

# IMX8MPCDUG

## i.MX 8M Plus Camera and Display Guide

Rev. LF5.15.71\_2.2.0 —  
16 December 2022

User guide

### Document information

Information	Content
Keywords	i.MX, Linux, LF5.15.71_2.2.0
Abstract	This document describes the Application Programming Interface (API) of the i.MX 8M Plus ISP Independent Sensor Interface (ISI) module.



## 1 ISP Independent Sensor Interface API

### 1.1 Overview

This document describes the Application Programming Interface (API) of the i.MX 8M Plus ISP Independent Sensor Interface (ISI) module.

Details of the i.MX 8M Plus ISP Independent Sensor Interface API are described in this document.

- Components such as data types, enumerations, relevant structures, and return codes are described first.
- Then function syntax and description are presented.

The API explained in this document is applicable to BSP release LF5.15.71\_2.2.0.

The code is written in C and parameter types follow standard C conventions. This document assumes that the reader understands the fundamentals of C language.

Currently, there are no deprecated functions in this API.

#### 1.1.1 Acronyms and conventions

Table 1. Acronyms

AE	Auto Exposure
AEC	Auto Exposure Control
AF	Auto Focus
AFM	Auto Focus Measurement
AHB	Advance High-Performance Bus
AWB	Auto White Balance
AXI	Advanced eXtensible Interface
BPT	Bad Pixel Table
CAC	Chromatic Aberration Correction
CPROC	Color Processing Module
CTRL	Control Logic Module
DPCC	Defect Pixel Cluster Correction
DPF	De-noising Pre-Filter
FMF	Focus Measure Function
HVS	Human Visual System
IE	Image Effects Module
ISP	Image Signal Processor
ISR	Interrupt Set/Enable Register
LSC	Lens Shade Correction
MI	Memory Interface
MIPI	Mobile Industry Processor Interface (MIPI) Alliance Standard for camera serial interface 2 (CSI-2)

Table 1. Acronyms...continued

MRZE	Main Resize Module
SIMP	Super Impose Module
SMIA	Standard Mobile Imaging Architecture
SoC	System on Chip
SRZE	Self
VSM	Video Stabilization Measurement
WDR	Wide Dynamic Range
YCbCr	Color space with one luma and two chroma components used for digital encoding

### Conventions

- The prefix "0x" indicates a hexadecimal number. For example, 0x32CF.
- The prefix "0b" indicates a binary number. For example, "0b0011.0010.1100.1111".
- Code snippets are given in Consolas or Courier typeset.

## 1.2 Independent Sensor Interface API Components

This section describes the API declared in the **isi/include** directory. Enumerations and structures are listed alphabetically in this document.

### 1.2.1 Numeric Data Types

The following common numeric data types are used.

Name	Data type
uint8_t	Unsigned 8-bit integer
int8_t	Signed 8-bit integer
uint16_t	Unsigned 16-bit integer
int16_t	Signed 16-bit integer
uint32_t	Unsigned 32-bit integer
int32_t	Signed 32-bit integer
uint64_t	Unsigned 64-bit integer
int64_t	Signed 64-bit integer
float	Float

### 1.2.2 RESULT Return Codes

This table specifies the return values for the API functions.

RESULT String Values	Description
RET_FAILURE	General failure
RET_INVALID_PARM	Invalid parameter
RET_NOTSUPP	Feature not supported

RESULT String Values	Description
RET_NULL_POINTER	Callback is a NULL pointer
RET_OUTOFMEM	Not enough memory available
RET_OUTOFRANGE	A configuration parameter is out of range
RET_PENDING	Command pending
RET_SUCCESS	Function successful
RET_WRONG_CONFIG	Given configuration is invalid
RET_WRONG_HANDLE	Invalid instance/HAL handle
RET_WRONG_STATE	Instance is in the wrong state to shut down

### 1.2.3 Enumerations

This section describes the enumeration definitions.

#### 1.2.3.1 IsiBayerPattern\_e

Specifies the sensor Bayer pattern mode.

Enumeration Values	Value	Description
ISI_BAYER_RGGB	0	RGGB Bayer pattern
ISI_BAYER_GRBG	1	GRBG Bayer pattern
ISI_BAYER_GBRG	2	GBRG Bayer pattern
ISI_BAYER_BGGR	3	BGGR Bayer pattern
ISI_BAYER_MAX	4	Maximum number of sensor Bayer pattern components

#### 1.2.3.2 IsiColorComponent\_e

Specifies the color components.

Enumeration Values	Value	Description
ISI_COLOR_COMPONENT_RED	0	Red color component
ISI_COLOR_COMPONENT_GREENR	1	GreenR color component
ISI_COLOR_COMPONENT_GREENB	2	GreenB color component
ISI_COLOR_COMPONENT_BLUE	3	Blue color component
ISI_COLOR_COMPONENT_MAX	4	Maximum number of color components.

#### 1.2.3.3 IsiExpoFrmType\_e

Specifies the sensor exposure time.

Enumeration Values	Value	Description
ISI_EXPO_FRAME_TYPE_1FRAME	0	1 frame exposure type

Enumeration Values	Value	Description
ISI_EXPO_FRAME_TYPE_2FRAMES	1	2 frames exposure type
ISI_EXPO_FRAME_TYPE_3FRAMES	2	3 frames exposure type
ISI_EXPO_FRAME_TYPE_4FRAMES	3	4 frames exposure type

#### 1.2.3.4 IsiFocus\_e

Specifies the focus position type.

Enumeration Values	Value	Description
ISI_FOCUS_MODE_ABSOLUTE	0	Absolute position type
ISI_FOCUS_MODE_RELATIVE	1	Relative position type
ISI_FOCUS_MODE_MAX	2	Maximum number of focus position types.

#### 1.2.3.5 IsiHdrMode\_e

Specifies the sensor HDR mode.

Enumeration Values	Value	Description
ISI_MODE_LINEAR	0	Linear HDR mode
ISI_MODE_HDR_STITCH	1	Stitch HDR mode
ISI_MODE_HDR_NATIVE	2	Native HDR mode
ISI_MODE_HDR_MAX	3	Maximum number of HDR modes.

#### 1.2.3.6 IsiSensorTpgMode\_e

Specifies the sensor test pattern mode.

Enumeration Values	Value	Description
ISI_TPG_DISABLE	0	Disable mode
ISI_TPG_MODE_0	1	Mode 0
ISI_TPG_MODE_1	2	Mode 1
ISI_TPG_MODE_2	3	Mode 2
ISI_TPG_MODE_3	4	Mode 3
ISI_TPG_MODE_4	5	Mode 4
ISI_TPG_MODE_5	6	Mode 5
ISI_TPG_MAX	7	Maximum number of sensor test pattern modes

#### 1.2.3.7 IsiStitchingMode\_e

Specifies the sensor HDR stitching mode.

Enumeration Values	Value	Description
ISI_STITCHING_DUAL_DCG	0	Dual DCG mode 3x12-bit
ISI_STITCHING_3DOL	1	DOL3 frame 3x12-bit
ISI_STITCHING_LINEBYLINE	2	3x12-bit line by line without waiting
ISI_STITCHING_16BIT_COMPRESS	3	16-bit compressed data + 12-bit RAW
ISI_STITCHING_DUAL_DCG_NOWAIT	4	2x12-bit dual DCG without waiting
ISI_STITCHING_2DOL	5	DOL2 frame or 1 CG+VS sx12-bit RAW
ISI_STITCHING_L_AND_S	6	L+S 2x12-bit RAW
ISI_STITCHING_MAX	7	Maximum number of stitching modes

## 1.2.4 Structures

This section describes the structure definitions.

### 1.2.4.1 IsiCamDrvConfig\_t

This structure defines camera sensor driver-specific data.

Structure Members	Type	Description
CameraDriverID	uint32_t	Camera sensor driver ID
*pIsiHalQuerySensor	IsiHalQuerySensor_t	Query sensor mode with HAL handle.
*pfIsiGetSensorIss	IsiGetSensorIss_t	The function pointer to initialize the member IsiSensor in this current structure.
IsiSensor	IsiSensor_t	The structure includes the sensor name and the function pointers to control the sensor in the ISI layer.

### 1.2.4.2 IsidualGain\_t

This structure defines the sensor gain for dual frame exposure HDR.

Structure Members	Type	Description
dualSGain	uint32_t	Gain for short exposure frame (fixed point, q10)
dualGain	uint32_t	Gain for normal exposure frame (fixed point, q10)

### 1.2.4.3 IsidualInt\_t

This structure defines the sensor integration time for dual frame exposure HDR.

Structure Members	Type	Description
dualSIntTime	uint32_t	Integration time for short exposure frame in microsecond (fixed point, q10)
dualIntTime	uint32_t	Integration time for normal exposure frame in microseconds (fixed point, q10)

#### 1.2.4.4 IsiFocusCalibAttr\_t

This structure defines the focus calibration information.

Structure Members	Type	Description
minPos	int32_t	Minimum position
maxPos	int32_t	Maximum position
minStep	int32_t	Minimum step size

#### 1.2.4.5 IsiFocusPos\_t

This structure defines the focus position.

Structure Members	Type	Description
mode	IsiFocus_e	Focus position mode
Pos	int32_t	Focus position

#### 1.2.4.6 IsiLinearGain\_t

This structure defines the sensor gain (fixed point, q10) for linear mode.

```
typedef IsiLinearGain_t to uint32_t
```

#### 1.2.4.7 IsiLinearInt\_t

This structure defines the sensor integration time microsecond (fixed point, q10) for linear mode.

```
typedef IsiLinearInt_t to uint32_t
```

#### 1.2.4.8 IsiQuadGain\_t

This structure defines the sensor gain for quad-frame exposure HDR.

Structure Members	Type	Description
quadVSGain	uint32_t	Gain for very short exposure frame (fixed point, q10)
quadSGain	uint32_t	Gain for short exposure frame (fixed point, q10)
quadGain	uint32_t	Gain for normal exposure frame (fixed point, q10)
quadLGain	uint32_t	Gain for long exposure frame (fixed point, q10)

#### 1.2.4.9 IsiQuadInt\_t

This structure defines the integration time for quad-frame exposure HDR.

Structure Members	Type	Description
quadVSIntTime	uint32_t	Integration time for very short exposure frame in microseconds (fixed point, q10)
quadSIntSTime	uint32_t	Integration time for short exposure frame in microseconds (fixed point, q10)

Structure Members	Type	Description
quadIntTime	uint32_t	Integration time for normal exposure frame in microseconds (fixed point, q10)
quadLIntTime	uint32_t	Integration time for long exposure frame in microseconds (fixed point, q10)

#### 1.2.4.10 IsiSensor\_t

This structure defines attributes for the sensor.

Structure Members	Type	Description
*pszName	const char	Name of the camera-sensor
*pIsiSensorSetPowerIss	IsiSensorSetPowerIss_t	Set sensor power function
*pIsiCreateSensorIss_t	IsiCreateSensorIss_t	Create a sensor handle
*pIsiReleaseSensorIss	IsiReleaseSensorIss_t	Release sensor handle
*pIsiRegisterReadIss	IsiRegisterReadIss_t	Read sensor register
*pIsiRegisterWriteIss	IsiRegisterWriteIss_t	Write sensor register
*pIsiGetSensorModeIss	IsiGetSensorModeIss_t	Get sensor mode information
*pIsiSetSensorModeIss	IsiSetSensorModeIss_t	Set sensor mode index
*pIsiQuerySensorIss	IsiQuerySensorIss_t	Query support sensor mode
*pIsiGetCapsIss	IsiGetCapsIss_t	Get sensor capabilities
*pIsiSetupSensorIss	IsiSetupSensorIss_t	Set sensor format and initialize the sensor
*pIsiGetSensorRevisionIss	IsiGetSensorRevisionIss_t	Get sensor revision ID
*pIsiCheckSensorConnectionIss	IsiCheckSensorConnectionIss_t	Check the sensor connect status
*pIsiSensorSetStreamingIss	IsiSensorSetStreamingIss_t	Set streaming
*pIsiGetAeInfoIss_t	IsiGetAeInfoIss_t	Get AE information
*pIsiSetHdrRatioIss	IsiSetHdrRatioIss_t	Set HDR ratio
*pIsiGetIntegrationTimeIss	IsiGetIntegrationTimeIss_t	Get integration time
*pIsiSetIntegrationTimeIss	IsiSetIntegrationTimeIss_t	Set integration time
*pIsiGetGainIss	IsiGetGainIss_t	Get current sensor gain
*pIsiSetGainIss	IsiSetGainIss_t	Set sensor gain
*pIsiGetSensorFpsIss	IsiGetSensorFpsIss_t	Get current frame rate
*pIsiSetSensorFpsIss_t	IsiSetSensorFpsIss	Set sensor frame rate
*pIsiSetSensorAfpsLimitsIss	IsiSetSensorAfpsLimitsIss_t	Get auto FPS limit
*pIsiGetSensorIspStatusIss	IsiGetSensorIspStatusIss_t	Get ISP status (BLC and WB use sensor WB or ISP WB)



Structure Members	Type	Description
*pIsiGetAeStartExposureIss	IsiGetAeStartExposureIss_t	Get AE start exposure
*pIsiSetAeStartExposureIss	IsiSetAeStartExposureIss_t	Set AE start exposure
*pIsiSensorSetBlcIss	IsiSensorSetBlcIss_t	Set sensor BLC (if sensor BLC is used)
*pIsiSensorSetWBIss	IsiSensorSetWBIss_t	Set sensor WB (if sensor WB is used)
*pIsiSensorGetExpandCurveIss	IsiSensorGetExpandCurveIss_t	Get expand curve (if sensor data is compressed)
*pIsiActivateTestPatternIss_t	IsiActivateTestPatternIss	Set sensor test pattern
*pIsiFocusSetupIss	IsiFocusSetupIss	Create AF handle
*pIsiFocusReleaseIss	IsiFocusReleaseIss_t	Release AF handle
pIsiFocusSetIss_t	IsiFocusSetIss_t	Set focus position
pIsiFocusGetIss_t	IsiFocusGetIss_t	Get focus position
pIsiGetFocusCalibrateIss	IsiGetFocusCalibrateIss_t	Get focus calibration information

#### 1.2.4.11 IsiSensorAeInfo\_t

This structure defines the ISI layer AE information.

Structure Members	Type	Description
oneLineExpTime	uint32_t	Sensor one line exposure time in microsecond (fixed point, q10)
maxIntTime	IsiSensorIntTime_u	Maximum integration time in microsecond
minIntTime	IsiSensorIntTime_u	Minimum integration time in microsecond
maxAGain	IsiSensorGain_u	Maximum analog gain
minAGain	IsiSensorGain_u	Minimum analog gain
maxDGain	IsiSensorGain_u	Maximum digital gain
minDGain	IsiSensorGain_u	Minimum digital gain
gainStep	uint32_t	Sensor gain step (fixed point, q10)
currFps	uint32_t	Sensor current FPS (fixed point, q10)
maxFps	uint32_t	Sensor maximum FPS (fixed point, q10)
minFps	uint32_t	Sensor minimum FPS (fixed point, q10)

Structure Members	Type	Description
minAfps	uint32_t	Sensor minimum Auto FPS (fixed point, q10)
hdrRatio[ISI_EXPO_FRAME_TYPE_MAX-1]	uint32_t	Sensor HDR ratio (fixed point, q10) q10 ISI_EXPO_PARAS_FIX_FRACBITS ISI_EXPO_FRAME_TYPE_1FRAME: no ratio ISI_EXPO_FRAME_TYPE_2FRAMES: hdrRatio[0]=Normal/Short ISI_EXPO_FRAME_TYPE_3FRAMES: hdrRatio[0]=Long/Normal hdrRatio[1]=Normal/Short
intUpdateDlyFrm	uint8_t	Integration update delay frames.
gainUpdateDlyFrm	uint8_t	Gain update delay frames.

#### 1.2.4.12 IsiSensorBlc\_t

This structure defines the configuration structure used to set the sensor black level.

```
typedef IsiSensorBlc_t to sensor_blc_t
```

#### 1.2.4.13 IsiSensorCaps\_t

This structure defines the sensor capabilities.

Structure Members	Type	Description
FieldSelection	uint32_t	Sample fields selection: 0x1: sample all fields 0x2: sample only even fields 0x4: sample only odd fields
YCSequence	uint32_t	Output order of YUV data
Conv422	uint32_t	Color subsampling mode
HPol	uint32_t	Horizontal polarity
VPol	uint32_t	Vertical polarity
Edge	uint32_t	Sample edge
supportModeNum	uint32_t	Number of support modes
currentMode	uint32_t	Current mode

#### 1.2.4.14 IsiSensorContext\_t

This structure defines the sensor context.

Structure Members	Type	Description
SensorId	uint8_t	Sensor ID

Structure Members	Type	Description
I2cBusNum	uint8_t	The I2C bus to which the sensor is connected.
SlaveAddress	uint16_t	The I2C slave address to which the sensor is configured.
SensorInitAddr	uint32_t	Sensor initialized address
SensorInitSize	uint16_t	Sensor initialized size
NrOfAddressBytes	uint8_t	Number of address bytes.
NrOfDataBytes	uint8_t	Number of data bytes.
fd	int	/dev/v4l-subdev file description
HalHandle	HalHandle_t	Handle of HAL session to use. HalHandle_t is typedef void *.
*pSensor	IsiSensor_t	Pointer to the sensor device. IsiSensor_t is typedef IsiSensor_s.

#### 1.2.4.15 IsiSensorExpandCurve\_t

This structure defines the configuration structure used to set the sensor expand curve.

Typedef IsiSensorExpandCurve\_t to sensor\_expand\_curve\_t

#### 1.2.4.16 IsiSensorGain\_t

This structure defines the sensor gain.

Structure Members	Type	Description
expoFrmType	IsiExpoFrmType_e	Sensor exposure frame type
gain	IsiSensorGain_u	Sensor gain

#### 1.2.4.17 IsiSensorGain\_u

This union defines the sensor gain.

Structure Members	Type	Description
linearGainParas	IsiLinearGain_t	Linear gain parameters
dualGainParas	IsidualGain_t	Gain parameters for dual exposure HDR
triGainParas	IsiTriGain_t	Gain parameters for tri-exposure HDR
quadGainParas	IsiQuadGain_t	Gain parameters for quad-exposure HDR

#### 1.2.4.18 IsiSensorInstanceConfig\_t

This structure defines the configuration structure used to create a new sensor instance.

Structure Members	Type	Description
SensorId	uint8_t	Sensor ID

Structure Members	Type	Description
SensorInitAddr	uint32_t	Sensor initialized address
SensorInitSize	uint16_t	Sensor initialized size
HalHandle	HalHandle_t	Handle of HAL session to use
HalDevID	uint32_t	HAL device ID of this sensor
I2cBusNum	uint8_t	The I2C bus to which the sensor is connected.
SlaveAddr	uint16_t	The I2C slave address to which the sensor is configured.
I2cAfBusNum	uint8_t	The I2C bus to which the AF module is connected.
SlaveAfAddr	uint16_t	The I2C slave address of which the AF module is configured.
SensorModeIndex	uint32_t	The current sensor mode index
*pSensor	IsiSensor_t	The pointer to the sensor driver interface
hSensor	IsiSensorHandle_t	Sensor handle returned by IsiCreateSensor Iss IsiSensorHandle_t is typedef void *.
szSensorNode Name [32]	char	Sensor node name

#### 1.2.4.19 IsiSensorIntTime\_t

This structure defines the sensor integration time.

Structure Members	Type	Description
expoFrmType	IsiExpoFrmType_e	Sensor exposure frame type
IntegrationTime	IsiSensorIntTime_u	Sensor integration time

#### 1.2.4.20 IsiSensorIntTime\_u

This union defines the sensor integration time.

Structure Members	Type	Description
linearInt	IsiLinearGain_t	Linear integration time
dualInt	IsidualGain_t	Integration time for dual exposure HDR
triInt	IsiTriGain_t	Integration time for tri-exposure HDR
quadInt	IsiQuadGain_t	Integration time for quad-exposure HDR

#### 1.2.4.21 IsiSensorIspStatus\_t

This structure defines the ISP status of sensor.

Structure Members	Type	Description
useSensorAWB	bool_t	0: use ISP WB 1: use sensor WB

Structure Members	Type	Description
useSensorBLC	bool_t	0: use ISP BLC 1: use sensor BLC

#### 1.2.4.22 IsiSensorMipiInfo

This structure defines Sensor-specific information for MIPI.

Structure Members	Type	Description
ucMipiLanes	uint8_t	Number of MIPI lanes used by the sensor.

#### 1.2.4.23 IsiSensorMode\_t

```
typedef IsiSensorMode_t to vvcam_mode_info_t
```

#### 1.2.4.24 IsiSensorModeInfoArray\_t

```
typedef IsiSensorModeInfoArray_t to vvcam_mode_info_array_t
```

#### 1.2.4.25 IsiSensorWB\_t

This structure defines the configuration structure used to set the sensor WB.

```
typedef IsiSensorWB_t to sensor_white_balance_t
```

#### 1.2.4.26 IsiTriGain\_t

This structure defines the sensor gain for tri-frame exposure HDR.

Structure Members	Type	Description
triSGain	uint32_t	Gain for short exposure frame (fixed point, q10)
triGain	uint32_t	Gain for normal exposure frame (fixed point, q10)
triLGain	uint32_t	Gain for long exposure frame (fixed point, q10)

#### 1.2.4.27 IsiTriInt\_t

This structure defines the integration time for tri-frame exposure HDR.

Structure Members	Type	Description
triSIntTime	uint32_t	Integration time for very short exposure frame in microseconds (fixed point, q10)
triIntTime	uint32_t	Integration time for normal exposure frame in microseconds (fixed point, q10)
triLIntTime	uint32_t	Integration time for long exposure frame in microseconds (fixed point, q10)

#### 1.2.4.28 sensor\_blc\_t

Structure Members	Type	Description
red	uint32_t	Red BLC level

Structure Members	Type	Description
gr	uint32_t	'Gr' BLC level
gb	uint32_t	'Gb' BLC level
blue	uint32_t	Blue BLC level

#### 1.2.4.29 sensor\_data\_compress\_t

Structure Members	Type	Description
enable	uint32_t	0: sensor data is not compressed 1: sensor data is compressed
x_bit	uint32_t	If sensor data is compressed, x_bit represents the data bit width before compression.
y_bit	uint32_t	If sensor data is compressed, y_bit represents the data bit width after compression.

#### 1.2.4.30 sensor\_expand\_curve\_t

Structure Members	Type	Description
x_bit	uint32_t	Input bit width of data decompression curve
y_bit	uint32_t	Output bit width of data decompression curve
expand_px[64]	uint8_t	Data decompression curve input interval index.exp: $1 \ll \text{expand\_px}[i] = \text{expand\_x\_data}[i+1] - \text{expand\_x\_data}[i]$
expand_x_data[65]	uint32_t	65 points of data decompression curve input
expand_y_data[65]	uint32_t	65 points of data decompression curve output

#### 1.2.4.31 sensor\_hdr\_artio\_t

Structure Members	Type	Description
ratio_l_s	uint32_t	Sensor HDR exposure ratio of long exposure to short exposure (fixed point, q10)
ratio_s_vs	uint32_t	Sensor HDR exposure ratio of short exposure to very short exposure (fixed point, q10)
accuracy	uint32_t	Sensor HDR accuracy (fixed point, q10)

#### 1.2.4.32 sensor\_mipi\_info\_s

Structure Members	Type	Description
mipi_lane	uint32_t	MIPI lane

#### 1.2.4.33 sensor\_test\_pattern\_t

Structure Members	Type	Description
enable	uint8_t	Enable/disable sensor test pattern
pattern	uint32_t	Sensor test pattern

## 1.2.4.34 sensor\_white\_balance\_t

Structure Members	Type	Description
r_gain	uint32_t	White Balance (WB) R gain
gr_gain	uint32_t	'WB Gr' gain
gb_gain	uint32_t	'WB Gb' gain
b_gain	uint32_t	'WB B' gain

## 1.2.4.35 vvcam\_ae\_info\_t

Structure Members	Type	Description
def_frm_len_lines	uint32_t	Sensor default frame length lines (is always set to the sensor default mode VTS)
curr_frm_len_lines	uint32_t	Current frame length lines
one_line_exp_time_ns	uint32_t	One line exposure time (in ns) (always = sensor PCLK * HTS)
max_longintegration_line	uint32_t	Maximum long integration line
min_longintegration_line	uint32_t	Minimum long integration line
max_integration_line	uint32_t	Maximum exposure line
min_integration_line	uint32_t	Minimum exposure line
max_vsintegration_line	uint32_t	Maximum very short integration time in microseconds
min_vsintegration_line	uint32_t	Minimum very short integration time in microseconds
max_long_again	uint32_t	Maximum long analog gain (fixed point, q10)
min_long_again	uint32_t	Minimum long analog gain (fixed point, q10)
max_long_dgain	uint32_t	Maximum long digital gain (fixed point, q10)
min_long_dgain	uint32_t	Minimum long digital gain (fixed point, q10)
max_again	uint32_t	Maximum analog gain (fixed point, q10)
min_again	uint32_t	Minimum analog gain (fixed point, q10)
max_dgain	uint32_t	Maximum digital gain (fixed point, q10)
min_dgain	uint32_t	Minimum digital gain (fixed point, q10)
max_short_again	uint32_t	Maximum short analog gain (fixed point, q10)
min_short_again	uint32_t	Minimum short analog gain (fixed point, q10)
max_short_dgain	uint32_t	Maximum short digital gain (fixed point, q10)
min_short_dgain	uint32_t	Minimum short digital gain (fixed point, q10)
start_exposure	uint32_t	Start exposure (exposure lines*gain (fixed point, q10))

Structure Members	Type	Description
gain_step	uint32_t	Gain step (fixed point, q10)
cur_fps	uint32_t	Current frame rate (fixed point, q10)
max_fps	uint32_t	Maximum FPS (fixed point, q10)
min_fps	uint32_t	Minimum FPS (fixed point, q10)
min_afps	uint32_t	Minimum analog FPS (fixed point, q10)
int_update_delay_frm	uint8_t	Integration update delay frame
gain_update_delay_frm	uint8_t	Gain update delay frame
hdr_radio	sensor_hdr_artio_s	HDR radio

#### 1.2.4.36 vvcam\_clk\_s

Structure Members	Type	Description
status	uint32_t	CLK enable status
sensor_mclk	unsigned long	Sensor MIPI clock
csi_max_pixel_clk	unsigned long	Sensor maximum pixel clock

#### 1.2.4.37 vvcam\_mode\_info\_array\_t

This structure is an abstraction of `vvcam_mode_info`.

Structure Members	Type	Description
count	uint32_t	Number of modes supported
modes[VVCAM_SUPPORT_MAX_MODE_COUNT]	vvcam_mode_info	Structure of sensor features

#### 1.2.4.38 vvcam\_mode\_info\_t

Structure Members	Type	Description
index	uint32_t	Mode index
width	uint32_t	Image width
height	uint32_t	Image height
hdr_mode	uint32_t	HDR mode
stitching_mode	uint32_t	HDR stitching mode
bit_width	uint32_t	Sensor bit width
data_compress	sensor_data_compress_t	Sensor data is compressed
bayer_pattern	uint32_t	Bayer mode
ae_info	vvcam_ae_info_t	AE information



Structure Members	Type	Description
mipi_info	sensor_mipi_info_s	Sensor MIPI Information
preg_data	void *	Sensor register configuration point
reg_data_count	uint32_t	Sensor register configuration size

#### 1.2.4.39 vvcam\_sccb\_array\_s

Structure Members	Type	Description
count	uint32_t	Number of SCCB registers
*sccb_data	vvcam_sccb_data_s	SCCB registers data

#### 1.2.4.40 vvcam\_sccb\_cfg\_s

Structure Members	Type	Description
slave_addr	uint8_t	Registers slave address
addr_byte	uint8_t	Registers address byte
data_byte	uint8_t	Registers data byte

#### 1.2.4.41 vvcam\_sccb\_data\_s

Structure Members	Type	Description
addr	uint32_t	Address of the register
data	uint32_t	Data of the register

### 1.3 Independent Sensor Interface Functions

This section provides an overview of the functions for independent sensor interface.

#### 1.3.1 General API Functions

##### IsiSensorSetPowerIss

Description:

This function controls the sensor power.

Syntax:

```
RESULT IsiSensorSetPowerIss (  
    bool_t_t    on  
);
```

Parameters:

on	Sensor power status
----	---------------------

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_FAILURE, RET\_NOTSUPP

### IsiReadRegister

#### Description:

This function reads the value from the specified register from the image sensor device.

#### Syntax:

```
RESULT IsiReadRegister (  
    IsiSensorHandle_t    handle,  
    const uint32_t       RegAddress,  
    uint32_t             *pRegValue  
);
```

#### Parameters:

handle	Sensor instance handle.
RegAddress	Register address.
*pRegValue	Register value read from the register.

#### Returns

[RESULT](#) Return Code: RET\_SUCCESS, RET\_FAILURE, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER, RET\_NOTSUPP

### IsiWriteRegister

#### Description:

This function writes a given number of bytes to the image sensor device by calling the corresponding sensor function.

#### Syntax:

```
RESULT IsiWriteRegister (  
    IsiSensorHandle_t    handle,  
    const uint32_t       RegAddress,  
    const uint32_t       RegValue  
);
```

#### Parameters:

handle	Sensor instance handle.
RegAddress	Register address.
RegValue	Register value to write.

#### Returns

[RESULT](#) Return Code: RET\_SUCCESS, RET\_FAILURE, RET\_WRONG\_HANDLE, RET\_NOTSUPP, RET\_NULL\_POINTER

### IsiCreateSensorIcss

#### Description:

This function creates a sensor instance.

Syntax:

```
RESULT IsiCreateSensorIss (  
    IsiSensorInstanceConfig_t *pConfig  
);
```

Parameters:

*pConfig	Pointer to the configuration of the new sensor instance.
----------	--

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER, RET\_OUTOFMEM

### IsiGetSensorModelIss

Description:

This function is used to get the sensor mode info by sensor mode index.

