Rockchip RV1126 UVC delay optimization

File identification: RK-KF-YF-540

Release version: V1.0.0

Date: 2021-08-04

Document Confidentiality: □ Top Secret □ Secret □ Internal Information ■ Public

Disclaimer

This document is provided "as is". Rockchip Microelectronics Co., Ltd. ("the company", the same below) does not make any statements, information and content of this document The accuracy, reliability, completeness, marketability, specific purpose and non-infringement of the company provide any express or implied statement or guarantee. This article The file is only used as a reference for instructions.

Due to product version upgrades or other reasons, this document may be updated or modified from time to time without any notice.

Trademark statement

"Rockchip", "Rockchip" and "Rockchip" are all registered trademarks of our company and are owned by our company.

All other registered trademarks or trademarks that may be mentioned in this document are owned by their respective owners.

Rockchip Microelectronics Co., Ltd.

Rockchip Electronics Co., Ltd.

Address: No. 18, Area A, Software Park, Tongpan Road, Fuzhou City, Fujian Province

URL: www.rock-chips.com

Customer Service Hotline: + 86-4007-700-590

Customer Service Fax: + 86-591-83951833

Customer Service Email: fae@rock-chips.com

Preface

Overview

This document aims to guide engineers how to use the UVC delay optimization function of Rockchip RV1126 Linux platform.

product version

Chip name	Kernel version
RV1126 / 1109	Linux 4.19

Audience

This document (this guide) is mainly applicable to the following engineers:

Technical Support Engineer

Software Development Engineer

Revision record

version number	author	Modified date	Modify the description
V1.0.0	HuangJC 、 CW 、 LXH 、 TZB 、 LQH	2021-08-04	initial version

content

Introduction

Rockchip RV1126 Linux platform supports UVC full-featured preview, which is common to smart screen cameras, live cameras, conference cameras, etc.
For UVC products, the delay index is a key focus item. This document can guide customers on how to optimize the delay index according to their product configuration.
Help improve product user experience and increase competitiveness.

UVC delayed decomposition

Breakdown block diagram

UVC preview delay is mainly divided into device end processing: sensor outflow, isp processing, ispp processing, encoding, usb processing and sending stream, and host end
Processing: usb receiving stream, application fetching stream decoding processing, display processing.

The RV1126 UVC product is mainly used as the device side, so the host side processing is not in the optimization scope of this document.

The device end delay is broken down as follows (take SC500AI as an example):

in:

1. Sensor exposure time: different sensors and different light environments are different, generally 10-33ms, the control range can be configured through IQ,
   It must be weighed against the requirements of image quality;

2. Raw transmission: The general sensor configuration is to connect to cif to enter, and vicap to fetch the stream. This is time-consuming mipi transmission, which is also the same as the sensor itsel
   Off, generally 16-33ms;
3. ISP processing: ISP processing is time-consuming, and different sensors and IQ configurations are different;
4. ispp processing: ispp processing time-consuming, mainly nr, tnr, fec modules are time-consuming, generally modules do not need to open fec;
5. rga processing and scheduling: the application layer may customize the rga preprocessing image data operation caused by time-consuming, such as cropping, zooming, etc.;
6. uvc encoding processing: uvc calls the encoder according to the format required by the Host. The encoding processing takes time, and the encoding time of different formats is different, the same
   The mjpeg format takes a long time at the resolution;
7. Usb processing and transmission: The usb driver is time-consuming for frame encoding and transmission to the host.

Take the SC500AI sensor (1440p) connected product as an example, the time-consuming data for previewing 1440p @ 30fps mjpeg format image is as follows:
Below, you can see that it is relatively long, and the comparison with the public version needs to be optimized to at least 100ms.

Total time on Device: 10 + 16 + 8 + 8 + 7 + 18 + 33 + 27 + 22 = 142ms

Delay printing method

This chapter introduces the delay statistics method of each module of the RV1126 platform, which needs to be printed in the preview state.

