

**君正®**

**Kiva-SDK 开发指南**

Date: April. 2022

**君正®**

**Kiva-SDK 开发指南**

Copyright © Ingenic Semiconductor Co. Ltd 2022. All rights reserved.

**Release history**

|  |  |  |  |
| --- | --- | --- | --- |
| Date |  | Revision | Change |
| Feb.2022 |  | 1.0 | First release |

**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.

**北京君正集成电路股份有限公司**

**地址: 北京市海淀区西北旺东路10号院东区14号楼君正大厦**

**电话:(86-10)56345000**

**传真:(86-10)56345001**

**Http: //www.ingenic.cn**

**合肥君正科技有限公司**

**地址：安徽省合肥高新区望江西路800号创新园C3楼9层**

**电话：[(86-0551)68998700](C:/Users/Administrator/AppData/Roaming/Foxmail7/Temp-14820-20220210114029/Tel:(86-0551)68998700-)**

**传真：[(86-0551)68998701](C:/Users/Administrator/AppData/Roaming/Foxmail7/Temp-14820-20220210114029/Tel:(86-0551)68998700-)**

**Http: //www.ingenic.cn**

**目录**

[1. 开发资源介绍 3](#_Toc21647)

[1.1. Kiva SDK介绍 3](#_Toc3237)

[2. Kiva应用代码介绍 4](#_Toc26006)

[2.1. Kiva固件 4](#_Toc21399)

[2.1.1. 固件验证 4](#_Toc27081)

[2.1.2. Kiva固件制作 4](#_Toc26238)

[2.2. Kiva源码结构介绍 4](#_Toc21169)

[2.2.1. 目录结构 4](#_Toc18069)

[3. Kiva 系统构建 5](#_Toc25378)

[3.1. Uboot编译 5](#_Toc32702)

[3.2. Kernel编译 6](#_Toc596)

[3.3. 驱动编译 6](#_Toc9156)

[3.3.1. 视频芯片平台需要加载的驱动 6](#_Toc27787)

[3.3.2. USB Camera 需要加载的驱动 7](#_Toc20700)

[3.4. 应用程序编译 7](#_Toc22056)

[3.5. 文件系统的制作 7](#_Toc9465)

[3.5.1. 修改根文件系统 7](#_Toc31771)

[3.5.2. 打包文件系统 8](#_Toc19690)

[4. 高级特性功能移植说明 9](#_Toc13580)

[4.1. AF功能 9](#_Toc5481)

[4.1.1. 功能简介 9](#_Toc28859)

[4.1.2. 移植步骤 9](#_Toc21877)

[4.1.3. 注意事项 10](#_Toc16228)

[4.2. C位功能 11](#_Toc1203)

[4.2.1. 功能简介 11](#_Toc2339)

[4.2.2. 移植步骤 11](#_Toc9645)

[4.2.3. 注意事项（T31和T40的差异点） 12](#_Toc7603)

[4.3. PTZ云台相机功能 14](#_Toc11041)

[4.4. 快速启动 14](#_Toc18181)

[4.4.1. iboot编译 14](#_Toc30917)

[4.4.2. 内核编译 15](#_Toc7533)

[4.4.3. 内核挂载 15](#_Toc25386)

[4.5. CDC升级 15](#_Toc4743)

[5. 功能验证 16](#_Toc28573)

[5.1. 音频 16](#_Toc4102)

[5.1.1. 麦克风录音功能测试 16](#_Toc23612)

[5.1.2. 麦克风音量调节功能测试 17](#_Toc12791)

[5.2. 视频 18](#_Toc31135)

[5.2.1. 视频播放功能测试 18](#_Toc8271)

[5.2.2. 视频切换分辨率功能测试 19](#_Toc27321)

[5.2.3. 视频组件功能测试 20](#_Toc27066)

