管理等功能, 支持网络协议栈、JFFS2/EXT2/FAT 文件系统, 同时还支持多种设备驱动如 MMC/SD 卡、USB 设备、LCD 驱动等。

SDK 源码目录结构中, 在 halley2/platform/u-boot 目录下有 u-boot 源码。平台中的所有设置都已配置完成可直接使用, 下面列出几个主要配置用户可根据自己的实际需求自行修改。

2.1. 配置环境变量

进入u-boot中后, 在include/configs/halley2.h文件中进行对u-boot的简单配置。

2.1.1. 修改引导参数

主要修改文件系统的类型和所在的分区位置分别有rootfstype和root指定, 以及Linux内核的启动地址0x80800000。

```
#define CONFIG_BOOTARGS    BOOTARGS_COMMON    "ip=off    init=/linuxrc\nrootfstype=jffs2 root=/dev/mtdblock2 rw"\n#define CONFIG_BOOTCOMMAND "sfcnor read 0x40000 0x300000 0x80800000 ;bootm\n0x80800000"
```

2.1.2. 修改 DDR 频率

```
#define CONFIG_SYS_MPLL_FREQ    600000000    /*If MPLL not use mast be set 0*/\n#define CONFIG_SYS_MEM_FREQ    200000000
```

NOTE: 通过CONFIG_SYS_MEM_FREQ修改DDR频率时, 修改值必须为主频MPLL值CONFIG_SYS_MPLL_FREQ的整数倍, 建议主频不要超过200M。

2.1.3. 修改 IP 地址

```
#define CONFIG_SERVERIP    192.168.4.13\n#define CONFIG_IPADDR    192.168.4.90\n#define CONFIG_GATEWAYIP    192.168.4.1\n#define CONFIG_NETMASK    255.255.255.0\n#define CONFIG_ETHADDR    00:11:22:33:44:55
```

2.1.4. 搭建网络文件系统

搭建网络文件系统主要修改u-boot的引导参数 BOOTARGS_COMMON, 具体修改可参考如下配

置(其中user为自定义用户名)：

```
#define CONFIG_BOOTARGS
BOOTARGS_COMMON "ip=192.168.4.254:192.168.4.1:192.168.4.1:255.255.255.0
rootdelay=2 nfsroot=192.168.4.13:/home/fpga/user/rootfs rw"
```

NOTE: `nfsroot=192.168.4.13:/home/fpga/user/rootfs` 为NFS网络文件系统的共享目录。

2.2. 编译 U-boot

修改完所需配置项后即可编译u-boot

1. 清除上次编译

```
$ make distclean
```

2. 选择板级支持配置

在boards.cfg文件中以列出了当前u-boot所支持的所有配置，根据开发板是spi-nor版本还是spi-nand版本进行编译：

编译nor版本 (Halley2_mini_v2.0(SPI-nor)开发板, Halley2_v2.0(SPI-nor)开发板)

```
$ make halley2_v10_uImage_sfc_nor
```

编译nand版本 (尚未发布nand的开发板)

```
$ make halley2_v10_uImage_spi_nand
```

3. LINUX 内核和驱动

SDK源码目录结构中，在halley2/platform/kernel目录下有kernel源码。

3. LINUX 内核和驱动

3.1. GPIO

3.1.1. 配置文件

1.GPIO管理的基本思想

GPIO所有PIN脚，在芯片焊接到电路板那一刻，其功能就已经固定，哪些用于设备IO、哪些用作真正的GPIO，由板级层在一开始就完全确定，驱动只需要操作真正的GPIO。

2.接口函数

由文件arch/mips/xburst/soc-x1000/common/gpio.c中实现。在GPIOLIB框架下实现基础接口，由GPIOLIB提供调用接口，接口定义头文件为<include/linux/gpio.h>，提供接口有：

函数接口	功能
gpio_is_valid	判断GPIO号是否有效，有效才进行申请操作
gpio_request	申请GPIO
gpio_free	释放GPIO
gpio_direction_input	设置GPIO为输入
gpio_direction_output	设置GPIO为输出

gpio_get_value	读取GPIO值，GPIO为输入或者输出
gpio_set_value	设置GPIO输出值，前提GPIO配置为输出
gpio_to_irq	通过GPIO获取中断号

3.1.2. 使用方法

- 1) 判断得到的gpio号是否有效，然后申请gpio。
- 2) 根据使用配置为输入或者输出。（默认为输入）

使用过程中可以读取管脚电平或配置输出电平（gpio_get/set_value）。

- 3) 用作中断源。通过gpio_to_irq得到中断号，再申请注册该中断ISR。
 - a) 注册中断时最好指定触发类别。（默认为下降沿）
 - b) 中断使用中可以通过set_irq_type改变中断触发方式。
- 4) 最后（驱动移除）释放gpio。

参考例子:

drivers/input/keyboard/gpio_keys.c

3.2. I2C

1.添加设备信息

添加设备信息放在arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/i2c_bus.c文件中。

2.注册从设备

在目录'arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/'中的board_base.c文件中的board_base_init函数会调用i2c_register_board_info将设备注册成i2c从设备。在启动kernel时，看内核打印，是否成功读取id号，如果读取成功则注册成功，否则没有。

3.3. SPI nand

根据开发板上的存储介质类型，选择相应的驱动方式。这里主要是SPI nand的使用说明，包括配置项的选择、分区的修改以及烧录工具的更新。

Spi nand文件系统

文件系统为UBI。

(注意：目前发布开发板的flash中没有nand,只是nor)

3.3.1. 分区划分

在SDK源码中提供的u-boot和kernel已经对SPI nand分区进行了默认配置，分区划分如下：

Nand分区划分：

分区	u-boot	kernel	Rootfs (ubi)	Data (ubi)
----	--------	--------	--------------	------------

	1M	8M	40M	所剩余空间
--	----	----	-----	-------

I 整体编译

目前发布的SDK中已对spi nand做了可整体编译的支持，若需使用spi nand需按如下操作进行整体编译。

在“halley2/platform/development/device/device.mk”文件中，选择MAKE_SPI_NAND=y，在“halley2/platform”目录下执行以下命令进行整体编译，编译完成后即可在“halley2/platform/out/target/product/halley2/image”目录下生成支持SPI nand包括u-boot，kernel，文件系统在内的所有镜像文件。

```
$make
```

注：对于SPI

Nand的存储介质，暂不支持OTA升级。因此MAKE_SPI_NAND和MAKE_OTA不能同时选中。

II 部分编译

U-boot: halley2_v10_uImage_spi_nand

编译: \$ make halley2_v10_uImage_spi_nand

Kernel: halley2_nand_v10_linux_defconfig

编译: \$ make halley2_nand_v10_linux_defconfig

\$ make uImage

3.3.2. 分区变更

在对分区重新进行划分时，需要同时修改u-boot和kernel中的分区信息，二者的分区信息必须一致。并须更新烧录工具固件重新烧录u-boot、kernel、文件系统。

1. 修改u-boot分区信息

文件位置: platform/u-boot/drivers/mtd/nand/jz_spinand.c

修改结构体: struct jz_spinand_partition

2. 修改kernel分区信息

文件位置: platform/kernel/arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/spi_bus.c

修改结构体: struct mtd_partition

3. 更新烧录工具

如果nand的分区信息发生变化后，必须要在使用烧录工具烧录前，更新烧录工具的firmware。

1) 编译烧录工具firmware

进入u-boot

```
$ make burner_x1000_lpddr
```

2) 替换烧录工具firmware

用u-boot下生成的u-boot.bin替换烧录工具下“firmware/x1000”下的u-boot.bin。

3. 4. AUDIO

Audio是一个音频模块，使用alsa实现了DMIC和AIC的功能。

3.4.1. u-boot&kernel 驱动配置使用方法

3.4.1.1. u-boot 配置方法

u-boot使用默认配置。

3.4.1.2. kernel 配置方法

A) 板级注册

(1) audio设备在板级的如下文件中被定义: arch/mips/xburst/soc-x1000/common/platform.c

(2) audio设备板级名称: jz-asoc-aic-dma, jz-asoc-aic, icdc-d3, jz-asoc-dmic-dma, jz-asoc-aic-dmic, jz-asoc-pcm-dma, jz-asoc-pcm, spdif dump, pcm dump, dmic dump

B) 驱动代码

```
sound/soc/ingenic/icodec/icdc_d3.c
sound/soc/ingenic/icodec/pcm_dump.c
sound/soc/ingenic/icodec/dmic_dump.c
sound/soc/ingenic/asoc-v13/
sound/soc/ingenic/asoc-v12/asoc-aic-v12.c
sound/soc/ingenic/asoc-board/phoenix_icdc.c
```

C) audio驱动的使用, 需将menuconfig里面的如下的几个选项进行选择使能

(1) 使能AIC

Device Drivers

- > Sound card support
 - > Advanced Linux Sound Architecture
 - > ALSA for SoC audio support
 - > ASoC support for Ingenic
 - > jz board type select(Audio support for phoenix with internal codec)

(2) 使能DMIC

Device Drivers

- > Sound card support
 - > Advanced Linux Sound Architecture
 - > ALSA for SoC audio support
 - > ASoC support for Ingenic
 - >Support DMIC for record

```
jz board type select
nu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes,
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
able

[*] soc x1000 codec type select (Audio support for phoenix with internal codec) --->
[*] Support DMIC for record
```

NOTE: 我们的芯片上有两个DMIC控制器，一个集成在内部codec上不需要单独配置，另一个独立在芯片上需要单独配置。两个控制器不能同时使用。

Audio DMA选择:

1. Device Drivers

- >Sound card support
 - >Advanced Linux Sound Architecture
 - >ALSA for SoC audio support
 - >ASoC support for Ingenic
 - >JZ audio dma clear auto dirty memory

```
ASoC support for Ingenic
menu. <Enter> selects submenus --->. Highlighted letters are
s features. Press <Esc><Esc> to exit, <?> for Help, </> for Se
apable

--- ASoC support for Ingenic
jz board type select --->
[*] JZ audio dma clear auto dirty memory
```

2. Device Drivers

- >DMA Engine support
 - >XBURST DMA V13 support

```

DMA Engine support
the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
arizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Leg
le capable

--- DMA Engine support
[ ] DMA Engine debugging
*** DMA Devices ***
< > Synopsys DesignWare AHB DMA support
<*> XBURST DMA V13 support
< > XBURST DMA V12 support
< > Timberdale FPGA DMA support
*** DMA Clients ***
[ ] Network: TCP receive copy offload
[ ] Async_tx: Offload support for the async_tx api
< > DMA Test client

