君正®

Halley5 软件平台快速开发指南

Author: 系统软件部

Version: 1.0
Date: 2020.11.30

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

君正®

Halley5 软件平台快速开发指南

Release history

Date	Revision	Change
2020.11.30	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	编译、烧录、测试
摘　　要	本文档适用于君正推出的 halley5 开发板，目的在于帮助开发者快速的了解开发板配套的软件开发平台的组织结构、软件编译、程序烧录以及相应的功能测试演示。
缩 略 语	SDK 工程——工程、SDK 工程顶层目录——工程目录
参考资料	

1 SDK 工程下载

开发板配套的 SDK 源码工程均是通过百度网盘的方式进行对外发布。本节将讲述如何下载最新的源码工程以及解压相关的压缩包，文中附带下载链接。

1.1 SDK 下载

网盘下载链接：https://pan.baidu.com/s/1PxHJhv7j_oXkFTjAVNInxA 提取码:6svw 该链接下有开发板的芯片数据手册、开发板的硬件原理图以及 SDK 软件包。这里主要讲述软件包的下载，其他资料用户可根据自己的需要自行选择下载。这里选择最新的 SDK 软件包进行下载，如图 1-1 所示，如果有更新的版本请选择最新的。

图 1-1

1.2 Buildroot 下载

Halley5 配套的软件开发平台需要使用 Buildroot 来提供最基础的文件系统，因此这里需要下载君正已经修改制作的 Buildroot 工程。网盘下载链接：
https://pan.baidu.com/s/1e57Ze_WXoOHGFN6xhcNLkQ 提取码：gx42，同样这里也选择最新的工程下载。*（特别需要注意的是 d1-xxxx 的版本号一定要和 SDK 的版本号相对应，原则上推荐都下载最新的）*

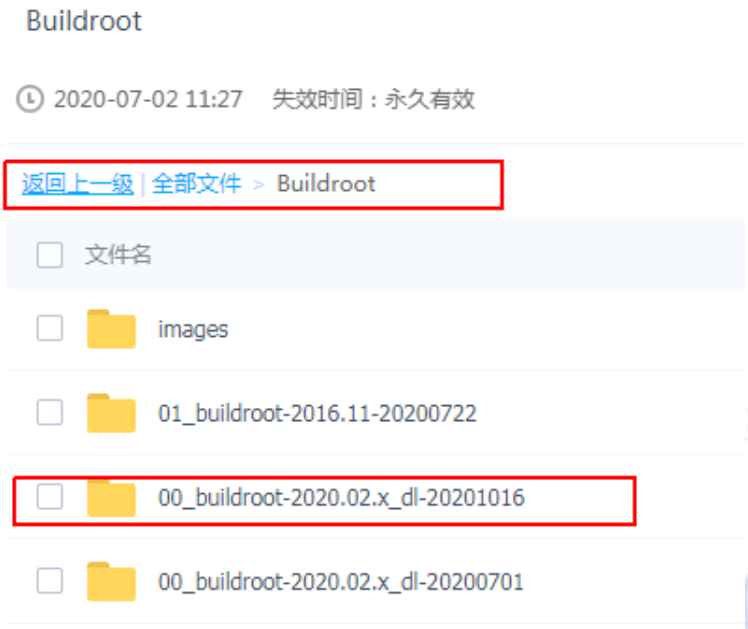


图 1-2

1.3 工程解压

1、将下载完成的 SDK 工程目录中的 ingenic-linux-kernel4.4.94-x2000_v12-v5.0-20201016.tar.bz2 文件拷贝到一个新建的目录下。

2、将下载完成的 Buildroot 工程目录下的 dl.tar.bz2 拷贝到上述新建的目录下。

3、执行命令进行解压

```
$ tar -xf ingenic-linux-kernel4.4.94-x2000_v12-v5.0-20201016.tar.bz2

$ tar -xf dl.tar.bz2 -C ./ingenic-linux-kernel4.4.94-x2000_v12-v5.0-20201016/buildroot/
```

2 SDK 简介

2.1 工程整体结构

Halley5 配套的软件开发平台是君正开发的一套 linux 系统的发布、开发平台，平台的编译系统可支持 c 文件、cpp 文件。同时可支持静态库、动态库以及可执行文件的编译生成。本软件开发平台还能够快速方便的集成、添加第三方应用和库文件。工程支持整体编译、模块的单独编译以及选择性编译。本软件开发平台的工程目录如图 2-1 所示。

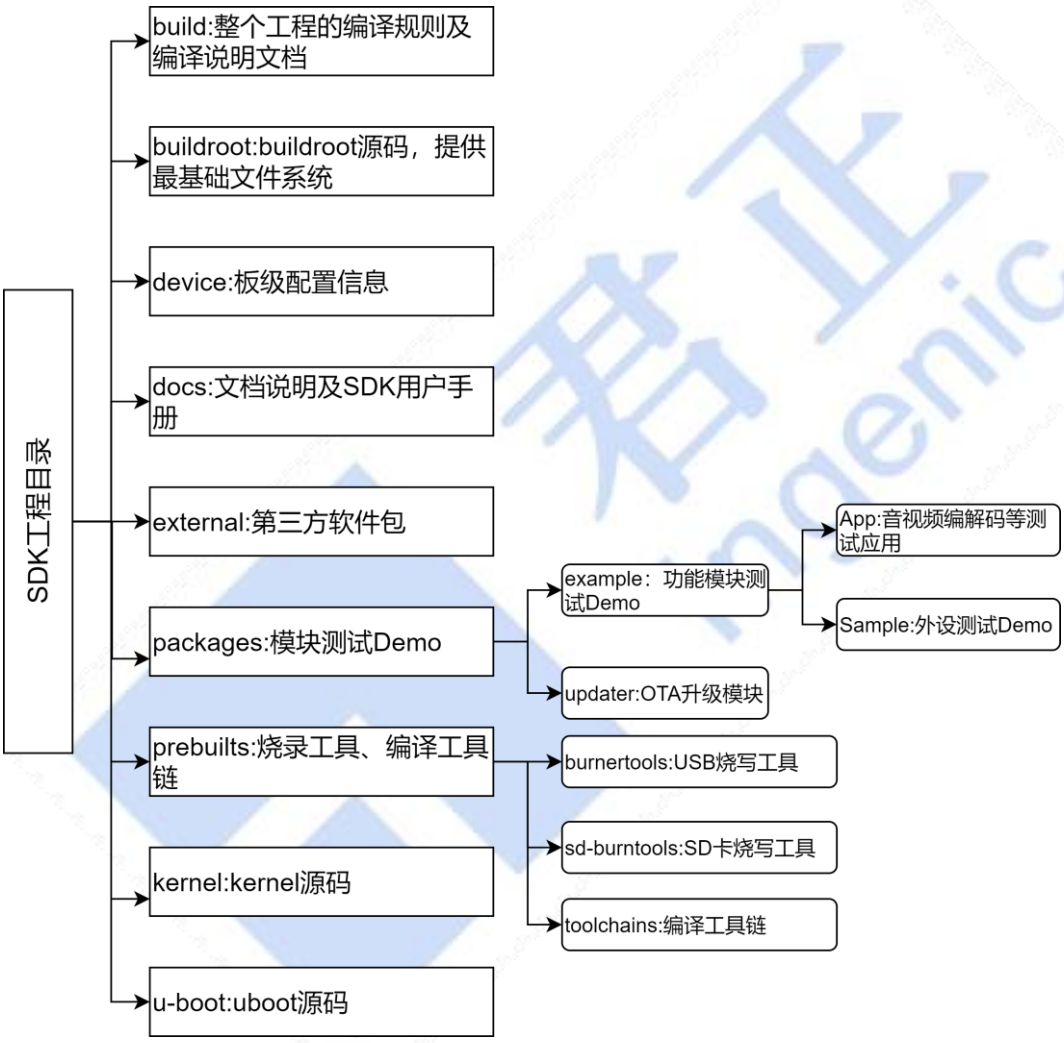


图 2-1

2.2 文档说明

本文档目的是为了帮助开发者快速的了解如何对 SDK 的编译以及程序的烧录，如果用户需要更加深入的了解 Halley5 的各个外设模块和应用开发，可以参考 Halley5 配套发布的其它文档。各文档的介绍说明请参考表 1 。

表 1

文档名	文档内容说明
00- Halley5 软件平台快速开发指南	帮助开发者快速了解平台软件的编译和烧写以及测试
01- Halley5 uboot 开发手册	U-boot 的配置编译、内核加载以及文件系统的挂载
02- Halley5 Linux4.4 内核开发手册	各内核驱动模块的配置编译，以及相应驱动源码的在内核结构的位置
03- Halley5 平台应用开发手册	Buildroot 的配置编译、用户应用程序的添加和编译、应用程序的调试方法
04 USBCloner烧录工具快速上手指南.pdf	USB 烧录工具在 ubuntu 和 windows 的烧录演示，该文档位于工程目录/prebuilts/burnertools/Doc_Chinese
04- USBCloner烧录工具说明文档.pdf	USB 纤细使用说明/prebuilts/burnertools/Doc_Chinese