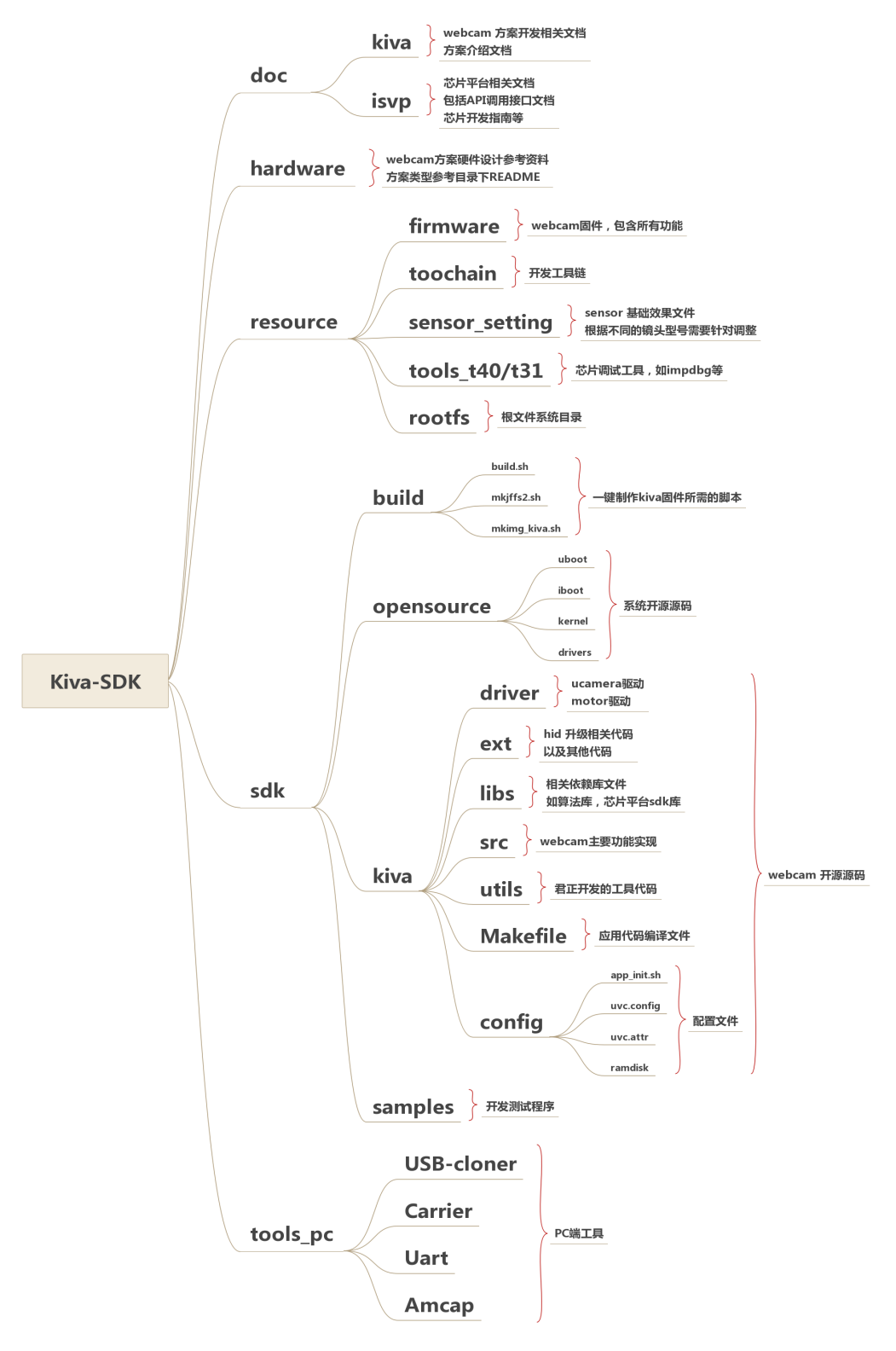
[5.3. HID升级功能 20](#_Toc17639)

[6. 其他文档 23](#_Toc4610)

# 开发资源介绍

## Kiva SDK介绍

Kiva是webcam uvc方案的名称，Kiva-SDK即WebCame方案软件开发工具包，包括开源源码、文档、Samples等等。开发者可以通过SDK快速的开展WebCam产品功能开发。以下是 Kiva SDK的内容概览图：

****

**图1-1 SDK组成结构**

# Kiva应用代码介绍

## Kiva固件

Kiva的固件烧录到板子中可以正常的启动uvc程序，包含基础的视频和音频，已满足大部分的需求。

### 固件验证

在SDK中有发布的完整固件包，路径为：resource/kiva\_firmware/，用户可以使用相应版本的固件先验证硬件的正确性，具体的验证方式见第5章功能验证

### Kiva固件制作

在sdk的kiva目录下提供编译完整固件的脚本build/build.sh，客户可以通过本固件一键编译完整的uvc功能固件，步骤如下：

cd Kiva/build/

./build.sh

脚本中的主要参数说明参考build.sh文件头部注释。

## Kiva源码结构介绍

### 目录结构

driver：usbcamera驱动和dw9714电机驱动

config: 配置文件，如uvc.config，uvc.attr等

ext：包含hid升级程序代码

libs: 相关依赖库，包括AI相关功能依赖的算法库，芯片平台的sdk库等

src: uvc应用代码逻辑部分以及功能实现代码

main.c: 程序入口代码，各功能模块初始化

imp\_common.c: 芯片sdk的初始化函数

modules: 各功能模块实现代码，如C位功能；具体使用请参考第4章高级特性功能移植说明

utils: 君正开发的一些工具代码

Makefile: 生成应用程序ucamera的编译文件

# Kiva 系统构建

本章节主要介绍Kiva 方案系统构建，包括uboot、kernel、文件系统以及整个固件的构建过程。对于开发者而言，建议参考此文档构建。

## Uboot编译

Uboot编译流程：

u-boot可单独编译，不依赖其他代码。u-boot的板机配置文件位于 include/configs/，如T31的isvp\_t31.h。默认编译配置文件介绍如表3-1。