```

3.4.1.3. Audio 驱动功能验证方法及流程

以下（1）（2）（3）所列出的audio功能验证方法均是在串口端输入命令进行验证。

（1）AIC功能验证

录音前设置(选择使用AMIC):

```
amixer cset numid=17,iface=MIXER,name='ADC Mux' 0
```

模拟麦克风内部codec录音:

```

amixer cset numid=4,iface=MIXER,name='Digital Capture Volume'
20 amixer cset numid=6,iface=MIXER,name='Mic Volume' 3
arecord -D hw:0,0 -c 2 -f S16_LE -r 44100 -d 10 a.wav

```

我们的模拟mic只支持单通道录音。由于不同的Mic物理特性不同，在录音之前需要调整mic的两级录音增益到合适的值，以防止数据丢失。

查看录音结果，按如下命令进行播放之前的录音文件:

```
aplay a.wav
```

注：测试所有工具amixer，arecord，aplay的使用说明详见本小结结尾处。

（2）DMIC功能验证

DMIC录音:

```
arecord -D hw:0,2 -c 2 -f S16_LE -r 8000 -d 10 b.wav
```

Dmic 支持的采样率为8000 和 16000。

（3）混响

1. ADC数据叠加上DAC通路的数据流(录音时将放音通路的数据做为背景):

```

amixer cset numid=6,iface=MIXER,name='Mic Volume' 2
amixer cset numid=4,iface=MIXER,name='Digital Capture Volume' 30
amixer cset numid=17,iface=MIXER,name='ADC Mux' 0
amixer cset numid=12,iface=MIXER,name='ADC Mode Mux' 1
amixer cset numid=11,iface=MIXER,name='AIADC Mux' 2
amixer cset numid=10,iface=MIXER,name='ADC MIXER Mux' 2
amixer cset numid=9,iface=MIXER,name='mixer Enable' 1

```

2. DAC数据叠加上ADC通路的数据流(放音时将录音通路的数据做为背景):

```
amixer cset numid=6,iface=MIXER,name='Mic Volume' 4
amixer cset numid=16,iface=MIXER,name='DAC_MERCURY VMux' 0
amixer cset numid=15,iface=MIXER,name='MERCURY AIDAC MIXER Mux' 2
amixer cset numid=14,iface=MIXER,name='DAC Mode Mux' 1
amixer cset numid=13,iface=MIXER,name='MERCURY AIDAC Mux' 2
amixer cset numid=9,iface=MIXER,name='mixer Enable' 1
```

录音播放: arecord -D hw:0,0 -c 2 -f S16_LE -r 8000 -d 10 a.wav&
aplay a.wav &

NOTE:

(1)使用混响功能需要对alsa子系统及内部codec通路有一定了解。

(2)

在使用混音功能时以上1,2两种功能要保证播放和录音通路有数据才能得到比较明显的效果。即aplay和arecord同时执行。另外混响功能必须使用内部codec录音。

(3) 在使用dmic录音时不使用hpfl的时候, 需要减小增益。

* audio 测试工具说明:

1. amixer

amixer

是alsa在用户态抽象出的控件, 用于配置底层硬件, 用户根据自己的需要进行配置, 以下命令的执行是在串口端输入的。

◆输入以下命令获取控件列表:

amixer controls

◆输入以下命令获取controls 的具体状态:

amixer cget controls

◆输入以下命令设置相应controls为value:

amixer cset controls value

◆codec mic的音量

numid=6,iface=MIXER,name='Mic Volume'

◆高通滤波

numid=7,iface=MIXER,name='ADC High Pass Filter Switch'

◆混音方式选择

numid=10,iface=MIXER,name='ADC MIXER Mux'

◆录音模式选择

numid=12,iface=MIXER,name='ADC Mode Mux'

◆模拟mic数字mic选择

numid=17,iface=MIXER,name='ADC Mux'

◆播放音量

numid=3,iface=MIXER,name='Playback Mixer Volume'

◆混音模式

numid=11,iface=MIXER,name='AIADC Mux'

◆播放方式选择

numid=14,iface=MIXER,name='DAC Mode Mux'

◆ 播放混音选择

numid=16,iface=MIXER,name='DAC_MERCURY VMux'

◆ 暂未支持

numid=16,iface=MIXER,name='DAC_TITANIUM VMux'

◆ 数字播放混音音量

numid=5,iface=MIXER,name='Digital Capture Mixer Volume'

◆ 数字录音音量

numid=4,iface=MIXER,name='Digital Capture Volume'

◆ 数字播放静音

numid=8,iface=MIXER,name='Digital Playback mute'

◆ 播放混音方式

numid=15,iface=MIXER,name='MERCURY AIDAC MIXER Mux'

numid=13,iface=MIXER,name='MERCURY AIDAC Mux'

◆ 播放音量

numid=1,iface=MIXER,name='MERCURY Playback Volume'

◆ 不需设置

numid=2,iface=MIXER,name='TITANIUM Playback Volume'

◆ 混音使能

numid=9,iface=MIXER,name='mixer Enable'

2. arecord录音

-D参数用于指定音频设备PCM

hw的第一个参数用来指定声卡号，第二个参数用于指定设备号

-c 用于指定声道数

-f用于指定数据格式

-r用于指定采样频率

-d用于指定录音时间

--help 获取帮助

3. aplay放音

参数设置与arecord一致。

3.5. TF Card

3.5.1. 板级配置

1. 板级配置文件：

arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/mmc.c

其中：

struct jzmmc_platform_data tf_pdata描述sd卡设备。

struct jzmmc_platform_data sdio_pdata 描述sdio类设备，一般指sdio-wifi。

2. 电路连接:

sd卡接到MSC0，sdio-wifi接到MSC1。

3.5.2. TF card 驱动程序

1. 驱动程序文件:

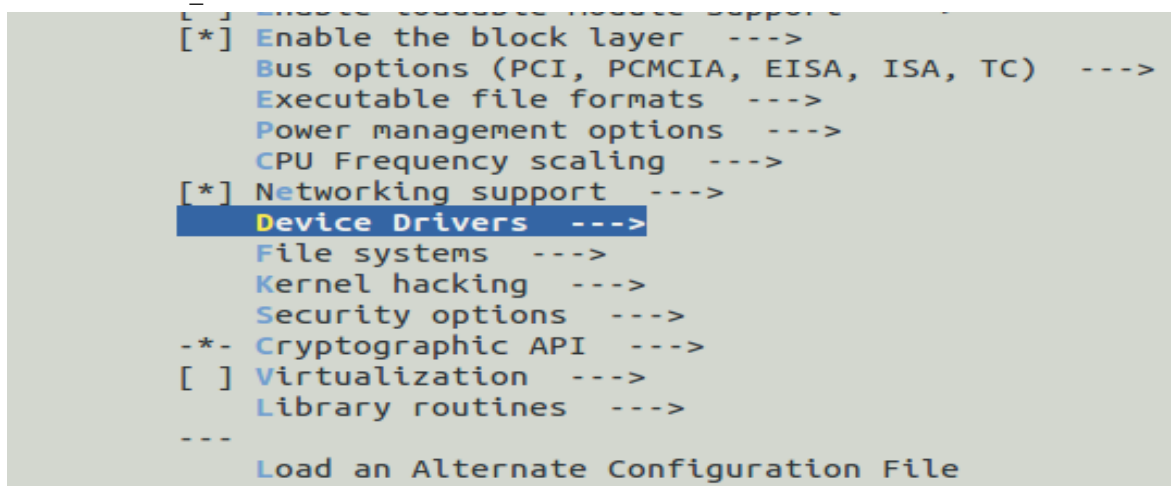
drivers/mmc/host/jzmmc_v12.c

2. MMC驱动编译选项配置如下:

Device Drivers

->MMC/SD/SDIO card support

->JZMMC_V12 MMC0



```
[*] Enable the block layer --->
    Bus options (PCI, PCMCIA, EISA, ISA, TC) --->
    Executable file formats --->
    Power management options --->
    CPU Frequency scaling --->
[*] Networking support --->
    Device Drivers --->
        File systems --->
        Kernel hacking --->
        Security options --->
    *- Cryptographic API --->
    [ ] Virtualization --->
        Library routines --->
---
Load an Alternate Configuration File
```



```

Graphics support --->
<*> Sound card support --->
[ ] HID Devices --->
[*] USB support --->
<*> MMC/SD/SDIO card support --->
< > Sony MemoryStick card support (EXPERIMENTAL) --->
[ ] LED Support --->
[ ] Near Field Communication (NFC) devices --->
v(+)

<Select>    < Exit >    < Help >

```

```

JZMMC_V12 GPIO function pins select (GPIO A, data with 4 bit) --->
(24000000) msc0 max frequency
[ ] Use PIO mode for MSC0
[*] JZMMC_V12 MMC1
JZMMC_V12 GPIO function pins select (GPIO C, Data width 4 bit) --->
(24000000) MSC1 max frequency
[ ] Use PIO mode for MSC1
[ ] JZMMC_V12 MMC2
< > Secure Digital Host Controller Interface support
< > VUB300 USB to SDIO/SD/MMC Host Controller support
< > USB SD Host Controller (USHC) support

```

3.5.3. SD 卡挂载

文件系统默认支持了SD卡的自动mount功能，SD卡插上后默认被mount到文件系统的/mnt/sd目录下，输入以下命令查看是否自动挂载：

```
$mount
```

```

rootfs on / type rootfs (rw)
/dev/root on / type jffs2 (rw,relatime)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
tmpfs on /dev type tmpfs (rw,relatime)
tmpfs on /tmp type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
tmpfs on /dev/shm type tmpfs (rw,relatime)
/dev/mmcblk0p1 on /mnt/sd type vfat (rw,relatime,fmask=0000,dmask=0000,allow_utime=0022,codepage=cp437,ioccharset=iso8859-1,shortname=mixed,e)

