

目录

1、虚拟机环境搭建	2
2、解压 SDK.....	2
2、编译 uboot	3
3、编译 kernel.....	3
4、根文件系统	7
4.1 编译	7
4.2 文件系统及驱动说明	8
4.3 jffs2 文件系统	9
5、烧写镜像文件	11
5.1 Uboot 命令行烧写	11
5.2 Hitool 烧写	11
6、驱动加载	12
6.1 手动加载	12
6.2 自动加载	12
7、图像测试	12
7.1 sample 程序测试	12

1、虚拟机环境搭建

Sudo apt-get install vim

Sudo apt-get install ssh //开启 ssh 服务，方便 winSCP 传 windows 文件到虚拟机

修改 /etc/default/tftpd-hpa 如下：

```
【
    TFTP_DIRECTORY="/home/xstrive/hisi/tftpboot"
】
```

参考开发指南《Hi3516EV200／Hi3516EV300／Hi3518EV300／Hi3516DV200 开发环境用户指南.pdf》：

Hi3516ev200\SDK\Hi3516E V200R001C01SPC012\ReleaseDoc\zh\01.software\board\OSDRV

2、解压 SDK

说明：请在 linux 环境下解压，并解压在同一目录下。

源码路径：

Hi3516ev200\SDK\Hi3516E

V200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\board\Hi3516EV200_SDK_V1.0
.1.2\package

参照文档：

Hi3516ev200\SDK\Hi3516E

V200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\board\Hi3516EV200_SDK_V1.0
.1.2\package\osdrv\opensource\kernel\readme_cn.txt

【

本文档简要说明将内核的补丁打到 v4.9.37 的 linux kernel 上。

1、从 linux 开源社区下载 v4.9.37 版本的内核：

- 1)进入网站： www.kernel.org
- 2)选择 HTTP 协议资源的 <https://www.kernel.org/pub/>选项,进入子页面
- 3)选择 linux/菜单项，进入子页面
- 4)选择 kernel/菜单项，进入子页面
- 5)选择 v4.x/菜单项，进入子页面
- 6)下载 linux-4.9.37.tar.gz（或 linux-4.9.37.tar.xz）

2、打补丁

- 1)将下载的 linux-4.9.37.tar.gz 存放到 osdrv 的 opensource/kernel 目录中
- 2)在 linux 服务器中进入 osdrv 的根目录,执行如下命令：
cd opensource/kernel
tar -zxf linux-4.9.37.tar.gz
mv linux-4.9.37 linux-4.9.y
cd linux-4.9.y

```
patch -p1 < ../linux-4.9.37.patch
cd ../
tar -czf linux-4.9.y.tgz linux-4.9.y
cd ../../
```

注意：

若下载的内核格式为 linux-4.9.37.tar.xz，

第一步先用：

```
xz -d linux-4.9.37.tar.xz
```

命令将 linux-4.9.37.tar.xz 解压为 linux-4.9.37.tar。

第二步再用：

```
tar -xvf linux-4.9.37.tar
```

解压 linux-4.9.37.tar。

】

编译参照文档：

Hi3516ev200\SDK\Hi3516E

V200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\board\Hi3516EV200_SDK_V1.0

.1.2\package\osdrv\ readme_cn.txt

(1)清除整个 osdrv 目录的编译文件：

```
make clean
```

(2)彻底清除整个 osdrv 目录的编译文件，除清除编译文件外，还删除已编译好的镜像：

```
make distclean
```

(3)编译整个 osdrv 目录：

```
make OSDRV_CROSS=arm-himix100-linux CHIP=hi3518ev300 all
```

2、编译 uboot

如非必要，请不要编译烧写 uboot。

Uboot 移植参照文档《Hi3516EV200／Hi3516EV300／Hi3518EV300／Hi3516D V200 U-boot 移植应用开发指南.pdf》：

SDK\Hi3516E V200R001C01SPC012\ReleaseDoc\zh\01.software\board\OSDRV\

SDK 自动 uboot 镜像路径：

SDK\Hi3516EV200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\board\Hi3516E

