Intel Image Processing Unit 3 (IPU3) Imaging Unit (ImgU) driver

Introduction

This file documents the Intel IPU3 (3rd generation Image Processing Unit) Imaging Unit drivers located under drivers/media/pci/intel/ipu3 (CIO2) as well as under drivers/staging/media/ipu3 (ImgU).

The Intel IPU3 found in certain Kaby Lake (as well as certain Sky Lake) platforms (U/Y processor lines) is made up of two parts namely the Imaging Unit (ImgU) and the CIO2 device (MIPI CSI2 receiver).

The CIO2 device receives the raw Bayer data from the sensors and outputs the frames in a format that is specific to the IPU3 (for consumption by the IPU3 ImgU). The CIO2 driver is available as drivers/media/pci/intel/ipu3/ipu3-cio2* and is enabled through the CONFIG_VIDEO_IPU3_CIO2 config option.

The Imaging Unit (ImgU) is responsible for processing images captured by the IPU3 CIO2 device. The ImgU driver sources can be found under drivers/staging/media/ipu3 directory. The driver is enabled through the CONFIG_VIDEO_IPU3_IMGU config option.

The two driver modules are named ipu3_csi2 and ipu3_imgu, respectively.

The drivers has been tested on Kaby Lake platforms (U/Y processor lines).

Both of the drivers implement V4L2, Media Controller and V4L2 sub-device interfaces. The IPU3 CIO2 driver supports camera sensors connected to the CIO2 MIPI CSI-2 interfaces through V4L2 sub-device sensor drivers.

CIO2

The CIO2 is represented as a single V4L2 subdev, which provides a V4L2 subdev interface to the user space. There is a video node for each CSI-2 receiver, with a single media controller interface for the entire device.

The CIO2 contains four independent capture channel, each with its own MIPI CSI-2 receiver and DMA engine. Each channel is modelled as a V4L2 sub-device exposed to userspace as a V4L2 sub-device node and has two pads:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 51)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{0.8cm}|p{4.0cm}|p{4.0cm}|
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 53)

Unknown directive type "flat-table".

```
.. flat-table::
    :header-rows: 1

    * - Pad
      - Direction
      - Purpose

    * - 0
      - sink
      - MIPI CSI-2 input, connected to the sensor subdev

    * - 1
      - source
      - Raw video capture, connected to the V4L2 video interface
```

The V4L2 video interfaces model the DMA engines. They are exposed to userspace as V4L2 video device nodes.

Capturing frames in raw Bayer format

CIO2 MIPI CSI2 receiver is used to capture frames (in packed raw Bayer format) from the raw sensors connected to the CSI2

ports. The captured frames are used as input to the ImgU driver.

Image processing using IPU3 ImgU requires tools such as raw2pnm [2], and yavta [3] due to the following unique requirements and / or features specific to IPU3.

-- The IPU3 CSI2 receiver outputs the captured frames from the sensor in packed raw Bayer format that is specific to IPU3.

-- Multiple video nodes have to be operated simultaneously.

Let us take the example of ov5670 sensor connected to CSI2 port 0, for a 2592x1944 image capture.

Using the media controller APIs, the ov5670 sensor is configured to send frames in packed raw Bayer format to IPU3 CSI2 receiver.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 93)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    # This example assumes /dev/media0 as the CIO2 media device
    export MDEV=/dev/media0

    # and that ov5670 sensor is connected to i2c bus 10 with address 0x36
    export SDEV=$(media-ctl -d $MDEV -e "ov5670 10-0036")

    # Establish the link for the media devices using media-ctl [#f3]_
    media-ctl -d $MDEV -l "ov5670:0 -> ipu3-csi2 0:0[1]"

    # Set the format for the media devices
    media-ctl -d $MDEV -V "ov5670:0 [fmt:SGRBG10/2592x1944]"
    media-ctl -d $MDEV -V "ipu3-csi2 0:0 [fmt:SGRBG10/2592x1944]"
    media-ctl -d $MDEV -V "ipu3-csi2 0:1 [fmt:SGRBG10/2592x1944]"
```

Once the media pipeline is configured, desired sensor specific settings (such as exposure and gain settings) can be set, using the yavta tool.

e.g

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 114)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    yavta -w 0x009e0903 444 $SDEV
    yavta -w 0x009e0913 1024 $SDEV
    yavta -w 0x009e0911 2046 $SDEV
```

Once the desired sensor settings are set, frame captures can be done as below.