3 SDK 编译

3.1 整体编译

本节主要介绍快速对 SDK 工程进行整体编译的方法，编译步骤具体如下：

1、设置环境变量

在工程目录下执行命令：

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

2、选择 device

在工程目录下执行命令：

```
$ lunch
```

命令执行结束后会显示不同的配置方案，请根据开发板的配置输入编号进行选择，例如 halley5.v10或halley5.v20。同时需要注意区分开发板的存储类型，如nand、nor和msc。关于 lunch配置中的各选项的说明可参考表 2 的解读。本文档里halley5.v20_nand_4.4.94-eng 配置进行演示。

表 2

选项编号	配置选项名称	配置说明
1	halley5.v10_nand_4.4.94-eng	适用于编译v10版本的开发板的nand启动
4	halley5.v10_nor_4.4.94-eng	适用于编译v10版本的开发板的nor启动
7	halley5.v10_msc_4.4.94-eng	适用于编译v10版本的开发板的SD卡启动
10	halley5.v10_nand_4.4.94_ota-eng	适用于编译v10版本的开发板的OTA升级
13	halley5.v20_nand_4.4.94-eng	适用于编译v20版本的开发板的nand启动
16	halley5.v20_nor_4.4.94-eng	适用于编译v20版本的开发板的nor启动
19	halley5.v20_msc_4.4.94-eng	适用于编译v20版本的开发板的SD卡启动
22	halley5.v20_nand_4.4.94_ota-eng	适用于编译v20版本的开发板的OTA升级
25	halley5.v20_msc_4.4.94_burn-eng	适用于编译v20版本的开发板的SD卡烧录时的SD卡启动镜像

3、工程编译

在工程目录下执行 make 或 make MAKE_JLEVEL=8 或 make -j8（多线程编译）

```
$ make -j8
```

注意：由于机器性能的不同编译时长也会有差异，通常会在 2 小时以上。

至此，工程的整体编译工作已完成。编译结果的输出具体如下：

① 工程目录/out/product/halley5/image，生成 kernel、system.ubifs、uboot 镜像。

② 工程目录/out/product/halley5/obj 是 buildroot、kernel、uboot、packages 等编译过程中的输出的中间文件。

4、工程 clean

```
$ make clean
```

3.2 单独编译

本节主要介绍如何对 uboot、kernel、buildroot 和模块的单独编译，最后还将介绍如何打包文件系统镜像。在进行模块单独编译前需要进行环境变量设置和 device 的选择，具体操作请参考

2.1 中的相关描述，如当前终端已经设置无须重新设置。（*需要注意的是单独编译的前提条件是工程已进行了整体编译*）

3.2.1 编译 uboot

1、清除 uboot 编译结果的方法，在工程目录下执行命令：

```
$ make uboot-clean
```

2、编译 uboot 的方法，在工程目录下执行命令：

```
$ make uboot
```

3.2.2 编译 kernel

1、清除 kernel 编译结果的方法，在工程目录下执行命令：

```
$ make kernel-clean
```

2、配置 kernel 的方法，在工程目录下执行命令：

```
$ make kernel-menuconfig
```

3、编译 kernel 的方法，在工程目录下执行命令：

```
$ make kernel -j8
```

3.2.3 编译 buildroot

1、清除 buildroot 编译中间文件的方法，在工程目录下执行命令：

```
$ make buildroot-clean

$ make buildroot-distclean
```

2、配置 buildroot 的方法，在工程目录下执行命令：

```
$ make buildroot-menuconfig
```

3、编译 buildroot 的方法，在工程目录下执行命令：

```
$ make buildroot -j8
```

4、局部修改 buildroot 后编译 buildroot 的方法，在工程目录下执行命令

```
$ make buildroot-rebuild
```

5、在对一些第三方库或者是内核工具软件如，opencv 和 i2c-tools_4.1 做出修改后可以通过执行 buildroot 内置的命令进行单独编译。

```
$ make buildroot-opencv-rebuild

$ make buildroot-i2c-tools-rebuild
```

buildroot 工程所包含的第三方软件位于 buildroot/packages 目录下，编译输出位于 /out/product/halley5/obj/buildroot-intermediate/build。

6、关于 buildroot 其它命令的支持可以通过 help 选项进行查询，在工程目录下执行命令：

```
$ make buildroot-help
```

3.2.4 编译模块

本工程支持对单个模块的编译及对单个模块的 clean 操作。工程的编译系统初始化后，存

在两套单模块的编译方法，一种是通用的模块编译方法 mma 命令，另外一种是直接在工程目录下使用 make 命令。在单独编译模块之前，需要确保当前终端的环境变量已设置，具体请参考 2.1 小节中的相关描述。

3.2.4.1 mma 命令单独编译模块

mma 命令为通用单模块编译方法，不区分此模块为 host 端还是 target 端。此命令的具体使用方法为：

1、进入到所要编译模块目录下，此模块目录下含有 Build.mk 文件。

2、执行 mma 命令，此模块以及此模块所依赖的模块都会被编译。

3.2.4.2 Target 端模块编译

Tartget 端模块编译是在工程目录下进行的，命令格式为：

```
$ make  $(LOCAL_MODULE)
```

命令使用示例，以工程目录 /packages/example/App/v412-h264dec 模块为例，该模块在其目录下的 Build.mk 文件中 LOCAL_MODULE 赋值为：v412-h264dec。单独编译模块只需要在工程目录下执行命令：

```
$  make v412-h264dec
```

编译后的可执行文件最终将被直接安装到文件系统的相应路径下，可根据编译后最终的安装路径找到相应的可执行文件。

3.2.4.3 Target 端模块 clean

Tartget 端模块 clean 可在工程目录下进行，命令格式为：

```
$ make $(LOCAL_MODULE)-clean
```

命令使用示例，以工程目录 /packages/example/App/v412-h264dec 模块为例，该模块在其目录下的 Build.mk 文件中 LOCAL_MODULE 赋值为：v412-h264dec。单独 clean 概模块只需要在工程目录下执行命令：

```
$ make v412-h264dec-clean
```

3.2.5 编译文件系统镜像

单独编译文件系统镜像的方法为，在工程目录下执行命令：

```
$ make systemimage
```

命令执行结束后，会在工程目录/out/product/halley5/image 下重新生成 system 的镜像。

3.2.6 打包文件系统镜像

打包文件系统镜像，将编译后输出的文件打包成最终的文件系统镜像，用于烧录到开发板。打包文件系统镜像需要在工程目录下执行命令：

```
$ make post-image
```

最终的文件系统镜像输出在工程目录/out/product/halley5/image 目录下。

4 烧录工具使用

本节主要讲述本开发平台配套的 USB 烧录工具的使用，此 USB 烧录工具可在 ubuntu12.04、windowsXP 以及 Windows7 以上的版本，均支持 32 位和 64 位。本开发指南只针对在 ubuntu 下使用 USB 烧录工具进行相应的烧写演示。烧录工具位于工程目录/prebuilts/burnertools。进入到烧写工具目录下，输入解压烧录工具文件命令：

```
$    tar -xf cloner-2.5.10.6-ubuntu_alpha.tar.gz
```

解压完成后进入到相应的烧写工具目录下，准备进行 uboot、kernel、rootfs 的烧录。开始烧录前需要确保开发板的两个 Type-C 接口和 PC 已连接。

注意:实际烧录工具版本可能与文档不一致，以 SDK 发布为准.

支持的烧录配置如下表所示，注意区分 X2000 和 X2000E 系列芯片

烧录配置	说明
x2000_mmc0_lpddr3_linux.cfg	➢ eMMC 启动默认烧录配置
x2000_mmc1_dtb_lpddr3_linux.cfg	SD 卡启动外部 devicetree 烧录配置
x2000_mmc1_lpddr3_linux.cfg	➢ SD 卡启动默认烧录配置
x2000_sfc_nand_dtb_lpddr3_linux.cfg	Nand 启动外部 devicetree 烧录配置
x2000_sfc_nand_lpddr3_linux.cfg	➢ Nand 启动默认配置
x2000_sfc_nand_ota_lpddr3_linux.cfg	Nand 启动，支持 OTA 升级的烧录配置
x2000_sfc_nor_lpddr3_linux.cfg	➢ Nor 启动默认烧录配置
x2000e_mmc0_lpddr2_linux.cfg	➢ eMMC 启动默认烧录配置
x2000e_mmc1_dtb_lpddr2_linux.cfg	SD 卡启动外部 devicetree 烧录配置
x2000e_mmc1_lpddr2_linux.cfg	➢ SD 卡启动默认烧录配置
x2000e_sfc_nand_dtb_lpddr2_linux.cfg	Nand 启动外部 devicetree 烧录配置
x2000e_sfc_nand_lpddr2_linux.cfg	➢ Nand 启动默认配置
x2000e_sfc_nand_ota_lpddr2_linux.cfg	Nand 启动，支持 OTA 升级的烧录配置
x2000e_sfc_nor_lpddr2_linux.cfg	➢ Nor 启动默认烧录配置