V200_SDK_V1.0.1.2\package\hi3518ev300_spi_image_uclibc\ u-boot-hi3518ev300.bin

3、编译 kernel

内核打补丁后的路径：*/osdrv/opensource/kernel/linux-4.9.y

(1) 内核配置：

步骤 1 手动拷贝.config 文件：

```
cp arch/arm/configs/hi3518ev300_full_defconfig .config
```

步骤 2 用户通过“make menuconfig”进行内核配置：

```
make ARCH=arm CROSS_COMPILE=arm-himix100-linux- menuconfig
```

步骤 3 选择需要的模块

步骤 4 选择完毕后，保存并退出。

(2) 编译 kernel:

```
make ARCH=arm CROSS_COMPILE=arm-himix100-linux- uImage -j20
```

说明：

如果编译过程中出现错误，按顺序执行以下命令：

```
make ARCH=arm CROSS_COMPILE=arm-himix100-linux- clean
```

```
make ARCH=arm CROSS_COMPILE=arm-himix100-linux- menuconfig
```

```
make ARCH=arm CROSS_COMPILE=arm-himix100-linux- uImage
```

```
CHK include/config/kernel.release
CHK include/generated/uapi/linux/version.h
CHK include/generated/utsrelease.h
CHK include/generated/timeconst.h
CHK include/generated/bounds.h
CHK include/generated/asm-offsets.h
CALL scripts/checksyscalls.sh
CHK include/generated/compile.h
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
DTB arch/arm/boot/dts/hi3516ev200-demb.dtb
Kernel: arch/arm/boot/zImage-dtb is ready
UIMAGE arch/arm/boot/uImage
Image Name: Linux-4.9.37
Created: Wed Jun 29 03:50:42 2022
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 3397596 Bytes = 3317.96 kB = 3.24 MB
Load Address: 40008000
Entry Point: 40008000
Kernel: arch/arm/boot/uImage is ready
xstrive@ubuntu:~/xkw/project/hi3516ev200/src/osdrv/opensource/kernel/linux-4.9.y
$ cp arch/arm/boot/uImage ~/hisi/tftpboot/xkw/hi3516ev200/
xstrive@ubuntu:~/xkw/project/hi3516ev200/src/osdrv/opensource/kernel/linux-4.9.y
$
```

(3)其他配置:

➤ 内核支持 jffs2 文件系统:

```
Make menuconfig
```

```
File systems ->
```

```
<*>Miscellaneous filesystems ->
```

```
<*>Journaled Flash File System v2 (JFFS2)support
```

➤ 若制作支持 squashfs 的内核镜像。进入 linux-4.9.y 目录下，执行以下命令：

```
cp arch/arm/configs/hi3516ev200_XXX_defconfig .config
```

```
make ARCH=arm CROSS_COMPILE=arm-himixXXX-linux- menuconfig （保存退出即可）
```

打开支持 xz 压缩算法选项：

```
File systems --->
```

```
[*] Miscellaneous filesystems --->
```

```
<*> SquashFS 4.0 - Squashed file system support
```

```
[*] Include support for XZ compressed file systems
```

保存配置

```
make ARCH=arm CROSS_COMPILE=arm-himix100-linux- uImage
```

备注：运行内核报错报以下错误，原因为内核配置未支持 squashfs

Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(31,2)

- USB 作为网口、串口、U 盘、摄像头配置：参考《外围设备驱动 操作指南.pdf》

Device Drivers --->

[*] USB support --->

<*> DesignWare USB3 DRD Core Support

DWC3 Mode Selection (Dual Role mode) --->

<*> USB Gadget Support --->

<*> USB functions configurable through configs

[] Generic serial bulk in/out (NEW)

[*] Abstract Control Model (CDC ACM)

[] Object Exchange Model (CDC OBEX) (NEW)

[] Network Control Model (CDC NCM) (NEW)

[*] Ethernet Control Model (CDC ECM)