1. Sensor exposure time printing method:

Add a line to the machine RkLunch.sh script to enable AIQ's AE printing:

```
export persist_camera_engine_log = 0x1ff3
```

When uvc is previewed, the serial port will output the following AE print:

```
[00: 18: 32.026984 [AEC Car Rental: XCAM DEBUG rk_aiq_ae_algo.cpp: 7200: >>> Framenum = 156
Why Piris = 512 ;
Sgain = 1 .000000 , Stime = 0 .002196 , mgain = 1 .687500 , mtime = 0 .029992 , lgain = 0 .000000 , l
time = 0 .000000
```

You can see that the exposure time of 2 frames (HDR) printed under the EVB public version (os04a sensor) configuration above are: 2.2ms and 30ms (Stime = 0.002196, mtime = 0.029992), non-HDR only has one time printed with a value indicating the exposure time.

2. Mipi transmission (cif) time-consuming printing method:

You can print csi transport delay through cat / proc / uvcinfo and it takes 31ms (EVB public version), as follows:

```
[root @ RV1126_RV1109: /] # cat proc / uvcinfo
vc on: 1 1 0 0
csi sequence: 2263 2263 0 0
csi transport delay: 30224003 31716170 0 0  #mipi It takes time to transmit, here there are 2 frames when hdr is turned on
stream sequence: 2264 2264 0 0
stream sequence map: 2263 2263 0 0
stream vb2 done : 25604586 25601086 0 0
usb transport bytes: 44650
usb transport delay: 2398084  #usb transport delay print 2.3ms after transmission
```

After the cif optimization patch is applied later, there are also cif nodes that can directly view the time-consuming.

3. Time-consuming printing method of isp / ispp:

```
[root @ RV1126_RV1109: /] # cat proc / rkispp *
rkispp0 version: v01.06.00
clk_ispp   491519999
aclk_ispp   491519999
hclk_ispp   245760000
Interrupt Cnt: 127584 ErrCnt: 0
```

Page 4

```
Input rkisp0 Format: FBC420 Size: 2688x1520 (frame: 42527 rate: 33ms
delay: 6ms)   # delay before optimization: 15ms
Output rkispp_scale0 Format: NV12 Size: 1920x1080 (frame: 42527 rate: 33ms
delay: 18ms) # Delaybefore optimization: 37ms
TNR ON (0xf00000f) (mode: 3to1) (global gain: disable) (frame: 42527
time: 8ms idle) CNT: 0x0 STATE: 0x1e000000
NR ON (0x57) (external gain: enable) (frame: 42527 time: 6ms idle)
0x5f0: 0x0 0x5f4: 0x0
SHARP ON (0x19) (YNR input filter: ON) (local ratio: OFF) 0x630: 0x0
FEC OFF (0x2) (frame: 0 time: 0ms idle) 0xc90: 0x0
ORB OFF (0x0)
Monitor ON Cnt: 0
```

The time consumption of isp can be viewed in the delay print of the Input line, and the total time consumption of isp + ispp can be viewed in Output. The delay of the line is printed.

4. Coding and uvc scheduling transmission time-consuming printing method:

```
[root @ RV1126_RV1109: /] # touch / tmp / uvc_use_time && sleep 1 && rm
/ tmp / uvc_use_time

mpp [667]: [shm_control_uvc] [sendUVCBuffer]: isp-> aiserver seq: 1883 latency
time: 20302 us, 20 ms
[uvc_ipc] [recvUVCBuffer]: isp-> aiserver-> ipc seq: 1883 latency time: 20968 us,
20 ms
[uvc_app] [test_mpp_run]: isp-> aiserver-> ipc-> mpp_get_buf seq: 1883 latency
time: 21380 us, 21 ms
[uvc_app] [test_mpp_run]: mpp_enc + getbuf time: 12 ms, try_count: 15
[uvc_app] [test_mpp_run]: isp-> aiserver-> ipc-> mpp_enc_ok latency time: 32054
us, 32 ms
[uvc_app] [uvc_delay_time_calcu_after_get]: isp-> mpp-> usb_ready seq: 1883
latency time: 33061 us, 33 ms
[uvc_app] [uvc_delay_time_calcu_before_get]: isp-> mpp-> usb_ready-> usb_send_ok
seq: 1883 latency time: 35726 us, 35 ms
```

