君正®

Halley5 平台应用开发手册

Author: 系统软件部

Version: 1.0
Date: 2020.12.7

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

君正®

Halley5 平台应用开发手册

Release history

Date	Revision	Change
2020.12.7	1.0	First release

Disclaimer

Ingenic Semiconductor Co., Ltd.

Ingenic Headquarters, East Bldg. 14, Courtyard #10
Xibeiwang East Road, Haidian District, Beijing, China,
Tel: 86-10-56345000
Fax:86-10-56345001
Http: //www.ingenic.com

北京君正集成电路股份有限公司

地址:北京市海淀区东北西路中关村软件园二期君正总部大楼
电话: 86-10-56345000
传真: 86-10-56345001
http: //www.ingenic.com

目 录

关 键 词	文档 格式
摘　　要	本文档说明文档的基本格式。
缩 略 语	
参考资料	

1 简介

Halley5 软件平台是君正开发的一套 linux 系统发布、开发的平台。平台通过 Build.mk 文件进行指导性编译。支持如下功能。

- 支持编译 c 文件，cpp 文件。
- 支持动态库，静态库的编译
- 支持可执行 bin 文件的编译
- 支持添加第三方应用和库
- 支持模块的单独编译

1.1 目录结构介绍

平台的目录文件结构如下图所示。

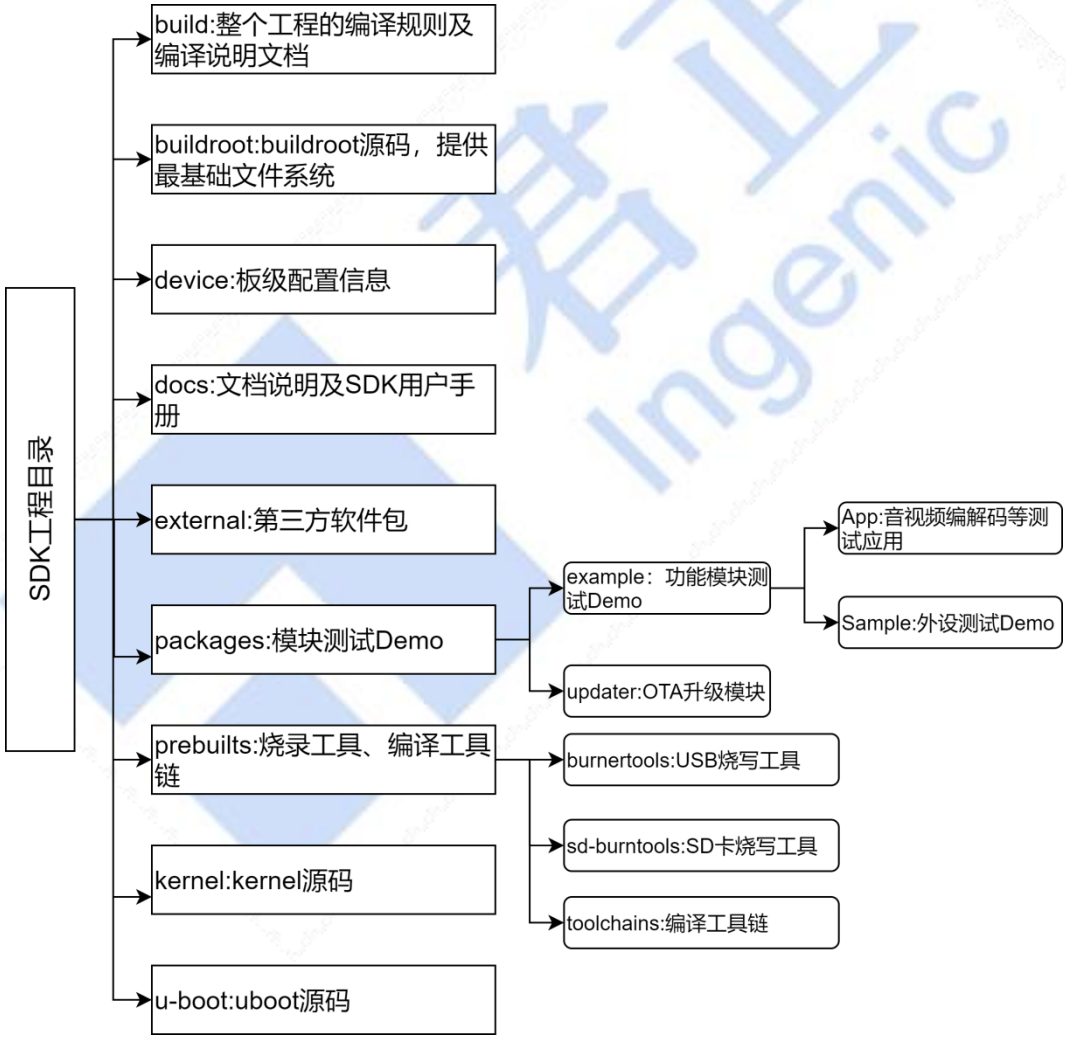


图 1-1 halley5 平台目录结构

1.2 特别注意

该文档所有内容均以 X2000 平台下的 halley5 开发板为例进行说明，开发人员可根据自己实际下载到的代码进行参考。

2 君正平台编译配置和命令介绍

平台的编译配置主要以 buildroot 中的配置为例进行说明。

2.1 君正平台默认配置

2.1.1 buildroot 中的默认配置

halley5 软件平台针对使用不同的存储介质，提供了几个默认配置，默认配置存放在 buildroot/configs/文件夹下。如下为对应板级为 halley5 的几个不同存储介质的默认配置。

halley5_msc_burn_defconfig
halley5_msc_defconfig
halley5_nand_defconfig
halley5_nand_qt_defconfig
halley5_nor_defconfig

2.1.2 buildroot 中的自定义配置

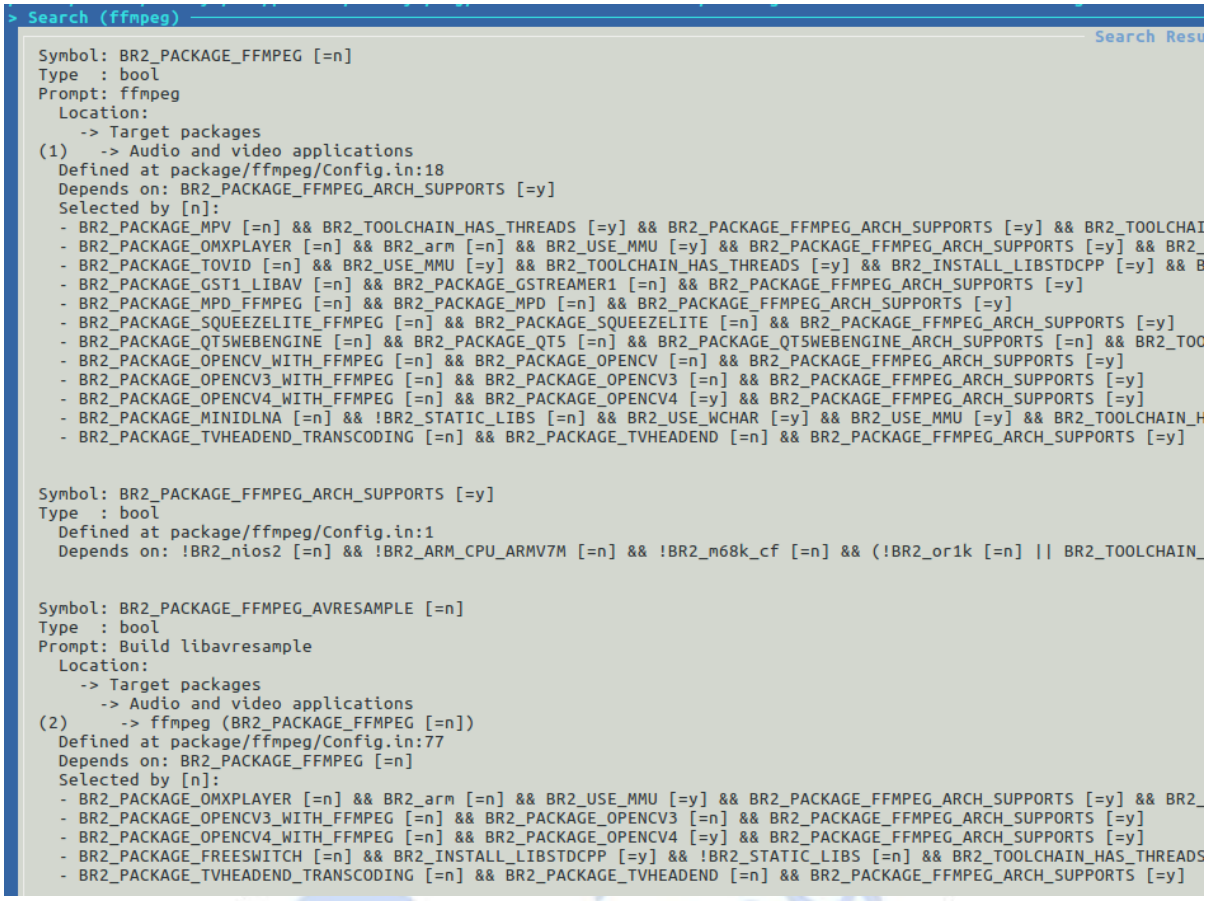
buildroot 中的默认配置提供了一些常用的工具和设置，并不能满足所有用户的需求，用户可以根据自己的需求就行自定义的配置。

比如要在 buildroot 中添加一个 ffmpeg 的工具。操作如下。

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 按”/“键进入搜索模式，在搜索框中输入 ffmpeg。搜索出来的结果如下图所示。



4） 根据搜索结果上 Location 的提示，进入 Target packages/Audio and video applications/配置项下（或按 1），然后勾选并进入 ffmpeg，根据自己的需求选择想要添加的功能。

5） 保存退出

2.2 buildroot 中的君正厂商配置

buildroot 中君正厂商配置指在 buildroot 配置中 Ingenic packages 下的一些配置，可配置的内容如下图所示。

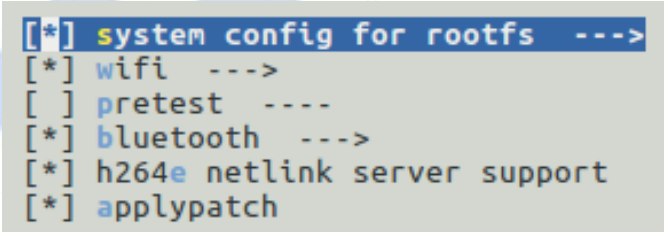


图 2-1 君正厂商配置

2.2.1 system config for rootfs 的配置

在 halley5 平台中，可以对 system config for rootfs 进行一些系统的配置。主要包括如下几项配置

表 2-1 君正厂商的 rootfs 可配置项

可配置项	说明
enable adb service	打开 adb 服务
set dns server	打开 dns 服务
cache dns	
mount usr data: mount_ubifs.sh userdata /usr/data/	开机挂载用户数据分区
enable sd burn	打开 sd 卡烧录功能
enable rootlogin when using open ssh	以管理员身份登录系统
enable usb composite gadgets	

如下图所示。

```
--- system config for rootfs
[ ]   enable adb service
[ ]   set dns server
-*-   install mount_ubifs.sh
[ ]   cache dns
[*]   mount usr data: mount_ubifs.sh userdata /usr/data/
[ ]   enable sd burn
[*]   enable rootlogin when using open ssh
[*]   enable usb composite gadgets
```