4.1 烧录 SFC Nand

烧写 SFC Nand 的具体方式为：

1、进入烧写工具目录，执行命令：

```
$ ./cloner
```

2、命令执行结束后将打开一个图形界面软件，首先先进行相应的配置点击 Config。

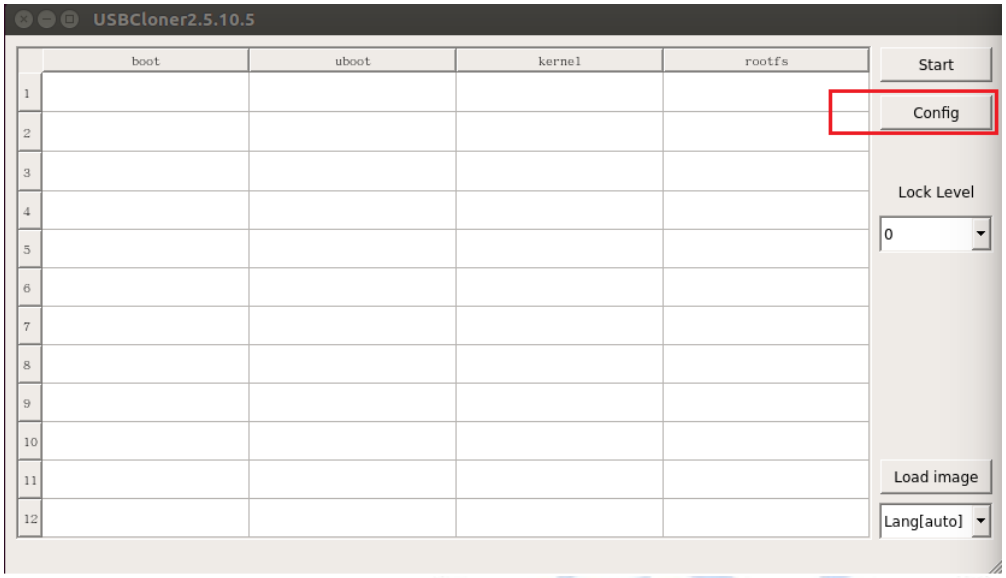


图 3-1

3、基本信息的配置，点击 INFO，进入基本配置信息，Platform 选择 x2000,Board 选择 x2000_sfc_nand_lpddr3_linux.cfg，具体可参图 3-2。

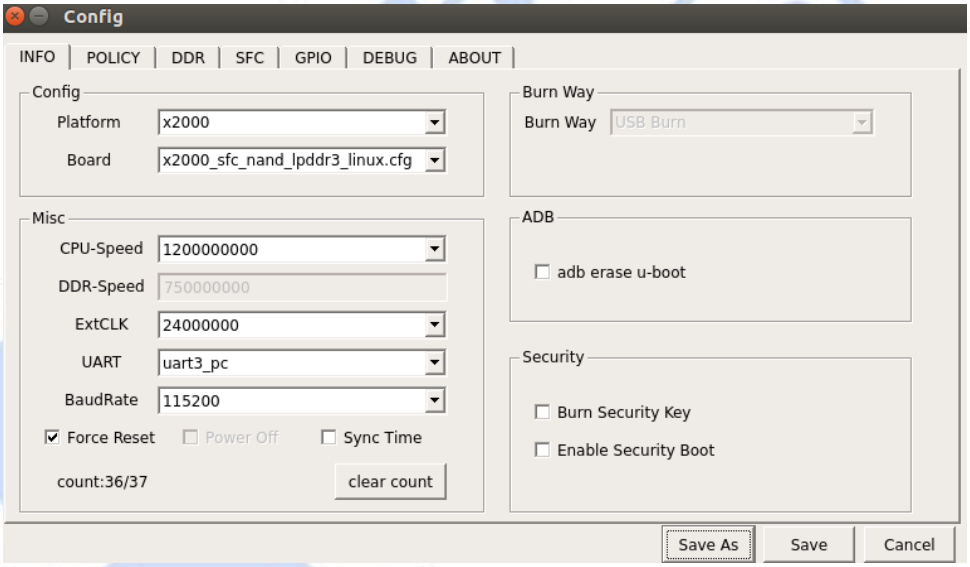


图 3-2

4、配置镜像文件路径，点击 POLICY 进行相应的配置。顺序设置 uboot、kernel 和 rootfs 镜像文件的相应路径（文件名分别是 uboot、kernel、system.ubifs），点击其对应的 setting 即可进行文件选择。配置可参考图 3-3。

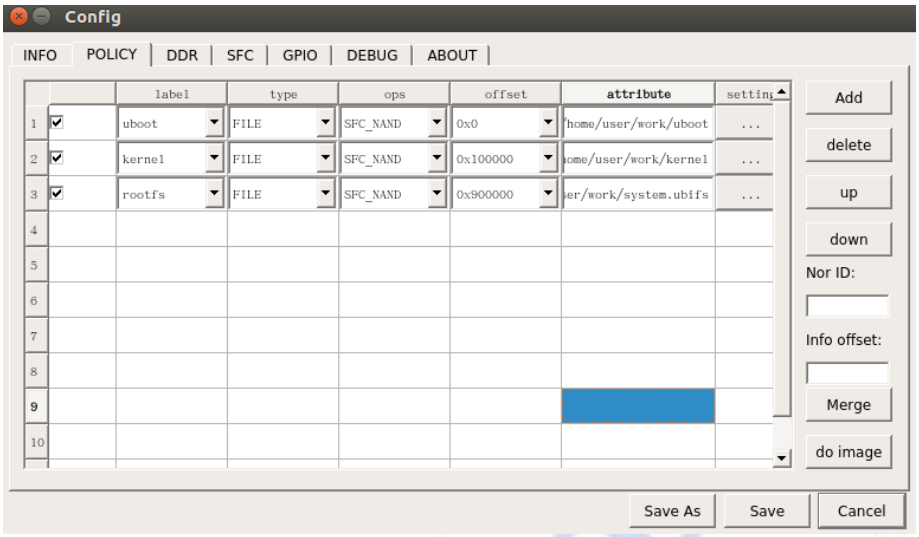


图 3-3

5、设置完成后点击 Save 保存配置，准备烧写。

6、点击烧录工具主界面的 Start 按钮。

7、开发板进入烧写状态，先按下开发板的 BOOT_SEL0 按键，然后按下 RST_N 按键，待蓝色指示灯亮起后，再分别松开 RST_N 和 BOOTL_SEL0 按键。

8、等待烧录工具各项烧录完成到 100%，如图 3-4 所示。

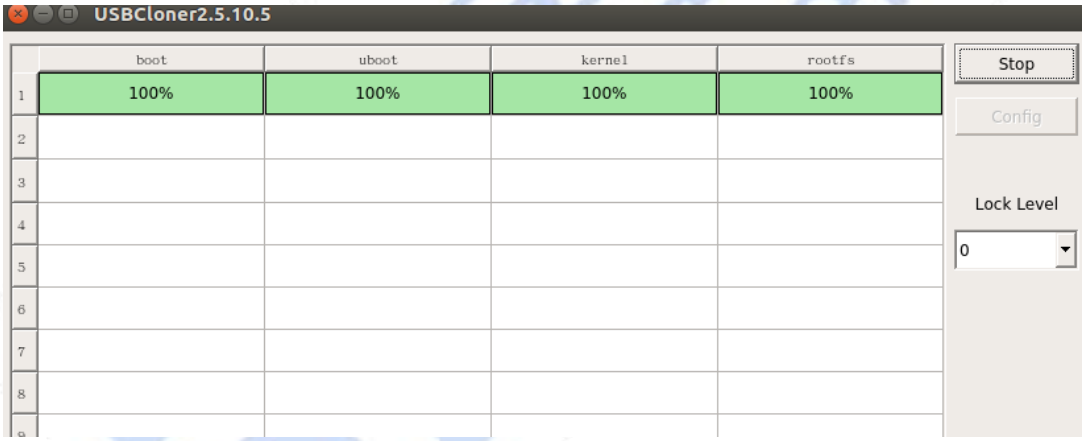


图 3-4

4.2 烧录 SFC Nor

烧录 SFC Nor 和 SFC Nand 的烧写差异在于配置的不同，具体的 配置方法为：

1、基本信息的配置，点击 INFO，进入基本配置信息，Platform 选择 x2000,Board 选择 x2000_sfc_nor_lpddr3_linux.cfg，具体可参图 3-5。