[] Ethernet Control Model (CDC ECM) subset (NEW)

[*] RNDIS

[] Ethernet Emulation Model (EEM) (NEW)

[] Mass storage

[] Audio Class 1.0

[] Audio Class 2.0

[*] USB Webcam function

< > USB Gadget Drivers

PHY Subsystem --->

<*> HISI XVP USB2 PHY Driverr

<*> Sound card support --->

<*> Advanced Linux Sound Architecture --->

[*] USB sound devices (NEW) --->

<*> USB Audio/MIDI driver

- USB 选 host 模式，外接 usb 转以太网适配器（发货自带型号：**ASIX AX88772B USB 2.0 Ethernet**）

设置 usb 模式：

Device Drivers --->

[*] USB support --->

<*> DesignWare USB3 DRD Core Support

DWC3 Mode Selection (Host only mode) --->

注意：如下图，若选 Dual Role mode，则需要设置下 arch/arm/boot/dts/hi3518ev300.dtsi 设备树文件 usb 结点为 host 模式

```

usbdrd3_0: usb3-0{
    compatible = "hisil,dwusb2";
    reg = <0x10030000 0x10000>,
        <0x12010000 0x1000>;
    #address-cells = <1>;
    #size-cells = <1>;
    crg_offset = <0x140>;
    crg_ctrl_def_mask = <0x3308>;
    crg_ctrl_def_val = <0x1308>;
    clocks = <&clock HI3518EV300_USB2_BUS_CLK>,
        <&clock HI3518EV300_USB2_REF_CLK>,
        <&clock HI3518EV300_USB2_UTMI_CLK>;
    clock-names = "usb2_bus_clk",
        "usb2_ref_clk",
        "usb2_utmi_clk";
    resets = <&clock 0x140 3>;
    reset-names = "vcc_reset";
    ranges;

    hidwc3@0x100e0000 {
        compatible = "snps,dwc3";
        reg = <0x10030000 0x10000>;
        interrupts = <0 39 4>;
        interrupt-names = "peripheral";
        phys = <&usb2_phy0>;
        phy-names = "usb2-phy";
        maximum-speed = "high-speed";
        /*dr_mode = "peripheral"; xkw test */
        dr_mode = "host";
        eps_directions = <0x6a>;
        snps,eps_new_init;
        snps,usb2-lpm-disable;
    };
};

```

选择适配器

Device Drivers -->

[*] Network device support -->

<*> USB Network Adapters -->

<*> Multi-purpose USB Networking Framework

<*> ASIX AX88xxx Based USB 2.0 Ethernet Adapters

<*> ASIX AX88179/178A USB 3.0/2.0 to Gigabit Ethernet

-*- CDC Ethernet support (smart devices such as cable modems)

➤ 支持 wifi 模块:

Networking support -->

wireless -->

<M> cfg80211 - wireless configuration API

[*] enable powersave by default

[*] cfg80211 wireless extensions compatibility

<M> Generic IEEE 802.11 Networking Stack (mac80211)

再还有一个配置:

Device Drivers -->

[*] Network device support -->

[*] Wireless LAN -->

<*> IEE 802.11

```
notkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, </> 1
</> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

(-)
< > Atmel at76c503/at76c505/at76c505a USB cards
[*] Broadcom devices
< > Broadcom 43xx wireless support (mac80211 stack)
< > Broadcom 43xx-legacy wireless support (mac80211 stack)
< > Broadcom IEEE802.11n PCIe SoftMAC WLAN driver
< > Broadcom IEEE802.11n embedded FullMAC WLAN driver
[*] Cisco devices
[*] Intel devices
[*] Intersil devices
<*> IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
[*] Support downloading firmware images with Host AP driver
[*] Support for non-volatile firmware download
< > Softmac Prism54 support
[*] Marvell devices
< > Marvell 8xxx Libertas WLAN driver support
< > Marvell 8xxx Libertas WLAN driver support with thin firmware
< > Marvell WiFi-Ex Driver
↓(+)

<Select> < Exit > < Help > < Save > < Load >
```