Syntax:

```
RESULT IsiGetSensorModeIss (  
    IsiSensorHandle_t *handle,  
    void *pmode  
);
```

Parameters:

* handle	Sensor instance handle.
*pmode	Pointer to the vvcam_mode_info data structure.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER, RET\_OUTOFMEM

### IsiSetSensorModelIss

Description:

This function gets sensor mode information by sensor mode index.

Syntax:

```
RESULT IsiSetSensorModeIss (  
    IsiSensorHandle_t handle,  
    IsiSensorMode_t *pmode  
);
```

Parameters:

* handle	Sensor instance handle.
*pmode	Pointer to the IsiSensorMode_t data structure.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NOTSUPP

### IsiQuerySensorIss

Description:

This function is used to query the sensor support modes info.

Syntax:

```
RESULT IsiQuerySensorIss (  
    IsiSensorHandle_t *handle,  
    vvcam_mode_info_array_t *pSensorInfo  
);
```

Parameters:

* handle	Sensor instance handle.
* pSensorInfo	Pointer to the vvcam_mode_info_array_s data structure.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER, RET\_OUTOFMEM

### IsiReleaseSensorIss

Description:

This function destroys/releases a sensor instance.

Syntax:

```
RESULT IsiReleaseSensorIss (  
    IsiSensorHandle_t handle  
);
```

Parameters:

Handle	Sensor instance handle.
--------	-------------------------

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NOTSUPP

### IsiGetCapsIss

Description:

This function fills in the correct pointers for the sensor description structure.

Syntax:

```
RESULT IsiGetCapsIss (  
    IsiSensorHandle_t handle,  
    IsiSensorCaps_t *pIsiSensorCaps  
);
```

## Parameters:

handle	Sensor instance handle.
*pIsiSensorCaps	Pointer to the IsiSensorCaps_t data structure.

## Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER

**IsiSetupSensorIss**

## Description:

This function sets up the image sensor with the specified configuration.

## Syntax:

```
RESULT IsiSetupSensorIss (  
    IsiSensorHandle_t    handle,  
    IsiSensorConfig_t    *pConfig  
);
```

## Parameters:

handle	Sensor instance handle.
*pConfig	Pointer to the IsiSensorCaps_t data structure. (typedef IsiSensorCaps_t IsiSensorConfig_t)

## Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER

**IsiSensorSetStreamingIss**

## Description:

This function enables/disables streaming of sensor data, if possible.

## Syntax:

```
RESULT IsiSensorSetStreamingIss (  
    IsiSensorHandle_t    handle,  
    bool_t               on  
);
```

## Parameters:

handle	Sensor instance handle.
on	New streaming state. BOOL_TRUE = on; BOOL_FALSE = off

## Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE,  
RET\_NULL\_POINTER, RET\_WRONG\_STATE

### IsiCheckSensorConnectionIss

#### Description:

This function checks the connection to the camera sensor, if possible.

#### Syntax:

```
RESULT IsiCheckSensorConnectionIss (  
    IsiSensorHandle_t    handle  
);
```

#### Parameters:

handle	Sensor instance handle.
--------	-------------------------

#### Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER

### IsiGetSensorRevisionIss

#### Description:

This function reads the sensor revision register and returns it.

#### Syntax:

```
RESULT IsiGetSensorRevisionIss (  
    IsiSensorHandle_t    handle,  
    uint32_t             *p_value  
);
```

#### Parameters:

handle	Sensor instance handle.
*p_value	Pointer to the sensor revision register value.

#### Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER, RET\_NOTSUPP

### IsiRegisterWriteIss

#### Description:

This function writes a given number of bytes to the image sensor device by calling the corresponding sensor function.

#### Syntax:

```
RESULT IsiRegisterWriteIss (  
    IsiSensorHandle_t    handle,  
    const uint32_t        address,  
    const uint32_t        *p_value  
);
```

Parameters:

handle	Sensor instance handle.
address	Register address.
*p_value	Register value to write.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NOTSUPP

### IsiRegisterReadIss

Description:

This function reads the value from the specified register from the image sensor device.

Syntax:

```
RESULT IsiRegisterReadIss (  
    IsiSensorHandle_t    handle,  
    const uint32_t       address,  
    uint32_t             value  
);
```

Parameters:

handle	Sensor instance handle.
address	Register address.
value	Register value read from the register.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER, RET\_NOTSUPP

## 1.3.2 AEC API Functions

### IsiGetAeInfoIss

Description:

This function returns the AE basic information.

Syntax:

```
RESULT IsiGetAeInfoIss(  
    IsiSensorHandle_t    handle,  
    IsiSensorAeInfo_t    *pAeInfo  
);
```

Parameters:

handle	Sensor instance handle.
*pAeInfo	Pointer to the IsiSensorAeInfo_t structure.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER, RET\_NOTSUPP, RET\_WRONG\_HANDLE

### IsiSetHdrRatioIss

Description:

This function sets the HDR ratio.

Syntax:

```
RESULT IsiSetHdrRatioIss (
    IsiSensorHandle_t    handle,
    uint8_t              hdrRatioNum ,
    uint32_t*            HdrRatio
);
```

Parameters:

handle	Sensor instance handle.
hdrRatioNum	HDR ratio count.
HdrRatio*	Pointer to the HDR ratio value (fixed point, q10).

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_NULL\_POINTER, RET\_NOTSUPP, RET\_WRONG\_HANDLE

### IsiGetGainIss

Description:

This function reads gain values from the image sensor module.

Syntax:

```
RESULT IsiGetGainIss (
    IsiSensorHandle_t    handle,
    float                *pGain
);
```

Parameters:

handle	Sensor instance handle.
*pGain	Pointer to the gain values.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER

### IsiSetGainIss

Description:



This function writes gain values to the image sensor module.

Syntax:

```
RESULT IsiSetGainIss (
    IsiSensorHandle_t  handle,
    float              NewGain,
    float              *pSetGain,
    float              *hdr_ratio
);
```

Parameters:

handle	Sensor instance handle.
NewGain	Gain to be set.
*pSetGain	Pointer to the gain values.
&hdr_ratio	Pointer to the HDR ratio.

Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER

### IsiGetSensorFpsIss

Description:

This function returns the sensor current frame rate.

Syntax:

```
RESULT IsiGetSensorFpsIss (
    IsiSensorHandle_t  handle,
    uint32_t           *pFps,
);
```

Parameters:

handle	Sensor instance handle.
*pFps	Pointer to the frame rate (fixed point, q10).

Returns

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER, RET\_NOTSUPP

### IsiSetSensorFpsIss

Description:

This function sets the sensor frame rate.

Syntax:

```
RESULT IsiSetSensorFpsIss (
    IsiSensorHandle_t  handle,
```

```
uint32_t      Fps,  
);
```

**Parameters:**

handle	Sensor instance handle.
Fps	Frame rate (fixed point, q10).

**Returns**

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
RET_NULL_POINTER, RET_NOTSUPP, RET_FAILURE
```

**IsiSetSensorAfpsLimitsIss****Description:**

This function set the minimum FPS for auto FPS control.

**Syntax:**

```
RESULT IsiSetSensorAfpsLimitsIss (  
IsiSensorHandle_t      handle,  
uint32_t      minAfps  
);
```

**Parameters:**

handle	Sensor instance handle.
minAfps	Minimum FPS (fixed point, q10) for auto FPS.

**Returns**

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
RET_NULL_POINTER, RET_NOTSUPP
```

**IsiGetAeStartExposureIss****Description:**

This function returns the AE start exposure (IntegrationTime(μs) x Gain) (fixed point, q10).

**Syntax:**

```
RESULT IsiSensorGetStartExposure (  
IsiSensorHandle_t      handle,  
uint64_t      *pExposure  
);
```

**Parameters:**

handle	Sensor instance handle.
*pExposure	Pointer to the AE Start Exposure (Integration Time(μs) x Gain); the value is fixed point q10.

### Returns

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
RET_NULL_POINTER, RET_NOTSUPP
```

### IsiSensorAeSetStartExposure

#### Description:

This function sets the AE start exposure (IntegrationTime x Gain).

#### Syntax:

```
RESULT IsiSensorAeSetStartExposure (  
    IsiSensorHandle_t    handle,  
    uint64_t             fExposure  
);
```

#### Parameters:

handle	Sensor instance handle.
fExposure	The AE Start Exposure (IntegrationTime(μs) x Gain) to set, the value is fixed point q10

### Returns

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
RET_NULL_POINTER, RET_NOTSUPP
```

### IsiGetIntegrationTimeIss

#### Description:

This function gets the current integration time.

#### Syntax:

```
RESULT IsiGetIntegrationTimeIss (  
    IsiSensorHandle_t    handle,  
    IsiSensorIntTime_t    *pIntegrationTime  
);
```

#### Parameters:

handle	Sensor instance handle (such as OV2725).
*pIntegrationTime	Pointer to the data structure IsiSensorIntTime_t.

### Returns

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
RET_NULL_POINTER, RET_NOTSUPP
```

### IsiSetIntegrationTimeIss

#### Description:

This function sets the integration time.

## Syntax:

```
RESULT IsiSetIntegrationTimeIss (  
    IsiSensorHandle_t    handle,  
    IsiSensorIntTime_t    *pIntegrationTime,  
);
```

## Parameters:

handle	Sensor instance handle.
*pIntegrationTime	Pointer to the data structure <code>IsiSensorIntTime_t</code> .

## Returns

RESULT Return Code: RET\_SUCCESS, RET\_NULL\_POINTER, RET\_NOTSUPP, RET\_WRONG\_HANDLE

**IsiGetSensorFpsIss**

## Description:

This function is used to get the sensor current frame rate.

## Syntax:

```
RESULT IsiGetSensorFpsIss (  
    IsiSensorHandle_t    handle,  
    uint32_t            *pFps,  
);
```

## Parameters:

handle	Sensor instance handle.
*pFps	Pointer to frame rate.

## Returns:

[RESULT](#) Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER

**IsiSetSensorFpsIss**

## Description:

This function is used to set the sensor frame rate.

## Syntax:

```
RESULT IsiSetSensorFpsIss (  
    IsiSensorHandle_t    handle,  
    uint32_t            Fps,  
);
```

## Parameters:

handle	Sensor instance handle.
--------	-------------------------

Fps	frame rate.
-----	-------------

Returns:

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER

### IsiGetIntegrationTimeIncrementIss

Description:

This function returns the smallest possible integration time increment.

Syntax:

```
RESULT IsiGetIntegrationTimeIncrementIss (
    IsiSensorHandle_t    handle,
    float                *pIncr
);
```

Parameters:

handle	Sensor instance handle.
*pIncr	Pointer to the smallest possible integration time increment.

Returns:

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER

## 1.3.3 AWB API Functions

### IsiGetSensorIspStatusIss

Description:

This function gets the sensor ISP status.

Syntax:

```
RESULT IsiGetSensorIspStatusIss (
    IsiSensorHandle_t    handle,
    IsiSensorIspStatus_t *pSensorIspStatus
);
```

Parameters:

handle	Sensor instance handle.
*pSensorIspStatus	Pointer to the sensor ISP status structure IsiSensorIspStatus_t.

Returns

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER, RET\_NOTSUPP

### IsiSensorSetBlcIss

Description:

This function is used to set the sensor black level.

Syntax:

```
RESULT IsiSensorSetBlcIss(IsiSensorHandle_t handle,  
    sensor_blc_t *pblc);
```

Parameters:

handle	Sensor instance handle.
pblc	Pointer to the sensor_blc_t structure.

Returns:

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
    RET_NULL_POINTER
```

### IsiSensorSetWB

Description:

This function used to set the sensor white balance.

Syntax:

```
RESULT IsiSensorSetWB (  
    IsiSensorHandle_t    handle,  
    IsiSensorWB_t        *pWb  
);
```

Parameters:

handle	Sensor instance handle.
*pWb	Pointer to the IsiSensorWB_t data structure.

Returns:

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,  
    RET_NULL_POINTER, RET_NOTSUPP
```

## 1.3.4 Expand API Functions

### IsiSensorGetExpandCurveIss

Description:

This function used to get the sensor expand curve.

Syntax:

```
RESULT IsiSensorGetExpandCurveIss (  
    IsiSensorHandle_t    handle,
```

```
sensor_expand_curve_t *pexpand_curve  
);
```

Parameters:

handle	Sensor instance handle.
pexpand_curve	Pointer to the sensor_expand_curve_t structure.

Returns:

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER

### 1.3.5 AF API Functions

#### IsiFocusSetupIss

Description:

This function sets up the focus module.

Syntax:

```
RESULT IsiFocusSetupIss(  
IsiSensorHandle_t      handle  
);
```

Parameters:

handle	Sensor instance handle.
--------	-------------------------

Returns

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NOTSUPP

#### IsiFocusReleaseIss

Description:

This function releases the focus module.

Syntax:

```
RESULT IsiFocusReleaseIss (  
IsiSensorHandle_t      handle,  
);
```

Parameters:

handle	Sensor instance handle.
--------	-------------------------

Returns

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NULL\_POINTER, RET\_NOTSUPP

IsiFocusSetIss

Description:

This function sets the focus position.

Syntax:

```
RESULT IsiFocusSetIss (
    IsiSensorHandle_t    handle,
    IsiFocusPos_t        *pPos
);
```

Parameters:

handle	Sensor instance handle.
*pPos	Pointer to the focus position data structure IsiFocusPos_t.

Returns

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE, RET_NOTSUPP
```

IsiFocusGetIss

Description:

This function gets the focus position.

Syntax:

```
RESULT IsiFocusGetIss (
    IsiSensorHandle_t handle,
    IsiFocusPos_t *pPos
);
```

Parameters:

handle	Sensor instance handle.
*pPos	Pointer to the focus position data structure IsiFocusPos_t.

Returns

```
RESULT Return Code: RET_SUCCESS, RET_WRONG_HANDLE,
    RET_NULL_POINTER, RET_NOTSUPP
```

IsiFocusCalibrateIss

Description:

This function gets the focus calibration information.

Syntax:

```
RESULT IsiFocusCalibrateIss (
    IsiSensorHandle_t    handle
    IsiFoucsCalibAttr_t  *pFocusCalib
);
```



```
);
```

**Parameters:**

handle	Sensor instance handle.
*pFocusCalib	Pointer to the focus calibration data structure IsiFocusCalibAttr_t.

**Returns**

RESULT Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE, RET\_NOTSUPP

### 1.3.6 Test Pattern API Functions

**IsiActivateTestPattern****Description:**

This function activates or deactivates the test pattern of the sensor (default pattern: color bar).

**Syntax:**

```
RESULT IsiActivateTestPattern (  
    IsiSensorHandle_t    handle,  
    IsiSensorTpgMode_e    tpgMode  
);
```

**Parameters:**

handle	Sensor instance handle.
tpgMode	TPG mode.

**Returns**

RESULT Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE,  
RET\_WRONGSTATE, RET\_NULL\_POINTER, RET\_NOTSUPP

### 1.3.7 Miscellaneous API Functions

**IsiDumpAllRegisters****Description:**

This function dumps all registers to the specified file.

**Syntax:**

```
RESULT IsiDumpAllRegisters(  
    IsiSensorHandle_t    handle,  
    const uint8_t        *filename  
);
```

**Parameters:**

handle	Sensor instance handle.
*filename	File name to dump all registers.

Returns:

**RESULT** Return Code: RET\_SUCCESS, RET\_WRONG\_HANDLE,  
RET\_NULL\_POINTER

## 2 Camera Sensor Porting Guide

### 2.1 Overview

This chapter describes the architecture of the i.MX 8M PLUS Image Signal Processing (ISP sensor driver, API functions, and calling process. It also describes the methods to add new APIs and the implementation process for mounting different sensors.

#### Acronyms and Conventions

3A: Auto Exposure, Auto Focus, Auto White Balance

AE: Auto Exposure

AF: Auto Focus

API: Application Programming Interface

AWB: Automatic White Balance

BLC: Black Level Correction

fps: Frames Per Second

I2C: Inter-Integrated Circuit

IOCTL: Input Output Control

ISI: Independent Sensor Interface

ISP: Image Signal Processing

ISS: Image Sensor Specific

VVCAM: Vivante's kernel driver integration layer

WB: White Balance

The prefix "0x" or suffix "H" indicates a hexadecimal number—for example, "0x32CF" or "32CFH".

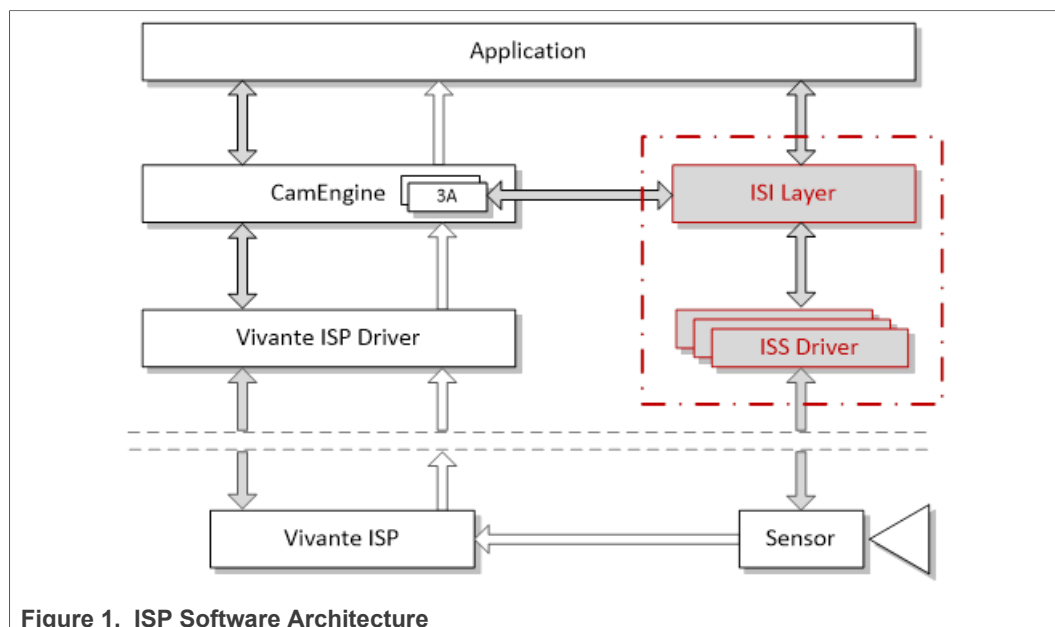
The prefix "0b" indicates a binary number—for example, "0b0011.0010.1100.1111".

Code snippets are given in Consolas typeset.

### 2.2 ISP Software Architecture

In the ISP framework, the application layer and 3A (Auto Exposure, Auto Focus, Auto White Balance) layer calls the sensor API. It is done using function pointers in the ISS through the ISI layer code. The data stream which is output by the sensor is sent directly

to ISP for processing. In the following figure, the gray arrows represent the function calls and the white arrows represent the direction of the output image data of the sensor.



### 2.2.1 ISS (Image Sensor Specific) Driver

- Sensor specific implementation
- Sensor specific attributes and behavior from:
  - Sensor data sheet
  - Calibration data

### 2.2.2 ISP Sensor Module Block Diagrams

The i.MX 8M Plus ISP sensor module is organized as shown in the following figures.

- **Sensor Module in Linux Kernel:** I2C is called in the kernel to read and write the sensor register as shown in [Figure 2](#) below.
  - **ISI Layer:** includes the interface to call the corresponding sensor functions, function pointers to mount different sensors, and the structure composed of these function pointers.
    - **ISS:** uses function pointers so that the ISP driver code can use different sensors independently without modifying the code of other modules.
    - **Sensor API:** includes sensor power-on, initialization, reading and writing sensor registers, configuring sensor resolution, exposure parameters, obtaining current sensor configuration parameters and other functions.
  - **VVCAM:** i.MX 8M Plus ISP kernel driver integration layer which includes ISP, MIPI, camera sensor, and I<sup>2</sup>C kernel driver.
    - **Sensor Driver:** performs sensor API operations on sensor hardware.
    - **I<sup>2</sup>C:** Read-Write Sensor Register. When writing a register, its value must be a 32-bit value. There is no restriction on reading a register.
    - **Kernel Working Mode:** VVCAM has two types of working modes in the kernel:

- **V4L2 Mode:** kernel driver that acts as a part of V4L2 kernel driver, register device name, and operations as a V4L2 subdevice style. This mode is compatible with the V4L2 sensor device format.

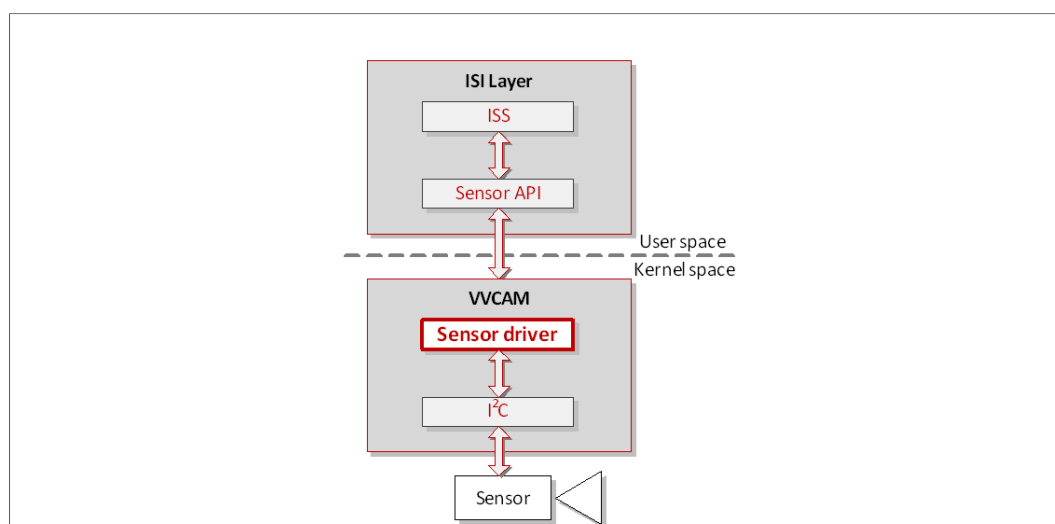


Figure 2. Sensor Module in Linux Kernel

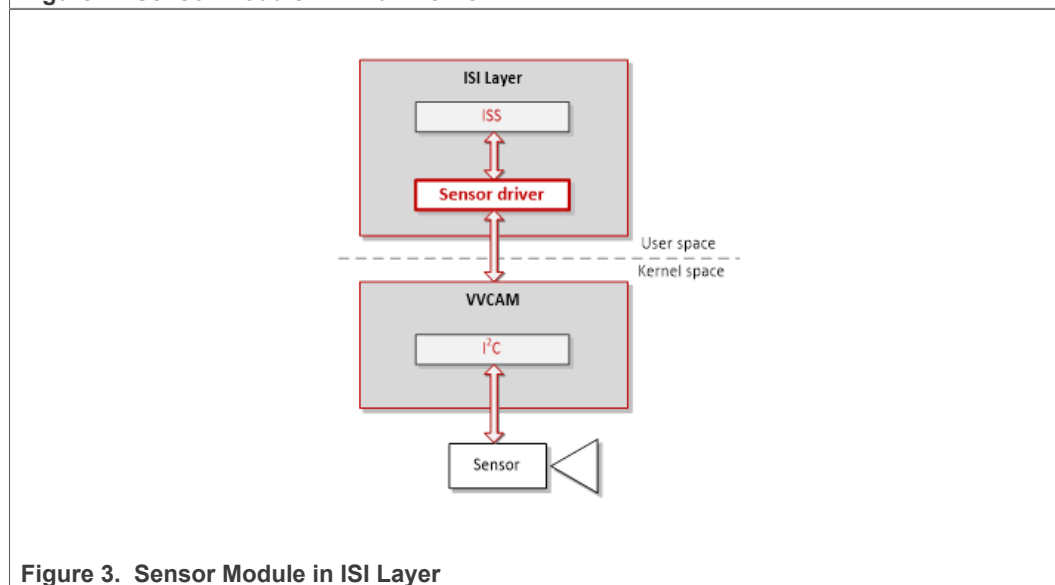


Figure 3. Sensor Module in ISI Layer

## 2.3 ISP Independent Sensor Interface (ISI) API reference

For additional information on the ISI API, see [Section 1](#). Structures and functions are provided here for convenience.

### 2.3.1 ISI Structures

#### 2.3.1.1 IsiCamDrvConfig\_s

This structure defines camera sensor driver-specific data.

Structure Members	Type	Description
CameraDriverID	uint32_t	Camera sensor driver ID
*pIsiHalQuerySensor	IsiHalQuerySensor_t	Query sensor mode with HAL handle.
*pfIsiGetSensorIss	IsiGetSensorIss_t	The function pointer to initialize the member IsiSensor in current structure.
IsiSensor	IsiSensor_t	The structure includes the sensor name and the function pointers to control the sensor in the ISI layer.

### 2.3.1.2 IsiSensor\_t

This structure defines the configuration structure used to create a sensor instance. For data structure definition details, see [Section 1](#).

Structure Members	Type	Description
*pszName	const char	Name of the camera-sensor
pIsiSensorSetPowerIss	IsiSensorSetPowerIss_t	Set sensor power function
pIsiCreateSensorIss_t	IsiCreateSensorIss_t	Create sensor handle
pIsiReleaseSensorIss	IsiReleaseSensorIss_t	Release sensor handle
pIsiRegisterReadIss	IsiRegisterReadIss_t	Read sensor reg
pIsiRegisterWriteIss	IsiRegisterWriteIss_t	Write sensor reg
pIsiGetSensorModeIss	IsiGetSensorModeIss_t	Get sensor mode info
pIsiSetSensorModeIss	IsiSetSensorModeIss_t	Set sensor mode index
pIsiQuerySensorIss	IsiQuerySensorIss_t	Query support sensor mode
pIsiGetCapsIss	IsiGetCapsIss_t	Get sensor caps ability
pIsiSetupSensorIss	IsiSetupSensorIss_t	Set sensor format and initialize sensor
pIsiGetSensorRevisionIss	IsiGetSensorRevisionIss_t	Get sensor revision id
pIsiCheckSensorConnectionIss	IsiCheckSensorConnectionIss_t	Check sensor connect status
pIsiSensorSetStreamingIss	IsiSensorSetStreamingIss_t	Set streaming
pIsiGetAeInfoIss_t	IsiGetAeInfoIss_t	Get AE information
pIsiSetHdrRatioIss	IsiSetHdrRatioIss_t	Set HDR ratio
pIsiGetIntegrationTimeIss	IsiGetIntegrationTimeIss_t	Get integration time
pIsiSetIntegrationTimeIss	IsiSetIntegrationTimeIss_t	Set integration time
pIsiGetGainIss	IsiGetGainIss_t	Get Current sensor gain
pIsiSetGainIss	IsiSetGainIss_t	Set sensor gain
pIsiGetSensorFpsIss	IsiGetSensorFpsIss_t	Get current frame rate

Structure Members	Type	Description
pIsiSetSensorFpsIss_t	IsiSetSensorFpsIss	Set sensor frame rate
pIsiSetSensorAfpsLimitsIss	IsiSetSensorAfpsLimitsIss_t	Get auto FPS limit
pIsiGetSensorIspStatusIss	IsiGetSensorIspStatusIss_t	Get ISP status (BLC and WB use sensor WB or ISP WB)
pIsiGetAeStartExposureIss	IsiGetAeStartExposureIss_t	Get AE start exposure
pIsiSetAeStartExposureIss	IsiSetAeStartExposureIss_t	Set AE start exposure
pIsiSensorSetBlcIss	IsiSensorSetBlcIss_t	Set sensor BLC (if using sensor BLC)
pIsiSensorSetWBIss	IsiSensorSetWBIss_t	Set sensor WB (if using sensor Wb)
pIsiSensorGetExpandCurveIss	IsiSensorGetExpandCurveIss_t	Get expand curve (if sensor data is compressed)
pIsiActivateTestPatternIss_t	IsiActivateTestPatternIss	Set sensor test pattern
pIsiFocusSetupIss	IsiFocusSetupIss	Create AF handle
pIsiFocusReleaseIss	IsiFocusReleaseIss_t	Release AF handle
pIsiFocusSetIss_t	IsiFocusSetIss_t	Set focus position
pIsiFocusGetIss_t	IsiFocusGetIss	Get focus position
pIsiGetFocusCalibrateIss	IsiGetFocusCalibrateIss	Get focus calibration information

### 2.3.1.3 IsiSensorInstanceConfig\_s

This structure defines the configuration structure used to create a sensor instance.

Structure Members	Type	Description
SensorId	uint8_t	Sensor ID
SensorInitAddr	uint32_t	Sensor initialized address
SensorInitSize	uint16_t	Sensor initialized size
HalHandle	HalHandle_t	Handle of HAL session to use
HalDevID	uint32_t	HAL device ID of this sensor
I2cAfBusNum	uint8_t	The I2C bus of focus module
SlaveAfAddr	uint16_t	The I2C slave address of focus module
SensorModeIndex	uint32_t	Sensor mode index
szSensorNodeName[32]	char	Sensor node name
I2cBusNum	uint8_t	The I2C bus of sensor
SlaveAddr	uint16_t	The I2C slave address of sensor

Structure Members	Type	Description
*pSensor	<a href="#">Section 2.3.1.2</a>	The pointer to the sensor driver interface
hSensor	IsiSensorHandle_t	Sensor handle returned by IsiCreateSensorIss

### 2.3.2 ISI Functions

The following ISI API uses the function pointers defined in the [IsiSensor\\_s](#) data structure to call the corresponding sensor functions defined in the [Sensor API Reference](#) section.