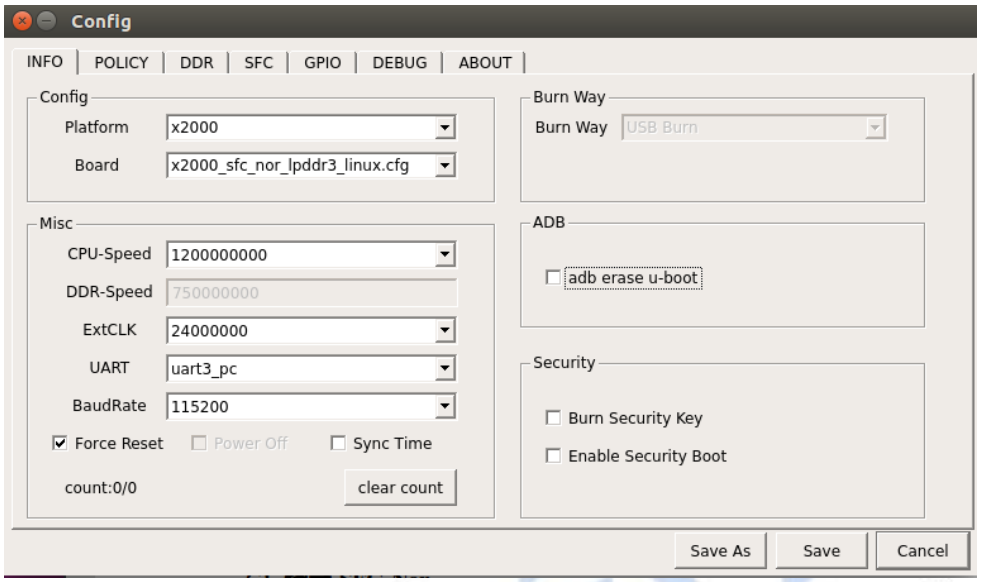


图 3-5

2、配置镜像文件路径，点击 POLICY 进行相应的配置。顺序设置 uboot、kernel 和 rootfs 镜像文件的相应路径，点击其对应的 setting 即可进行文件选择。配置可参考图 3-6。

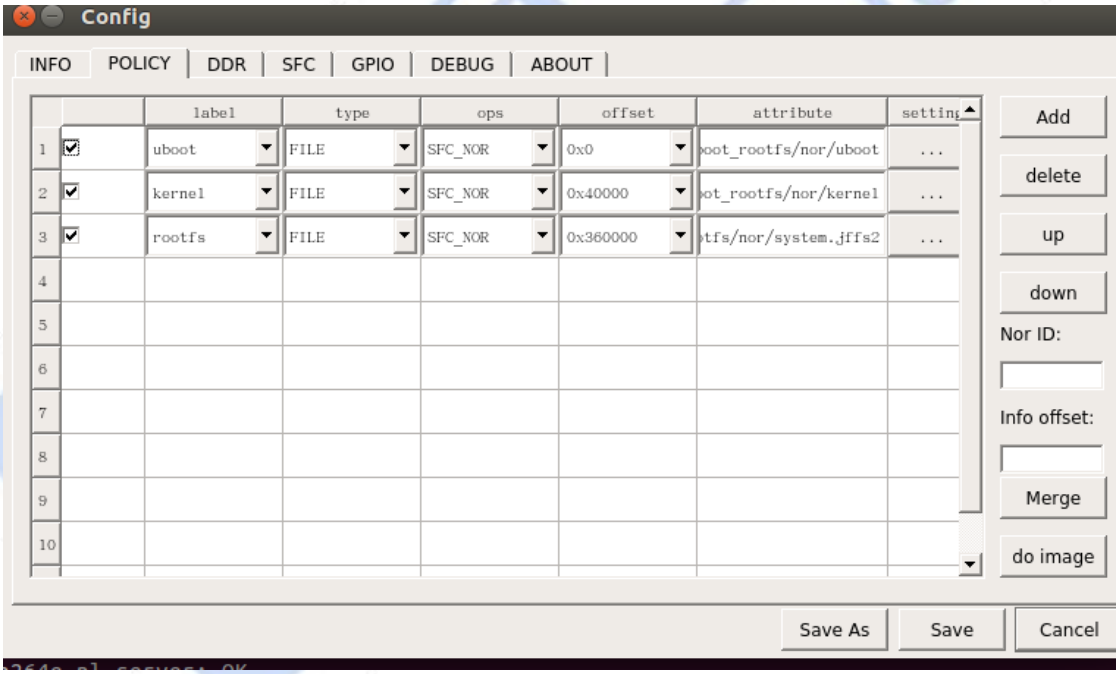


图 3-6

3、SFC 配置，由于烧录工具支持的 flash 种类较多，在进行烧写前需要根据板载的 nor flash 的型号解决 ID 冲突的问题。点击 norinfo，根据 name 查找到对应板载 nor flash 的 ID 并浏览是否有标红冲突的，如果有冲突择将其他的删除。配置界面参考图 3-7。

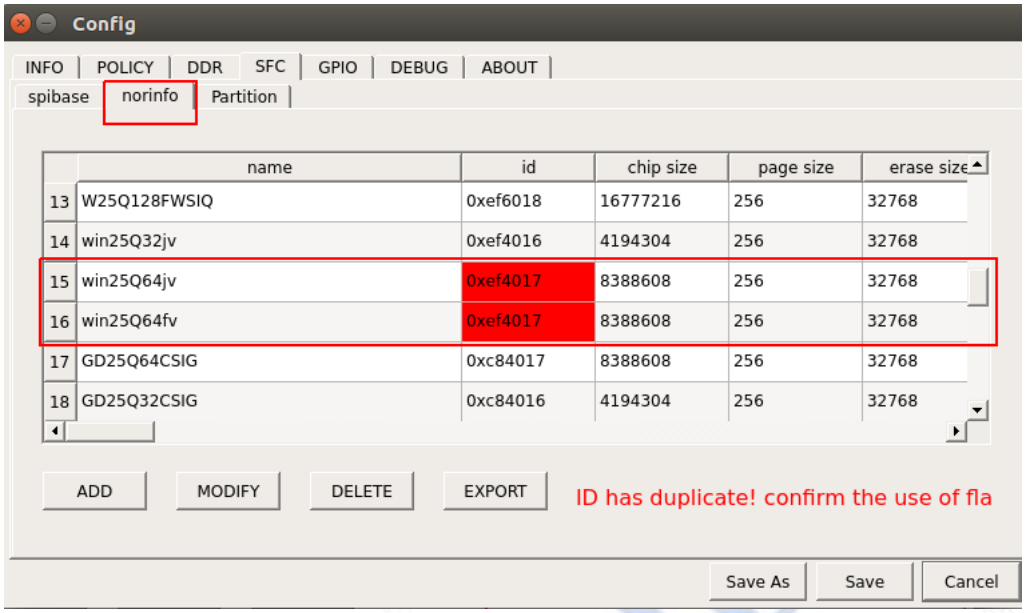


图 3-7

4、设置完成后点击 Save 保存配置，准备烧写。

5、点击烧录工具主界面的 Start 按钮。

6、开发板进入烧写状态，先按下开发板的 BOOT_SEL0 按键，然后按下 RST_N 按键，待蓝色指示灯亮起后，再分别松开 RST_N 和 BOOTL_SEL0 按键。

7、等待烧录工具各项烧录到 100%，如图 3-4 所示。

4.3 烧录 SD 卡

本小节主要讲述如何通过 USB 烧录工具将镜像文件烧录到 SD 卡，以及如何从 SD 卡启动。烧录 SD 卡和烧录 Nand 的区别主要在于配置的差异，具体的 配置为：

1、基本信息的配置，点击 INFO，进入基本配置信息，Platform 选择 x2000,Board 选择 x2000_mmc1_lpddr3_linux.cfg，具体可参图 3-8。

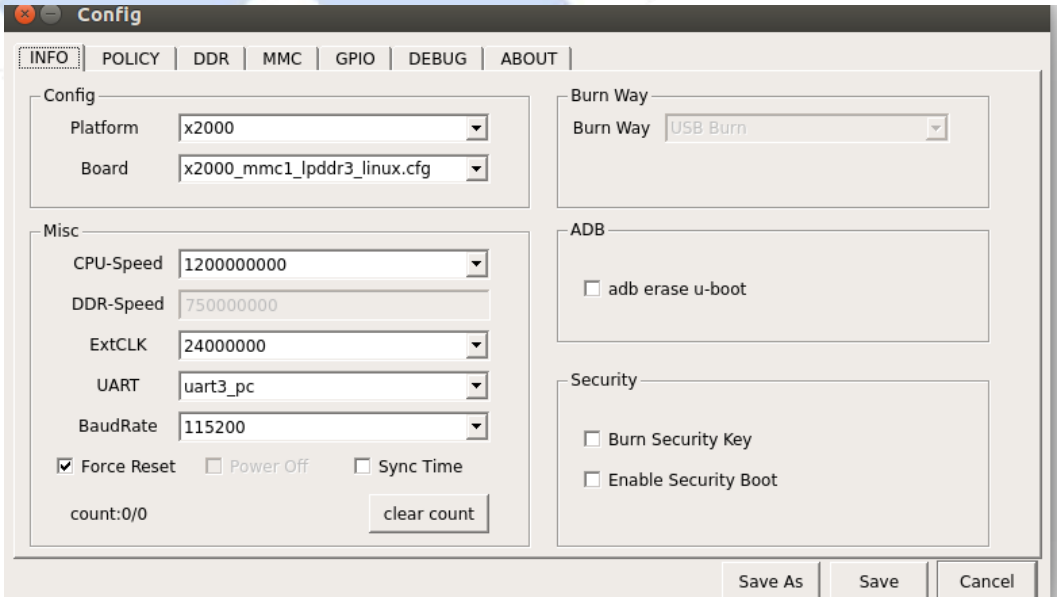


图 3-8

2、配置镜像文件路径，点击 POLICY 进行相应的配置。顺序设置 uboot、kernel 和 rootfs 镜像文件的相应路径（文件名分别是 uboot、kernel、system.ext2），点击其对应的 setting 即可进行文件选择。配置可参考图 3-9。需要注意的是，不同的存储介质使用的文件系统格式是不同的，均需要再编译之前重新进行 lunch 配置再编译。