主要是上面三个配置。

重新编译内核，烧到板子里面。

4、根文件系统

4.1 编译

说明：

SDK 软件包中已经包括配置好的完整的根文件系统，如果无特别需求，可直接使用。要添加自己开发的应用程序，只需将应用程序和相应的库文件拷贝到根文件系统的对应目录即可。

直接使用 SDK 提供的基本文件系统位置：

Hi3516ev200\SDK\Hi3516E

V200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\board\Hi3516EV200_SDK_V1.0
.1.2\package\ rootfs_uclibc.tgz

如有需要制作根文件系统，参考《Hi3516EV200／Hi3516EV300／Hi3518EV300／Hi3516DV200
开发环境用户指南.pdf》：

Hi3516ev200\SDK\Hi3516E V200R001C01SPC012\ReleaseDoc\zh\01.software\board\OSDRV\开发
环境用户指南.pdf

(1) 获取 busybox 源码：

成功安装 SDK 后，busybox 完整源代码就存放在 osdrv/目录中。要获取 busybox 源代码
也可以从网站 <http://www.busybox.net> 下载。

```
ixstrive@ubuntu:~/xkw/project/hi3516ev200/src$ ls
osdrv  mpp  osal  osdrv  rootfs_uclibc
osdrv.tgz  mpp.tgz  osal.tgz  osdrv.tgz  rootfs_uclibc.tgz
```

(2)配置 busybox

- cp osdrv/opensource/busybox/busybox-1.26.2/config_v100_a7_softfp_neon osdrv/opensource/busybox/busybox-1.26.2/.config //指定配置文件
config_v100_a7_softfp_neon 对应 32bit 操作系统工具链 arm-himix100-linux-
- make menuconfig 根据自己的需求选择配置

(3)编译和安装 busybox

make

make install

编译并安装成功后，在 busybox 目录下的_install 目录下生成以下目录及文件：

```
xstrive@ubuntu:~/xkw/project/hi3516ev200/src/osdrv/opensource/busybox/busybox-1.26.2/_install$ ls
bin  linuxrc  sbin  usr
```

(3)制作根文件系统

步骤 1: mkdir rootbox

cd rootbox

cp -R ../../opensource/busybox/busybox-1.26.2/_install/*.

mkdir etc dev lib tmp var mnt home proc

步骤 2:配置 etc、lib、dev 目录的必需文件

etc 目录可参考系统/etc 下的文件。其中最主要的文件包括 inittab、fstab、init.d/rcS 文件等，这些文件最好从 busybox 的 examples 目录下拷贝过来，根据需要自行修改。

dev 目录下的设备文件，可以直接从系统中拷贝过来或者使用 mknod 命令生成需要的设备文件。拷贝文件时请使用 cp -R file。

lib 目录是存放应用程序所需要的库文件，请根据应用程序需要拷贝相应的库文件。

备注：

为了便于调试，默认发布的版本中没有设置 root 密码；

为了保证系统安全，请客户在产品中自行设置 root 密码。

4.2 文件系统及驱动说明

SDK 自带驱动路径：

SDK\Hi3516EV200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\board\Hi3516EV200_SDK_V1.0.1.2\package\mpp\ko

```
xstrive@ubuntu:~/xkw/project/hi3516ev200/src/mpp/ko$ ls
extdrv          hi3516ev200_h264e.ko  hi3516ev200_vedu.ko  hi_osal.ko
hi3516ev200_acodec.ko  hi3516ev200_h265e.ko  hi3516ev200_venc.ko  hi_user.ko
hi3516ev200_adc.ko    hi3516ev200_isp.ko    hi3516ev200_vgs.ko    load3516dv200
hi3516ev200_adec.ko    hi3516ev200_ive.ko    hi3516ev200_vi.ko     load3516ev200
hi3516ev200_aenc.ko    hi3516ev200_jpege.ko  hi3516ev200_vo.ko     load3516ev300
hi3516ev200_ai.ko      hi3516ev200_pm.ko     hi3516ev200_vpss.ko   load3518ev300
hi3516ev200_aio.ko     hi3516ev200_rc.ko     hi3516ev200_wdt.ko    sys_config.ko
hi3516ev200_ao.ko      hi3516ev200_rgn.ko    hi_cipher.ko
hi3516ev200_base.ko    hi3516ev200_sys.ko    hifb.ko
hi3516ev200_chnl.ko     hi3516ev200_tde.ko    hi_mipi_rx.ko
```