**注释：1080p 常规建议用T31L/N，2k 方案推荐T31X，高级功能(比如跑自己算法)建议T31A或者T40N, 4K方案使用T40XP**

**表3-1 Kiva 各方案对应uboot编译文件**

|  |  |  |
| --- | --- | --- |
| 芯片型号 | 编译命令 | 说明 |
| T31N | make isvp\_t31\_sfcnor | 表示编译norflash启动的uboot，针对K1-2M方案 |
| T31L | isvp\_t31\_sfcnor\_lite\_k1\_2m | 表示编译norflash启动的uboot，针对K1-2M 方案 |
| T31X | make isvp\_t31\_sfcnor\_ddr128M | 表示编译norflash启动的uboot，针对K1-4M方案 |
| T31A | make isvp\_t31a\_sfcnor\_ddr128M | 表示编译norflash启动的uboot，针对K1-4M方案 |
| T40XP | make isvp\_t40xp\_sfcnor\_k1\_4k | 表示编译norflash启动的uboot，针对K1-4K方案 |

编译步骤：cd sdk/opensource/uboot

**第一步：**$ make distclean 清除旧配置

**第二步：**$ make isvp\_xxx 根据相应方案，编译对应的uboot，生成对应的u-boot-with-spl.bin

● Uboot配置文件中常见修改的地方：

1. CONFIG\_BOOTARGS，主要修改点是内核启动以后的内存配置，分区大小配置。

注：

1. mem表示内核启动以后保留内存，rmem表示预留给SDK的内存（包括ISP模块的内存）；T40中使用算法的话必须使用nmem设置npu所需的内存，两者或者三者相加为芯片真实内存大小；具体大小可参考代码。
2. appfs分区默认使用flash剩余大小，使用“-”表示，在具体使用中如果分区比实际的flash剩余空间小，建议把“-”改为实际的分区大小

2) CONFIG\_BOOTCOMMAND，配置uboot启动执行的命令。例如：norflash启动模式"sf probe;sf read 0x80600000 0x40000 0x280000; bootm 0x80600000"

3) CONFIG\_BOOTDELAY，配置uboot的等待时间。

4) 需要添加新的norflash 芯片的支持**。**

5) 更多uboot的使用请参考isvp中的芯片开发文档《Txx\_开发指南.pdf》

## Kernel编译

kernel可单独编译，不依赖其他代码。以isvp板级编译为例，进入kernel源码目录。在arch/mips/configs/ 文件夹下存放了内核的板级配置文件；Kiva 方案板级文件为kiva\_uvc\_defconfig；

**第一步：**使用相对配置好的板级文件 $ make kiva\_uvc\_defconfig

**第二步：**根据需求选择性编译 $ make menuconfig

**第三步：**编译内核文件 $ make uImage

注意：对于kernel之外的ko编译，依赖kernel，且kernel必须经过编译。每当kernel更改之后，可能会出现ko无法insmod的情况，或者可以insmod但会出现未知错误。这是因为kernel重新编译后，函数Symbol表有变化，需要重新编译ko driver以正确link函数。此时需要重新编译ko**。**

内核默认的config留有一定余量，而实际产品往往需要进行内核裁减以释放更多的空间。

注：内核的深度裁减有一定的技术难度，开发者尽量在深入了解配置的情况下再进行深度裁减。

**源码位置：sdk/opensource/kernel**

## 驱动编译

Kiva 驱动包括 视频芯片平台必要的驱动和USBCamera 需要的驱动。

### 视频芯片平台需要加载的驱动

平台驱动：sdk/opensource/drivers

**表3-2 T31 平台需要加载的驱动**

|  |  |
| --- | --- |
| 驱动名称 | 驱动介绍 |
| tx-isp-txx.ko | ISP驱动 |
| sensor\_xxxx\_txx.ko | Sensor驱动 |
| avpu.ko | 视频编码驱动 |
| audio.ko | 音频驱动 |
| soc-nna.ko | Npu驱动（T40） |

### USB Camera 需要加载的驱动

USB Camera 驱动：Kiva/driver/

**表3-3 USB Camera需要加载的驱动**

|  |  |
| --- | --- |
| 驱动名称 | 驱动介绍 |
| usbcamera.ko | usbcamera 驱动 |
| libcomposite.ko | (T31)Composite 驱动，usbcamera 依赖。编译完内核后，Make modules |
| videobuf2-vmalloc.ko | (T31)V4L2驱动，usbcamera 依赖。编译完内核后，Make modules |

## 应用程序编译

源码位置：sdk/kiva

编译：

cd sdk/kiva

make clean;

make;