e.g

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 124)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    yavta --data-prefix -u -c10 -n5 -I -s2592x1944 --file=/tmp/frame-#.bin \
          -f IPU3_SGRBG10 $(media-ctl -d $MDEV -e "ipu3-cio2 0")
```

With the above command, 10 frames are captured at 2592x1944 resolution, with sGRBG10 format and output as IPU3_SGRBG10 format.

The captured frames are available as /tmp/frame-#.bin files.

ImgU

The ImgU is represented as two V4L2 subdevs, each of which provides a V4L2 subdev interface to the user space.

Each V4L2 subdev represents a pipe, which can support a maximum of 2 streams. This helps to support advanced camera features like Continuous View Finder (CVF) and Snapshot During Video(SDV).

The ImgU contains two independent pipes, each modelled as a V4L2 sub-device exposed to userspace as a V4L2 sub-device node.

Each pipe has two sink pads and three source pads for the following purpose:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 149)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{0.8cm}|p{4.0cm}|p{4.0cm}|
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 151)

Unknown directive type "flat-table".

```
.. flat-table::
    :header-rows: 1

    * - Pad
      - Direction
      - Purpose

    * - 0
      - sink
      - Input raw video stream

    * - 1
      - sink
      - Processing parameters

    * - 2
      - source
      - Output processed video stream

    * - 3
      - source
      - Output viewfinder video stream

    * - 4
      - source
      - 3A statistics
```

Each pad is connected to a corresponding V4L2 video interface, exposed to userspace as a V4L2 video device node.

Device operation

With ImgU, once the input video node ("ipu3-imgu 0/1":0, in <entity>:<pad-number> format) is queued with buffer (in packed raw Bayer format), ImgU starts processing the buffer and produces the video output in YUV format and statistics output on respective output nodes. The driver is expected to have buffers ready for all of parameter, output and statistics nodes, when input video node is queued with buffer.

At a minimum, all of input, main output, 3A statistics and viewfinder video nodes should be enabled for IPU3 to start image processing.

Each ImgU V4L2 subdev has the following set of video nodes.

input, output and viewfinder video nodes

The frames (in packed raw Bayer format specific to the IPU3) received by the input video node is processed by the IPU3 Imaging Unit and are output to 2 video nodes, with each targeting a different purpose (main output and viewfinder output).

Details onand the Bayer format specific to the IPU3 can be found in :ref:`v4l2-pix-fmt-ipu3-sbggr10`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)

(media)ipu3.rst, line 204); *backlink*

Unknown interpreted text role "ref".

The driver supports V4L2 Video Capture Interface as defined at :ref:`devices`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 207); *backlink*

Unknown interpreted text role "ref".

Only the multi-planar API is supported. More details can be found at :ref:`planar-apis`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 209); *backlink*

Unknown interpreted text role "ref".

Parameters video node

The parameters video node receives the ImgU algorithm parameters that are used to configure how the ImgU algorithms process the image.

Details on processing parameters specific to the IPU3 can be found in :ref:`v4l2-meta-fmt-params`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 218); *backlink*

Unknown interpreted text role "ref".

3A statistics video node

3A statistics video node is used by the ImgU driver to output the 3A (auto focus, auto exposure and auto white balance) statistics for the frames that are being processed by the ImgU to user space applications. User space applications can use this statistics data to compute the desired algorithm parameters for the ImgU.

Configuring the Intel IPU3

The IPU3 ImgU pipelines can be configured using the Media Controller, defined at :ref:`media_controller`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 233); *backlink*

Unknown interpreted text role "ref".

Running mode and firmware binary selection

ImgU works based on firmware, currently the ImgU firmware support run 2 pipes in time-sharing with single input frame data. Each pipe can run at certain mode - "VIDEO" or "STILL", "VIDEO" mode is commonly used for video frames capture, and "STILL" is used for still frame capture. However, you can also select "VIDEO" to capture still frames if you want to capture images with less system load and power. For "STILL" mode, ImgU will try to use smaller BDS factor and output larger bayer frame for further YUV processing than "VIDEO" mode to get high quality images. Besides, "STILL" mode need XNR3 to do noise reduction, hence "STILL" mode will need more power and memory bandwidth than "VIDEO" mode. TNR will be enabled in "VIDEO" mode and bypassed by "STILL" mode. ImgU is running at "VIDEO" mode by default, the user can use v4l2 control V4L2_CID_INTEL_IPU3_MODE (currently defined in drivers/staging/media/ipu3/include/uapi/intel-ipu3.h) to query and set the running mode. For user, there is no difference for buffer queueing between the "VIDEO" and "STILL" mode, mandatory input and main output node should be enabled and buffers need be queued, the statistics and the view-finder queues are optional.

The firmware binary will be selected according to current running mode, such log "using binary if_to_osys_striped " or "using binary if_to_osys_primary_striped" could be observed if you enable the ImgU dynamic debug, the binary if_to_osys_striped is selected for "VIDEO" and the binary "if_to_osys_primary_striped" is selected for "STILL".

Processing the image in raw Bayer format

Configuring ImgU V4L2 subdev for image processing

The ImgU V4L2 subdevs have to be configured with media controller APIs to have all the video nodes setup correctly.

Let us take "ipu3-imgu 0" subdev as an example.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 275)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    media-ctl -d $MDEV -r
    media-ctl -d $MDEV -l "ipu3-imgu 0 input":0 -> "ipu3-imgu 0":0[1]
    media-ctl -d $MDEV -l "ipu3-imgu 0":2 -> "ipu3-imgu 0 output":0[1]
    media-ctl -d $MDEV -l "ipu3-imgu 0":3 -> "ipu3-imgu 0 viewfinder":0[1]
    media-ctl -d $MDEV -l "ipu3-imgu 0":4 -> "ipu3-imgu 0 3a stat":0[1]
```