The buf from ispp has time stamp information, and the time-consuming of subsequent stages can be counted through the printing of the application layer, as in the above EVB (with optimization Time-consuming printing:

Encoding time: mpp_enc + getbuf time: 12 ms

uvc transmission time: 35-32 = 3 ms

UVC delay optimization

Taking SC500AI as an example, the optimization scheme is as shown in the figure below. The parallel preemption scheme is mainly used, and the total delay on the device side after optimization:

10ms (exposure) + 11ms (transmission) + 57ms (vicap dqbuf-> usb qbuf # 4) + 7ms (usb # 4) = about 85ms

The following introduces specific optimization schemes for each time-consuming module on the device side for reference and use of other ··nsors and software configuration products.

Sensor exposure time optimization

In the above example, the sensor exposure time itself is already very short, and there is no room for optimization. If other sensor configurations can be used through the above method, Print to confirm the current exposure time.

Ways to reduce exposure time:

1. Search LinearAE and TimeDot under iq file to shorten the maximum exposure time;
2. For non-HDR, you can generally change the TimeDot in the DAY project, and the configuration MTimeDot for HDR should generally be changed. Observe through AE printing Whether it takes effect.

cif / isp / ispp time-consuming optimization

The combination part is related to the sensor access mode configured in the product dts, which is mainly divided into isp_in access and cif access:

isp pass-through optimization solution

To use the isp_in access method, you can use the isp pass-through method to optimize the delay. The specific method is:

Compile the kernel:

git revert 000c394b31f9a95de9ac17e4335b892876bd2e60 compile the kernel and burn the kernel directory separately
Zboot.img generated under.

The above-mentioned default changes the corresponding product of ai camera dts to isp in access mode. If customers use their own dts, please refer to modify it to isp in access
Method configuration can be.

Modify the device file directly:

adb shell; vi /etc/aicamera.sh; add export normal_no_read_back = 1 before starting ispserver

Page 6

Acceptance method: Through cat / proc / rkispp * in the preview, confirm that the delay in the Input line is printed as 0, indicating that the modification is successful.

Profit: Depending on the specific sensor product situation, generally the profit is more than 8ms.

Limitations: DeHaze / Enhance mode cannot be enabled in the pass-through mode. If the original image quality of the product is fogged, it will be affected. The customer needs
Weigh the loss of picture quality by yourself.

line configuration parallel optimization scheme

This part of the optimization scheme is mainly based on the characteristics of line-by-line processing. When processing a frame of data, the previous-level module sends the data in advance, and the ne
The processing of the blocks achieves the parallel effect and the optimization is time-consuming. Therefore, this solution requires specific product measurements to optimize the line settings, usually 1
The above benefits.

To facilitate time-consuming statistics, you can modify the timestamp in the following way, starting from cif, and you can directly see the total cost of the three stages through the node
Time:

```
diff --git a / drivers / media / platform / rockchip / isp / csi.c
b / drivers / media / platform / rockchip / isp / csi.c
index 631c1de..56ef0bf 100644
--- a / drivers / media / platform / rockchip / isp / csi.c
+++ b / drivers / media / platform / rockchip / isp / csi.c
@@ -645.7 +645.7 @@ static void rkisp_dev_trigger_handle (struct rkisp_device
* dev, u32 cmd)
                isp-> dmarx_dev.pre_frame = isp-> dmarx_dev.cur_frame;
                isp-> dmarx_dev.cur_frame.id = t.frame_id;
                isp-> dmarx_dev.cur_frame.sof_timestamp = t.sof_timestamp;
- isp-> dmarx_dev.cur_frame.timestamp = t.frame_timestamp;
+ isp-> dmarx_dev.cur_frame.timestamp =
t.sof_timestamp; // t.frame_timestamp;
                mode = t.mode;
                times = t.times;
                hw-> cur_dev_id = id;
```