图 2-2 君正厂商配置中 rootfs 可选配置内容

具体的配置方法如下。

1） 进入 halley5 工程
2） 执行 make buildroot-menuconfig
3） 配置 Ingenic packages/system config for rootfs/
4） 选择需要配置的内容
5） 保存退出

2.2.2 wifi 配置

halley5 平台中，可以对 WIFI 进行配置，以下为 WIFI 可配置的几个内容。

表 2-2 君正厂商 WIFI 的可配置项

可配置项	说明
common wifi device	选择常见的 WIFI
select wifi chip	选择 WIFI 芯片的型号
wpa_supplicant.conf file path	设置 wpa_supplicant.conf 文件路径
hostapd.conf file path	设置 hostapd.conf 文件路径
hostap interface	设置 hostap 接口名称
wifi auto startup	配置 WIFI 是否自动启动
wifi use ap mode	配置 WIFI 是否使用热点模式
wifi mac address	设置 WIFI 的 mac 地址
wifi mac address path	配置 mac 地址文件路径

如下图所示。

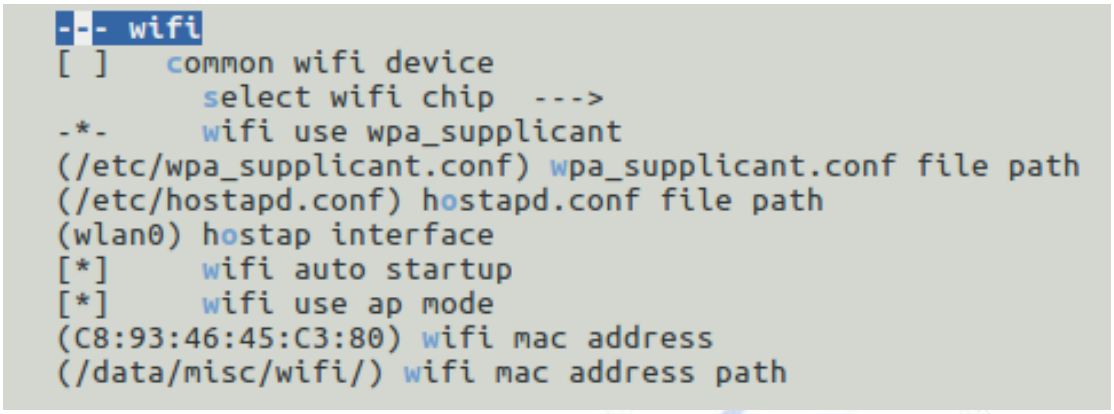


图 2-3 君正厂商配置中 wifi 可选配置项

WIFI 配置的方法如下。

1）进入 halley5 工程
2）执行 make buildroot-menuconfig
3）配置 Ingenic packages/ wifi/
4）选择想要配置的项
5）保存退出

2.2.3 pretest 配置

halley5 平台中，可以配置 pretest 对板载的模块进行测试，可以进行测试的功能如下所示。

表 2-3 君正厂商蓝牙的可配置项

配置项	说明
bluetooth test	蓝牙功能测试
wifi test	WIFI 功能测试
network test	网络功能测试
mmc test	mmc 测试，可以选择不同的测试模式
amic test	amic 测试
dmic test	dmic 测试，可以选择测试 dmic 的数量
speaker test	扬声器测试
camera test	摄像头测试，可以选择摄像头的个数
lcd test	lcd 显示测试
usb test	usb 测试，可以选择 usb 的模式 host and device
touchscreen test	触摸屏功能测试

如下图所示。

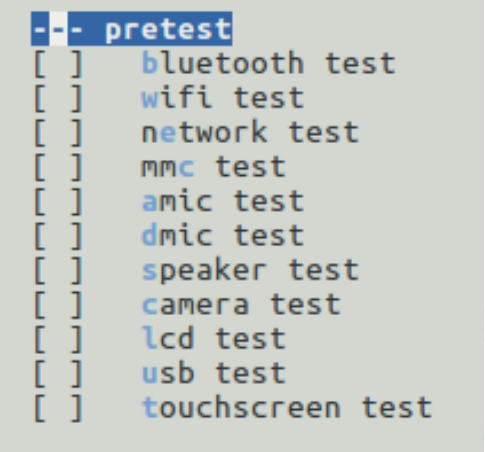


图 2-4 君正厂商配置中 pretest 的可选择配置项

配置 pretest 的方法如下。

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Ingenic packages/pretest，配置选项的路径如下图所示。

4） 选择要配置的配置项

5） 保存退出

2.2.4 bluetooth 配置

halley5 平台中，可以配置蓝牙是否支持 BSA server，以及选择使用的蓝牙芯片。具体的配置方法如下。

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Ingenic packages/bluetooth，配置选项的路径如下图所示。

```
Symbol: BR2_PACKAGE_BLUETOOTH [=y]
Type  : bool
Prompt: bluetooth
  Location:
(1) -> Ingenic packages
  Defined at package/ingenic/bluetooth/Config.in:1
```

图 2-5 君正厂商配置中 bluetooth 的配置路径

2.2.5 h264e netlink server support 配置

halley5 平台中，可以配置是否支持 h264e netlink server，具体的配置方法如下。

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Ingenic packages/h264e netlink server support，配置选项的路径如下图所示。

```
Symbol: BR2_PACKAGE_H264E_NL_SERVER [=y]
Type  : bool
Prompt: h264e netlink server support
  Location:
(1) -> Ingenic packages
  Defined at package/ingenic/h264e-nl-server/Config.in:1
```

图 2-6 君正厂商配置中 h264e netlink server support 的配置路径

2.2.6 applypatch 配置

halley5 平台中，可以配置是否使用 applypatch，具体的配置方法如下。

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Ingenic packages/applypatch，配置选项的路径如下图所示。

```
Symbol: BR2_PACKAGE_APPLYPATCH [=y]
Type  : bool
Prompt: applypatch
  Location:
(1) -> Ingenic packages
  Defined at package/ingenic/applypatch/Config.in:1
```

图 2-7 君正厂商配置中 applypatch 的配置路径

2.3 编译命令

2.3.1 编译规则

在编译之前，首先要完成编译工具链和编译配置的初始化。执行如下指令初始化编译工具链。

```
source build/envsetup.sh
```

然后执行如下指令加载编译的配置。

```
lunch
```

执行完成后会列出一些可供选择的配置，如下图所示。

```
You're building on Linux

Lunch menu... pick a combo:
     1. halley5.v10_nand_4.4.94-eng
     2. halley5.v10_nand_4.4.94-user
     3. halley5.v10_nand_4.4.94-userdebug
     4. halley5.v10_nor_4.4.94-eng
     5. halley5.v10_nor_4.4.94-user
     6. halley5.v10_nor_4.4.94-userdebug
     7. halley5.v10_msc_4.4.94-eng
     8. halley5.v10_msc_4.4.94-user
     9. halley5.v10_msc_4.4.94-userdebug
     10. halley5.v10_nand_4.4.94_ota-eng
     11. halley5.v10_nand_4.4.94_ota-user
     12. halley5.v10_nand_4.4.94_ota-userdebug
     13. halley5.v20_nand_4.4.94-eng
     14. halley5.v20_nand_4.4.94-user
     15. halley5.v20_nand_4.4.94-userdebug
     16. halley5.v20_nor_4.4.94-eng
     17. halley5.v20_nor_4.4.94-user
     18. halley5.v20_nor_4.4.94-userdebug
     19. halley5.v20_msc_4.4.94-eng
     20. halley5.v20_msc_4.4.94-user
     21. halley5.v20_msc_4.4.94-userdebug
     22. halley5.v20_nand_4.4.94_ota-eng
     23. halley5.v20_nand_4.4.94_ota-user
     24. halley5.v20_nand_4.4.94_ota-userdebug
     25. halley5.v20_msc_4.4.94_burn-eng
     26. halley5.v20_msc_4.4.94_burn-user
     27. halley5.v20_msc_4.4.94_burn-userdebug

Which would you like? [halley5.v10_nand_4.4.94-eng]
```

配置的选择请参考《Halley5 软件平台快速开发指南》

如下为平台编译的一些常用命令。进入 halley5 软件平台根目录，执行下列指令编译相应的内容。

表 2-4 halley5 平台编译命令

编译命令	说明
make	整体编译工程
make buildroot	单独编译 buildroot
make buildroot-menuconfig	配置 buildroot
make buildroot-clean	清除 buildroot 编译产生的一些文件
make buildroot-distclean	彻底清除 buildroot 编译产生的文件，包括下载的库
make buildroot-<packname>	编译 buildroot 内的软件包
make kernel	单独编译 kernel
make kernel-menuconfig	配置 kernel
make kernel-clean	清除 kernel 编译产生的一些文件
make kernel-distclean	彻底清除 kernel 编译产生的文件
make uboot	单独编译 uboot
make uboot-menuconfig	配置 uboot
make uboot-clean	清除 uboot 编译产生的一些文件
make uboot-distclean	彻底清除 uboot 编译产生的文件
make (module naem)	单独编译模块

2.3.2 编译生成的文件

编译完成后，生成 out 目录,结构如下：

```
product
    └─── halley5              // 板级
        ├─── image        // 生成的目标文件（uboot、kernel、文件系统）
        ├─── obj              //模块编译过程中生成的中间文件及目标文件（strip 前）
        ├─── sysroot          //文件系统（strip 前）
        └─── system       //文件系统（包含 strip 后的模块）
```

单独编译 buildroot 后，生成的 rootfs 会存放在 obj/buildroot-intermediate/images/目录下，当执行 ”make“ 指令整体编译后，会将生成的 rootfs 搬运到 inage 目录下。

2.3.3 单独编译模块的方法

单个模块编译规则如下，在工程的顶层目录下执行：

```
make $(LOCAL_MODULE)
```

即 make “模块名” 就可单独编译单个模块，以 packages/example/App/cimutils 下的 cimutils 测试模块为例，此模块 LOCAL_MODULE（Build.mk 中）赋值为：cimutils，执行如下命令即可单独编译 cimutils 模块：

```
make cimutils
```

2.3.4 单独编译 buildroot 包的方法

在 halley5 平台的 buildroot/packages/目录下，有一些可供选择的软件包。当需要使用这些软件包时，可以通过如下方式进行编译。

```
make buildroot-<packname>
```

如要编译 buildroot/packages/下的 ffmpeg，执行如下指令编译。

```
make buildroot-ffmpeg
```

2.3.5 模块内容改动后编译的方法

当模块编译完之后，又对模块的内容进行了修改，使用单独编译模块的方法可能会导致更改的内容不会更新到最终生成的文件系统中。最有效的方法就是删掉模块编译产生的中间文件，然后重新编译模块。

下面以 cimutils 模块为例，介绍模块内容改动之后的编译方法。

1） 模块内容改动后，需删除 out 目录下的中间文件，再进行编译。Cimutils 产生的中间文件路径为: out/product/halley5/obj/DEPANNER/cimutils-intermediate,删除以 cimutils 为前缀的目录

2） 执行单独编译模块的指令编译模块。即：make cimutils

3） 执行 make 指令，将生成的内容更新到 out/product/halley5/image/下的系统镜像中。

注： packages 下软件包生成的中间文件路径一般在 out/product/halley5/obj/DEPANNER/ 或者 out/product/halley5/obj/out/product/halley5/obj/packages/example/App/目录下。buildroot/下的软件包生成路径一般在 out/product/halley5/obj/buildroot-intermediate/build/目录下。