Also the pipe mode of the corresponding V4L2 subdev should be set as desired (e.g 0 for video mode or 1 for still mode) through the control id 0x009819a1 as below.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 287)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    yavta -w "0x009819A1 1" /dev/v4l-subdev7
```

Certain hardware blocks in ImgU pipeline can change the frame resolution by cropping or scaling, these hardware blocks include Input Feeder(IF), Bayer Down Scaler (BDS) and Geometric Distortion Correction (GDC). There is also a block which can change the frame resolution - YUV Scaler, it is only applicable to the secondary output.

RAW Bayer frames go through these ImgU pipeline hardware blocks and the final processed image output to the DDR memory.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 300)

Unknown directive type "kernel-figure".

```
.. kernel-figure::  ipu3_rcb.svg
   :alt: ipu3 resolution blocks image

   IPU3 resolution change hardware blocks
```

Input Feeder

Input Feeder gets the Bayer frame data from the sensor, it can enable cropping of lines and columns from the frame and then store pixels into device's internal pixel buffer which are ready to readout by following blocks.

Bayer Down Scaler

Bayer Down Scaler is capable of performing image scaling in Bayer domain, the downscale factor can be configured from 1X to 1/4X in each axis with configuration steps of 0.03125 (1/32).

Geometric Distortion Correction

Geometric Distortion Correction is used to perform correction of distortions and image filtering. It needs some extra filter and envelope padding pixels to work, so the input resolution of GDC should be larger than the output resolution.

YUV Scaler

YUV Scaler which similar with BDS, but it is mainly do image down scaling in YUV domain, it can support up to 1/12X down scaling, but it can not be applied to the main output.

The ImgU V4L2 subdev has to be configured with the supported resolutions in all the above hardware blocks, for a given input resolution. For a given supported resolution for an input frame, the Input Feeder, Bayer Down Scaler and GDC blocks should be configured with the supported resolutions as each hardware block has its own alignment requirement.

You must configure the output resolution of the hardware blocks smartly to meet the hardware requirement along with keeping the maximum field of view. The intermediate resolutions can be generated by specific tool -

https://github.com/intel/intel-ipu3-pipecfg

This tool can be used to generate intermediate resolutions. More information can be obtained by looking at the following IPU3 ImgU configuration table.

https://chromium.googlesource.com/chromiumos/overlays/board-overlays/+/master

Under baseboard-poppy/media-libs/cros-camera-hal-configs-poppy/files/gcss directory, graph_settings_ov5670.xml can be used as an example.

The following steps prepare the ImgU pipeline for the image processing.

1. The ImgU V4L2 subdev data format should be set by using the VIDIOC_SUBDEV_S_FMT on pad 0, using the GDC width and height obtained above.

2. The ImgU V4L2 subdev cropping should be set by using the VIDIOC_SUBDEV_S_SELECTION on pad 0, with V4L2_SEL_TGT_CROP as the target, using the input feeder height and width.

3. The ImgU V4L2 subdev composing should be set by using the VIDIOC_SUBDEV_S_SELECTION on pad 0, with V4L2_SEL_TGT_COMPOSE as the target, using the BDS height and width.

For the ov5670 example, for an input frame with a resolution of 2592x1944 (which is input to the ImgU subdev pad 0), the corresponding resolutions for input feeder, BDS and GDC are 2592x1944, 2592x1944 and 2560x1920 respectively.

Once this is done, the received raw Bayer frames can be input to the ImgU V4L2 subdev as below, using the open source application v4l2n [2].

For an image captured with 2592x1944 [5] resolution, with desired output resolution as 2560x1920 and viewfinder resolution as 2560x1920, the following v4l2n command can be used. This helps process the raw Bayer frames and produces the desired results for the main output image and the viewfinder output, in NV12 format.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 377)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    v4l2n --pipe=4 --load=/tmp/frame-#.bin --open=/dev/video4
          --fmt=type:VIDEO_OUTPUT_MPLANE,width=2592,height=1944,pixelformat=0X47337069 \
          --reqbufs=type:VIDEO_OUTPUT_MPLANE,count:1 --pipe=1 \
          --output=/tmp/frames.out --open=/dev/video5 \
          --fmt=type:VIDEO_CAPTURE_MPLANE,width=2560,height=1920,pixelformat=NV12 \
          --reqbufs=type:VIDEO_CAPTURE_MPLANE,count:1 --pipe=2 \
          --output=/tmp/frames.vf --open=/dev/video6 \
          --fmt=type:VIDEO_CAPTURE_MPLANE,width=2560,height=1920,pixelformat=NV12 \
          --reqbufs=type:VIDEO_CAPTURE_MPLANE,count:1 --pipe=3 --open=/dev/video7 \
          --output=/tmp/frames.3A --fmt=type:META_CAPTURE,? \
          --reqbufs=count:1,type:META_CAPTURE --pipe=1,2,3,4 --stream=5
```

