君正®

Kiva-SDK 开发指南

Date: April. 2022

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

君正®

Kiva-SDK 开发指南

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**
传真：**(86-0551)68998701**
Http: //www.ingenic.cn

目录

1. 开发资源介绍

1.1. 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：

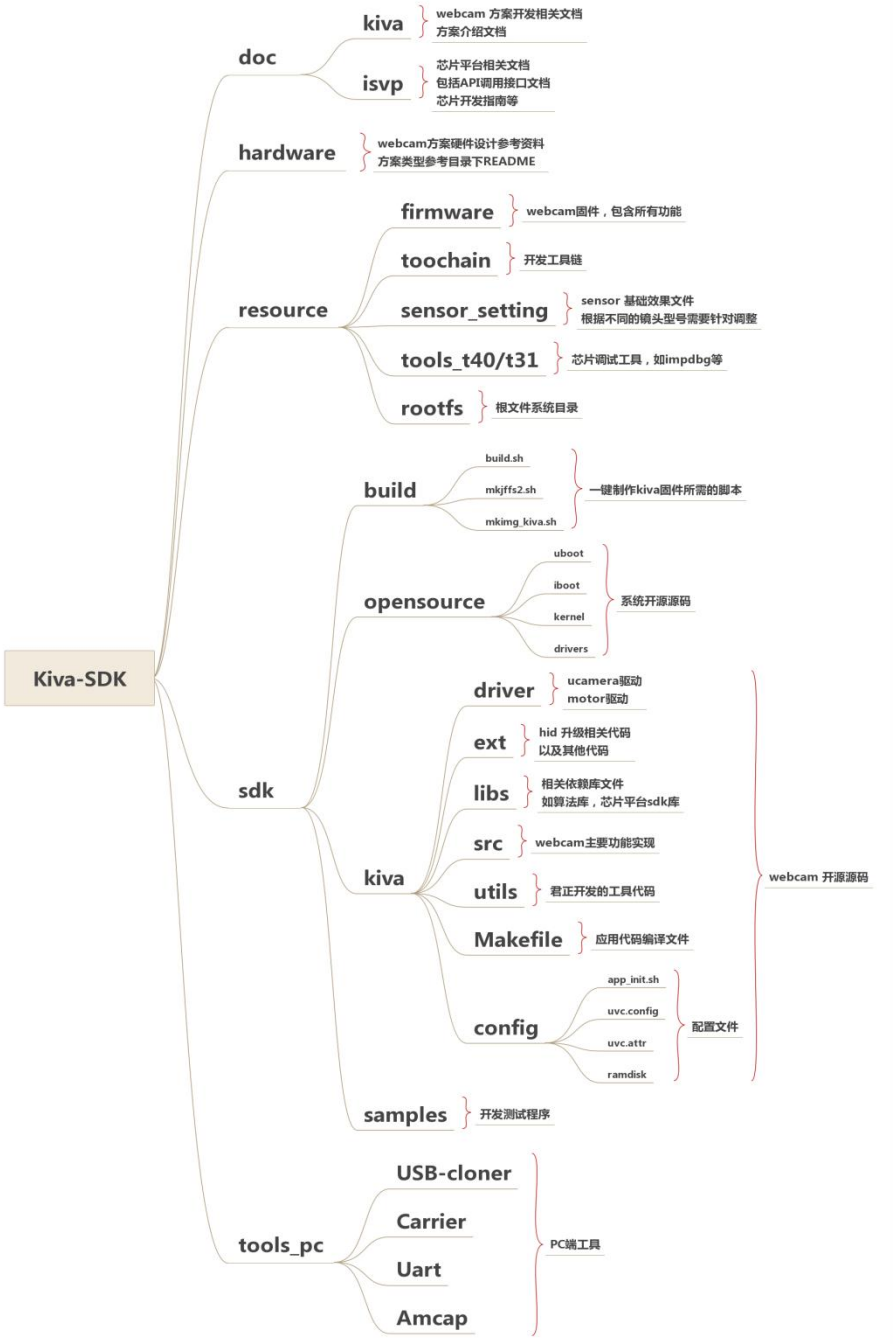


图 1-1 SDK 组成结构

2. Kiva 应用代码介绍

2.1. 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。

2.1.1. 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

2.1.2. 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。

2.2. Kiva 源码结构介绍

2.2.1. 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

3. 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。

3.1. 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》

3.2. 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

3.3. 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。

3.3.1. 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）

3.3.2. 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

3.4. 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

3.5. 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。

3.5.1. 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。

3.5.2.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。

4. 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。

4.1. AF 功能

4.1.1. 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。

4.1.2. 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);
```

4.1.3.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。

4.1.3.1. 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。

4.1.3.2. 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。

4.1.3.3. 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。

4.1.3.4. 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。

4.2. C 位功能

4.2.1. 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)
```