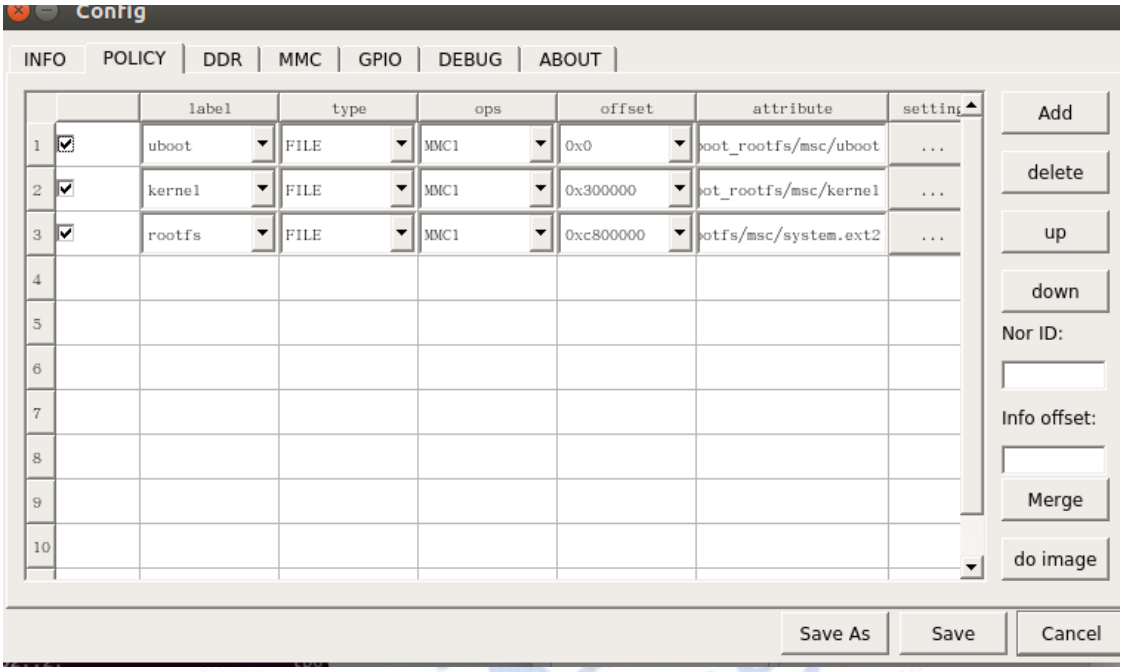


图 3-9

3、MMC 选项配置，如图 3-10 所示。

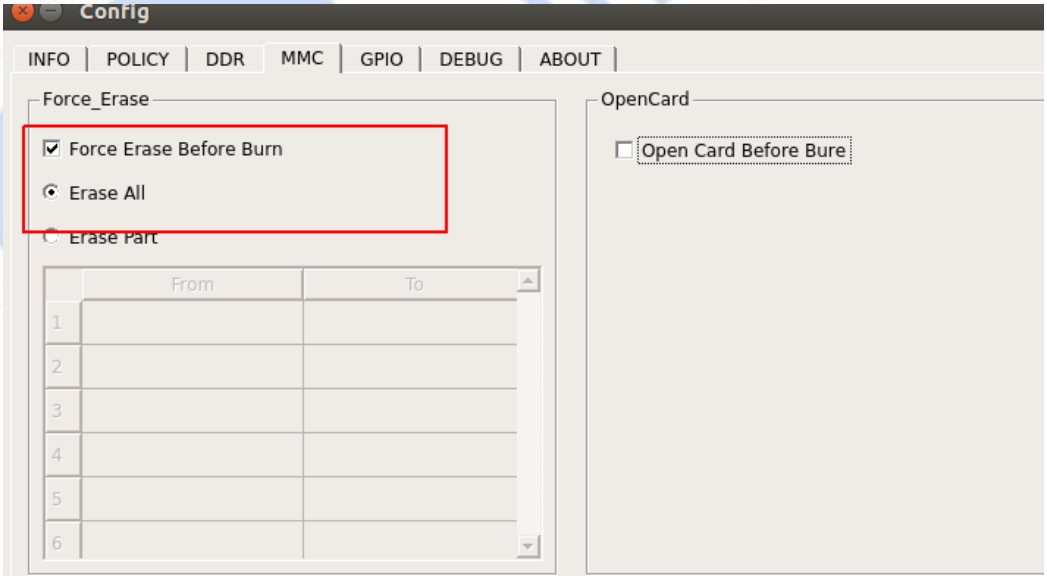


图 3-10

4、设置完成后点击 Save 保存配置，准备烧写。

5、点击烧录工具主界面的 Start 按钮。

6、开发板进入烧写状态，先按下开发板的 BOOT_SEL1 按键，然后按下 RST_N 按键，待蓝色指示灯亮起后，再分别松开 RST_N 和 BOOTL_SEL1 按键。

7、等待烧录工具各项烧录到 100%，如图 3-4 所示。

8、烧录完成后从 SD 卡启动：先按下开发板的 BOOT_SEL0 按键，然后按下 RST_N 按键，待蓝色指示灯亮起后，再分别松开 RST_N 和 BOOTL_SEL0 按键。

4.4 烧录 eMMC

Halley5 平台本身没有 eMMC，由于 eMMC 与 SD 类似，可以参考 SD 烧录过程。

4.5 烧录工具的擦除选项

本烧录工具擦除选项有整体擦除和非整体擦除，在第一次烧写时需要选择整体擦除，如果后续只想烧录某一个文件则可以将该选项去掉。相关配置界面如图 3-11 所示。

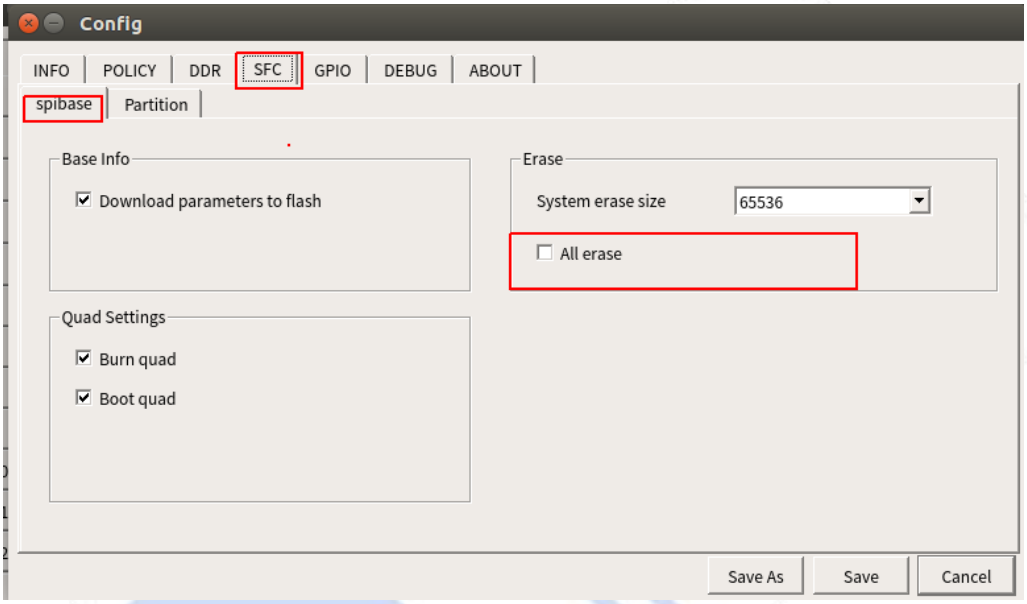


图 3-11

4.6 其他配置

4.6.1 配置串口

串口是开发过程中最为重要的工具之一，在使用串口调试之前需要进行串口环境的配置：

1、在 ubuntu 安装串口工具，执行命令：

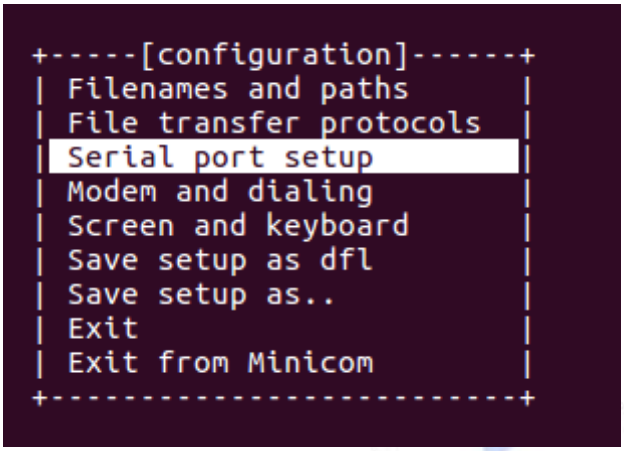
```
$    sudo apt-get install minicom
```

