IMX8MPCDUG

i.MX 8M Plus Camera and Display Guide

Rev. LF6.6.3_1.0.0 — 29 March 2024 **User guide**

Document information

Information	Content
Keywords	i.MX, Linux, LF6.6.3_1.0.0
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 LF6.6.3_1.0.0.

Example code shown in this document are all from Vesilisicon, and thus the copyright belongs to Vesilisicon.

The code are 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)
MRZE	Main Resize Module

Table 1. Acronyms...*continued*

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
`RET_NULL_POINTER`	Callback is a NULL pointer
`RET_OUTOFMEM`	Not enough memory available

RESULT String Values	Description
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
ISI_EXPO_FRAME_TYPE_2FRAMES	1	2 frames exposure type
ISI_EXPO_FRAME_TYPE_3FRAMES	2	3 frames exposure type

Enumeration Values	Value	Description
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

Enumeration Values	Value	Description
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)
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)
*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

Structure Members	Type	Description
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)
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 hdr Ratio[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
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
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 IsiCreateSensorIss IsiSensorHandle_t is typedef void *.
szSensorNodeName[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
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_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
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<<expand_px[i] = expand_x_data[i+1] - 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

Structure Members	Type	Description
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))
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
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
```

IsiCreateSensorIss

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
```

IsiGetSensorModeIss

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
```

IsiSetSensorModeIss

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 (IntegrationTime(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 IsiSensorIntTime_t.

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.

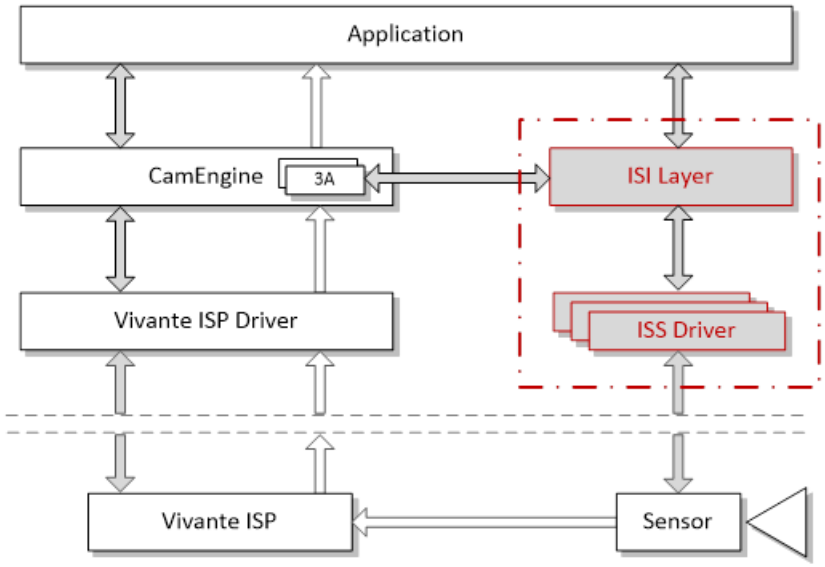


Figure 1. ISP Software Architecture

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 the following figure.

- **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^2C kernel driver.
  - **Sensor Driver**: performs sensor API operations on sensor hardware.
  - **I2C:** 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.
    - The Native mode will not be used from the 6.1 kernel version.

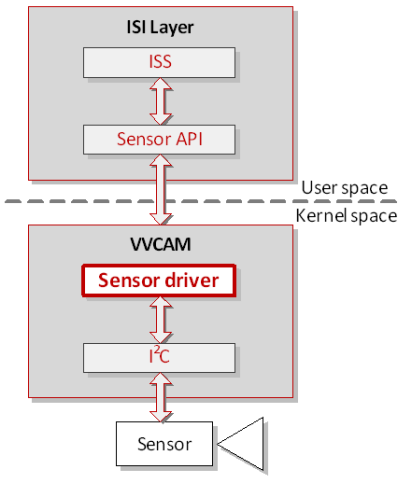


Figure 2. Sensor Module in Linux Kernel

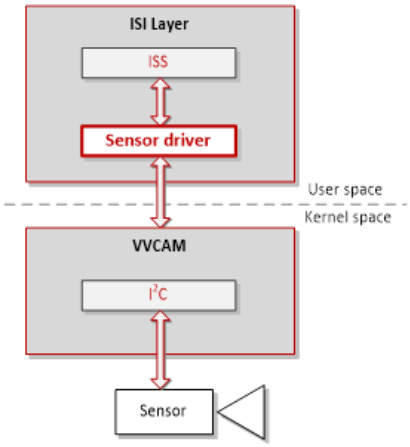


Figure 3. Sensor Module in ISI Layer

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.

Structure Members	Type	Description
*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
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)

Structure Members	Type	Description
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
*pSensor	Section 2.3.1.2	The pointer to the sensor driver interface
hSensor	IsiSensorHandle_t	Sensor handle returned by IsiCreate SensorIss

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

Table 2. ISI functions*...continued*

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

Table 3. Sensor API reference...*continued*

Sensor API	Description
`<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.

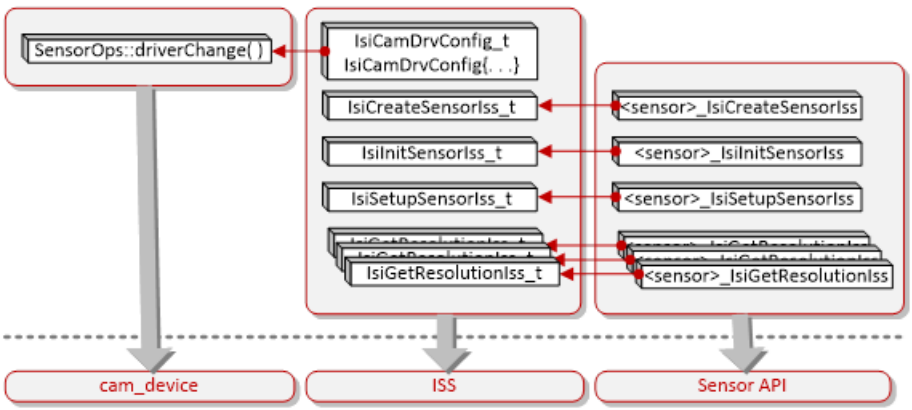


Figure 4. User Space Flow

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 V4L2 Mode as described below.

2.4.2.1 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.

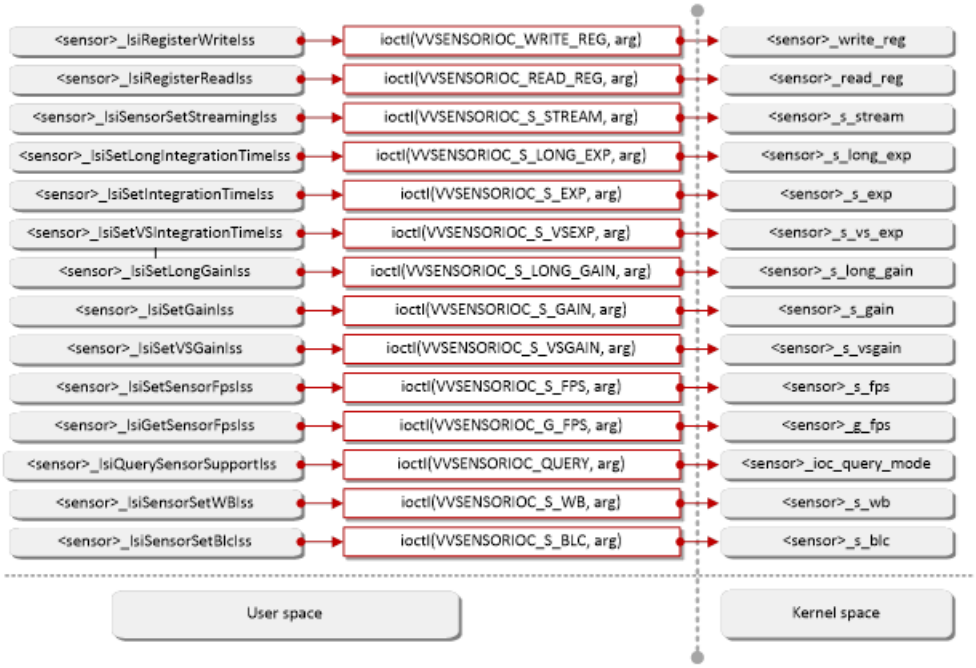


Figure 5. IOCTL Call Flow in V4L2 Mode

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
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<<expand_px[i] = expand_x_data[i+1] - 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)

Structure Members	Type	Description
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)
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	Section 2.5.2.4	HDR radio

2.5.2.9 vvcam_clk_s

Structure Members	Type	Description
status	uint32_t	CLK enable status

Structure Members	Type	Description
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 vvcam_mode_info	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
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	Section 2.5.2.8	AE information
mipi_info	Section 2.5.2.5	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	vvcam_sccb_data_s	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
mipi_info	sensor_mipi_info_s	Feature of MIPI CSI interface: lanes and data width
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

Structure Members	Type	Description
sensor_query	int32_t (*sensor_query) (void *ctx, vvcam_mode_info_array_t *pmode_info_arry)	Function pointer to query sensor information

2.5.3 Sensor Driver API

V4l2 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 struct vvcam_sensor_function_s <sensor>_function	Mounts sensor driver to function pointer
sensor_query(…)	Query sensor information
sensor_write_reg(…)	Write register
sensor_read_reg(…)	Read register
sensor_write_reg_arry(…)	Write register array
sensor_get_chip_id(…)	Get the chip ID in sensor
sensor_init(…)	Initialize the vvcam_mode_info data structure
sensor_set_stream (…)	Turn the flow of the sensor on or off
sensor_set_exp(…)	Write the exposure time of 3A decomposition exposure parameter to the register of the sensor
sensor_set_vs_exp(…)	Write the exposure time of 3A decomposition exposure parameter for a very short exposure frame to the the register of the sensor
sensor_calc_gain(…)	Calculate sensor gain
sensor_set_gain(…)	Set the gain in multiples rather than dB
sensor_set_vs_gain (…)	Set the gain of the very short exposure frame in multiples rather than dB
sensor_set_fps(…)	Set the frame rate of sensor
sensor_set_resolution(…)	Set the resolution of sensor
sensor_set_hdr_mode(…)	Select the HDR mode of sensor

Table 6. Sensor V4l2 Driver API

API Name	Description
<sensor>_g_clk(…)	Get sensor clock
<sensor>_power_on(…)	Power on sensor
<sensor>_power_off(…)	Power off sensor
<sensor>_s_power(…)	Set sensor power
<sensor>_write_reg(…)	Write data to the specified register
<sensor>_read_reg(…)	Read data from the specified register

Table 6. Sensor V4l2 Driver API*...continued*

API Name	Description
<sensor>_write_reg_arry(…)	Write register array
<sensor>_query_capability(…)	Query sensor capability
<sensor>_query_supports(…)	Query sensor support modes
<sensor>_get_sensor_id(…)	Get sensor ID
<sensor>_get_reserve_id(…)	Get reserve sensor ID
<sensor>_get_sensor_mode(…)	Get sensor mode
<sensor>_set_sensor_mode(…)	Set sensor mode
<sensor>_set_lexp(…)	Write the exposure time of 3A decomposition exposure parameter for a long exposure frame to the the register of the sensor
<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

Table 6. Sensor V4l2 Driver API*...continued*

API Name	Description
<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.*

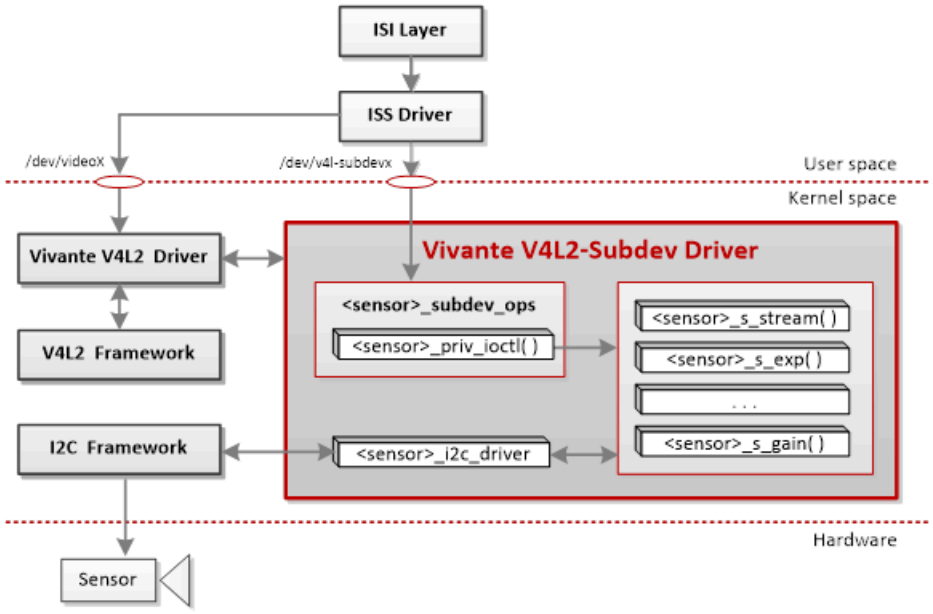


Figure 6. VVCAM Software Architecture in V4L2 Mode

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 <500000000>;
        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 pwdn 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_entiny;
  }
  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_entiny:
  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.

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

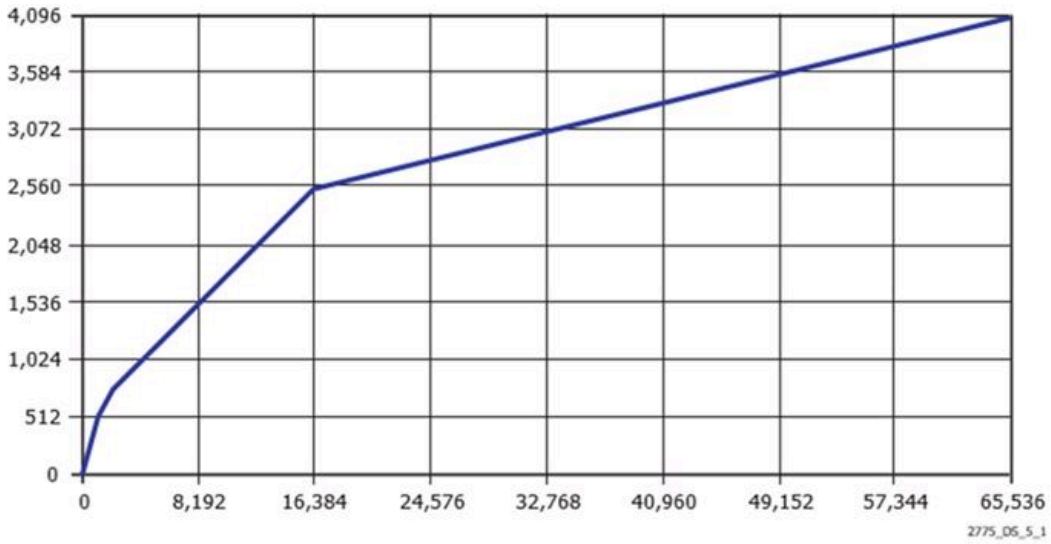


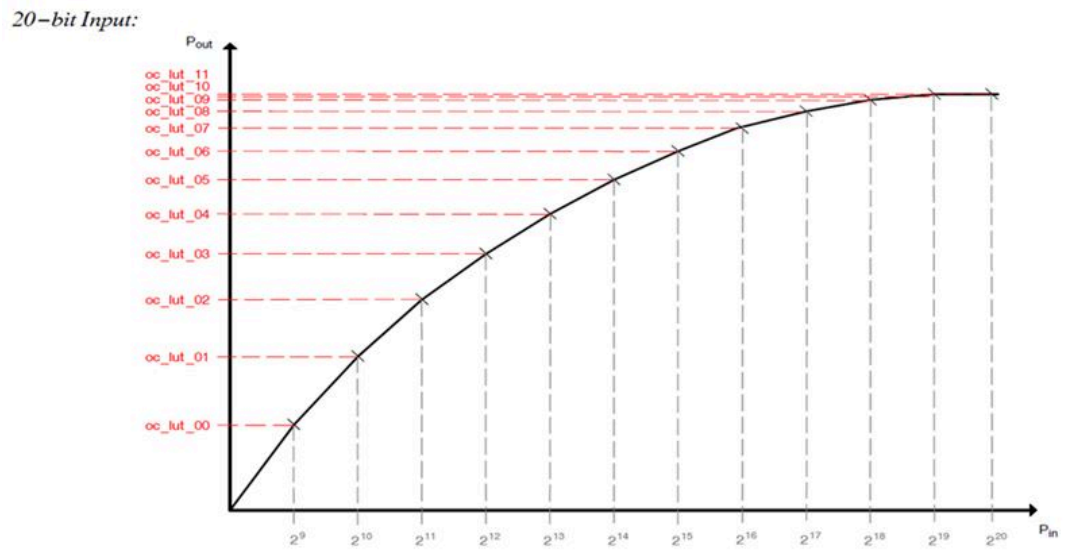
Figure 7. 16-bit to 12-bit PWL compression

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

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

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

Figure 8. 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,};
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};
```