4.2.2. 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);
```

4.2.3.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

4.2.3.1. 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;
```

```
/* 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;
}
```

4.2.3.2. 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;
```

4.2.3.3. 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

```
[   37.992994] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.072995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.152995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.232995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.312995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.392995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.472995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.552996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.632995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.712995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.792995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.872995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   38.952996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.032995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.112995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.192995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.272996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.352995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.432995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.512995] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.592996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.672996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.752996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
[   39.832996] ispcore: irq-status 0x00000600, err is 0x200,0x3f8,084c is 0x0
```

4.3. PTZ 云台相机功能

待更新...

4.4. 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。

4.4.1. 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

4.4.2. 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)

```
285 # CONFIG_SYSFS_DEPRECATED is not set
286 # CONFIG_RELAY is not set
287 CONFIG_BLK_DEV_INITRD=y
288 CONFIG_INITRAMFS_SOURCE="../../../solution/Kiva/customer/base/k1-4k/rootfs_dir/root-uclibc-toolchain720"
289 CONFIG_INITRAMFS_ROOT_UID=0
290 CONFIG_INITRAMFS_ROOT_GID=0
291 # CONFIG_RD_GZIP is not set
292 # CONFIG_RD_BZIP2 is not set
```

3.执行 make isvp_k1_4k_uvc_quick_start_defconfig

4.使用 uImage

4.4.3. 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！

```
#ifndef SPI_NOR_U_IMAGE_OFFSET
#define SPI_NOR_U_IMAGE_OFFSET   0x8000
#endif