```

出现最后一行“/dev/mmcblk0p1 on /mnt/sd type vfat”说明挂载成功。

备注：自动挂载的脚本默认将TF卡挂载在第一个分区，如果自动挂载不成功请查看TF卡分区信息，手动挂载相应的分区。

注意：目前发布的halley2 开发板不支持SD卡热插拔，需要手动操作以下步骤：

- 1.插入SD卡后，从新启动开发板
- 2.执行 `ls dev/mmcblk0` 查看是否有SD 卡设备，若有继续执行下面的步骤，若无，重复第一步。
- 3.执行 `mount /dev/mmcblk0p* /mnt` ， 挂在你的SD 到/mnt 下 ，注意 /dev/mmcblk0p* 为你的SD卡分区，需视个人开发板具体情况而定。

3. 6. USB

A) 板级注册文件

平台设备注册文件: arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/ board_base.c

在“arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/misc.c”中申请userusb 的dete pin, ID pin , bus pin

B) 驱动路径

控制器驱动路径: drivers/usb/dwc2/

C) USB驱动配置

USB 主要分为host 和device 两种功能, 两种功能均需在menuconfig中配置使用, 下面以otg为例:

OTG配置方法:

Device Drivers

->USB support (USB_SUPPORT [=y])

->Support for Host-side USB (USB [=y])

->USB routine power management (autosuspend) and wakeup (USB_SUSPEND [=y])

```
< > USB Printer support
< > R8A66597 HCD support
< > Host Wire Adapter (HWA) driver (EXPERIMENTAL)
< > Inventra Highspeed Dual Role Controller (TI, ADI, ...)
<*> DesignWare Core USB 2.0 Hi-Speed On-The-Go(OTG)
    Driver Mode (Both Host and Device) --->
    [ ] Allow use dwc2 drvbus function pin
    [*] Allow wakeup when usb cable plug/unplug
    [ ] Board has no plug detect facility
```

如果需要otg 同时作为device 和host 需要选上如下选项:

3. 6. 1. Host

3. 6. 1. 1. Host 功能配置方法

以U盘为例:

(1)Device Drivers

->USB support

```
< > USB Printer support
< > USB Wireless Device Management support
< > USB Test and Measurement Class support
    *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may
    *** also be needed; see USB_STORAGE Help for more info *
<*> USB Mass Storage support
    [ ] USB Mass Storage verbose debug
< > Realtek Card Reader support
```

->USB Mass storage support

(2)Device Drivers

->SCSI device

->SCSI disk support

由于USB Host功能需要上层驱动的支持，因此需要选上SCSI 驱动支持。

Device Drivers

->SCSI device support

3. 6. 1. 2. USB 功能验证

1.

文件系统默认支持了USB热插拔，插入U盘后，U盘会默认被mount到文件系统/mnt/sda目录下，插入USB设备后串口打印输出如下：

```
[ 22.092290] jz-dwc2 jz-dwc2: set vbus on(on) for host mode
[ 22.202174] USB connect
[ 22.902209] usb 1-1: new high speed USB device number 2 using dwc2
[ 23.328262] usb 1-1: New USB device found, idVendor=0930, idProduct=6545
[ 23.342090] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 23.356745] usb 1-1: Product: DataTraveler 2.0
[ 23.365858] usb 1-1: Manufacturer: Kingston
[ 23.374570] usb 1-1: SerialNumber: C86000BDBA09EF60CA285106
[ 23.412410] scsi0 : usb-storage 1-1:1.0
[ 24.475824] scsi 0:0:0:0: Direct-Access Kingston DataTraveler 2.0 PMAP PQ: 0 ANSI: 4
[ 25.751099] sd 0:0:0:0: [sda] 30497664 512-byte logical blocks: (15.6 GB/14.5 GiB)
[ 25.768695] sd 0:0:0:0: [sda] Write Protect is off
[ 25.779135] sd 0:0:0:0: [sda] No Caching mode page present
[ 25.790501] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 25.807171] sd 0:0:0:0: [sda] No Caching mode page present
[ 25.832513] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 25.879083] sda:sda1
[ 25.895075] sd 0:0:0:0: [sda] No Caching mode page present
[ 25.932372] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 25.964595] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

2. 在串口端输入以下命令查看是否自动挂载：

\$mount

```
# mount
rootfs on / type rootfs (rw)
/dev/root on / type jffs2 (rw,relatime)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
tmpfs on /dev type tmpfs (rw,relatime)
tmpfs on /tmp type tmpfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
tmpfs on /dev/shm type tmpfs (rw,relatime)
/dev/sda1 on /mnt/sda type vfat (rw,relatime,fmask=0000,dmask=0000,allow_utime=0022,codepage=cp437,iocharset=iso8859-1,showmount=0)
#
```

出现最后一行“/dev/sda1 on /mnt/sda type vfat”则代表已经自动挂载。

注意：目前发布的halley2开发板文件系统暂不支持自动挂载，需要执行一下步骤进行手动挂载

1. 执行ls dev/sda 查看是否存在设备文件，若存在，则需要从新插入
2. 执行mount dev/sda1 /mnt 挂载u 盘到/mnt目录下

3.6.2. USB Device

3.6.2.1. USB Device 功能配置方法

Device Drivers

->USB support

->USB Gadget support

```
< > 10 warrior driver support
< > USB testing driver
< > iSight firmware loading support
< > USB YUREX driver support
< * > USB Gadget Support --->
      *** OTG and related infrastructure ***
[ ] Hold a wakelock when USB connected

USB Gadget Support
[ ] Debugging information files (DEVELOPMENT)
[ ] Debugging information files in debugfs (DEVELOPMENT)
(2) Maximum VBUS Power usage (2-500 mA)
USB Peripheral Controller (DesignWare USB 2.0 Hi-Speed OTG Controller) --->
< * > USB Gadget Drivers (Android Gadget) --->
      Android Gadget
[ ] Android usb-ccid-serial transparent transmission function support
[ ] Android rawbulk framework function support

->USB Gadget Drivers (Android Gadget)
```

目前gadget 功能使用 android gadget ， android gadget默认支持 mass storage, adb功能具体操作方法以 adb 为例子。

3.6.2.2. USB Device 验证方法

在串口输入:

```
$cd /sys/class/android_usb/android0
$echo 0 > enable
$echo 18d1 > idVendor
$echo d002 > idProduct
$echo mass_storage,adb > functions
$echo 1 > enable
```

在pc端输入:

```
$adb devices
```

3.6.2.3. USB Device 功能验证分析

PC端执行如下命令，查看usb 设备:

```
$
adb
devices
user@user-desktop:~$ adb devices
List of devices attached
Ingenic device
user@user-desktop:~$
```

如图PC端adb 发现”Ingenic device”说明USB Device功能正常。

3.7. LCD

3.7.1. 板级注册

LCD

板级注册文件路径为： /arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/lcd/lcd-truly_tft240240_2_e.c

3.7.2. drivers 文件描述

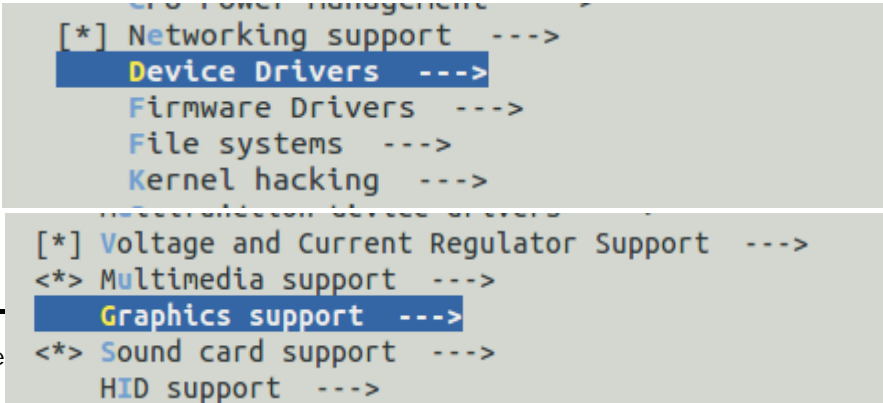
文件描述	路径
驱动代码	halley2/platform/kernel/drivers/video/jz_fb_v13/jz_fb.c
电源管理代码	halley2/platform/kernel/drivers/video/backlight/truly_tft240240_2_e.c
背光代码	halley2/platform/kernel/drivers/video/backlight/pwm_bl.c

3.7.3. 驱动配置方法

Menuconfig中需配置，选择如下几个选项，LCD方可正常使用：

1.Device Drivers

- >Graphics support
- > Support for frame buffer devices



```
< > Lowlevel video output switch controls
<*> Support for frame buffer devices --->
[ ] Exynos Video driver support --->
[*] Backlight & LCD device support --->
```

2.Device Drivers

-->Graphics support

--><*>Backlight & LCD device support

--> Lowlevel LCD controls

SLCD TRULY TFT240240-2-E with control IC st7789s (240x240)

SLCDC USE TE SIGNAL

SLCDC CONTINUA TRANSFER

Lowlevel Backlight controls

Generic PWM based Backlight Driver V13

```
<*> Support for frame buffer devices --->
[ ] Exynos Video driver support --->
[*] Backlight & LCD device support --->
    Console display driver support --->
[ ] Bootup logo --->
<*> JZ LCDC framebuffer V1.3 --->
```

-- Backlight & LCD device support

```
<*> Lowlevel LCD controls
< > Platform LCD controls
[ ] LCD panel reset enable
< > AT070TN93 LCD Driver
< > AU0_A043FL01V2 LCD Driver
< > KD50G2_40NM_A2 panel(800x480)
< > BYD BM8766U panel(800x480)
```