3 君正平台应用介绍

君正平台应用指的是君正自己开发的一些应用。如下表所示。

表 3-1 君正平台应用

平台应用	说明
v4l2-jpegenc	jpeg 编码
v4l2-jpegdec	jpeg 解码
v4l2-h264enc	H264 编码
v4l2-h264dec	H264 解码
cimutils	图片预览和拍照接口
webcam_gadget	UVC 输出接口
v4l2-isp-tunning	亮度，对比度，饱和度，锐度的调节接口
mjpg-streamer	双目摄像头应用
stereo	用于 mjpg-streamer 中的标定
linphone	视频通话应用

3.1 v4l2-jpegenc

3.1.1 功能简介

v4l2-jpegenc 用于 VPU JPEG 的编码，用于将 nv12、nv21 和 tile420 格式文件转换为 JPEG 格式文件。

- 源码路径：packages/example/App/v4l2-jpegenc/src
- 软件流程图如下：

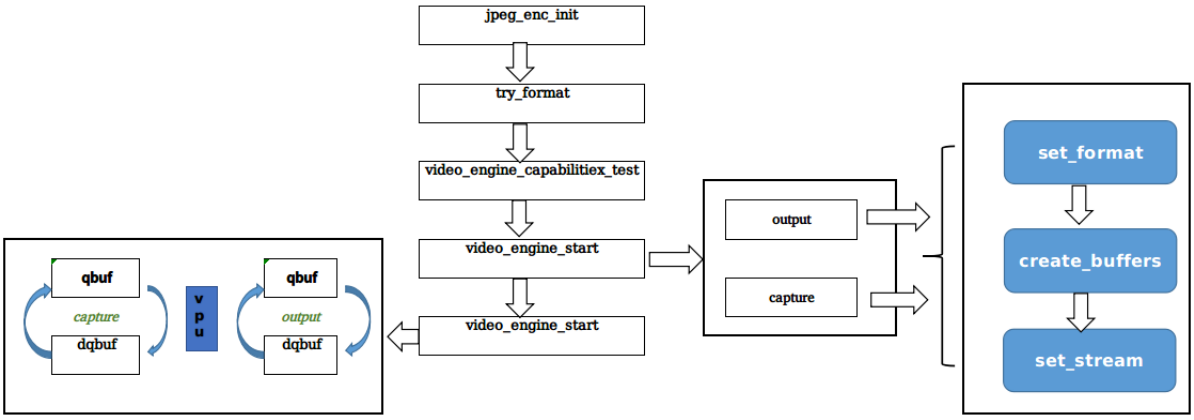


图 3-1 君正平台应用 V4l2-jpegenc 软件流程图

3.1.2 使用方法

3.1.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make v4l2-jpegenc
```

```
make
```

模块编译完成后，生成可执行文件 v4l2-jpegenc，存放在 out/product/halley5/system/usr/bin/ 目录下。

3.1.2.2 模块使用

在 halley5 开发板上运行如下指令运行模块。

```
v4l2_jpegenc -v /dev/video1 -f filename.yuv -w 1280 -h 720 -o /tmp/output.jpg
```

参数：

```
-v  指定 helex 的设备节点
-f  指定 yuv 文件
-w 指定 yuv 文件的宽
-h  指定 yuv
-o 指定输出文件路径和名称
```

测试结果：

查看编码后的 output.jpg 文件是否显示正常。

3.2 v4l2-jpegdec

3.2.1 功能简介

v4l2-jpegdec 用于 VPU JPEG 的解码，用于将 JPEG 格式文件转换为 nv12 格式文件。

- 源码路径：packages/example/App/v4l2-jpegdec/src
- 软件流程图如下：

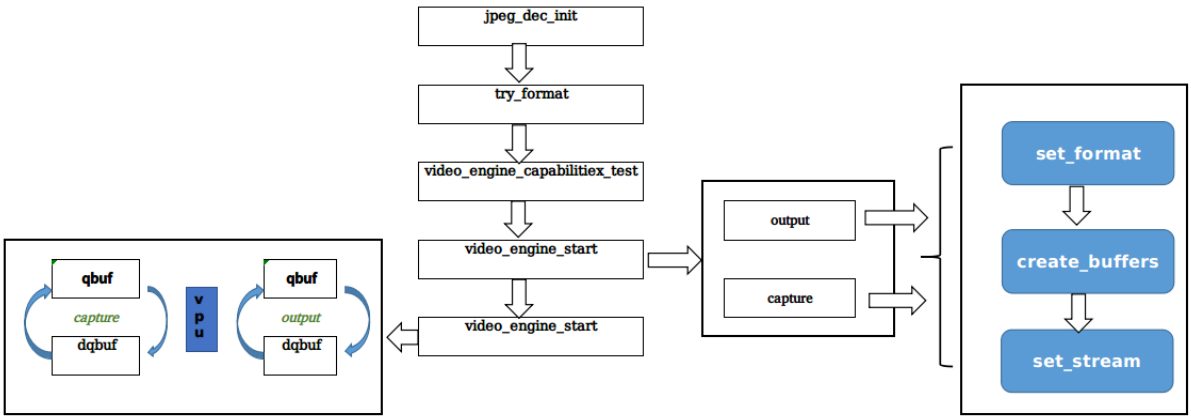


图 3-2 君正平台应用 V4l2-jpegdec 软件流程图

3.2.2 使用方法

3.2.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make v4l2-jpegdec
make
```

模块编译完成后，生成可执行文件 v4l2-jpegdec，存放在 out/product/halley5/system/usr/bin/ 目录下。

3.2.2.2 模块使用

在 halley5 开发板上执行如下指令运行模块。

```
v4l2_jpegdec -v /dev/video1 -f filename.jpg  -w 1280  -h 720 -o /tmp/output.yuv
```

参数：

```
-v  指定 helix 的设备节点
-f  指定 JPG 文件
-w 指定 JPG 文件的宽度
-h 指定 JPG 文件的高度
-o 指定输出文件路径和名称
```

测试结果：

查看解码后的 output.yuv 文件是否显示正常。

3.3 v4l2-h264enc

3.3.1 功能简介

V4l2-h264enc 用于 VPU H264 的编码，用于将 nv12、nv21 和 tile420 格式文件转换为 h264 格式文件。

- 源码路径：packages/example/App/v4l2-h264enc/src
- 软件流程图如下：

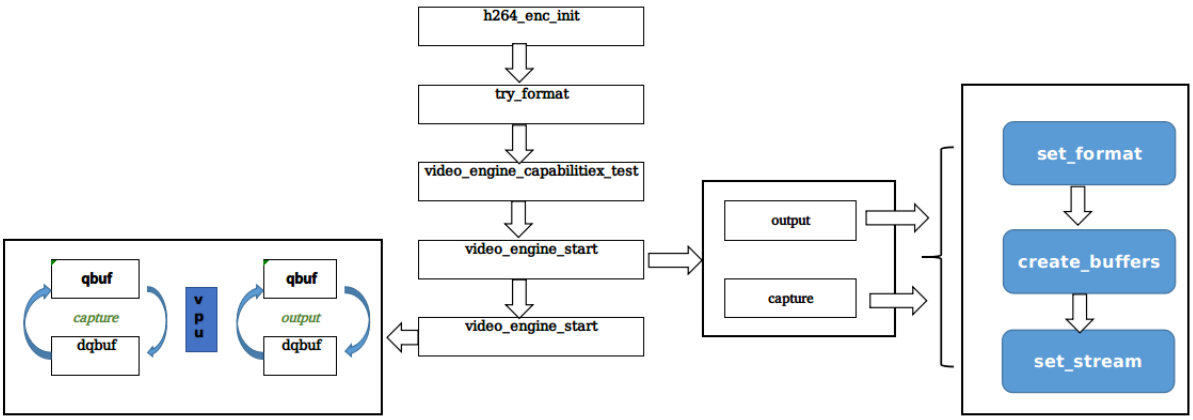


图 3-3 君正平台应用 v4l2-h264enc 软件流程图

3.3.2 使用方法

3.3.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make v4l2-h264enc
make
```

模块编译完成后,生成可执行文件 v4l2-h264enc，存放在 out/product/halley5/system/usr/bin/ 目录下。

3.3.2.2 模块使用

在 halley5 开发板上执行如下指令运行模块。

```
v4l2_h264enc -t nv12 -v /dev/video1 -f filename.yuv -w 1280 -h 720 -o /tmp/output.h264
```

参数：

```
-t  指定输入文件格式
-v  指定 helix 的设备节点
-f  指定 yuv 文件
-w 指定 yuv 文件的宽度
-h 指定 yuv 文件的高度
-o 指定输出文件路径和名称
```

测试结果：

查看编码后的 output.h264 文件是否显示正常

3.4 v4l2-h264dec

3.4.1 功能简介

v4l2-h264dec 用于 VPU h264 的解码，用于将 h264 格式文件转换为 nv12、nv21 格式文件。

源码路径：packages/example/App/v4l2-h264enc/src

软件流程图如下：

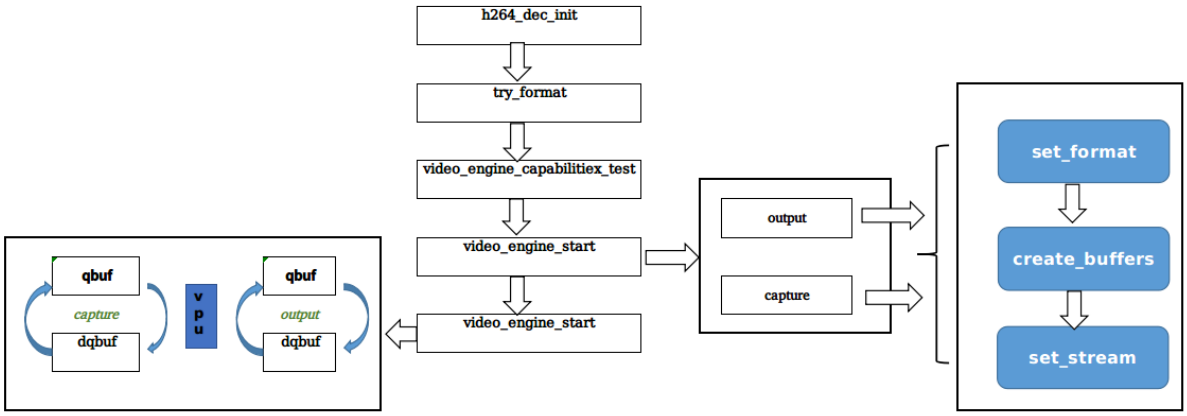


图 3-4 君正平台应用 V4l2-h264dec 软件流程图

3.4.2 使用方法

3.4.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make v4l2-h264dec
make
```

模块编译完成后,生成可执行文件 v4l2-h264dec，存放在 out/product/halley5/system/usr/bin/目录下。

3.4.2.2 模块使用

在 halley5 开发板上执行如下指令运行模块。

```
v4l2_h264dec -t nv12 -v /dev/video2 -f filename.h264 -w 1280 -h 720
```

参数：

```
-t  指定输出文件的格式
-v  指定 felix 的设备节点
-f  指定 h264 文件
-w 指定 h264 文件的宽度
-h 指定 h264 文件的高度
```

