# hcRTOS USB摄像头使用说明

## 1. 概述

hcRTOS USB摄像头支持，是基于libuvc进行开发的，下面会列举出对应的examples来展示如何使用，以及部分API的说明

> 目前hcRTOS 支持的usb 摄像头分辨率情况如下:
>
> - MJPG:
>   - 1080P 30Hz （部分图片压缩率不足的可能没法支持）
>   - 720P 30HZ 或者以下
> - YUV:
>   - 320x240 30HZ 或者以下

## 环境搭建以及examples展示

### menuconfig配置

> - **必须选择**
>   - BR2_PACKAGE_LIBUSB
>   - BR2_PACKAGE_LIBUVC
> - **可选**
>   - BR2_PACKAGE_LIBUVC_EXAMPLES　　## 用于使能 examples
>   - LIBUVC_NUM_TRANSFER_BUFS　　## 用于配置缓冲buffer的数量，每个buffer的大小为一帧图片的大小

完成menuconfig配置之后， 编译命令如下

```
make libuvc-rebuild all

make libuvc-rebuild cmds-rebuild all   ### 如果使能了 BR2_PACKAGE_LIBUVC_EXAMPLES
```

### H3 examples 展示

如果上面的menuconfig中使能了 `BR2_PACKAGE_LIBUVC_EXAMPLES` 的话, 在烧录好程序后, 可以通过串口敲以下命令来展示hcRTOS 如何使用usb 摄像头

- **命令** `uvc_demo`
  - 这个example **需要用户先将usb摄像头和U盘插到开发版上**, 输入命令后 examples会运行10秒, 会将所抓取的图像写到用于指定的路径
  - 命令格式 `uvc_demo /media/sda1` , 需要指定图片保存的路径, 如这里指定的路径是 /media/sda1
- **命令** `uvc_demo2`
  - 这个example **需要用户先将usb摄像头插到开发版上**, 输入命令后 examples会运行10秒, 这个期间会将图像显示到屏幕上
  - 命令格式 `uvc_demo2`
- **命令** `uvc_demo3`
  - 这个example 是基于 `uvc_demo2` 上修改, **增加支持热插拔功能**. 输入命令后, 任何时间插入USB摄像头都能在屏幕上显示，反之只要拔出屏幕就恢复原本的UI显示
  - 命令格式 `uvc_demo3`

## H2 部分API的说明

具体API的使用, 推荐参考example的源码 (**components\cmds\source\usb\usb_libuvc_demo3.c**)

components\cmds\source\usb\libuvc_examples\usb_libuvc_demo3.c

### H3 需要包含的头文件

```
1  #include <libusb.h>
2  #include <libuvc/libuvc.h>
```

## H3 常用API

```c
// 初始化 libuvc
uvc_error_t uvc_init(uvc_context_t **ctx, struct libusb_context
*usb_ctx);

// 注销 libuvc
void uvc_exit(uvc_context_t *ctx);

// 获取 USB摄像头的对象(uvc_device_t)
uvc_error_t uvc_find_device(
    uvc_context_t *ctx,
    uvc_device_t **dev,
    int vid, int pid, const char *sn);

// 获取指定摄像头源自哪个USB端口
uint8_t uvc_get_bus_number(uvc_device_t *dev);

// 打开指定摄像头，获得摄像头句柄(uvc_device_handle_t)
uvc_error_t uvc_open(
    uvc_device_t *dev,
    uvc_device_handle_t **devh);

// 关闭指定摄像头
void uvc_close(uvc_device_handle_t *devh);

// 通过打印展示当前摄像图支持什么格式和分辨率的输出
void uvc_print_diag(uvc_device_handle_t *devh, FILE *stream);

// 按照所指定的分辨率/帧数/图片格式对摄像头进行配置
uvc_error_t uvc_get_stream_ctrl_format_size(
    uvc_device_handle_t *devh,
    uvc_stream_ctrl_t *ctrl,
    enum uvc_frame_format format,
    int width, int height,
    int fps
    );

// 开始摄像，之后每帧图像都会调用到所指定的回调函数(uvc_frame_callback_t *cb)
uvc_error_t uvc_start_streaming(
    uvc_device_handle_t *devh,
    uvc_stream_ctrl_t *ctrl,
    uvc_frame_callback_t *cb,
    void *user_ptr,
    uint8_t flags);

// 停止摄像
void uvc_stop_streaming(uvc_device_handle_t *devh);

```