```
< > KFM701A21_1A TFT Smart LCD panel(400x240)
<*> SLCD TRULY TFT240240-2-E with control IC st7789s (240x240)
<*> SLCDC USE TE SIGNAL
< > USE GPIO SIMULATE TO MCU INIT
<*> SLCDC CONTINUA TRANSFER
[ ] KD301_M03545_0317A panel(320x480)
[ ] TM035PDH03 panel(320x480)
<*> Lowlevel Backlight controls
< > Generic (aka Sharp Corgi) Backlight Driver
< > Generic PWM based Backlight Driver
<*> Generic PWM based Backlight Driver V13
```

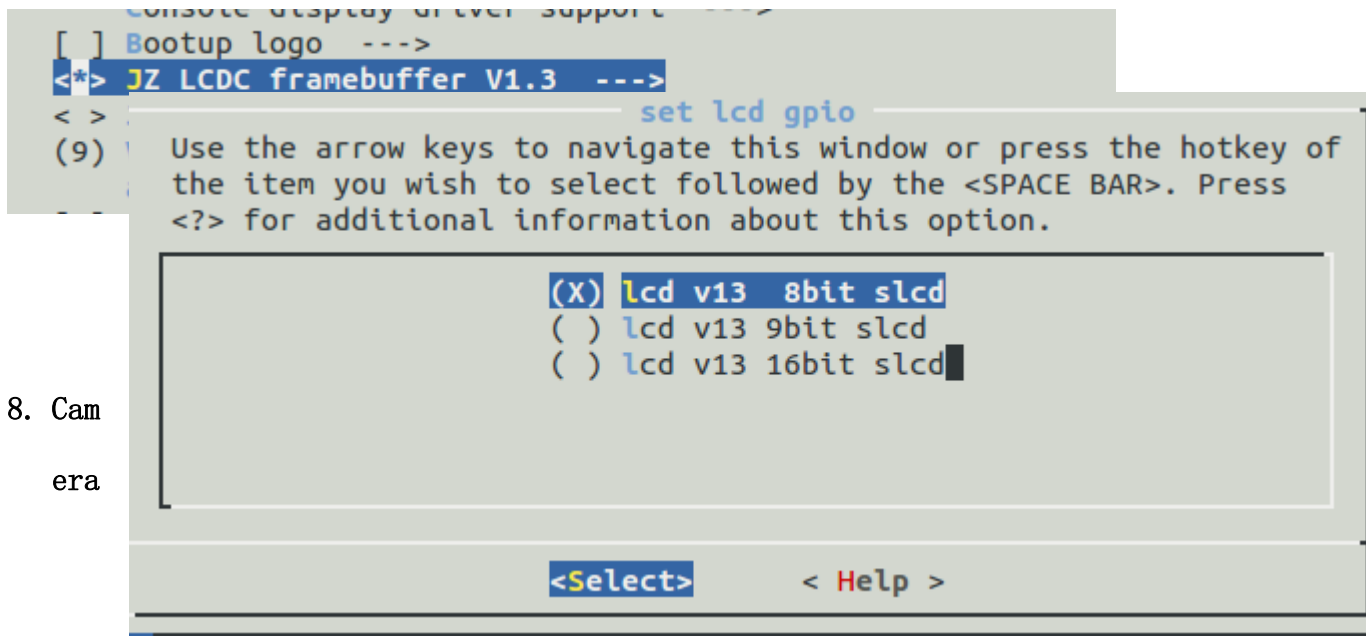
3.Device Drivers

-->Graphics support

--><*>JZ LCDC framebuffer V1.3

-->set lcd gpio (lcd v13 8bit slcd)

---> lcd v13 8bit slcd



3.8. Cam

era

3.8.1. 板级配置文件

板级配置文件路径如下:

“/arch/mips/xburst/soc-x1000/chip-x1000/halley2/common/cim_board.c”

3.8.2. drivers 文件描述

文件描述	板级名称	路径
控制器代码	jz-cim	halley2/platform/kernel/drivers/media/video/jz_camera_v13.c
摄像头代码	ov5640-front	halley2/platform/kernel/drivers/media/video/ov5640.c

3.8.3. camera 驱动配置方法

3.8.3.1. cim 控制器配置

1.Device Drivers

->Multimedia support

->V4L platform devices

->Soc camera support

ingenic cim driver used on camera x1000

```

CPU Power Management --->
[*] Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->

```

```

Broadcom specific AMBA --->
Multifunction device drivers --->
[*] Voltage and Current Regulator Support --->
<*> Multimedia support --->
Graphics support --->
<*> Sound card support --->
HID support --->

```

```

*** Media drivers ***
[*] Media USB Adapters --->
[*] V4L platform devices --->
[ ] Memory-to-memory multimedia devices --->
[ ] Media test drivers --->
*** Supported MMC/SDIO adapters ***

```

```

<*> SoC camera support
<*> platform camera support
< > ingenic cim driver used on jz4775/jz4780
<*> ingenic cim driver used on camera x1000
< > SuperH Mobile MIPI CSI-2 Interface driver
< > SuperH Mobile CEU Interface driver

```

2. Device Drivers

->Multimedia support

->Sensors used on soc_camera driver

->ov5640 camera support

```

< > Cypress firmware helper routines
*** Media ancillary drivers (tuners, sensors, i2c, frontends) ***
[*] Autoselect ancillary drivers (tuners, sensors, i2c, frontends)
Sensors used on soc_camera driver --->

```

```

< > ov5642 camera support
<*> ov5640 camera support
< > ov6650 sensor support

```


3.8.3.2. VPU 配置

1. Machine selection

->Soc type

->jz imem

```
Machine selection --->
Endianness selection (Little endian) --->
CPU selection --->
Kernel type --->

System type (Ingenic Xburst based machines) --->
[*] SOC type --->

[ ] early run init process
[ ] fast test reset_dll
[*] jz imem
[ ] jz ota updata
```

2. Device Drivers

->Graphics support

->JZ VPU driver

```
CPU Power Management --->
[*] Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->

[*] Voltage and Current Regulator Support --->
<*> Multimedia support --->
Graphics support --->
<*> Sound card support --->

< > JZ virtual framebuffer(just for test) --->
(9) Vsync skip ratio[0..9]
allocation frame buffer (alloc 3 frame buffers) --->
[*] JZ VPU driver --->
```

3.8.3.3. 串口配置

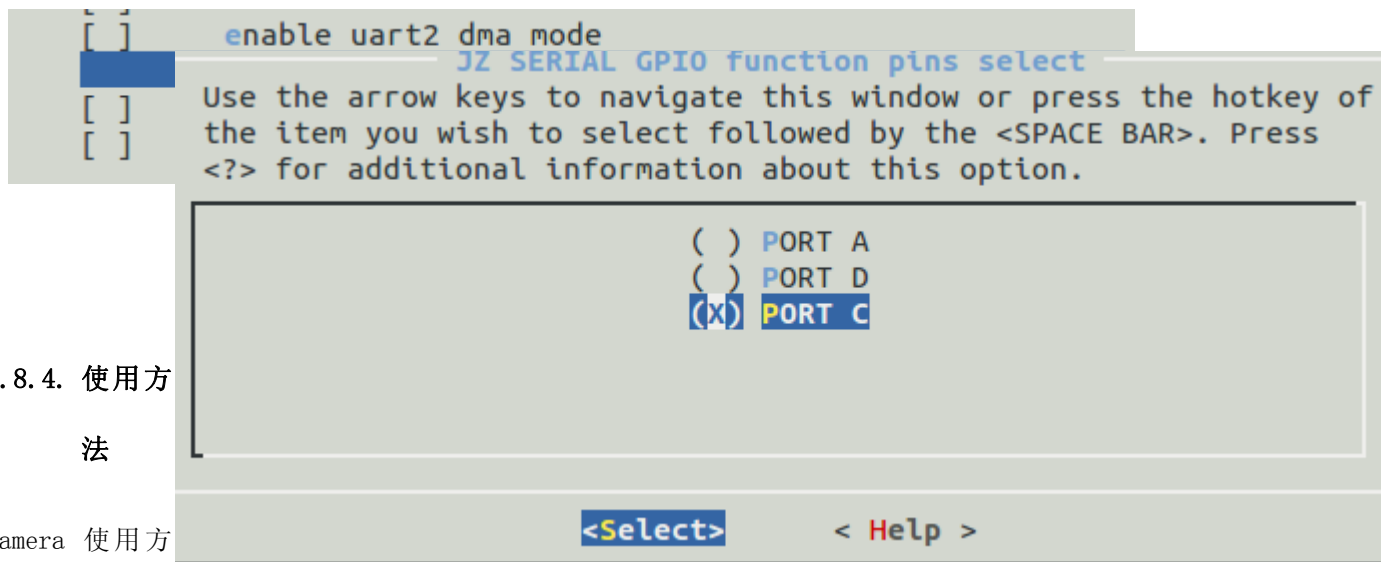
1. Device Drivers

->Character devices

->serial drivers

->JZ SERIAL GPIO function pins select (PORT C)

->PORT C



3.8.4. 使用方 法

camera 使用方
法 详 见

第六章测试用例6.1。

注意：目前发布的halley2

默认配置都不支持camera,但驱动中支持，若需要使用，则需手动配置执行3.8.3.1,3.8.3.2两个小节。

3.9. USB Camera

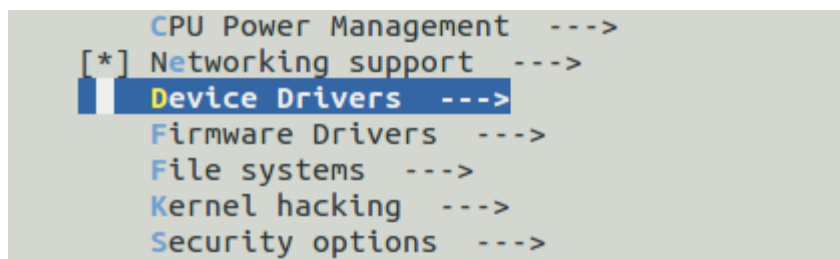
3.9.1. USB Camera 驱动配置方法

-> Device Drivers

-> Multimedia support

->Media USB Adapters

->USB Video Class (UVC)



```

Multifunction device drivers --->
[*] Voltage and Current Regulator Support --->
<*> Multimedia support --->
    Graphics support --->
    <*> Sound card support --->
    HID support --->

< > V4L2 int device (DEPRECATED)
    *** Media drivers ***
    [*] Media USB Adapters --->
    [*] V4L platform devices --->
    [ ] Memory-to-memory multimedia devices --->

--- Media USB Adapters
    *** Webcam devices ***
    <*> USB Video Class (UVC)
    [ ] UVC input events device support

```

3.9.2. USB camera 验证、使用方法

USB Camera使用方法详见第六章测试用例6.2。

注意：目前发布的halley2 所有默认配置以及Halley2_mini_v2.0(SPI-nor)开发板都不支持usb camera,但驱动中支持，若在Halley2_v2.0(SPI-nor)开发板需要使用，则需手动执行3.9.1配置

3.10. 休眠唤醒

系统如果长时间处于闲置状态时,可以让其进入睡眠模式。在这种模式下,系统大部分模块都置于低功耗模式,DRAM 处于自刷新模式并保存程序运行的现场,只保留 RTC 时钟工作以唤醒系统。

3.10.1. 休眠唤醒配置、使用方法

