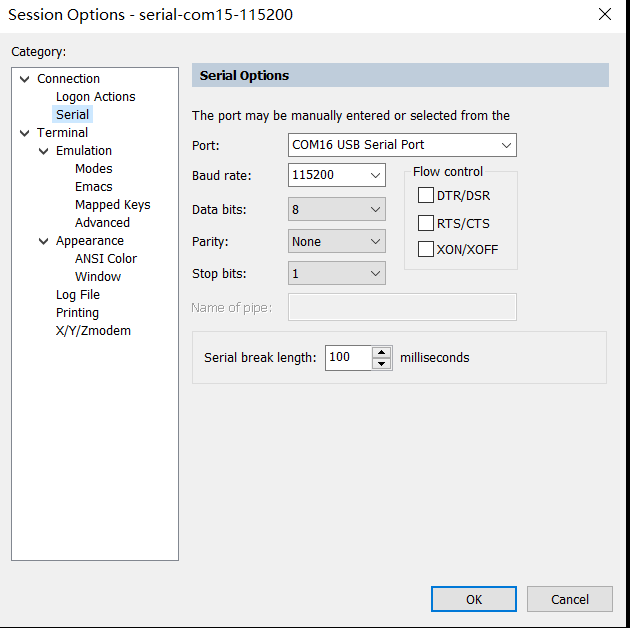
1. 调试方法说明

1.） 串口调试方法：

在板上找到debug uart 的三根针，具体位置可以问下硬件，使用usb2uart 转换器连接电脑和板子。 PC 端我一般使用 Securecrt， 配置如下：



注意：波特率手工选115200， Flow control 的都不要选。

打开串口后， 敲回车就能看到 命令行提示符。

2.） SSH 的使用

Rv1106 板目前不支持

3.) nfs 使用方法：

nfs 在调试时，可以避免频繁拷贝文件到板上，方便板子使用PC机的文件。

在PC端的linux (Ubuntu)执行以下操作：

sudo apt install nfs-kernel-server

sudo vi /etc/exports 添加：

/yourhome/yourpath 192.168.1.0/24(rw,sync)

注：上面一句 根据实际情况修改ip及路径。

sudo exportfs -a

sudo systemctl restart nfs-kernel-server

板端：

mkdir -p /data/nfs

mount -t nfs -o nolock 192.168.1.104://yourhome/yourpath /data/nfs

注： 上面一句 根据实际情况修改ip及路径。

4.） adb shell

使用方法有两种：

1. 从板子连接上usb 线 到电脑， pc端 要安装 adb drvier的工具软件。 PC 端运行 adb shell 命令即可。
2. 使用网络连接adb shell, 这个需要先确定板端的ip 地址，aaa,bbb,ccc.ddd, 然后PC 端输入如下命令：

adb connect aaa.bbb.ccc.ddd

连接成功后，即可使用 adb shell

1. 板子文件分区说明

板子的文件系统在emmc + ram上，主要介绍以下几个：

/dev/root on / type squashfs (ro,relatime)

/dev/ubiblock5\_0 on /oem type squashfs (ro,relatime)

/dev/ubi7\_0 on /userdata type ubifs (rw,relatime,assert=read-only,ubi=7,vol=0)

1.） root 根文件系统， 使用squashfs， 不可修改，压缩的文件系统；存储大部分的库文件，可执行文件等。

3.）/oem 使用squashfs 文件系统，不可修改，存放客户相关的程序，及启动文件（RkLunch.sh）等。 只读。

4.）/usrerdata 用户数据， ubifs 可读写。程序运行时，需要经常修改和保存的数据可以放这里。

1. 编译环境搭建

链接：https://pan.baidu.com/s/1t58FbdC9p0QIDWCh69h25A

提取码：l57a

该工具链是直接从SDK 下拷贝出来的，请按以下步骤进行安装。

安装方法：

1. 使用ubuntu 18 或16的平台 安装。

1. 从网盘下载 env.7z（也可能是env\*.tar.gz） 及env.sh
2. ) env.7z 为交叉编译工具链 + sysroot。 Env.sh 为工具链配置脚本
3. ）env.7z， env.sh放到你的ubuntu 任意工作目录下，先解压env.7z