测试结果：

查看 lcd 屏幕图片是否显示正常

3.5 cimutils

3.5.1 功能简介

cimutils 测试 cim/isp camera 图像预览和拍照，支持图片 JPEG、raw 和 bmp 格式输出

- 源码路径：packages/example/App/cimutils
- 软件流程图如下：

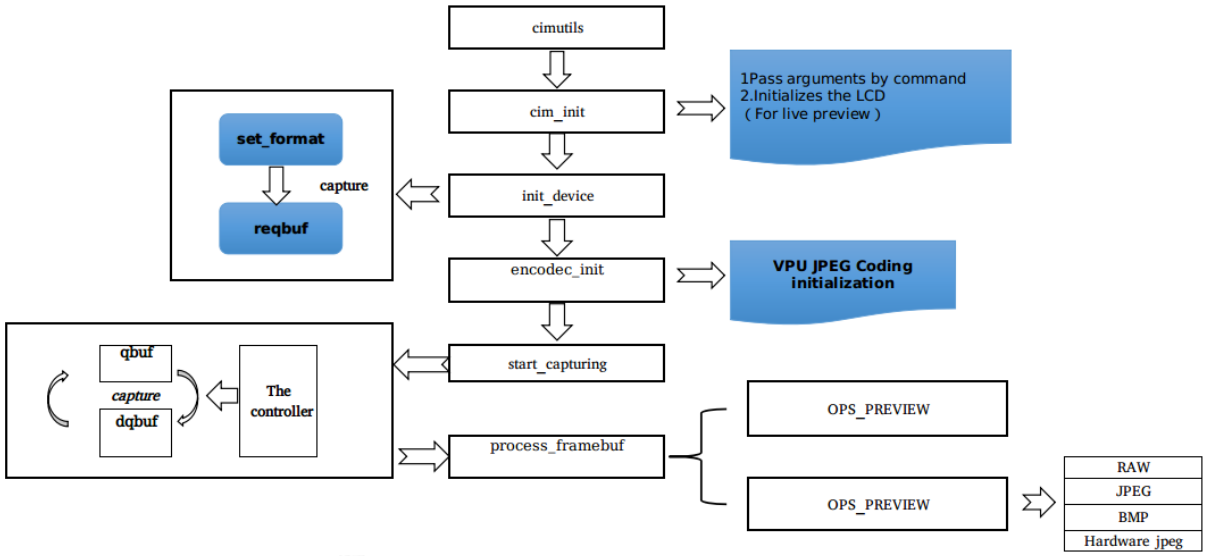


图 3-5 君正平台应用 Cimutils 软件流程图

3.5.2 使用方法

3.5.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make cimutils
```

```
make
```

模块编译完成后,生成可执行文件 cimutils，存放在 out/product/halley5/system/usr/bin/目录下。

3.5.2.2 模块使用

在 halley5 开发板上执行如下指令运行模块。

```
cimutils  -v cim -E helix -C -f filename.jpg -t nv12 -x 320 -y 240
```

参数：

```
-v 指定 isp/cim 节点
-E 指定 helix 硬件编码
-f 指定输出文件类型
-t 指定输入格式
-x 指定图片宽度
-y 指定图片高度
```

测试结果：

查看 filename.jpg 文件是否显示正常

3.6 webcam_gadget

3.6.1 功能简介

webcam_gadget 测试 UVC 输出

- 源码路径：packages/example/App/usb-gadget/webcam
- 软件流程图如下：

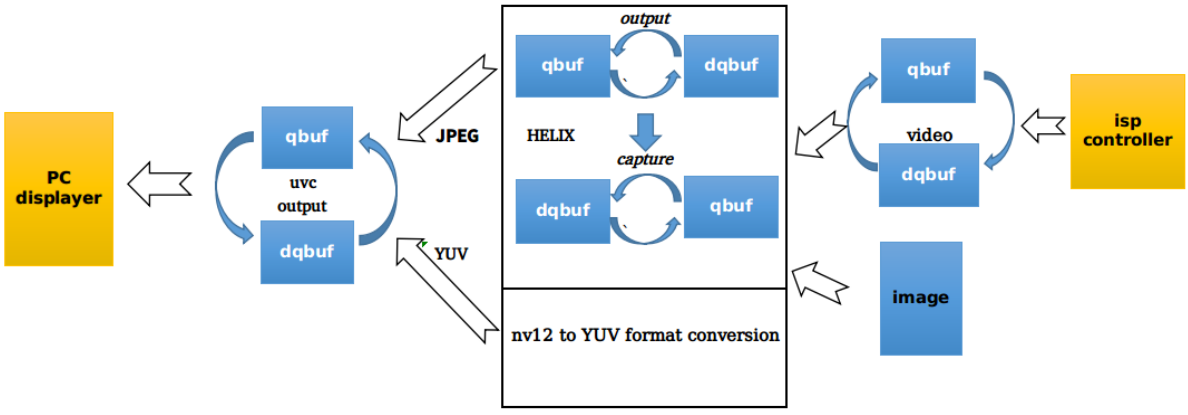


图 3-6 君正平台应用 webcam_gadget 软件流程图

3.6.2 使用方法

3.6.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make webcam_gadget
make
```

模块编译完成后,生成 webcam_gadget，存放在 out/product/halley5/system/usr/sbin/目录下。

3.6.2.2 模块使用

- 静态图片测试命令：

```
webcam_gadget -u /dev/video7 -v /dev/video4 -e /dev/video0 -d -i /1280x720.yuv
```

- Camera 采集测试命令：

```
webcam_gadget -u /dev/video7 -v /dev/video4 -e /dev/video0
```

参数：

```
-b              Use bulk mode
-d              Do not use any real V4L2 capture device
-h              Print this help screen and exit
-i              images dir for [uvc-WxH.jpg uvc-WxH.yuv]
-m           Streaming mult for ISOC (b/w 0 and 2)
-n            Number of Video buffers (b/w 2 and 32)
-o       <IO method> Select UVC IO method:
                        0 = MMAP
                        1 = USER_PTR
-s       <speed>     Select USB bus speed (b/w 0 and 2)
                    0 = Full Speed (FS)
                    1 = High Speed (HS)
                    2 = Super Speed (SS)
-t              Streaming burst (b/w 0 and 15)
-u device       UVC Video Output device
-v device       V4L2 Video Capture device
-e device       HELIX Video Capture device
```

测试结果：

查看 pc 端是否显示正常

3.7 v4l2-isp-tunning

3.7.1 功能介绍

v4l2-isp-tuning 的主要功能是操作 mscaler 某通道的节点，调节图像的亮度，锐度，饱和度和对比度，并通过同一 msclaer 的另一通道节点观察调节效果。

- 源码路径：packages/example/App/v4l2-isp-tuning
- 软件流程图如下：

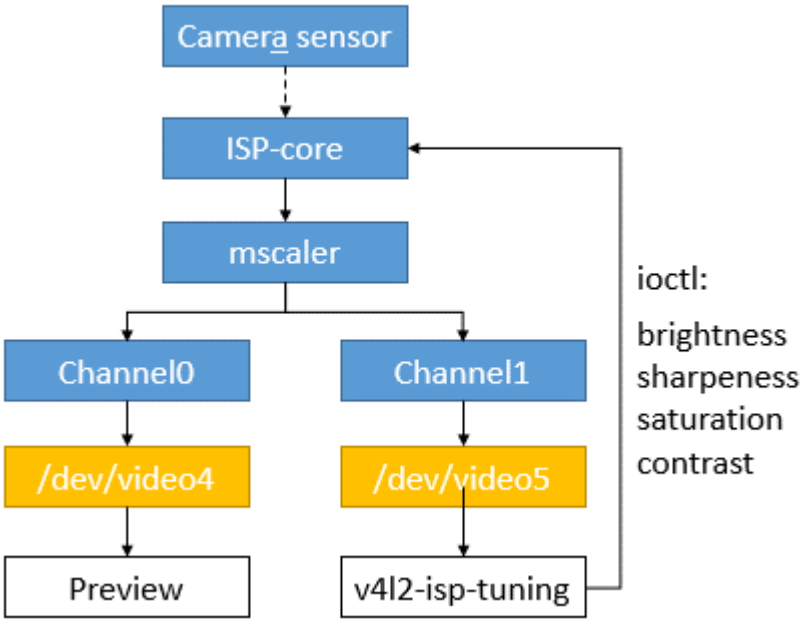


图 3-7 君正平台应用 v4l2-isp-tunning 软件流程图

3.7.2 使用方法

3.7.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行模块编译。

```
make v4l2-isp-tunning
make
```

模块编译完成后,生成 v4l2-isp-tuning，存放在 out/product/halley5/system/usr/bin/目录下。

3.7.2.2 模块使用

1）配置内核驱动 drivers/media/platform/ingenic-isp/isp-drv.h 文件中宏定义 MSCALER_MAX_CH 的值为 2，打开 msclaer 的两个通道，重新编译并启动；

2） 驱动加载成功后会生成 mscaler 对应的两个通道节点（mscaler0： /dev/video4 /dev/video5 mscaler1：/dev/video8 /dev/video9）；

3） 通过 mscaler 两个通道节点中的一个（例如/dev/video4）预览图像效果，可使用 ffmpeg，mjpeg-streamer 等平台工具，或使用用户自定义的预览直播工具；

4) 执行 v4l2-isp-tuning 操作 mscaler 两个通道节点中的另一个（例如/dev/video5），调整效果参数，并观察预览图像效果的变化，直至获得满意的效果，命令行及参数示意如下：

```
v4l2-isp-tuning -w 640 -h 480 -i 5 -s 128 -c 128 -S 128 -b 128
```

参数：

```
-w 指定图像宽度；
-h 指定图像高度；
-i 指定操作的 video 节点号；
-s 指定锐度，最小值 0，最大值 255，默认值 128；
```

```
-c 指定对比度，最小值 0，最大值 255，默认值 128；
-S 指定饱和度，最小值 0，最大值 255，默认值 128；
-b 指定亮度，最小值 0，最大值 255，默认值 128；
```

5) 用户可仿照 v4l2-isp-tuning 源码中效果参数配置的方法（标准 ioctl），在自定义的应用中将效果参数配置为想要的值。

3.8 mjpg-streamer

3.8.1 功能简介

mjpeg-streamer 是一个展示君正双目摄像头的应用

- 源码路径：packages/example/App/mjpg-streamer/mjpg-streamer-experimental

3.8.2 使用方法

3.8.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行编译。

```
make mjpg-streamer
make
```

模块编译完成后,生成 mjpg-streamer，存放在 out/product/halley5/system/usr/bin/目录下。同时，会生成一些库文件，存放在 out/product/halley5/system/usr/lib/mjpg-streamer 目录下。

3.8.2.2 模块使用

1）执行如下指令给板子配置网络

```
Ifconfig eth0 192.168.4.49
```

IP 需根据自己的情况进行配置，要确保其他设备可以 ping 通板子

2）在 halley5 开发板上执行如下指令开启摄像头的图像采集

```
mjpg_streamer -i "/usr/lib/mjpg-streamer/input_syncframes.so -r 1280x720" -o
"/usr/lib/mjpg-streamer/output_http.so -w /usr/share/mjpg-streamer/www"
```