```

Power management options
->Suspend to RAM and standby
    Run-time PM core functionality
    Log time spent in suspend

```

3.10.2. 休眠唤醒验证方法

确保上述3.9.1所列配置正确选择后,重新编译内核、烧录、启动，在串口端执行以下命令使系统进入休眠(休眠后无法输入)

```
$echo mem >/sys/power/state
```

按下power键后唤醒。

3.11. Voice trigger

3.11.1. Voice trigger 简介

Voice

trigger(语音休眠唤醒)主要是利用linux系统和君正处理器支持休眠唤醒等特点,使君正方案达到更好省电效果。

Voice

trigger的代码可划分为两部分:第一部分为语音识别测试的代码;第二部分为语音休眠唤醒的代码。

文件描述:

语音休眠唤醒固件代码: drivers/char/voice_wakeup_v13/

语音识别测试驱动代码: drivers/char/jz_wakeup_v13.c

提供给其他应用的DMIC驱动: drivers/char/jz_dmic_v13.c

测试用例代码: tools/wakeup-test/wakup.c

测试用例所需的语音比对文件: tools/wakeup-test/ivModel_v21.irf

linux系统休眠后的入口文件: arch/mips/xburst/soc-x1000/common/pm_p0.c

3.11.2. Voice trigger 驱动配置方法

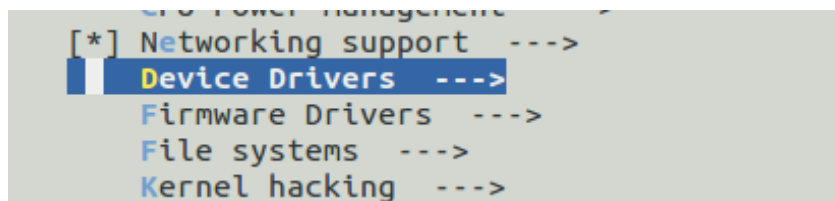
1.在“drivers/char/voice_wakeup_v13/wakeup_module/”目录下,首先执行make clean,然后执行./mkmodule.sh

2.Device Drivers

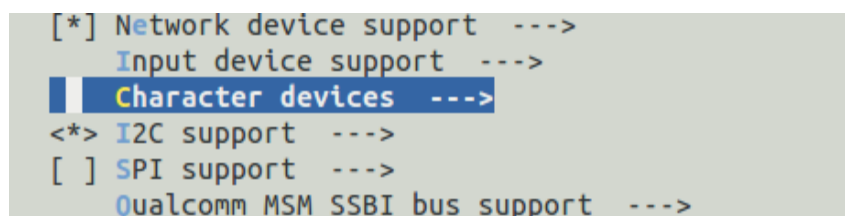
--> Character devices

--> Ingenic Dmic Driver v13

Ingenic Voice Wakeup Driver V13



```
[*] Networking support --->
[*] Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
```



```
[*] Network device support --->
Input device support --->
[*] Character devices --->
<*> I2C support --->
[ ] SPI support --->
Qualcomm MSM SSBI bus support --->
```

```
[ ] Ingenic Dmic Driver
[*] Ingenic Dmic Driver v13
[*] Ingenic Voice Wakeup Driver V13
[ ] Ingenic Test second refresh for watch(test driver.)
```

上述选项配置完成后执行以下命令进行编译：

```
$ make uImage
```

编译完成后会在“platform/kernel/arch/mips/boot”目录下产生voice trigger 需要的uImage。

3.11.3. 验证方法

将“halley2/platform/kernel/tools/wakeup-test”目录下的wakeup 及ivModel_v21.irf拷贝到文件系统。
在串口端输入以下命令（后台运行）：

```
# ./wakeup ivModel_v21.irf &
# open file[ivModel_v21.irf], ok!
open file[/dev/jz-wakeup] ok![ 2019.532480] enable wakeup function

open file[/sys/class/jz-wakeup/jz-wakeup/wakeup] ok!
read don[ 2019.546428] module open open_cnt = 1
e!!!!
#### begin read !!!!!

$ ./wakeup ivModel_v21.irf &
```

出现上述打印后对着开发板上的音频模块音频输入“灵犀灵犀”，串口出现如下打印：

```
WKUPOK[ 2026.691348] sys wakeup ok!--wakeup_timer_handler():158, wakeup_pending:1
[ 2026.705626] [Voice Wakeup] wakeup by voice.
#####ret:9, wakeup_ok
sh: input: not found
#####read ok!, wakeup ok!
#### begin read !!!!!
```

之后使开发板进入休眠状态（具体操作参见3.10.2），休眠之后通过音频“灵犀灵犀”唤醒系统。

注意：Halley2_v2.0(SPI-nor)开发板需要在J7 或 J8处插上DMIC

3. 12. AES-RSA

3.12.1. 驱动名称及路径

AES-RSA的驱动名称是“jz_security”

（1）设备节点：/dev/jz_security

使用如下的方式来操作驱动：

open(/dev/jz-security, ...) → ioctl(xxx...) → ioctl(xxx...) → ... → ioctl(xxx) --> close(device)

(2) 驱动文件路径:

“platform/kernel/drivers/misc/jz_security/”

3.12.2. 驱动配置

Device Drivers

--> Misc devices

--> JZ SECURITY Driver(AES && RSA)

```
Power management options --->
CPU Power Management --->
[*] Networking support --->
Device Drivers --->
Firmware Drivers --->
File systems --->
Kernel hacking --->
Security options --->
```

```
< > Parallel port support --->
[*] Block devices --->
Misc devices --->
< > ATA/ATAPI/MFM/RLL support (DEPRECATED) --->
SCSI device support --->
< > Serial ATA and Parallel ATA drivers --->
```

```
[ ] JZ V13 EFUSE Driver
[ ] JZ V11_IRDA Driver
[*] JZ SECURITY Driver(AES && RSA)
[ ] DEBUG of JZ_SECURITY DRIVER (NEW)
[ ] ingenic efuse support --->
```

3.12.3. IOCTL 命令定义

```
#define SECURITY_INTERNAL_CHANGE_KEY (0xffff0010)
/*设置AES-KEY*/
#define SECURITY_INTERNAL_AES (0xffff0020)
/*AES加密或解密*/
#define SECURITY_RSA (0xffff0030)
/*RSA加密或解密*/
```

3.12.4. 驱动结构体描述

以下驱动结构体定义路径为“/halley2-3.10-rdm/platform/kernel/tools/security-test”。

(1) 定义AES-KEY结构如下:

```
struct rsa_aes_packet {
    unsigned short oklen; //old AES-KEY len (unit:word)
    unsigned short nklen; // new AES-KEYlen (unit:word)
    unsigned int * okey; // old AES-KEYaddr
    unsigned int * nkey; // new AES-KEYaddr
};
```

也可以用数组来存储 AES-KEY，具体定义方式可见下例:

```
unsigned int user_key[9]=
{
    0x00040004,
    /*bit31~16: old key length ,bit15~bit0:new key length*/
    0x1,0x2,0x3,0x4, /*old AES-KEY*/
    0x4, 0x5,0x6,0x7, /*new AES-KEY*/
}
```

注：初始化AES-KEY时，需将“old AES-KEY”“new AES-KEY”设置成相同的初值。

(2) RSA参数结构

用户使用RSA加解密“AES-KEY”，描述RSA的结构体为“struct rsa_param”，具体结构如下:

```
struct rsa_param {
    unsigned int in_len;      //input data length (units:word)
    unsigned int key_len;     //private or public key length(units:word)
    unsigned int n_len;      //n length(units:word)
    unsigned int out_len;     //ouput length(units:word)
    unsigned int *input;      //input data buffer
    unsigned int *key;        /*Ku or KR*/ buffer
    unsigned int *n;          /*N*/buffer
    unsigned int *output;     //encrypted data or decrypted data buffer
    unsigned int mode;        //mode: 1,encrypt 0,decrypt
};
```

(3) CHANGE-KEY 结构体

```
struct change_key_param {
    int len;                  //rsa_enc_data len in bytes.
    int *rsa_enc_data;        //okey_len,nkey_len,okey,nkey
    int *n_ku_kr;             //NKU or NKR buffer, for rsa decrypt (62 words)
    int init_mode;            //init key 1;init key, 0:change key
};
```

其中rsa_enc_data是上述结构体“struct rsa_aes_packet”经过rac加密后的数据，上述加密数据的NKU或NKR。

(4) AES 加解密参数

目前我们只支持ECB模式，AES-KEY大小为128-bit，支持每次四word的AES加密解密。

```
struct aes_param {
    unsigned int in_len;           //input data length (units:word)
    unsigned int key_len;          //private or public key length(units:word)
    unsigned int n_len;            //n length(units:word)
    unsigned int out_len;          //ouput length(units:word)
    unsigned int *input;           //input data buffer
    unsigned int *key;             /*Ku or KR*/ buffer
    unsigned int *n;               /*N*/buffer
    unsigned int *output;          //encrypted data or decrypted data buffer
    unsigned int mode;             //mode: 1,encrypt 0,decrypt
};
```

4 编程指导

一：打开设备并初始化 AES控制器。

二：使用正确格式的AES-KEY，并进行RSA加密。

三：使用步骤1中的结果，设置AES-KEY到CPU。

四：执行AES加密或解密。

若想改变AES-KEY，请执行步骤二，步骤三，

在步骤二中应注意“1”为初始key值，“0”为改变后的key值。

3.12.5. USER API

(1) rsa加解密

```
int do_rsa(unsigned int fd,    //file node
    unsigned int orig_aes_len, // original AES_KEY length
    unsigned int * rsa_key,    //KU or KR
    unsigned int * n,
    unsigned int * input,      // input_data
    unsigned int * output,     //encrypted or decrypted data
    unsigned int mode);        //mode: 0:encryption 1:decryption
```

(2) 设置AES-KEY或改变AES-KEY

```
int setup_aes_key(int fd,    //file node
    unsigned int *key,        //encrypted AES-KEY by using RSA
    int len,                  // length of key
    unsigned int *nku_kr,     //NKU ok NKR(62 words)
    int init_mode);           //init_mode: 1:init code , 0:change code
```

(3) AES加密解密