7z x env.7z

Tar.gz 的工具链使用 tar zxf env\*\*.tar.gz 来解压， 注意 不能使用windows 的解压工具来解压，再拷贝过去。

3. 初始化编译环境变量：

修改env.sh 的第一行和最后一行 的路径，使其符合实际的env（上面解压后的目录） 目录的路径。

source env.sh

5. 验证 编译环境是否oK

$CC -v

......

COLLECT\_GCC=arm-rockchip830-linux-uclibcgnueabihf-gcc

有以上类似信息说明正常。

1. 系统相关函数操作说明：
2. Gpio 操作

头文件 sdkdir/include/gpio.h

// 初始化gpio方向（0 -- 输入， 1 --- 输出）

int comm\_gpio\_set\_direction(int gpioid, int isout);

// 读出 gpio口 电平，

int comm\_gpio\_get\_level(int gpioid);

// 写入 gpio口电平， ishigh ： 1， 高， 0， 低

int comm\_gpio\_set\_level(int gpioid, int ishigh);

注： 上面参数gpioid 的计算方法如下：

GPIO3\_D6 的gpioid = 3 \* 32 + (‘D’ - ‘A’) \* 8 + 6 = 96 + 3 \* 8 +6 = 126

函数返回值： < 0 失败， >=0 成功。

1. Pwm 操作

头文件 sdkdir/include/pwm.h

/\* pwmid --- pwm 的id, 对应不同 pwm, >=2

percent\_of\_duty --- 占空比 0 ~ 100

freq ------------ 表示pwm的 输出频率， 一般可以取 1000 - 1000000 之间的值。

\*/

int pwm\_set(int pwmid, int freq, int percent\_of\_duty);

int pwm\_start(int pwmid);

/\* 可以运行时，调整占空比

\*/

int pwm\_set\_duty(int pwmid, int percent\_of\_duty);

/\* 针对 pwm\_start

\*/

int pwm\_stop(int pwmid);

/\* 针对 pwm\_set。 释放相关资源。

\*/

int pwm\_close(int pwmid);

注意：Pwmid 的取值 可用从>= 2 开始尝试， 0， 1是系统占用，不要使用。 ls /sys/class/pwm 可以看到目前板上 的PWM 情况。

1. Adc 操作

目前只能读取， 最大值1023 --》 1.8V.

访问 Adc 节点：

cat /sys/devices/platform/ff3c0000.saradc/iio:device0/in\_voltage1\_raw

其中红色的1，表示adc1, 如果是adc0， 则将1改为0。

访问方法：

文件系统api ： fopen, fread, open, read....

1. Audio 操作：

说明： 录音部分 放在 app\_set\_default\_codec\_param() 函数参数 设置使能，录音数据会作为rtsp 数据流的一部分传给server。 放音部分见 文件：sdk/include/ao\_c\_if.h。目前支持 g711u, g711a , pcm 格式播放。API 使用sample如下：