Table 2. ISI functions

ISI API	Function Description
IsiSensorSetPowerIss (...)	Power-up/power-down the sensor
IsiReadRegister (...)	Read sensor register value
IsiWriteRegister (...)	Write sensor register value
IsiCreateSensorIss (...)	Create a sensor instance and assign resources to sensor
IsiReleaseSensorIss (...)	Release sensor a sensor instance
IsiGetSensorModeIss (...)	Get sensor mode info
IsiSetSensorModeIss (...)	Set sensor mode index
IsiQuerySensorIss (...)	Query support sensor mode
IsiGetCapsIss (...)	Get sensor caps ability
IsiSetupSensorIss (...)	Set sensor format and int sensor
IsiCheckSensorConnectionIss (...)	Check sensor connect status
IsiGetSensorRevisionIss (...)	Get sensor ID
IsiSensorSetStreamingIss (...)	Set sensor streaming status
IsiGetAeInfoIss (...)	Get AE information
IsiSetHdrRatioIss (...)	Set HDR ratio
IsiGetIntegrationTimeIss (...)	Get exposure time in microseconds (fixed point q10)
IsiSetIntegrationTimeIss (...)	Set exposure time in microseconds (fixed point q10)
IsiGetGainIss (...)	Get sensor gain (fixed point q10)
IsiSetGainIss (...)	Set sensor gain (fixed point q10)
IsiGetSensorFpsIss (...)	Get sensor frame rate (fixed point q10)
IsiSetSensorFpsIss (...)	Set sensor frame rate (fixed point q10)
IsiSetSensorAfpsLimitsIss (...)	Set auto FPS limits (fixed point q10)
IsiGetSensorIspStatusIss (...)	Get sensor module (BLC, WB) status
IsiGetAeStartExposureIss (...)	Get AE start exposure (exposure time us* gain) (fixed point q10)
IsiSetAeStartExposureIss (...)	Set AE start exposure (exposure time us* gain) (fixed point q10)

Table 2. ISI functions...continued

ISI API	Function Description
IsiSensorSetBlc (...)	Set sensor BLC
IsiSensorSetWB (...)	Set sensor WB
IsiSensorGetExpandCurve (...)	Get sensor expand curve
IsiActivateTestPattern (...)	Set sensor test pattern mode
IsiFocusSetupIss (...)	AF module setup
IsiFocusReleaseIss (...)	AF module release
IsiFocusSetIss (...)	Set focus position
IsiFocusGetIss (...)	Get focus position
IsiFocusCalibrateIss (...)	Get focus calibration information
IsiDumpAllRegisters (...)	Reserved

### 2.3.3 Sensor API Reference

This section describes the API defined in `units/isi/drv/<sensor>/source/<sensor>.c`, where `<sensor>` is the name of the sensor (for example, OV2775). You can refer to the APIs in the following table to define your own API for the sensor which you are using. The upper application layer can use the structure of [IsiCamDrvConfig\\_t](#) to call the following functions.

Table 3. Sensor API reference

Sensor API	Description
<b>SENSOR STRUCTURES</b>	
<a href="#">IsiCamDrvConfig_t</a>	Provide a structure for upper layer to access function pointer
<b>SENSOR FUNCTIONS</b>	
<code>&lt;sensor&gt;_IsiSensorSetPowerIss (...)</code>	Power-up/power-down the sensor
<code>&lt;sensor&gt;_IsiReadRegister (...)</code>	Read sensor register value
<code>&lt;sensor&gt;_IsiWriteRegister (...)</code>	Write sensor register value
<code>&lt;sensor&gt;_IsiCreateSensorIss (...)</code>	Create a sensor instance and assign resources to sensor
<code>&lt;sensor&gt;_IsiReleaseSensorIss (...)</code>	Release a sensor instance
<code>&lt;sensor&gt;_IsiGetSensorModeIss (...)</code>	Get sensor mode information
<code>&lt;sensor&gt;_IsiSetSensorModeIss (...)</code>	Set sensor mode index
<code>&lt;sensor&gt;_IsiQuerySensorIss (...)</code>	Query support sensor mode
<code>&lt;sensor&gt;_IsiGetCapsIss (...)</code>	Get sensor caps ability
<code>&lt;sensor&gt;_IsiSetupSensorIss (...)</code>	Set sensor format and int sensor
<code>&lt;sensor&gt;_IsiCheckSensorConnectionIss (...)</code>	Check sensor connect status
<code>&lt;sensor&gt;_IsiGetSensorRevisionIss (...)</code>	Get sensor ID



Table 3. Sensor API reference...continued

Sensor API	Description
<sensor>_IsiSensorSetStreamingIss (...)	Set sensor streaming status
<sensor>_IsiGetAeInfoIss (...)	Get AE info
<sensor>_IsiSetHdrRatioIss (...)	Set HDR ratio
<sensor>_IsiGetIntegrationTimeIss (...)	Get exposure time us (fixed point q10)
<sensor>_IsiSetIntegrationTimeIss (...)	Set exposure time us (fixed point q10)
<sensor>_IsiGetGainIss (...)	Get sensor gain (fixed point q10)
<sensor>_IsiSetGainIss (...)	Set sensor gain (fixed point q10)
<sensor>_IsiGetSensorFpsIss (...)	Get sensor frame rate (fixed point q10)
<sensor>_IsiSetSensorFpsIss (...)	Set sensor frame rate (fixed point q10)
<sensor>_IsiSetSensorAfpsLimitsIss (...)	Set auto FPS limits (fixed point q10)
<sensor>_IsiGetSensorIspStatusIss (...)	Get sensor module (BLC, WB) status
<sensor>_IsiGetAeStartExposureIss (...)	Get AE start exposure (exposure time us* gain) (fixed point q10)
<sensor>_IsiSetAeStartExposureIss (...)	Set AE start exposure (exposure time us* gain) (fixed point q10)
<sensor>_IsiSensorSetBlc (...)	Set sensor BLC
<sensor>_IsiSensorSetWB (...)	Set sensor WB
<sensor>_IsiSensorGetExpandCurve (...)	Get sensor expand curve
<sensor>_IsiActivateTestPattern (...)	Set sensor test pattern mode
<sensor>_IsiFocusSetupIss (...)	AF module setup
<sensor>_IsiFocusReleaseIss (...)	AF module release
<sensor>_IsiFocusSetIss (...)	Set focus position
<sensor>_IsiFocusGetIss (...)	Get focus position
<sensor>_IsiFocusCalibrateIss (...)	Get focus calibration information

### 2.3.4 ISS Sensor Driver User Space Flow

#### Function Pointers

In the ISS (Image Sensor Specific) driver, we define function pointers of the same type as the sensor API and integrate these function pointers into the [IsiSensor\\_s](#) data structure. The driver then integrates the [IsiSensor\\_s](#) structure, camera driver ID, and [IsiGetSensorIss\\_t](#) function pointers into the [IsiCamDrvConfig\\_s](#) data structure. In the function corresponding to the [IsiGetSensorIss\\_t](#) function pointer, the driver mounts the sensor API to the function pointer defined in the ISS layer. The application layer can operate the sensor API by accessing this data structure.

#### Sensor Defines

There are #defines for the sensor which are unique to each sensor. These #defines must be set according to the requirements of the application.

Sensor Exposure Function

The exposure function in the sensor is also different for each sensor. To modify the exposure function, refer to the data sheet of the sensor for specific implementation methods. The IsiGetSensorIss\_t function pointer interface defined in ISI corresponds to the sensor API. Each ISI API calls the corresponding sensor API through the function pointer.

The application layer obtains the address of the function pointer with the IsiCamDrvConfig\_t data structure through the SensorOps::driverChange() function.

```
SensorOps::driverChange(std::string driverFileName, std::string
    calibFileName) {
    ...
    DCT_ASSERT(!pCamDrvConfig-
    >pfIsiGetSensorIss(&pCamDrvConfig->IsiSensor));
    pSensor = &pCamDrvConfig->IsiSensor;
```

At the same time, the application layer passes this address down to ISS so that ISS can access different sensors.

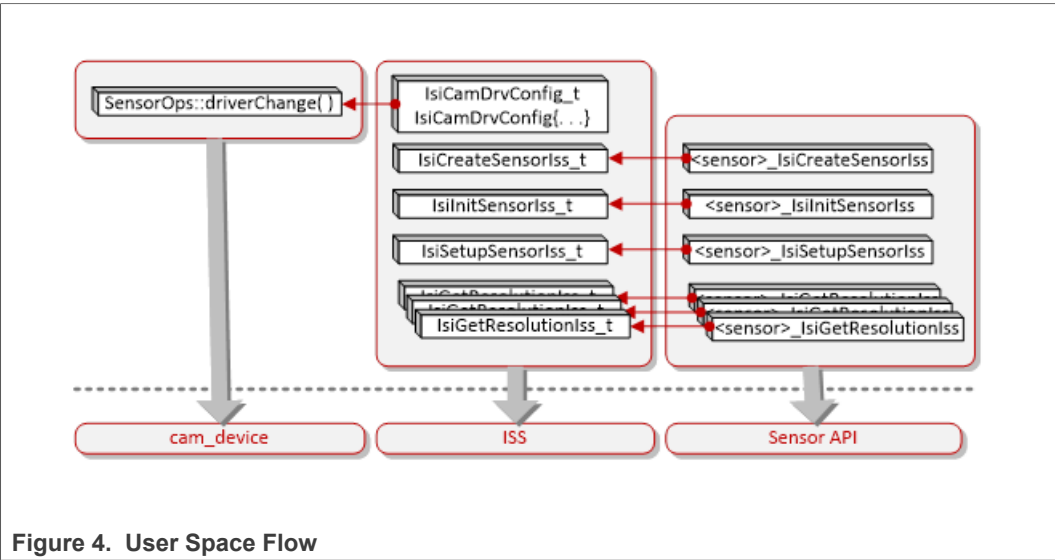


Figure 4. User Space Flow

2.4 IOCTL Introduction

The interface in the user space cannot operate the functions directly in the kernel space. Commands and parameters of the operations are called with the use of IOCTL commands.

2.4.1 IOCTL Commands

The corresponding operations for IOCTL commands in the kernel space are shown in the following table.

**Table 4. IOCTL Commands (Native Mode)**

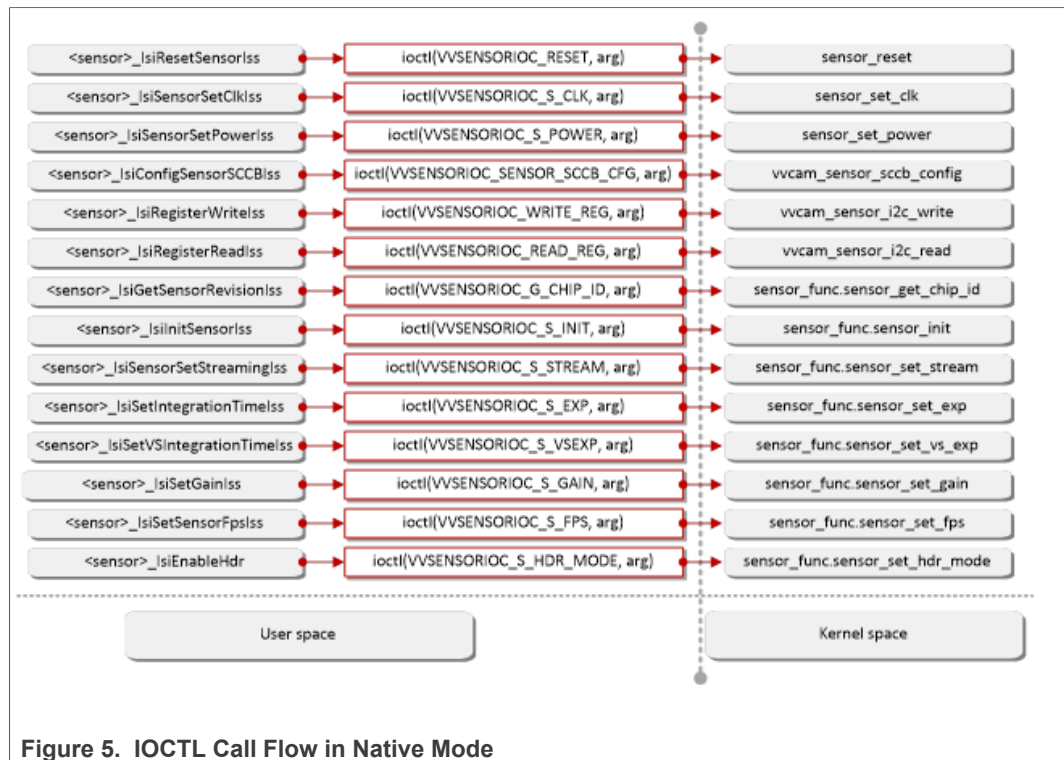
IOCTL	IOCTL Operation
VVSENSORIOC_RESET	Reset sensor
VVSENSORIOC_S_POWER	Set sensor power
VVSENSORIOC_G_POWER	Get sensor power
VVSENSORIOC_S_CLK	Set sensor clock
VVSENSORIOC_G_CLK	Get sensor clock
VVSENSORIOC_QUERY	Query support sensor mode
VVSENSORIOC_S_SENSOR_MODE	Set sensor mode
VVSENSORIOC_G_SENSOR_MODE	Get sensor mode
VVSENSORIOC_READ_REG	Read register
VVSENSORIOC_WRITE_REG	Write register
VVSENSORIOC_READ_ARRAY	Read register array
VVSENSORIOC_WRITE_ARRAY	Write register array
VVSENSORIOC_G_NAME	Get sensor name
VVSENSORIOC_G_RESERVE_ID	Get reserve sensor ID
VVSENSORIOC_G_CHIP_ID	Get chip ID
VVSENSORIOC_S_INIT	Set sensor initialization
VVSENSORIOC_S_STREAM	Set sensor stream
VVSENSORIOC_S_LONG_EXP	Set long exposure
VVSENSORIOC_S_EXP	Set exposure
VVSENSORIOC_S_VSEXP	Set very short exposure
VVSENSORIOC_S_LONG_GAIN	Set long gain
VVSENSORIOC_S_GAIN	Set gain
VVSENSORIOC_S_VSGAIN	Set very short gain
VVSENSORIOC_S_FPS	Set frame rate
VVSENSORIOC_G_FPS	Get frame rate
VVSENSORIOC_S_HDR_RADIO	Set HDR ratio
VVSENSORIOC_S_WB	Set white balance
VVSENSORIOC_S_BLC	Set black level correction
VVSENSORIOC_G_EXPAND_CURVE	Get expand curve
VVSENSORIOC_S_TEST_PATTERN	Set test pattern

## 2.4.2 IOCTL Call Flow

The IOCTL supports both Native Mode and V4L2 Mode as described below.

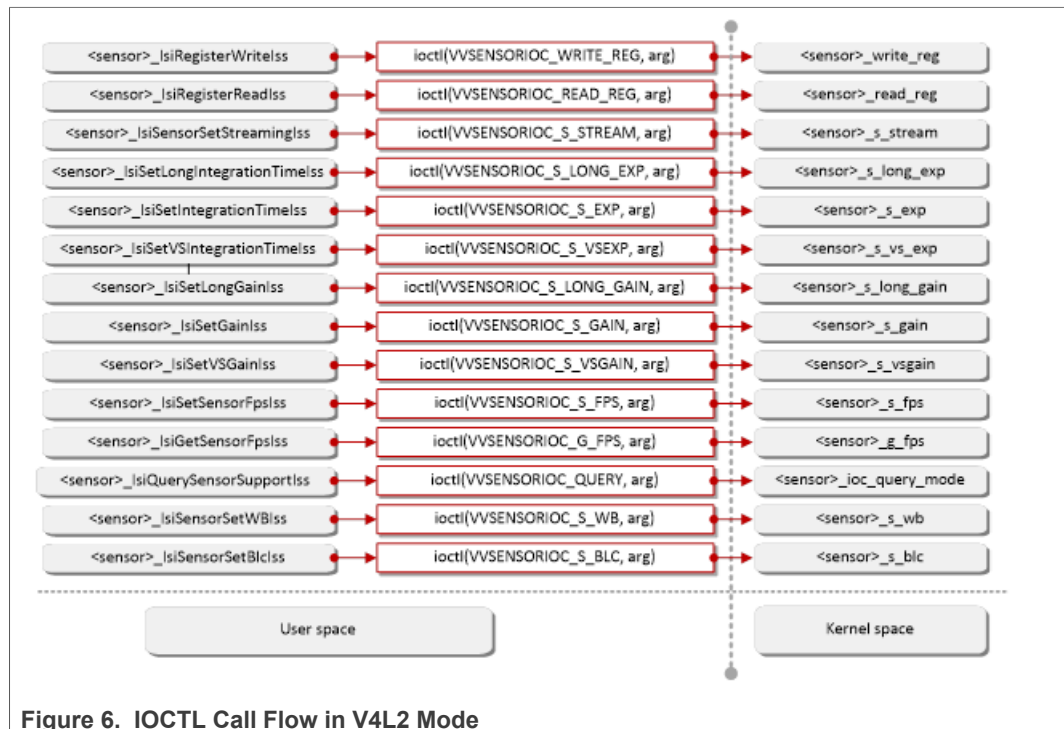
### 2.4.2.1 Native Mode

The figure below shows the IOCTL call flow in Native mode.



#### 2.4.2.2 V4L2 Mode

The figure below shows the IOCTL call flow in V4L2 mode. For more details, refer to the [VVCAM Flow in V4L2 Mode](#) section.



## 2.5 VVCam API Reference

This section describes the API declared in **vvcam/common/vvsensor.h**.

### 2.5.1 Sensor Driver Enumerations

#### 2.5.1.1 SENSOR\_BAYER\_PATTERN\_E

enum Members	Description
BAYER_RGGB	Bayer RGGB pattern mode
BAYER_GRBG	Bayer GRBG pattern mode
BAYER_GBRG	Bayer GBRB pattern mode
BAYER_BGGR	Bayer BGGR pattern mode
BAYER_MAX	Number of Bayer pattern modes

#### 2.5.1.2 sensor\_hdr\_mode\_e

enum Members	Description
SENSOR_MODE_LINEAR	Linear mode.
SENSOR_MODE_HDR_STITCH	ISP HDR mode.
SENSOR_MODE_HDR_NATIVE	Before ISP processes the different exposure images, they are combined in the sensor.

#### 2.5.1.3 sensor\_stitching\_mode\_e

enum Members	Description
SENSOR_STITCHING_DUAL_DCG	Dual DCG mode 3x12-bit
SENSOR_STITCHING_3DOL	3 DOL frame 3x12-bit
SENSOR_STITCHING_LINEBYLINE	3x12-bit line by line without waiting
SENSOR_STITCHING_16BIT_COMPRESS	16-bit compressed data + 12-bit RAW
SENSOR_STITCHING_DUAL_DCG_NOWAIT	2x12-bit dual DCG without waiting
SENSOR_STITCHING_2DOL	DOL2 frame or 1 CG+VS sx12-bit RAW
SENSOR_STITCHING_L_AND_S	L+S 2x12-bit RAW
SENSOR_STITCHING_MAX	Number of sensor stitching modes

### 2.5.2 Sensor Driver Structures

#### 2.5.2.1 sensor\_blc\_t

Structure Members	Type	Description
red	uint32_t	Red Black Level Correction (BLC) level

Structure Members	Type	Description
gr	uint32_t	Gr BLC level
gb	uint32_t	Gb BLC level
blue	uint32_t	Blue BLC level

### 2.5.2.2 sensor\_data\_compress\_t

Structure Members	Type	Description
enable	uint32_t	0: sensor data is not compressed 1: sensor data is compressed
x_bit	uint32_t	If sensor data is compressed, x_bit represents the data bit width before compression.
y_bit	uint32_t	If sensor data is compressed, y_bit represents the data bit width after compression.

### 2.5.2.3 sensor\_expand\_curve\_t

Structure Members	Type	Description
x_bit	uint32_t	Input bit width of data decompression curve
y_bit	uint32_t	Output bit width of data decompression curve
expand_px[64]	uint8_t	Data decompression curve input interval index. exp: $1 \leq \text{expand\_px}[i] = \text{expand\_x\_data}[i+1] - \text{expand\_x\_data}[i]$
expand_x_data[65]	uint32_t	65 points of data decompression curve input
expand_y_data[65]	uint32_t	65 points of data decompression curve output

### 2.5.2.4 sensor\_hdr\_artio\_t

Structure Members	Type	Description
ratio_l_s	uint32_t	Sensor HDR exposure ratio of long exposure to short exposure (fixed point, q10)
ratio_s_vs	uint32_t	Sensor HDR exposure ratio of short exposure to very short exposure (fixed point, q10)
accuracy	uint32_t	Sensor HDR accuracy (fixed point, q10)

### 2.5.2.5 sensor\_mipi\_info

Structure Members	Type	Description
mipi_lane	uint32_t	MIPI lane

## 2.5.2.6 sensor\_test\_pattern\_t

Structure Members	Type	Description
enable	uint8_t	Enable/disable sensor test pattern
pattern	uint32_t	Sensor test pattern

## 2.5.2.7 sensor\_white\_balance\_t

Structure Members	Type	Description
r_gain	uint32_t	White Balance (WB) R gain
gr_gain	uint32_t	WB Gr gain
gb_gain	uint32_t	WB Gb gain
b_gain	uint32_t	WB B gain

## 2.5.2.8 vvcam\_ae\_info\_t

Structure Members	Type	Description
def_frm_len_lines	uint32_t	Sensor default Frame length lines (always in sensor default mode VTS)
curr_frm_len_lines	uint32_t	Current Frame length lines
one_line_exp_time_ns	uint32_t	One line exposure time (in ns) (always = sensor PCLK * HTS)
max_longintegration_line	uint32_t	Maximum long integration line
min_longintegration_line	uint32_t	Minimum long integration line
max_integration_line	uint32_t	Maximum exposure line
min_integration_line	uint32_t	Minimum exposure line
max_vsintegration_line	uint32_t	Maximum very short integration time in micro second
min_vsintegration_line	uint32_t	Minimum very short integration time in micro second
max_long_again	uint32_t	Maximum long analog gain (fixed point, q10)
min_long_again	uint32_t	Minimum long analog gain (fixed point, q10)
max_long_dgain	uint32_t	Maximum long digital gain (fixed point, q10)
min_long_dgain	uint32_t	Minimum long digital gain (fixed point, q10)
max_again	uint32_t	Maximum analog gain (fixed point, q10)
min_again	uint32_t	Minimum analog gain (fixed point, q10)
max_dgain	uint32_t	Maximum digital gain (fixed point, q10)
min_dgain	uint32_t	Minimum digital gain (fixed point, q10)

Structure Members	Type	Description
max_short_again	uint32_t	Maximum short analog gain (fixed point, q10)
min_short_again	uint32_t	Minimum short analog gain (fixed point, q10)
max_short_dgain	uint32_t	Maximum short digital gain (fixed point, q10)
min_short_dgain	uint32_t	Minimum short digital gain (fixed point, q10)
start_exposure	uint32_t	Start exposure (exposure lines*gain (fixed point, q10))
gain_step	uint32_t	Gain step (fixed point, q10)
cur_fps	uint32_t	Current frame rate (fixed point, q10)
max_fps	uint32_t	Maximum FPS (fixed point, q10)
min_fps	uint32_t	Minimum FPS (fixed point, q10)
min_afps	uint32_t	Minimum analog FPS (fixed point, q10)
int_update_delay_frm	uint8_t	Integration update delay frame
gain_update_delay_frm	uint8_t	Gain update delay frame
hdr_radio	<a href="#">Section 2.5.2.4</a>	HDR radio

#### 2.5.2.9 vvcam\_clk\_s

Structure Members	Type	Description
status	uint32_t	CLK enable status
sensor_mclk	unsigned long	Sensor MIPI clock
csi_max_pixel_clk	unsigned long	Sensor maximum pixel clock

#### 2.5.2.10 vvcam\_mode\_info\_array\_t

This structure is an abstraction of `vvcam_mode_info`.

Structure Members	Type	Description
count	uint32_t	Number of modes supported
modes[VVCAM_SUPPORT_MAX_MODE_COUNT]	struct <a href="#">vvcam_mode_info</a>	Structure of sensor feature

#### 2.5.2.11 vvcam\_mode\_info\_t

Structure Members	Type	Description
index	uint32_t	Mode index
width	uint32_t	Image width
height	uint32_t	Image height
hdr_mode	uint32_t	HDR mode
stitching_mode	uint32_t	HDR stitching mode



Structure Members	Type	Description
bit_width	uint32_t	Sensor bit width
data_compress	<a href="#">sensor_data_compress_t</a>	Sensor data is compressed
bayer_pattern	uint32_t	Bayer mode
ae_info	<a href="#">Section 2.5.2.8</a>	AE information
mipl_info	<a href="#">Section 2.5.2.5</a>	Sensor MIPI Information
preg_data	void *	Sensor register configuration point
reg_data_count	uint32_t	Sensor register configuration size

#### 2.5.2.12 vvcam\_sccb\_array\_s

Structure Members	Type	Description
count	uint32_t	Number of SCCB registers
*sccb_data	<a href="#">vvcam_sccb_data_s</a>	SCCB registers data

#### 2.5.2.13 vvcam\_sccb\_cfg\_s

Structure Members	Type	Description
slave_addr	uint8_t	Registers slave address
addr_byte	uint8_t	Registers address byte
data_byte	uint8_t	Registers data byte

#### 2.5.2.14 vvcam\_sccb\_data\_s

Structure Members	Type	Description
addr	uint32_t	Registers address
data	uint32_t	Registers data

#### 2.5.2.15 vvcam\_sensor\_function\_s

This structure defines function pointers corresponding to functions in the sensor driver. This structure is only used in Native mode.

Structure Members	Type	Description
sensor_name[16]	uint8_t	Sensor name
reserve_id	uint32_t	Correct sensor ID which is stored in sensor driver
sensor_clk	uint32_t	Input clock Frequency (Hz) of sensor
mipl_info	<a href="#">sensor_mipl_info_s</a>	Feature of MIPI CSI interface: lanes and data width

Structure Members	Type	Description
sensor_get_chip_id	int32_t (*sensor_get_chip_id) (void *ctx, uint32_t *chip_id)	Function pointer to get ID from sensor register
sensor_init	int32_t (*sensor_init) (void *ctx, vvcam_mode_info_t *pmode)	Function pointer to initialize the vvcam_mode_info data structure
sensor_set_stream	int32_t (*sensor_set_stream) (void *ctx, uint32_t status)	Function pointer to open/close sensor data stream
sensor_set_exp	int32_t (*sensor_set_exp) (void *ctx, uint32_t exp_line)	Function pointer to set the exposure time in line unit
sensor_set_vs_exp	int32_t (*sensor_set_vs_exp) (void *ctx, uint32_t exp_line)	Function pointer to set the exposure time of the very short exposure frame in HDR mode
sensor_set_gain	int32_t (*sensor_set_gain) (void *ctx, uint32_t gain)	Function pointer to set the gain in multiples rather than dB
sensor_set_vs_gain	int32_t (*sensor_set_vs_gain) (void *ctx, uint32_t gain)	Function pointer to set the gain of the very short exposure frame in multiples rather than dB
sensor_set_fps	int32_t (*sensor_set_fps) (void *ctx, uint32_t fps)	Function pointer to set the frame rate of sensor in FPS(frames per second)
sensor_set_resolution	int32_t (*sensor_set_resolution) (void *ctx, uint32_t width, uint32_t height)	Function pointer to set the resolution of sensor
sensor_set_hdr_mode	int32_t (*sensor_set_hdr_mode) (void *ctx, uint32_t hdr_mode)	Function pointer to select the HDR mode of sensor
sensor_query	int32_t (*sensor_query) (void *ctx, <a href="#">vvcam_mode_info_array_t</a> *pmode_info_array)	Function pointer to query sensor information

### 2.5.3 Sensor Driver API

V4I2 Sensor Driver API is declared in file `<sensor>_mipi_v3.c`, where `<sensor>` is the name of the sensor (for example, OV2775).