#ifndef SD_U_IMAGE_OFFSET
#define SD_U_IMAGE_OFFSET       0x300000
#endif

#define MAX_U_IMAGE_LIMIT       (4096 * 1024) // 4MB

#ifdef CONFIG_NOR_UPDATE
#define SPI_NOR_U_IMAGE_UPDATE_KENERL_OFFSET            (12*1024*1024 + 256*1024)
#define SPI_NOR_U_IMAGE_UPDATE_FLAG_OFFSET              (16*1024*1024 - 32*1024)
#define CPM_KENREL_UPDATE_SING  0x55504454
#define CPM_KENREL_LOG_SING     0x55503343
#define UPDATE_FLAG_LEN         4
#endif
lcon/Luban/platform/T31/iboot/src/iboot.c[+] [FORMAT=unix] [TYPE=C] [POS=39,1][19%]
```

4.5. CDC 升级

待更新...

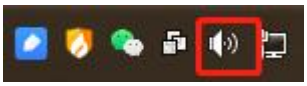
5. 功能验证

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》

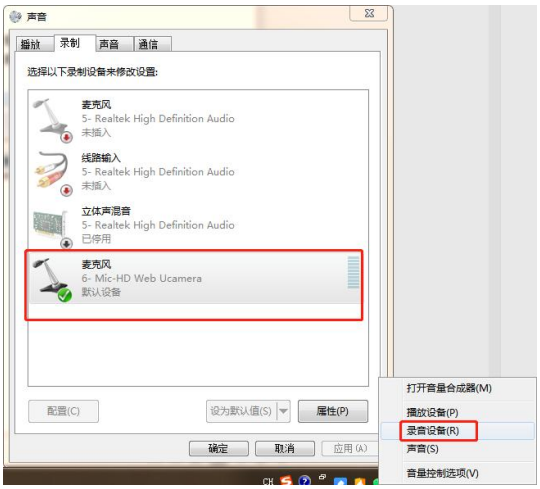
5.1. 音频

5.1.1. 麦克风录音功能测试

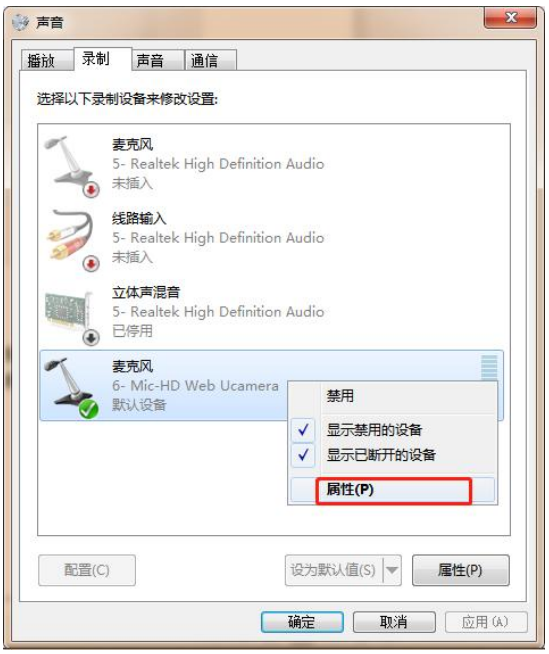
首先右击电脑右下方喇叭标志，如下图：



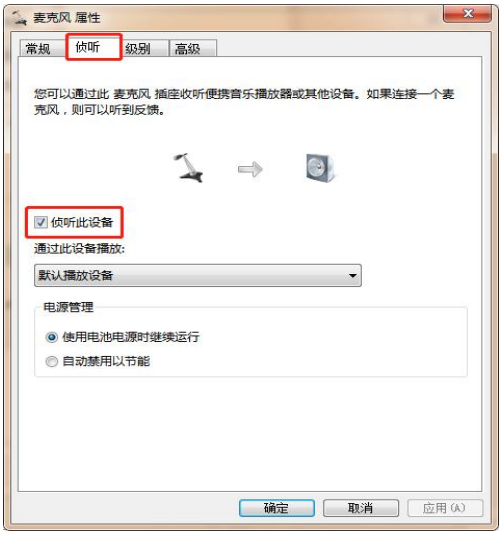
选择录音设备->麦克风



右键麦克风，选择属性