**注释：默认支持uclibc**

## 文件系统的制作

对于NorFlash Based系统，存储空间一般较小，因此会选用压缩文件系统。在君正平台推荐以下两种文件系统：

A. squashfs：只读文件系统，压缩率高

B. jffs2：可读写文件系统，可选择压缩方式

一种系统搭建的方案是，系统的rootfs以及不需要经常修改的系统分区采用squashfs，而配置分区等需要经常读写的分区采用jffs2。

另外，文件系统有glibc与uclibc两种选择，glibc的特点是支持功能全面，但是占用存储稍多；uclibc的特点是占用存储空间小于glibc，但是支持功能比glibc稍少。开发者应该根据自己实际情况选用适合的libc方式。

对于一般的应用场景，推荐客户使用uclibc搭建系统，Kiva 方案默认使用uclibc。

### 修改根文件系统

可以将rootfs压缩包解压，替换需要修改的资源即可。例如，实际产品往往需要裁减或者自定义busybox，可以自行编译busybox，将busybox strip后copy到rootfs中，并将bin，sbin，user/bin，usr/sbin下的软连接通过-R的方式copy到目标目录。

默认根文件系统位置：resource/rootfs

如果用户需要修改根文件系统，推荐用户使用rootfs 压缩包解压修改，然后在打包成squashfs。

如果无任何修改，直接使用君正提供的即可，方便快速开发。

### 打包文件系统

squashfs打包方式：

mksquashfs rootfs[输入文件夹] rootfs.squashfs[输出文件名] -comp xz

jffs2打包方式：

mkfs.jffs2 -o jffs2.img[输出文件名] -r jffs2\_dir[输入文件夹] -e 0x8000[擦除大小32K] -s 0x40000[页大小256K] -n -l -X zlib --pad=0x300000[输出镜像pad到3MB大小]

建议通过设备端擦除再直接mount的方式创建jiff2分区。具体操作如下；

首先正确的编译出uboot，uImage，rootfs，appfs三个分区的内容；然后通过烧录器或其它烧录方法把norflash全部擦除，最后烧录到相应位置。

系统起来以后根据uboot的分区信息分别进行手动挂载，例如：mount -t jffs2 /dev/mtdblock3 /system；

注意：jffs2制作某个分区为jffs2文件，分区大小必须为nor erase\_size的整数倍，这个是官方驱动的要求。

# 高级特性功能移植说明

高级特性功能根据方案需求酌情添加，主要包含AF功能，C位功能，快速启动，以及后续其他新增功能，如PTZ云台相机功能，CDC升级等。

新增特性功能特别注意，带算法的功能，建议一个方案跑一个算法，保证系统基础性能不受影响。

## AF功能

### 功能简介

AF即AutoFocus（自动对焦），它是基于清晰度值的获取动态调整电机移动的位置，从而能够保证人或物体在不同的距离都能够清晰显示。

T31以及T40用的都是同一套对焦算法逻辑，后面介绍以T31的移植为模板。

该功能需要加载电机gsensor-dw9714.ko驱动，在kiva/driver/common/motor\_driver目录下。

### 移植步骤

实现流程：获取清晰度值->检测清晰度值变化趋势->移动电机->完成对焦

功能实现在module\_af\_control.c中。

主要函数说明：

外部调用接口，初始化和反初始化

/\*

\* 打开电机设备

\* 设置电机的默认移动位置

\* 创建主对焦以及辅助对焦线程

\*/

int module\_autofocus\_init(void \*param);

/\*

\* 关闭电机设备

\*/

int module\_autofocus\_deinit();

内部主要接口说明:

/\*

\*主对焦流程函数，包含如下内容

\* 1.设置AF图像权重

\* 2.AF触发逻辑

\* 3.调用主对焦算法

\*/

static void \*af\_control\_process(void \*args);

/\*

\* 辅助对焦流程函数,调用辅助对焦算法

\*/

static void \*af\_check\_process(void \*args);

/\*

\* 返回autoFocus接口

\*/

static int autoFocusClimb(int motor\_step);

/\*

\* 返回autoFocus接口

\*/

static int autoFocusDoubleCheck(int motor\_step);

/\*

\* 主对焦算法

\*/

static int autoFocus(int motor\_step, int trigger\_rate, int isDoubleCheck);

/\*

\* 辅助对焦算法

\*/

static int autoFocusCheck(int fd, int motor\_step);