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 though sensor HTS/VTS.

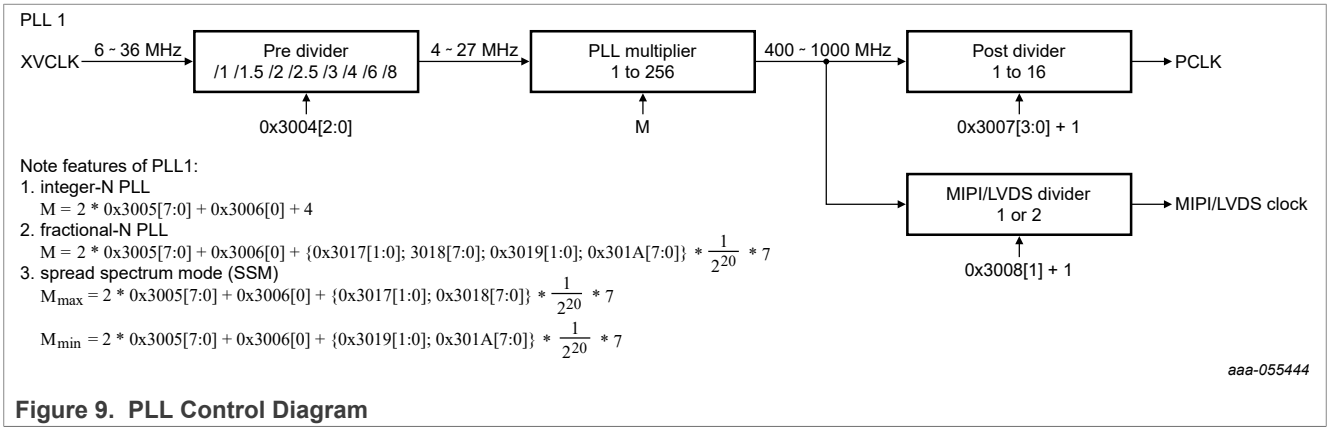


Figure 9. PLL Control Diagram

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
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
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`

Table 8. Key video IOTCLs*...continued*

IOCTL	Type	Description
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
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_CTRLS	.vidioc_g_ext_ctrls	Get feature control commands
VIDIOC_S_EXT_CTRLS	.vidioc_s_ext_ctrls	Set feature control commands

3.2.2 IOCTL call flow

IOTCL call flow is described in Figure 10 and the ISP reference code is based on this implementation.

This flow will be expanded in the future.

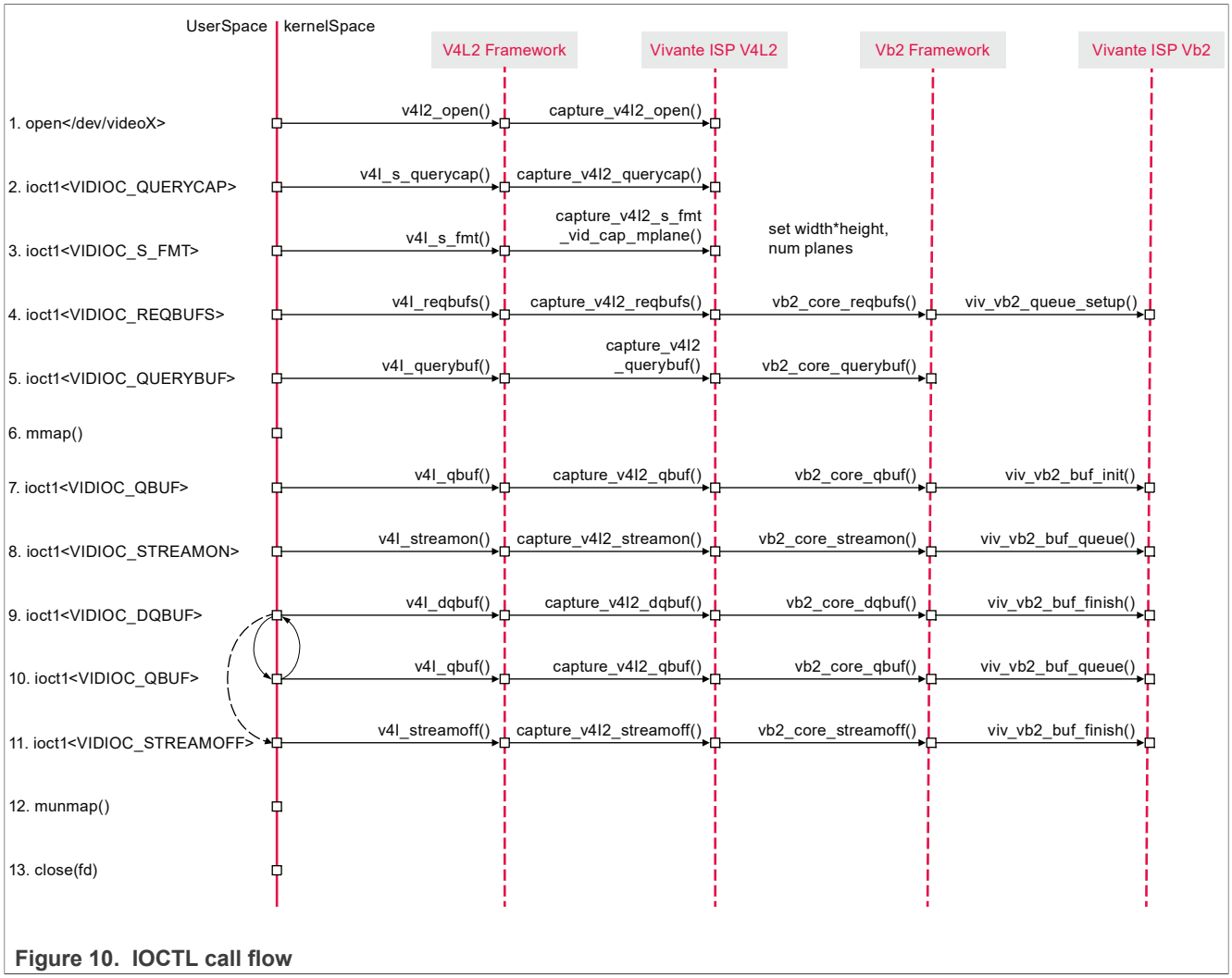


Figure 10. IOCTL call flow

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/v6.1/userspace-api/media/v4l/vidioc-reqbufs.html
- VIDIOC_QUERYBUF
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/vidioc-querybuf.html
- VIDIOC_QBUF
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/vidioc-qbuf.html
- VIDIOC_DQBUF
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/vidioc-qbuf.html
- VIDIOC_STREAMON
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/vidioc-streamon.html
- VIDIOC_STREAMOFF
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/vidioc-streamon.html

3.2.3.2 Buffer functions

- mmap
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/func-mmap.html
- munmap
  Link: https://www.kernel.org/doc/html/v6.1/userspace-api/media/v4l/func-munmap.html
- poll
  Link: https://www.kernel.org/doc/html/v6.1/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
`command`	unsigned int	Extension based on `V4L2_CID_PRIVATE_BASE`
`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 11.