参数说明：

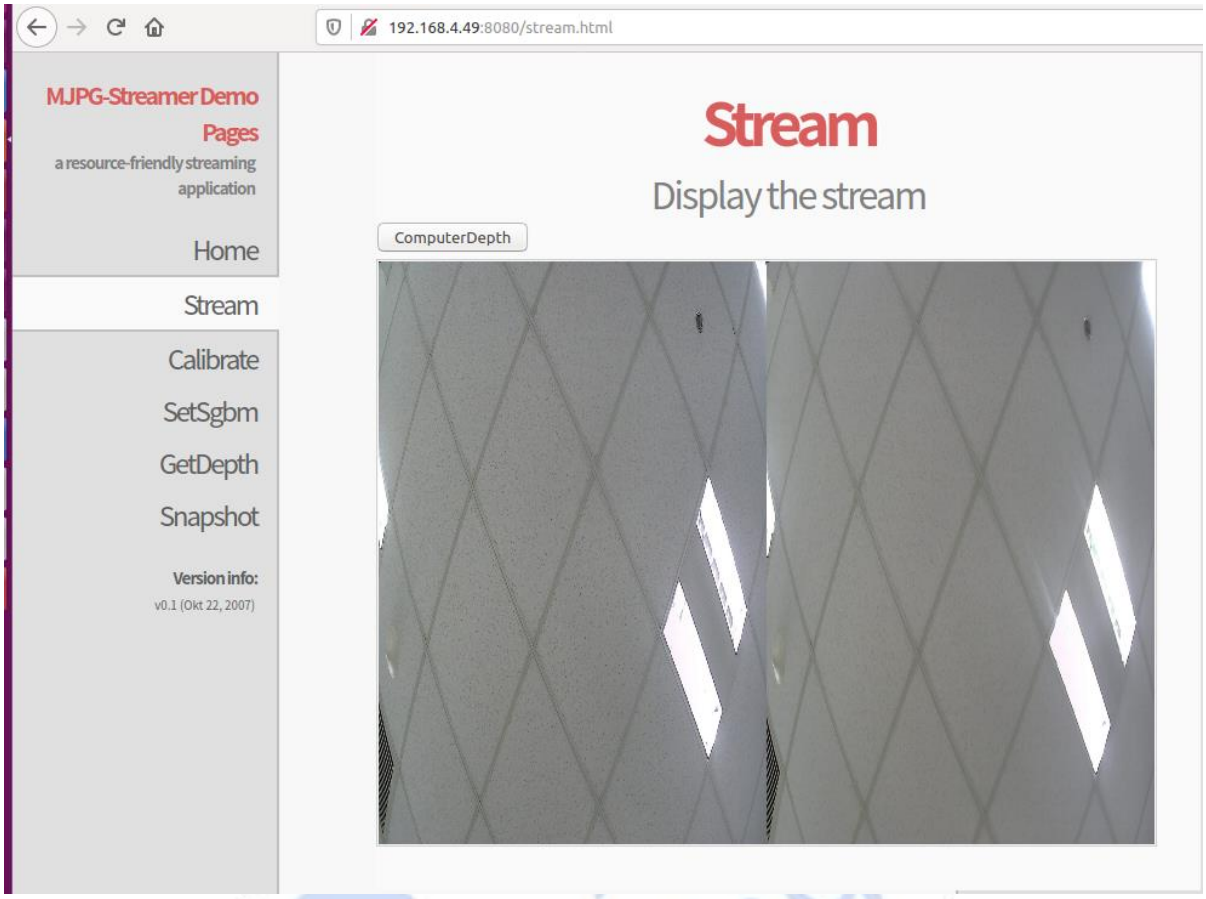
```
-i input
-o output
-r 指定图像缩放后的大小
-w 网页输出
```

3）使用手机或电脑等设备连接目标板查看图像

启动设备端的浏览器,在网址栏中输入目标板的 ip 和端口号（8080）

```
http://192.168.4.49:8080
```

能够在设备端看到双目摄像头采集的图像，如下图所示。



3.9 stereo

3.9.1 功能介绍

stereo 是 mjpg-streamer 中用于标定的一个库

- 源码路径：packages/example/App/stereo_demo

3.9.2 使用方法

在 mjpg-streamer 标定功能中被调用

3.9.2.1 模块编译

当编译 mjpg-streamer 时，会先编译该模块，一般不用单独编译该模块，单独编译该模块的方法为：进入 halley5 平台的根目录，执行如下指令进行编译。

```
make stereo-demo
```

3.10 linphone

3.10.1 功能介绍

linphone 是一个视频通话的应用

- 源码路径：halley5/packages/example/App/linphone-desktop

3.10.2 使用方法

3.10.2.1 模块编译

进入 halley5 平台的根目录，执行如下指令进行编译。

```
make linphone
make
```

3.10.2.2 模块使用

```
linphonec -V
```

参数：

```
-C 使能摄像头，但不将图像显示到 LCD 上
-D 使能 LCD 显示
-V 使能 LCD 显示和摄像头功能，类似于 linphone -C -D 的组合
```

4 向目标文件系统添加和删除应用程序

4.1 device base.mk 介绍

device_base.mk 用于配置工程编译的模块。位于 device/halley5（板级）/目录下。其中的宏 PRODUCT_MODULES 用于添加或删减通用工程编译的模块。在该文件中还有针对不同文件系统和存储介质的默认配置。这些都可以根据自己的需求进行个性化配置。

4.2 rootfs-overlay 文件夹介绍

rootfs-overlay 文件夹在 device/halley5（板级）/目录下，该文件夹下的文件在编译生成文件系统时，会替换文件系统中的同名文件。从而可以个性化的配置自己的文件系统。

5 向开发板导入和运行应用程序

5.1 使用 adb 工具导入应用程序

5.1.1 配置 adb 工具

在 halley5 平台中可以直接配置安装 adb 工具，方法如下：

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Ingenic packages/system config for rootfs/enable adb service,如下图所示。

```
Symbol: BR2_SYSTEM_CONFIG_ADB_DEVICES_NAME_PREFIX [=]
Type  : string
Prompt: adb devices name prefix
  Location:
    -> Ingenic packages
      -> system config for rootfs (BR2_PACKAGE_SYSTEM_CONFIG [=y])
(2)     -> enable adb service (BR2_SYSTEM_CONFIG_ENABLE_ADB [=n])
  Defined at package/ingenic/system_config/Config.in:26
  Depends on: BR2_PACKAGE_SYSTEM_CONFIG [=y] && BR2_SYSTEM_CONFIG_ENABLE_ADB [=n]
```

图 5-1 buildroot 中配置 adb 工具的路径

PC 端安装 adb 工具（以 Ubuntu16.04 为例）：

安装命令为：

```
sudo apt-get install adb
```

安装完成后执行 adb version 查看 adb 版本，如果正常显示版本信息说明安装成功。

5.1.2 adb 工具使用

1） 传输文件到板子

```
adb push 本地路径（pc 端要传输到板子上文件的路径） 目标路径（文件存放在板子上的路径）
```

2） 从板子上下载文件到 pc

```
adb pull 目标路径（文件在板子上的路径） 本地路径（pc 端要保存文件的路径）
```

3） adb 常用指令：

表 5-1 adb 常用命令

adb 常用命令	说明
adb help	查看 adb 的命令帮助
adb devices	查看设备
adb shell	进终端
adb kill-server	停止服务
adb start-server	打开服务
adb push 本地路径 目标路径	上传文件到设备
adb pull 目标路径 本地路径	从设备下载文件到 PC

5.2 使用 sshd 导入应用程序

5.2.1 配置 sshd 工具

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Target packages/Networking applications/openssh，如下图所示。

```
Symbol: BR2_PACKAGE_OPENSSH [=y]
Type  : bool
Prompt: openssh
  Location:
    -> Target packages
(1)   -> Networking applications
  Defined at package/openssh/Config.in:1
  Depends on: BR2_USE_MMU [=y]
  Selects: BR2_PACKAGE_OPENSSL [=y] && BR2_PACKAGE_ZLIB [=y]
  Selected by [n]:
  - BR2_PACKAGE_SSHFS [=n] && BR2_USE_WCHAR [=y] && BR2_TOOLCHAIN_HAS
  - BR2_PACKAGE_MOSH [=n] && BR2_PACKAGE_PROTOBUF_ARCH_SUPPORTS [=y]
```

图 5-2 buildroot 中配置 sshd 工具的路径

5.2.2 sshd 工具的使用

使用 sshd 工具之前，请先确保板子和要通讯的主机是能够相互 ping 通的。

sshd（secure shell）服务使用 ssh 协议远程开启其他主机 shell 的服务。使用时，通过 pc 打开板子的一个 shell 服务。登录板子的指令如下。

```
ssh -X 远程主机用户@远程主机 ip
```

如板子上登录的用户名为 root，ip 为 192.168.4.49，登录的方法为：

```
ssh -X root@192.168.4.49
```

登录的密码默认是 123456，可以在 buildroot 中进行设置，方法如下。

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 System configuration/Enable root login with password/Root password

如下图为成功登录的指令操作。

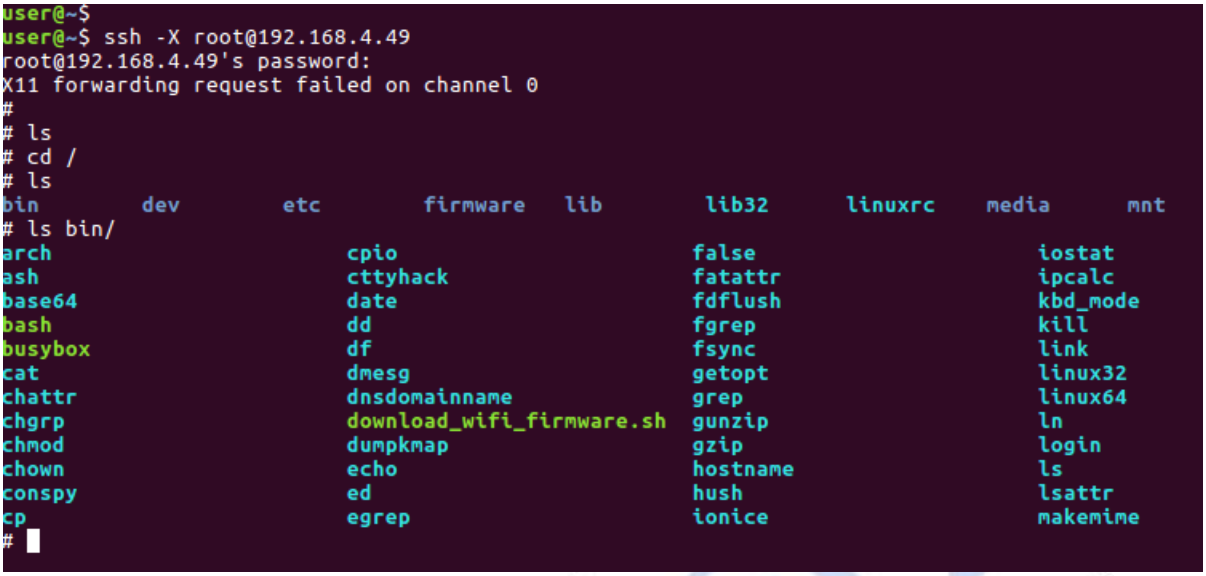


图 5-3 使用 ssh 工具成功登录板子截图

5.3 使用 telnetd 导入应用程序

telnetd 服务在配置文件中已经进行了添加，无需再进行配置

5.3.1 telnetd 使用方法

使用 telnet 工具之前，请先确保板子和要通讯的主机是能够相互 ping 通的。

telnet 命令通常用来远程登录。使用的方法如下。

```
telnet [参数]  [主机]
```

其中可选的参数如下所示。

```
-8 允许使用 8 位字符资料，包括输入与输出。
-a 尝试自动登入远端系统。
-b <主机别名>使用别名指定远端主机名称。
-d 启动排错模式。
-E 滤除脱离字符。
-f 此参数的效果和指定“-F”参数相同。
-F 使用 Kerberos V5 认证时，加上此参数可把本地主机的认证数据上传到远端主机。
-k <域名>使用 Kerberos 认证时，加上此参数让远端主机采用指定的领域名，而非该主机的域名。
-K 不自动登入远端主机。
-l <用户名称> 指定要登入远端主机的用户名称。
-L 允许输出 8 位字符资料。
-n <记录文件>指定文件记录相关信息。
-r 使用类似 rlogin 指令的用户界面。
-x 假设主机有支持数据加密的功能，就使用它。
```