### 注意事项

在运行AF之前需要对镜头进行定焦操作。由于不同的镜头有不同的规格，大致按照2M以及4M镜头两种规格，通过上位机或应用程序进行定焦。

#### 2M镜头定焦

2M的镜头按照375的电机code值（根据电流大小通过公式换算出的电机移动位置），对准10cm左右的物体，转动镜头直至物体清晰。

#### 4M镜头定焦

4M的镜头按照520的电机code值，对准1.2m左右的物体，转动镜头直至物体清晰。

这里对的物体尽量选择清晰度测试卡。

#### 上位机方式定焦

打开potplayer，将“焦点”自动关闭，手动按照上面描述进行对应调节电机code值，再转动镜头。



#### 应用程序定焦

在电机驱动中的sample目录，编译生成sample\_motor应用程序，拷贝到板子中运行，输入相应的电机code值，再转动镜头。在运行前也需将自动对焦关闭。

## C位功能

### 功能简介

基于人脸检测算法实现自动将人员框定在屏幕中央，主要有两种模式，单人模式和多人模式

单人模式是根据人脸距离跟随最近人脸移动跟踪，又称AI-EPTZ

多人模式是根据人员的多少调整屏幕框大小并保持居中，又称为AutoFraming

T40使用算法需要在bootargs中增加nmem参数如下图和加载soc-nna.ko驱动

bootargs=console=ttyS1,115200n8 mem=100M@0x0 rmem=128M@0x6400000 nmem=28M@0xE400000 init=/linuxrc rootfstype=squashfs root=/dev/mtdblock2 rw mtdparts=jz\_sfc:256k(boot),2048k(kernel),2560k(root),-(appfs)

### 移植步骤

实现流程：人脸检测->移动缩放策略->移动和缩放

功能实现在module\_face\_zoom.c中

主要函数说明：

外部调用接口，模块初始化和反初始化

 /\*

 \* facezoom功能初始化

 \* 次码流通道视频参数创建和使能

 \* 算法IVS通道创建和绑定

 \* 主要线程初始化

 \*/

 int module\_facezoom\_init(imp\_isp\_attr\_t isp\_param, facezoom\_func\_param\_t fz\_param);

 /\*

 \* facezoom功能反初始化

 \* 线程关闭

 \* 次码流通道失能和销毁

 \* 算法IVS通道解绑和销毁

 \*/

 int module\_facezoom\_deinit(facezoom\_func\_param\_t fz\_param);

内部主要函数说明：

/\*

\* 人脸检测算法流程,输出人脸检测结果

\*/

void \*face\_detect\_process(void\* none)；

/\*

\* facezoom实现流程,实现过程区别单人模式和多人模式

\*/

void \*face\_zoom\_process(void \*none)；

/\*

\* 单人模式实现流程

\*/

int face\_zoom\_single\_mode(void);

/\*

\* 多人模式实现流程

\*/

int face\_zoom\_multi\_mode(void);

/\*

\* 移动和缩放（fcrop）

\*/

int face\_zoom\_set(float scaler\_level, int face\_center\_x, int face\_center\_y);

### 注意事项（T31和T40的差异点）

T31平台和T40平台的fcrop的实现方式不同，所以在C位功能的实现上也略微有些差异，T31上使用fcrop会作用在所有通道上，T40上每一个通道有单独的fcrop设置

#### fcrop调用接口不同

/\* T31 \*/

IMPISPFrontCrop fcrop\_obj;

IMP\_ISP\_Tuning\_GetFrontCrop(&fcrop\_obj);

if((fcrop\_obj.fcrop\_left == zoomleft\_cur) && (fcrop\_obj.fcrop\_top == zoomtop\_cur) \

&& (fcrop\_obj.fcrop\_width == zoomwidth\_cur) && (fcrop\_obj.fcrop\_height == zoomheight\_cur)) {

return 0;

}

fcrop\_obj.fcrop\_enable = 1;

fcrop\_obj.fcrop\_left = zoomleft\_cur;

fcrop\_obj.fcrop\_top = zoomtop\_cur;

fcrop\_obj.fcrop\_width = zoomwidth\_cur;

fcrop\_obj.fcrop\_height = zoomheight\_cur;

ret = IMP\_ISP\_Tuning\_SetFrontCrop(&fcrop\_obj);

if (ret < 0) {

printf("IMP Set Fcrop failed=%d\n",\_\_LINE\_\_);

return -1;

}

/\* T40 \*/

IMPISPAutoZoom zoom\_attr;

IMP\_ISP\_Tuning\_GetAutoZoom(IMPVI\_MAIN, &zoom\_attr);

if((zoom\_attr.zoom\_left[0] == zoomleft\_cur) && (zoom\_attr.zoom\_top[0] == zoomtop\_cur) \

&& (zoom\_attr.zoom\_width[0] == zoomwidth\_cur) && (zoom\_attr.zoom\_height[0] == zoomheight\_cur)) {

return 0;

}

zoom\_attr.zoom\_chx\_en[0] = 1;

zoom\_attr.zoom\_left[0] = zoomleft\_cur;

zoom\_attr.zoom\_top[0] = zoomtop\_cur;

zoom\_attr.zoom\_width[0] = zoomwidth\_cur;

zoom\_attr.zoom\_height[0] = zoomheight\_cur;

ret = IMP\_ISP\_Tuning\_SetAutoZoom(IMPVI\_MAIN, &zoom\_attr);

if (ret < 0) {

Ucamera\_DEBUG("IMP Set Fcrop failed=%d\n",\_\_LINE\_\_);

return -1;