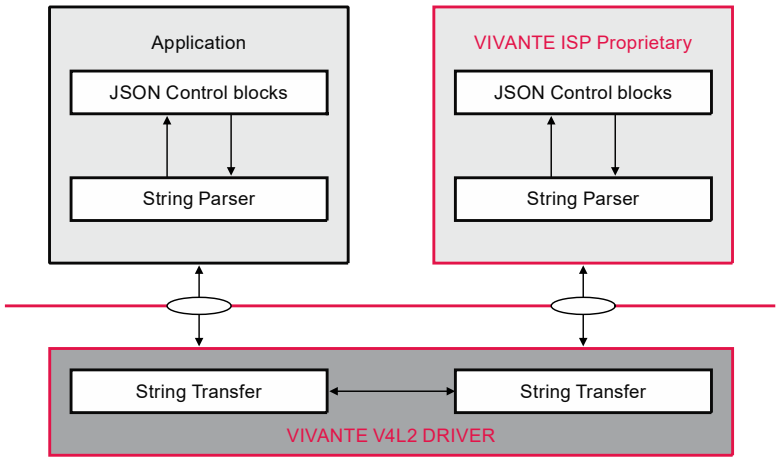


Figure 11. Feature control block diagram

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_CTRLS`
  Link: http://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-g-ext-ctrls.html
- `VIDIOC_S_EXT_CTRLS`
  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
`id`	`__u32`	V4L2 ISP SW feature control words
`size`	`__u32`	String length
`reserved2[1]`	`__u32`	
`value`	`union of __s32`	

Table 10. v4l2_ext_control Structure*...continued*

v4l2_ext_control structure member	Type	Description
value64	union of __s64	
string	union of char *	String transfer pointer
p_u8	union of __u8 *	
p_u16	union of __u16 *	
p_u32	union of __u32 *	
ptr	union of void*	

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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 11. Control words for IF_AE_G_CFG

Control Word	Description	Valid Values
mode	Configuration mode	1: Disabled evaluation 2: Fix evaluation 3: Adaptive evaluation
damping.over	Damping upper limit for luminance over set point. The larger the value, the smoother the convergence	[0.0 … 1.0]
damping.under	Damping lower limit for luminance under set point. The larger the value, the smoother the convergence	[0.0 … 1.0]
set.point	Target luminance point	[0 … 255]
clm.tolerance	Calculation accuracy; AE will make adjustments when the difference ratio between `set.point` and actual point over the `clm.tolerance`	[0 … 100]
weight	Weights of 5x5 blocks	[0 … 16]

  Usage Guide:
  – This macro can be called after turning the stream on.
  – This macro can be called when the AE is either disabled or enabled.
- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

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
`damping.over`	Damping upper limit for luminance over set point. The larger the value, the smoother the convergence	[0.0 … 1.0]
`damping.under`	Damping lower limit for luminance under set point. The larger the value, the smoother the convergence	[0.0 … 1.0]
`set.point`	Target luminance point	[0 … 255]
`clm.tolerance`	Calculation accuracy; AE will make adjustments when the difference ratio between `set.point` and actual point over the `clm.tolerance`	[0 … 100]
`weight`	Weights of 5x5 blocks	[0 … 16]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AE is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AE is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AE is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 15. Control words for IF_AE_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AE is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 16. Control words for IF_AE_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AE is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_AE_RESET**
  This macro definition is identical to the string "ae.reset".
  Description: Resets the Auto Exposure control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 17. Control words for IF_AE_RESET

Control word	Description	Valid Values
N/A	-	-

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the AE is either disabled or enabled and takes effect when enabled.

- **IF_AE_G_STATUS**
  This macro definition is identical to the string "ae.g.status".
  Description: Gets the status of Auto Exposure control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 18. Control words for IF_AE_G_STATUS

Control word	Description	Valid Values
`hist`	Current histogram of image	*Resolution-specific*
`luma`	Mean luminance measured	[0 … 255]
`object.region`	Measurement windows block	[0 … 255]

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the AE is either disabled or enabled.

- **IF_AE_G_ISO**
  This macro definition is identical to the string "ae.g.sensitivity".
  Description: Gets sensitivity of Auto Exposure control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 19. Control words for IF_AE_G_ISO

Control word	Description	Valid Values
`sensitivity`	Sensitivity (ISO) of Auto Exposure control	[100 … 1600]
`max`	Maximum Sensitivity	[100 … 1600]
`min`	Minimum Sensitivity	[100 … 1600]

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the AE is either disabled or enabled.

- **IF_AE_S_ISO**
  This macro definition is identical to the string "ae.s.sensitivity".
  Description: Sets sensitivity of Auto Exposure control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 20. Control words for IF_AE_S_ISO

Control word	Description	Valid Values
`sensitivity`	Sensitivity (ISO) of Auto Exposure control	[100 … 1600]

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called and takes effect only when the AE is disabled.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 21. Control words for IF_AF_G_CFG

Control word	Description	Valid Values
`algorithm`	Selects the search algorithm type	– 1: Full range – 2: Adaptive range – 3: Hill climbing
`oneshot`	Auto focus once and keep the configuration	– true – false
`mode`	Auto focus mode: it will automatically calculate appropriate length when mode is Auto.	– 1: Manual – 2: Auto
`ength`	Position of auto focus module	[0 … 100]
`win.startX`	Measuring window left start position	*Sensor-specific*
`win.startY`	Measuring window top start position	*Sensor-specific*
`win.width`	Measurement window width	*Sensor-specific*
`win.height`	Measurement window height	*Sensor-specific*

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called when the AF is either disabled or enabled.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 22. Control words for IF_AF_S_CFG

Control word	Description	Valid Values
`algorithm`	Selects the search algorithm type	– 1: Full range – 2: Adaptive range – 3: Hill climbing
`oneshot`	Auto focus once and keep the configuration	– true

Table 22. Control words for IF_AF_S_CFG*...continued*

Control word	Description	Valid Values
		– false
`mode`	Auto focus mode: it will automatically calculate appropriate length when mode is Auto.	– 1: Manual – 2: Auto
`ength`	Position of auto focus module	[0 … 100]
`win.startX`	Measuring window left start position	*Sensor-specific*
`win.startY`	Measuring window top start position	*Sensor-specific*
`win.width`	Measurement window width	*Sensor-specific*
`win.height`	Measurement window height	*Sensor-specific*

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the AF is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 23. Control words for IF_AF_G_EN

Control word	Description	Valid Values
`enable`	The state of the Auto Focus	– true – false

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AF is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 24. Control words for IF_AF_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AF is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_AF_G_AVI**

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

  Description: Gets the availability of the Auto Focus control.

Parameters:

– `Json::Value &jRequest (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 25. Control words for IF_AF_G_AVI

Control word	Description	Valid Values
`available`	The availability of the Auto Focus	– true – false

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the AF is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 26. Control Words for IF_AWB_G_CFG

Control word	Description	Valid Values
`mode`	AWB mode; auto mode automatically calculates the appropriate illumination profile.	– 1: Manual – 2: Auto
`index`	The index of the illumination profile; it will affect the AWB adjustment effect	– 0: A – 1: D50 – 2: D65 – 3: F2 (CWF) – 4: F11 (TL84)
`damping`	Changes white balance smoothly through temporal damping	– true – false

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the AWB is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 27. Control Words for IF_AWB_S_CFG

Control word	Description	Valid Values
`mode`	AWB mode; auto mode automatically calculates the appropriate illumination profile.	– 1: Manual – 2: Auto
`index`	The index of the illumination profile; it will affect the AWB adjustment effect	– 0: A – 1: D50 – 2: D65 – 3: F2 (CWF)

Table 27. Control Words for IF_AWB_S_CFG*...continued*

Control word	Description	Valid Values
		– 4: F11 (TL84)
`damping`	Changes white balance smoothly through temporal damping	– true – false

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called only when AWB is disabled and takes effect when AWB is enabled again.

– If mode is set to auto, it will automatically calculate the appropriate index of illumination profile regardless of the illumination index.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 28. Control words for IF_AWB_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AWB is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 29. Control words for IF_AWB_S_EN

Control word	Description	Valid Values
`enable`	Enables or disables Auto White Balance	– true – false

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AWB is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_AWB_RESET**

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

  Description: Resets the Auto White Balance control.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 30. Control words for IF_AWB_RESET

Control word	Description	Valid Values
N/A	-	-

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the AWB is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 31. Control words for IF_AWB_S_MEASWIN

Control word	Description	Valid Values
`left`	Measuring window left start position	*Sensor-specific*
`top`	Measuring window top start position	*Sensor-specific*
`width`	Measuring window width	*Sensor-specific*
`height`	Measuring window height	*Sensor-specific*

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called only when AWB is disabled and takes effect when AWB is enabled again.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

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*

Usage Guide:
– This macro can be called after turning the stream on.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

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*

Usage Guide:
- This macro can be called both before and after turning the stream on.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 34. Control words for IF_CAC_G_EN

Control word	Description	Valid Values
`enable`	The state of the Chromatic Aberration Correction	– true – false

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called when the CAC is either disabled or enabled.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 35. Control words for IF_CAC_S_EN

Control word	Description	Valid Values
`enable`	Enables or disables Chromatic Aberration Correction	– true – false

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called when the CAC is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 36. Control words for IF_CNR_G_CFG

Control Word	Description	Valid Values
`tc1`	The CNR threshold value of the Cb channel. The larger the value, the stronger the noise reduction	[0 … 32767]
`tc2`	The CNR threshold value of the Cr channel. The larger the value, the stronger the noise reduction	[0 … 32767]

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the CNR is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 37. Control words for IF_CNR_S_CFG

Control word	Description	Valid Values
`tc1`	The CNR threshold value of the Cb channel. The larger the value, the stronger the noise reduction	[0 … 32767]
`tc2`	The CNR threshold value of the Cr channel. The larger the value, the stronger the noise reduction	[0 … 32767]

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called when the CNR is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 38. Control Words for IF_CNR_G_EN

Control Word	Description	Valid Values
`enable`	The state of the Chroma Noise Reduction control	– true – false

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the CNR is either disabled or enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 39. Control Words for IF_CNR_S_EN

Control Word	Description	Valid Values
`enable`	Enables or disables Chroma Noise Reduction control	– true – false

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the CNR is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

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.
`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.

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the CPROC is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

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.
`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.

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the CPROC is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 42. Control words for IF_CPROC_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the CPROC is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 43. Control words for IF_CPROC_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the CPROC is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_CPROC_S_COEFF**

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

  Description: Sets index of coefficient configurations for the Color Processing control.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 44. Control words for IF_CPROC_S_COEFF

Control word	Description	Valid Values
`index`	Index of coefficient configurations for CPROC	[0, 1]

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the CPROC is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 45. Control words for IF_DEMOSAIC_G_CFG

Control word	Description	Valid Values
`mode`	Demosaic mode	1: Normal 2: Bypass
`threshold`	Demosaic threshold	[0..255] – 0: Maximum edge sensitivity – 255: No texture detection

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the DEMOSAIC is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 46. Control words for IF_DEMOSAIC_S_CFG

Control word	Description	Valid Values
`mode`	Demosaic mode	1: Normal 2: Bypass
`threshold`	Demosaic threshold	[0..255] – 0: Maximum edge sensitivity – 255: No texture detection

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the DEMOSAIC is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 47. Control words for IF_DEMOSAIC_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the DEMOSAIC is either disabled or enabled.

- **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 (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 48. Control words for IF_DEMOSAIC_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the DEMOSAIC is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.
– If DEMOSAIC is disabled, it will output a black and white image.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 49. Control words for IF_DPCC_G_EN

Control word	Description	Valid Values
`enable`	The state of the Defect Pixel Cluster Correction	– true – false

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the DPCC is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 50. Control words for IF_DPCC_S_EN

Control word	Description	Valid Values
`enable`	Enables or disables Defect Pixel Cluster Correction	– true – false

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the DPCC is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 51. 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`	The spatial filter's sigma of the green channel	[1 … 255]
`sigma.red.blue`	The spatial filter's sigma of the Red/Blue channel	[1 … 255]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the DPF is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 52. 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`	The spatial filter's sigma of the green channel	[1 … 255]
`sigma.red.blue`	The spatial filter's sigma of the Red/Blue channel	[1 … 255]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called only when the DPF is disabled and takes effect when enabled again.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 53. Control words for IF_DPF_G_EN

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

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called when the DPF is either disabled or enabled.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 54. Control words for IF_DPF_S_EN

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

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called when the DPF is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 55. 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*

Usage Guide:
- This macro can be called after turning the stream on.
- This macro can be called when the AE is either disabled or enabled.

- **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 (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 56. Control words for IF_EC_S_CFG

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

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called and takes effect only when the AE is disabled.