如登录用户名为 root，ip 为 192.168.4.49 的主机时执行如下操作

```
telnet 192.168.4.49
```

然后按照提示输入用户名和密码即可。

5.4 使用 tftp 导入应用程序

5.4.1 配置 tftp 工具

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 然后配置 Target package/Networking applications/dnsmasq/tftp support，配置如下图：

```
Symbol: BR2_PACKAGE_DNSMASQ_TFTP [=y]
Type  : bool
Prompt: tftp support
  Location:
    -> Target packages
      -> Networking applications
(2)     -> dnsmasq (BR2_PACKAGE_DNSMASQ [=y])
  Defined at package/dnsmasq/Config.in:11
  Depends on: BR2_PACKAGE_DNSMASQ [=y]
```

图 5-4 buildroot 中配置 tftp 工具的路径

5.4.2 tftp 工具使用

使用 tftp 工具之前，请先确保板子和要通讯的主机是能够相互 ping 通的。

1）下载文件

板子作为 client 端，使用下面的命令下载文件到板子中

```
tftp -g -l 目标文件名 -r 源文件名 服务器地址
```

如想要把服务器端的一个文件名为 A.txt 的文件下载到板子端，并将新下载的文件重命名为 B.txt，使用如下指令。

```
tftp -g -l B.txt -r A.txt 192.168.4.50
```

2）上传文件

从板子（clinet）上传文件到 Server 时，使用下面的命令

```
tftp -p -r 目标文件名 -l 源文件名 服务器地址
```

如从板子上传文件 C.txt 到 Server 的 tftp 根目标下，并更名为 D.txt 使用如下指令

```
tftp -p -r D.txt -l C.txt 192.168.4.50
```

3）参数说明

```
-l      local 的缩写，后跟存在于 Client 的源文件名，或下载 Client 后重命名的文件名；
-r      remote 的缩写，后跟 Server 即 PC 机 tftp 服务器根目录中的源文件名，或上传 Server 后重命名后的文件名；
-g      get 的缩写，下载文件时用；
-p      put 的缩写，上传文件时用。
```

5.5 使用 lrzsz 导入应用程序

5.5.1 配置 lrzsz 工具

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 然后配置 Target package/Networking applications/lrzsz，配置如下图：

```
BR2_PACKAGE_LRZSZ:

Portable and fast implementation of the X/Y/Zmodem protocols.

http://www.ohse.de/uwe/software/lrzsz.html

Symbol: BR2_PACKAGE_LRZSZ [=y]
Type  : bool
Prompt: lrzsz
  Location:
    -> Target packages
      -> Networking applications
  Defined at package/lrzsz/Config.in:1
  Depends on: !BR2_STATIC_LIBS [=n]
```

图 5-5 buildroot 中配置 lrzsz 工具的路径

5.5.2 使用 lrzsz 工具（minicom）

1） sz 命令（发送文件）

eg：sz filename(传输的文件默认再打开 minicom 的路径下面)

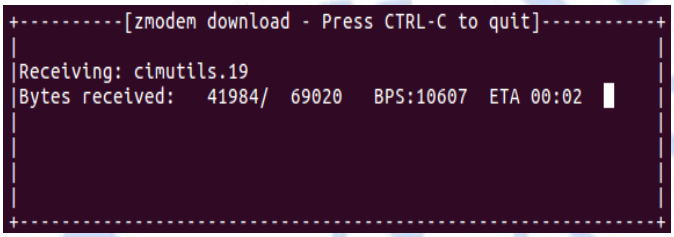


图 5-6 使用 lrzsz 工具发送文件截图

2） rz 命令（接收文件）

➢ 首先输入 rz：

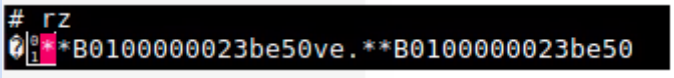


图 5-7 lrzsz 工具的 rz 命令

➢ 按 ctrl+A+Z，再按下 S（Send file）

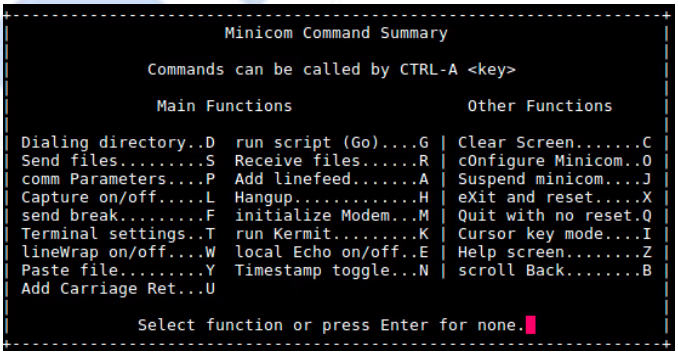


图 5-8 使用 lrzsz 工具时 minicom 的配置截图

➢ 选择 zmodem，进入文件选择界面

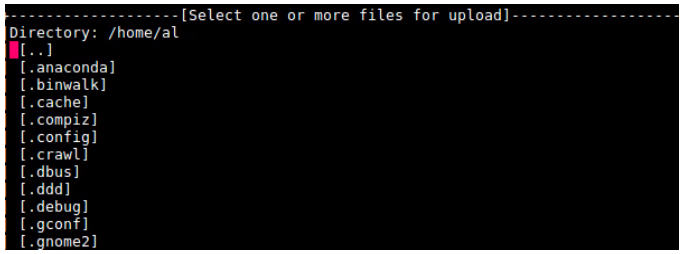


图 5-9 使用 lrzsz 工具时文件选择页面截图

- 上下键或者 PageUp/PageDown 移动，空格选择文件

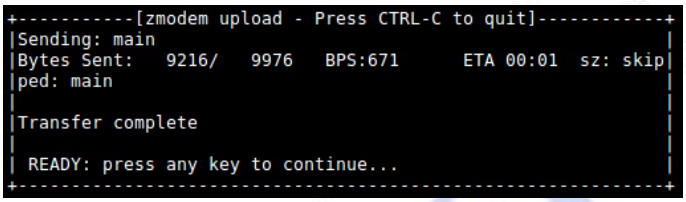


图 5-10 使用 lrzsz 工具时选择文件截图

注：更详细的操作参数，请使用 （--help）查看。

6 平台编译规则介绍（通用）

平台编译不同的模块需要包含不同的核心宏，类似于要使用 printf 函数，需要包含 stdio.h 头文件。添加不同的模块要使用的核心宏不同，下面介绍了几种不同类型模块的编译方法。

6.1 创建平台应用程序

6.1.1 编译核心宏

表 6-1 平台应用核心编译宏

编译核心宏	说明
BUILD_EXECUTABLE	编译生成可执行 bin 文件
BUILD_SHARED_LIBRARY	编译生成动态库
BUILD_STATIC_LIBRARY	编译生成静态库

6.1.2 常用的编译接口宏

接口宏用于配置编译的参数，常用的接口宏如下表所示。

表 6-2 平台应用接口宏

接口宏	说明
LOCAL_MODULE	模块名
LOCAL_MODULE_TAGS	模块所属开发模式
LOCAL_SRC_FILES	源文件
LOCAL_LDLIBS	链接参数
LOCAL_CFLAGS	C 编译参数
LOCAL_CPPFLAGS	CPP 编译参数
LOCAL_C_INCLUDES	编译所需头文件
LOCAL_MODULE_PATH	指定目标文件的 copy 路径，不定义此宏目标文件 copy 到默认路径下
LOCAL_SHARED_LIBRARIES	本模块编译所依赖的其他动态库，此动态库为工程中其他模块产生。
LOCAL_STATIC_LIBRARIES	本模块编译所依赖的其他静态库，此静态库为工程中其他模块产生
LOCAL_DEPANNER_MODULES	本模块的编译依赖的其他模块（生成的库，头文件等），此处为所依赖模块的 module 名。

6.1.3 编译示例

1） 可执行 bin 文件

本示例为工程 packages/example/App/cimutils/Build.mk

```
LOCAL_PATH := $(my-dir)

# build cimutils
include $(CLEAR_VARS)
LOCAL_MODULE=cimutils
LOCAL_MODULE_TAGS:= optional

LOCAL_SRC_FILES:= main.c       \
          convert_soft.c    \
         process_picture/saveraw.c    \
         process_picture/savebmp.c    \
         process_picture/savejpeg.c   \
          lcd/framebuffer.c    \
          v4l2/v4l2.c    \
          v4l2enc/v4l2_jpegenc.c    \
          cim/cim_fmt.c    \
          cim/video.c \
          cim/process.c    \
          cim/convert.c

LOCAL_LDLIBS := -lc -lstdc++ -ljpeg
LOCAL_C_INCLUDES:= include
LOCAL_DEPANNER_MODULES := CameraDemo.sh

include $(BUILD_EXECUTABLE)
```

2）生成动态库

本示例为工程 external/tinyalsa/Build.mk

```
LOCAL_PATH := $(my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE := libtinyalsa
LOCAL_MODULE_TAGS :=optional
LOCAL_SRC_FILES := pcm.c \
                   mixer.c LOCAL_EXPORT_C_INCLUDE_FILES:=include/tinyalsa/asoundlib.h
LOCAL_CFLAGS := -Wa,-mips32r2 -O2 -G 0 -Wall -fPIC -shared

include $(BUILD_SHARED_LIBRARY)
```

注：①静态库和动态库的路径默认存放在 system 的 usr/lib 下（system 中的静态库是从 sysroot 目录下的静态库生成的，sysroot 中的静态库未经过 strip），可执行程序默认存放在 system 的 usr/bin 下。

其他的参见工程下 build/core/envsetup.mk 文件或者上述的 out 目录介绍。

6.2 导入第三方应用程序

在添加第三方模块时，首先要确保模块能够正常编译通过，再添加到 platform 上，在添加放置第三方功能模块时，建议放置到 external 目录下，并参照 external 目录下各个的 Build.mk 文件编写所添模块的 Build.mk 文件，对所添模块进行指导性编译。