}

#### 坐标同步

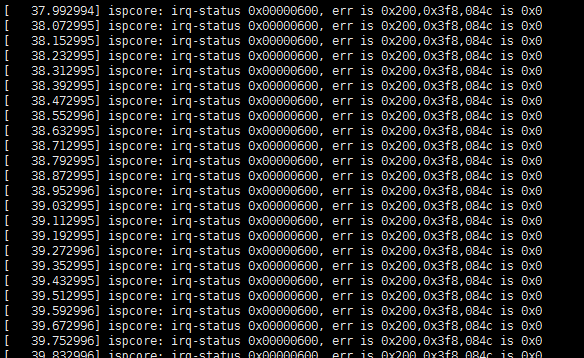
T31的人脸坐标要跟随前一次fcrop的坐标和缩放比例同步，具体体现为

face\_center\_x = last\_fcrop\_left + face\_zoom[face\_index].face\_center\_x \* ALGO\_CHN\_SCALER / last\_scaler;

face\_center\_y = last\_fcrop\_top + face\_zoom[face\_index].face\_center\_y \* ALGO\_CHN\_SCALER / last\_scaler;

#### 反初始化

T31在使用fcrop之后不可以只关闭主码流，所以开关流所有通道必须同步，所以uvc中切换分辨率视频流反初始化过程需要把facezoom的反初始化同步实行，实现方式是通过void \*face\_zoom\_control\_process(void \*none)线程控制C位功能的初始化和反初始化流程，否则会出现如下图错误



## PTZ云台相机功能

待更新...

## 快速启动

Kiva的快起功能主要配合使用iboot和带ramdisk的内核配置。

### iboot编译

|  |  |
| --- | --- |
| 芯片型号 | 编译命令 |
| T31N | make iboot\_t31\_sfcnor\_lite\_lzo |
| T31L | make iboot\_t31\_sfcnor\_lite\_lzo |
| T31X | make iboot\_t31\_sfcnor\_lzo |
| T31A | make iboot\_t31\_sfcnor\_lzo |
| T40XP | make iboot\_t40\_sfcnor\_lzo |

进入sdk/opensource/iboot

执行make iboot\_t40\_sfcnor\_lzo

使用iboot.bin

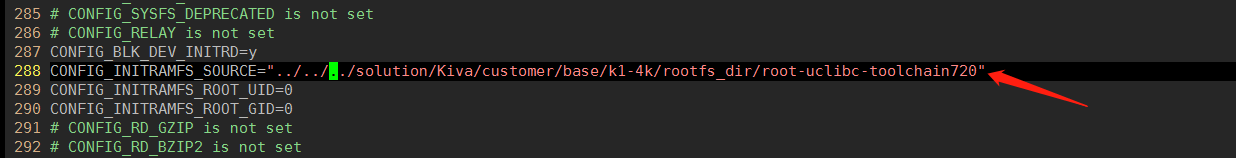
### 内核编译

1.ramdisk目录编译

进入sdk/kiva/config/ramdisk，使用root权限执行mknod.sh脚本生成rootfs目录

2.进入sdk/opensource/kernel

修改arch/mips/configs/isvp\_k1\_4k\_uvc\_quick\_start\_defconfig中的root路径为1.中生成rootfs文件夹所在路径(T31 内核配置文件为kiva\_uvc\_ramdisk\_defconfig)



3.执行make isvp\_k1\_4k\_uvc\_quick\_start\_defconfig

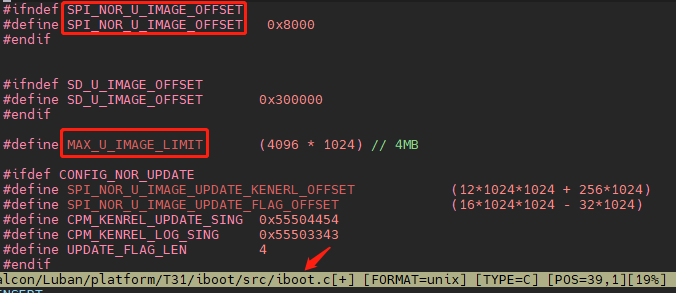
4.使用uImage

### 内核挂载

以T31 iboot为例（T40同理），在iboot/src/iboot.c中

1. SPI\_NOR\_U\_IMAGE\_OFFSET指定了内核的偏移地址0x8000即32k Byte。
2. MAX\_U\_IMAGE\_LIMIT指定了内核最大的大小4M Byte。

如需打包固件，请注意这两个参数，可根据需要自行调整。确保打包固件中内核分区偏移和iboot引导内核的地址偏移保持一致！



## CDC升级

待更新...