- **IF_EC_G_STATUS**
  This macro definition is identical to the string "ec.g.status".
  Description: Gets the status of Exposure control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 57. Control words for IF_EC_G_STATUS

Control Word	Description	Valid Values
`gain`	Exposure gain range	Node description string
`gain.max`	Sub-node parameter, Maximum gain	*sensor-specific*
`gain.min`	Sub-node parameter, Minimum gain	*sensor-specific*
`gain.step`	Sub-node parameter, Reserved for later use	*sensor-specific*
`time`	Exposure time range	Node description string
`time.max`	Sub-node parameter, Maximum integration time	*sensor-specific*
`time.min`	Sub-node parameter, Minimum integration time	*sensor-specific*
`time.step`	Sub-node parameter, Reserved for later use	*sensor-specific*

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AE is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 58. Control words for IF_FILTER_G_CFG

Control word	Description	Valid Values
`auto`	Auto control; automatically adjusts denoise and sharpen values.	– true – false
`denoise`	Denoise level	[0..10]
`sharpen`	Sharpen level	[0..10]
`chrH`	Chroma filter horizontal mode	[0,1,2,3]
`chrV`	Chroma filter vertical mode	[0,1,2,3]

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the FILTER is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 59. Control words for IF_FILTER_S_CFG

Control word	Description	Valid Values
`auto`	Auto control; automatically adjusts denoise and sharpen values.	– true – false
`denoise`	Denoise level	[0..10]
`sharpen`	Sharpen level	[0..10]
`chrH`	Chroma filter horizontal mode	[0,1,2,3]
`chrV`	Chroma filter vertical mode	[0,1,2,3]

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called only when the FILTER is disabled and takes effect when enabled again.
– If auto is set to true, it will automatically calculate the appropriate value of denoise and sharpen.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 60. Control words for IF_FILTER_G_EN

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

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the FILTER is either disabled or enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 61. Control words for IF_FILTER_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the FILTER is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_FILTER_S_TBL**

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

  Description: Sets the Filter control table for driver automatic adjustments.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 62. Control words for IF_FILTER_S_TBL

Control word	Description	Valid Values
`table`	Filter auto table	Node description string
`columns`	Sub-node description string, table's column property	["HDR"; "Gain"; "Integration Time"; "Denoising"; "Sharpening"]
`rows`	Sub-node description string, values of properties in the table	HDR: enable/disable HDR Gain: *sensor-specific* Integration Time: *sensor-specific* Denoising: [0..10] Sharpening:[0..10]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called only when the FILTER is disabled and takes effect when enabled again and FILTER's auto is true.

- **IF_FILTER_G_TBL**

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

  Description: Gets the Filter control table.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 63. Control words for IF_FILTER_G_TBL

Control word	Description	Valid Values
`table`	Filter auto table	Node description string
`columns`	Sub-node description string, table's column property	["HDR"; "Gain"; "Integration Time";

Table 63. Control words for IF_FILTER_G_TBL*...continued*

Control word	Description	Valid Values
		"Denoising"; "Sharpening"]
`rows`	Sub-node description string, values of properties in the table	HDR: enable/disable HDR Gain: *sensor-specific* Integration Time: *sensor-specific* Denoising: [0..10] Sharpening: [0..10]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the FILTER is either disabled or enabled.

- **IF_FILTER_G_STATUS**

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

  Description: Gets the status of Filter control table.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 64. Control words for IF_FILTER_G_STATUS

Control word	Description	Valid Values
`gain`	Sensor gain	*sensor-specific*
`Integration.time`	Sensor integration time	*sensor-specific*

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the FILTER is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 65. Control words for IF_GC_G_CURVE

Control word	Description	Valid Values
`gc.curve`	Gamma curve; data array length is 17	[0 … 1023]
`gc.mode`	Selects the gamma segmentation mode. Logarithmic: logarithmic segmentation from 0 to 4095, (64,64,64,64,128,128,128,128,256,256,256,512, 512,512,512,512) Equidistant: equidistant segmentation from 0 to 4095, (256, 256, ...); all 16 segments are 256.	– 1: logarithmic mode – 2: equidistant mode

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the GC is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 66. Control words for IF_GC_S_CURVE

Control word	Description	Valid Values
`gc.curve`	Gamma curve; data array length is 17	[0 … 1023]
`gc.mode`	Selects the gamma segmentation mode. Logarithmic: logarithmic segmentation from 0 to 4095, (64,64,64,64,128,128,128,128,256,256,256,512, 512,512,512,512) Equidistant: equidistant segmentation from 0 to 4095, (256, 256, ...); all 16 segments are 256.	– 1: logarithmic mode – 2: equidistant mode

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called when the GC is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 67. Control words for IF_GC_G_CFG

Control word	Description	Valid Values
`gc.mode`	Selects the gamma segmentation mode. Logarithmic: logarithmic segmentation from 0 to 4095, (64,64,64,64,128,128,128,128,256,256,256,512,512,512,512,512) Equidistant: equidistant segmentation from 0 to 4095, (256, 256, ...); all 16 segments are 256.	– 1: logarithmic mode – 2: equidistant mode
`gc.curve`	Gamma curve; data array length is 17	[0 … 1023]

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the GC is either disabled or enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 68. Control words for IF_GC_G_CFG

Control word	Description	Valid Values
`gc.mode`	Selects the gamma segmentation mode. Logarithmic: logarithmic segmentation from 0 to 4095, (64,64,64,64,128,128,128,128,256,256,256,512, 512,512,512,512) Equidistant: equidistant segmentation from 0 to 4095, (256, 256, ...); all 16 segments are 256.	– 1: logarithmic mode – 2: equidistant mode
`gc.curve`	Gamma curve; data array length is 17	[0 … 1023]

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the GC is either disabled or enabled and takes effect when enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 69. Control words for IF_GC_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the GC is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 70. Control words for IF_GC_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the GC is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 71. Control words for IF_HDR_G_CFG

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

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the HDR is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 72. Control words for IF_HDR_S_CFG

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

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the HDR is either disabled or enabled and takes effect when enabled and the sensor mode is HDR mode.
– `extension.bit` is not valid for NXP.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 73. Control words for IF_HDR_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.
– This macro can be called when the HDR is either disabled or enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 74. Control words for IF_HDR_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the HDR is either disabled or enabled and takes effect when the setting value is the opposite of the previous value, and the sensor mode is HDR mode.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 75. Control words for IF_LSC_G_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the LSC is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 76. Control words for IF_LSC_S_EN

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

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the LSC is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_LSC_G_STATUS**

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

  Description: Gets the status of Lens Shade Correction control.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 77. Control words for IF_LSC_G_STATUS

Control Word	Description	Valid Values
`map:red`	LSC gains red	[0 … 3.999]
`map:green.r`	LSC gains green.r	[0 … 3.999]
`map:green.b`	LSC gains green.b	[0 … 3.999]
`map:blue`	LSC gains blue	[0 … 3.999]
`map:xsize`	Horizontal orientation block size	[10 … 1024]
`map:ysize`	Vertical orientation block size	[8 … 1024]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the LSC is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 78. Control words for IF_WDR_G_CFG

Control Word	Property	Description	Valid Values
`generation`	Request	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3
`y.m`	Response	WDR1 curve Ym value	[tone mapping curve values]
`d.y`		WDR1 curve dY value	[tone mapping curve values]
`auto`		WDR3 running mode	– true – false
`auto.level`		WDR3 auto level	[0, 100]
`strength`		WDR2 or WDR3 strength	[0, 128]
`gain.max`		WDR3 gain max	[0, 128]
`strength.global`		WDR3 global strength, image global contrast	[0, 128]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the WDR is either disabled or enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 79. Control words for IF_WDR_S_CFG

Control Word	Description	Valid Values
`generation`	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3
`y.m`	WDR1 curve Ym value	[tone mapping curve values]
`d.y`	WDR1 curve dY value	[tone mapping curve values]
`auto`	WDR3 running mode	– true – false
`auto.level`	WDR3 auto level	[0, 100]
`strength`	WDR2 or WDR3 strength	[0, 128]
`gain.max`	WDR3 gain max	[0, 128]
`strength.global`	WDR3 global strength, image global contrast	[0, 128]

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called only when the module is disabled and takes effect when enabled again.
– If auto is set to true, it will automatically calculate the appropriate strength, `gain.max`, and `strength.global` parameter based on auto table.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 80. Control words for IF_WDR_G_EN

Control Word	Property	Description	Valid Values
`enable`	Response	The state of WDR	– true – false
`generation`	Request	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3

Usage Guide:
– This macro can be called after turning the stream on.
– This macro can be called when the WDR is either disabled or enabled.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 81. Control words for IF_WDR_S_EN

Control Word	Description	Valid Values
`enable`	Enables or disables WDR	– true – false
`generation`	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the WDR is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **IF_WDR_RESET**

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

  Description: Resets the WDR control.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 82. Control words for IF_WDR_RESET

Control word	Description	Valid Values
`generation`	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the WDR is either disabled or enabled and takes effect when enabled.

- **IF_WDR_G_STATUS**

  This macro definition is identical to the string "wdr.g.status".

  Description: Gets the gain and integration time values for the WDR control.

  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 83. Control words for IF_WDR_G_STATUS

Control word	Property	Description	Valid Values
`gain`	Response	WDR gain	*sensor-specific*
`Integration.time`		WDR integration time	*sensor-specific*
`generation`	Request	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the WDR is either disabled or enabled.

- **IF_WDR_S_TBL**

This macro definition is identical to the string "wdr.s.tb".
Description: Sets the table of the WDR control for driver automatic adjustments.
Parameters:

- `Json::Value &jRequest (input parameter included control words)`
- `Json::Value &jResponse (output parameter included return value)`

Table 84. Control words for IF_WDR_S_TBL

Control word	Description	Valid Values
`table`	WDR table	Node description string
`generation`	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3
`columns`	Sub-node description string, table's column property	["HDR"; "Gain"; "Integration Time"; "Strength"; "Max Gain"; "Global Curve"]
`rows`	Sub-node description string, values of properties in the table	HDR: enable/disable HDR Gain: *sensor-specific* Integration Time: *sensor-specific* Strength: [0, 128] Max Gain: [0, 128] Global Curve: [0, 128]

Usage Guide:

- This macro can be called after turning the stream on.
- This macro can be called only when the WDR is disabled and takes effect when enabled again and WDR auto is true.

- **IF_WDR_G_TBL**
  This macro definition is identical to the string "wdr.g.tbl".
  Description: Gets the table of the WDR control.
  Parameters:
  - `Json::Value &jRequest (input parameter included control words)`
  - `Json::Value &jResponse (output parameter included return value)`

Table 85. Control words for IF_WDR_G_TBL

Control word	Property	Description	Valid Values
`generation`	Request	WDR generation	– 0: GWDR (not supported for NXP) – 1: WDR2 (not supported for NXP) – 2: WDR3
`table`	Response	WDR table	Node description string
`columns`	Response	Sub-node description string, table's column property	["HDR"; "Gain"; "Integration Time"; "Strength"; "Max Gain"; "Global Curve"]

Table 85. Control words for IF_WDR_G_TBL*...continued*

Control word	Property	Description	Valid Values
`rows`	Response	Sub-node description string, values of properties in the table	HDR: enable/disable HDR Gain: *sensor-specific* Integration Time: *sensor-specific* Strength: [0, 128] Max Gain: [0, 128] Global Curve: [0, 128]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the WDR is either disabled or enabled.

- **IF_WB_G_CFG**
  This macro definition is identical to the string "wb.g.cfg".
  Description: Gets the configuration values for the WB control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 86. Control words for IF_WB_G_CFG

Control Word	Description	Valid Values
`matrix`	Color correction Matrix (X-Talk)	[<9 matrix values>], [-8, 7.992]
`offset`	[red, green, blue] offset	[<3 matrix values>], [-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]

Usage Guide:

– This macro can be called after turning the stream on.

– This macro can be called when the AWB is either disabled or enabled.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 87. Control words for IF_WB_S_CFG

Control Word	Description	Valid Values
`matrix`	Color correction Matrix (X-Talk)	[<9 matrix values>], [-8, 7.992]
`offset`	[red, green, blue] offset	[<3 matrix values>], [-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]

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called and takes effect only when the AWB is disabled.

- **IF_WB_S_GAIN**
  This macro definition is identical to the string "wb.s.gain".
  Description: Sets the gain values for the WB control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 88. Control words for IF_WB_S_GAIN

Control Word	Description	Valid Values
`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]

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called and takes effect only when the AWB is disabled.