cif optimization

Cif is mainly time-consuming for mipi to transmit buf. Different sensors are not the same. After the cif driver is optimized and submitted, it can be directly previewed.
When printing the rkcif node to see the specific time-consuming:

```
commit ec0640c0189029f10bc5f9e239f28ec9a0b206d2
Author: Zefa Chen <zefa.chen@rock-chips.com>
Date: Mon Jul 5 17:39:39 2021 +0800

    media: rockchip: cif: mipi wakes up buf by line int

    Signed-off-by: Zefa Chen <zefa.chen@rock-chips.com>
    Change-Id: If10afeec22ce89a52f7c0e0e454005ca3c3cdc5e
    (cherry picked from commit 500585fdc15b6338a6157f967f29f6d01c98097e)


[root @ RV1126_RV1109: /] # cat proc / rkcif *
Driver Version: v00.01.0a
Work mode: ping pong
Monitor mode: idle
aclk_cif: 491519999
hclk_cif: 245760000
dclk_cif: 297000000
Input Info:
        src subdev: m01_f_sc500ai 1 -0030
```

Page 7

```
        interface: mipi csi2
        lanes: 4
```

```
        vc channel: 0
        hdr mode: normal
        format: SBGGR10_1X10 / 2560x1440 @ 30
        crop.bounds: (0, 0 ) / 2560x1440
    Output Info:
        format: BG10 / 2560x1440 (0.0)
        compact: enable
        frame amount: 296
        early: 0 ms
        readout: 16 ms
        rate: 33 ms
        fps: 30
        irq statistics:
                total: 296
                csi over flow: 0
                csi bandwidth varnish: 0
                all err count: 0
                frame dma end: 296
```

You can see the time-consuming print item: readout: 16 ms. The mipi transmission time is related to the sensor, which is generally fixed and cannot be optimized temporarily.
The cif-driven optimization scheme is: send data to the next-level module (isp) in advance by row, and after applying the above cif optimization patch, configure it through the boot script
Set the line value to check whether the screen is normal after the actual test preview.

The test method can be to shake your hand at the bottom of the screen to observe whether there is image tearing between the bottom screen lines, if there is tearing (post-processing
Time-consuming is close to or less than the lead time), you need to increase the line value, the line configuration command is as follows:

```
diff --git a / oem / oem_uvcc / RkLunch.sh b / oem / oem_uvcc / RkLunch.sh
index 473c3a1..6c746d8 100755
--- a / oem / oem_uvcc / RkLunch.sh
+++ b / oem / oem_uvcc / RkLunch.sh
@@ -48.6 +48.15 @@ echo "camera_max_height = $ {camera_max_height}"
  export CAMERA_MAX_WIDTH = $ {camera_max_width}
  export CAMERA_MAX_HEIGHT = $ {camera_max_height}

+ # line config
+ cif_line = $ (($ {camera_max_height} / 4 * 3))
+ echo $ cif_line> / sys / devices / platform / rkcif_mipi_lvds / line_int_num
+
  #rockit log level ctrl: 1: fatal error; 2: error; 3: warning; 4: infomational;
5: debug level; 6: verbose
  export rt_log_level = 3
```

isp / ispp optimization

isp / ispp is the focus of time-consuming optimization. The more modules you open, the longer it takes. At present, the driver has defaulted to include a time stamp in each frame.
Information, you can directly view the specific time-consuming (delay printing) by printing proc / rkispp * and other nodes during preview:

```
[root @ RV1126_RV1109: /] # cat proc / rkispp *
rkispp0 version: v01.06.00
clk_ispp   350000000
aclk_ispp   491519999
hclk_ispp   245760000
Interrupt Cnt: 22358 ErrCnt: 0
```

Page 8

```
Input rkisp0 Format: FBC420 Size: 2560x1440 (frame: 5641 rate: 32ms delay: 10ms)
Output rkispp_m_bypass Format: NV12 Size: 2560x1440 (frame: 5640 rate: 32ms
delay: 46ms)
TNR ON (0xf00000f) (mode: 3to1) (global gain: disable) (frame: 5641 time: 9ms
idle) CNT: 0x0 STATE: 0x1e000000
NR ON (0x17) (external gain: enable) (frame: 5641 time: 6ms working)
0x5f0: 0x7002c 0x5f4: 0x3e0b
SHARP ON (0x1b) (YNR input filter: ON) (local ratio: ON) 0x630: 0x72b
FEC ON (0x80000021) (frame: 5640 time: 19ms idle) 0xc90: 0x0
ORB OFF (0x0)
Monitor ON Cnt: 0
```

In addition, to view the status of isp and ispp, you can directly view the /proc/rkisp* node, which will display which image processing internal module functions are opened.
Since the image effect directly affects this time-consuming (noise, hdr, etc.), it is recommended to complete the image turning work before the delay optimization.
Optimize the image after it is stable.

