Rockchip AI ISP Developer Guide

ID: RK-KF-YF-A08

Release Version: V1.0.6

Release Date: 2024-01-26

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document provides instructions for AI ISP development.

Product Version

Chipset	Kernel Version
RV1126/RV1109	Linux 4.19
RV1103/RV1106	Linux 5.10

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	Lucien Ma	2023-07-28	Initial version
V1.0.1	Lucien Ma	2023-09-09	(1) Update the debugging method of denoising intensity. (2) Add parameters related to motion denoising. Add visualization tools
V1.0.2	Fenrir Lin	2023-09-11	Add development instructions and reference code
V1.0.3	Fenrir Lin	2023-10-08	Removed parameter debugging section which will be maintained by other documents
V1.0.4	Fenrir Lin	2023-10-23	Added RV1106 related instructions and application process
V1.0.5	Fenrir Lin	2023-11-07	Update performance parameters
V1.0.6	Fenrir Lin	2024-01-26	Update some data

Contents

1. Overview

AI ISP is used to perform low-light noise reduction and intelligent enhancement processing on images output from VI, which ensures clear images with no motion blur and low noise, even in dim or low-light conditions.

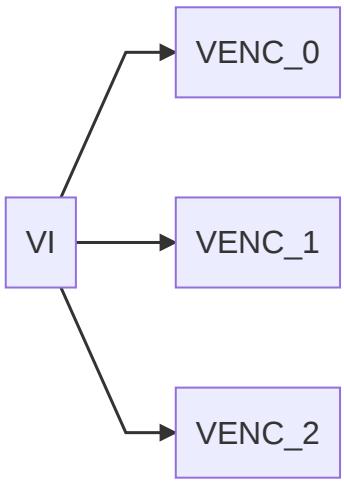
Main Features:

1. After processing by ISP, AI ISP performs a secondary processing of the image before sending it to the subsequent stage.
2. On the RV1126 platform, when using AI ISP, dynamic and static judgments will utilize a new detection library.
3. To improve the frame rate, several modules within AI ISP process in parallel, which requires an increase in the number of VI module buffers. For the RV1126 platform at 400w@25fps, it is recommended to have at least 5 buffers, while for the RV1106 platform at 400w@10fps, 1 buffer should suffice.

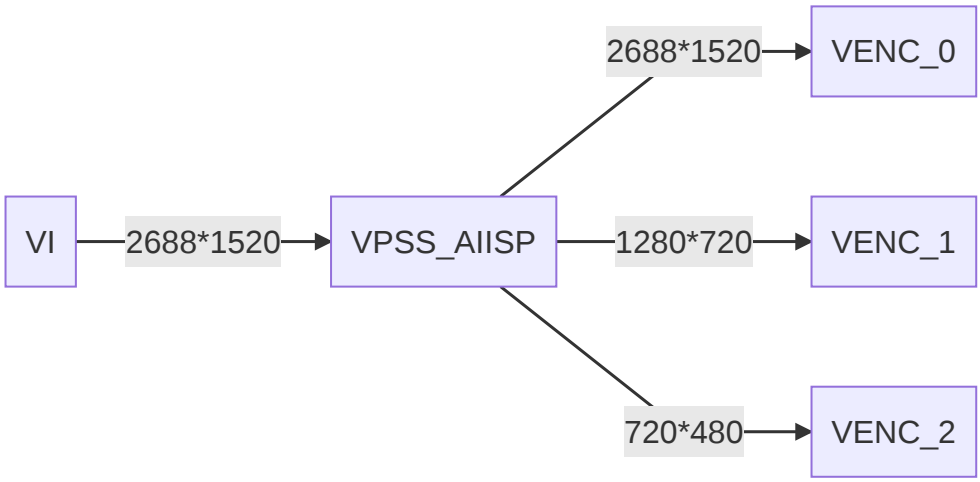
2. Developer Guide

Because all images need to be processed through AI ISP, it is integrated into Rockit's VPSS module, which allows for a one to multiple out effect.

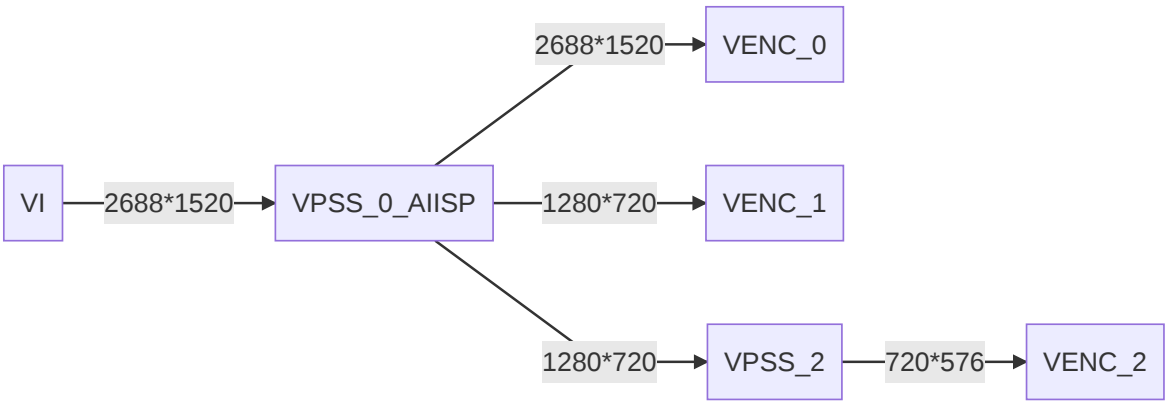
Typically, the media data flow is as shown in the diagram, with multiple channels outputting from VI.