- **IF_WB_S_OFFSET**
  This macro definition is identical to the string "wb.s.offset".
  Description: Sets the offset values for the WB control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 89. Control words for IF_WB_S_OFFSET

Control Word	Description	Valid Values
`offset`	[red, green, blue] offset	[<3 matrix values>], [-2048, 2047]

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called and takes effect only when the AWB is disabled.

- **IF_WB_S_CCM**
  This macro definition is identical to the string "wb.s.ccm".
  Description: Sets the ccm values for the WB control.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 90. Control words for IF_WB_S_CCM

Control Word	Description	Valid Values
`matrix`	Color correction Matrix (X-Talk)	[<9 matrix values>], [-8, 7.992]

Usage Guide:
– This macro can be called both before and after turning the stream on.
– This macro can be called and takes effect only when the AWB is disabled.

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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 91. Control Words for IF_DWE_S_PARAMS

Control Word	Description	Valid Values
`dwe`	Get the dewarp node	Node description string
`mode`	Sub-node parameter: Dewarp type	– 1: lens distortion – 2: fisheye expand – 4: split screen (not supported) – 8: fisheye dewarp
`hflip`	Sub-node parameter: Set horizontal flip true or false	– true – false
`vflip`	Sub-node parameter: Set vertical flip true or false	– true – false
`bypass`	Sub-node parameter: Bypass dewarp true or false	– true – false
`mat`	Sub-node parameter: Camera matrix [0~8], Distortion coefficient [9~16]	Need calibration

  Usage Guide:
  – This macro can be called both before and after turning the stream on.
  – This macro can be called when the DWE bypass is either disabled or enabled.
- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 92. Control words for IF_DWE_G_PARAMS

Control Word	Description	Valid Values
`dwe`	Get the dewarp node	Node description string
`mode`	Sub-node parameter: Dewarp type	– 1: lens distortion – 2: fisheye expand – 4: split screen (not supported) – 8: fisheye dewarp
`hflip`	Sub-node parameter: Set horizontal flip true or false	– true – false
`vflip`	Sub-node parameter: Set vertical flip true or false	– true

Table 92. Control words for IF_DWE_G_PARAMS*...continued*

Control Word	Description	Valid Values
		– false
`bypass`	Sub-node parameter: Bypass dewarp true or false	– true – false
`mat`	Sub-node parameter: Camera matrix [0~8], Distortion coefficient [9~16]	Need calibration

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the DWE bypass is either disabled or enabled.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 93. Control words for IF_DWE_S_HFLIP

Control Word	Description	Valid Values
`dwe`	Get the dewarp node	Node description string
`hflip`	Sub-node parameter: Set vertical flip	– true – false

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the DWE bypass is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 94. Control words for IF_DWE_S_VFLIP

Control Word	Description	Valid Values
`dwe`	Get the dewarp node	Node description string
`vflip`	Sub-node parameter: Set horizontal flip	– true – false

Usage Guide:

– This macro can be called both before and after turning the stream on.

– This macro can be called when the DWE bypass is either disabled or enabled and takes effect when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 95. Control words for IF_DWE_S_BYPASS

Control word	Description	Valid Values
`dwe`	Get the dewarp node	Node description string
`bypass`	Sub-node parameter: Bypass dewarp true or false	– true – false

Usage Guide:

– This macro can be called both before and after turning the stream on.
– It can take effect only when the setting value is the opposite of the previous value.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 96. Control words for IF_DWE_S_MODE

Control word	Description	Valid Values
`mode`	Sub-node parameter: Sensor mode index	*Sensor-specific*
`dwe`	Get the dewarp node	– true – false

Usage Guide:

– This macro can be called before turning the stream on.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 97. Control words for IF_DWE_S_MAT

Control word	Description	Valid Values
`dwe`	Get the dewarp node	Node description string
`mat`	Sub-node parameter: Camera matrix [0~8], Distortion coefficient [9~16]	Need calibration

Usage Guide:

– This macro can be called both before and after turning the stream on.
– This macro can be called when the DWE bypass is either disabled or enabled and takes effect when bypass is disabled.

- **IF_DWE_S_TYPE**
  This macro definition is identical to the string "dwe.s.type".
  Description: Sets the Dewarp type.

Parameters:

– `Json::Value &jRequest (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 98. Control words for IF_DWE_S_TYPE

Control word	Description	Valid Values
`type`	Sub-node parameter: Dewarp type	1: lens distortion 2: fisheye expand 4: split screen (not supported) 8: fisheye dewarp
`dwe`	Get the dewarp node	Node description string

Usage Guide:

– This macro can be called both before and after turning the stream on.
– This macro can be called when the DWE bypass is either disabled or enabled and takes effect when bypass is disabled.

- **VIV_V4L_DWE_SET_CROP**
  Description: Crops the image.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 99. 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*
`bounds_width`	Real width of sensor input	*Sensor-specific*
`bounds_height`	Real height of sensor input	*Sensor-specific*

Usage Guide:

– This macro can be called before turning the stream on.

- **VIV_V4L_DWE_SET_SCALE**
  Description: Scales the image.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 100. 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*
`start_y`	The starting position of the Y-axis	*Sensor-specific*
`width`	Scale width	*Sensor-specific*

Table 100. Control words for VIV_V4L_DWE_SET_SCALE...*continued*

Control Word	Description	Valid Values
`height`	Scale height	*Sensor-specific*

Usage Guide:

– This macro can be called before turning the stream on.

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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 101. 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*
`size:bounds_width`	Real width of sensor input	*Sensor-specific*
`size:bounds_height`	Real height of sensor output	*Sensor-specific*
`size:top`	Beginning of valid pixel in top position	*Sensor-specific*
`size:left`	Beginning of valid pixel in left position	*Sensor-specific*
`size:width`	Valid width pixels of sensor input	*Sensor-specific*
`size:height`	Valid height pixels of sensor input	*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*

Usage Guide:

– This macro can be called both before and after turning the stream on.

- **IF_SENSOR_G_MODE**
  This macro definition is identical to the string "sensor.g.mode".
  Description: Gets the current mode of the sensor.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 102. Control words for IF_SENSOR_G_MODE

Control word	Description	Valid Values
`cursensormode`	Sensor current mode index	*Sensor-specific*

Table 102. Control words for IF_SENSOR_G_MODE...*continued*

Control word	Description	Valid Values
`maxsensormode`	Sensor max index	*Sensor-specific*

Usage Guide:

– This macro can be called after turning the stream on.

- **IF_SENSOR_S_MODE**
  This macro definition is identical to the string "sensor.s.mode".
  Description: Sets the sensor mode.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 103. Control words for IF_SENSOR_S_MODE

Control word	Description	Valid Values
`cursensormode`	Sensor current mode index	*Sensor-specific*

Usage Guide:

– This macro can be called before turning the stream on.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 104. Control words for IF_SENSOR_G_RESW

Control word	Description	Valid Values
`resw`	Resolution width	*Sensor-specific*

Usage Guide:

– This macro can be called after turning the stream on.

- **IF_SENSOR_G_RESH**
  This macro definition is identical to the string "sensor.g.resh".
  Description: Gets the sensor resolution height.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 105. Control words for IF_SENSOR_G_RESH

Control word	Description	Valid Values
`resh`	Resolution height	*Sensor-specific*

Usage Guide:

– This macro can be called after turning the stream on.

- **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 (input parameter included control words)`

– `Json::Value &jResponse (output parameter included return value)`

Table 106. Control words for IF_SENSOR_G_REG

Control word	Property	Description	Valid Values
`value`	Response	Register value	*Sensor-specific*
`address`	Request	Register address	*Sensor-specific*

Usage Guide:

– This macro can be called both before and after turning the stream on.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 107. Control words for IF_SENSOR_S_REG

Control word	Description	Valid Values
`value`	Register value	*Sensor-specific*
`address`	Register address	*Sensor-specific*

Usage Guide:

– This macro can be called both before and after turning the stream on.

- **IF_S_FPS**

  This macro definition is identical to the string "s.fps".
  Description: Sets the sensor frame rate.
  Parameters:

  – `Json::Value &jRequest (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 108. Control words for IF_S_FPS

Control word	Description	Valid Values
`fps`	Sensor frame rate	*Sensor-specific*

Usage Guide:

– This macro can be called both before and after turning the stream on.

- **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 (input parameter included control words)`

  – `Json::Value &jResponse (output parameter included return value)`

Table 109. Control words for IF_SENSOR_LIB_PRELOAD

Control word	Description	Valid Values
N/A	-	-

Usage Guide:
– This macro can be called before turning the stream on.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 110. Control words for IF_SENSOR_G_SEC

Control Word	Description	Valid Values
`exposure`	AE start exposure = IntegrationTime x Gain	Float value

Usage Guide:
– This macro can be called after turning the stream on.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 111. Control words for IF_SENSOR_S_SEC

Control Word	Description	Valid Values
`exposure`	AE start exposure = IntegrationTime x Gain	Float value

Usage Guide:
– This macro can be called before turning the stream on.

- **IF_SENSOR_S_TESTPAT**
  This macro definition is identical to the string "sensor.s.testpat".
  Description: Sets sensor test pattern.
  Parameters:
  – `Json::Value &jRequest (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 112. 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

Usage Guide:
– This macro can be called both before and after turning the stream on.

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, control pipeline on/off.

Parameters:

– `Json::Value &jRequest (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 113. Control words for IF_PIPELINE_S_WARM_UP

Control Word	Description	Valid Values
`enable`	Control pipeline on/off	– true – false

Usage Guide:

– This macro can be called after turning the stream on.

- **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 (input parameter included control words)`
  – `Json::Value &jResponse (output parameter included return value)`

Table 114. Control words for IF_PIPELINE_S_DWE_ONOFF

Control Word	Description	Valid Values
`enable`	Enable or disable dewarp.	– true – false

Usage Guide:

– This macro can be called before turning the stream on.

- **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.
- **IF_PIPELINE_G_3A_LOCK**
  This macro definition is identical to the string "pipeline.g.3a.lock".
  Description: Gets pipeline AF/AE/AWB lock status.

Parameters:
– `Json::Value &jRequest (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 115. Control words for IF_PIPELINE_G_3A_LOCK

Control Word	Description	Valid Values
`lock.status`	3A lock status	[0 … 7] Bit 0: AF lock Bit 1: AE lock Bit 2: AWB lock

Usage Guide:
– This macro can be called after turning the stream on.

IF_PIPELINE_S_3A_LOCK
This macro definition is identical to the string "pipeline.s.3a.lock".
Description: Sets pipeline AF/AE/AWB lock status.
Parameters:
– `Json::Value &jRequest (input parameter included control words)`
– `Json::Value &jResponse (output parameter included return value)`

Table 116. Control words for IF_PIPELINE_S_3A_LOCK

Control Word	Description	Valid Values
`lock.status`	3A lock status	[0 … 7] Bit 0: AF lock Bit 1: AE lock Bit 2: AWB lock

Usage Guide:
– This macro can be called after turning the stream on.

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 12.

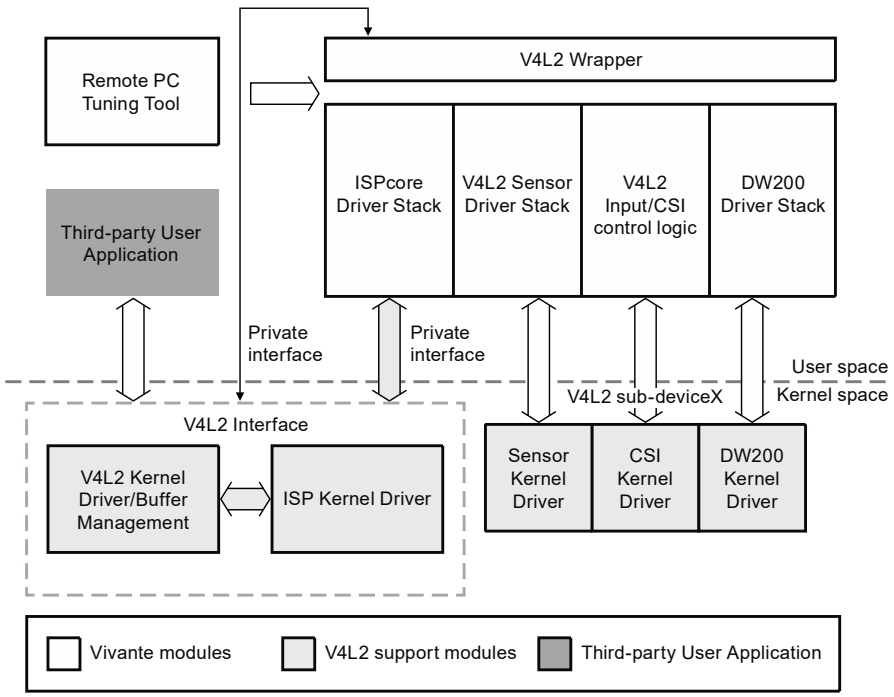


Figure 12. ISP software V4L2 stack

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 117. ISP device nodes

Device node/driver	Description
`/dev/mediax`	Enumerate video devices and subdevices.
`/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 `V4L2_device` and `video_device` and implement the operational functions in the `video_device and vb2_queue`
ISP kernel driver	ISP kernel driver, implements read/write registers, and so on.