You can also use yavta [3] command to do same thing as above:

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 393)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    yavta --data-prefix -Bcapture-mplane -c10 -n5 -I -s2592x1944 \
          --file=frame-#.out-f NV12 /dev/video5 & \
    yavta --data-prefix -Bcapture-mplane -c10 -n5 -I -s2592x1944 \
          --file=frame-#.vf -f NV12 /dev/video6 & \
    yavta --data-prefix -Bmeta-capture -c10 -n5 -I \
          --file=frame-#.3a /dev/video7 & \
    yavta --data-prefix -Boutput-mplane -c10 -n5 -I -s2592x1944 \
          --file=/tmp/frame-in.cio2 -f IPU3_SGRBG10 /dev/video4
```

where /dev/video4, /dev/video5, /dev/video6 and /dev/video7 devices point to input, output, viewfinder and 3A statistics video nodes respectively.

Converting the raw Bayer image into YUV domain

The processed images after the above step, can be converted to YUV domain as below.

Main output frames

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 416)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    raw2pnm -x2560 -y1920 -fNV12 /tmp/frames.out /tmp/frames.out.ppm
```

where 2560x1920 is output resolution, NV12 is the video format, followed by input frame and output PNM file.

Viewfinder output frames

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 426)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    raw2pnm -x2560 -y1920 -fNV12 /tmp/frames.vf /tmp/frames.vf.ppm
```

where 2560x1920 is output resolution, NV12 is the video format, followed by input frame and output PNM file.

Example user space code for IPU3

User space code that configures and uses IPU3 is available here.

https://chromium.googlesource.com/chromiumos/platform/arc-camera/+/master/

The source can be located under hal/intel directory.

Overview of IPU3 pipeline

IPU3 pipeline has a number of image processing stages, each of which takes a set of parameters as input. The major stages of pipelines are shown here:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\media\(linux-master)(Documentation)(admin-guide)(media)ipu3.rst, line 448)

Unknown directive type "kernel-render".

```
.. kernel-render:: DOT
   :alt: IPU3 ImgU Pipeline
   :caption: IPU3 ImgU Pipeline Diagram

   digraph "IPU3 ImgU" {
       node [shape=box]
       splines="ortho"
       rankdir="LR"

       a [label="Raw pixels"]
       b [label="Bayer Downscaling"]
       c [label="Optical Black Correction"]
       d [label="Linearization"]
       e [label="Lens Shading Correction"]
       f [label="White Balance / Exposure / Focus Apply"]
       g [label="Bayer Noise Reduction"]
       h [label="ANR"]
       i [label="Demosaicing"]
       j [label="Color Correction Matrix"]
       k [label="Gamma correction"]
       l [label="Color Space Conversion"]
```

```
        m [label="Chroma Down Scaling"]
        n [label="Chromatic Noise Reduction"]
        o [label="Total Color Correction"]
        p [label="XNR3"]
        q [label="TNR"]
        r [label="DDR", style=filled, fillcolor=yellow, shape=cylinder]
        s [label="YUV Downscaling"]
        t [label="DDR", style=filled, fillcolor=yellow, shape=cylinder]