6.2.1 第三方应用程序核心宏

表 6-3 第三方应用核心宏

编译核心宏	说明
BUILD_THIRDPART	编译第三方模块

6.2.2 第三方应用常用接口宏

接口宏用于配置编译的参数，常用的接口宏如下表所示。

表 6-4 第三方应用接口宏

接口宏	说明
LOCAL_MODULE	模块名
LOCAL_MODULE_TAGS	模块所属开发模式
LOCAL_MODULE_GEN_BINRARY_FILES	模块产生的可执行程序（需含路径）
LOCAL_MODULE_GEN_STATIC_FILES	模块产生的静态库文件 （含路径）
LOCAL_MODULE_GEN_SHARED_FILES	模块产生的动态库文件 （含路径）
LOCAL_MODULE_PATH	指定目标文件的 copy 路径，不定义此宏目标文件将 copy 到默认路径下
LOCAL_MODULE_CONFIG_FILES	模块配置后生成的文件
LOCAL_MODULE_CONFIG	模块的配置方法
LOCAL_MODULE_COMPILE	模块的编译方法
LOCAL_MODULE_COMPILE_CLEAN	模块的 clean 方法
LOCAL_EXPORT_C_INCLUDE_FILES	模块对外导出的文件，所导出文件被其他模块使用
LOCAL_EXPORT_C_INCLUDE_DIRS	模块对外导出的目录，所导出目录下的文件被其他模块使用
LOCAL_DEPANNER_MODULES	本模块的编译依赖的其他模块（生成的库，头文件等），此处为所依赖模块的 module 名

6.2.3 编译示例

本示例为工程 packages/example/App/linphone-desktop/Build.mk

```
LOCAL_PATH := $(my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE=linphone-desktop
LOCAL_DEPANNER_MODULES:=isp
LOCAL_MODULE_TAGS:=optional
LOCAL_MODULE_GEN_BINRARY_FILES=OUTPUT/no-ui/bin/linphonec
LOCAL_MODULE_GEN_SHARED_FILES=OUTPUT/no-ui/lib/libavcodec.so
                              OUTPUT/no-ui/lib/libavcodec.so.53 \
                              ......
                              OUTPUT/no-ui/lib/libxml2.so.2
LOCAL_MODULE_CONFIG_FILES:= Makefile
LOCAL_MODULE_CONFIG:=./prepare.py \
                              -L -dv \
                               ......
                          -DENABLE_INGENIC_VIDEOOUT=ON
LOCAL_MODULE_COMPILE=make -j$(MAKE_JLEVEL); mkdir -p
   $(TOP_DIR)/$(LOCAL_PATH)/OUTPUT/no-ui/data/linphone
LOCAL_MODULE_COMPILE_CLEAN=./prepare.py -c


include $(BUILD_THIRDPART)
```

6.3 添加用于拷贝文件的模块

在开发过程中经常需要将一些库和头文件不加任何修改的从一个路径下 copy 到另外一个目标路径下，为此平台编译系统添加了 prebuild 机制。

6.3.1 编译核心宏

表 6-5 拷贝模块核心宏

编译核心宏	说明
BUILD_PREBUILT	一次 copy 一个文件
BUILD_MULTI_COPY	一次 copy 多个文件

6.3.2 常用接口宏

接口宏用于配置编译的参数，常用的接口宏如下表所示。

表 6-6 拷贝模块接口宏

接口宏	说明
LOCAL_MODULE	模块名
LOCAL_MODULE_TAGS	模块所属开发模式
LOCAL_MODULE_CLASS	模块所属哪种 prebuild 模式 DIR/FILES
LOCAL_MODULE_PATH	文件（夹）copy 的目标路径
LOCAL_COPY_FILES	源文件
LOCAL_MODULE_DIR	源文件夹

6.3.3 编译示例

1） 复制单个文件

本示例为工程 packages/example/App/cimutils/Build.mk

```
# copy the LOCAL_SRC_FILES

include $(CLEAR_VARS)
LOCAL_MODULE := CameraDemo.sh
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE_PATH := $(TARGET_FS_BUILD)/$(TARGET_TESTSUIT_DIR)/CameraDemo
LOCAL_COPY_FILES := CameraDemo.sh:CameraDemo.sh

include $(BUILD_PREBUILT)
```

其中宏 LOCAL_COPY_FILES := CameraDemo.sh:CameraDemo.sh,使用“:”分割，前面为拷贝后的名字（可包含路径，且必须有目标文件名），后面为本地的文件名（包含路径）

2） 复制多个文件

本示例为工程 external/android/include/Build.mk

```
LOCAL_PATH := $(my-dir)
# copy include

include $(CLEAR_VARS)

LOCAL_MODULE := android-include
LOCAL_MODULE_TAGS :=optional
LOCAL_MODULE_PATH :=$(OUT_DEVICE_INCLUDE_DIR)
LOCAL_MODULE_CLASS := DIR
LOCAL_MODULE_DIR := android-include


include $(BUILD_MULTI_COPY)
```

6.4 通过命令自动生成模块

创建好模块目录以后，在该目录下运行 autotouchmk，然后选择所要添加的模块类型（包括可执行模块、第三方模块、复制模块、静态库模块、动态库模块）。

运行 autotouchmk 命令以后，会有以下提示：

```
This command is used to generate the commonly used template, specific content need to edit
**************************************************
Please select device target type you want to build

    A:execute
    B:thirdpart
    C:static library
    D:shared library
    E:prebuilt
    F:multi copy
```

可执行模块：输入 A；第三方模块：输入 B；静态库模块：输入 C；动态库模块：输入 D；复制模块：输入 E；多项复制模块：输入 F；

然后会在当前目录下生成一个 Build.mk 文件，自动生成的 Build.mk 会将基本会用的宏都列出来，根据需要对其中的宏进行赋值，不需要的可以不进行修改。自动生成的 Build.mk 文件对于大多数宏都是没有赋值的，需要用户根据自己的需要进行赋值。

6.4.1 生成复制模块示例

在需要创建编译模块的目录下执行 autotouchmk，然后会有如下的提示信息。

```
This command is used to generate the commonly used template, specific content need to edit
**************************************************
Please select device target type you want to build
    A:execute
    B:thirdpart
    C:static library
    D:shared library
    E:prebuilt
    F:multi copy
Input module :E
```

我们选择 E 选项，创建复制模块，生成的内容如下。

```
LOCAL_PATH := $(my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE =      #name of the module:
LOCAL_MODULE_TAGS := optional          #Development mode of module ,userdebug,eng,optional
LOCAL_MODULE_PATH := $(TARGET_FS_BUILD)/     #which directory the target file copy
LOCAL_MODULE_CLASS:=
LOCAL_MODULE_DIR :=                          #src module dir
LOCAL_COPY_FILES :=
include $(BUILD_PREBUILT)
```

6.5 文件系统覆盖 rootfs-overlay

halley5 平台提供一个 rootfs-overlay 机制用于修改生成文件系统中的内容。rootfs-overlay 是一个文件夹，该文件夹位于 device/（halley5）板级/目录下。当该文件夹下存在与文件系统中相同文件名的文件时，会用当前目录下的文件替换文件系统中的文件。

6.6 制作根文件系统

halley5 平台支持多种类型文件系统的制作，可以通过配置文件系统的参数，制作不同的文件系统，下面是几个具体的文件系统制作方法。

6.6.1 jffs2 文件系统制作

1） 进入 halley5 工程
2） 执行 make buildroot-menuconfig
3） 配置 Filesystem images/jffs2 root filesystem/

如下图所示为 jffs2 文件系统的参考配置。

```
[*] jffs2 root filesystem
      Flash Type (Select custom erase size)  --->
(0x8000) Erase block size
[*]   Do not use Cleanmarker
[*]   Pad output
(0x0)   Pad output size (0x0 = to end of EB) (NEW)
      Endianess (little-endian)  --->
[ ]   Produce a summarized JFFS2 image (NEW)
[ ]   Select custom virtual memory page size (NEW)
```

图 6-1 jffs2 文件系统的参考配置

配置完成后，在工程目录中执行 make 指令进行编译，编译生成的文件会存放在 out/product/halley5/image/目录下。

6.6.2 ubifs 文件系统制作

1） 进入 halley5 工程
2） 执行 make buildroot-menuconfig
3） 配置 Filesystem images/ubifs root filesystem/

如下图所示为 ubifs 文件系统的参考配置。

```
[*] ubifs root filesystem
(0x1f000) logical eraseblock size
(0x800) minimum I/O unit size
(2048) maximum logical eraseblock count
      ubifs runtime compression (lzo)  --->
      Compression method (no compression)  --->
()    Additional mkfs.ubifs options
```

图 6-2 ubifs 文件系统的参考配置

配置完成后，在工程目录中执行 make 指令进行编译，编译生成的文件会存放在 out/product/halley5/image/目录下。

6.6.3 ext4 文件系统制作

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Filesystem images/ext2/3/4 root filesystem/

如下为 ext4 文件系统的参考配置。

```
[*] ext2/3/4 root filesystem
      ext2/3/4 variant (ext2 (rev1))  --->
()    filesystem label
(120M) exact size
(0)   exact number of inodes (leave at 0 for auto calculation)
(5)   reserved blocks percentage
(-O ^64bit) additional mke2fs options
      Compression method (no compression)  --->
```

图 6-3 ext4 文件系统的参考配置

配置完成后，在工程目录中执行 make 指令进行编译，编译生成的文件会存放在 out/product/halley5/image/目录下。

7 应用程序 debug

7.1 使用 gdb 命令行工具进行 debug

7.1.1 gdb 工具配置

1） 进入 halley5 工程

2） 执行 make buildroot-menuconfig

3） 配置 Target packages/Debugging, profiling and benchmark/gdb，如下图所示。

```
Symbol: BR2_PACKAGE_GDB_SERVER [=n]
Type  : bool
Prompt: gdbserver
  Location:
    -> Target packages
      -> Debugging, profiling and benchmark
(1)     -> gdb (BR2_PACKAGE_GDB [=n])
  Defined at package/gdb/Config.in:48
  Depends on: BR2_PACKAGE_GDB [=n] && !BR2_TOOLCHAIN_EXTERN
  Selected by [n]:
  - BR2_PACKAGE_GDB [=n] && BR2_TOOLCHAIN_HAS_THREADS [=y]
```

图 7-1 buildroot 中 gdb 配置路径

7.1.2 gdb 本地调试

使用 gdb 调试，在可执行程序编译时，需要加-g 选项。然后执行下面的指令进入 gdb 调试模式。

gdb 可执行文件名。

调试的界面如下图所示。

```
zhxiao@SW-PowerEdge-R610:~$
zhxiao@SW-PowerEdge-R610:~$ gdb test
GNU gdb (Ubuntu 7.7-0ubuntu3) 7.7
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test...done.
(gdb)
(gdb)
(gdb) list
9               int c = 0;
10
11              c = a + b;
12
13              return c;
14      }
15
16
17      int main(void)
18      {
(gdb)
19              int sum;
20              int i;
21
22              for(i = 0; i < 10; i++)
23              {
24                      sum += add(i,i);
25              }
26
27              printf("sum = %d\n",sum);
28
(gdb)
29              return 0;
30      }
```

图 7-2 gdb 本地调试界面

7.1.3 gdbserver + gdb 远程调试

在很多情况下，需要对一个应用程序进行反复调试，特别是复杂的程序。采用 GDB 方法调试，由于嵌入式系统资源有限性，一般不能直接在目标系统上进行调试，通常采用 gdb+gdbserver 的方式进行调试。gdbserver 在目标系统中运行，gdb 则在宿主机上运行。