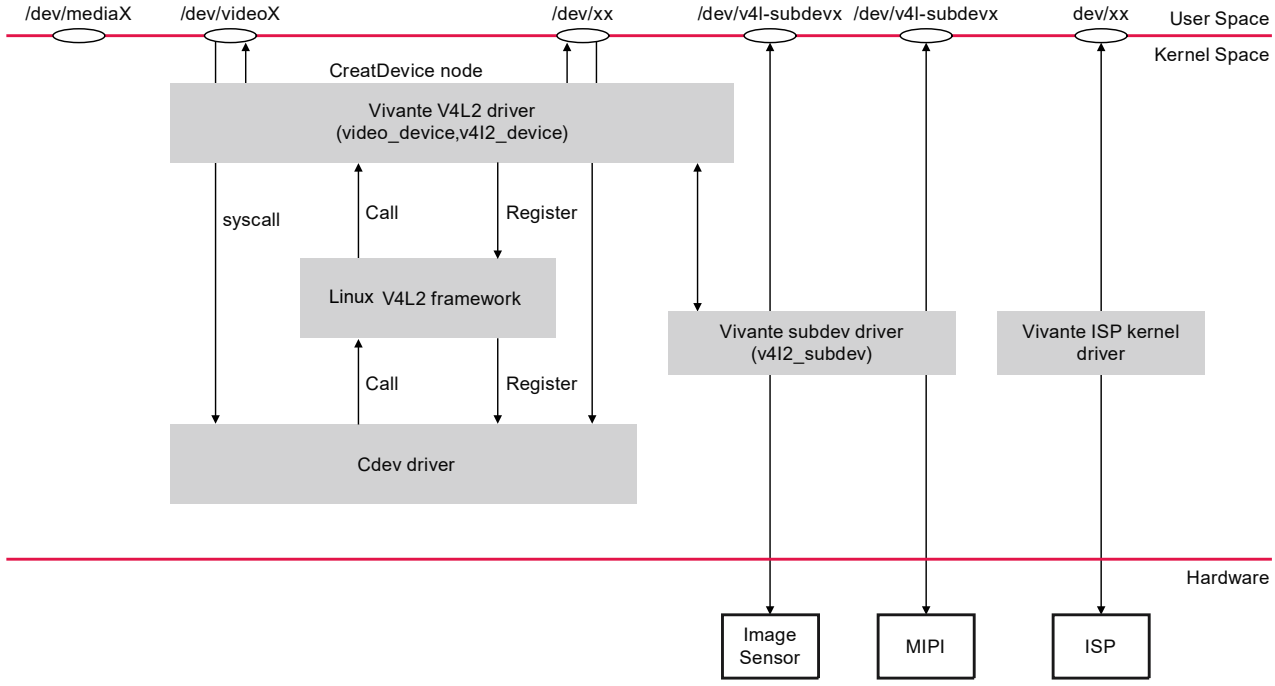


Figure 13. V4L2 kernel driver block diagram

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.

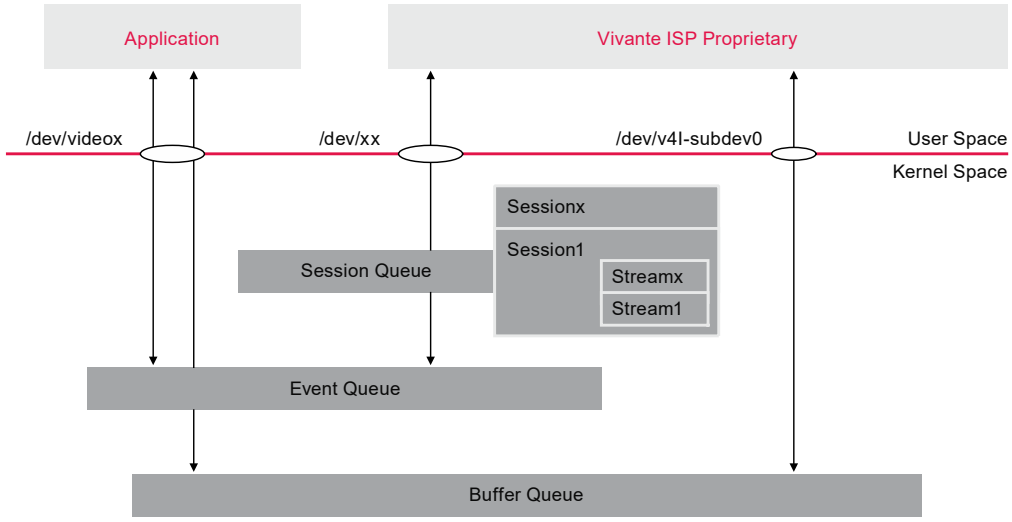


Figure 14. User application communication to ISP proprietary software stack

3.3.4 ISP V4L2 buffer management

There are three memory types as described in Table 118 and Figure 15.

Table 118. 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

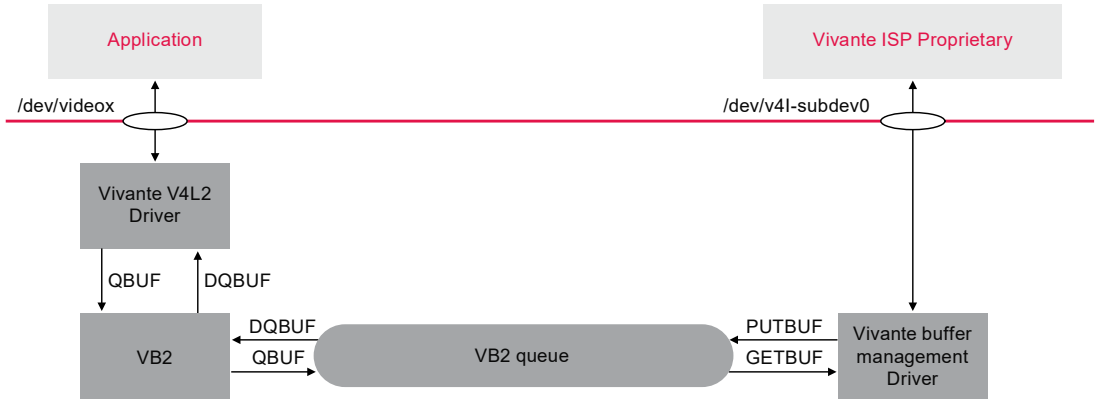


Figure 15. ISP V4L2 buffer management

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.

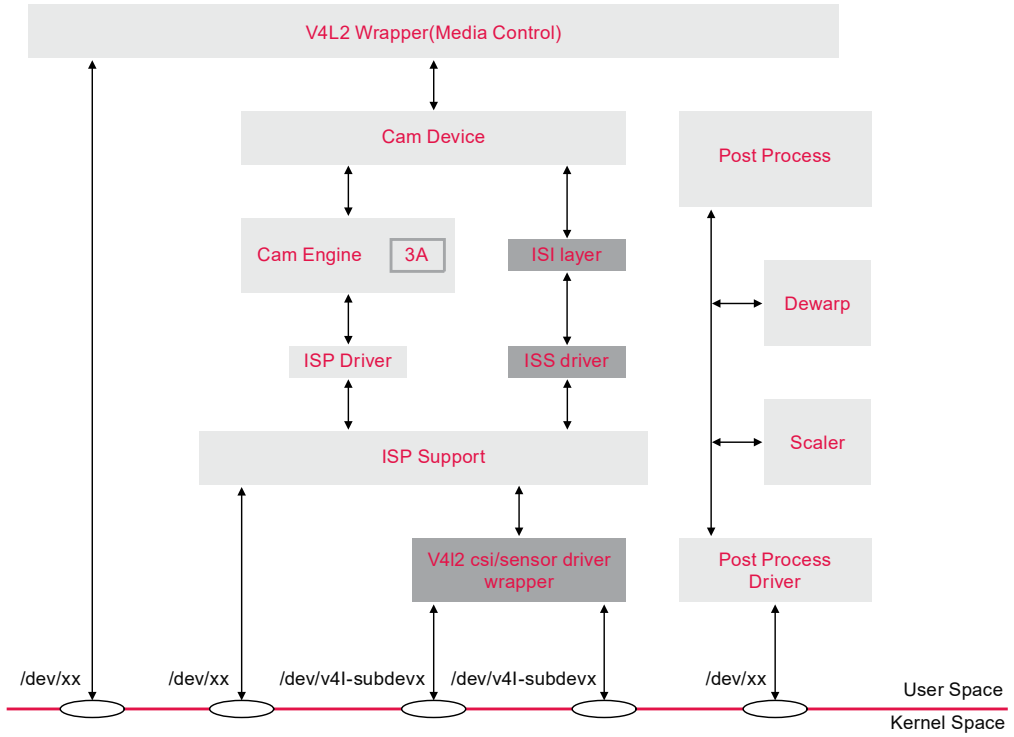


Figure 16. ISP proprietary software stack

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.

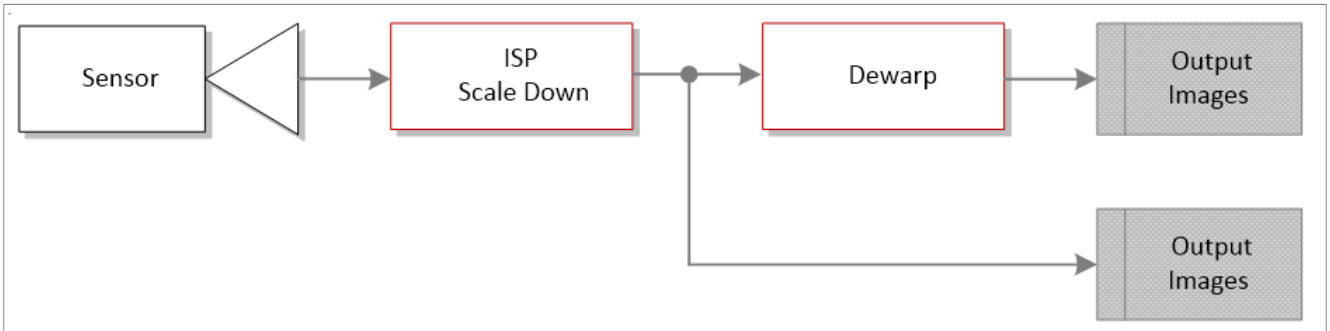


Figure 17. ISP Image Output Flow

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.

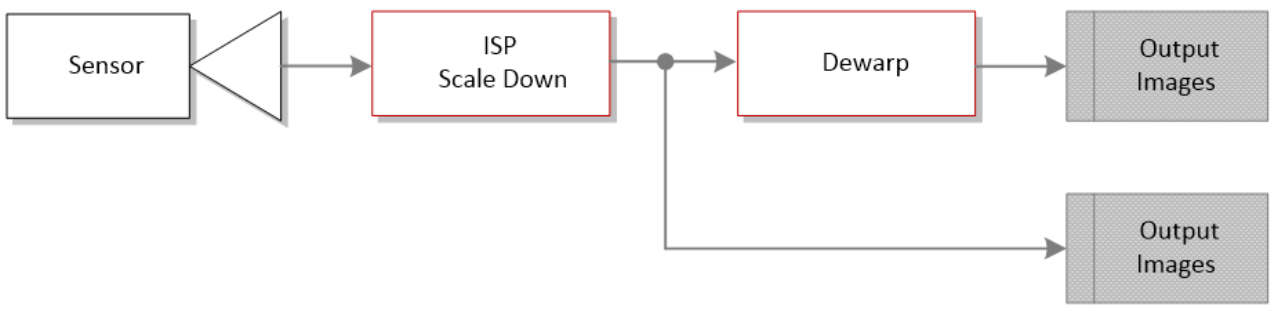


Figure 18. ISP Dewarp Calibration Flow

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,
        "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"   : [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 119. Sensor mode and resolution

Sensor Mode	Resolution
0	3840x2160 Linear
1	3840x2160 HDR
2	1920x1080 Linear

Table 119. Sensor mode and resolution...*continued*

Sensor Mode	Resolution
3	1920x1080 HDR

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 ISP Unit Test vvext User Manual

5.1 Overview

5.1.1 Introduction

`vvext` is a binary executable provided in the release package that provides functions to control features in the Vivante Image Signal Processor (ISP) IP and can be executed from a Command Prompt window.