**Table 5. Sensor Native Driver API**

API Name	Description
extern <a href="#">struct vvcam_sensor_function_s</a> <sensor>_function	Mounts sensor driver to function pointer
sensor_query(...)	Query sensor information
sensor_write_reg(...)	Write register

Table 5. Sensor Native Driver API...continued

API Name	Description
<code>sensor_read_reg(...)</code>	Read register
<code>sensor_write_reg_array(...)</code>	Write register array
<code>sensor_get_chip_id(...)</code>	Get the chip ID in sensor
<code>sensor_init(...)</code>	Initialize the <code>vvcam_mode_info</code> data structure
<code>sensor_set_stream(...)</code>	Turn the flow of the sensor on or off
<code>sensor_set_exp(...)</code>	Write the exposure time of 3A decomposition exposure parameter to the register of the sensor
<code>sensor_set_vs_exp(...)</code>	Write the exposure time of 3A decomposition exposure parameter for a very short exposure frame to the the register of the sensor
<code>sensor_calc_gain(...)</code>	Calculate sensor gain
<code>sensor_set_gain(...)</code>	Set the gain in multiples rather than dB
<code>sensor_set_vs_gain(...)</code>	Set the gain of the very short exposure frame in multiples rather than dB
<code>sensor_set_fps(...)</code>	Set the frame rate of sensor
<code>sensor_set_resolution(...)</code>	Set the resolution of sensor
<code>sensor_set_hdr_mode(...)</code>	Select the HDR mode of sensor

Table 6. Sensor V4I2 Driver API

API Name	Description
<code>&lt;sensor&gt;_g_clk(...)</code>	Get sensor clock
<code>&lt;sensor&gt;_power_on(...)</code>	Power on sensor
<code>&lt;sensor&gt;_power_off(...)</code>	Power off sensor
<code>&lt;sensor&gt;_s_power(...)</code>	Set sensor power
<code>&lt;sensor&gt;_write_reg(...)</code>	Write data to the specified register
<code>&lt;sensor&gt;_read_reg(...)</code>	Read data from the specified register
<code>&lt;sensor&gt;_write_reg_array(...)</code>	Write register array
<code>&lt;sensor&gt;_query_capability(...)</code>	Query sensor capability
<code>&lt;sensor&gt;_query_supports(...)</code>	Query sensor support modes
<code>&lt;sensor&gt;_get_sensor_id(...)</code>	Get sensor ID
<code>&lt;sensor&gt;_get_reserve_id(...)</code>	Get reserve sensor ID
<code>&lt;sensor&gt;_get_sensor_mode(...)</code>	Get sensor mode
<code>&lt;sensor&gt;_set_sensor_mode(...)</code>	Set sensor mode
<code>&lt;sensor&gt;_set_lexp(...)</code>	Write the exposure time of 3A decomposition exposure parameter for a long exposure frame to the the register of the sensor

Table 6. Sensor V4l2 Driver API...continued

API Name	Description
<sensor>_set_exp(...)	Write the exposure time of 3A decomposition exposure parameter to the the register of the sensor
<sensor>_set_vsexp(...)	Write the exposure time of 3A decomposition exposure parameter for a very short exposure frame to the the register of the sensor
<sensor>_set_lgain(...)	Set the gain of the long exposure frame in multiples rather than dB
<sensor>_set_gain(...)	Set the gain in multiples rather than dB
<sensor>_set_vsgain(...)	Set the gain of the very short exposure frame in multiples rather than dB
<sensor>_set_fps(...)	Set sensor FPS
<sensor>_get_fps(...)	Get sensor FPS
<sensor>_set_test_pattern(...)	Set test pattern
<sensor>_set_ratio(...)	Set sensor HDR ratio
<sensor>_set_blc(...)	Set sensor sub BLC
<sensor>_set_wb(...)	Set white balance
<sensor>_get_expand_curve(...)	Get sensor expand curve
<sensor>_get_format_code(...)	Get format code
<sensor>_s_stream(...)	Start or stop the sensor
<sensor>_enum_mbus_code(...)	Enum MBUS code
<sensor>_set_fmt(...)	Set sensor format
<sensor>_get_fmt(...)	Get sensor format
<sensor>_priv_ioctl(...)	Private I/O control
<sensor>_link_setup(...)	Link setup
<sensor>_regulator_enable(...)	Enable regulator
<sensor>_regulator_disable(...)	Disable regulator
<sensor>_set_clk_rate(...)	Set clock rate
<sensor>_reset(...)	Reset sensor
<sensor>_retrieve_capture_properties(...)	Retrieve capture properties
<sensor>_probe(...)	Probe sensor
<sensor>_remove(...)	Remove sensor
<sensor>_suspend(...)	Suspend sensor
<sensor>_resume(...)	Resume sensor

## 2.6 Camera Sensor Driver in V4L2 Mode

### 2.6.1 VVCAM Flow in V4L2 Mode

Read through this section carefully before porting the new sensor driver in V4L2 Mode. If you have any problems during the sensor porting process, refer to the existing sensor driver of the platform in your source code release.

To add a function interface, refer to the following sections:

- [ISI API Reference](#)
- [ISS Sensor Driver User Space Flow](#)
- [Sensor API Reference](#)
- [Section 2.6.1](#) (this section)

Both hub and sensor kernel driver must add corresponding interfaces and calls. While porting the sensor, be aware that different sensors in the sensor data sheet have different conversion methods when converting the exposure parameters which are passed down from the 3A modules to the values written in the registers. The sensor data must be accurately defined.

To port the camera sensor, the following steps must be taken as described in the following sections:

1. Create sensor DTB file in kernel.
2. Create sensor V4L2 driver in VVCAM (*VVCAM is the kernel driver integration layer of Vivante*).
3. Create sensor ISI API in ISI Layer.

#### 2.6.1.1 Sensor Driver Software Architecture in V4L2 Mode

The software architecture of the sensor driver in V4L2 Mode is shown in the figure below. The V4L2-subdev driver is defined in file `vvcam/v4l2/sensor/<sensor>/<sensor>_xxxx.c`, where `<sensor>` is the name of the sensor (for example, OV2775).

A device node of the sensor named `v4l-subdevx` can be created in `/dev` for direct access. Function `<sensor>_priv_ioctl()` is used in the kernel space to receive the commands and parameters passed down by the user space through `ioctl()`. It is also used to call the corresponding functions in `<sensor>_xxxx.c` according to the commands.

**Note:** Developers should replace the Vivante V4L2-Subdev Driver with their own sensor as shown in the figure below.

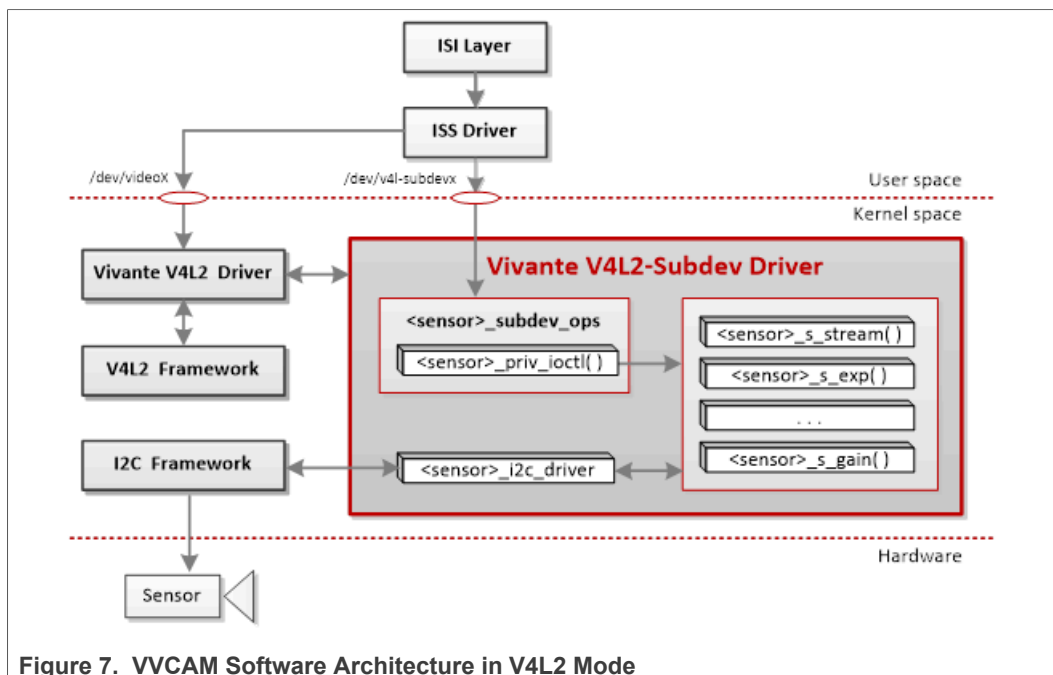


Figure 7. VVCAM Software Architecture in V4L2 Mode

## 2.6.2 Camera Sensor Porting Setup in V4L2 Mode

### 2.6.2.1 Create Sensor DTS File in Kernel

When adding a sensor, a new DTS file must be added to support the kernel device driver. If sensor0 is connected to i2C2, add a sensor device to the i2C2 node.

For example:

```
&i2c2 {
    /delete-node/ov5640_mipi@3c;
    ov2775_0: ov2775_mipi@36 {
        compatible = "ovti,ov2775";
        reg = <0x36>;
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_csi0_pwn>, <&pinctrl_csi0_rst>,
        <&pinctrl_csi_mclk>;
        clocks = <&clk_IMX8MP_CLK_IPP_DO_CLKO2>;
        clock-names = "csi_mclk";
        assigned-clocks = <&clk_IMX8MP_CLK_IPP_DO_CLKO2>;
        assigned-clock-parents = <&clk_IMX8MP_CLK_24M>;
        assigned-clock-rates = <24000000>;
        csi_id = <0>;
        pwn-gpios = <&gpio2 11 GPIO_ACTIVE_HIGH>;
        rst-gpios = <&gpio1 6 GPIO_ACTIVE_LOW>;
        mclk = <24000000>;
        mclk_source = <0>;
        status = "okay";
        port {
            ov2775_mipi_0_ep: endpoint {
                data-lanes = <1 2 3 4>;
                clock-lanes = <0>;
                max-pixel-frequency = /bits/ 64 <5000000000>;
                remote-endpoint = <&mipi_csi0_ep>;
            };
        }
    };
};
```

```
};
};
};
```

### 2.6.2.2 Create Sensor V4L2 Driver in VVCAM

1. Create struct `vvcam_mode_info_s` for your support modes information.

```
static struct vvcam_mode_info_s pov2775_mode_info[] = {
{
    .index          = 0,
    .width          = 1920,
    .height         = 1080,
    .hdr_mode       = SENSOR_MODE_LINEAR,
    .bit_width      = 12,
    .data_compress  = {
        .enable = 0,
    },
    .bayer_pattern  = BAYER_BGGR,
    .ae_info = {
        .def_frm_len_lines    = 0x466,
        .curr_frm_len_lines   = 0x466,
        .one_line_exp_time_ns = 29625,
        .max_integration_line  = 0x466 - 4,
        .min_integration_line  = 1,
        .max_again            = 8 * 1024,
        .min_again            = 2 * 1024,
        .max_dgain            = 4 * 1024,
        .min_dgain            = 1.5 * 1024,
        .gain_step            = 4,
        .start_exposure       = 3 * 400 * 1024,
        .cur_fps               = 30 * 1024,
        .max_fps               = 30 * 1024,
        .min_fps               = 5 * 1024,
        .min_afps              = 5 * 1024,
        .int_update_delay_frm  = 1,
        .gain_update_delay_frm = 1,
    },
    .mipi_info = {
        .mipi_lane = 4,
    },
    .preg_data      = ov2775_init_setting_1080p,
    .reg_data_count = ARRAY_SIZE(ov2775_init_setting_1080p),
},
{
    .index          = 1,
    .width          = 1920,
    .height         = 1080,
    .hdr_mode       = SENSOR_MODE_HDR_STITCH,
    .stitching_mode = SENSOR_STITCHING_DUAL_DCG,
    .bit_width      = 12,
    .data_compress  = {
        .enable = 0,
    },
    .bayer_pattern  = BAYER_BGGR,
    .ae_info = {
        .def_frm_len_lines    = 0x466,
        .curr_frm_len_lines   = 0x466,
        .one_line_exp_time_ns = 59167,
        .max_integration_line  = 0x400,
```

```

        .min_integration_line      = 1,
        .max_vsintegration_line    = 44,
        .min_vsintegration_line    = 1,
        .max_long_again            = 8 * 1024 * DCG_CONVERSION_GAIN,
        .min_long_again            = 1 * 1024 * DCG_CONVERSION_GAIN,
        .max_long_dgain            = 4 * 1024,
        .min_long_dgain            = 2 * 1024,
        .max_again                 = 8 * 1024,
        .min_again                 = 2 * 1024,
        .max_dgain                 = 4 * 1024,
        .min_dgain                 = 1.5 * 1024,
        .max_short_again           = 8 * 1024,
        .min_short_again           = 2 * 1024,
        .max_short_dgain           = 4 * 1024,
        .min_short_dgain           = 1.5 * 1024,
        .start_exposure            = 3 * 400 * 1024,
        .gain_step                  = 4,
        .cur_fps                    = 30 * 1024,
        .max_fps                    = 30 * 1024,
        .min_fps                    = 5 * 1024,
        .min_afps                   = 5 * 1024,
        .hdr_ratio                  = {
            .ratio_l_s = 8 * 1024,
            .ratio_s_vs = 8 * 1024,
            .accuracy = 1024,
        },
        .int_update_delay_frm = 1,
        .gain_update_delay_frm = 1,
    },
    .mipi_info = {
        .mipi_lane = 4,
    },
    .preg_data = ov2775_init_setting_1080p_hdr,
    .reg_data_count =
ARRAY_SIZE(ov2775_init_setting_1080p_hdr),
    },
    {
        .index                = 2,
        .width                 = 1920,
        .height                = 1080,
        .hdr_mode              = SENSOR_MODE_HDR_NATIVE,
        .stitching_mode        = SENSOR_STITCHING_DUAL_DCG_NOWAIT,
        .bit_width             = 12,
        .data_compress         = {
            .enable = 1,
            .x_bit  = 16,
            .y_bit  = 12,
        },
    },
    .bayer_pattern = BAYER_BGGR,
    .ae_info = {
        .def_frm_len_lines    = 0x466,
        .curr_frm_len_lines   = 0x466,
        .one_line_exp_time_ns = 59167,
        .max_integration_line  = 0x466 - 4,
        .min_integration_line  = 1,
        .max_long_again        = 8 * 1024 * DCG_CONVERSION_GAIN,
        .min_long_again        = 1 * 1024 * DCG_CONVERSION_GAIN,
        .max_long_dgain        = 4 * 1024,
        .min_long_dgain        = 2 * 1024,
        .max_again             = 8 * 1024 ,
    },

```



```

        .min_again      = 2 * 1024,
        .max_dgain      = 4 * 1024,
        .min_dgain      = 1.5 * 1024,
        .start_exposure = 3 * 400 * 1024,
        .gain_step       = 4,
        .cur_fps         = 30 * 1024,
        .max_fps         = 30 * 1024,
        .min_fps         = 5 * 1024,
        .min_afps        = 5 * 1024,
        .hdr_ratio       = {
            .ratio_l_s = 8 * 1024,
            .ratio_s_vs = 8 * 1024,
            .accuracy = 1024,
        },
        .int_update_delay_frm = 1,
        .gain_update_delay_frm = 1,
    },
    .mipi_info = {
        .mipi_lane = 4,
    },
    .preg_data = ov2775_1080p_native_hdr_regs,
    .reg_data_count =
        ARRAY_SIZE(ov2775_1080p_native_hdr_regs),
    },
};

```

## 2. Register the new sensor V4L2 driver.

```

static int ov2775_probe(struct i2c_client *client,
                        const struct i2c_device_id *id)
{
    int retval;
    struct device *dev = &client->dev;
    struct v4l2_subdev *sd;
    struct ov2775 *sensor;
    u32 chip_id = 0;
    u8 reg_val = 0;
    pr_info("enter %s\n", __func__);
    sensor = devm_kmalloc(dev, sizeof(*sensor), GFP_KERNEL);
    if (!sensor)
        return -ENOMEM;
    memset(sensor, 0, sizeof(*sensor));
    sensor->i2c_client = client;
    sensor->pwn_gpio = of_get_named_gpio(dev->of_node, "pwn-
gpios", 0);
    if (!gpio_is_valid(sensor->pwn_gpio))
        dev_warn(dev, "No sensor pwn pin available");
    else {
        retval = devm_gpio_request_one(dev, sensor->pwn_gpio,
            GPIOF_OUT_INIT_HIGH,
            "ov2775_mipi_pwdn");
        if (retval < 0) {
            dev_warn(dev, "Failed to set power pin\n");
            dev_warn(dev, "retval=%d\n", retval);
            return retval;
        }
    }
    sensor->rst_gpio = of_get_named_gpio(dev->of_node, "rst-
gpios", 0);
    if (!gpio_is_valid(sensor->rst_gpio))
        dev_warn(dev, "No sensor reset pin available");
}

```

```

else {
    retval = devm_gpio_request_one(dev, sensor->rst_gpio,
        GPIOF_OUT_INIT_HIGH,
        "ov2775_mipi_reset");
    if (retval < 0) {
        dev_warn(dev, "Failed to set reset pin\n");
        return retval;
    }
}
sensor->sensor_clk = devm_clk_get(dev, "csi_mclk");
if (IS_ERR(sensor->sensor_clk)) {
    sensor->sensor_clk = NULL;
    dev_err(dev, "clock-frequency missing or invalid\n");
    return PTR_ERR(sensor->sensor_clk);
}
retval = of_property_read_u32(dev->of_node, "mclk",
&(sensor->mclk));
if (retval) {
    dev_err(dev, "mclk missing or invalid\n");
    return retval;
}
retval = of_property_read_u32(dev->of_node, "mclk_source",
    (u32 *)&(sensor->mclk_source));
if (retval) {
    dev_err(dev, "mclk_source missing or invalid\n");
    return retval;
}
retval = of_property_read_u32(dev->of_node, "csi_id",
&(sensor->csi_id));
if (retval) {
    dev_err(dev, "csi id missing or invalid\n");
    return retval;
}
retval = ov2775_retrieve_capture_properties(sensor, &sensor-
>ocp);
if (retval) {
    dev_warn(dev, "retrive capture properties error\n");
}
sensor->io_regulator = devm_regulator_get(dev, "DOVDD");
if (IS_ERR(sensor->io_regulator)) {
    dev_err(dev, "cannot get io regulator\n");
    return PTR_ERR(sensor->io_regulator);
}
sensor->core_regulator = devm_regulator_get(dev, "DVDD");
if (IS_ERR(sensor->core_regulator)) {
    dev_err(dev, "cannot get core regulator\n");
    return PTR_ERR(sensor->core_regulator);
}
sensor->analog_regulator = devm_regulator_get(dev, "AVDD");
if (IS_ERR(sensor->analog_regulator)) {
    dev_err(dev, "cannot get analog regulator\n");
    return PTR_ERR(sensor->analog_regulator);
}
retval = ov2775_regulator_enable(sensor);
if (retval) {
    dev_err(dev, "regulator enable failed\n");
    return retval;
}
ov2775_set_clk_rate(sensor);
retval = clk_prepare_enable(sensor->sensor_clk);

```

```

if (retval < 0) {
    dev_err(dev, "%s: enable sensor clk fail\n", __func__);
    goto probe_err_regulator_disable;
}
retval = ov2775_power_on(sensor);
if (retval < 0) {
    dev_err(dev, "%s: sensor power on fail\n", __func__);
    goto probe_err_regulator_disable;
}
ov2775_reset(sensor);
ov2775_read_reg(sensor, 0x300a, &reg_val);
chip_id |= reg_val << 8;
ov2775_read_reg(sensor, 0x300b, &reg_val);
chip_id |= reg_val;
if (chip_id != 0x2770) {
    pr_warn("camera ov2775 is not found\n");
    retval = -ENODEV;
    goto probe_err_power_off;
}
sd = &sensor->subdev;
v4l2_i2c_subdev_init(sd, client, &ov2775_subdev_ops);
sd->flags |= V4L2_SUBDEV_FL_HAS_DEVNODE;
sd->dev = &client->dev;
sd->entity.ops = &ov2775_sd_media_ops;
sd->entity.function = MEDIA_ENT_F_CAM_SENSOR;
sensor->pads[OV2775_SENS_PAD_SOURCE].flags =
MEDIA_PAD_FL_SOURCE;
retval = media_entity_pads_init(&sd->entity,
    OV2775_SENS_PADS_NUM,
    sensor->pads);
if (retval < 0)
    goto probe_err_power_off;
retval = v4l2_async_register_subdev_sensor_common(sd);
if (retval < 0) {
    dev_err(&client->dev, "%s--Async register failed, ret=%d\n",
        __func__, retval);
    goto probe_err_free_entity;
}
memcpy(&sensor->cur_mode, &pov2775_mode_info[0],
    sizeof(struct vvcam_mode_info_s));
mutex_init(&sensor->lock);
pr_info("%s camera mipi ov2775, is found\n", __func__);
return 0;
probe_err_free_entity:
media_entity_cleanup(&sd->entity);
probe_err_power_off:
ov2775_power_off(sensor);
probe_err_regulator_disable:
ov2775_regulator_disable(sensor);
return retval;
}

```

### 3. Implement the v4l2\_subdev\_ops data structure.

```

static struct v4l2_subdev_video_ops ov2775_subdev_video_ops =
{
    .s_stream = ov2775_s_stream,
};
static const struct v4l2_subdev_pad_ops ov2775_subdev_pad_ops
= {

```

```

.enum_mbus_code = ov2775_enum_mbus_code,
.set_fmt = ov2775_set_fmt,
.get_fmt = ov2775_get_fmt,
};
static struct v4l2_subdev_core_ops ov2775_subdev_core_ops = {
.s_power = ov2775_s_power,
.ioctl = ov2775_priv_ioctl,
};
static struct v4l2_subdev_ops ov2775_subdev_ops = {
.core = &ov2775_subdev_core_ops,
.video = &ov2775_subdev_video_ops,
.pad = &ov2775_subdev_pad_ops,
};

```

#### 4. Implement the sensor private IOCTL.

```

static long ov2775_priv_ioctl(struct v4l2_subdev *sd,
                             unsigned int cmd,
                             void *arg)
{
    struct i2c_client *client = v4l2_get_subdevdata(sd);
    struct ov2775 *sensor = client_to_ov2775(client);
    long ret = 0;
    struct vvcam_sccb_data_s sensor_reg;
    mutex_lock(&sensor->lock);
    switch (cmd) {
    case VVSENSORIOC_S_POWER:
        ret = 0;
        break;
    case VVSENSORIOC_S_CLK:
        ret = 0;
        break;
    case VVSENSORIOC_G_CLK:
        ret = ov2775_get_clk(sensor, arg);
        break;
    case VVSENSORIOC_RESET:
        ret = 0;
        break;
    case VIDIOC_QUERYCAP:
        ret = ov2775_query_capability(sensor, arg);
        break;
    case VVSENSORIOC_QUERY:
        ret = ov2775_query_supports(sensor, arg);
        break;
    case VVSENSORIOC_G_CHIP_ID:
        ret = ov2775_get_sensor_id(sensor, arg);
        break;
    case VVSENSORIOC_G_RESERVE_ID:
        ret = ov2775_get_reserve_id(sensor, arg);
        break;
    case VVSENSORIOC_G_SENSOR_MODE:
        ret = ov2775_get_sensor_mode(sensor, arg);
        break;
    case VVSENSORIOC_S_SENSOR_MODE:
        ret = ov2775_set_sensor_mode(sensor, arg);
        break;
    case VVSENSORIOC_S_STREAM:
        ret = ov2775_s_stream(&sensor->subdev, *(int *)arg);
        break;
    case VVSENSORIOC_WRITE_REG:
        ret = copy_from_user(&sensor_reg, arg,

```

```
        sizeof(struct vvcam_sccb_data_s));
    ret |= ov2775_write_reg(sensor, sensor_reg.addr,
        sensor_reg.data);
    break;
case VVSENSORIOC_READ_REG:
    ret = copy_from_user(&sensor_reg, arg,
        sizeof(struct vvcam_sccb_data_s));
    ret |= ov2775_read_reg(sensor, sensor_reg.addr,
        (u8 *)&sensor_reg.data);
    ret |= copy_to_user(arg, &sensor_reg,
        sizeof(struct vvcam_sccb_data_s));
    break;
case VVSENSORIOC_S_LONG_EXP:
    ret = ov2775_set_lexp(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_S_EXP:
    ret = ov2775_set_exp(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_S_VSEXP:
    ret = ov2775_set_vsexp(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_S_LONG_GAIN:
    ret = ov2775_set_lgain(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_S_GAIN:
    ret = ov2775_set_gain(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_S_VSGAIN:
    ret = ov2775_set_vsgain(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_S_FPS:
    ret = ov2775_set_fps(sensor, *(u32 *)arg);
    break;
case VVSENSORIOC_G_FPS:
    ret = ov2775_get_fps(sensor, (u32 *)arg);
    break;
case VVSENSORIOC_S_HDR_RADIO:
    ret = ov2775_set_ratio(sensor, arg);
    break;
case VVSENSORIOC_S_BLC:
    ret = ov2775_set_blc(sensor, arg);
    break;
case VVSENSORIOC_S_WB:
    ret = ov2775_set_wb(sensor, arg);
    break;
case VVSENSORIOC_G_EXPAND_CURVE:
    ret = ov2775_get_expand_curve(sensor, arg);
    break;
case VVSENSORIOC_S_TEST_PATTERN:
    ret = ov2775_set_test_pattern(sensor, arg);
    break;
default:
    break;
}
mutex_unlock(&sensor->lock);
return ret;
}
```

### 2.6.2.3 Create Sensor ISI API in ISI Layer

1. Typedef your `<sensor>_Context_t` for the sensor handle.

```
typedef struct OV2755_Context_s
{
    IsiSensorContext_t IsiCtx;
    struct vvcam_mode_info_s CurMode;
    IsiSensorAeInfo_t AeInfo;
    IsiSensorIntTime_t IntTime;
    uint32_t LongIntLine;
    uint32_t IntLine;
    uint32_t ShortIntLine;
    IsiSensorGain_t SensorGain;
    uint32_t minAfps;
    uint64_t AEStartExposure;
} OV2775_Context_t;
```

2. Implement `IsiCamDrvConfig` for the sensor library callback function. Define the [IsiCamDrvConfig\\_s](#) data structure. Data members defined in this data structure include the sensor ID (CameraDriverID) and the function pointer to the [IsiSensor](#) data structure. Using the address of the `IsiCamDrvConfig_t` structure, the driver can then access the sensor API attached to the function pointer.

```
IsiCamDrvConfig_t IsiCamDrvConfig = {
    .CameraDriverID = 0x2770,
    .pIsiHalQuerySensor = <sensor>_IsiHalQuerySensorIss,
    .pfIsiGetSensorIss = <sensor>_IsiGetSensorIss
}
```

**Note:**

- *IsiCamDrvConfig is defined in file: `units/isi/drv/<sensor>/source/<sensor>.c`.*
- *`<sensor>_IsiHalQuerySensorIss()` uses the IOCTL command `VVSENSORIOC_QUERY` to get all the modes supported by `<sensor>`.*

`<sensor>_IsiGetSensorIss()` can initialize the [IsiSensor](#) data structure. It is called by an upper-level application described in the [ISS Sensor Driver User Space Flow](#) section. Then the application can get the address of all the callback functions. `<sensor>_IsiGetSensorIss` is defined as follows.

```
RESULT <sensor>_IsiGetSensorIss(IsiSensor_t *pIsiSensor)
...
pIsiSensor->pIsiCreateSensorIss = <sensor>_IsiCreateSensorIss;
pIsiSensor->pIsiReleaseSensorIss = <sensor>_IsiReleaseSensorIss;
pIsiSensor->pIsiRegisterReadIss = <sensor>_IsiRegisterReadIss;
pIsiSensor->pIsiRegisterWriteIss = <sensor>_IsiRegisterWriteIss;
pIsiSensor->pIsiGetSensorModeIss = <sensor>_IsiGetSensorModeIss;
...
};
```

**Note:** It is described in the [Sensor API Reference](#) section.

### 2.6.3 Native HDR Mode Porting

For native HDR mode, two-exposure HDRs and three-exposure sensor HDRs are supported with the following caveats:

1. When a new native HDR mode is added, `hdr_mode` must be set to `SENSOR_MODE_HDR_NATIVE`, and `stitching_mode` must be set to the stitching mode corresponding to the sensor in use.

```
static struct vvcam_mode_info_s pov2775_mode_info[] = {
...
{
    .index          = 2,
    .width          = 1920,
    .height         = 1080,
    .hdr_mode       = SENSOR_MODE_HDR_NATIVE,
    .stitching_mode = SENSOR_STITCHING_DUAL_DCG_NOWAIT,
    .bit_width      = 12,
    .data_compress  = {
        .enable = 1,
        .x_bit  = 16,
        .y_bit  = 12,
    },
    .bayer_pattern  = BAYER_BGGR,
    .ae_info = {
        .def_frm_len_lines      = 0x466,
        .curr_frm_len_lines     = 0x466,
        .one_line_exp_time_ns   = 59167,
        .max_integration_line    = 0x466 - 4,
        .min_integration_line    = 1,
        .max_long_again         = 8 * 1024 * DCG_CONVERSION_GAIN,
        .min_long_again         = 1 * 1024 * DCG_CONVERSION_GAIN,
        .max_long_dgain         = 4 * 1024,
        .min_long_dgain         = 2 * 1024,
        .max_again              = 8 * 1024,
        .min_again              = 2 * 1024,
        .max_dgain              = 4 * 1024,
        .min_dgain              = 1.5 * 1024,
        .start_exposure         = 3 * 400 * 1024,
        .gain_step              = 4,
        .cur_fps                 = 30 * 1024,
        .max_fps                 = 30 * 1024,
        .min_fps                 = 5 * 1024,
        .min_afps                = 5 * 1024,
        .hdr_ratio               = {
            .ratio_l_s = 8 * 1024,
            .ratio_s_vs = 8 * 1024,
            .accuracy  = 1024,
        },
        .int_update_delay_frm = 1,
        .gain_update_delay_frm = 1,
    },
    .mipi_info = {
        .mipi_lane = 4,
    },
    .preg_data = ov2775_1080p_native_hdr_regs,
    .reg_data_count =
        ARRAY_SIZE(ov2775_1080p_native_hdr_regs),
    },
};
```

2. Set the Native HDR default stitching ratio for two-exposure, use `ratio_s_vs` (long/short) as the stitching ratio for three-exposure, and set both `ratio_l_s` (long/normal) and `ratio_s_vs` (normal/short).

```
static struct vvcam_mode_info_s pov2775_mode_info[] = {
...
{
    .index          = 2,
    .width          = 1920,
    .height         = 1080,
    .hdr_mode       = SENSOR_MODE_HDR_NATIVE,
    .stitching_mode = SENSOR_STITCHING_DUAL_DCG_NOWAIT,
    .bit_width      = 12,
    .data_compress  = {
        .enable = 1,
        .x_bit  = 16,
        .y_bit  = 12,
    },
    .bayer_pattern  = BAYER_BGGR,
    .ae_info = {
        .def_frm_len_lines      = 0x466,
        .curr_frm_len_lines     = 0x466,
        .one_line_exp_time_ns   = 59167,
        .max_integration_line    = 0x466 - 4,
        .min_integration_line    = 1,
        .max_long_again         = 8 * 1024 * DCG_CONVERSION_GAIN,
        .min_long_again         = 1 * 1024 * DCG_CONVERSION_GAIN,
        .max_long_dgain         = 4 * 1024,
        .min_long_dgain         = 2 * 1024,
        .max_again              = 8 * 1024,
        .min_again              = 2 * 1024,
        .max_dgain              = 4 * 1024,
        .min_dgain              = 1.5 * 1024,
        .start_exposure         = 3 * 400 * 1024,
        .gain_step              = 4,
        .cur_fps                 = 30 * 1024,
        .max_fps                 = 30 * 1024,
        .min_fps                 = 5 * 1024,
        .min_afps                = 5 * 1024,
        .hdr_ratio               = {
            .ratio_l_s = 8 * 1024,
            .ratio_s_vs = 8 * 1024,
            .accuracy = 1024,
        },
        .int_update_delay_frm = 1,
        .gain_update_delay_frm = 1,
    },
    .mipi_info = {
        .mipi_lane = 4,
    },
    .preg_data = ov2775_1080p_native_hdr_regs,
    .reg_data_count =
        ARRAY_SIZE(ov2775_1080p_native_hdr_regs),
    },
};
```

3. Generally, native HDR performs data compression at the sensor end, and the decompression function of the ISP module must be enabled. Setting `data_compress.enable = 1` means the sensor data has been compressed, and



the data is compressed from `x_bit` to `y_bit`. The decompression curve API must be implemented as described in the [Sensor Compand Curve](#) section.

```
static struct vvcam_mode_info_s pov2775_mode_info[] = {
...
{
    .index          = 2,
    .width          = 1920,
    .height         = 1080,
    .hdr_mode       = SENSOR_MODE_HDR_NATIVE,
    .stitching_mode = SENSOR_STITCHING_DUAL_DCG_NOWAIT,
    .bit_width      = 12,
    .data_compress  = {
        .enable = 1,
        .x_bit  = 16,
        .y_bit  = 12,
    },
    .bayer_pattern  = BAYER_BGGR,
    .ae_info = {
        .def_frm_len_lines      = 0x466,
        .curr_frm_len_lines     = 0x466,
        .one_line_exp_time_ns   = 59167,
        .max_integration_line    = 0x466 - 4,
        .min_integration_line    = 1,
        .max_long_again         = 8 * 1024 * DCG_CONVERSION_GAIN,
        .min_long_again         = 1 * 1024 * DCG_CONVERSION_GAIN,
        .max_long_dgain         = 4 * 1024,
        .min_long_dgain         = 2 * 1024,
        .max_again              = 8 * 1024,
        .min_again              = 2 * 1024,
        .max_dgain               = 4 * 1024,
        .min_dgain               = 1.5 * 1024,
        .start_exposure         = 3 * 400 * 1024,
        .gain_step              = 4,
        .cur_fps                 = 30 * 1024,
        .max_fps                 = 30 * 1024,
        .min_fps                 = 5 * 1024,
        .min_afps                = 5 * 1024,
        .hdr_ratio               = {
            .ratio_l_s = 8 * 1024,
            .ratio_s_vs = 8 * 1024,
            .accuracy  = 1024,
        },
    },
    .int_update_delay_frm = 1,
    .gain_update_delay_frm = 1,
},
    .mipi_info = {
        .mipi_lane = 4,
    },
    .preg_data = ov2775_1080p_native_hdr_regs,
    .reg_data_count =
        ARRAY_SIZE(ov2775_1080p_native_hdr_regs),
    },
};
```

4. Native HDR BLS and WB usually need to be done on the sensor side, so it is necessary to realize the sensor WB and BLS interface and configure ISP to use sensor BLS and WB as described in [Section 2.6.5](#) section.