7.1.3.1 gdbserver 的安装

gdbserver 可以使用源码编译产生，halley5 中提供了 gdb 的源码。在 buildroot/dl/gdb/目录下。

halley5 中也提供了编译完成了 gdbserver，用户只需将该 gdbserver 文件拷贝到板子中即可，其路径如下：

prebuilts/toolchains/mips-gcc720-glibc226/mips-linux-gnu/libc/mfp64/usr/bin/gdbserver

7.1.3.2 gdbserver + gdb 调试方法

在目标板上执行如下指令：

```
gdbserver 宿主机 IP:端口号 要调试的可执行程序
```

其中端口号可自由设置。

在宿主机上执行如下指令：

```
mips-linux-gnu-gdb 要调试的可执行程序
```

然后会进入 gdb 调试界面，接着执行指令

```
target remote 目标板 IP:端口号
```

其中的可执行程序需和目标板上运行的相同，端口号也必须相同。

如下图为使用 gdbserver + gdb 方式调试的调试过程截图

```
# ./gdbserver 192.168.4.50:1234 hello
Process hello created; pid = 5343
Listening on port 1234
```

图 7-3 目标板执行指令后等待连接

```
user@tftpboot$ mips-linux-gnu-gdb hello
GNU gdb (Ingenic r4.1.0-gcc720 2018.08-06) 7.11.50.20160803-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=mips-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello...done.
(gdb)
(gdb) target remote 192.168.4.49:1234
Remote debugging using 192.168.4.49:1234
0x004002b0 in __start ()
(gdb)
```

图 7-4 宿主机进入 gdb 调试界面并连接目标板

```
# ./gdbserver 192.168.4.50:1234 hello
Process hello created; pid = 5343
Listening on port 1234
Remote debugging from host 192.168.4.50
```

图 7-5 目标板与宿主机成功建立连接

```
(gdb)
(gdb) target remote 192.168.4.49:1234
Remote debugging using 192.168.4.49:1234
0x004002b0 in __start ()
(gdb)
(gdb) list
1       #include <stdio.h>
2
3
4
5       int main(void)
6       {
7               int i;
8
9               for(i = 0; i < 10; i ++)
10              {
(gdb) list
11                      printf("[%d] hello world!\n",i);
12              }
13
14              printf("hello world test over!\n");
15      }
16
17
18
19
20
(gdb) b main
Breakpoint 1 at 0x40024c: file helloword.c, line 6.
(gdb) c
Continuing.

Breakpoint 1, main () at helloword.c:6
6       {
(gdb) r
The "remote" target does not support "run".  Try "help target" or "continue".
(gdb) n
[Inferior 1 (process 5343) exited normally]
(gdb) n
The program is not being run.
(gdb)
```

图 7-6 宿主机进行调试

```
# ./gdbserver 192.168.4.50:1234 hello
Process hello created; pid = 5343
Listening on port 1234
Remote debugging from host 192.168.4.50
[0] hello world!
[1] hello world!
[2] hello world!
[3] hello world!
[4] hello world!
[5] hello world!
[6] hello world!
[7] hello world!
[8] hello world!
[9] hello world!
hello world test over!

Child exited with status 0
#
```

图 7-7 目标板打印调试信息

7.1.4 gdb 调试使用命令

1）运行命令

run：简记为 r ，其作用是运行程序，当遇到断点后，程序会在断点处停止运行，等待用户输入下一步的命令。

continue （简写 c ）：继续执行，到下一个断点处（或运行结束）

next：（简写 n），单步跟踪程序，当遇到函数调用时，也不进入此函数体；此命令同 step 的主要区别是，step 遇到用户自定义的函数，将步进到函数中去运行，而 next 则直接调用函数，不会进入到函数体内。

step （简写 s）：单步调试如果有函数调用，则进入函数；与命令 n 不同，n 是不进入调用的函数的

until：当你厌倦了在一个循环体内单步跟踪时，这个命令可以运行程序直到退出循环体。

until+行号： 运行至某行，不仅仅用来跳出循环

finish： 运行程序，直到当前函数完成返回，并打印函数返回时的堆栈地址和返回值及参数值等信息。

call 函数(参数)：调用程序中可见的函数，并传递“参数”，如：call gdb_test(55)

quit：简记为 q ，退出 gdb

2）设置断点

break n （简写 b n）:在第 n 行处设置断点

b fn1 if a>b：条件断点设置

break func（break 缩写为 b）：在函数 func()的入口处设置断点

delete 断点号 n：删除第 n 个断点

disable 断点号 n：暂停第 n 个断点

enable 断点号 n：开启第 n 个断点

clear 行号 n：清除第 n 行的断点

info b （info breakpoints） ：显示当前程序的断点设置情况

delete breakpoints：清除所有断点

3）查看源码

list ：简记为 l，其作用是列出程序的源代码，默认每次显示 10 行。

list 行号：将显示当前文件以“行号”为中心的前后 10 行代码

list 函数名：将显示“函数名”所在函数的源代码，如：list main

list ：不带参数，将接着上一次 list 命令的，输出下边的内容。

4）打印表达式

print 表达式：简记为 p，其中“表达式”可以是任何当前正在被测试程序的有效表达式，比如当前正在调试 C 语言的程序，那么“表达式”可以是任何 C 语言的有效表达式，包括数字，变量甚至是函数调用。

print a：将显示整数 a 的值

print ++a：将把 a 中的值加 1,并显示出来

print name：将显示字符串 name 的值

print gdb_test(22)：将以整数 22 作为参数调用 gdb_test() 函数

print gdb_test(a)：将以变量 a 作为参数调用 gdb_test() 函数

display 表达式：在单步运行时将非常有用，使用 display 命令设置一个表达式后，它将在每次单步进行指令后，紧接着输出被设置的表达式及值。如： display a

watch 表达式：设置一个监视点，一旦被监视的“表达式”的值改变，gdb 将强行终止正在被调试的程序。如： watch a

whatis ：查询变量或函数

info function： 查询函数

扩展 info locals： 显示当前堆栈页的所有变量

5）查看运行信息

```
where/bt ：当前运行的堆栈列表；
bt backtrace 显示当前调用堆栈
up/down 改变堆栈显示的深度
set args 参数:指定运行时的参数
show args：查看设置好的参数
info program：查看程序的是否在运行，进程号，被暂停的原因。
```

6）分割窗口

```
layout：用于分割窗口，可以一边查看代码，一边测试
layout src：显示源代码窗口
layout asm：显示反汇编窗口
layout regs：显示源代码/反汇编和 CPU 寄存器窗口
layout split：显示源代码和反汇编窗口
Ctrl + L：刷新窗口
```

7.2 使用 vscode 集成开发环境进行 debug

7.2.1 vccode 安装及使用

VScode 官方安装包下载: [https://code.visualstudio.com/Download]
VScode 官方安装参考文档:[https://code.visualstudio.com/docs/setup/linux]
VScode 官方调试使用参考文档:[https://code.visualstudio.com/docs/cpp/cpp-debug]
VScode 官方调试配置参考文档:
[https://code.visualstudio.com/docs/cpp/launch-json-reference]

7.2.2 调试环境搭建及工程编译

在 halley5 工程顶层目录下，可通过以下命令进行工程交叉调试配置

```
make  <ProjectName>-vscode PRO=<ProjectName>
```

其中的<ProjectName>可以是任意名字，为了方便记忆一般以板级名字作为 ProjectName
命令执行完成后会在当前目录先生成一个<ProjectName>.code-workspace。
进入工程目录后执行以下命令进入工作区（执行命令前请确认已经成功安装了 vscode）

```
code  <ProjectName>.code-workspace
```

命令执行完毕将会进入 vscode 可视化调试阶段。
在需要编译的源文件界面上，点击左上方绿色“△”图案，或点击菜单栏的 Run 选择其中的 StartDebugging 可进行文件的编译并调试。如下图所示。

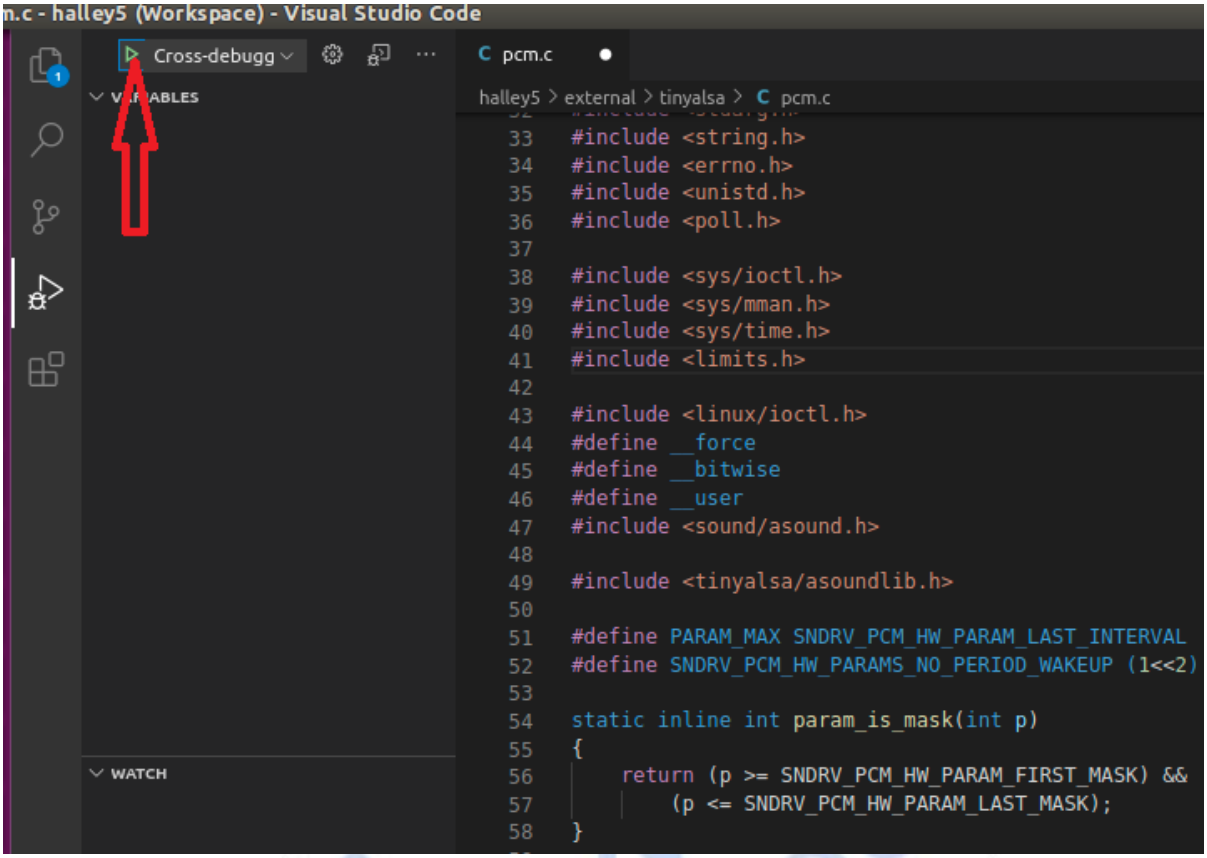


图 7-8 vscode 界面截图

在 halley5 软件工程顶层目录下，可通过以下命令进行工程配置信息清除

```
make  <ProjectName>-clean PRO=<ProjectName>
```

7.2.3 开始调试

F5 启动调试，可以通过 F9 设断点，F11 单步等来调试程序。