```
int do_aes(int fd, //file node
    unsigned int *input, //input data
    unsigned int in_len, //input and output data length(unit:word)
    unsigned int *output, //output data(encrypted data or decrypted data)
    unsigned int mode);   //mode :0,encryption 1:decryption
```


4. LINUX 根文件系统

Linux 内核编译好之后,还必须有根文件系统(root filesystem)才能使一个Linux系统正常运行。

工程platform下默认采用jffs2文件系统（存储介质为spi nor flash）详见本章4.1.1,在”platform/development/device/device.mk”文件中选中”MAKE_SPI_NAND”变量的条件下，采用的是ubi文件系统（存储介质是 spi nand flash)详见本章4.1.2。

4.1. 制作文件系统

4.1.1. Jffs2 文件系统

1. 选择文件系统类型

在platform中默认采用jffs2文件系统，因此不需要做任何选择，可以编译制作。

2. 制作文件系统

进入platform目录下，执行如下命令：

```
$ make
```

在make时会利用development/rootfs目录下已打包好的文件系统tar包system.tar，在out/target/product/halley2/image/目录下生成镜像文件system.jffs2，使用默认大小12.5M。

4.1.2. Ubi 文件系统

1. 选择文件系统类型

在platform目录下，进入development/device/目录编辑device.mk文件。选中MAKE_SPI_NAND变量。

如：MAKE_SPI_NAND=y

2. 制作文件系统

进入platform目录下，执行如下命令：

```
$ make
```

在make时会利用development/rootfs/ubi目录下已打包好的文件系统tar包system-ubi.tar，在out/target/product/halley2/image/目录下生成镜像文件system.ubi。

注：由于nand的大小为128M，rootfs的分区最大可为115M，因此文件系统的镜像大小不能超过115M。

注意：halley2目前发布的版本暂不支持ubi 文件系统。

4.2. 扩展文件系统

根据4.1可以制作出平台自带的文件系统，如果需要添加新的应用或功能，可以在自带文件系统的基础上进行扩展。

在out/target/product/halley2/system目录（文件系统system.tar包的解压目录）下的相应目录文件中添加所扩展的应用文件即可。

添加完扩展的文件后，在platform目录下执行如下命令重新编译。

\$ make

注：扩展后的文件系统如需长久使用应打包替换

development/rootfs目录下的system.tar或及时备份，否则make distclean时会删除out/target/product/目录下的所有东西。

打包替换文件系统tar包：

i. 进入out/target/product/halley2/system/目录

\$ cd out/target/product/halley2/system/

ii. 打包

\$ tar cvf ../system.tar ./*

iii. 替换

如需替换，可以将打包好的文件系统tar包，替换掉rootfs目录下相应文件类型的tar包。

4.2.1. Jffs2 文件系统

如果更换 nor flash 的情况下，存储空间发生变化后，可以在 development/device/device.mk 文件中修改 ROOTFS_JFFS2_SIZE 变量来改变文件系统镜像的默认大小，同时擦除其大小发生变化可在此处通过ROOTFS_JFFS2_NORFLASH_ERASESIZE变量进行修改。

```
ROOTFS_JFFS2_NORFLASH_ERASESIZE:= 0x8000
```

```
ROOTFS_JFFS2_SIZE:= 0xc80000 #文件系统镜像大小十六进制表示
```

注：在扩展文件系统的时候，文件系统的大小可能增大。理论上jffs2的压缩比为2: 1，因此文件系统大小不要超过25M。

4.2.2. Ubi 文件系统

开发板自带nand的大小无法满足你的开发需求时，更换nand flash的情况下。需要根据所更换的nand数据手册，确定其页的大小、逻辑块擦除大小和最大的逻辑擦除块数量。可以在development/device/device.mk文件中进行修改，具体修改参数如下：

```
#nand flash config
```

```
ROOTFS_UBIFS_LEBSIZE := 0x1f000
```

```
ROOTFS_UBIFS_MAXLEBCNT := 2048
```

```
ROOTFS_UBIFS_MINIOSIZE := 0x800
```

注意：halley2 发布的开发板不带nand flash,只为nor flash.

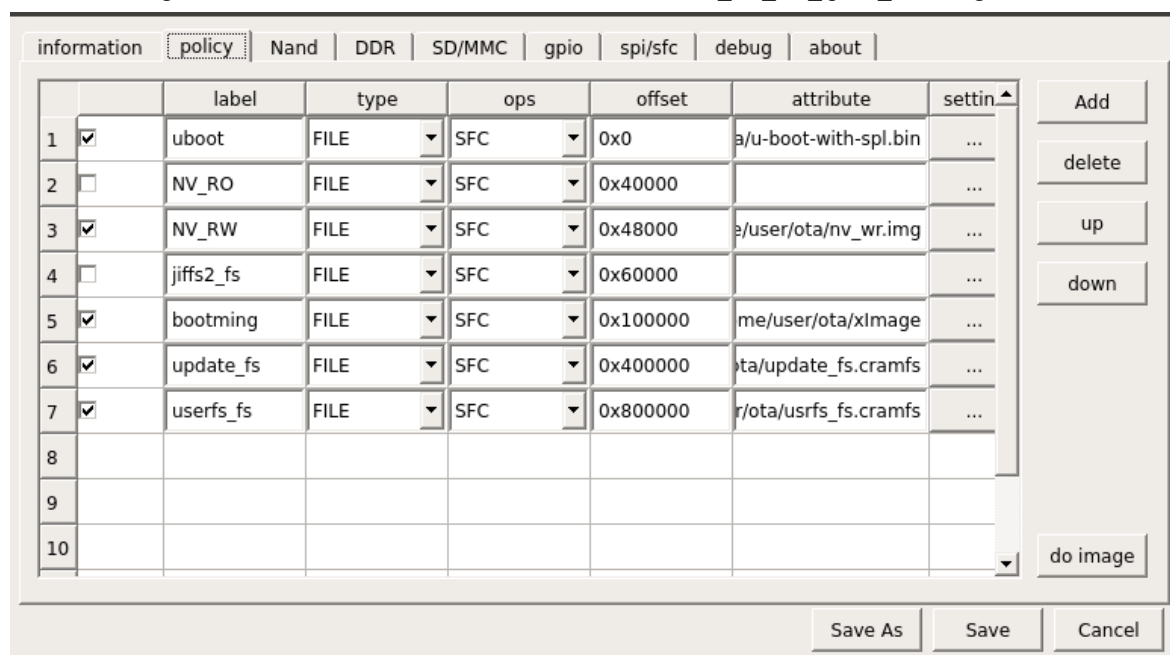
5. OTA

（注意：halley 2 发布的版本目前尚不支持OTA）

5.1. 环境准备

5.1.1. 烧录工具分区配置

在 config 配置的“information”选项中“board”选择“x1000_sfc_ota_lpddr_linux.cfg”



其中：

1. u-boot的烧录地址为0x0
2. NV_RO区的烧录地址为0x40000
3. NV_RW区的烧录地址为0x48000
4. jiffs2的烧录地址为0x60000
5. xImage的烧录地址为0x100000
6. updatefs的烧录地址为0x400000
7. usrfs的烧录地址为0x800000

NOTE: jiffs2不需烧录。

5.1.2. 制作升级镜像

1. 修改参数配置

(1) 进入halley2/platform/development/device目录，打开device.mk配置文件。

(2) 在“export MAKE_OTA=”后添加 “y” 如下图所示：

```

1 export DEVICE_NAME:=phoenix
2 export PRODUCT_NAME:=product
3 export OUT_DIR=out
4 export OUT_HOST=host
5 export OUT_TARGET=target
6 export SYSTEM-OTA=system_ota/ota_fs
7 export SYSTEM-OTA-USR=system_ota/ota_usr_fs
8 export MAKE_OTA=y

```

5.2. 编译

在 platform 目录下输入以下命令进行整体编译：

```
$ make install
```

编译完成后在 halley2/platform/out/target/product/halley2/image 下生成升级所需的镜像。

生成对应的镜像如下：

文件名	作用
u-boot-with-spl.bin	boot的镜像
xImage	kernel的镜像
update_fs.cramfs	update_fs文件的镜像（升级用的文件系统）
usrfs_fs.cramfs	usrfs_fs文件的镜像（用户用的文件系统）

5.3. NV_RW 分区 bin 文件的制作

使用“dd if=/dev/zero of=xxx.bin bs=1024 count=96”

命令，把生成的xxx.bin烧到NV_RW分区，进行清NV_RW区操作。

NOTE：测试之前，需要先清NV_RW分区。

5.4. 升级包的制作

1. 把要升级的新的kernel, updatefs_fs,

usrfs_fs对应的镜像放在halley2/platform/development/ota/updatezip/image路径下。

文件名	作用
u-boot-with-spl.bin	boot的镜像
xImage	kernel的镜像
update_fs.cramfs	update_fs文件的镜像（升级用的文件系统）
usrfs_fs.cramfs	usrfs_fs文件的镜像（用户用的文件系统）

halley2/platform/development/ota/updatezip/split/scrip文件夹放升级脚本update.script。

2.在halley2/platform/development/ota目录下，
执行mkzip.sh脚本制作升级包并上传到服务器。第一个参数为version，第二个参数为要上传的升级包的url。

例： `$. /mkzip.sh 20150808 http://192.168.1.200/ota/download-bliu`

5.5. 升级

等系统运行起来之后，连接 wifi，输入 wr_flag URL，读取 nv 中的升级标志位，进行升级。在串口端输入以下命令：

例： `$ wr_flag http://192.168.1.200/ota/download-bliu`

升级完成之后，建议删除上次的info和升级包，以免影响下次制作新的升级包。

NOTE： 1) 如果输入的url无效，会使用备用的url，如果两个url都无效，使用默认url。

2)

在烧录完ota的代码之后，若想烧录其他代码请在烧录工具中选中全部擦除项即“config->sfc->all erase”，方能正常烧录。

6. 测试用例

6.1. Camera 测试

打开串口，开发板上电启动之后，进入到”/testsuit/cim/”目录，在当前目录输入以下命令使用Camera功能：

1) 照相