1. IPC demo 说明：

目前有以下测试项目（在rv1106\_sdk\_custom/app/comm\_app.c 里定义）

//#define TEST\_DEFAULT\_ENC\_MODIFY

//#define TEST\_ENC\_MODIFY\_IN\_RUNNING

//#define TEST\_OSD

//#define TEST\_SNAPSHOT

可以根据需要打开， 编译后， 运行； 在PC 端可以通过VLC, 查看效果，如下路径：

Stream main

rtsp://xx.xx.xx.xx/stream/av0\_0

Stream sub

rtsp://xx.xx.xx.xx/stream/av0\_1

不过默认不跑这些测试，需要运行的话，还需要设置：

#define PC\_ISP\_SETTING 0

另外 ai/ai.cpp 里也有一些测试项，可以根据需要打开，进行功能验证，如下：

//#define YUV\_TO\_RGB

//#define RGB\_SAVE\_TO\_FILE

//#define JPEG\_OSD\_TEST

//#define TEST\_AI\_RESULT

Demo 编译方法：

Source env.sh

cd build

make

正常会生成rv1106\_ipc， 放到设备上/data/下运行即可。

1. IPC API 说明：

相关API具体使用 方法见 rv1106\_sdk\_custom/app/comm\_app.c 里的代码。

以下函数返回值说明：

0 ----- 表示正常返回。

<0 --- 表示失败。

1. Stream 相关API

目前demo SDK 支持 STEAM\_MAIN, STREAM\_SUB 两个数据流。 可以根据需要， 或者硬件的差异 ，支持更多的 数据流。

// 初始化 某个数据流 ； 在此函数调用之后，可以调用 app\_set\_default\_codec\_param（...），修改如分辨率 ,GOP 等相关参数。

int app\_initCodec(int stream\_index);

//设置 Stream 上 每一帧 编码后的 数据 callback 函数。 一般在CB\_ENC\_DATA cb 里将数据 传给 rtsp server或录像 线程。

//注意： 此函数要在 app\_startCodec（）前面调用。 如果没有调用该函数，而直接调用app\_startCodec（）, 则底层app 不会启动编码过程。

int app\_set\_cb\_enc\_data(int stream\_index, CB\_ENC\_DATA cb);

// 开始某个数据流， 包括启动 VI, 编码， OSD 等。

int app\_startCodec(int stream\_index) ;

// 停止某个数据流，并释放其相关资源。

int app\_deinitCodec(int stream\_index);

1. Stream 相关参数设定api

//此函数在 app\_startCodec(...)前调用才有效。

//pCodec 里的 成员为-1 表示使用默认值，不修改。 目前支持的成员有 resolution， level， frame\_rate，Iframe\_interval， video\_encode\_type。 avcodec\_video\_encode\_t 的定义参考 rv1126\_ipc\_custom/inc/comm\_type.h

int app\_set\_default\_codec\_param(int stream\_index, avcodec\_video\_encode\_t \*pCodec);

// 此函数在 app\_startCodec（...）后调用才有效。pCodec 使用方法同 app\_set\_default\_codec\_param（...）; 该函数设置后，参数会在函数返回前生效。

int app\_set\_codec\_param(int stream\_index,avcodec\_video\_encode\_t \*pCodec);

// 暂没有实现， 后续看是否配合 ircut 来实现?

int app\_set\_Color2Gray(int bColorFlag /\* 0: 黑白 1: 彩色\*/);

1. Stream 上 OSD 的操作 （ 每个 stream 支持最多8个OSD 设置）

// 操作字符显示的

// bg\_palette\_table 的长度 为5, 就是说目前支持 5种 自定义透明色；如果设置为NULL, 则使用默认的。pFontpath 是ttf 字库文件路径。

// 注意：此函数需要在 start codec 之前设置

int app\_set\_osd\_cfg(const char \*pFontpath, unsigned int \* bg\_palette\_table, unsigned char bg\_alpha);

// 此函数设置OSD 的 fontsize, back color ,front color， 需要配合 app\_set\_stream\_osd\_context（）的使用。 设置后立即生效。

int app\_set\_stream\_osd\_config (int stream\_index, avcodec\_osd\_config\_t \*osd\_config);

// 此函数 用于取得 某stream的 某 index 的osd\_config.

int app\_get\_stream\_osd\_config (int stream\_index, avcodec\_osd\_config\_t \*osd\_config);

// 此函数设置某OSD 的 enable/disable, 显示字符串（支持汉字）， 需要配合app\_set\_osd\_cfg（）的使用。 设置后立即生效。

int app\_set\_stream\_osd\_context (int stream\_index, avcodec\_osd\_context\_t \*osd\_context);

// 此函数 用于取得 某stream的 某 index 的osd\_. 配置内容

int app\_get\_stream\_osd\_context (int stream\_index, avcodec\_osd\_context\_t \*osd\_config);