## 2.6.4 Sensor Compand Curve

In the [vvcam\\_mode\\_info\\_t](#) data structure, the [sensor\\_data\\_compress\\_t](#) data structure describes whether the sensor data is compressed or not. If the sensor data is compressed, the [sensor\\_data\\_compress\\_t](#) data structure describes the data compression type.

### Note:

- The maximum bit width for the expand module is 20 bits.
- To remove the expand module, set `data_compress.enable = 0`.
- The compand and decompression curves vary by sensors. For example, the curves of OV2775 are different from those of OS08A20.

Example:

For OV2775 native HDR, sensor data is compressed from 16 bits to 12 bits. So, `x_bit=16` and `y_bit=12`.

It determines the type of decompression curve used by the compand module.

```
{
    .index = 2,
    .width = 1920,
    .height = 1080,
    .fps = 30,
    .hdr_mode = SENSOR_MODE_HDR_NATIVE,
    .bit_width = 12,
    .data_compress.enable = 1,
    .data_compress.x_bit = 16,
    .data_compress.y_bit = 12,
    .bayer_pattern = BAYER_BGGR,
    .ae_info = {
        .DefaultFrameLengthLines = 0x466,
        .one_line_exp_time_ns = 59167,
        .max_interrgation_time = 0x466 - 2,
        .min_interrgation_time = 1,
        .gain_accuracy = 1024,
        .max_gain = 21 * 1024,
        .min_gain = 3 * 1024,
    },
    .preg_data = ov2775_1080p_native_hdr_regs,
    .reg_data_count = ARRAY_SIZE(ov2775_1080p_native_hdr_regs),
}
```

ISP decompresses according to the specified compression method. If the sensor is compressed from 16-bit to 12-bit, the compand module calls the `<sensor>_get_expand_curve()` function to get the 12-bit to 16-bit expand curve as defined in the [sensor\\_expand\\_curve\\_s](#) data structure.

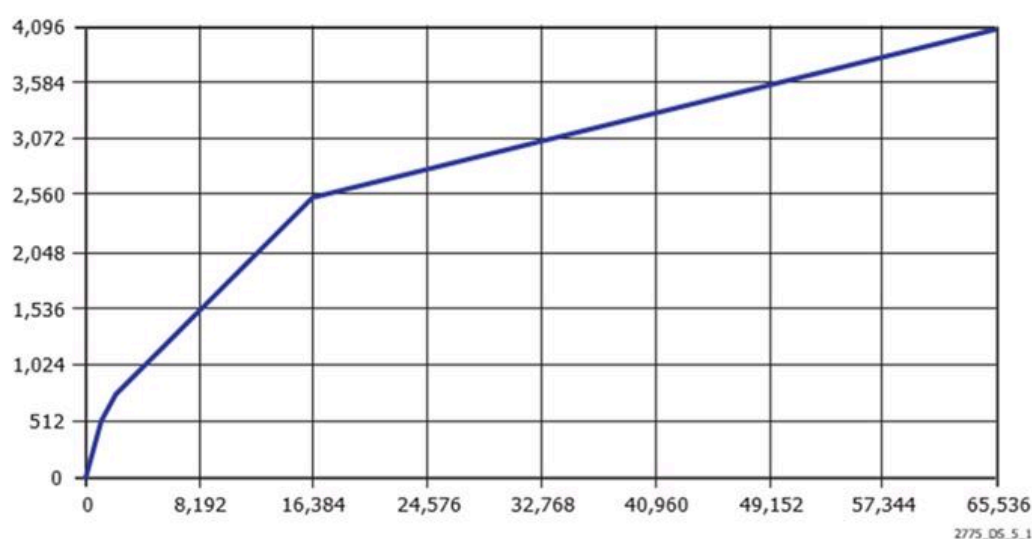
See below the limitations of the expand curve.

```
(1 << pexpand_curve->expand_px[i]) =
pexpand_curve->expand_x_data[i+1] - pexpand_curve-
>expand_x_data[i]
```

For example, OV2775 expand curve.

The OV2775 has a data compression from 16-bit to 12-bit by a 4-piece piece-wise linear (PWL) curve. The following formula defines the curve and is shown in the following figure.

$$y_{out\_12b} = \begin{cases} \frac{y_{in\_16b}}{2}, & y_{in\_16b} < 1024 \\ \frac{y_{in\_16b}}{4} + 256, & 1024 \leq y_{in\_16b} < 2048 \\ \frac{y_{in\_16b}}{8} + 512, & 2048 \leq y_{in\_16b} < 16384 \\ \frac{y_{in\_16b}}{32} + 2048, & y_{in\_16b} \geq 16384 \end{cases}$$



**Figure 8. 16-bit to 12-bit PWL compression**

The backend processor can decompress 12-bit data to 16-bit data using the following formula.

$$y_{out\_16b} = \begin{cases} 2 \times y_{in\_12b} & y_{in\_12b} < 512 \\ 4 \times (y_{in\_12b} - 256), & 512 \leq y_{in\_12b} < 768 \\ 8 \times (y_{in\_12b} - 512), & 768 \leq y_{in\_12b} < 2560 \\ 32 \times (y_{in\_12b} - 2048), & y_{in\_12b} \geq 2560 \end{cases}$$

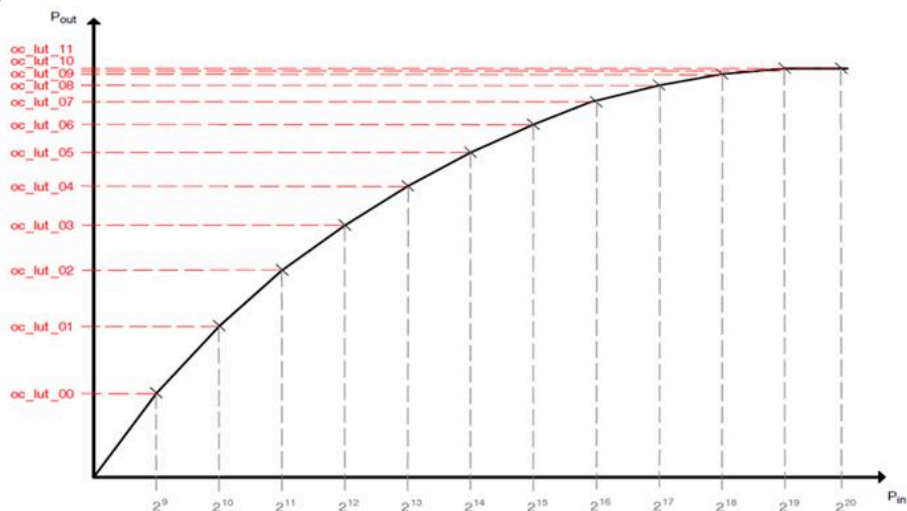
```
int ov2775_get_expand_curve(struct ov2775 *sensor,  
    sensor_expand_curve_t* pexpand_curve)  
{  
    int i;  
    if ((pexpand_curve->x_bit) == 12 && (pexpand_curve->y_bit ==  
        16))  
    {  
        uint8_t expand_px[64] = {6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,  
            6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
```

```

6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6};
memcpy(pexpand_curve->expand_px,expand_px,sizeof(expand_px));
pexpand_curve->expand_x_data[0] = 0;
pexpand_curve->expand_y_data[0] = 0;
for(i = 1; i < 65; i++)
{
pexpand_curve->expand_x_data[i] =
(1 << pexpand_curve->expand_px[i-1]) +
pexpand_curve->expand_x_data[i-1];
if (pexpand_curve->expand_x_data[i] < 512)
{
pexpand_curve->expand_y_data[i] =
pexpand_curve->expand_x_data[i] << 1;
}
else if (pexpand_curve->expand_x_data[i] < 768)
{
pexpand_curve->expand_y_data[i] =
(pexpand_curve->expand_x_data[i] - 256) << 2;
}
else if (pexpand_curve->expand_x_data[i] < 2560)
{
pexpand_curve->expand_y_data[i] =
(pexpand_curve->expand_x_data[i] - 512) << 3;
}
else
{
pexpand_curve->expand_y_data[i] =
(pexpand_curve->expand_x_data[i] - 2048) << 5;
}
}
return 0;
}
return (-1);
}
ar0820 20-bit to12-bit as 16-bit output:

```

20-bit Input:



20 → 12-bit	
0x2000	1:1
0x4000	
0x8000	
0x8200	1:64
0x8600	
0x8E00	
0x9E00	
0xBE00	
0xC200	1:1024
0xCA00	
0xDA00	
0xFA00	

**Figure 9. 20-bit to 12-bit as 16-bit output**

The automatic values of the knee-points can be read back from the `oc_lut_XX` registers but cannot be changed (writes to the `oc_lut_XX` registers are ignored). All the knee-point registers are MSB-aligned. For example, a programmed value of 0x2000 acts as 0x200 when the output is 12-bit data and acts as 0x2000 when the output is 16-bit data.

The expand curve is defined as follows:

```
expand_px[64] = {13, 13, 14, 9, 10, 11, 12, 13,  
10, 11, 12, 13, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0};  
expand_x_data[65]  
    ={0,0x2000,0x4000,0x8000,0x8200,0x8600,0x8e00,0x9e00,0xbe00,  
0xc200,0xca00,0xda00,0xfa00,0xfa01,0xfa02,0xfa03,0xfa04,  
0xfa05,0xfa06,0xfa07,0xfa08,0xfa09,0xfa0a,0xfa0b,0xfa0c,  
0xfa0d,0xfa0e,0xfa0f,0xfa10,0xfa11,0xfa12,0xfa13,0xfa14,  
0xfa15,0xfa16,0xfa17,0xfa18,0xfa19,0xfa1a,0xfa1b,0xfa1c,  
0xfa1d,0xfa1e,0xfa1f,0xfa20,0xfa21,0xfa22,0xfa23,0xfa24,  
0xfa25,0xfa26,0xfa27,0xfa28,0xfa29,0xfa2a,0xfa2b,0xfa2c,  
0xfa2d,0xfa2e,0xfa2f,0xfa30,0xfa31,0xfa32,0xfa33,0xfa34};  
expand_y_data[65] = {0x00,  
0x200, 0x400, 0x800, 0x1000, 0x2000, 0x4000, 0x8000, 0x10000,  
0x20000, 0x40000, 0x80000, 0x100000, 0x100000, 0x100000,  
0x100000,0x100000,  
0x100000,0x100000,0x100000,0x100000, 0x100000, 0x100000,  
0x100000,0x100000,  
0x100000,0x100000,0x100000,0x100000, 0x100000, 0x100000,  
0x100000,0x100000,  
0x100000,0x100000,0x100000,0x100000, 0x100000, 0x100000,  
0x100000,
```

```
0x100000,0x100000,0x100000,0x100000, 0x100000, 0x100000,
0x100000,0x100000,
0x100000,0x100000,0x100000,0x100000, 0x100000, 0x100000,
0x100000,0x100000,
0x100000,0x100000,0x100000,0x100000, 0x100000, 0x100000,
0x100000,0x100000};
```

**Note:** Sensor data is 16-bit output, so `data_compress` must set `x_bit = 20` and `y_bit = 16`.

```
.data_compress = {
.enable = 1,
.x_bit = 20,
.y_bit = 16,
},
```

## 2.6.5 Sensor White Balance and Black Level Correction (BLC)

ISP AWB is used in normal mode. In native HDR mode, black level and white balance calibration should be done before the image synthesis at the sensor.

To enable the WB mode of the sensor, an interface must be provided to set the AWB mode to `ISI_SENSOR_AWB_MODE_SENSOR`. In this `ISI_SENSOR_AWB_MODE_SENSOR` mode, ISP does not perform white balance and black level reduction. Set the sensor for black level and white balance calibration using [VVSensorIOC\\_S\\_WB](#) and [VVSensorIOC\\_S\\_BLC](#).

Example:

```
static RESULT OV2775_IsiGetSensorAWBModeIss(IsiSensorHandle_t
handle,
IsiSensorAwbMode_t *pawbmode)
{
OV2775_Context_t *pOV2775Ctx = (OV2775_Context_t *) handle;
if (pOV2775Ctx == NULL || pOV2775Ctx->IsiCtx.HalHandle == NULL)
{
return RET_NULL_POINTER;
}
if (pOV2775Ctx->SensorMode.hdr_mode == SENSOR_MODE_HDR_NATIVE)
{
*pawbmode = ISI_SENSOR_AWB_MODE_SENSOR;
}
else {
*pawbmode = ISI_SENSOR_AWB_MODE_NORMAL;
}
return RET_SUCCESS;
}
```

## 2.7 Camera Timing Issue Solution

One of the following situations may occur for the dual sensor configuration:

- The second sensor FPS is different from the first sensor.
- The second sensor image displays jitter.
- The second sensor initial brightness of the image changes from time to time.

**Solution:**

The timing may be adjusted to solve this problem in the following example (for example, ov2775):

1. Read the sensor register as shown in the figure below.
2. Calculate the MIPI clock and Pclk.
3. Reduce the size of the sensor PLL multiplier slightly until the image returns to normal and the FPS of the two sensors is the same.
4. After the image returns to normal, adjust the FPS through sensor HTS/VTs.

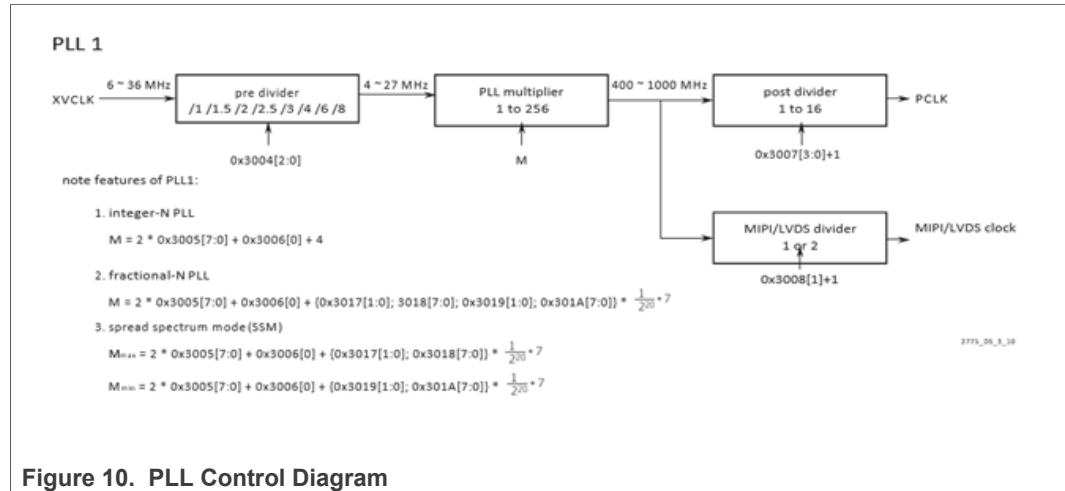


Figure 10. PLL Control Diagram

## 3 ISP Using V4L2 Interface

### 3.1 Overview

This document describes the ISP software Application Programming Interface (API) using Video For Linux 2. The ISP software V4L2 API controls the ISP hardware, sensor hardware, and its calibration data from the Linux standard API. The kernel V4L2 driver handles the API commands and requests from the V4L2 user application. It communicates to the ISP software stack and delivers image buffers to the V4L2 user application.

Currently, there are no deprecated functions in this API.

#### 3.1.1 Requirements/dependencies

- Linux environment is compatible with V4L2.

#### 3.1.2 Supported features

ISP features which are listed in [Table 7](#) are currently supported in the ISP V4L2 API.

Table 7. ISP features

Feature	Abbreviation
Auto Focus	AF
Auto Exposure	AE
Auto White Balance	AWB
Auto Video Stabilization	AVS

Table 7. ISP features...continued

Feature	Abbreviation
Black Level Subtraction	BLS
Chromatic Aberration Correction	CAC
Color Noise Reduction	CNR
Color Processing	CPROC
Demosaic	--
Defect Pixel Cluster Correction	DPCC
De-noising Pre-filter	DPF
High Dynamic Range	HDR
Image Effect	IE
Lens Shade Correction	LSC
Noise Reduce 2D	2DNR
Noise Reduce 3D	3DNR
Wide Dynamic Range	WDR

Sensor features: Additional functionality provided in future releases.

## 3.2 V4L2 API components

The ISP software V4L2 API is written in ANSI C++ code and is defined in the `v4l2/video/sub` folder. All commands are performed in the user space using an IOCTL interface which calls kernel space actions directly. The IOCTL control words are described in the [IOCTL Interface and Commands](#).

The ISP software V4L2 API components are defined in the following sections:

- Buffer API
- Event API
- Feature control API

### 3.2.1 IOCTL interface and commands

V4L2 provides Input and Output Control (IOCTL) interfaces to communicate directly with device drivers. [Table 8](#) lists key IOCTLs relevant to the ISP V4L2 software. Each IOCTL command corresponds to an operation function.

Table 8. Key video IOTCLs

IOCTL	Type	Description
VIDIOC_QUERYCAP	.vidioc_querycap	Query the capabilities of the driver, such as V4L2_CAP_STREAMING
VIDIOC_ENUM_FRAMESIZES	vidioc_enum_framesizes	Enum support resolution
VIDIOC_S_FMT	.vidioc_s_fmt_*	Set format information
VIDIOC_REQBUFS	.vidioc_reqbufs	Request buffers. Buffer types: DMA, MMAP, USER_PTR



Table 8. Key video IOCTLs...continued

IOCTL	Type	Description
VIDIOC_QBUF	.vidioc_qbuf	Enqueue buffer to kernel, then the driver fills this buffer
VIDIOC_QUERYBUF	.vidioc_querybuf	Get buffer information from the kernel and mmap
VIDIOC_DQBUF	.vidioc_dqbuf	De-queue the buffer from the kernel. User gets frame data
VIDIOC_STREAMON	.vidioc_streamon	Start stream
VIDIOC_STREAMOFF	.vidioc_streamoff	Close stream
VIDIOC_G_EXT_CTRL	.vidioc_g_ext_ctrls	Get feature control commands
VIDIOC_S_EXT_CTRL	.vidioc_s_ext_ctrls	Set feature control commands

### 3.2.2 IOCTL call flow

IOCTL call flow is described in [Figure 11](#) and the ISP reference code is based on this implementation.

This flow will be expanded in the future.

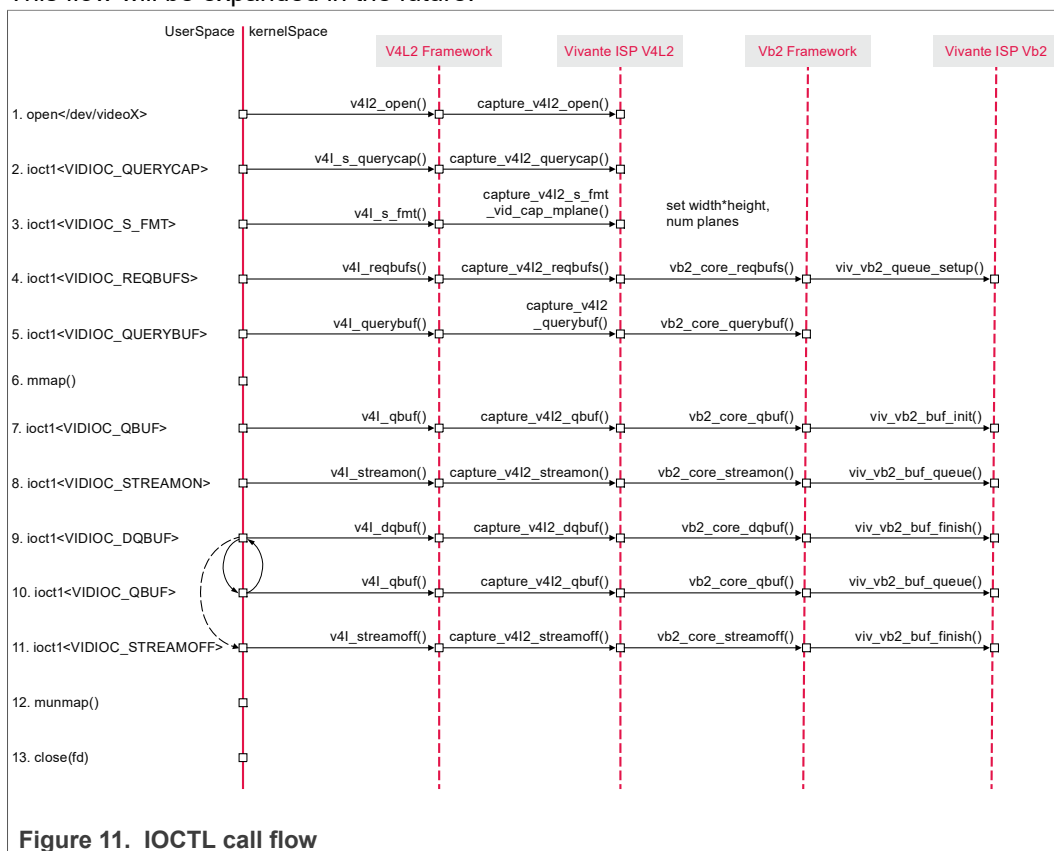


Figure 11. IOCTL call flow

### 3.2.3 Buffer API

A buffer contains data exchanged by the application and driver using memory mapping I/O. Only pointers to buffers are exchanged; the data itself is not copied. The primary intent of memory mapping is to map buffers in device memory into the address space of the application.

The V4L2 driver supports the following buffer IOCTLs:

- VIDIOC\_REQBUFS
- VIDIOC\_QUERYBUF
- VIDIOC\_QBUF
- VIDIOC\_DQBUF
- VIDIOC\_STREAMON
- VIDIOC\_STREAMOFF

In addition, the following functions are supported.

- `mmap()`
- `munmap()`
- `select()`
- `poll()`

#### 3.2.3.1 Buffer IOCTL control words

- VIDIOC\_REQBUFS  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/vidioc-reqbufs.html>
- VIDIOC\_QUERYBUF  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/vidioc-querybuf.html>
- VIDIOC\_QBUF  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/vidioc-qbuf.html>
- VIDIOC\_DQBUF  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/vidioc-qbuf.html>
- VIDIOC\_STREAMON  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/vidioc-streamon.html>
- VIDIOC\_STREAMOFF  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/vidioc-streamon.html>

#### 3.2.3.2 Buffer functions

- `mmap`  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/func-mmap.html>
- `munmap`  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/func-munmap.html>
- `poll`  
Link: <https://www.kernel.org/doc/html/v5.15/userspace-api/media/v4l/func-poll.html>

### 3.2.4 Event API

The V4L2 event interface provides a means for a user to get notified immediately on certain conditions taking place on a device.

To receive events, first the user must subscribe to an event using the `VIDIOC_SUBSCRIBE_EVENT` and the `VIDIOC_UNSUBSCRIBE_EVENT` IOCTLs. Once an event is subscribed, the events of subscribed types are de-queueable using the `VIDIOC_DQEVENT` IOCTL. Events may be unsubscribed using the `VIDIOC_UNSUBSCRIBE_EVENT` IOCTL. The information on de-queueable events is obtained by using `poll()` system calls on video devices. The V4L2 events use `POLLPRI` events on `poll` system calls.

The V4L2 driver supports the following event IOCTLs:

- `VIDIOC_SUBSCRIBE_EVENT`
- `VIDIOC_UNSUBSCRIBE_EVENT`
- `VIDIOC_DQEVENT`

In addition, the following function is supported.

- `poll()`

#### 3.2.4.1 Event IOCTL control words

- `VIDIOC_SUBSCRIBE_EVENT`  
Link: <http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-subscribe-event.html>
- `VIDIOC_UNSUBSCRIBE_EVENT`  
Link: <http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-subscribe-event.html>
- `VIDIOC_DQEVENT`  
Link: <http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-dqevent.html>

#### 3.2.4.2 Event functions

- `poll`  
Link: <http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/func-poll.html>

#### 3.2.4.3 Private event

The private event is an extension based on `V4L2_EVENT_PRIVATE_START`. It defines ID of the private event source, defines event data struct `knl_v4l2_event_data` based on struct `v4l2_event.u.data[64]`.

Private event type:

- `KNL_VIVCAM_V4L2_EVENT_TYPE`

ID:

- `KNL_VIVCAM_NOTIFY`

Struct definition:

- Struct `knl_v4l2_event_data`, 64 bytes.

Table 9. Private event

Structure member	Type	Description
<code>command</code>	unsigned int	Extension based on <code>V4L2_CID_PRIVATE_BASE</code>

Table 9. Private event...continued

Structure member	Type	Description
status	unsigned int	
session_id	unsigned int	
stream_id	unsigned int	
nop1	unsigned int	Reserved for future extensions
...	...	...
nop12	unsigned int	

### 3.2.5 Feature control API

The feature control API, uses JavaScript Object Notation (JSON) objects in user application threads and shares the objects directly with the daemon using share memory methods.

The ISP daemon sets ISPCore feature control words directly with the JSON parameters. In the user space and kernel space transfer, the `Json::Value` object is translated to a char string. Then, it is transferred between the user and kernel space as shown in Figure 12.

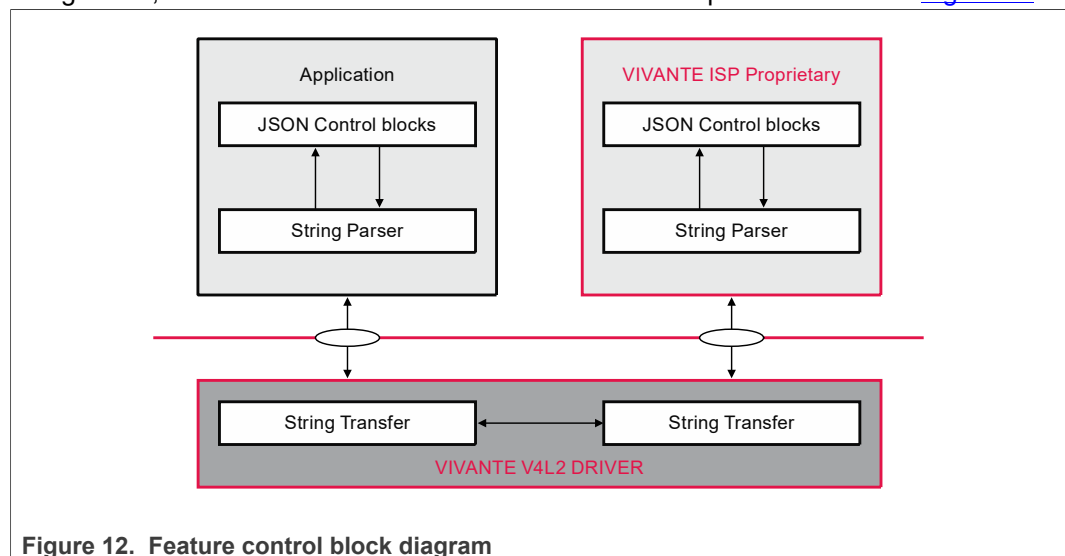


Figure 12. Feature control block diagram

#### 3.2.5.1 String parser

The JSON format used for the APIs and the string transfer can be handled using open source code.