Optimization 1: isp ispp boost frequency, optimize 6ms +

```
diff --git a / drivers / media / platform / rockchip / isp / hw.c
b / drivers / media / platform / rockchip / isp / hw.c
index d6d3ca3..7b1a01e 100644
--- a / drivers / media / platform / rockchip / isp / hw.c
+++ b / drivers / media / platform / rockchip / isp / hw.c
@@ -353.13 +353.13 @@ static const struct isp_clk_info rv1126_isp_clk_rate [] = {
                .clk_rate = 20,
                .refer_data = 0,
        }, {
- .clk_rate = 300,
+ .clk_rate = 600,
                .refer_data = 1920, // width
        }, {
- .clk_rate = 400,
+ .clk_rate = 600,
                .refer_data = 2688,
        }, {
- .clk_rate = 500,
+ .clk_rate = 600,
                .refer_data = 3072,
        }, {
                .clk_rate = 600,
diff --git a / drivers / media / platform / rockchip / ispp / hw.c
b / drivers / media / platform / rockchip / ispp / hw.c
index 95ca8f1..e0ac476e 100644
--- a / drivers / media / platform / rockchip / ispp / hw.c
+++ b / drivers / media / platform / rockchip / ispp / hw.c
@@ -181.13 +181.13 @@ static const struct ispp_clk_info rv1126_ispp_clk_rate [] =
{
                .clk_rate = 150,
                .refer_data = 0,
        }, {
- .clk_rate = 250,
+ .clk_rate = 500,
                .refer_data = 1920 // width
        }, {
- .clk_rate = 350,
+ .clk_rate = 500,
                .refer_data = 2688,
```

Page 9

```
        }, {
- .clk_rate = 400,
+ .clk_rate = 500,
                .refer_data = 3072,
        }, {
                .clk_rate = 500,
```

Optimization 2: Add line configuration to isp, and send data to tnr module in advance for parallel

Apply the following to submit the patch, the default line configuration is closed, you need to configure the node in the script such as CIF, different sensor configurations are different, in principle The advance time of line configuration needs to be less than the processing time of tnr in ispp, and the actual measurement shall prevail.

```
commit 35dd5b74d555d8b8f874549fe6ab76aadbbf5800
Author: Cai YiWei <yw@rock-chips.com>
Date: Thu Jul 8 18:16:32 2021 +0800

    media: rockchip: isp: frame buffer done early

    config wait-line to isp virtual device dts node,
    or echo value to debug node before open isp video.
    / sys / module / video_rkisp / parameters / wait_line

    Change-Id: I5c73c90117455663620b4c025e78aa6233ca40b9
    Signed-off-by: Cai YiWei <yw@rock-chips.com>
```

Note: This line configuration needs more margin. If the advance time is longer than tnr, it may be torn or jammed.

Optimization 3: ispp adds line configuration, and sends data to the subsequent modules in parallel

Apply the following to submit the patch, the default line configuration is closed, you need to configure the node in the script such as cif / isp, different sensor configurations are different, in principle The advance time of the line configuration needs to be less than the post-processing time, and the actual measurement shall prevail. If the time is greater than the post-processing time, image tearing wi

```
commit 4549cfa4134c1ee9d0ee2423a0e555486e5d5793
Author: Cai YiWei <yw@rock-chips.com>
Date: Wed Jul 7 09:27:45 2021 +0800

    media: rockchip: ispp: frame buffer done early

    config wait-line to ispp virtual device dts node,
    or ispp debug node before open ispp video.
    / sys / module / video_rkispp / parameters / wait_line

    for example: output is 2688x1520, config
    wait-line to 768 (128 align), vb2 buffer
    will done when poll image processing greater
    than 768, wait-line less than (height - 128) is valid.

    Change-Id: I4a448cc6baffbb5794eef91965e4b2bc349aa5ed
    Signed-off-by: Cai YiWei <yw@rock-chips.com>
```

Optimization 4: Ispp tnr 3to1 changed to 2to1

Through the information printed by the rkispp0 node, you can confirm whether the TNR module has turned on the 3to1 mode, which will delay one more frame in the hardware processing To synthesize the image, turning it off will increase the delay gain by one frame (33ms), which needs to be confirmed. The closing method is search in the xml file of IQ 3to1 can be changed to 0 configuration, and check whether it is already 2to1 through the information printed by the rkispp0 node during preview.