// 此函数在某个stream上 用于画框； cfg->index < 8

int app\_set\_stream\_osd\_rect (int stream\_index, avcodec\_osd\_rect\_t \*cfg);

// 此函数在某个stream上 用于画线； cfg->index < 8

int app\_set\_stream\_osd\_line (int stream\_index, avcodec\_osd\_line\_t \*cfg);

// 此函数在某个stream上 用于画图； cfg->index < 8

int app\_set\_stream\_osd\_graph (int stream\_index, avcodec\_osd\_graph\_t \*cfg);

// 此函数设置某snapshot的OSD 的 enable/disable, 显示字符串（~~支持汉字~~），一般在app\_capture\_pic（）之前调用 。 设置后立即生效。

int app\_set\_capture\_osd\_context (int stream\_index, avcodec\_osd\_context\_t \*osd\_context);

备注：最新版本 有支持 software OSD, 一个视频

1. Stream 上 snapshot 操作

// 从某个Stream 取得快照 数据，jpg 格式，该数据可保存为JPG 文件

int app\_capture\_pic(int stream\_index, unsigned char \*pBuff,int \*iBuffLen);

1. Stream 上 的 视频数据 处理

// 设置 Stream 上 每一帧 的YUV 数据 callback 函数。 cb里不能执行较长延时操作，最好不要操作10ms的处理；

int app\_set\_cb\_yuv\_data(int stream\_index, CB\_YUV\_DATA cb);

// 设置 stream 上 每一帧 的rgb数据回调 函数。 cb里不能执行较长延时操作，最好不要操作10ms的处理；该函数 配合 app\_set\_rgb\_convert\_ratio（），app\_set\_rgb\_crop（）的使用。

// 注意： 如果 想自己在CB\_YUV\_DATA cb里 处理 YUV转换rgb, 则不需要调用下面三个函数；

int app\_set\_cb\_rgb\_data(int stream\_index, CB\_RGB\_DATA cb);

// 设置 CB\_RGB\_DATA cb 里视频数据的 裁剪 尺寸

int app\_set\_rgb\_crop(int stream\_index, CROP\_RECT \* rect);

// 设置 CB\_RGB\_DATA cb 的回调频率，默认是每帧都回调。 divisor 表示 多少帧里回调一次。

int app\_set\_rgb\_convert\_ratio(int stream\_index, int divisor);

1. 取得Stream上 的 相关参数的函数

// 取得某个stream的 视频 分辨率

int app\_get\_resolution(int stream\_index, int \* w, int \*h);

// 取得当前摄像头的camid

int app\_get\_camid(int stream\_index);

1. AI 相关处理函数

// AI 初始化处理函数， 一般会调用app\_set\_ai\_process\_para（），以及AI 识别库的初始化。

int comm\_ai\_init()

// 设置AI 处理的参数， 如 ai 回调函数， 回调频率参数， 回调用户指针（回调时再传给caller）; 最大缓存buffer 数。

int app\_set\_ai\_process\_para(int stream\_index, CB\_AI\_DATA cb, int cb\_ratio, void \* puser, int max\_cached\_packet\_num);

// 开始AI 的数据流及 处理流程; 此函数需要在 comm\_ai\_init （...） 之后调用。

int app\_start\_ai\_process(void);

// 根据 nn\_obj\_p （ai 处理结果） 在 视频上画框。

int app\_set\_nn\_obj(int stream\_index, nn\_object\_array\_t \*nn\_obj\_p);

1. 视频数据转换函数：

目前实现nv12 ->rgb的转换函数，可以参考ai.cpp里的使用。

void \* rga\_proc\_init(int src\_w, int src\_h, PIXEL\_FORMAT\_E s\_format);

bool rga\_proc\_covert\_rgb(void \*hd, VIDEO\_FRAME\_INFO\_S \* frame, CROP\_RECT \*crop, unsigned char \*\* outbuf\_ptr, bool is\_output\_rgb888);

bool rga\_proc\_deinit(void \*hd);