拷贝驱动到根文件系统中:

```
cd rootfs_uclibc/usr/  
mkdir ext/hisi/ -p  
cp mpp/ko/* rootfs_uclibc/usr/ext/hisi/
```

添加挂载 jffs2 的目录:

```
cd rootfs_uclibc/usr/  
mkdir sampels
```

```
cd rootfs_uclibc/etc/  
ln -s ../usr/samples/resolv.conf ../etc/resolv.conf
```

4.3 jffs2 文件系统

(1)修改启动文件 autorun.sh

```
cd rootfs_uclibc/usr/samples/  
touch autorun.sh  
touch resolv.conf  
autorun.sh 文件中添加如下代码:
```

【

```
#!/bin/sh  
#ifconfig eth0 up  
#ifconfig eth0 192.168.0.200  
#route add default gw 192.168.0.1 eth0  
#echo "nameserver 114.114.114.114" > /etc/resolv.conf
```

```
ifconfig lo up  
#echo "1" > /proc/sys/net/ipv4/ip_forward  
#echo "1" > /proc/sys/net/ipv4/ip_no_pmtu_disc  
#echo "2" > /proc/sys/net/ipv4/conf/all/arp_ignore  
#echo "1" > /proc/sys/net/ipv4/conf/all/arp_announce  
#echo "0" > /proc/sys/net/ipv4/tcp_timestamps  
#echo "8192" > /proc/sys/vm/min_free_kbytes  
#echo "200" > /proc/sys/vm/vfs_cache_pressure  
#echo "2" > /proc/cpu/alignment  
#echo "1300" > /sys/class/net/eth0/mtu  
export LD_LIBRARY_PATH='/usr/samples'  
export PATH='/usr/bin:/usr/sbin:/bin:/sbin'  
ulimit -s 4096  
telnetd  
#insmod /usr/samples/reg_cmd.ko  
#insmod /usr/samples/GobiNet.ko  
cd /usr/ext/hisi && ./load3518ev300 -a -sensor0 imx327 -osmem 32M -offline
```

```

ifconfig eth0 up
ifconfig eth0 192.168.0.200
route add default gw 192.168.0.1 eth0
echo "nameserver 114.114.114.114" > /etc/resolv.conf

```

】

(2) 修改 rcS
/etc/init.d/rcS

【

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

```
/bin/mount -a
```

```
echo "
```

```

      -----
      \ _ _ _ _ _
      // \ \ \ \
      / _ _ \ - _ _
      // // //
      _ _ _ // / \ \ \ _ _ _
      _ _ _ _ _ \ \ \ _ _ _ _ _

```

```
"
```

```
for initscript in /etc/init.d/S[0-9][0-9]*
```

```
do
```

```
    if [ -x $initscript ] ;
```

```
    then
```

```
        echo "[RCS]: $initscript"
```

```
        $initscript
```

```
    fi
```

```
done
```

```
mount tmpfs /tmp -t tmpfs
```

```
#mount -t jffs2 /dev/mtdblock3 /usr/samples/
```

```
#mount -t jffs2 /dev/mtdblock4 /mnt/mtd/
```

```
cd /usr/samples/
```

```
sh autorun.sh
```