For example:

1. `Json::Value` to char string:

```
String Json::Value::toStyledString(Json::Value)
```

2. char string to `Json::Value`:

```
Json::CharReaderBuilder::parse(const char* beginDoc,
                               const char* endDoc,
                               Value& root, bool
                               collectComments = true);
```

### 3.2.5.2 String transfer

All feature-related JSON-String entities are transferred using the following IOCTLs:

- `VIDIOC_G_EXT_CTRL`  
Link: <http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-g-ext-ctrls.html>
- `VIDIOC_S_EXT_CTRL`  
Link: <http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-g-ext-ctrls.html>

For a detailed example, refer to the code `appshell/vvext/vvext.cpp`.

The char string memory block exchange using the `v4l2_ext_control` struct, as shown in [Table 10](#).

**Table 10. v4l2\_ext\_control Structure**

v4l2_ext_control structure member	Type	Description
<code>id</code>	<code>__u32</code>	V4L2 ISP SW feature control words
<code>size</code>	<code>__u32</code>	String length
<code>reserved2[1]</code>	<code>__u32</code>	
<code>value</code>	union of <code>__s32</code>	
<code>value64</code>	union of <code>__s64</code>	
<code>string</code>	union of <code>char *</code>	String transfer pointer
<code>p_u8</code>	union of <code>__u8 *</code>	
<code>p_u16</code>	union of <code>__u16 *</code>	
<code>p_u32</code>	union of <code>__u32 *</code>	
<code>ptr</code>	union of <code>void*</code>	

### 3.2.5.3 Feature control words

Interface header file: `mediacontrol/include_api/ioctl_cmds.h`.

#### • IF\_AE\_G\_CFG

This macro definition is identical to the string "ae.g.cfg".

Description: Gets the configuration values for the Auto Exposure control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 11. Control words for IF\_AE\_G\_CFG**

Control Word	Description	Valid Values
<code>mode</code>	Configuration mode	1: Disabled evaluation 2: Fix evaluation 3: Adaptive evaluation
<code>damp.over</code>	Damping upper limit	[0.0 ... 1.0]
<code>damp.under</code>	Damping lower limit	[0.0 ... 1.0]
<code>set.point</code>	Set point	[0 ... 255]
<code>clm.tolerance</code>	Calculation accuracy	[0 ... 100]

#### • IF\_AE\_S\_CFG

This macro definition is identical to the string "ae.s.cfg".

Description: Sets the configuration values for the Auto Exposure control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 12. Control words for IF\_AE\_S\_CFG**

Control Word	Description	Valid Values
mode	Configuration mode	1: Disabled evaluation 2: Fix evaluation 3: Adaptive evaluation
damp.over	Damping upper limit	[0.0 ... 1.0]
damp.under	Damping lower limit	[0.0 ... 1.0]
set.point	Set point	[0 ... 255]
clm.tolerance	Calculation accuracy	[0 ... 100]

#### • IF\_AE\_G\_ECM

This macro definition is identical to the string "ae.g.ecm".

Description: Gets the ECM (Exposure Control Module) values for the Auto Exposure control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 13. Control words for IF\_AE\_G\_ECM**

Control word	Description	Valid Values
flicker.period	The flag of Auto Exposure flicker period	[0 ... 2] – 0: Flicker Period off – 1: 100 Hz – 2: 120 Hz
afps	Auto FPS control value	true false

#### • IF\_AE\_S\_ECM

This macro definition is identical to the string "ae.s.ecm".

Description: Sets the ECM (Exposure Control Module) values for the Auto Exposure control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 14. Control words for IF\_AE\_S\_ECM

Control word	Description	Valid Values
flicker.period	The flag of Auto Exposure flicker period	[0 ... 2] – 0: Flicker Period off – 1: 100 Hz – 2: 120 Hz
afps	Auto FPS control value	– true – false

- **IF\_AE\_G\_EN**

This macro definition is identical to the string "ae.g.en".

Description: Gets the enabled/disabled state of the Auto Exposure control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 15. Control words for IF\_AE\_G\_EN

Control word	Description	Valid Values
enable	The state of Auto Exposure	– true – false

- **IF\_AE\_S\_EN**

This macro definition is identical to the string "ae.s.en".

Description: Sets the enabled/disabled state of the Auto Exposure control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 16. Control words for IF\_AE\_S\_EN

Control word	Description	Valid Values
enable	Enable or disable Auto Exposure	– true – false

- **IF\_AE\_RESET**

This macro definition is identical to the string "ae.reset".

Description: Reset the Auto Exposure control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 17. Control words for IF\_AE\_RESET

Control word	Description	Valid Values
N/A	-	-

- **IF\_AF\_G\_CFG**

This macro definition is identical to the string "af.g.cfg".

Description: Gets the configuration of the Auto Focus control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 18. Control words for IF\_AF\_G\_CFG

Control word	Description	Valid Values
algorithm	Algorithm type	<ul style="list-style-type: none"> <li>– 1: full range</li> <li>– 2: adaptive range</li> <li>– 3: hill climbing</li> </ul>
oneshot	Trigger mode is one shot	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

• **IF\_AF\_S\_CFG**

This macro definition is identical to the string "af.s.cfg".

Description: Sets the configuration of the Auto Focus control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 19. Control words for IF\_AF\_S\_CFG

Control word	Description	Valid Values
algorithm	Algorithm type	<ul style="list-style-type: none"> <li>– 1: full range</li> <li>– 2: adaptive range</li> <li>– 3: hill climbing</li> </ul>
oneshot	Trigger mode is one shot	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

• **IF\_AF\_G\_EN**

This macro definition is identical to the string "af.g.en".

Description: Gets the enabled/disabled state of the Auto Focus control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 20. Control words for IF\_AF\_G\_EN

Control word	Description	Valid Values
enable	The state of the Auto Focus	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

• **IF\_AF\_S\_EN**

This macro definition is identical to the string "af.s.en".

Description: Sets the enabled/disabled state of the Auto Focus control.

## Parameters:

- `Json::Value &jRequest`



– Json::Value &jResponse

**Table 21. Control words for IF\_AF\_S\_EN**

Control word	Description	Valid Values
enable	Enable or disable Auto Focus	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

#### • IF\_AWB\_G\_CFG

This macro definition is identical to the string "awb.g.cfg".

Description: Gets the configuration of the Auto White Balance control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 22. Control Words for IF\_AWB\_G\_CFG**

Control word	Description	Valid Values
mode	AWB mode	<ul style="list-style-type: none"> <li>– 1: manual</li> <li>– 2: auto</li> </ul>
index	The index of calibration data in the database	[0 ... 32]
damping	Have damped data	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

#### • IF\_AWB\_S\_CFG

This macro definition is identical to the string "awb.s.cfg".

Description: Sets the mode and index of the Auto White Balance control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 23. Control Words for IF\_AWB\_S\_CFG**

Control word	Description	Valid Values
mode	AWB mode	<ul style="list-style-type: none"> <li>– 1: manual</li> <li>– 2: auto</li> </ul>
index	The index of calibration data in the database	[0 ... 32]
damping	Damping data	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

#### • IF\_AWB\_G\_EN

This macro definition is identical to the string "awb.g.en".

Description: Gets the enabled/disabled state of the Auto White Balance control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 24. Control words for IF\_AWB\_G\_EN**

Control word	Description	Valid Values
enable	The state of the AWB control	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

• **IF\_AWB\_S\_EN**

This macro definition is identical to the string "awb.s.en".

Description: Sets the enabled/disabled state of the Auto White Balance control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 25. Control words for IF\_AWB\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Auto White Balance	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

• **IF\_AWB\_RESET**

This macro definition is identical to the string "awb.reset".

Description: Resets the Auto White Balance control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 26. Control words for IF\_AWB\_RESET**

Control word	Description	Valid Values
N/A	-	-

• **IF\_AWB\_S\_MEASWIN**

This macro definition is identical to the string "awb.s.measwin".

Description: Sets measuring window of the Auto White Balance.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 27. Control words for IF\_AWB\_S\_MEASWIN**

Control word	Description	Valid Values
left	Measuring window left start position	<i>Sensor-specific</i>
top	Measuring window top start position	<i>Sensor-specific</i>
width	Measuring window width	<i>Sensor-specific</i>
height	Measuring window height	<i>Sensor-specific</i>

• **IF\_AVS\_G\_CFG**

This macro definition is identical to the string "avs.g.cfg".

Description: Gets the configuration values for the Auto Video Stabilization control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 28. Control words for IF\_AVS\_G\_CFG**

Control word	Description	Valid Values
use.params	AVS use params	true false
acceleration	AVS has acceleration	[0.0 ... 100.0]
base.gain	Base gain of AVS	[0.00 ... 1.00]
fall.off	AVS has fallen off	[0.00 ... 1.00]
num.itp.points	The number of ITP points	[1 ... 65536]
theta	Theta	[0.0 ... 1.0]

#### • IF\_AVS\_S\_CFG

This macro definition is identical to the string "avs.s.cfg".

Description: Sets the configuration values for the Auto Video Stabilization control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 29. Control words IF\_AVS\_S\_CFG**

Control word	Description	Valid Values
use.params	AVS use params	true false
acceleration	AVS has acceleration	[0.0 ... 100.0]
base.gain	Base gain of AVS	[0.00 ... 1.00]
fall.off	AVS has fallen off	[0.00 ... 1.00]
num.itp.points	The number of ITP points	[1 ... 65536]
theta	Theta	[0.0 ... 1.0]

#### • IF\_AVS\_G\_EN

This macro definition is identical to the string "avs.g.en".

Description: Gets the enabled/disabled state of the Auto Video Stabilization control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 30. Control words for IF\_AVS\_G\_EN**

Control word	Description	Valid Values
enable	The state of the AVS	– true – false

#### • IF\_AVS\_S\_EN

This macro definition is identical to the string "avs.s.en".

Description: Sets the enabled/disabled state of the Auto Video Stabilization control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 31. Control words for IF\_AVS\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables AVS	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

• **IF\_BLS\_G\_CFG**

This macro definition is identical to the string "bls.g.cfg".

Description: Gets the configuration values for the Black Level Subtraction control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 32. Control words for IF\_BLS\_G\_CFG**

Control word	Description	Valid Values
red	The red data information	Sensor-specific
green.r	The Gr data information	Sensor-specific
green.b	The Gb data information	Sensor-specific
blue	The blue data information	Sensor-specific

• **IF\_BLS\_S\_CFG**

This macro definition is identical to the string "bls.s.cfg".

Description: Sets the configuration values for the Black Level Subtraction control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 33. Control words for IF\_BLS\_S\_CFG**

Control word	Description	Valid Values
red	The red data information	Sensor-specific
green.r	The Gr data information	Sensor-specific
green.b	The Gb data information	Sensor-specific
blue	The blue data information	Sensor-specific

• **IF\_CAC\_G\_EN**

This macro definition is identical to the string "cac.g.en".

Description: Gets the enabled/disabled state of the Chromatic Aberration Correction control.

Parameters:

– Json::Value &jRequest  
 – Json::Value &jResponse

**Table 34. Control words for IF\_CAC\_G\_EN**

Control word	Description	Valid Values
enable	The state of the Chromatic Aberration Correction	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

- **IF\_CAC\_S\_EN**

This macro definition is identical to the string "cac.s.en".

Description: Sets the enabled/disabled state of the Chromatic Aberration Correction control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 35. Control words for IF\_CAC\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Chromatic Aberration Correction	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

- **IF\_CNR\_G\_CFG**

This macro definition is identical to the string "cnr.g.cfg".

Description: Gets the configuration values for the Chroma Noise Reduction control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 36. Control words for IF\_CNR\_G\_CFG**

Control Word	Description	Valid Values
tc1	The CNR threshold value of the Cb channel.	[0 ... 32767]
tc2	The CNR threshold value of the Cr channel.	[0 ... 32767]

- **IF\_CNR\_S\_CFG**

This macro definition is identical to the string "cnr.s.cfg".

Description: Sets the configuration values for the Chroma Noise Reduction control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 37. Control words for IF\_CNR\_S\_CFG**

Control word	Description	Valid Values
tc1	The CNR threshold value of the Cb channel.	[0 ... 32767]
tc2	The CNR threshold value of the Cr channel.	[0 ... 32767]

- **IF\_CNR\_G\_EN**

This macro definition is identical to the string "cnr.s.en".

**Description:**

Gets the enabled/disabled state of the Chroma Noise Reduction control.

**Parameters:**

- Json::Value &jRequest
- Json::Value &jResponse

Table 38. Control Words for IF\_CNR\_G\_EN

Control Word	Description	Valid Values
enable	The state of the Chroma Noise Reduction control	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

- **IF\_CNR\_S\_EN**

This macro definition is identical to the string "cnr.s.en".

**Description:**

Sets the enabled/disabled state of the Chroma Noise Reduction control.

**Parameters:**

- Json::Value &jRequest
- Json::Value &jResponse

Table 39. Control Words for IF\_CNR\_S\_EN

Control Word	Description	Valid Values
enable	Enables or disables Chroma Noise Reduction control	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

- **IF\_CPROC\_G\_CFG**

This macro definition is identical to the string "cproc.g.cfg".

Description: Gets the configuration values for the Color Processing control.

**Parameters:**

- Json::Value &jRequest
- Json::Value &jResponse

Table 40. Control words for IF\_CPROC\_G\_CFG

Control Word	Description	Valid Values
brightness	Brightness value	[-128 ... 127] – Default: -15 – Greater than 1: increases brightness; – less than 1 decreases brightness.
chroma.out	CPROC chrominance pixel clipping range at output	– 1: CbCr_out clipping range [16 ... 240] according to ITU-R BT.601 standard – 2: full UV_out clipping range [0 ... 255]
contrast	Contrast value	[0 ... 1.9921875] – Default: 1.1 – Greater than 1: increases contrast; – less than 1 decreases contrast.

Table 40. Control words for IF\_CPROC\_G\_CFG...continued

Control Word	Description	Valid Values
hue	Hue value	[-90 ... 89] – Default: 0 – Greater than 1: increases hue; – less than 1 decreases hue.
luma.in	CPROC luminance input range (offset processing)	– 1: Y_in range [64 ... 940] according to ITU-R BT.601 standard; offset of 64 is subtracted from Y_in – 2: Y_in full range [0 ... 1023]; no offset is subtracted from Y_in
luma.out	CPROC luminance output clipping range	– 1: Y_out clipping range [16 ... 235]; offset of 16 is added to Y_out according to ITU-R BT.601 standard – 2: Y_out clipping range [0 ... 255]; no offset is added to Y_out
saturation	Saturation value	[0 ... 1.9921875] – Default: 1 – Greater than 1: increases saturation; – less than 1 decreases saturation.

- **IF\_CPROC\_S\_CFG**

This macro definition is identical to the string "cproc.s.cfg".

Description: Sets the configuration values for the Color Processing control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 41. Control words for IF\_CPROC\_S\_CFG

Control Word	Description	Valid Values
brightness	Brightness value	[-128 ... 127] – Default: -15 – Greater than 1: increases brightness; – less than 1 decreases brightness.
chroma.out	CPROC chrominance pixel clipping range at output	– 1: CbCr_out clipping range [16 ... 240] according to ITU-R BT.601 standard – 2: full UV_out clipping range [0 ... 255]
contrast	Contrast value	[0 ... 1.9921875] – Default: 1.1 – Greater than 1: increases contrast; – less than 1 decreases contrast.

Table 41. Control words for IF\_CPROC\_S\_CFG...continued

Control Word	Description	Valid Values
hue	Hue value	[-90 ... 89] – Default: 0 – Greater than 1: increases hue; – less than 1 decreases hue.
luma.in	CPROC luminance input range (offset processing)	– 1: Y_in range [64 ... 940] according to ITU-R BT.601 standard; offset of 64 is subtracted from Y_in – 2: Y_in full range [0 ... 1023]; no offset is subtracted from Y_in
luma.out	CPROC luminance output clipping range	– 1: Y_out clipping range [16 ... 235]; offset of 16 is added to Y_out according to ITU-R BT.601 standard – 2: Y_out clipping range [0 ... 255]; no offset is added to Y_out
saturation	Saturation value	[0 ... 1.9921875] – Default: 1 – Greater than 1: increases saturation; – less than 1 decreases saturation.

#### • IF\_CPROC\_G\_EN

This macro definition is identical to the string "cproc.g.en".

Description: Gets the enabled/disabled state of the Color Processing control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 42. Control words for IF\_CPROC\_G\_EN

Control word	Description	Valid Values
enable	The state of the CPROC	– true – false

#### • IF\_CPROC\_S\_EN

This macro definition is identical to the string "cproc.s.en".

Description: Sets the enabled/disabled state of the Color Processing control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 43. Control words for IF\_CPROC\_S\_EN

Control word	Description	Valid Values
enable	Enable or disable CPROC	– true – false

#### • IF\_DEMOSAIC\_G\_CFG

This macro definition is identical to the string "dmsc.g.cfg".

Description: Gets the configuration values for the Demosaic control.

Parameters:

- Json::Value &jRequest



– Json::Value &jResponse

**Table 44. Control words for IF\_DEMOSAIC\_G\_CFG**

Control word	Description	Valid Values
mode	Demosaic mode	1: normal 2: bypass
threshold	Demosaic threshold	[0..255]

• **IF\_DEMOSAIC\_S\_CFG**

This macro definition is identical to the string "dmsc.s.cfg".

Description: Sets the configuration values for the Demosaic control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 45. Control words for IF\_DEMOSAIC\_S\_CFG**

Control word	Description	Valid Values
mode	Demosaic mode	– 1: normal – 2: bypass
threshold	Demosaic threshold	[0..255]

• **IF\_DEMOSAIC\_G\_EN**

This macro definition is identical to the string "demosaic.g.en".

Description: Gets the enabled/disabled state of the Demosaic control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 46. Control words for IF\_DEMOSAIC\_G\_EN**

Control word	Description	Valid Values
enable	The state of the Demosaic control	– true – false

• **IF\_DEMOSAIC\_S\_EN**

This macro definition is identical to the string "demosaic.s.en".

Description: Sets the enabled/disabled state of the Demosaic control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 47. Control words for IF\_DEMOSAIC\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Demosaic	– true – false

• **IF\_DPCC\_G\_EN**

This macro definition is identical to the string "dpcc.g.en".

Description: Gets the enabled/disabled state of the Defect Pixel Cluster Correction control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 48. Control words for IF\_DPCC\_G\_EN

Control word	Description	Valid Values
enable	The state of the Defect Pixel Cluster Correction	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

## • IF\_DPCC\_S\_EN

This macro definition is identical to the string "dpcc.s.en".

Description: Sets the enabled/disabled state of the Defect Pixel Cluster Correction control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 49. Control words for IF\_DPCC\_S\_EN

Control word	Description	Valid Values
enable	Enables or disables Defect Pixel Cluster Correction	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

## • IF\_DPF\_G\_CFG

This macro definition is identical to the string "dpf.g.cfg".

Description: Gets the configuration values for the De-noising Pre-Filter control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 50. Control words for IF\_DPF\_G\_CFG

Control word	Description	Valid Values
gradient	Gradient value for dynamic strength calculation	[0 ... 128]
offset	Offset value for dynamic strength calculation	[-128 ... 128]
min	Upper bound for dynamic strength calculation	[0 ... 128]
div	Division factor for dynamic strength calculation	[0 ... 64]
sigma.green	Sigma value for green pixel	[1 ... 255]
sigma.red.blue	Sigma value for red/blue pixel	[1 ... 255]

## • IF\_DPF\_S\_CFG

This macro definition is identical to the string "dpf.s.cfg".

Description: Sets the configuration values for the De-noising Pre-Filter control.

## Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 51. Control words for IF\_DPF\_S\_CFG

Control word	Description	Valid Values
gradient	Gradient value for dynamic strength calculation	[0 ... 128]
offset	Offset value for dynamic strength calculation	[-128 ... 128]
min	Upper bound for dynamic strength calculation	[0 ... 128]
div	Division factor for dynamic strength calculation	[0 ... 64]
sigma.green	Sigma value for green pixel	[1 ... 255]
sigma.red.blue	Sigma value for red/blue pixel	[1 ... 255]

- **IF\_DPF\_G\_EN**

This macro definition is identical to the string "dpf.g.en".

Description: Gets the enabled/disabled state of the De-noising Pre-Filter control.

Parameters:

- 
- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 52. Control words for IF\_DPF\_G\_EN

Control word	Description	Valid Values
enable	The state of the De-noising Pre-Filter	– true – false

- **IF\_DPF\_S\_EN**

This macro definition is identical to the string "dpf.s.en".

Description: Sets the enabled/disabled state of the De-noising Pre-Filter control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 53. Control words for IF\_DPF\_S\_EN

Control word	Description	Valid Values
enable	Enables or disables De-noising Pre-Filter	– true – false

- **IF\_EC\_G\_CFG**

This macro definition is identical to the string "ec.g.cfg".

Description: Gets the configuration values for the Exposure Control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 54. Control words for IF\_EC\_G\_CFG

Control Word	Description	Valid Values
gain	Exposure gain	sensor-specific
gain.min	Minimum gain	sensor-specific
gain.max	Maximum gain	sensor-specific
time	Exposure time	sensor-specific
inte.min	Minimum exposure time	sensor-specific
inte.max	Maximum exposure time	sensor-specific

- **IF\_EC\_S\_CFG**

This macro definition is identical to the string "ec.s.cfg".

Description: Sets the configuration values for the Exposure Control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 55. Control words for IF\_EC\_S\_CFG

Control word	Description	Valid Values
gain	Exposure gain	sensor-specific
time	Exposure time	sensor-specific

- **IF\_EE\_G\_CFG**

This macro definition is identical to the string "ee.g.cfg".

Description: Gets the configuration values for the Edge Enhancement control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 56. Control words for IF\_EE\_G\_CFG

Control Word	Description	Valid Values
auto	EE mode	true false
config	EE configure parameters	Edge gain: [0 ... 65535] Strength: [0 ... 128] UV gain: [0 ... 65535] Y gain down: [0 ... 65535] Y gain up: [0 ... 65535]

- **IF\_EE\_S\_CFG**

This macro definition is identical to the string "ee.s.cfg".

Description: Sets the configuration values for the Edge Enhancement control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 57. Control words for IF\_EE\_S\_CFG

Control Word	Description	Valid Values
auto	EE mode	true false
config	EE configure parameters	Edge gain: [0 ... 65535] Strength: [0 ... 128] UV gain: [0 ... 65535] Y gain down: [0 ... 65535] Y gain up: [0 ... 65535]

- **IF\_EE\_G\_EN**

This macro definition is identical to the string "ee.g.en".

Description: Gets the enabled/disabled state of the Edge Enhancement control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 58. Control words for IF\_EE\_G\_EN

Control word	Description	Valid Values
enable	The state of the Edge Enhancement control	– true – false

- **IF\_EE\_S\_EN**

This macro definition is identical to the string "ee.s.en".

Description: Sets the enabled/disabled state of the Edge Enhancement control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 59. Control words for IF\_EE\_S\_EN

Control word	Description	Valid Values
enable	Enables or disables Edge Enhancement	– true – false

- **IF\_EE\_RESET**

This macro definition is identical to the string "ee.reset".

Description: Resets the Edge Enhancement control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 60. Control words for IF\_EE\_RESET

Control word	Description	Valid Values
N/A	-	-

- **IF\_EE\_S\_TBL**

This macro definition is identical to the string "ee.s.tbl".

Description: Sets the EE control.

Parameters:

- `Json::Value &jRequest`

– Json::Value &jResponse

**Table 61. Control words for IF\_EE\_S\_TBL**

Control word	Description	Valid Values
table	EE table	description string

- **IF\_FILTER\_G\_CFG**

This macro definition is identical to the string "filter.g.cfg".

Description: Gets the configuration values for the Filter control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 62. Control words for IF\_FILTER\_G\_CFG**

Control word	Description	Valid Values
auto	Auto control	true false
denoise	Denoise	[0..10]
sharpen	Sharpen	[0..10]

- **IF\_FILTER\_S\_CFG**

This macro definition is identical to the string "filter.s.cfg".

Description: Sets the configuration values for the Filter control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 63. Control words for IF\_FILTER\_S\_CFG**

Control word	Description	Valid Values
auto	Auto control	true false
denoise	Denoise	[0..10]
sharpen	Sharpen	[0..10]

- **IF\_FILTER\_G\_EN**

This macro definition is identical to the string "filter.g.en".

Description: Gets the enabled/disabled state of the Filter control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 64. Control words for IF\_FILTER\_G\_EN**

Control word	Description	Valid Values
enable	The state of the Filter control	true false

- **IF\_FILTER\_S\_EN**

This macro definition is identical to the string "filter.s.en".

Description: Sets the enabled/disabled state of the Filter control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 65. Control words for IF\_FILTER\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Filter	true false

• **IF\_FILTER\_S\_TBL**

This macro definition is identical to the string "filter.s.tbl".

Description: Sets the Filter control table.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 66. Control words for IF\_FILTER\_S\_TBL**

Control word	Description	Valid Values
table	Filter table	description string

• **IF\_GC\_G\_CURVE**

This macro definition is identical to the string "gc.g.curve".

Description: Gets the configuration values for the Gamma control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 67. Control words for IF\_GC\_G\_CURVE**

Control word	Description	Valid Values
curve	Gamma curve	[0 ... 1023]

• **IF\_GC\_S\_CURVE**

This macro definition is identical to the string "gc.s.curve".

Description: Sets the configuration values for the Gamma Control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 68. Control words for IF\_GC\_S\_CURVE**

Control word	Description	Valid Values
curve	Gamma curve	[0 ... 1023]

• **IF\_GC\_G\_CFG**

This macro definition is identical to the string "gc.g.cfg".

Description: Gets the configuration values for the Gamma control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 69. Control words for IF\_GC\_G\_CFG**

Control word	Description	Valid Values
gc.mode	The mode of the Gamma Control.	– 1: logarithmic mode – 2: equidistant mode

- **IF\_GC\_S\_CFG**

This macro definition is identical to the string "gc.s.cfg".

Description: Sets the configuration values for the Gamma control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 70. Control words for IF\_GC\_G\_CFG**

Control word	Description	Valid Values
gc.mode	The mode of the Gamma Control.	<ul style="list-style-type: none"> <li>– 1: logarithmic mode</li> <li>– 2: equidistant mode</li> </ul>

- **IF\_GC\_G\_EN**

This macro definition is identical to the string "gc.g.en".

Description: Gets the enabled/disabled state of the Gamma Control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 71. Control words for IF\_GC\_G\_EN**

Control word	Description	Valid Values
enable	The state of the Gamma Control	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

- **IF\_GC\_S\_EN**

This macro definition is identical to the string "gc.s.en".

Description: Sets the enabled/disabled state of the Gamma Control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 72. Control words for IF\_GC\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Gamma Control	<ul style="list-style-type: none"> <li>– true</li> <li>– false</li> </ul>

- **IF\_HDR\_G\_CFG**

This macro definition is identical to the string "hdr.g.cfg".

Description: Gets the configuration of the High Dynamic Range control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 73. Control words for IF\_HDR\_G\_CFG**

Control word	Description	Valid Values
extension.bit	Extension bit	[0..4]
exposure.ratio	Exposure ratio	[0..16]

- **IF\_HDR\_S\_CFG**



This macro definition is identical to the string "hdr.s.cfg".

Description: Sets the configuration of the High Dynamic Range control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 74. Control words for IF\_HDR\_S\_CFG**

Control Word	Description	Valid Values
extension.bit	Extension bit	[0..4]
exposure.ratio	Exposure ratio	[0..16]

#### • IF\_HDR\_G\_EN

This macro definition is identical to the string "hdr.g.en".

Description: Gets the enabled/disabled state of the High Dynamic Range control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 75. Control words for IF\_HDR\_G\_EN**

Control word	Description	Valid Values
enable	The state of High Dynamic Range	- true - false

#### • IF\_HDR\_S\_EN

This macro definition is identical to the string "hdr.s.en".

Description: Sets the enabled/disabled state of the High Dynamic Range control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 76. Control words for IF\_HDR\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables High Dynamic Range	- true - false

#### • IF\_HDR\_RESET

This macro definition is identical to the string "hdr.reset".

Description: Resets the High Dynamic Range control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 77. Control words for IF\_HDR\_RESET**

Control word	Description	Valid Values
N/A	-	-

#### • IF\_IE\_G\_CFG

This macro definition is identical to the string "ie.g.cfg".

Description: Gets the configuration values for the Image Effects control.

## Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 78. Control words for IF\_IE\_G\_CFG

Control Word	Description	Valid Values
mode	IE working modes	1: Set a fixed chrominance of 128 (neutral gray) 2: Luminance and chrominance data is being inverted 3: Chrominance is changed to produce a historical like brownish image color 4: Converting picture to grayscale while maintaining one color component 5: Edge detection, looks like a relief made of metal 6: Edge detection, looks like a pencil drawing 7: Edge detection, looks like a sharper drawing
range	Image Effects configuration range	1: pixel value range according to BT.601 2: YCbCr full range [0 ... 255]
config	Image Effects configuration	---
tint.cb	Sepia Tint Cb of sepia mode	[0 ... 255]
tint.cr	Sepia Tint Cr of sepia mode	[0 ... 255]
selection	Color selection of color mode	1: red, green, and blue 2: blue 3: green 4: green and blue 5: red 6: red and blue 7: red and green
threshold	Color threshold of color mode	[0 ... 255]
emboss:coeff	Coefficient of emboss mode	[-128 ... 127]
sketch:coeff	Coefficient of sketch mode	[-128 ... 127]
sharpen:factor	Factor of sharpen mode	[0 ... 255]
sharpen:threshold	Threshold of sharpen mode	[0 ... 255]
sharpen:coeff	Coefficient of sharpen mode	[-128 ... 127]

## • IF\_IE\_S\_CFG

This macro definition is identical to the string "ie.s.cfg".