# 功能验证

开发人员通过君正研发的USB烧录工具将完整的固件包烧录之后可以验证硬件是否正常，包括音频验证和视频验证，烧录固件有疑问的参考/doc/isvp/Ingenictool/《USB-cloner烧录工具使用说明.pdf》

## 音频

### 麦克风录音功能测试

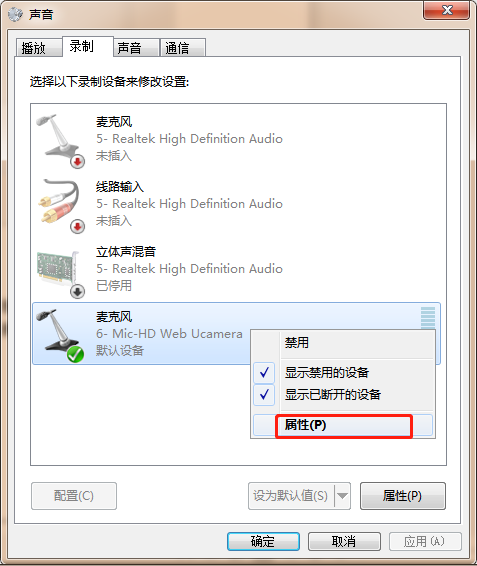
首先右击电脑右下方喇叭标志，如下图：



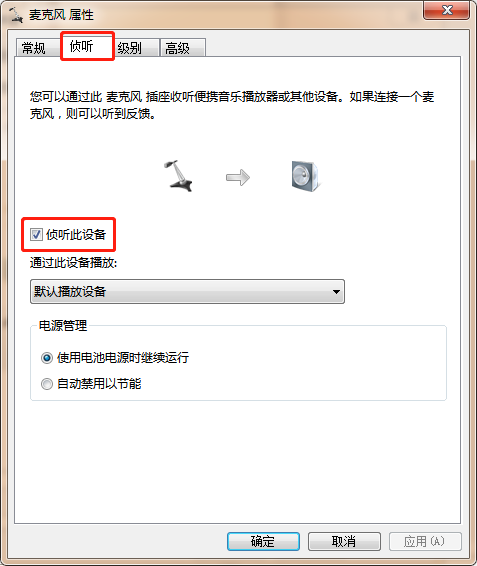
选择录音设备->麦克风



右键麦克风，选择属性



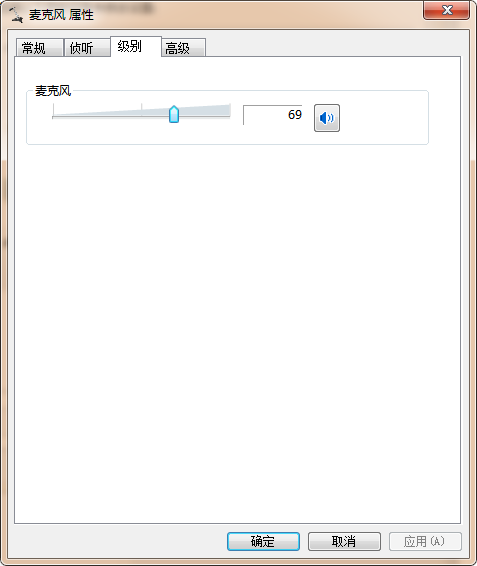
选择侦听->侦听此设备



此时电脑就会播放麦克风捕捉到的声音。因为设备和电脑距离较近，此时可能会发出刺耳噪音，属于正常信息。此时麦克风录音功能正常

### 麦克风音量调节功能测试

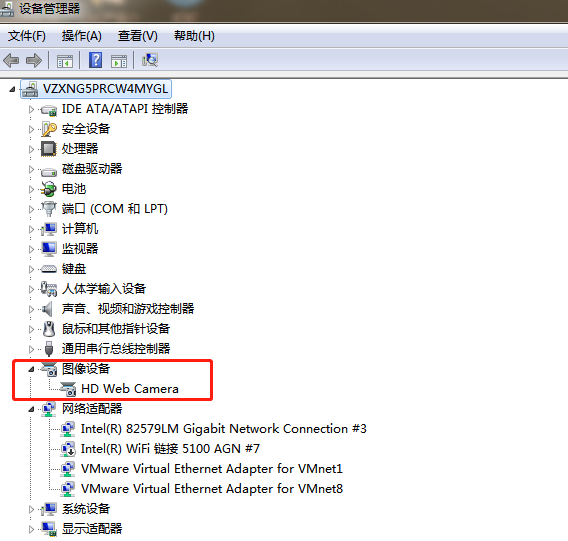
选择级别调节下方音量控制选项，如下图，查看音量是否变化，若音量变化，则音量调节功能正常