Refer to the line configuration patch:

Page 10

The following is a reference line configuration patch, you can first measure and tune based on this configuration:

```
commit f855a3fb926555c3fed8b4d2b1a2323761455400
Author: Mark Huang <Mark Huang@rock-chips.com>
Date: Thu Jul 29 09:36:38 2021 +0800

    oem_uvcc: add isp / ispp / cif line config

    Signed-off-by: Mark Huang <Mark Huang@rock-chips.com>

diff --git a / oem / oem_uvcc / RkLunch.sh b / oem / oem_uvcc / RkLunch.sh
index 473c3a1..6c746d8 100755
--- a / oem / oem_uvcc / RkLunch.sh
+++ b / oem / oem_uvcc / RkLunch.sh
@@ -48.6 +48.15 @@ echo "camera_max_height = $ {camera_max_height}"
  export CAMERA_MAX_WIDTH = $ {camera_max_width}
  export CAMERA_MAX_HEIGHT = $ {camera_max_height}

+ # line config
+ isp_line = $ (($ {camera_max_height} / 2))
+ ispp_line = $ (($ {camera_max_height} / 4 * 3)) #no fec
+ cif_line = $ (($ {camera_max_height} / 4 * 3))
+ echo "isp_line = $ isp_line, ispp_line = $ isp_line, cif_line = $ cif_line"
+ echo $ isp_line> / sys / module / video_rkisp / parameters / wait_line
+ echo $ ispp_line> / sys / module / video_rkispp / parameters / wait_line
+ echo $ cif_line> / sys / devices / platform / rkcif_mipi_lvds / line_int_num
+
  #rockit log level ctrl: 1: fatal error; 2: error; 3: warning; 4: infomational;
5: debug level; 6: verbose
  export rt_log_level = 3
```

Partial optimization of isp / ispp can also be combined with the pass-through optimization method, which is subject to the actual test results f the customer's product and the picture quality trade-off.

rga processing and scheduling time-consuming optimization

This part is related to the application processing of the customer's product. The default SDK delay is not time-consuming in this part. In the example, the optimization processing of this part is combine The crop and zoom processing part of EPTZ and rkrga plug-ins is optimized to 5ms.

The relevant submissions in the SDK are as follows:

```
commit cb5eb4e8a2a57ea2229406675baa6900f659ba36
Author: lqh <kevin.lin@rock-chips.com>
Date: Wed Jul 7 17:56:12 2021 +0800

    [aiserver / uvc_graph] Optimize UVC process.
```

```
    Using rkzoom node to operate ptz and small resolution display.
    Delete link that is no longer in use, now support 3 main link,
    under eptz mode, using eptz link, under nomoral display mode that
    resolution> 480P, using uvc mode, other sinces using uvc_zoom mode
    which support ptz feture, and 480P / 240P 16: 9 output.

    Signed-off-by: lqh <kevin.lin@rock-chips.com>
    Change-Id: Icf8fc50967f0be2de11e1fe6108bd5251fd92039

commit 9b7097526d0eebc7fdd4a779472338f994b70feb
```



```
Author: lqh <kevin.lin@rock-chips.com>
Date: Tue Jul 6 14:10:04 2021 +0800

    [aiserver / vendor / zoom] add zoom operation.

    Using RTNodeVFilterZoom.cpp to instead operation in librockit.
    Combines eptz and ptz operation, reduce 1 time rga operation.

    Signed-off-by: lqh <kevin.lin@rock-chips.com>
    Change-Id: I3185f850b6b4eacff6807d04f690a161ea9c5d81
```

Time-consuming optimization of mpp encoding

This part of the time consumption is positively correlated with the encoding format and resolution. The encoding time is: mjpeg> h264≈h265.