Description: Sets the configuration values for the Image Effects control.

## Parameters:

- Json::Value &jRequest

– Json::Value &jResponse

**Table 79. Control words for IF\_IE\_S\_CFG**

Control Word	Description	Valid Values
mode	IE working modes	1: Set a fixed chrominance of 128 (neutral gray) 2: Luminance and chrominance data is being inverted 3: Chrominance is changed to produce a historical like brownish image color 4: Converting picture to grayscale while maintaining one color component 5: Edge detection, looks like a relief made of metal 6: Edge detection, looks like a pencil drawing 7: Edge detection, looks like a sharper drawing
range	Image Effects configuration range	1: pixel value range according to BT.601 2: YCbCr full range [0 ... 255]
config	Image Effects configuration	---
tint.cb	Sepia Tint Cb of sepia mode	[0 ... 255]
tint.cr	Sepia Tint Cr of sepia mode	[0 ... 255]
selection	Color selection of color mode	1: red, green, and blue 2: blue 3: green 4: green and blue 5: red 6: red and blue 7: red and green
threshold	Color threshold of color mode	[0 ... 255]
emboss:coeff	Coefficient of emboss mode	[-128 ... 127]
sketch:coeff	Coefficient of sketch mode	[-128 ... 127]
sharpen:factor	Factor of sharpen mode	[0 ... 255]
sharpen:threshold	Threshold of sharpen mode	[0 ... 255]
sharpen:coeff	Coefficient of sharpen mode	[-128 ... 127]

• **IF\_IE\_G\_EN**

This macro definition is identical to the string "ie.g.en".

Description: Gets the enabled/disabled state of the Image Effects control.

Parameters:

– Json::Value &jRequest

– Json::Value &jResponse

**Table 80. Control words for IF\_IE\_G\_EN**

Control word	Description	Valid Values
enable	The state of the Image Effects control	– true – false

#### • IF\_IE\_S\_EN

This macro definition is identical to the string "ie.s.en".

Description: Sets the enabled/disabled state of the Image Effects control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 81. Control words for IF\_IE\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Image Effects	– true – false

#### • IF\_LSC\_G\_EN

This macro definition is identical to the string "lsc.g.en".

Description: Gets the enabled/disabled state of the Lens Shade Correction control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 82. Control words for IF\_LSC\_G\_EN**

Control word	Description	Valid Values
enable	The state of Lens Shade Correction	– true – false

#### • IF\_LSC\_S\_EN

This macro definition is identical to the string "lsc.s.en".

Description: Sets the enabled/disabled state of the Lens Shade Correction control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 83. Control words for IF\_LSC\_S\_EN**

Control word	Description	Valid Values
enable	Enables or disables Lens Shade Correction	– true – false

#### • IF\_2DNR\_G\_CFG

This macro definition is identical to the string "2dnr.g.cfg".

Description: Gets the configuration values for the 2DNR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

Table 84. Control words for IF\_2DNR\_G\_CFG

Control Word	Description	Valid Values
generation	2DNR generation	[0 ... 2]
auto	3DNR running mode	true false
denoise.pregama.strength	Denoise pregame strength	[0, 1]
denoise.strength	Configuration strength	[0 ... 100]
sigma	Sigma strength	[0 ... 100]

- **IF\_2DNR\_S\_CFG**

This macro definition is identical to the string "2dnr.s.cfg".

Description: Sets the configuration values for the 2DNR control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 85. Control words for IF\_2DNR\_S\_CFG

Control Word	Description	Valid Values
generation	2DNR generation	[0 ... 2]
auto	3DNR running mode	true false
denoise.pregama.strength	Denoise pregame strength	[0, 1]
denoise.strength	Configuration strength	[0 ... 100]
sigma	Sigma strength	[0 ... 100]

- **IF\_2DNR\_G\_EN**

This macro definition is identical to the string "2dnr.g.en".

Description: Gets the enabled/disabled state of the 2DNR control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 86. Control words for IF\_2DNR\_G\_EN

Control Word	Description	Valid Values
enable	The state of 2DNR	true false
generation	2DNR generation	[0 ... 2]

- **IF\_2DNR\_S\_EN**

This macro definition is identical to the string "2dnr.s.en".

Description: Sets the enabled/disabled state of the 2DNR control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 87. Control words for IF\_2DNR\_S\_EN

Control Word	Description	Valid Values
enable	Enables or disables 2DNR	true false
generation	2DNR generation	[0 ... 2]

- **IF\_2DNR\_RESET**

This macro definition is identical to the string "2dnr.reset".

Description: Resets the 2DNR control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 88. Control words for IF\_2DNR\_RESET

Control word	Description	Valid Values
generation	NR2D generation	[0 ... 2]

- **IF\_2DNR\_S\_TBL**

This macro definition is identical to the string "2dnr.s.tbl".

Description: Resets the 2DNR control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 89. Control words for IF\_2DNR\_S\_TBL

Control word	Description	Valid Values
table	2DNR table	description string

- **IF\_3DNR\_G\_CFG**

This macro definition is identical to the string "3dnr.g.cfg".

Description: Gets the configuration values for the 3DNR control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 90. Control words for IF\_3DNR\_G\_CFG

Control Word	Description	Valid Values
generation	3DNR generation	[0 ... 2]
auto	3DNR running mode	true false
strength	3DNR strength	[0 ... 128]
delta.factor	Delta factor value	[0 ... 1023]
motion.factor	motion factor value	[0 ... 1000000]

- **IF\_3DNR\_S\_CFG**

This macro definition is identical to the string "3dnr.s.cfg".

Description: Sets the configuration values for the 3DNR control.

Parameters:

- `Json::Value &jRequest`

– Json::Value &jResponse

**Table 91. Control words for IF\_3DNR\_S\_CFG**

Control Word	Description	Valid Values
generation	3DNR generation	[0 ... 2]
auto	3DNR running mode	true false
strength	3DNR strength	[0 ... 128]
delta.factor	Delta factor value	[0 ... 1023]
motion.factor	motion factor value	[0 ... 1000000]

• **IF\_3DNR\_G\_EN**

This macro definition is identical to the string "3dnr.g.en".

Description: Gets the enabled/disabled state of the 3DNR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 92. Control words for IF\_3DNR\_G\_EN**

Control Word	Description	Valid Values
enable	The state of 3DNR	true false
generation	3DNR generation	[0 ... 2]

• **IF\_3DNR\_S\_EN**

This macro definition is identical to the string "3dnr.s.en".

Description: Sets the enabled/disabled state of the 3DNR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 93. Control words for IF\_3DNR\_S\_EN**

Control Word	Description	Valid Values
enable	Enable or disable 3DNR	true false
generation	3DNR generation	[0 ... 2]

• **IF\_3DNR\_RESET**

This macro definition is identical to the string "3dnr.reset".

Description: Resets the 3DNR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 94. Control words for IF\_3DNR\_RESET**

Control word	Description	Valid Values
generation	NR3D generation	[0 ... 2]

• **IF\_3DNR\_S\_TBL**

This macro definition is identical to the string "3dnr.reset".

Description: Resets the 3DNR control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 95. Control words for IF\_3DNR\_S\_TBL**

Control word	Description	Valid Values
table	3DNR table	description string
generation	3DNR generation	[0 ... 2]

#### • IF\_WDR\_G\_CFG

This macro definition is identical to the string "wdr.g.cfg".

Description: Gets the configuration values for the WDR control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 96. Control words for IF\_WDR\_G\_CFG**

Control Word	Description	Valid Values
generation	WDR generation	[0 ... 2]
curve	WDR1 curve value	[tone mapping curve values]
auto	WDR3 running mode	true false
strength	WDR2 or WDR3 strength	[0, 128]
gain.max	WDR3 gain max	[0, 128]
strength.global	WDR3 global strength	[0, 128]

#### • IF\_WDR\_S\_CFG

This macro definition is identical to the string "wdr.s.cfg".

Description: Sets the configuration values for the WDR control.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

**Table 97. Control words for IF\_WDR\_S\_CFG**

Control Word	Description	Valid Values
generation	WDR generation	[0 ... 2]
curve	WDR1 curve value	[tone mapping curve values]
auto	WDR3 running mode	true false
strength	WDR2 or WDR3 strength	[0, 128]
gain.max	WDR3 gain max	[0, 128]
strength.global	WDR3 global strength	[0, 128]

#### • IF\_WDR\_G\_EN

This macro definition is identical to the string "wdr.g.en".

Description: Gets the enabled/disabled state of the WDR control.

Parameters:

- `Json::Value &jRequest`



– Json::Value &jResponse

**Table 98. Control words for IF\_WDR\_G\_EN**

Control Word	Description	Valid Values
enable	The state of WDR	true false
generation	WDR generation	[0 ... 2]

• **IF\_WDR\_S\_EN**

This macro definition is identical to the string "wdr.s.en".

Description: Sets the enabled/disabled state of the WDR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 99. Control words for IF\_WDR\_S\_EN**

Control Word	Description	Valid Values
enable	Enables or disables WDR	true false
generation	WDR generation	[0 ... 2]

• **IF\_WDR\_RESET**

This macro definition is identical to the string "wdr.reset".

Description: Resets the WDR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 100. Control words for IF\_WDR\_RESET**

Control word	Description	Valid Values
generation	WDR generation	[0 ... 2]

• **IF\_WDR\_S\_TBL**

This macro definition is identical to the string "wdr.s.tb".

Description: Sets the table of the WDR control.

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

**Table 101. Control words for IF\_WDR\_S\_TBL**

Control word	Description	Valid Values
table	WDR table	description string
generation	WDR generation	[0 ... 2]

• **IF\_WB\_G\_CFG**

Description: Gets the configuration values for the WB control.

This macro definition is identical to the string "wb.g.cfg".

Parameters:

– Json::Value &jRequest  
– Json::Value &jResponse

Table 102. Control words for IF\_WB\_G\_CFG

Control Word	Description	Valid Values
matrix	Matrix	[<matrix values>]
offset	Offset	[-2048, 2047]
red	Cc offset red	[-2048, 2047]
green	Cc offset green	[-2048, 2047]
blue	Cc offset blue	[-2048, 2047]
red	WB gains red	[0.000, 3.999]
green.r	WB gains green.r	[0.000, 3.999]
green.b	WB gains green.b	[0.000, 3.999]
blue	WB gains blue	[0.000, 3.999]
wb.gains	WB gains	[0.000, 3.999]

- **IF\_WB\_S\_CFG**

This macro definition is identical to the string "wb.s.cfg".

Description: Sets the configuration values for the WB control.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 103. Control words for IF\_WB\_S\_CFG

Control Word	Description	Valid Values
matrix	Matrix	[<matrix values>]
offset	Offset	[-2048, 2047]
red	Cc offset red	[-2048, 2047]
green	Cc offset green	[-2048, 2047]
blue	Cc offset blue	[-2048, 2047]
red	WB gains red	[0.000, 3.999]
green.r	WB gains green.r	[0.000, 3.999]
green.b	WB gains green.b	[0.000, 3.999]
blue	WB gains blue	[0.000, 3.999]
wb.gains	WB gains	[0.000, 3.999]

### 3.2.5.4 Dewarp control words

**Note:** Requires hardware with dewarp capability.

- **IF\_DWE\_S\_PARAMS**

This macro definition is identical to the string "dwe.s.params".

Description: Sets the dewarp parameters: input format, output format, ROI, scale, split, dewarp type, and so on.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 104. Control Words for IF\_DWE\_S\_PARAMS

Control Word	Description	Valid Values
dwe	Get the dewarp node	node description string
mode	Dewarp type	1: lens distortion 2: 'fisheye expand' 4: split screen 8: 'fisheye dewarp'
hflip	Set horizontal flip true or false	– true – false
vflip	Set vertical flip true or false	– true – false
bypass	Bypass dewarp true or false	– true – false
mat	Camera matrix [0-8] Distortion coefficient [9-16]	Need calibration

- **IF\_DWE\_G\_PARAMS**

This macro definition is identical to the string "dwe.g.params".

Description: Gets the Dewarp parameters: input format, output format, ROI, scale, split, dewarp type, and so on.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 105. Control words for IF\_DWE\_G\_PARAMS

Control Word	Description	Valid Values
dwe	Get the dewarp node	node description string
mode	Dewarp type	1: lens distortion 2: 'fisheye expand' 4: split screen 8: fisheye dewarp
hflip	Set horizontal flip true or false	– true – false
vflip	Set vertical flip true or false	– true – false
bypass	Bypass dewarp true or false	– true – false
mat	Camera matrix [0~8] Distortion coefficient [9~16]	Need calibration

- **IF\_DWE\_S\_HFLIP**

This macro definition is identical to the string "dwe.s.hflip".

Description: Sets the image horizontal flip parameters.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 106. Control words for IF\_DWE\_S\_HFLIP

Control Word	Description	Valid Values
dwe	Get the dewarp node	node description string
hflip	Set vertical flip	– true – false

- **IF\_DWE\_S\_VFLIP**

This macro definition is identical to the string "dwe.s.vflip".

Description: Sets the image vertical flip parameters.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 107. Control words for IF\_DWE\_S\_VFLIP

Control Word	Description	Valid Values
dwe	Get the dewarp node	node description string
vflip	Set horizontal flip	– true – false

- **IF\_DWE\_S\_BYPASS**

This macro definition is identical to the string "dwe.s.bypass".

Description: Sets the Dewarp bypass true or false.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 108. Control words for IF\_DWE\_S\_BYPASS

Control word	Description	Valid Values
dwe	Get the dewarp node	node description string
bypass	Bypass dewarp true or false	– true – false

- **IF\_DWE\_S\_MODE**

This macro definition is identical to the string "dwe.s.mode".

Description: Sets the Dewarp mode index.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 109. Control words for IF\_DWE\_S\_MODE

Control word	Description	Valid Values
mode	Sensor mode index	Sensor-specific
dwe	Get the dewarp node	– true – false

- **IF\_DWE\_S\_MAT**

This macro definition is identical to the string "dwe.s.mat".

Description: Sets the camera matrix and distortion coefficient.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 110. Control words for IF\_DWE\_S\_MAT

Control word	Description	Valid Values
dwe	Get the dewarp node	node description string
mat	Camera matrix [0-8] Distortion coefficient [9-16]	Need calibration

- **IF\_DWE\_S\_TYPE**

This macro definition is identical to the string "dwe.s.type".

Description: Sets the Dewarp type.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 111. Control words for IF\_DWE\_S\_TYPE

Control word	Description	Valid Values
type	Dewarp type	1: lens distortion 2: 'fisheye expand' 4: split screen 8: 'fisheye dewarp'
dwe	Get the dewarp node	node description string

- **VIV\_V4L\_DWE\_SET\_CROP**

Description: Crops the image.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 112. Control words for VIV\_V4L\_DWE\_SET\_CROP

Control Word	Description	Valid Values
crop	Get the crop node	node description string
start_x	The starting position of the X-axis	Sensor-specific
start_y	The starting position of the Y-axis	Sensor-specific
width	Crop width	Sensor-specific
height	Crop height	Sensor-specific

- **VIV\_V4L\_DWE\_SET\_SCALE**

Description: Scales the image.

Parameters:

- `Json::Value &jRequest`
- `Json::Value &jResponse`

Table 113. Control words for VIV\_V4L\_DWE\_SET\_SCALE

Control Word	Description	Valid Values
scale	Get the scale node	node description string
start_x	The starting position of the X-axis	Sensor-specific

Table 113. Control words for VIV\_V4L\_DWE\_SET\_SCALE...continued

Control Word	Description	Valid Values
start_y	The starting position of the Y-axis	Sensor-specific
width	Scale width	Sensor-specific
height	Scale height	Sensor-specific

### 3.2.5.5 Sensor Control Words

#### • IF\_SENSOR\_QUERY

This macro definition is identical to the string "sensor.query".

Description: Queries sensor information.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 114. Control words for Sensor Control Words

Control Word	Description	Valid Values
current	Sensor current mode	Sensor-specific
default	Sensor default mode	Sensor-specific
index	Sensor index	Sensor-specific
width	Resolution width	Sensor-specific
height	Resolution height	Sensor-specific
fps	Sensor support maximum frame rate	Sensor-specific
hdr_mode	The mode corresponds to the sensor index	Sensor-specific
bit_width	Sensor support bit width	Sensor-specific
bayer_pattern	Sensor support Bayer pattern	Sensor-specific
stitching_mode	Sensor support stitching mode	Sensor-specific

#### • IF\_SENSOR\_G\_MODE

This macro definition is identical to the string "sensor.g.mode".

Description: Gets sensor current mode.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 115. Control words for IF\_SENSOR\_G\_MODE

Control word	Description	Valid Values
mode	Sensor current index	Sensor-specific

#### • IF\_SENSOR\_S\_MODE

This macro definition is identical to the string "sensor.s.mode".

Description: Sets sensor mode.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 116. Control words for IF\_SENSOR\_S\_MODE**

Control word	Description	Valid Values
mode	Sensor mode index	Sensor-specific
CalibXmlName	Calibration file name	Sensor-specific

- **IF\_SENSOR\_G\_RESW**

This macro definition is identical to the string "sensor.g.resw".

Description: Gets the sensor resolution width.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 117. Control words for IF\_SENSOR\_G\_RESW**

Control word	Description	Valid Values
resw	Resolution width	Sensor-specific

- **IF\_SENSOR\_G\_RESH**

This macro definition is identical to the string "sensor.g.resw".

Description: Gets the sensor resolution height.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 118. Control words for IF\_SENSOR\_G\_RESW**

Control word	Description	Valid Values
resh	Resolution height	Sensor-specific

- **IF\_SENSOR\_G\_REG**

This macro definition is identical to the string "sensor.g.reg".

Description: Gets the sensor register value.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

**Table 119. Control words for IF\_SENSOR\_G\_REG**

Control word	Description	Valid Values
value	Register value	Sensor-specific
address	Register address	Sensor-specific

- **IF\_SENSOR\_S\_REG**

This macro definition is identical to the string "sensor.s.reg".

Description: Sets the sensor register value.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 120. Control words for IF\_SENSOR\_S\_REG

Control word	Description	Valid Values
value	Register value	Sensor-specific
address	Register address	Sensor-specific

- **IF\_S\_FPS**

This macro definition is identical to the string "s.fps".

Description: Sets the sensor frame rate.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 121. Control words for IF\_S\_FPS

Control word	Description	Valid Values
fps	Sensor frame rate	Sensor-specific

- **IF\_SENSOR\_LIB\_PRELOAD**

This macro definition is identical to the string "sensor.lib.preload".

Description: Loads the sensor calibration file.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 122. Control words for IF\_SENSOR\_LIB\_PRELOAD

Control word	Description	Valid Values
N/A	-	-

- **IF\_SENSOR\_G\_SEC**

- This macro definition is identical to the string "sensor.g.sec".

Description: Get sensor start exposure (IntegrationTime x Gain).

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 123. Control words for IF\_SENSOR\_G\_SEC

Control Word	Description	Valid Values
exposure	AE start exposure = IntegrationTime x Gain	Float value

- **IF\_SENSOR\_S\_SEC**

This macro definition is identical to the string "sensor.s.sec".

**Note:** Calling this function is only valid before the stream on.

Description: Sets sensor start exposure (IntegrationTime x Gain).

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse



Table 124. Control words for IF\_SENSOR\_S\_SEC

Control Word	Description	Valid Values
exposure	AE start exposure = IntegrationTime x Gain	Float value

- **IF\_SENSOR\_S\_TESTPAT**

This macro definition is identical to the string "sensor.s.testpat".

Description: Sets sensor test pattern.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 125. Control words for IF\_SENSOR\_S\_TESTPAT

Control Word	Description	Valid Values
test.pattern	Sensor test pattern mode	0: normal mode others: test pattern mode value

### 3.2.5.6 Pipeline Control Words

- **IF\_PIPELINE\_S\_WARM\_UP**

This macro definition is identical to the string "pipeline.s.warm.up".

Description: Warms up pipeline.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 126. Control words for IF\_PIPELINE\_S\_WARM\_UP

Control Word	Description	Valid Values
N/A	-	-

- **IF\_PIPELINE\_S\_SMP\_MODE**

This macro definition is identical to the string "pipeline.s.smp.mode".

Description: Reserved for later use.

- **IF\_PIPELINE\_S\_DWE\_ONOFF**

This macro definition is identical to the string "pipeline.s.dwe.onoff".

Description: Enables/disables DEWARP.

Parameters:

- Json::Value &jRequest
- Json::Value &jResponse

Table 127. Control words for IF\_PIPELINE\_S\_DWE\_ONOFF

Control Word	Description	Valid Values
enable	Enable or disable dewarp.	true false

- **IF\_PIPELINE\_S\_TESTPAT**

This macro definition is identical to the string "pipeline.s.testpat".

Description: Reserved for later use.

- **IF\_PIPELINE\_S\_RES\_IS\_OUT**

This macro definition is identical to the string "pipeline.s.res.is.out".

Description: Reserved for later use.

- **IF\_PIPELINE\_S\_RES\_MP\_OUT**

This macro definition is identical to the string "pipeline.s.res.mp.out".  
Description: Reserved for later use.

- **IF\_PIPELINE\_S\_MP\_FMT**  
This macro definition is identical to the string "pipeline.s.mp.fmt".  
Description: Reserved for later use.
- **IF\_PIPELINE\_QUERY**  
This macro definition is identical to the string "pipeline.query".  
Description: Reserved for later use.
- **IF\_PIPELINE\_CFG\_STATUS**  
This macro definition is identical to the string "pipeline.cfg.status".  
Description: Reserved for later use.

3.3 ISP software V4L2 programming overview

3.3.1 General concept

The high-level diagram of the ISP V4L2 software stack is shown in [Figure 13](#).

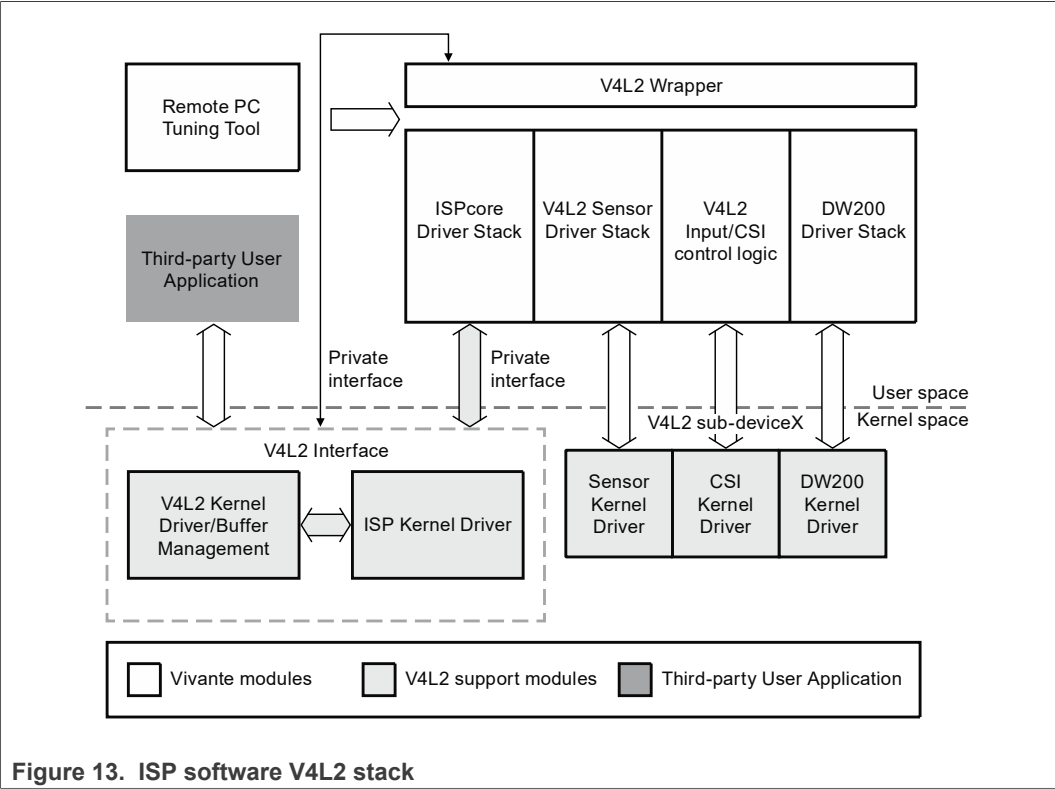


Figure 13. ISP software V4L2 stack

3.3.2 V4L2 kernel driver block diagram

ISP provides some device nodes in its file structure. Customers can operate the corresponding device through the appropriate device node(s).

Table 128. ISP device nodes

Device node/driver	Description
/dev/mediax	Enumerate video devices and subdevices.

Table 128. ISP device nodes...continued

Device node/driver	Description
/dev/videoX	Manage stream related operations and events, such as enqueue/dequeue buffers and enqueue/dequeue events
/dev/v4l2-subdevX	Manage buffers, and Control camera relevant hardware, such as MIPI/Sensor
/dev/xx	Private interface control and dispatch the commands, events, and so on.
V4L2 kernel driver	Register the <code>V4L2_device</code> and <code>video_device</code> and implement the operational functions in the <code>video_device</code> and <code>vb2_queue</code>
ISP kernel driver	ISP kernel driver, implements read/write registers, and so on.

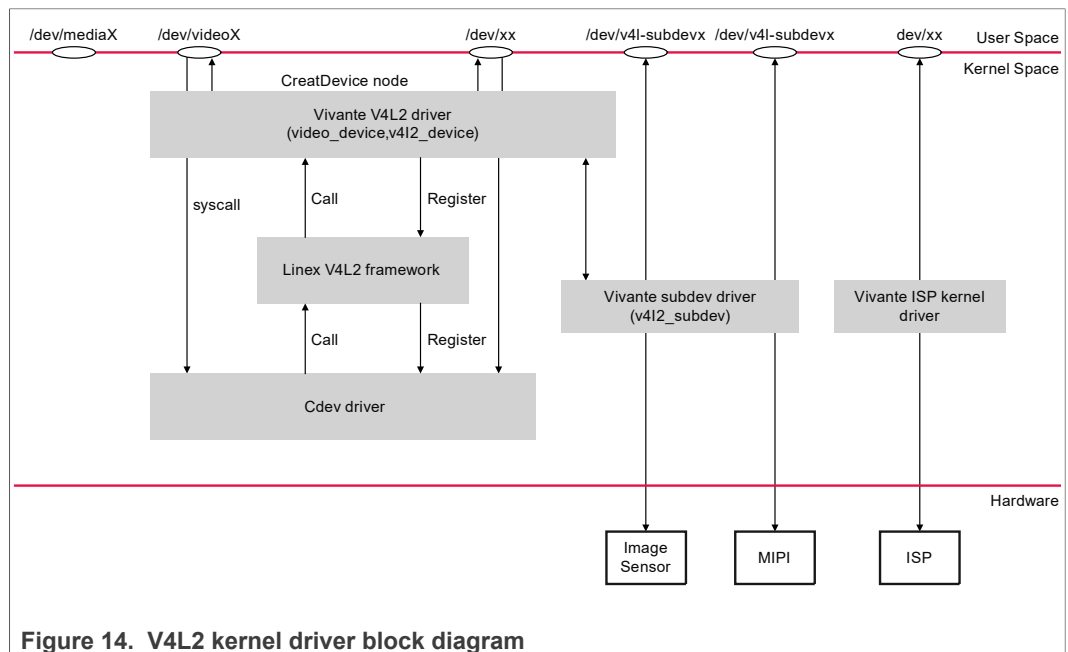


Figure 14. V4L2 kernel driver block diagram

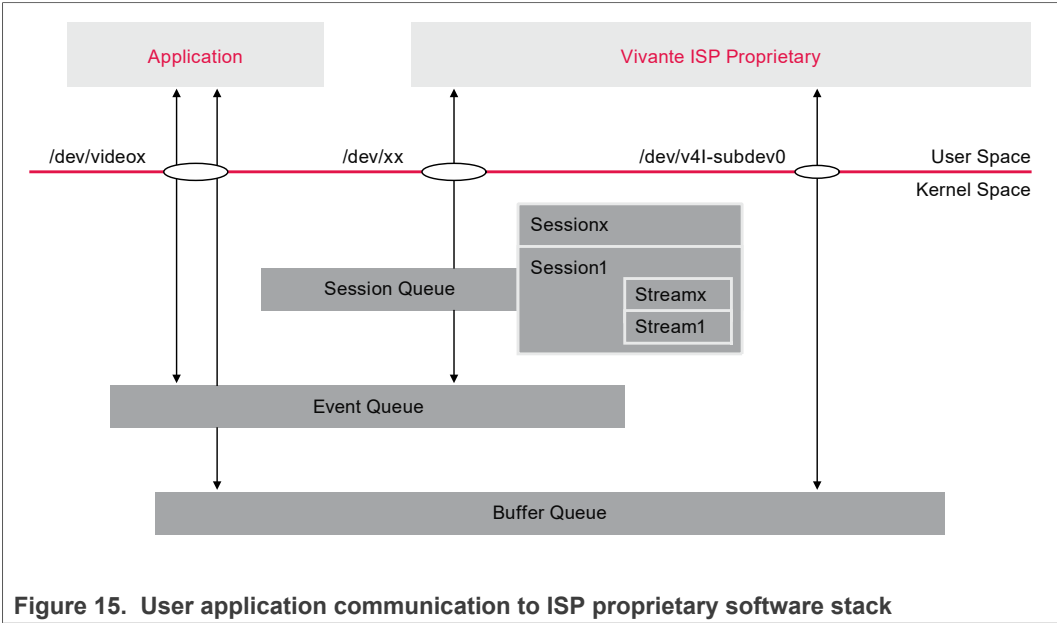
### 3.3.3 V4L2 third-party user application and ISP stack communication

The V4L2 third-party user application communicates directly with the kernel with V4L2 standard control words and V4L2 extension control commands. All the user application controls pass to the kernel space to the V4L2 kernel driver.