```
$/cimutils -I 0 -C -v -x 320 -y 240
```

2) 预览

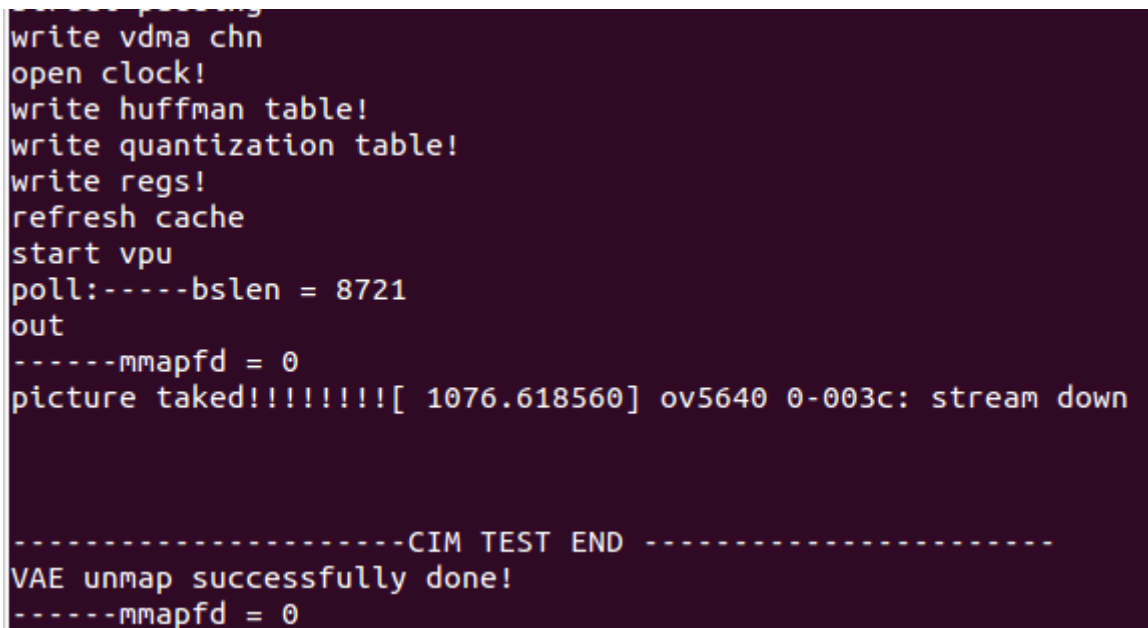
```
$/cimutils -I 0 -P -w 320 -h 240
```

3) 预览并照相

```
$/cimutils -I 0 -P -v -l -w 320 -h 240
```

注：在当前目录执行命令可查看详细帮助

```
$/cimutils --help
```



```
write vdma chn
open clock!
write huffman table!
write quantization table!
write regs!
refresh cache
start vpu
poll:-----bslen = 8721
out
-----mmapfd = 0
picture taked!!!!!!![ 1076.618560] ov5640 0-003c: stream down

-----CIM TEST END -----
VAE unmap successfully done!
-----mmapfd = 0
```

串口打印如下则 camera 照相成功：

照相成功后，会在当前路径下生成 test3.jpg 照片文件，可在pc端输入以下命令：

```
$ adb pull /testsuit/cim/test3.jpg .
```

命令执行成功即可在本地查看图片是否正确

6.2. USB Camera 测试

打开串口，开发板上电启动之后，进入到”/testsuit/grab/”目录，在当前目录输入以下命令使用Ca

mera功能:

```
$. /grab -w 320 -h 240 -c 10 -r 5 -y
```

注: 在当前目录下执行以下命令可查看具体参数说明:

```
$. /grab --help
```

串口打印如下则USB Camera照相成功:

照相成功后会在当前路径下产生“p-x.jpg”照片文件, 其中“x”为“0~9”。可在pc

端输入以下命令查看图片:

下面以查看“p-0.jpg”为例:

```
width = 320, height = 240
input_yuv = 1
grab version 0.1.4
video /dev/video0
coc --1
{ pixelformat = 'YUYV', description = 'YUYV 4:2:2 (YUYV)' }
coc --2
ddddddddddddCurrent data format information:
        width:320
        height:240
running.....
forever = 0, fcount=10
oooooooooooo 120 frame time = 32 429496733696ns/frame
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
yuv
savejpeg.c:get_pictureYUYV:293
[ 25.541770] dwc2 dwc2: Unlink after no-IRQ? Controller is probably using the wrong IRQ.
close fd 3
```

```
$adb pull /testsuit/grab/p-0.jpg .
```

命令执行成功可在本地查看“p-0.jpg”是否正确

6.3. Wi-Fi 连接测试

6.3.1. Halley2_v2.0(SPI-nor)开发板使用方法:

本次对airkiss的更新, 支持多种协议 (airkiss、cooe), 同时也修改了它的使用方式, 具体操作如下:

1. 在Android 手机上安装com.broadcom.cooeedemo-v1.4.0.apk。
2. 打开手机上的WI-FI 并连接到一个AP 路由器。
3. 运行BrcmCooee 应用。
4. 在BrcmCooee 应用中输入SSID 名称(第一个输入项), AP 路由器的密钥(第二个输入项) (确保BrcmCooee 应用中所连接的路由器和手机连接的是同一个)。
5. 启动halley2 开发板, 在串口端输入以下命令:

```
$ airkiss
```

在串口输出"Easy setup target library v3.2.0" 时点击手机应用界面的发送按钮, halley2 开发板将接收到SSID 和密钥, 并开始连接AP 路由器。

6. 可以通过ping www.baidu.com 或者网关来检测网络是否连通。

由于在第一次使用时，已经生成了配置文件，配置文件路径“/etc/wpa_supplicant.conf”。因此下次开机时将自动去连接网络，如果此时配置文件中保存的wifi名和密码不存在或不正确，将无法连接网络，可以重新执行airkiss配置网络。

6.3.2 Halley2_mini_v2.0(SPI-nor)开发板使用方法:

1. 在Android 手机上安装AirKissDebugger.apk。
2. 打开手机上的WI-FI 并连接到一个AP 路由器。
3. 运行AirKissDebugger 应用。
4. 在AirKissDebugger应用中输入SSID 名称(第一个输入项)，AP 路由器的密钥(第二个输入项) 和AESKey (第三个输入项)，默认AESKey 是"0123456789123456", 这里需要填充为"Wechatiothardwav”(确保AirkissDemo应用中所连接的路由器和手机连接的是同一个)



4. 在终端上执行airkiss命令
5. 在终端下显示如下图所示之后即联网完成，可以通过ping www.baidu.com 或者网关来检测网络是否连通。

```
Sending select for 192.168.1.102...
Lease of 192.168.1.102 obtained, lease time 7200
deleting routers
adding dns 8.8.8.8
adding dns 192.168.1.2
```

6. 若是 halley2 mini nor ,halley2mini nand 这两种开发板在执行 airkiss 出现 insmod 问题时，需要用户手动执行如下步骤解决：
 - a. cd kernel 目录下：执行 make modules
 - b. cp drivers/net/wireless/ssv6xxx/ssv6051.ko ~/
 - c. adb push ~/ssv6051.ko /opt/modules
 - d. 再次执行 airkiss，重复第 4,5 步

6.4. Bluez 测试

1. 启动Bluez

开发板上电启动后执行“bt_enable”启动bluez, 等待10s 左右重新出现提示符“#”, 启动完成。此时用手机搜索蓝牙设备即可找到名为“BlueZ”的设备(可执行配对操作)。

2. 向设备发送文件

执行如下命令:

```
# sdptool add OPUSH
```

将会出现如下提示信息:

```
OBEX Object Push service registered
```

此时用手机与开发板配对 (如果之前有配对操作则应先取消当前配对, 再进行重新配对)。

继续执行如下命令

```
# obex_test -b local 9
```

当出现如下提示信息:

```
Using Bluetooth RFCOMM transport
```

```
OBEX Interactive test client/server
```

```
>
```

发送“s”指令:

(请提前准备好要发送的文件, 避免时间太长, 1s 左右操作延时)

```
>s
```

然后手机向BlueZ 设备发送任意文件, 在程序提示Timeout while doing

OBEX_HandleInput()时再次输入“s”

```
>s
```

如果想继续上传文件需请从“# obex_test -b local 9”命令开始执行直到最后一步。

3. 设备接收文件

出现如下信息, 就开始接收文件, 直到文件接收完成(接收文件在当前路径下存储)