Take 1440p resolution as an example, the time-consuming situation of each format encoding part:

Encoding format	Time-consuming (ms)
MJPEG	25th
H264	16
H265	16

In the example, a separate framing coding optimization scheme is used for the 1440p mjpeg scene, and it needs to be configured according to the specific resolution format.
To support the mpp library alone, it is not universal, only the general optimization scheme is introduced: vpu boost frequency.

```
diff --git a / arch / arm / boot / dts / rv1126.dtsi b / arch / arm / boot / dts / rv1126.dtsi
index 4b0024c..6e7c2b2 100644
--- a / arch / arm / boot / dts / rv1126.dtsi
+++ b / arch / arm / boot / dts / rv1126.dtsi
@@ -2145.7 +2145.7 @@
                interrupts = <GIC_SPI 74 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <& cru ACLK_JPEG>, <& cru HCLK_JPEG>;
                clock-names = "aclk_vcodec", "hclk_vcodec";
- rockchip, normal-rates = <400000000>, <0>;
+ rockchip, normal-rates = <500000000>, <0>;
                rockchip, advanced-rates = <500000000>, <0>;
                rockchip, default-max-load = <2088960>;
                resets = <& cru SRST_JPEG_A>, <& cru SRST_JPEG_H>;
```

Note: For non-reference sdk public version hardware design, you need to pay attention to whether the voltage of vpu is not enough. If it is not enough, some pictures will appear on the screen. elephant.

Usb processing and scheduling time-consuming optimization

In the example, a separate framing transmission optimization scheme is used for the 1440p mjpeg scene with encoding synchronization, and it needs to be based on the specific resolutions. Format configuration, need to be separately coded patch package support, not universal, only introduce the general optimization scheme: optimize the transmission process introduced by cpu idle The scheduling and waiting time is time-consuming, and the optimization is about 8ms.

```
kernel $ git show c1b9c11178c08d3470558e0a815184342594622b
commit c1b9c11178c08d3470558e0a815184342594622b
Author: William Wu <william.wu@rock-chips.com>
Date: Fri May 7 11:49:41 2021 +0800
```

```
    usb: gadget: f_uvc: disallow the CPU to enter deeper sleep when streamon

    I test on RK3399 IND EVB running Android R system, and
    set the USB gadget as UVC function via configfs, both
    the USB 2.0 and 3.0 UVC have preview abnormal problems.
    If the CPU enter deeper sleep, the USB DMA transfer gets
    corrupted, and the UVC ISOC transmission lost data.

    This patch puts in a "pm_qos_latency" requirement which
    will keep the CPU out of the deeper sleep states when
    UVC stream on. And allow the CPU to enter deeper sleep
    state after UVC stream off.

    Change-Id: I808290f124c6a32da3888819348093a205bfad61
    Signed-off-by: William Wu <william.wu@rock-chips.com>

diff --git a / drivers / usb / gadget / function / f_uvc.c
b / drivers / usb / gadget / function / f_uvc.c
index 9f399f4..4998b96 100644
--- a / drivers / usb / gadget / function / f_uvc.c
+++ b / drivers / usb / gadget / function / f_uvc.c
@@ -1163.6 +1163.7 @@ static struct usb_function_instance * uvc_alloc_inst (void)
        opts-> streaming_interval = 1;
        opts-> streaming_maxpacket = 1024;
        opts-> uvc_num_request = UVC_NUM_REQUESTS;
+ opts-> pm_qos_latency = 0;

        ret = uvcg_attach_configfs (opts);
        if (ret <0) {
diff --git a / drivers / usb / gadget / function / u_uvc.h
b / drivers / usb / gadget / function / u_uvc.h
index be26665..cfa81bc 100644
--- a / drivers / usb / gadget / function / u_uvc.h
+++ b / drivers / usb / gadget / function / u_uvc.h
@@ -87.6 +87.7 @@ struct f_uvc_opts {
         * /
        struct mutex lock;
        int refcnt;
+ int pm_qos_latency;
  };

  #endif / * U_UVC_H * /
diff --git a / drivers / usb / gadget / function / uvc.h
b / drivers / usb / gadget / function / uvc.h
index 82ad867..eed4285 100644
--- a / drivers / usb / gadget / function / uvc.h
+++ b / drivers / usb / gadget / function / uvc.h
@@ -14.6 +14.7 @@
  #include <linux / spinlock.h>
  #include <linux / usb / composite.h>
  #include <linux / videodev2.h>
+ # include <linux / pm_qos.h>

  #include <media / v4l2-device.h>
  #include <media / v4l2-dev.h>
@@ -117.6 +118.8 @@ struct uvc_device {
        enum uvc_state state;
        struct usb_function func;
```