This section provides a guide to use `vvext` and its corresponding submodules.

A given version of the Vivante ISP IP may not support the features discussed herein. References to unsupported features may be ignored in this section.

5.1.2 Hardware and software requirements

Hardware requirement:

- 8MPLUSLPD4-EVK
- OV2775 sensor, BASLER sensor, or OS08A20 sensor

Software requirement:

- ISP software release 4.2.2_p21 and later

5.1.3 Test environment

- ISP Driver Version: 4.2.2
- Release Date: August 30, 2023
- Release Format: Source code

5.2 vvext usage

There are two ways to use the `vvext` binary executable:

- Direct Command
- Interactive Mode

5.2.1 Direct command

To use the Direct Command, open a new Command Prompt window.

In the Command Prompt window, type:

```
export LD_LIBRARY_PATH=$pwd:$LD_LIBRARY_PATH
```

To execute a `vvext` function, use the following format:

```
./vvext <video_id> "window name" "param string"
```

where `<video_id>` specifies the video device number in the `/dev` folder (e.g., video1, video2, …, video 9).

For example, to turn AWB off for the video device called video2, type:

```
./vvext 2 "AWB On/Off" "0"
```

5.2.2 Interactive mode

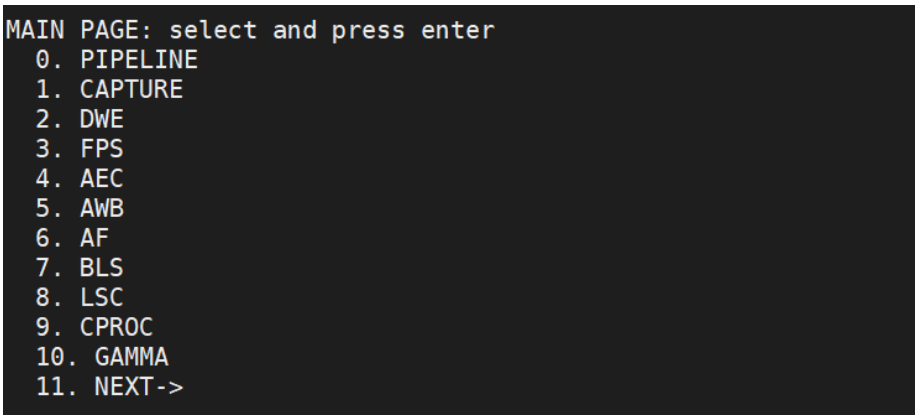
To use Interactive Mode, open a new Command Prompt window.

In the Command Prompt window, type:

```
export LD_LIBRARY_PATH=$pwd:$LD_LIBRARY_PATH
./vvext <video_id>
```

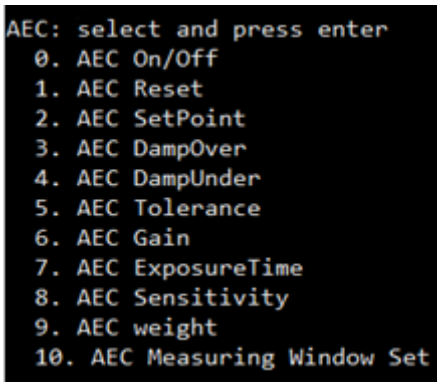
where `<video_id>` specifies the video device number in the `/dev` folder (e.g., video1, video2, …, video 9).

The following menu should appear in the Command Prompt window.



To select a function, type the number of the function to execute.

For example, to display the AEC page, type **4** and press Enter. The AEC page with its corresponding features will be displayed as in the following figure.



Select the desired feature and press Enter.

To return to the previous page, press Enter without a value.

5.3 vvext feature list

Table 120. vvext feature list

Feature	Function	Window Name (used by Direct Command)	Parameter
PIPELINE	Enable/disable sensor stream	PIPELINE	0: Disable; 1: Enable
CAPTURE	Save YUV file from video stream	CAPTURE	0
DWE	Enable/disable bypass	DEWARP BYPASS ON/OFF	1 integer value
	Switch dewarp mode	DEWARP_MODEL_LENS_DISTORTION_CORRECTION	0
	Switch dewarp mode	DEWARP_MODEL_FISHEYE_EXPAND	0
	Switch dewarp mode	DEWARP_MODEL_FISHEYE_DEWARP	0

Table 120. vvext feature list*...continued*

Feature	Function	Window Name (used by Direct Command)	Parameter
	Enable/disable hflip	DEWARP HFLIP ON/OFF	1 integer value
	Enable/disable vflip	DEWARP VFLIP ON/OFF	1 integer value
	Adjust dewarp matrix	DEWARP MATRIX	17 float values: 9 camera matrix values + 8 coefficient values
FPS	Set frame rate	FPS	1 integer value, sensor-specific
AEC	Enable/disable AEC	AEC On/Off	0: Disable; 1: Enable
	Reset to the default configuration	AEC Reset	1 integer value
	Adjust AEC setpoint	AEC SetPoint	1 float value, range: [0.0 … 255.0]
	Adjust AEC dampover	AEC DampOver	1 float value, [0.0 … 1.0]
	Adjust AEC dampunder	AEC DampUnder	1 float value, [0.0 … 1.0]
	Adjust AEC tolerance	AEC Tolerance	1 float value, [0.0 … 100.0]
	Set gain	AEC Gain	1 float value
	Set integration time	AEC Exposure Time	1 float value
	Set sensitivity (ISO)	AEC Sensitivity	1 integer value, typically from 100 to 1600
	Set weight	AEC weight	25 integer values, [0 … 16]
	Set AEC Measuring Window	AEC Measuring Window	[left, top, width, height]
AWB	Enable/disable AWB	AWB On/Off	0: Disable; 1: Enable
	Reset to default configuration	AWB Reset	1 integer value
	Enable/disable AWB Auto Control features	AWB Auto Ctrl item	3 integer values: AWB mode [1 .. 2: auto] Index [0..4] damping data [0 .. 1]
	Sample to gain red channel	GAIN RED to 3	1 integer value
	Sample to gain blue channel	GAIN BLUE to 3	1 integer value
	Adjust AWB gain	GAIN INPUT	4 float values: [1.0 - 3.999]
	Adjust matrix	CCM INPUT	9 float values: [-8 - 7.992]
	Adjust offset	Offset INPUT	3 values (R, G, B): [-2048 – 2047]
	Adjust AWB measuring window	Measuring Window Set	4 values: (left, top, width, height)
AF	Enable/disable AF	AF On/Off	1 integer value
	Switch manual/auto mode	AF WorkingMode	1: Manual; 2: Auto
	Enable/disable one shot and measure window	AF OneShot	5 values (oneshot enable, startX, startY, width, height)

Table 120. vvext feature list*...continued*

Feature	Function	Window Name (used by Direct Command)	Parameter
	Switch search algorithm	AF SearchAlgorithm	1 or 2 or 3
	Adjust AF length	AF Length	Range: 0 to 100
BLS	Adjust BLS params	BLS	4 values (R, Gr, Gb, B)
LSC	Enable/disable LSC	LSC ON/OFF	0: Disable; 1: Enable
CPROC	Enable/disable CPROC	CPROC ON/OFF	0: Disable; 1: Enable
	Adjust brightness	Adjust brightness	Range: -127 to 127, default 0
	Adjust contrast	Adjust contrast	Range: 0 to 1.99, default 1
	Adjust saturation	Adjust saturation	Range: 0 to 1.99, default 1
	Adjust hue	Adjust HUE	Range: -90 to 89, default 0
	CC coefficient	CC coefficient	Index 1 or 2
GAMMA	Enable/disable gamma	GAMMA ON/OFF	0: Disable; 1: Enable
	Switch gamma mode logarithmic/equidistant	SWITCH GAMMA SCALING MODE	0: EQU; 1: LOG
	Generate new gamma curve	GAMMA CURVE	Range: [0.1 to 4.0]
DEMOSAIC	Switch demosaic mode: normal/bypass	DEMOSAIC SWITCH MODE	0: Disable; 1: Enable
	Adjust demosaic threshold	DEMOSAIC THRESHOLD	Range: [0 to 255]
FILTER	Enable/disable filter	FILTER ON/OFF	0: Disable; 2: Enable
	Switch filter auto/manual mode	FILTER Mode	1: Manual; 2: Auto
	Input new denoise and sharpen level	Adjust denoise/sharpen level	2 values: [1 .. 10]
	Input new chroma horizontal mode	Adjust chroma horizontal mode	Range: [0 to 3]
	Input new chroma vertical mode	Adjust chroma vertical mode	Range: [0 to 3]
CAC	Enable/disable CA	CAC ON/OFF	0: Disable; 1: Enable
CNR	Enable/disable CNR	CNR ON/OFF	0: Disable; 1: Enable
	Adjust CNR threshold	Adjust CNR threshold	2 values: (CNR thresh1, CNR thresh2); Range: [0 … 32767]
DPCC	Enable/disable DPCC	DPCC ON/OFF	0: Disable; 1: Enable
DPF	Enable/disable DPF	DPF ON/OFF	0: Disable; 1: Enable
WDR3	Enable/disable WDR3	WDR3 ON/OFF	0: Disable; 1: Enable
	Switch WDR3 mode auto/ manual	WDR3 Auto/Manual	1: Manual; 2: Auto
	Adjust WDR3 strength, max Gain, globalStrength	WDR3 STRENGTH INPUT	3 values: [0 .. 128]

Table 120. vvext feature list...*continued*

Feature	Function	Window Name (used by Direct Command)	Parameter
HDR	Set extension bit and HDR ratio	HDR	1 integer value, 1 float value
REGGET	Read sensor register value	REGGET	1 hexadecimal value
REGSET	Write sensor register value	REGSET	2 hexadecimal values
PRELOAD	Load the sensor calibration file	PRELOAD	1 integer value

5.4 vvext submodule description

5.4.1 PIPELINE

- PIPELINE: Switches sensor stream enable

```
./vvext 2 "PIPELINE" "1"
```

- PIPELINE: Switches sensor stream disable

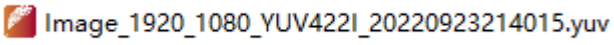
```
./vvext 2 "PIPELINE" "0"
```

5.4.2 CAPTURE

- CAPTURE: Takes a snapshot from the specified stream. The YUV file is saved with a filename formatted as follows: `Image_<width>_<height>_<YUV format>_<date><imageid>.yuv`

```
./vvext 2 "CAPTURE" 0
```

Example:



5.4.3 DWE

- DEWARP BYPASS ON: enable the dewarp bypass

```
./vvext 2 "DEWARP BYPASS ON/OFF" 1
```

- DEWARP BYPASS OFF: disable the dewarp bypass

```
./vvext 2 "DEWARP BYPASS ON/OFF" 1
```

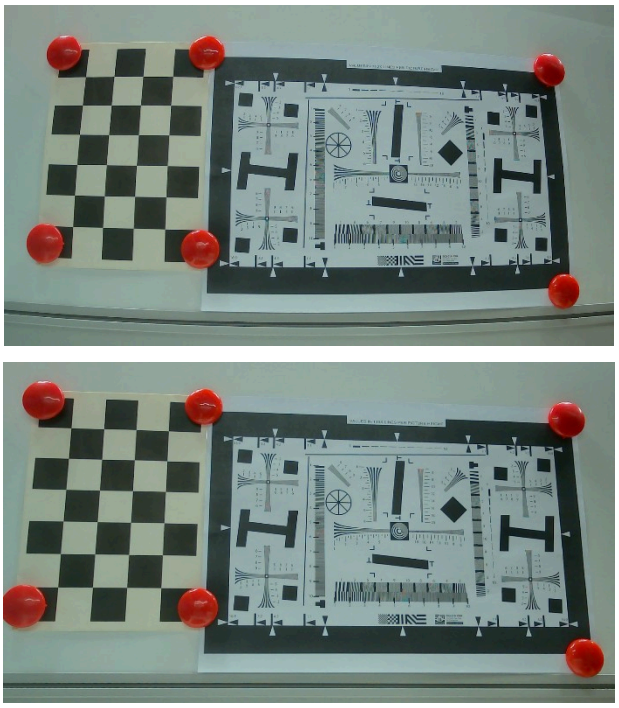


Figure 21. DWE Bypass On vs. Bypass Off

Note: *When the "DEWARP BYPASS ON/OFF" feature is used, the bypass variable value is set to the opposite value of the previous value, and it is independent of the value of the "params string."*