】

```
mkfs.jffs2 -r rootfs_uclibc -o rootfs.jffs2 -e 0x10000 -s 0x1000 -n
```

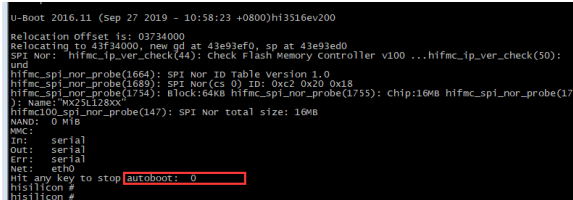
生成 rootfs.jffs2

备注：若 mkfs.jffs2 不可用，请根据提示自行下载。

5、烧写镜像文件

5.1 Uboot 命令行烧写

注意：如无必要，不用烧 uboot 镜像。



```
U-Boot 2016.11 (Sep 27 2019 - 10:58:23 +0800)hi3516ev200
Relocation Offset is: 03734000
Relocating to 43f34000, new gd at 43e93ef0, sp at 43e93ed0
SPI Nor: hifmc_ip_ver-check(44): Check Flash Memory Controller v100 ...hifmc_ip_ver-check(50): f
und
hifmc_spi_nor_probe(1664): SPI Nor ID Table Version 1.0
hifmc_spi_nor_probe(1689): SPI Nor(cs 0) ID: 0xc2 0x20 0x18
hifmc_spi_nor_probe(1754): Block:64KB hifmc_spi_nor_probe(1755): chip:16MB hifmc_spi_nor_probe(175
): Name: "w25q128k"
hifmc100_spi_nor_probe(147): SPI Nor total size: 16MB
WAND: 0 MiB
MMC:
In: serial
Out: serial
Err: serial
Net: eth0
Hit any key to stop autoboot: 0
hisilicon #
```

重启设备，stop autoboot 计时时回车，输入以下命令：

```
setenv serverip 192.168.0.209;setenv ipaddr 192.168.0.222;sa;
```

```
mw.b 0x42000000 ff 400000;tftp 0x42000000 xkw/hi3518ev300/ulmage;sf probe 0;sf erase
100000 400000;sf write 0x42000000 100000 400000
mw.b 0x42000000 ff b00000;tftp 0x42000000 xkw/hi3518ev300/rootfs.jffs2;sf probe 0;sf erase
500000 b00000;sf write 0x42000000 500000 b00000
```

```
setenv bootargs 'mem=32M console=ttyAMA0,115200 root=/dev/mtdblock2 rootfstype=jffs2 rw
init=/linuxrc mtdparts=hi_sfc:1M(boot),4M(kernel),11M(rootfs)'
setenv bootcmd 'sf probe 0;sf read 0x42000000 0x100000 0x400000;bootm 0x42000000'
sa;re
```

5.2 Hitool 烧写

选对串口几，传输方式选串口 按分区烧写，通过按添加图案按钮来添加内核，文件系统的选项，器件类型全部选 spi_nor,文件系统不选，直接是 none，Uboot 地址选 0 到 1M，内核地址选 1M 到 4M，文件系统选 5M 到 11M。点击烧写，然后板子重新上电就可以了，烧写过程可能需要 6,7 分钟。

Uboot 烧写工具：

SDK\Hi3516E V200R001C01SPC012\Hi3516EV200R001C01SPC012\01.software\pc\HiTool

Uboot 烧写参考文档《HiBurn 工具使用指南.pdf》：

SDK\Hi3516E V200R001C01SPC012\ReleaseDoc\zh\01.software\pc\HiTool

注意：

(1) uboot 选择海思 sdk 自带的 uboot，内核选择 ulmage_usbnetok，文件系统选择 3518ev300-root.jffs2.img

(2)烧完后，板子重新上电：

进入 uboot 界面 设置 uboot 参数(根据实际需要修改)：

```
setenv bootargs 'mem=32M console=ttyAMA0,115200 root=/dev/mtdblock2 rootfstype=jffs2
```

```
rw init=/linuxrc mtdparts=hi_sfc:1M(boot),4M(kernel),11M(rootfs)'
setenv bootcmd 'sf probe 0;sf read 0x42000000 0x100000 0x400000;bootm 0x42000000'
sa
依次执行以上命令，然后输入 sa 保存。
```