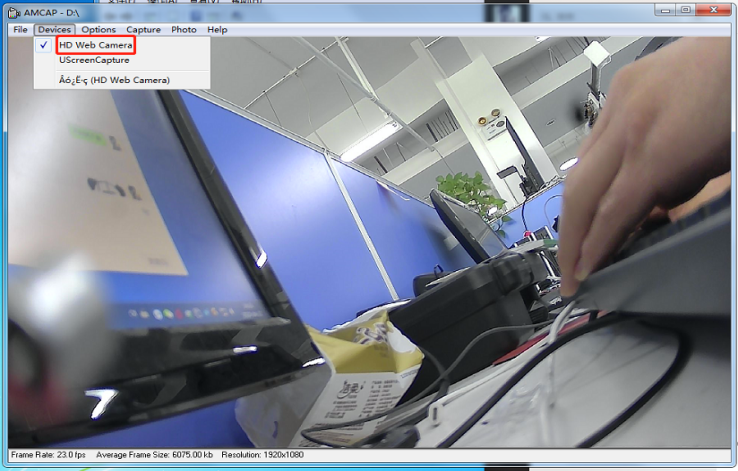
## 视频

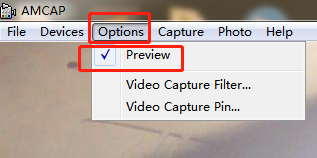
### 视频播放功能测试

将设备插入电脑中，设备管理器会出现图像设备



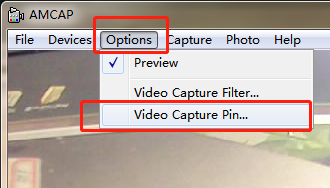
打开开发工具包中的amcap软件，打开devices->HD Web Camera，options->preview。此时就可以正常出图，若无异常则视频播放功能正常



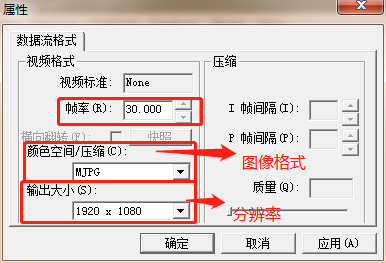


### 视频切换分辨率功能测试

点击option->video Capture pin



进入属性界面，从上到下依次为帧率，图像格式，分辨率信息，其中图像格式和分辨率可以调节

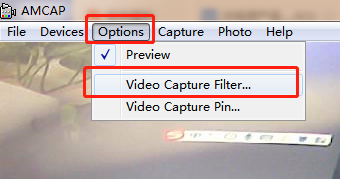


Amcap底部有基础信息，查看是否与属性调节的一致，若一致则分辨率与格式调节没有问题

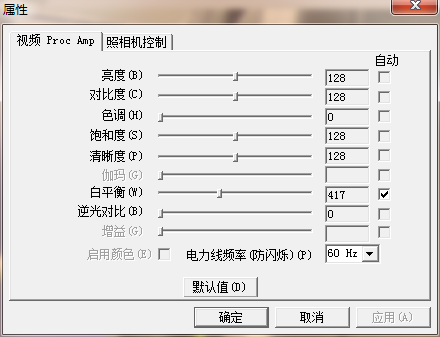


### 视频组件功能测试

打开option->Video Capture Filter



下列选项中只要不是灰色的都可以调节，请调节选项，查看图像是否发生变化，若发生变化则视频调节组件生效



## HID升级功能

将设备插入电脑中，打开工具包中的HID上位机工具，点击USB.exe运行即可

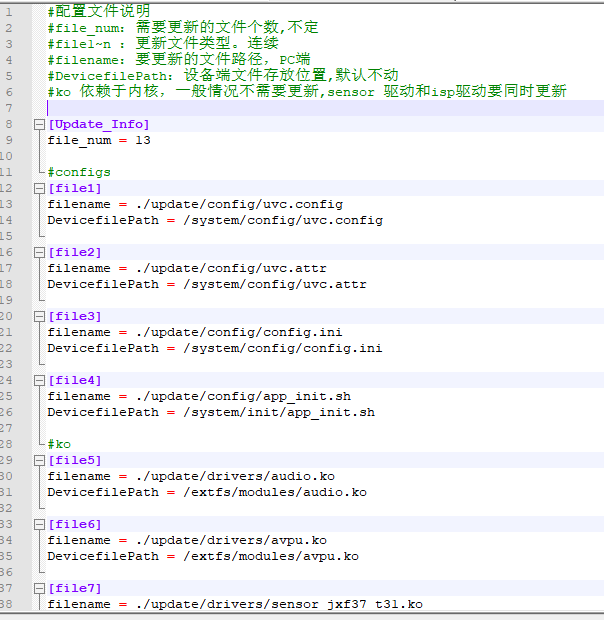
首先扫描并选择设备



点击扫描设备，获取待升级设备信息，选中

在升级操作前，修改配置文件：





例如：



只升级jxf37-t21.bin，配置file\_num = 1，同时配置升级文件的位置，待升级的上位机文件一定要放置到指定sj文件路径下。

修改配置文件后,按照上述操作，选择配置文件即可升级

# 其他文档

USB 烧录工具：

《USB烧录工具使用手册.pdf》

芯片平台开发文档：

《TXX\_开发指南.pdf》