1. IRCUT 相关函数

// ircut 初始化， 设置硬件的接口； type 支持 ADC 及 gpio ( 高电平 关闭ircut, 低电平打开ircut ) 输出 。

int app\_ircut\_init(int stream\_index, int open\_io, int close\_io, int adc\_num, IRCUT\_OUTPUT\_TYPE type);

// 对于 有adc/gpio 输出的灯板 ， 可以定时调用此函数，控制 ircut。

int app\_ircut\_refresh(int stream\_index);

// 手动设置 ircut 开/关。

int app\_ircut\_set\_open(int stream\_index, int isopen);

// 取得 感光 的adc 值

int app\_ircut\_get\_adc(int stream\_index, int \*adc);

1. Soft OSD API 介绍

头文件： ./comm\_inc/codec2\_process.h

注意：soft OSD 与 RK OSD 差异： 1. soft OSD每一个 stream，最多支持32个， 2. soft OSD操作实现是，在每一帧yuv数据上修改的。所以 不建议使用大尺度 的soft osd及 大量使用 soft osd, 可能较多占用CPU资源。 3. soft OSD 不支持 带背景色 的字体； 支持画斜线。

// yuv 上 直接画线函数

int app\_set\_stream\_sw\_osd\_line (int stream\_index, avcodec\_osd\_line\_t \*ocfg);

// 此函数 支持单色的bitmap

int app\_set\_stream\_sw\_osd\_bitmap (int stream\_index, avcodec\_osd\_bitmap\_t \*ocfg);

// yuv 画字符串

int app\_set\_stream\_sw\_osd\_str (int stream\_index, avcodec\_osd\_str\_t \*ocfg);

相关函数使用见 ai/ai.cpp 里的 comm\_ai\_process（）里的测试代码。

1. FAQ
2. 板子启动APP，及如何启动客制化。

板子启动时，默认会运行/oem/usr/bin/rv1106\_ipc 程序， 如果不想其运行，可以kill掉 （命令行： killall rerun.sh ; killall -9 rv1106\_ipc）;

客制化：如果板上有 /userdata/run\_init.sh（注意必须有 x 属性） 文件， 启动时会优先运行它， 不会再跑 默认的rv1106\_ipc.

如果板上没有 /userdata/run\_init.sh， 而有/userdata/rv1106\_ipc ， 启动时会优先运行它，不会再跑 默认的rv1106\_ipc.

1. .如何使用 网络 升级update\_ota.tar 文件（注意： 不能通过网路升级update.img， update.img 只能通过RK PC端工具来升级。）
2. 先把需要更新的固件上传到 IPC的/data/ 或者/mnt/sdcard下面 （注意 /data/ 或/mnt/sdcard/必须先清除一下， 保证 有两倍 升级文件尺寸 以上的剩余空间， 否则可能升级失败， 这个限制在2023年7月份以后image已经没有了， imae 不超过60M 及没有问题。），

网络的话，可以通过nfs等 来上传；或者 adb push 命令 。

update\_ota.tar 拷贝到板上以后， 输入命令： reboot recovery 就开始升级。

注意： 1.）升级文件 update\_ota.tar 的文件名不能修改，否则会升级失败。

2.） 目前网络升级 还不能修改 CMA size; 使用pc 端工具升级 update.img 能修改CMA size.