Page 13

```
        struct uvc_video video;
+ / * for creating and issuing QoS requests * /
+ struct pm_qos_request pm_qos;

        / * Descriptors * /
        struct {
diff --git a / drivers / usb / gadget / function / uvc_configfs.c
b / drivers / usb / gadget / function / uvc_configfs.c
index 4da4a34..4233770 100644
--- a / drivers / usb / gadget / function / uvc_configfs.c
+++ b / drivers / usb / gadget / function / uvc_configfs.c
@@ -2775.6 +2775.7 @@ UVCG_OPTS_ATTR (streaming_interval, streaming_interval,
16);
  UVCG_OPTS_ATTR (streaming_maxpacket, streaming_maxpacket, 3072);
  UVCG_OPTS_ATTR (streaming_maxburst, streaming_maxburst, 15);
```

```
  UVCG_OPTS_ATTR (uvc_num_request, uvc_num_request, UVC_MAX_NUM_REQUESTS);
+ UVCG_OPTS_ATTR (pm_qos_latency, pm_qos_latency, PM_QOS_LATENCY_ANY);

  #undef UVCG_OPTS_ATTR

@@ -2839.6 +2840.7 @@ static struct configfs_attribute * uvc_attrs [] = {
        & f_uvc_opts_attr_streaming_maxburst,
        & f_uvc_opts_attr_uvc_num_request,
        & f_uvc_opts_attr_device_name,
+ & f_uvc_opts_attr_pm_qos_latency,
        ZERO,
  };

diff --git a / drivers / usb / gadget / function / uvc_video.c
b / drivers / usb / gadget / function / uvc_video.c
index f1db6a3..ae95de3 100644
--- a / drivers / usb / gadget / function / uvc_video.c
+++ b / drivers / usb / gadget / function / uvc_video.c
@@ -13.6 +13.7 @@
  #include <linux / usb / gadget.h>
  #include <linux / usb / video.h>
  #include <linux / uvcinfo.h>
+ # include <linux / pm_qos.h>

  #include <media / v4l2-dev.h>

@@ -346.9 +347.13 @@ int uvcg_video_enable (struct uvc_video * video, int enable)

                uvc_video_free_requests (video);
                uvcg_queue_enable (& video-> queue, 0);
+ if (pm_qos_request_active (& uvc-> pm_qos))
+ pm_qos_remove_request (& uvc-> pm_qos);
                return 0;
        }

+ pm_qos_add_request (& uvc-> pm_qos, PM_QOS_CPU_DMA_LATENCY,
+ opts-> pm_qos_latency);
        if ((ret = uvcg_queue_enable (& video-> queue, 1)) <0)
                return ret;

diff --git a / drivers / usb / gadget / legacy / webcam.c
b / drivers / usb / gadget / legacy / webcam.c
index 89e7210..e68d921 100644
```

Page 14

```
--- a / drivers / usb / gadget / legacy / webcam.c
+++ b / drivers / usb / gadget / legacy / webcam.c
@@ -383.6 +383.7 @@ webcam_bind (struct usb_composite_dev * cdev)
        uvc_opts-> hs_streaming = uvc_hs_streaming_cls;
        uvc_opts-> ss_streaming = uvc_ss_streaming_cls;
        uvc_opts-> uvc_num_request = UVC_NUM_REQUESTS;
+ uvc_opts-> pm_qos_latency = 0;

        / * Allocate string descriptor numbers ... note that string contents
          * can be overridden by the composite_dev glue.
```