To enable the AI ISP function, it is necessary to modify the original video data stream to VI bound to VPSS and then output from multiple channels of VPSS



To minimize the bandwidth and improve RGA performance, it is recommended to use cascade mode.



2.1 Precondition

1. VPSS is required to support for AI ISP functionality. For RV1126, the Rockit library should be version v1.7.12 or higher, and for RV1106, the Rockit library should be version v1.4.62 or higher.
2. For RV1126, the aiq library should be version v1.0x77.7 or higher, and for RV1106, the aiq library should be version v5.0x4.0 or higher.
3. The SDK should include a directory named `media/rkpostisp`, which contains AI ISP-related libraries and models.

2.2 Reference Code

The code mainly calls the `RK_MPI_VPSS_SetGrpAIISPAttr` interface of the Rockit library and the `rk_aiq_uapi_sysctl_getAinrParams` interface of the aiq library (for RV1106, it's `rk_aiq_uapi2_sysctl_getAinrParams`). **The aiq interface is called to update the `ainr` related parameters, which are then passed along with image data to the AI ISP library via VPSS for processing before being output to various channels**.

Please refers to `media/sample/example/demo/sample_demo_aiisp.c` for details

```
#include "rkpostisp.h"
```

```
#include "rk_aiq_user_api_sysctl.h"
#include <rk_mpi_vpss.h>

static rk_ainr_param ai_param;

/* after rkaiq init */
static void *vpss_aiisp_callback(void *args) {
    rk_aiq_uapi_sysctl_getAinrParams(0, &ai_param);
    return (void *)&ai_param;
}

RK_S32 s32Ret = RK_FAILURE;
VPSS_GRP vpssGrpId = 0;
AIISP_ATTR_S stAIISPAttr;
/* after vpss vpssGrpId create */
memset(&stAIISPAttr, 0, sizeof(AIISP_ATTR_S));
stAIISPAttr.bEnable = RK_TRUE;
stAIISPAttr.stAiIspCallback.pfUpdateCallback = vpss_aiisp_callback;
stAIISPAttr.stAiIspCallback.pPrivateData = RK_NULL;
stAIISPAttr.pModelFilePath = "/oem/usr/lib/";
stAIISPAttr.u32FrameBufCnt = 2;
s32Ret = RK_MPI_VPSS_SetGrpAIISPAttr(vpssGrpId, &stAIISPAttr);
if (RK_SUCCESS != s32Ret) {
    return s32Ret;
}
/*  vi initialization and vi bind vpss code are omitted here */
```

2.3 Compilation Instructions

The IQ parameters used by AI ISP differ significantly from those used in regular scenarios. They are stored separately in the “ainr” directory. The rkipc application also use a separate type.

For detailed configuration, please refer to the `project/cfg/BoardConfig_IPC` directory, which contains configurations with the “AIISP” field. The key differences from other configurations are as follows:

```
export RK_CAMERA_SENSOR_IQFILES="ainr/os04a10_CMK-OT1607-FV1_M12-40IRC-4MP-F16.xml"

export RK_APP_TYPE=RKIPC_RV1126_AIISP
# RV1106 is RK_APP_TYPE=RKIPC_RV1106_AIISP
```

2.4 Performance Indicators

The corresponding version number is: git-3ed5ab0 Sat Nov 4 15:09:32 2023。

The following data is for reference only; actual application scenario test data should prevail.

Based on RV1126, with a CPU clock frequency of 1.5 GHz, the support performance is as follows:

Sensor resolution@frame rate, stream resolution
400w@20fps, 2688*1520 + 1280*720 + 720*576
800w@15fps, 3840*2160 + 1280*720 + 720*576

Based on RV1106G2, with a CPU clock frequency of 1.5G, The support performance is as follows:

Sensor resolution@frame rate, stream resolution
200w@20fps, 1920*1080 + 1280*720 + 720*576
400w@10fps, 2688*1520 + 1280*720 + 720*576
Binocular product with 200w@10fps for each camera, 1920*1080 + 1920*1080

3. Parameter Debug

Please refer to the RKAIISP Debugging Document which is located in the `media/rkpostisp` directory of the IPC SDK.

Note: By default, RKIPC will read image parameters from the ini configuration file and override them. To check the default IQ effect, Customers should manually change `init_form_ini` in `/userdata/rkipc.ini` to 0, then save it and execute `sync`, power off and reboot the device.