6、驱动加载

6.1 手动加载

刚烧写的 ulmage rootfs.squashfs 后，lsmod 查看加载驱动为空。

```
cd /usr/ext/hisi/
#insmod /usr/samples/GobiNet.ko
```

6.2 自动加载

制作 rootfs.jffs2 之前，修改~/rootfs_uclibc/etc/init.d/rcS，修改内容见 4.2 小节。

7、图像测试

7.1 sample 程序测试

Sample 路径：

编译 sample：

```
cd mpp
```

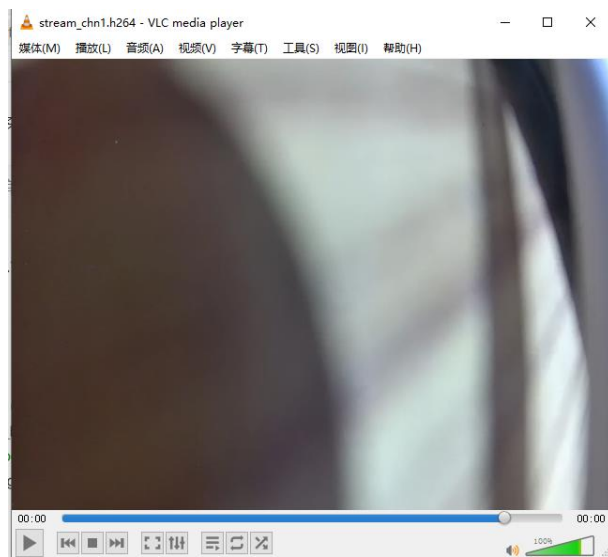
```
make
```

下载 src/mpp/sample/venc/sample_venc 到开发板

./sample_venc 0 0 //根据提示选择对应模式（比如依次选 0、c、0）

```
/usr/samples # ./sample_venc 0 0
[SAMPLE_COMM_VI_GetMipiLaneMode]-1854: support this chip 3516e200
[SAMPLE_COMM_VI_GetMipiLaneMode]-1854: support this chip 3516e200
[SAMPLE_COMM_VI_SetMipiAttr]-2098: ===== MipiDev 0, SetMipiAttr enWDRMode: 0
linear mode
=====
=====Sony imx307_2l sensor 1080P30fps(MIPI) init success!=====
=====
please input choose rc mode!
    c) cbr.
    v) vbr.
    a) avbr.
    x) cvbr.
    g) gvbr.
    f) fixqp
[SAMPLE_COMM_ISP_Thread]-376: ISP Dev 0 running !
random: crng init done
c
please input choose gop mode!
    0) NORMALP.
    1) DUALP.
    2) SMARTP.
0
please press 'q' to exit this sample, press any other key to snap one picture to file!
q
exit this sample!
program exit normally!
/usr/samples # ls
autorun.sh      sample_venc      stream_chn1.h264
resolv.conf     stream_chn0.h265  uvc_app
/usr/samples #
```

生成 stream_ch1.h264,通过 tftp 下载到 pc 端用 vlc 播放



备注:

(1) 若运行出下面错误, 确认下 load3518ev300 是否有可执行权限。

```
/usr/samples # ./sample_venc □[J5
open sys: No such file or directory
open err
: No such file or directory
open err
: No such file or directory
[SAMPLE_COMM_SYS_InitWithVbSupplement]-417: HI_MPI_VB_SetConf failed!
[SAMPLE_VENC_SYS_Init]-432: SAMPLE_COMM_SYS_GetPicSize failed!
[SAMPLE_VENC_MJPEG_JPEG]-1872: Init SYS err for 0xffffffff!
program exit abnormally!
/usr/samples # ls
□[1;32mautorun.sh□[0m □[1;32mresolv.conf□[0m □[1;32msample_venc□[0m □[1;32muvnc_app□[0m
/usr/samples #
```