2、串口工具配置

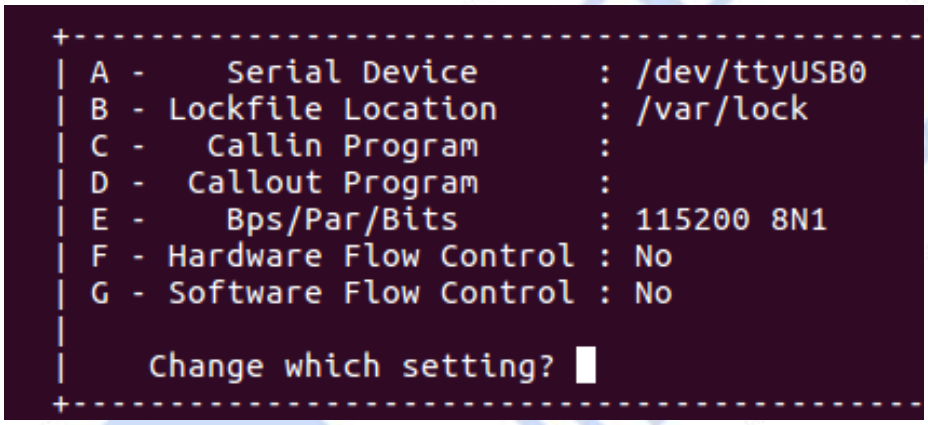
minicom 配置，再 PC 终端输入命令：

```
$    sudo minicom -s
```

命令执行后会进入如下界面：



选择 Serial port setup，回车进入串口配置，通过输入字母选择需要修改配置的选项，如下图所示：



关于配置选项的一些说明：1、Serial Device 需要根据实际设备名称来选择，通常是 ttyUSBx。3、选项 E，配置具体意义为波特率 115200,8 个数据位，无校验位以及一个停止位。3、选项 F 和 G，硬件和软件流控选项必须是 No。配置结束后回车进入 1 中的界面，选择 Save setup as dfl 保存配置，然后选择 Exit from minicom 退出。

4、串口工具使用

在终端下输入命令：

```
$    sudo minicom
```

4.6.2 安全烧录

参考 USBCloner 烧录工具说明文档.pdf

4.6.3 SN 烧录

参考 USBCloner 烧录工具说明文档.pdf

5 SD 卡烧工具使用

本节将介绍 SD 卡烧录工具的使用，该烧录方式可以通过读卡器将镜像写入到 SD 卡并启动，从而实现对板载的 Nand/Nor Flash 的烧写，该方法主要针对不方便使用 USB 烧写的情景。使用 SD 卡对 Nand 或者 Nor 进行镜像的烧写必须经过三个步骤：

a) SD 卡启动镜像的制作和烧写。
b) Nand/Nor 镜像的制作和写入到 SD 卡。
c) 从开发板 SD 卡启动并烧写 Nand/Nor。

5.1 SD 卡启动镜像制作

本小节介绍如何编译制作用于 SD 卡启动的镜像（*如果使用烧录工具提供的镜像文件，可以直接跳转至 5.2 小节*）。

1、进入工程目录下，执行命令：

```
$    source build/envsetup.sh

$    lunch
```

2、根据实际需求选择相应的配置：

```
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
```

3、修改设备树，由于 SD 卡烧写需要通过 device tree 获取 nand/nor flash 的分区信息，所以需要修改工程目录/kernel-4.4.94/arch/mips/boot/dts/ingenic/halley5_v20.dts，将 ingenic,use_ofpart_info = /bits/ 8 <0>中的/bits/ 8 <0>改为/bits/ 8 <1>，具体可参考图 4-1。

```
&sfc {
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&sfc_pe>;
        ingenic,sfc-max-frequency = <400000000>;
        ingenic,use_ofpart_info  = /bits/ 8 <1>;
        ingenic,spiflash_param_offset = <0>;

        norflash@0 {
                partitions {
                        compatible = "fixed-partitions";
                        #address-cells = <1>;
                        #size-cells = <1>;

                        /* spi nor flash partition */
                        uboot@0 {
                                label = "uboot";
                                reg = <0x0000000 0x40000>;
                                /*read-only;*/
                        };

                        kernel@40000 {
                                label = "kernel";
                                reg = <0x40000 0x300000>;
                        };

                        rootfs@360000 {
                                label = "rootfs";
                                reg = <0x360000 0xca0000>;
                        };
                };
        };
```

图 4-1

4、修改 UBOOT 配置，X2000 芯片内置的是 LPDDR3，X2000E 内置 LPDDR2。因此在不使用 USB 烧录

工具的情况下需要修改 u-boot 的配置，选择对应的 DDR 类型。修改文件路径：工程目录/u-boot/include/configs/halley5.h ，修改配置参考图 4-2。

```
#define CONFIG_DDR_INNOPHY
#define CONFIG_DDR_DLL_OFF
#define CONFIG_DDR_PARAMS_CREATOR
#define CONFIG_DDR_HOST_CC
/*#define CONFIG_DDR_TYPE_DDR3*/
#define CONFIG_DDR_TYPE_LPDDR3
/*#define CONFIG_DDR_TYPE_LPDDR2*/
#define CONFIG_DDR_CS0                  1       /* 1-connected, 0-disconnected */
#define CONFIG_DDR_CS1                  0       /* 1-connected, 0-disconnected */
#define CONFIG_DDR_DW32                 0       /* 1-32bit-width, 0-16bit-width */
/*#define CONFIG_DDR3_IS
```

图 4-2

5、在工程目录下执行编译命令：

```
$    make -j8
```

6、编译完成后在工程目录/out/product/halley5/image 下会重新生成相应的镜像文件（kernel、system.ext2、uboot）。

5.2 SD 卡启动镜像烧录

本小节主要讲述如何通过 PC 机将 SD 卡启动镜像写入到 SD 卡上指定位置，需要注意的是 SD 卡的设备节点/dev/sd*需要根据 PC 机上的实际名称自行确定，，勿直接执行以下步骤中的命令。烧录到 SD 卡的镜像文件可以是自己编译生成的，也可以直接使用烧录工具提供的镜像文件，对应关系如下表所示。

	自行编译	烧录工具自带的镜像文件	
命令中的文件名	X2000/X2000E	X2000	X2000E
uboot	uboot	uboot_lpddr3	uboot_lpddr2
kernel	kernel	kernel	kernel
system.ext2	system.ext2	system.ext2	system.ext2

1、将编译后生成的三个镜像文件拷贝到烧录工具目录下（*如果使用工具自带的镜像可忽略此步*）。

2、烧写 uboot 到 SD 卡指定位置，偏移为 0

```
$    sudo ./burn_sd.sh 0 uboot /dev/sd*
```

3、烧写 kernel 到 SD 卡指定位置，偏移为 6144

```
$    sudo ./burn_sd.sh 6144 kernel /dev/sd*
```

4、烧写 system.ext2 到 SD 卡指定位置，偏移为 409600

```
$    sudo ./burn_sd.sh 409600 system.ext2 /dev/sd*
```

5、烧录完成后，SD 卡被分为了 8 个分区,这里需要将第八个分区格式化用于存放要烧录到 nand/nor 的镜像以及配置文件。具体步步骤如下：

a. 格式化分区

```
$    sudo mkfs.ext2 /dev/sd*8
```

b. 挂载分区到 PC 机/mnt 目录

```
$    sudo mount /dev/sd*8 /mnt
```

c. 进入到/mnt 目录下，并创建 image 目录用于存放需要烧写到 nor/nand 的镜像文件

```
$    cd /mnt

$    sudo mkdir image
```