        { rank=same; a -> b -> c -> d -> e -> f -> g -> h -> i }
        { rank=same; j -> k -> l -> m -> n -> o -> p -> q -> s -> t}

        a -> j [style=invis, weight=10]
        i -> j
        q -> r
    }
```

The table below presents a description of the above algorithms.

Name	Description
Optical Black Correction	Optical Black Correction block subtracts a pre-defined value from the respective pixel values to obtain better image quality. Defined in struct ipu3_uapi_obgrid_param.
Linearization	This algo block uses linearization parameters to address non-linearity sensor effects. The Lookup table table is defined in struct ipu3_uapi_isp_lin_vmem_params.
SHD	Lens shading correction is used to correct spatial non-uniformity of the pixel response due to optical lens shading. This is done by applying a different gain for each pixel. The gain, black level etc are configured in struct ipu3_uapi_shd_config_static.
BNR	Bayer noise reduction block removes image noise by applying a bilateral filter. See struct ipu3_uapi_bnr_static_config for details.
ANR	Advanced Noise Reduction is a block based algorithm that performs noise reduction in the Bayer domain. The convolution matrix etc can be found in struct ipu3_uapi_anr_config.
DM	Demosaicing converts raw sensor data in Bayer format into RGB (Red, Green, Blue) presentation. Then add outputs of estimation of Y channel for following stream processing by Firmware. The struct is defined as struct ipu3_uapi_dm_config.
Color Correction	Color Correction algo transforms sensor specific color space to the standard "sRGB" color space. This is done by applying 3x3 matrix defined in struct ipu3_uapi_ccm_mat_config.
Gamma correction	Gamma correction struct ipu3_uapi_gamma_config is a basic non-linear tone mapping correction that is applied per pixel for each pixel component.
CSC	Color space conversion transforms each pixel from the RGB primary presentation to YUV (Y: brightness, UV: Luminance) presentation. This is done by applying a 3x3 matrix defined in struct ipu3_uapi_csc_mat_config
CDS	Chroma down sampling After the CSC is performed, the Chroma Down Sampling is applied for a UV plane down sampling by a factor of 2 in each direction for YUV 4:2:0 using a 4x2 configurable filter struct ipu3_uapi_cds_params.
CHNR	Chroma noise reduction This block processes only the chrominance pixels and performs noise reduction by cleaning the high frequency noise. See struct struct ipu3_uapi_yuvp1_chnr_config.
TCC	Total color correction as defined in struct struct ipu3_uapi_yuvp2_tcc_static_config.
XNR3	eXtreme Noise Reduction V3 is the third revision of noise reduction algorithm used to improve image quality. This removes the low frequency noise in the captured image. Two related structs are being defined, struct ipu3_uapi_isp_xnr3_params for ISP data memory and struct ipu3_uapi_isp_xnr3_vmem_params for vector memory.
TNR	Temporal Noise Reduction block compares successive frames in time to remove anomalies / noise in pixel values. struct ipu3_uapi_isp_tnr3_vmem_params and struct ipu3_uapi_isp_tnr3_params are defined for ISP vector and data memory respectively.

Other often encountered acronyms not listed in above table:

ACC
: Accelerator cluster

AWB_FR
: Auto white balance filter response statistics

BDS
: Bayer downscaler parameters

CCM
: Color correction matrix coefficients

IEFd

Image enhancement filter directed

Obgrid
: Optical black level compensation

OSYS
: Output system configuration

ROI
: Region of interest

YDS
: Y down sampling

YTM
: Y-tone mapping

A few stages of the pipeline will be executed by firmware running on the ISP processor, while many others will use a set of fixed hardware blocks also called accelerator cluster (ACC) to crunch pixel data and produce statistics.

ACC parameters of individual algorithms, as defined by struct ipu3_uapi_acc_param, can be chosen to be applied by the user space through struct struct ipu3_uapi_flags embedded in struct ipu3_uapi_params structure. For parameters that are configured as not enabled by the user space, the corresponding structs are ignored by the driver, in which case the existing configuration of the algorithm will be preserved.

References

[1] drivers/staging/media/ipu3/include/uapi/intel-ipu3.h

[2] (*1*,*2*) https://github.com/intel/nvt

[3] (*1*,*2*) http://git.ideasonboard.org/yavta.git

[4] http://git.ideasonboard.org/?p=media-ctl.git;a=summary

[5] ImgU limitation requires an additional 16x16 for all input resolutions