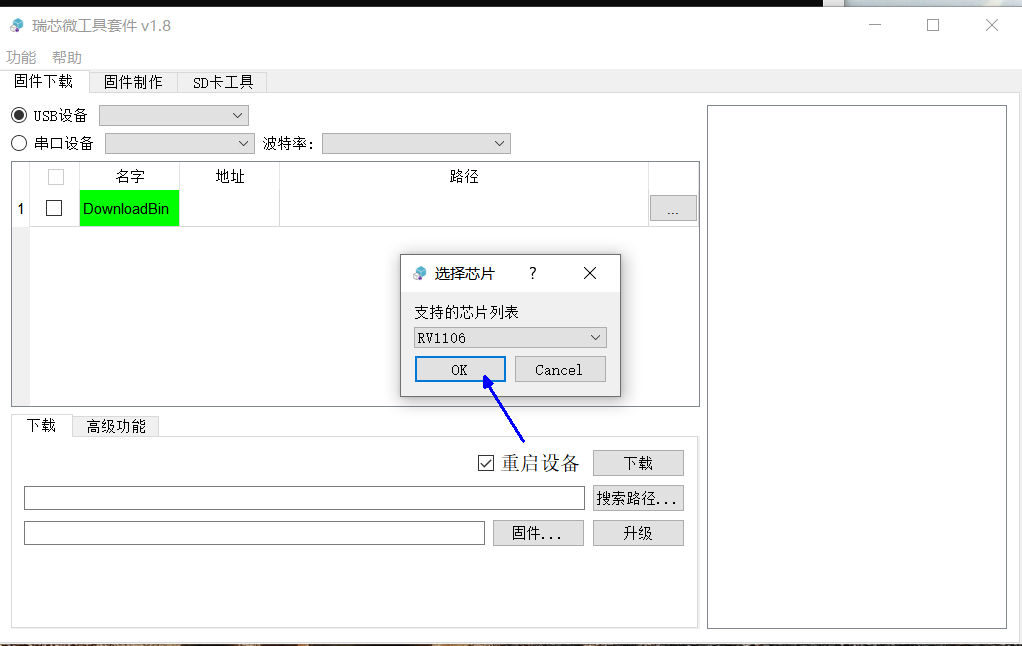
1. 如何使用RK 烧写工具进行升级

1.）进入烧写状态

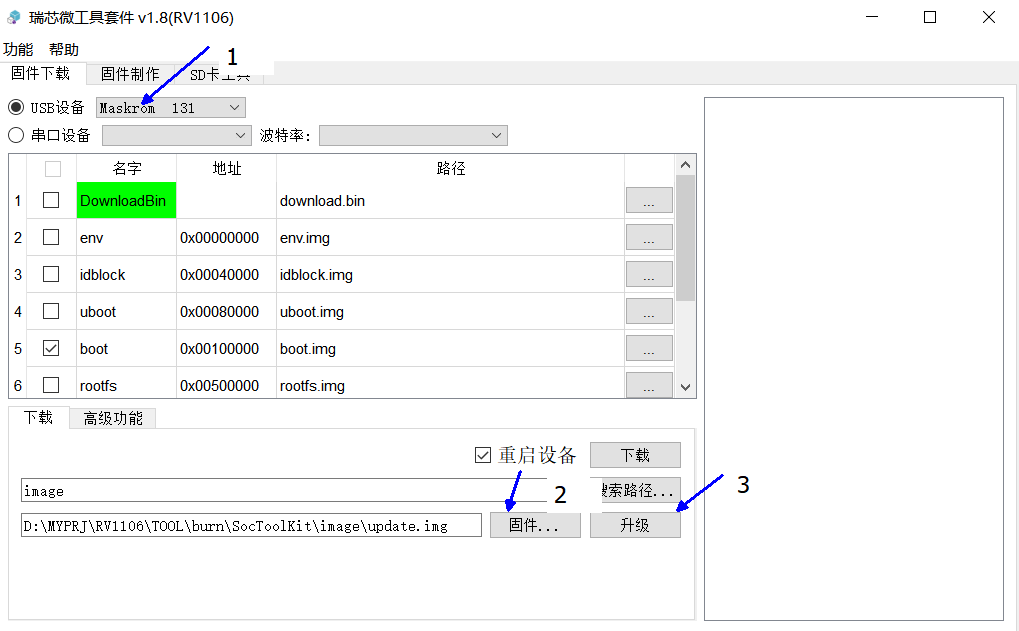
进入串口命令行，或adb shell的命令行，输入：

reboot loader

2.） 运行PC 端烧写工具（需要先安装驱动； 相关安装文件在网盘 资料\burner 下面），如下图

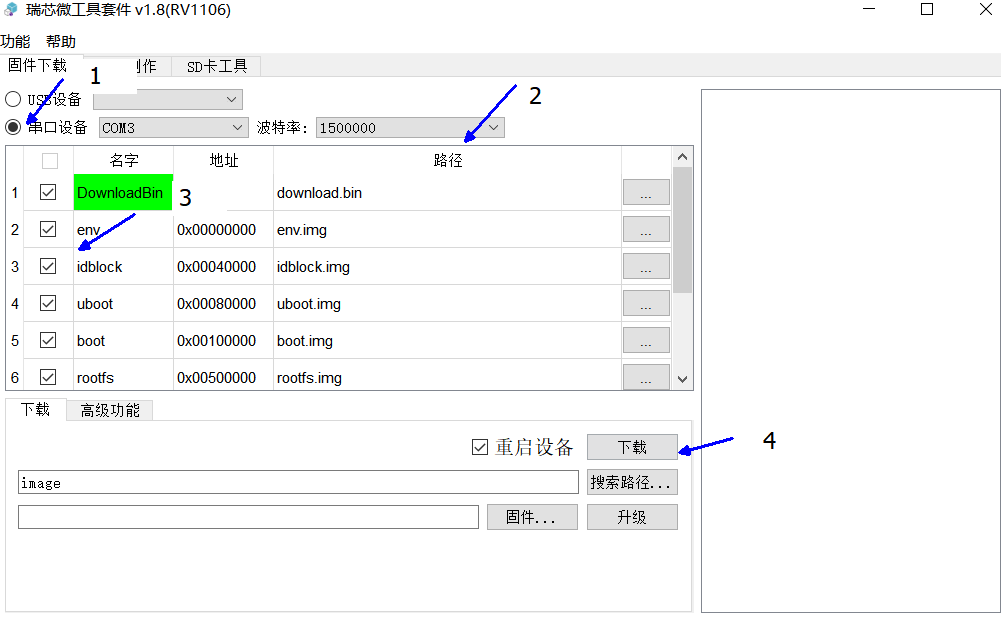


选RV1106 点OK； 然后进入USB 升级界面：



上图，确认下1， 有maskrom 字样， 然后 选 固件， 点击 升级按钮 正常就开始升级了。

有些设备没有外置usb口，可以使用串口升级的，设置界面如下：



按 1（选串口设备），2（选串口波特率为1.5M），3（勾选需要烧写的所有文件）， 4 （点下载按钮） 顺序操作即可。

1. 如何调试CMA 内存的大小

目前1106 的默认image 会预留66M的内存作为CMA 给底层使用； 当然这个值并不适合所有的情况，比如摄像头是200W的，这个太大，如果是500M的，还要500录像 + AI的，可能又太小，所以这里客户可以根据自己实测来调节CMA 的大小。方法大概如下：

1. ）系统里如何查看CMA 当前使用情况？

# cat /proc/meminfo

.....

CmaTotal: 73728 kB

CmaAllocated: 30244 kB

.......

注意： 需要先运行camera app ，这样就能看到实际使用的尺寸。 实际设定的CMA 尺寸应该比CmaAllocated 大若干M。

1. ） 如何 调节系统CMA 大小

RK 支持通过串口UBOOT下动态调节 CMA 的大小，方法如下：

1. ） 接上调试串口，然重启设备后按住Ctrl+C，直到出现 => <INTERRUPT> 字段，表⽰已进⼊U-Boot终端。
2. ) 设置CMA 方法如下



设置命令如下图示的两条。



注意，设置cmd时， 只能修改“ rk\_dma\_heap\_cma=” 后面的数据，其它参数printenv 输出的原样设置即可，不要修改。

客户在调试阶段，请使用以上方法修改CMA的大小； 后期批量时，可以通知我司来设置CMA的 值 到image里， 方便生产。