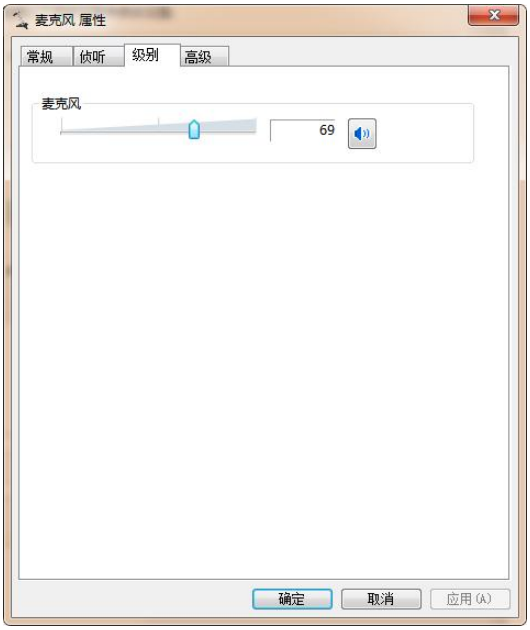
选择侦听->侦听此设备



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

5.1.2.麦克风音量调节功能测试

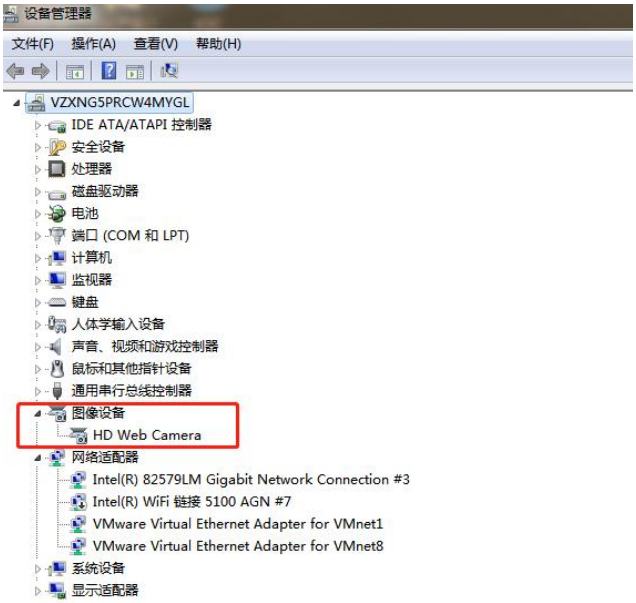
选择级别调节下方音量控制选项，如下图，查看音量是否变化，若音量变化，则音量调节功能正常



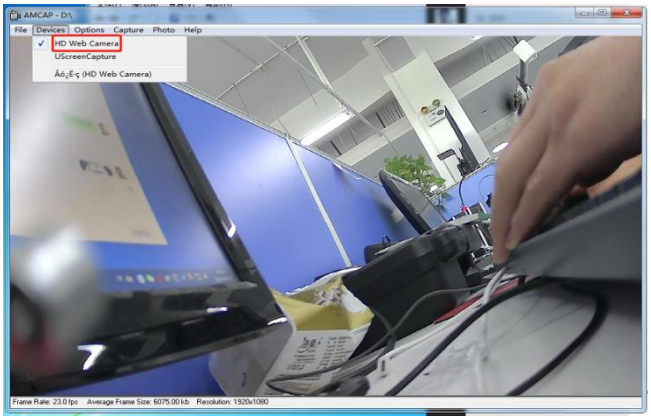
5.2. 视频

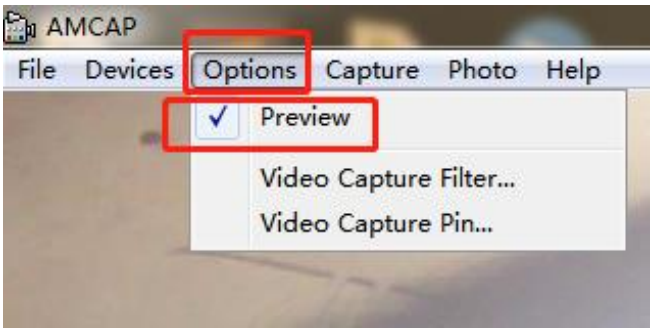
5.2.1. 视频播放功能测试

将设备插入电脑中，设备管理器会出现图像设备



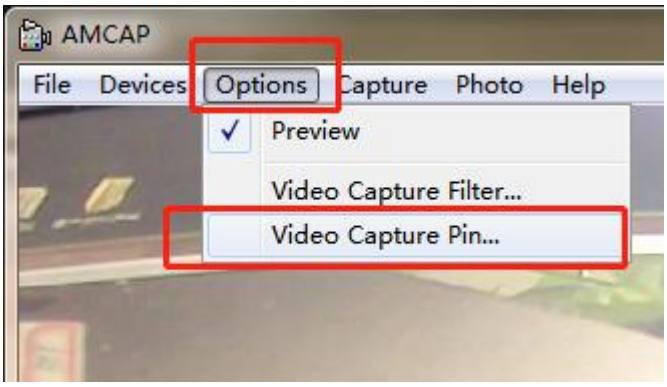
打开开发工具包中的 amcap 软件，打开 devices->HD Web Camera，options->preview。此时就可以正常出图，若无异常则视频播放功能正常





5.2.2. 视频切换分辨率功能测试

点击 option->video Capture pin

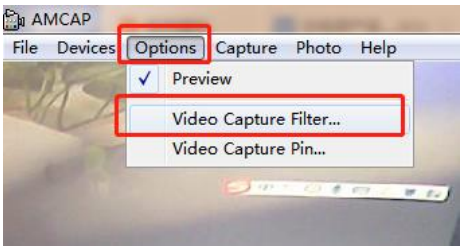


进入属性界面，从上到下依次为帧率，图像格式，分辨率信息，其中图像格式和分辨率可以调节

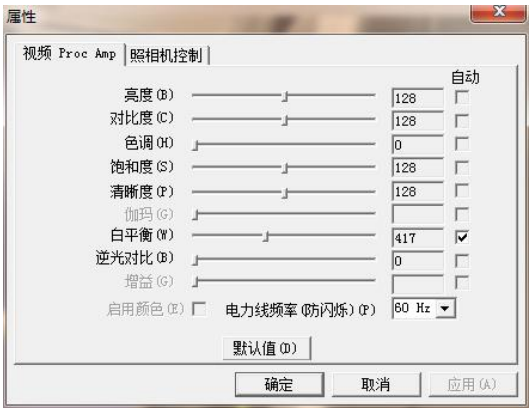
Amcap 底部有基础信息，查看是否与属性调节的一致，若一致则分辨率与格式调节没有问题

5.2.3. 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



5.3. 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

6. 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》