The V4L2 kernel driver handles the API commands and requests from the V4L2 user application. It communicates to the ISP software stack and delivers image buffers to the V4L2 user application.

Submodules that handle the event and buffer:

- Event Queue: send/get events to/from ISP proprietary software.
- Buffer Queue: manages the vb2 buffer.

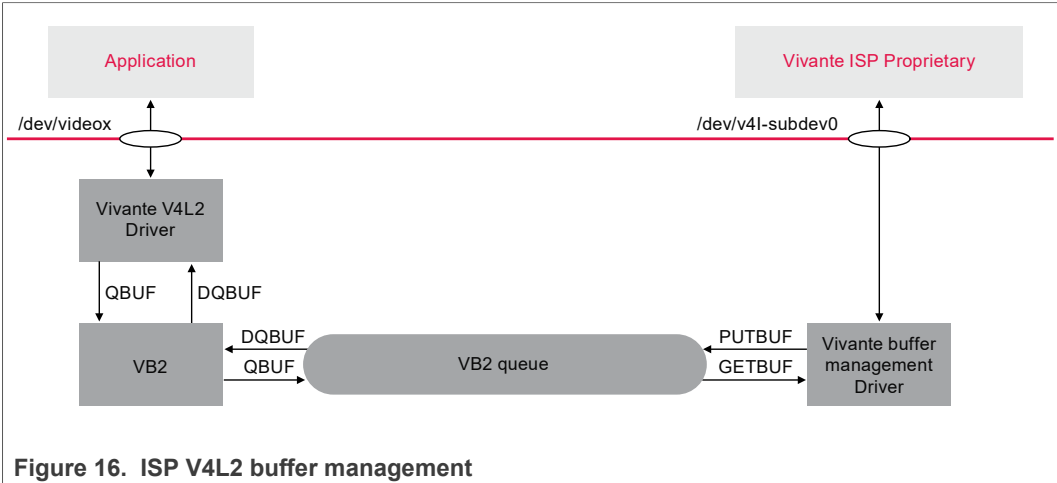


3.3.4 ISP V4L2 buffer management

There are three memory types as described in [Table 129](#) and [Figure 16](#).

Table 129. Memory types and buffer allocation

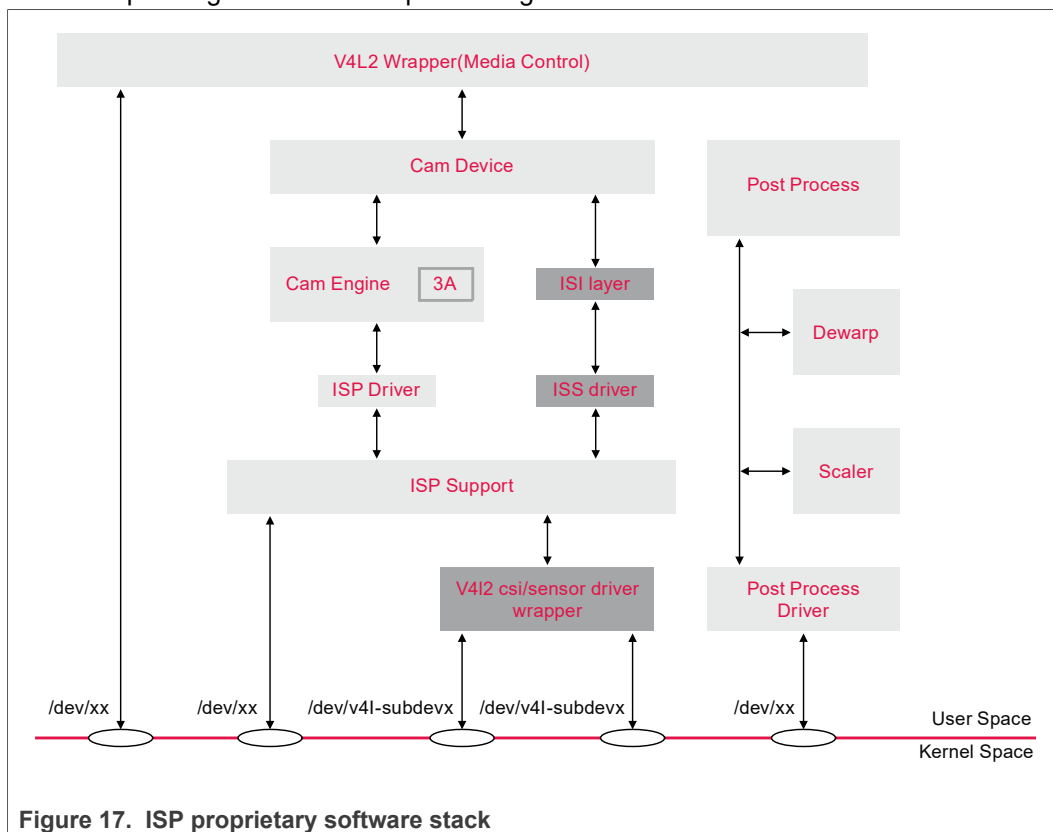
Memory type	Buffer allocation	Behavior
USERPTR	user space	User space and kernel space share the memory by buffer pointer
MMAP	kernel space	User space calls mmap to get pointer from kernel space
DMABUF	kernel space	User space gets the buffers using a file descriptor



**Note:** USERPTR mode is not supported.

### 3.3.5 ISP proprietary software stack

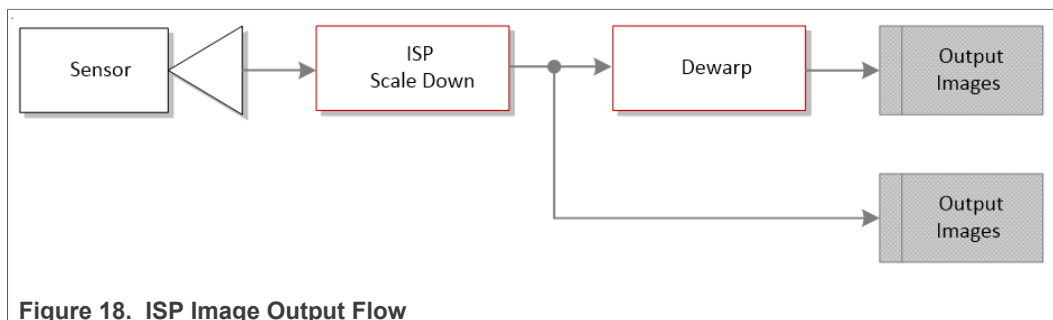
The camera manager receives messages for the kernel and dispatches these events to the corresponding submodule for processing.



### 3.4 Arbitrary Resolution Control

### 3.4.1 Introduction to Arbitrary Resolution

All resolutions are limited, where the minimum resolution is 176x144 and the maximum resolution is the sensor output resolution (refer to the sensor specification). The [`VIDIOC\_S\_FMT`](#) IOCTL which sets the format information must be aligned with width 16 and height 8.



The image output flow is shown in the figure above. If the Dewarp output is used, the data after the ISP scale down is used as the input of the Dewarp module. Thus, the

Dewarp correction parameters must be calibrated according to the size of the Dewarp input image. If there is no calibration parameter with the corresponding resolution, the system scales the calibration parameter of the current resolution according to the existing calibration parameters. In this case, the converted calibration data is not as accurate as the calibration data. Therefore, it is recommended to calibrate all resolutions used and add the resulting calibration data to the Dewarp configuration file.

### 3.4.2 Dewarp Calibration

This section describes dewarp calibration for the ISP + Dewarp IP configuration. It does not apply to the ISP-only case.

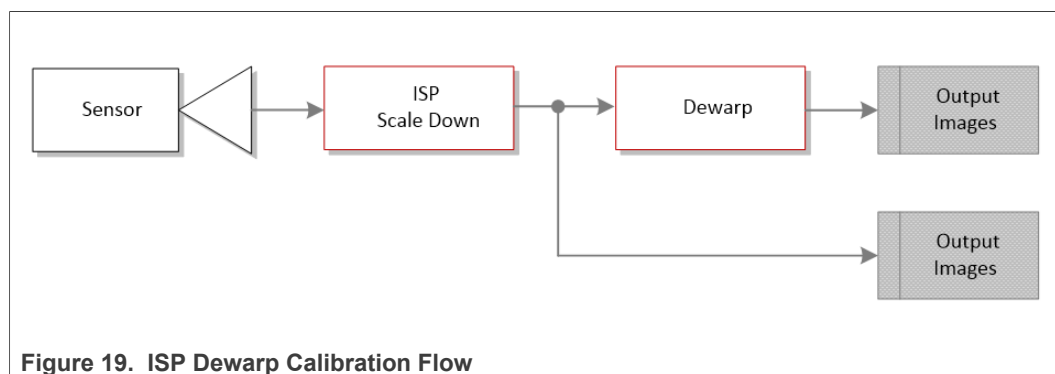


Figure 19. ISP Dewarp Calibration Flow

Use the following steps for Dewarp Calibration:

1. Use a set of default configurations and enable Dewarp bypass.

```

{
  "dewarpConfigArray" : [
    {
      "source_image": {
        "width" : 1920,
        "height" : 1080
      },
      "?dewarpType": "LENS_CORRECTION, FISHEYE_EXPAND,
      SPLIT_SCREEN",
      "dewarpType": "FISHEYE_DEWARP",
      "scale": {
        "roix" : 0,
        "roiy" : 0,
        "factor" : 1.0
      },
      "split": {
        "horizon_line" : 540,
        "vertical_line_up" : 960,
        "vertical_line_down": 960
      },
      "bypass" : true,
      "hflip" : false,
      "vflip" : false,
      "camera_matrix" : [6.5516074404594690e+002,0.0,
        9.6420599053623062e+002,
        0.0,6.5552406676868952e+002,5.3203601317192908e
        +002,0.0,0.0,1.0],
      "distortion_coeff":
        [-2.2095698671518085e-002,3.8543889520066955e-003,-
  
```

```
5.9060355970132873e-003,1.9007362178503509e-003,0.0,0.0,0.0,0.0],
    "perspective"      : [1.0, 0, 0, 0, 1, 0, 0, 0, 1]
  }
]
```

2. Capture YUV image. For example, video test is used to capture 720p YUYV images:

```
./isp_media_server CAMERA0 & ./video_test -w 1280 -h 720 -f
YUYV -t 2 -m0 -d0
```

3. Use an online YUV to JPEG image conversion tool to convert the YUV image to a JPEG image.
4. Use the JPEG image and the **Dewarp Calibration Tool** to get the dewarp calibration data. Refer to the document, `Vivante.DW.Calibration.Tool` for more details.
5. Add the dewarp calibration data to the Dewarp configuration file.

```
{
  "dewarpConfigArray" : [
    {
      "source_image": {
        "width" : 1920,
        "height" : 1080
      },
      "?dewarpType": "LENS_CORRECTION, FISHEYE_EXPAND,
SPLIT_SCREEN",
      "dewarpType": "FISHEYE_DEWARP",
      "scale": {
        "roix" : 0,
        "roiy" : 0,
        "factor" : 1.0
      },
      "split": {
        "horizon_line" : 540,
        "vertical_line_up" : 960,
        "vertical_line_down": 960
      },
      "bypass" : false,
      "hflip" : false,
      "vflip" : false,
      "camera_matrix" : [6.5516074404594690e+002,0.0,
        9.6420599053623062e+002,
        0.0,6.5552406676868952e+002,5.3203601317192908e
+002,0.0,0.0,1.0],
      "distortion_coeff":
[-2.2095698671518085e-002,3.8543889520066955e-003,-
5.9060355970132873e-003,1.9007362178503509e-003,0.0,0.0,0.0,0.0],
      "perspective"      : [1.0, 0, 0, 0, 1, 0, 0, 0, 1]
    }
  ]
  {
    "source_image": {
      "width" : 1280,
      "height" : 720
    },
    "?dewarpType": "LENS_CORRECTION, FISHEYE_EXPAND,
SPLIT_SCREEN",
    "dewarpType": "FISHEYE_DEWARP",
    "scale": {
      "roix" : 0,
```

```

        "roi_y" : 0,
        "factor" : 1.0
    },
    "split": {
        "horizon_line" : 540,
        "vertical_line_up" : 960,
        "vertical_line_down": 960
    },
    "bypass" : false,
    "hflip" : false,
    "vflip" : false,
    "camera_matrix" : [4.367738293639646e+002,0.0,
6.4280399369082041e+002,
0.0,4.3701604451245968e+002,3.5469067544795272e
+002,0.0,0.0,1.0],
    "distortion_coeff":
[-2.2095698671518085e-002,3.8543889520066955e-003,
-5.9060355970132873e-003,1.9007362178503509e-003,0.0,0.0,0.0,0.0],
    "perspective" : [1.0, 0, 0, 0, 1, 0, 0, 0, 1]
    }
}

```

## 4 ISP Software Arbitrary Resolution Switch Guide

### 4.1 ISP sensor input

#### 4.1.1 Sensor size limitation

SensorInputSize (bounds\_width x bounds\_height):

- Minimum resolution: 176 x 144
- Maximum resolution: 4096 x 3072
- Width alignment: 16 pixels
- Height alignment: 8 pixels

EffectiveDataSize (width x height):

- Minimum resolution: 176 x 144
- Maximum resolution: 4096 x 3072
- Width alignment: 16 pixels
- Height alignment: 8 pixels

Limitations:

- Offset top: no limitation
- Offset left: no limitation
- Bounds\_width >= (left + width)
- Bounds\_height >= (top + height)

Changing the sensor input must be supported by the sensor driver.

Modify `sensor0_entry.cfg` for `sensor0` or `sensor1_entry.cfg` for `sensor1`.

For the examples in this section:



- xml is the calibration XML file generated by the ISP Calibration Tools.
- dwe is the dewarp configuration file.

Table 130. Sensor mode and resolution

Sensor Mode	Resolution
0	<a href="#">3840x2160 Linear</a>
1	<a href="#">3840x2160 HDR</a>
2	<a href="#">1920x1080 Linear</a>
3	<a href="#">1920x1080 HDR</a>

#### 4.1.2 Sensor size configuration in driver

Some sensors have embedded data.

For example, for sensor input size of 1936 x 1096, the effective data size is 1920 x 1080 with offset 8 x 8, bounds\_width and bounds\_height indicate we use the entire sensor input size, the top, left, width, and height specify the rectangle of effective data area. ISP will crop and use the effective data size as the ISP input in the ACQ module.

```
.size          = {
    .bounds_width  = 1936,
    .bounds_height = 1096,
    .top           = 8,
    .left          = 8,
    .width         = 1920,
    .height        = 1080,
},
```

For a Normal Sensor, the sensor input data is all effective data, so:

- bounds\_width = width
- bounds\_height = height
- top = 0
- left = 0

```
.size          = {
    .bounds_width  = 1920,
    .bounds_height = 1080,
    .top           = 0,
    .left          = 0,
    .width         = 1920,
    .height        = 1080,
},
```

#### 4.1.3 Sensor Mode 0, 3840x2160 Linear

```
name="basler-vvcam"
drv = "daA3840_30mc.drv"
mode= 0
[mode.0]
xml = "DAA3840_30MC_4K-linear.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.1]
xml = "DAA3840_30MC_1080P-linear.xml"
```

```
dwe = "dewarp_config/daA3840_30mc_1080P.json"
[mode.2]
xml = "DAA3840_30MC_4K-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.3]
xml = "DAA3840_30MC_1080P-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
```

#### 4.1.4 Sensor Mode 1, 3840x2160 HDR

```
name="basler-vvcam"
drv = "daA3840_30mc.drv"
mode= 1
[mode.0]
xml = "DAA3840_30MC_4K-linear.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.1]
xml = "DAA3840_30MC_1080P-linear.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
[mode.2]
xml = "DAA3840_30MC_4K-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.3]
xml = "DAA3840_30MC_1080P-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
```

#### 4.1.5 Sensor Mode 2, 1920x1080 Linear

```
name="basler-vvcam"
drv = "daA3840_30mc.drv"
mode= 2
[mode.0]
xml = "DAA3840_30MC_4K-linear.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.1]
xml = "DAA3840_30MC_1080P-linear.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
[mode.2]
xml = "DAA3840_30MC_4K-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.3]
xml = "DAA3840_30MC_1080P-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
```

#### 4.1.6 Sensor Mode 3, 1920x1080 HDR

```
name="basler-vvcam"
drv = "daA3840_30mc.drv"
mode= 3
[mode.0]
xml = "DAA3840_30MC_4K-linear.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.1]
xml = "DAA3840_30MC_1080P-linear.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
```

```
[mode.2]
xml = "DAA3840_30MC_4K-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_4K.json"
[mode.3]
xml = "DAA3840_30MC_1080P-hdr.xml"
dwe = "dewarp_config/daA3840_30mc_1080P.json"
```

## 4.2 Arbitrary resolution output

- Minimum resolution: 176 x 144
- Maximum resolution: 4096 x 3072
- Width alignment: 16 pixels
- Height alignment: 8 pixels
- pixel\_format: V4L2\_PIX\_FMT\_YUYV, V4L2\_PIX\_FMT\_NV12, V4L2\_PIX\_FMT\_NV16

### 4.2.1 V4L2 API

Get the supported resolution:

```
int ioctl(int fd, VIDIOC_ENUM_FRAMESIZES, struct
v4l2_frmsizeenum *argp)
```

Set the arbitrary resolution output:

```
int ioctl(int fd, VIDIOC_S_FMT, struct v4l2_format *argp)
```

### 4.2.2 GStream example

3840 x 2160:

```
gst-launch-1.0 -v v4l2src device=/dev/video2 ! "video/x-raw,format=YUY2,width=3840,height=2160" ! queue ! waylandsink
```

1920 x 1080:

```
gst-launch-1.0 -v v4l2src device=/dev/video2 ! "video/x-raw,format=YUY2,width=1920,height=1080" ! queue ! waylandsink
```

1280 x 720:

```
gst-launch-1.0 -v v4l2src device=/dev/video2 ! "video/x-raw,format=YUY2,width=1280,height=720" ! queue ! waylandsink
```

### 4.2.3 video\_test example

- Minimum resolution: 176 x 144
- Maximum resolution: 4096 x 3072
- Width alignment: 16 pixels
- Height alignment: 8 pixels

3840 x 2160:

```
./video_test -w 3840 -h 2160 -f YUYV -t 1 -m 0 -d 2
```

1920 x 1080:

```
./video_test -w 1920 -h 1080 -f YUYV -t 1 -m 0 -d 2
```

1280 x 720:

```
./video_test -w 1280 -h 720 -f YUYV -t 1 -m 0 -d 2
```

### 4.3 Crop and scale

- Top alignment: 16 pixels
- Left alignment: 16 pixels
- Width alignment: 16 pixels
- Height alignment: 8 pixels
- Maximum resolution: 4096 x 3072
- Minimum resolution: 176 x 144

The scale up ratio is limited to a maximum of 4.0. There is no limitation for the scale down ratio.

#### 4.3.1 V4L2 API

```
int ioctl(int fd, VIDIOC_G_SELECTION, struct v4l2_selection
*argp)
int ioctl(int fd, VIDIOC_S_SELECTION, struct v4l2_selection
*argp)
```

#### 4.3.2 video\_test example

Crop:

```
./video_test -w 1280 -h 720 -f YUYV -t 1 -m 0 -d 2 -c 1280_720
./video_test -w 640 -h 480 -f YUYV -t 1 -m 0 -d 2 -c 640_480
```

Scale:

```
./video_test -w 3840 -h 2160 -f YUYV -t 1 -m 2 -d 2 -s
./video_test -w 1280 -h 720 -f YUYV -t 1 -m 2 -d 2 -s
```

Crop Then scale:

```
./video_test -w 3840 -h 2160 -f YUYV -t 1 -m 0 -d 2 -c 1280_720
-s
./video_test -w 640 -h 480 -f YUYV -t 1 -m 0 -d 2 -c 1280_720 -
s
```

## 5 Revision History

This table provides the revision history.

### Revision history

Revision number	Date	Substantive changes
L5.4.70_2.3.2	05/2021	Initial release.
LF5.10.35_2.0.0	06/2021	Released for LF5.10.35_2.0.0.
LF5.10.52_2.1.0	10/2021	Major content update for the Linux LF5.10.52_2.1.0 release.
LF5.10.72_2.2.0	12/2021	Updated for the LF5.10.72_2.2.0 release.
LF5.15.5_1.0.0	03/2022	Added Chapter 4 and some minor updates.
LF5.15.32_2.0.0	06/2022	Upgraded to the 5.15.32 kernel, U-Boot 2022.04, and Kirkstone Yocto
LF5.15.52_2.1.0	09/2022	Upgraded to the 5.15.52 kernel, and added the i.MX 93.
LF5.15.71_2.2.0	12/2022	Upgraded to the 5.15.71 kernel.

## 6 Legal information

### 6.1 Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

### 6.2 Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <http://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Suitability for use in non-automotive qualified products** — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

**Security** — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at [PSIRT@nxp.com](mailto:PSIRT@nxp.com)) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

### 6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

**NXP** — wordmark and logo are trademarks of NXP B.V.

## Contents

<b>1</b>	<b>ISP Independent Sensor Interface API</b>	<b>2</b>	1.2.4.41	vvcam_sccb_data_s	17
1.1	Overview	2	1.3	Independent Sensor Interface Functions	17
1.1.1	Acronyms and conventions	2	1.3.1	General API Functions	17
1.2	Independent Sensor Interface API		1.3.2	AEC API Functions	23
	Components	3	1.3.3	AWB API Functions	29
1.2.1	Numeric Data Types	3	1.3.4	Expand API Functions	30
1.2.2	RESULT Return Codes	3	1.3.5	AF API Functions	31
1.2.3	Enumerations	4	1.3.6	Test Pattern API Functions	33
1.2.3.1	IsiBayerPattern_e	4	1.3.7	Miscellaneous API Functions	33
1.2.3.2	IsiColorComponent_e	4	<b>2</b>	<b>Camera Sensor Porting Guide</b>	<b>34</b>
1.2.3.3	IsiExpoFrmType_e	4	2.1	Overview	34
1.2.3.4	IsiFocus_e	5	2.2	ISP Software Architecture	34
1.2.3.5	IsiHdrMode_e	5	2.2.1	ISS (Image Sensor Specific) Driver	35
1.2.3.6	IsiSensorTpgMode_e	5	2.2.2	ISP Sensor Module Block Diagrams	35
1.2.3.7	IsiStitchingMode_e	5	2.3	ISP Independent Sensor Interface (ISI) API	
1.2.4	Structures	6		reference	36
1.2.4.1	IsiCamDrvConfig_t	6	2.3.1	ISI Structures	36
1.2.4.2	IsidualGain_t	6	2.3.1.1	IsiCamDrvConfig_s	36
1.2.4.3	IsidualInt_t	6	2.3.1.2	IsiSensor_t	37
1.2.4.4	IsiFocusCalibAttr_t	7	2.3.1.3	IsiSensorInstanceConfig_s	38
1.2.4.5	IsiFocusPos_t	7	2.3.2	ISI Functions	39
1.2.4.6	IsiLinearGain_t	7	2.3.3	Sensor API Reference	40
1.2.4.7	IsiLinearInt_t	7	2.3.4	ISS Sensor Driver User Space Flow	41
1.2.4.8	IsiQuadGain_t	7	2.4	IOCTL Introduction	42
1.2.4.9	IsiQuadInt_t	7	2.4.1	IOCTL Commands	42
1.2.4.10	IsiSensor_t	8	2.4.2	IOCTL Call Flow	43
1.2.4.11	IsiSensorAelInfo_t	9	2.4.2.1	Native Mode	43
1.2.4.12	IsiSensorBlc_t	10	2.4.2.2	V4L2 Mode	44
1.2.4.13	IsiSensorCaps_t	10	2.5	VVCam API Reference	45
1.2.4.14	IsiSensorContext_t	10	2.5.1	Sensor Driver Enumerations	45
1.2.4.15	IsiSensorExpandCurve_t	11	2.5.1.1	SENSOR_BAYER_PATTERN_E	45
1.2.4.16	IsiSensorGain_t	11	2.5.1.2	sensor_hdr_mode_e	45
1.2.4.17	IsiSensorGain_u	11	2.5.1.3	sensor_stitching_mode_e	45
1.2.4.18	IsiSensorInstanceConfig_t	11	2.5.2	Sensor Driver Structures	45
1.2.4.19	IsiSensorIntTime_t	12	2.5.2.1	sensor_blc_t	45
1.2.4.20	IsiSensorIntTime_u	12	2.5.2.2	sensor_data_compress_t	46
1.2.4.21	IsiSensorIspStatus_t	12	2.5.2.3	sensor_expand_curve_t	46
1.2.4.22	IsiSensorMipiInfo	13	2.5.2.4	sensor_hdr_artio_t	46
1.2.4.23	IsiSensorMode_t	13	2.5.2.5	sensor_mipi_info	46
1.2.4.24	IsiSensorModelInfoArray_t	13	2.5.2.6	sensor_test_pattern_t	47
1.2.4.25	IsiSensorWB_t	13	2.5.2.7	sensor_white_balance_t	47
1.2.4.26	IsiTriGain_t	13	2.5.2.8	vvcam_ae_info_t	47
1.2.4.27	IsiTriInt_t	13	2.5.2.9	vvcam_clk_s	48
1.2.4.28	sensor_blc_t	13	2.5.2.10	vvcam_mode_info_array_t	48
1.2.4.29	sensor_data_compress_t	14	2.5.2.11	vvcam_mode_info_t	48
1.2.4.30	sensor_expand_curve_t	14	2.5.2.12	vvcam_sccb_array_s	49
1.2.4.31	sensor_hdr_artio_t	14	2.5.2.13	vvcam_sccb_cfg_s	49
1.2.4.32	sensor_mipi_info_s	14	2.5.2.14	vvcam_sccb_data_s	49
1.2.4.33	sensor_test_pattern_t	14	2.5.2.15	vvcam_sensor_function_s	49
1.2.4.34	sensor_white_balance_t	15	2.5.3	Sensor Driver API	50
1.2.4.35	vvcam_ae_info_t	15	2.6	Camera Sensor Driver in V4L2 Mode	53
1.2.4.36	vvcam_clk_s	16	2.6.1	VVCAM Flow in V4L2 Mode	53
1.2.4.37	vvcam_mode_info_array_t	16	2.6.1.1	Sensor Driver Software Architecture in	
1.2.4.38	vvcam_mode_info_t	16		V4L2 Mode	53
1.2.4.39	vvcam_sccb_array_s	17	2.6.2	Camera Sensor Porting Setup in V4L2	
1.2.4.40	vvcam_sccb_cfg_s	17		Mode	54

2.6.2.1	Create Sensor DTS File in Kernel .....	54	6	Legal information .....	126
2.6.2.2	Create Sensor V4L2 Driver in VVCAM .....	55			
2.6.2.3	Create Sensor ISI API in ISI Layer .....	62			
2.6.3	Native HDR Mode Porting .....	63			
2.6.4	Sensor Compand Curve .....	66			
2.6.5	Sensor White Balance and Black Level Correction (BLC) .....	70			
2.7	Camera Timing Issue Solution .....	70			
3	<b>ISP Using V4L2 Interface .....</b>	<b>71</b>			
3.1	Overview .....	71			
3.1.1	Requirements/dependencies .....	71			
3.1.2	Supported features .....	71			
3.2	V4L2 API components .....	72			
3.2.1	IOCTL interface and commands .....	72			
3.2.2	IOCTL call flow .....	73			
3.2.3	Buffer API .....	74			
3.2.3.1	Buffer IOCTL control words .....	74			
3.2.3.2	Buffer functions .....	74			
3.2.4	Event API .....	75			
3.2.4.1	Event IOCTL control words .....	75			
3.2.4.2	Event functions .....	75			
3.2.4.3	Private event .....	75			
3.2.5	Feature control API .....	76			
3.2.5.1	String parser .....	76			
3.2.5.2	String transfer .....	77			
3.2.5.3	Feature control words .....	77			
3.2.5.4	Dewarp control words .....	106			
3.2.5.5	Sensor Control Words .....	110			
3.2.5.6	Pipeline Control Words .....	113			
3.3	ISP software V4L2 programming overview ....	114			
3.3.1	General concept .....	114			
3.3.2	V4L2 kernel driver block diagram .....	114			
3.3.3	V4L2 third-party user application and ISP stack communication .....	115			
3.3.4	ISP V4L2 buffer management .....	116			
3.3.5	ISP proprietary software stack .....	117			
3.4	Arbitrary Resolution Control .....	117			
3.4.1	Introduction to Arbitrary Resolution .....	117			
3.4.2	Dewarp Calibration .....	118			
4	<b>ISP Software Arbitrary Resolution Switch Guide .....</b>	<b>120</b>			
4.1	ISP sensor input .....	120			
4.1.1	Sensor size limitation .....	120			
4.1.2	Sensor size configuration in driver .....	121			
4.1.3	Sensor Mode 0, 3840x2160 Linear .....	121			
4.1.4	Sensor Mode 1, 3840x2160 HDR .....	122			
4.1.5	Sensor Mode 2, 1920x1080 Linear .....	122			
4.1.6	Sensor Mode 3, 1920x1080 HDR .....	122			
4.2	Arbitrary resolution output .....	123			
4.2.1	V4L2 API .....	123			
4.2.2	GStream example .....	123			
4.2.3	video_test example .....	123			
4.3	Crop and scale .....	124			
4.3.1	V4L2 API .....	124			
4.3.2	video_test example .....	124			
5	<b>Revision History .....</b>	<b>125</b>			

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.