```
connect_server()
```

```
server request finished!
```

```
server_done() Command (00) has now finished
```

```
OBEX_HandleInput() returned 12
```

```
OBEX_HandleInput() returned 990
```

注意: Halley2_mini_v2.0(SPI-nor)开发板不支持蓝牙功能, Halley2_v2.0(SPI-nor)开发板的蓝牙尚存在问题

7. 常见问题及解决办法

7.1. Ubuntu 下 Oracle VM VirtualBox 虚拟机烧录问题

7.1.1. 问题现象

如果您在Ubuntu环境下用Oracle VM VirtualBox软件安装了Windows XP虚拟机，添加USB设备时，出现设备名乱码现象，具体情况如下：

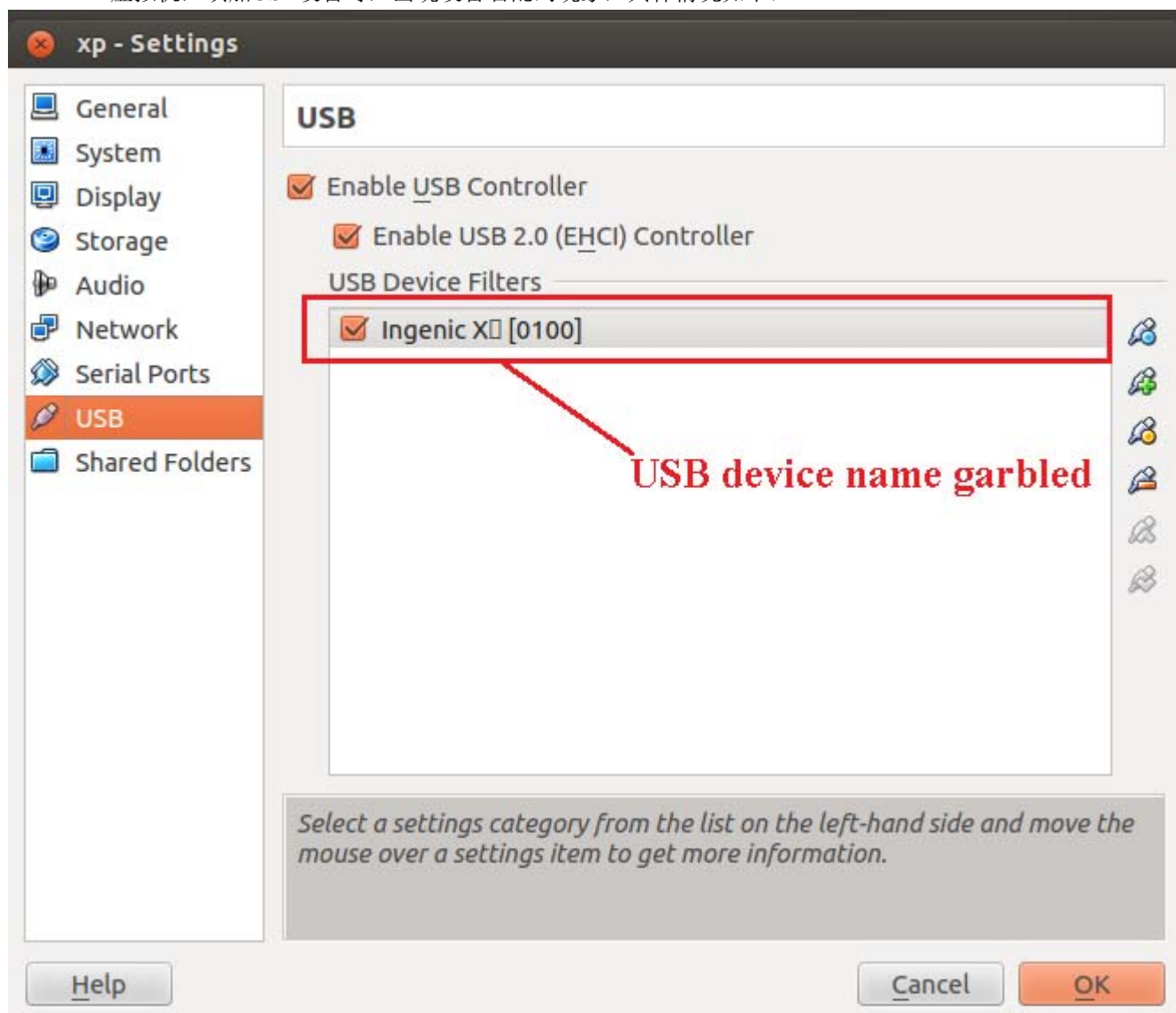


图 1

为了确保正常使用，建议一旦出现上述情况，就手动修改X1000 USB设备描述信息（详见本章7.1.2），否则虚拟机将无法启机，出现如下情况：

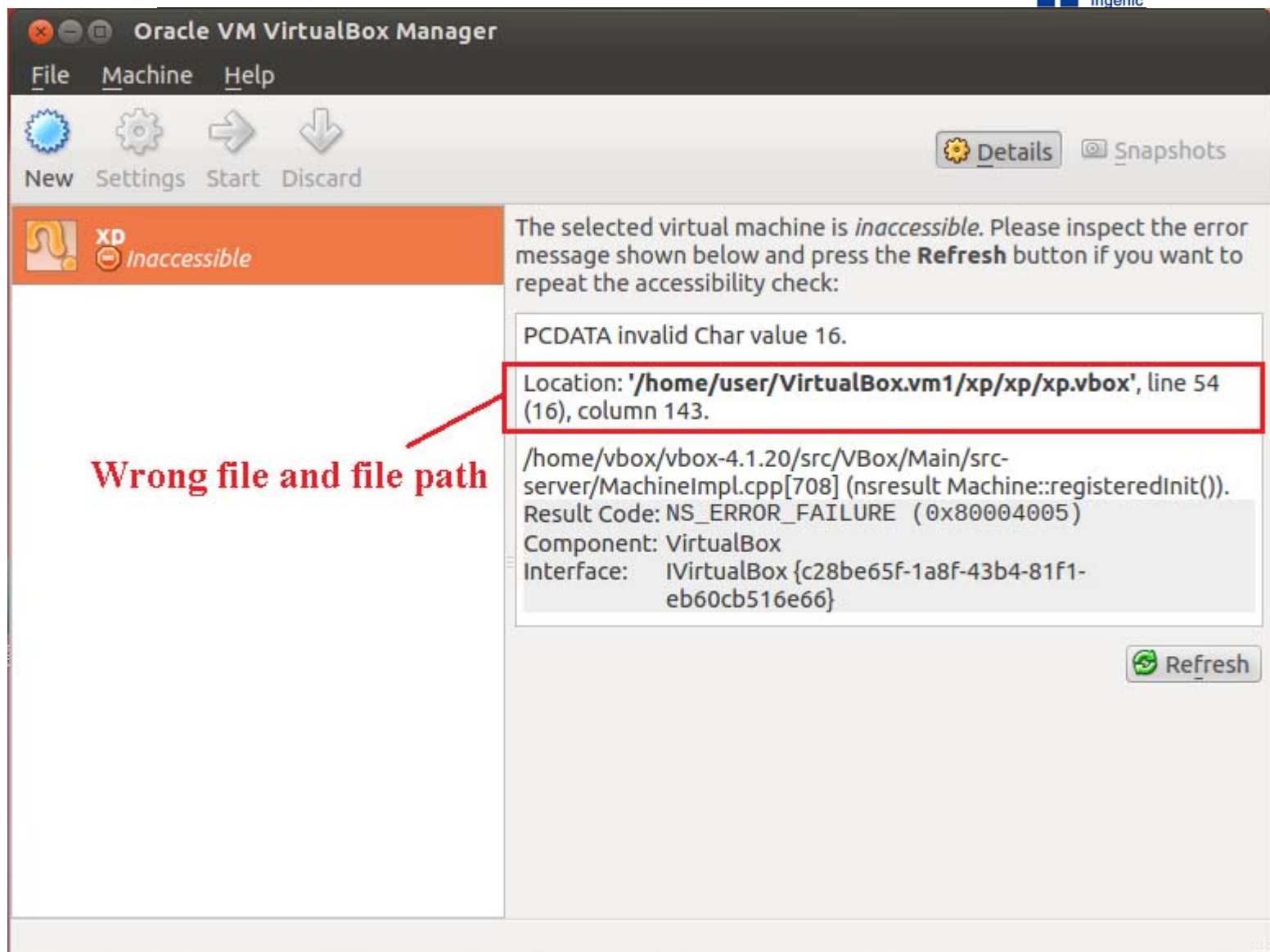


图 2

7.1.2. 解决办法

在7.1.1 图2

中已经有报错信息，按照图中提示，在Ubuntu环境下打开“/home/user/VirtualBox.vm1/xp/xp/xp.vbox”文件，删除第54行。

USB Filter Details

Name: ingenix X1000

Vendor ID: A108

Product ID: 1000

Revision: 0100

Manufacturer: Ingenic

Product:

Serial No.:

Port:

Remote: No

Cancel OK

Modified device name

Modified this option is null

将上图中显示乱码的选项“Name”中的内容修改为“ingenix X1000”,将“Product”选项中的内容清空,修改完成后显示如下,按下“OK”保存即可。

图5

7.2. Ubuntu 下 adb 问题

7.2.1. 问题现象

若您在Ubuntu环境下出现adb 无法使用的情况,具体现象如下:

```
user@user-FMVDB2A0C1:~$ adb push hello /
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
error: insufficient permissions for device
user@user-FMVDB2A0C1:~$
```

7.2.2. 解决办法

在超级用户权限下，在终端执行如下命令：

```
# adb kill-server
```

```
# adb start-server
```

命令执行完成后，退出超级用户，操作完成后adb 可正常使用。

8. 源码编译

8.1. 源码目录结构

SDK目录结构说明：

```
* README:      //README for this platform
* burnertools   //烧录工具
* docs          //开发文档和README
* toolchains:   //交叉工具链
* platform:     //平台开发所需要的所有源码文件
  |--kernel     //kernel源码
  |--u-boot     //u-boot源码
  |--development //平台相关内容
    |--device   //板级相关配置
    |           例如： kernel 和 u-boot的配置文件.
    |--rootfs   //文件系统的压缩包
    |--testsuit //所有的可执行测试程序
  |
  |--Makefile   //整体编译脚本
  |--tools
  |  |-- android //adb源码及生成adb工具
  |  |-- build   //Makefile和文件系统镜像制作工具
  |--out        //编译后文件的输出目录
    |--target/product/halley2
      |--image  //u-boot-with-spl.bin、uImage、system.jffs2
      |--testsuit //生成的测试程序
      |--tools  // 生成工具程序
      |--system //解压的文件系统
```

8.2. 整体编译

配置开发板类型（halley2_nor, halley2_nand, halley2_mini_nor, halley2_mini_nand）

以 halley2 mini nor 开发板为例

根据开发板类型，修改文件 platform/device/device.mk

export BOARD_NAME=halley2_mini_nor

I. 在 halley2 目录下,执行如下命令:

```
$ cd platform
$ source tools/build/source.sh
```

II 编译

```
$ make
```

编译完成，在当前目录 out/target/product/halley2/image/ 下生成三个镜像文件如下表:

文件名	作用
u-boot-with-spl.bin	u-boot 镜像
uImage	kernel 镜像
system.jffs2	文件系统镜像

III. 安装

如需安装测试程序和工具到系统中，则执行如下命令:

```
$ make install
```

该命令同时也会编译所有的工程源码。

8.3. 部分编译

首先, 在 halley2 目录下, 执行如下命令:

```
$ cd platform
$ source tools/build/source.sh
```

A. 编译 bootloader:

在 platform 目录下, 执行如下命令:

```
$ cd u-boot
$ make halley2_v10_uImage_sfc_nor (两款开发板皆可用)
```

将会在当前目录生成 u-boot-with-spl.bin.

B. 编译 kernel

在 platform 目录下, 执行如下命令:

```
$ cd kernel
$ make halley2_nor_v10_linux_defconfig (Halley2_v2.0(SPI-nor)开发板)
$ make
halley2_mini_nor_v10_linux_defconfig (Halley2_mini_v2.0(SPI-nor)开发板)
$ make uImage
```

将会在 arch/mips/boot/ 下生成 uImage.

备注: 测试主机可安装 Ubuntu 12.04 32bit 、Ubuntu 12.04 64bit 、Ubuntu 14.04 32bit 、Ubuntu 14.04 64bit, 本文档是在测试主机安装了 Ubuntu 14.04 64bit 环境下操作的。