(2)

```

jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afdec: 0x0020 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afdf0: 0x81b4 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afdf4: 0x1973 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afdf8: 0x6600 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afdfc: 0x4608 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afe00: 0x4608 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004afe04: 0x4608 instead
jffs2: Further such events for this erase block will not be printed
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0000: 0x3267 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0004: 0x1973 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0008: 0x3efe instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b000c: 0x361e instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0010: 0x7faa instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0014: 0x5f63 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0018: 0xc31f instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b001c: 0x067e instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0020: 0x80df instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004b0024: 0x0bc3 instead
jffs2: Further such events for this erase block will not be printed
jffs2: Node at 0x004cfd94 with length 0x0000075b would run over the end of the erase block
jffs2: Perhaps the file system was created with the wrong erase size?
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfd98: 0x075b instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfd9c: 0xaa51 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdad: 0x01c4 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdaf: 0x000d instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfda8: 0x81b4 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdac: 0x03e8 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdb0: 0xa224 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdb4: 0x4608 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdb8: 0x4608 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004cfdbc: 0x4608 instead
jffs2: Further such events for this erase block will not be printed
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0000: 0x79a8 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0004: 0xbc5e instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0008: 0xcfc6 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d000c: 0xd918 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0010: 0xbc62 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0014: 0xc77c instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0018: 0x1c09 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d001c: 0xe6b6 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0020: 0x63f6 instead
jffs2: jffs2_scan_eraseblock(): Magic bitmask 0x1985 not found at 0x004d0024: 0xa419 instead
jffs2: Further such events for this erase block will not be printed
VFS: Mounted root (jffs2 filesystem) readonly on device 31:2.
devtmpfs: mounted
Freeing unused kernel memory: 176K (c067e000 - c06aa000)
This architecture does not have kernel memory protection.
kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b

CPU: 0 PID: 1 comm: init Not tainted 4.9.37 #5
Hardware name: Generic DT based system
Backtrace:
[<0012f6c>] (dump_backtrace) from [<0013250>] (show_stack+0x18/0x1c)
r7:c243de44 r6:c05e8e68 r5:00000000 r4:c06dd2e8
[<0013238>] (show_stack) from [<02624bc>] (dump_stack+0x24/0x28)
[<0262498>] (dump_stack) from [<007867c>] (panic+0xe8/0x250)
[<0078598>] (panic) from [<001e2fc>] (do_exit+0x900/0x91c)
r3:c243e000 r2:c243e00c r1:0000000b r0:c05e8e68
r7:c243de44
[<001d9fc>] (do_exit) from [<001f12c>] (do_group_exit+0x48/0xac)
r7:c243e000
[<001f0e4>] (do_group_exit) from [<002881c>] (get_signal+0x2a8/0x520)
r5:c243c000 r4:c05e9a48
[<0028574>] (get_signal) from [<00123e8>] (do_signal+0xe0/0x400)
r10:00000000 r9:c243c000 r8:00000000 r7:10c53c7d r6:00000000 r5:c243dec4
r4:c243dfb0
[<0012308>] (do_signal) from [<00128cc>] (do_work_pending+0xad/0xb4)
r10:00000000 r9:c243c000 r8:00000000 r7:10c53c7d r6:c243dfb0 r5:00000000
r4:c243c000
[<0012828>] (do_work_pending) from [<000f4c>] (slow_work_pending+0xc/0x20)
r7:10c53c7d r6:ffffffff r5:60000010 r4:b6f6f004
---[ end kernel panic - not syncing: Attempted to kill init! exitcode=0x0000000b

```

仔细检查之后发现是制作文件系统时块大小设置错了，只要修改 mkfs.jffs2 后-e 的参数即可，例如本例，现在 spiflash 的块大小只有 64k，而我制作文件系统之时设置成 128k 即 0x20000，修改如下：

```
mkfs.jffs2 -r rootfs_uclibc -o rootfs.jffs2 -e 0x20000 -s 0x1000 -n
```

改为

```
mkfs.jffs2 -r rootfs_uclibc -o rootfs.jffs2 -e 0x10000 -s 0x1000 -n
```