6、至此，和 SD 卡相关的镜像已经制作完成，下面就如何使用 SD 卡烧写 nor 和 nand 分别讲述。

5.3 Nand 镜像制作和烧录

5.3.1 烧录 Nand 镜像制作

1、编译用于烧写到 nand flash 的镜像文件

a. 打开 uboot 中 flash 参数配置功能相关的宏，器对应的路径位于工程目录/u-boot/include/configs/halley5.h，打开如下配置的宏：

```
#define CONFIG_NAND_BUILTIN_PARAMS
```

b. 修改工程目录/u-boot/tools/ingenic-tools/sfc_builtin_params/nand_device.c，中的相关分区使其保持和设备树文件 halley5_v20.dts 中的&sfc->nandflash 节点的分区一致。

c. 在工程目录下执行命令，配置 nand 的编译环境，这里以 halley5.v20_nand_4.4.94-eng 为例。

```
$    lunch halley5.v20_nand_4.4.94-eng
```

d. 在工程目录下执行命令，重新编译生成相应的镜像文件

```
$    make  -j8
```

2、在 SD 卡烧录工具下新建一个目录，用于存放需要烧写到 nand 的镜像文件，注意这里的镜像文件应当是在使用 lunch 命令选择对应 nand 配置后重新编译后生成的镜像文件，请注意区分。

a.新建目录

```
$    mkdir nand_burn_file
```

b.将 SD 卡烧录工具目录下的 add_chip_info_host 拷贝到新建的目录下

```
$    cp add_chip_info_host nand_burn_file
```

c.将步骤 1 中编译后生成的三个镜像文件拷贝到新建的目录下

3、进入到上述新建的目录下，使用 add_chip_info_host 工具，处理编译后生成的 uboot 镜像，以区分不同的芯片，该工具位于 sd 卡烧录工具目录下，以 X2000 芯片为例直接在该目录下执行命令：

```
$    ./add_chip_info_host uboot 128 X2000          /* 针对 X2000 的命令 */                    $
./add_chip_info_host uboot 128 X2000E          /* 针对 X2000E 的命令 */

$    mv uboot U-BOOT.bin
```

4、制作 system.ubi，工程编译后生成 system.ubifs 是裸镜像，需要制作 ubi 卷标。

a.制作 ubinize 配置文件 ubinize.ini，放在 nand_burn_file 目录

参考文件 buildroot/package/ingenic/system_config/sd_burn/ubinize.ini（可直接将该文件拷贝一份）

b.开始制作 system.ubi，在 nand_burn_file 目录下执行命令：

```
$  ubinize -o system.ubi -p 131072 -m 2048 -s 2048 ubinize.ini
```

ubinize 参数：

-o：输出文件名。

-m：最小输入输出大小为 2KiB(2048bytes)，一般为页大小。

-p：物理可擦出块大小为 128KiB=每块的页数*页大小=64*2KiB=128KiB。

-s：用于 UBI 头部信息的最小输入输出单元，一般与最小输入输出单元(-m 参数)大小一样。

5、nand_burn.ini 制作，具体可参考工程目录下/prebuilts/sd-burntools/nand_burn.ini，配置的语法如图 4-3，如有需要可以自行修改。*（注意：nand_burn.ini 中的分区需要和步骤 1 中的分区保持严格一致如果需要添加分区，则需要对三个位置的配置进行修改）。*

```
[option]
erase=0/1
烧录文件到分区之前是否需要擦除该分区

[partition]
part0=nand-boot,0x0,0x600000,mtd0
part0:      第一个分区
nand-boot:  分区名
0x0:        分区偏移地址
0x600000:   分区大小
mtd0:       mtd分区号

[file]
file0=BOOT.bin,0x0
file0:      第一个文件
BOOT.bin:   文件名
0x0:        文件在nand中的偏移(不是分区内偏移，可以一个分区烧录多个文件)
```

图 4-3

5.3.2 Nand 镜像写入到 SD 卡

1、将制作好的 U-BOOT.bin、kernel、system.ubi 拷贝到 SD 卡的第八个分区的 image 目录下。在拷贝之前需要确认一下 sd 卡的第八个分区是否已挂载到 pc 机的/mnt 目录下，如果没有正确挂载请参考 4.2 节最后的说明。

```
$     ls /mnt                    /* 如果正确挂载  mnt 目录下应该有一个名为 image 的目录 */

$    sudo cp U-BOOT.bin kernel system.ubi /mnt/image/
```

2、将制作好的 nand_burn.ini 文件拷贝到 SD 卡的第八个分区下。

```
$    sudo cp nand_burn.ini /mnt/
```

至此烧录前的准备工作已完成，SD 卡第八个分区下的目录结构如下：

```
├── image
│   ├── kernel
│   ├── system.ubi
│   └── U-BOOT.bin
├── lost+found
└── nand_burn.ini
```

5.3.3 SD 卡启动烧录

1、将 SD 卡插入开发板，进行 SD 卡烧录 nand flash 操作：先按下开发板的 BOOT_SEL0 按键，然后按下 RST_N 按键，待蓝色指示灯亮起后，再分别松开 RST_N 和 BOOTL_SEL0 按键。

2、打印如图 4-5 信息则表示烧写成功。

```
=============== Start Burn ================
    .........
=============== Nand info ===============
    ........
=========================================
Partition:nand-uboot,   MTD:mtd0,       Offset:0x0,     Size:0x100000
Eraseing /dev/mtd0...
Erasing 128 Kibyt[    7.208980] random: nonblocking pool is initialized
e @ e0000 -- 100 % complete
Write /usr/mmcdata/mmcblk0p8/image/U-BOOT.bin to /dev/mtd0 at offset 0
Writing data to block 0 at offset 0x0
    .......
Partition:nand-kernel,  MTD:mtd1,       Offset:0x100000,        Size:0x800000
Eraseing /dev/mtd1...
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
Write /usr/mmcdata/mmcblk0p8/image/kernel to /dev/mtd1 at offset 0
Writing data to block 0 at offset 0x0
Writing data to block 1 at offset 0x20000
    ........
Write /usr/mmcdata/mmcblk0p8/image/system.ubi to /dev/mtd2 at offset 0
Writing data to block 0 at offset 0x0
Writing data to block 1 at offset 0x20000
    .......
Writing data to block 493 at offset 0x3da0000
handleSuccess
=============== Burn Success ================
```

图 4-5

5.4 Nor 镜像制作和烧录

详细介绍烧录 Nor 需要做的配置

5.5 其他

6 运行测试应用

本节将主要讲述如何执行相关模块的测试程序，以及测试程序正常执行后的效果。

6.1 系统启动成功标志

系统烧写完成后，会自行启动进入内核并挂载文件系统，正常进入系统后可以通过串口终端输入命令并正确的执行，成功启动的标志如图 6-1 所示。

```
# [   18.177040] dwc2 13500000.otg: bound driver configfs-gadget
[   18.578884] dwc2 13500000.otg: new device is high-speed
[   18.650942] dwc2 13500000.otg: new device is high-speed
[   18.711714] dwc2 13500000.otg: new address 4
[   18.736752] configfs-gadget gadget: high-speed config #1: c

#
# ls
bin         firmware    linuxrc     opt         run         target      usr
dev         lib         media       proc        sbin        testsuite   var
etc         lib32       mnt         root        sys         tmp
#
```

图 6-1

6.2 屏幕显示

本小节主要讲述执行 LCD 的测试应用程序，在液晶上显示图片来测试屏幕的正常显示。

测试前的条件说明：　1、正确连接开发板配套的液晶屏幕。

2、在 nand 启动或在 SD 卡启动的条件下进行测试。

3、开发板两个 USB 均与 PC 机连通

测试步骤：1、开发板上电启动，正常启动后会 LCD 会显示君正的 logo，如图 6-2 所示。

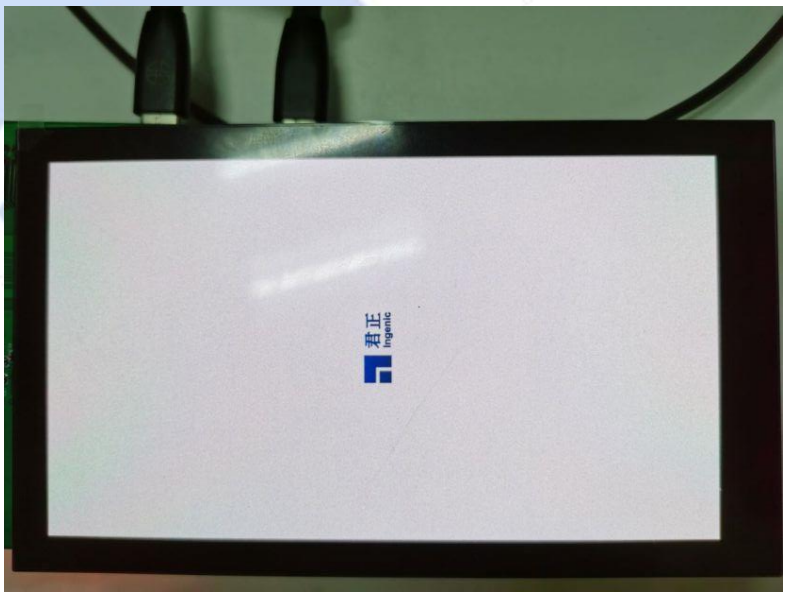


图 6-2

2、在开发板串口终端执行如下命令，使 LCD 显示图片，显示效果如图 6-3 所示。

```
#     cd /testsuite

#     ./dpu -n 100

/* dpu 命令参数解释:    -n 100 表示刷新 1 0 0 帧图像后结束测试程序 */
```

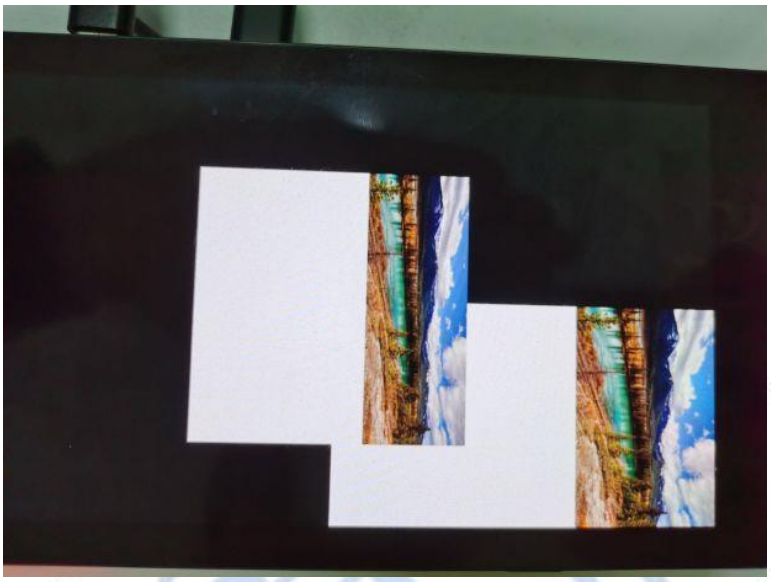


图 6-3

6.3 双摄像头预览

本小节主要讲述如何对双目摄像头进行预览测试，预览测试达到的效果是 LCD 实时显示摄像头采集到的图像数据。

测试方法：在开发板串口终端执行如下命令

```
#     cd /testsuite

#     ./Dual-Camera.sh
```

命令执行完成后会立即在 LCD 上显示实时画面，如图 6-4 所示。

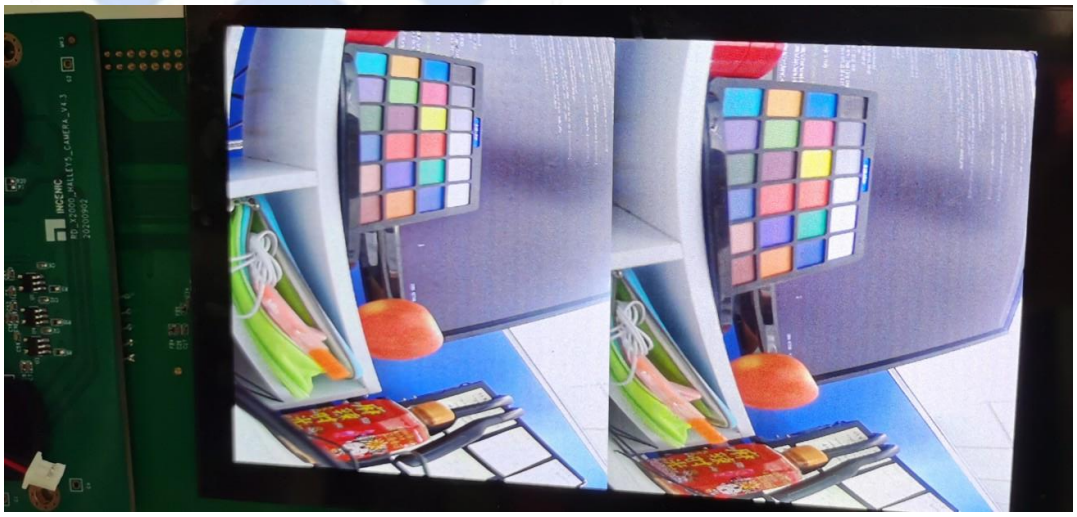


图 6-4

需要说明的是，当前 SDK 默认配置的双摄模组的版本是 V4.2 的，针对其他版本需要在重新配置。配置的具体方法请参考内核开发手册的相关内容。

6.4 音频播放和录制

本小节主要讲述如何使用开发板内置的声卡进行声音的录制和播放测试。

测试的前提条件：扬声器和开发板的接口已连接，如图 6-5 所示。

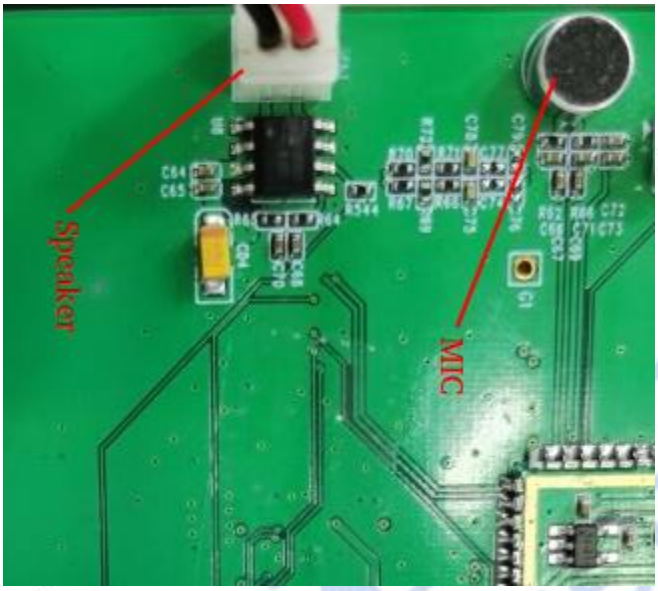


图 6-5

1、声音播放测试

a. 在工程目录 /packages/example/App/linphone-desktop/linphone-sdk/mediastreamer2/tester/sounds 任意选取一个 wav 文件，这里选择 hello16000.wav 文件，将该文件通过 adb push 方式从 PC 机传输到开发板的根文件目录下，adb 命令演示：

```
$ adb push hello16000.wav /
```

b. 在开发板的根目录下执行命令播放该 wav 文件

```
# amixer cset name='LO0_MUX' LI8

# aplay -D hw:0,0 hello16000.wav
```

c. 命令执行后即可听见扬声器播放的声音

2、声音录制测试

在开发板执行命令

```
# amixer cset name='LO6_MUX' LI2

# arecord -D hw:0,6 -c 1 -f S32_LE -r 16000  -d 5 baic0_16000-32-1.wav
```

命令执行后可以使用手机在 MIC 附近播放音乐，待声音录制程序执行结束后，可以按照声音播放的步骤来播放录制的声音。

7 FAQ

7.1 如何将编译好的可执行程序模块导入至开发板？

A:导入程序的方式有 adb、tftp 等方式，具体方法可参考<<Halley5 平台应用开发手册>>。

7.2 编译 kernel 出现问题

```
/home1/txiong/work/SDK/x2000_20201120/kernel-4.4.94 is not clean, please run 'make mrproper'
  in the '/home1/txiong/work/SDK/x2000_20201120/kernel-4.4.94' directory.
/home1/txiong/work/SDK/x2000_20201120/kernel-4.4.94/Makefile:997: recipe for target 'prepare3' failed
make[2]: *** [prepare3] Error 1
make[2]:Leavingdirectory '/home1/txiong/work/SDK/x2000_20201120/out/product/halley5/obj/kernel-intermediate'
Makefile:150: recipe for target 'sub-make' failed
```

出现上述现象通常是因为在 kernel 目录下执行了 make menuconfig 进行了配置，然后到工程目录下执行编译命令造成的。这个时候的解决办法:

1、到 kernel 目录下执行命令

```
$ make distclean
```

2、回到工程顶层目录执行 lunch 选择板级后，并执行命令编译内核

```
$ make kernel -j8
```

提示：本 SDK 所有的编译配置都可以在工程顶层目录进行编译配置，具体参考本文档相关内容，不建议进入到某个目录下单独进行配置。

7.3 USB 烧录失败

换一根 typeC 线或者换一个 USB 口重新尝试烧录，如果还是不能烧录成功，发送邮件到 support@ingenic.com

7.4 程序无法执行或者编译时链接失败

确保编译器使用 gcc version 7.2.0 (Ingenic r4.0.0-gcc720 2018.02-28)执行 mips-linux-gnu-gcc -v 可以查看.

```
Using built-in specs.
COLLECT_GCC=mips-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/home1/txiong/work/SDK/x1000/prebuilts/toolchains/mips-gcc520-glibc222/bin/../libexec/gcc/mips-linux-gnu/5.2.0/lto-wrapper
Target: mips-linux-gnu
Configured with: /home/toolchains/release/src/gcc-5-2015.11/configure
--build=i686-pc-linux-gnu --host=i686-pc-linux-gnu --target=mips-linux-gnu --enable-threads
--disable-libmudflap --disable-libssp --disable-libstdcxx-pch --with-arch-32=mips32r2
--with-arch-64=mips64r2 --with-float=hard --with-mips-plt --enable-extra-sgxxlite-multilibs
--with-gnu-as --with-gnu-ld
.....................................
--with-build-time-tools=/home/toolchains/release/install/opt/codesourcery/mips-linux-gnu/
```

```
bin
--with-build-time-tools=/home/toolchains/release/install/opt/codesourcery/mips-linux-gnu/
bin SED=sed
Thread model: posix
gcc version 5.2.0 (Ingenic r3.2.1-gcc520 2017.12-15)
```

如果没有上述现象，尝试进行编译环境变量设置和 lunch 板级选择，参考本文档的 SDK 编译部分的内容。