Three dewarp modes are supported:

– LENS_DISTORTION: Dewarp mode0

– FISHEYE_EXPAND: Dewarp mode1

– FISHEYE_DEWARP: Dewarp mode2

These modes can be changed on-the-fly, so dewarp must first be enabled (BYPASS OFF).

```
./vvext 2 "DEWARP BYPASS ON/OFF" 1
```

- LENS DISTORTION CORRECTION: Dewarp mode0

```
./vvext 2 "DEWARP_MODEL_LENS_DISTORTION_CORRECTION" 0
```

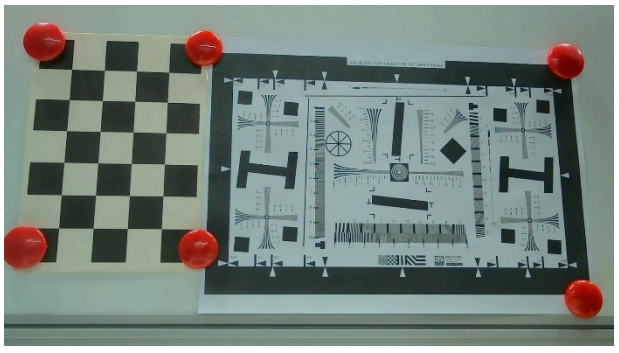


Figure 22. Dewarp Mode: LENS DISTORTION CORRECTION

- FISHEYE EXPAND: dewarp mode1

```
./vvext 2 "DEWARP_MODEL_FISHEYE_EXPAND" 0
```

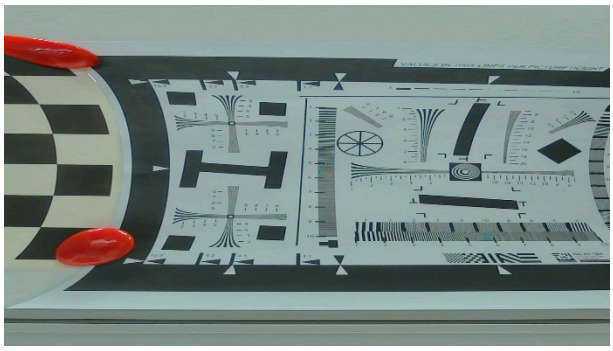


Figure 23. Dewarp Mode: FISHEYE EXPAN

- FISHEYE DEWARP: dewarp mode2

```
./vvext 2 "DEWARP_MODEL_FISHEYE_DEWARP" 0
```

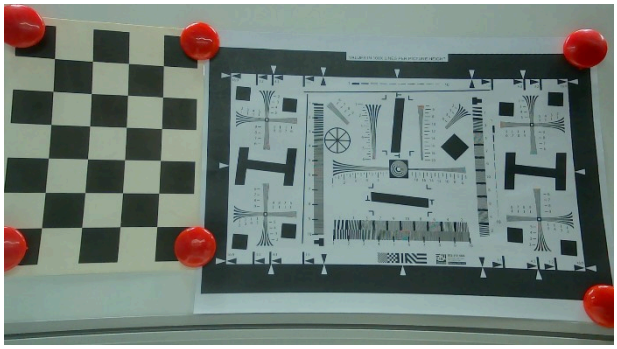
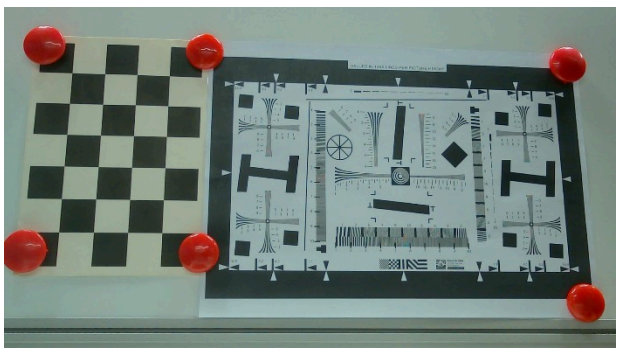


Figure 24. Dewarp Mode: FISHEYE DEWARP

- DEWARP HFLIP: horizontal flip

```
./vvext 2 "DEWARP HFLIP ON/OFF" 1
```

Note: *Horizontal flip is disabled (OFF) by default. When the "DEWARP HFLIP ON/OFF" feature is used, the horizontal flip variable value is set to the opposite value of the previous value, and it's independent of the value of the "params string."*



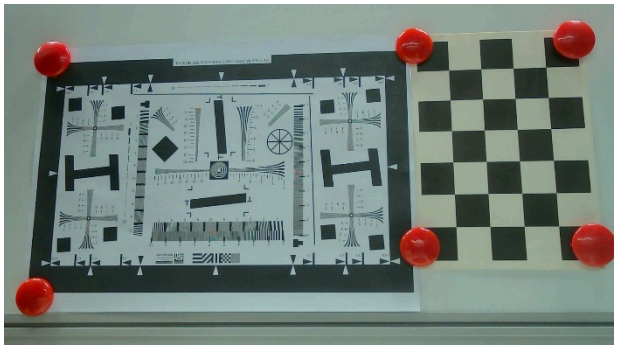


Figure 25. Horizontal Flip Off vs. Horizontal Flip On

- DEWARP VFLIP: vertical flip

```
./vvext 2 "DEWARP VFLIP ON/OFF" 1
```

Note: *Vertical flip is disabled (OFF) by default. When the "DEWARP VFLIP ON/OFF" feature is used, the vertical flip variable value is set to the opposite value of the previous value, and it is independent of the value of the "params string."*

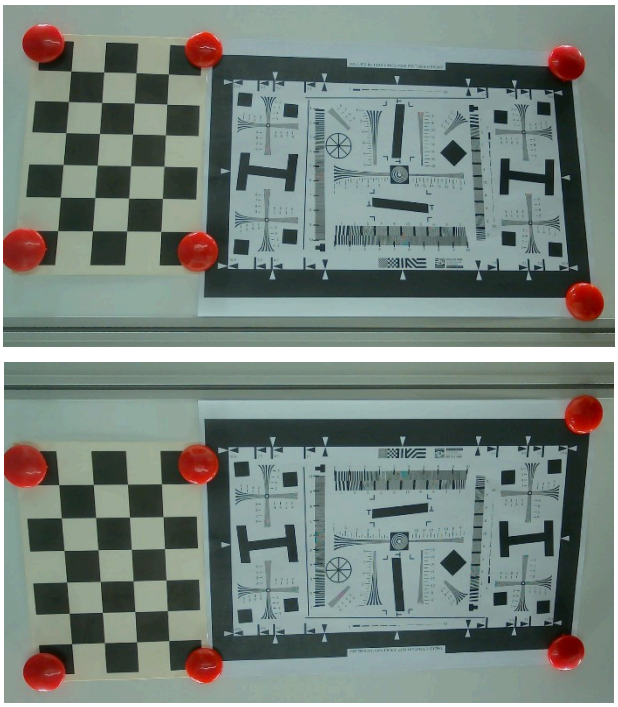


Figure 26. Vertical Flip Off vs. Vertical Flip On

- DEWARP MATRIX: dewarp matrix combines the camera matrix with distortion parameters; camera matrix [0~8], Distortion coefficient [9~16].
  Note: *DEWARP MATRIX takes effect only when the dewarp bypass is disabled.*
  If the default matrix is:
  "1894.261155, 0.000000, 963.747423, 0.000000, 1894.261155, 545.511265, 0.000000, 0.000000, 1.000000, -0.341290, 0.216663, -0.000722, 0.000407, -0.127021, 0.000000, 0.000000, 0.000000"
  To input a new matrix:

```
./vvext 2 "DEWARP MATRIX" "1000, 0.000000, 963.747423, 0.000000, 1890,
 532, 0.000000, 0.000000, 1.000000, -0.341290, 0.216663, -0.000722,
 0.000407,-0.127021, 0.000000, 0.000000, 0.000000"
```

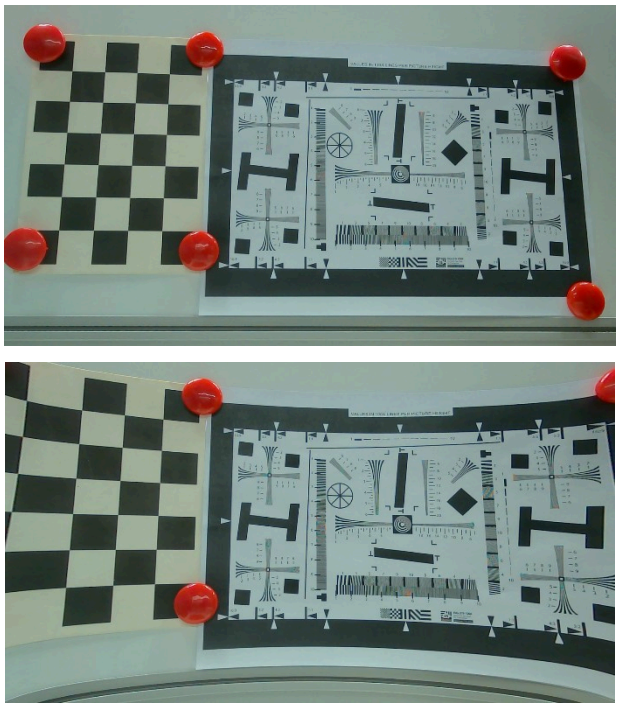


Figure 27. Dewarp Matrix

5.4.4 FPS

- FPS: controls the video display frame rate within a sensor-specific range.

```
./vvext 2 "FPS" 20
./vvext 2 "FPS" 10
```

5.4.5 AEC

- AEC ON: enables auto-exposure control.

```
./vvext 2 "AEC On/Off" 1
```

- AEC OFF: disables auto-exposure control.

```
./vvext 2 "AEC On/Off" 0
```

- AEC Reset: resets auto-exposure control configuration to the default value which depends on calibration xml. Configuration includes damp.over, damp.under, set point, and tolerance.

```
./vvext 2 "AEC Reset" 1
```

Note: *AEC Reset takes effect only when AEC is enabled.*

- AEC SetPoint: sets auto-exposure control target luminance value; range: [0.0 .. 255.0].

```
./vvext 2 "AEC SetPoint" 70
./vvext 2 "AEC SetPoint" 150
```

Note: *AEC SetPoint takes effect only when AEC is enabled.*

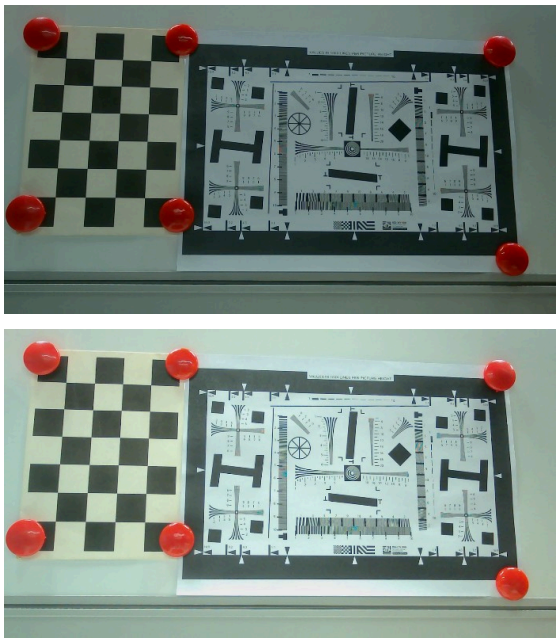


Figure 28. Setpoint 70 vs. Setpoint 150

- $Exposure_{(next\ frame)} = \left(1 - damp\right) * Exposure_{(target)} + damp * Exposure_{(last\ frame)}$
- AEC DampOver: sets auto-exposure control damping upper limit; range: [0.0 .. 1.0]. When the actual scene luminance value exceeds the SetPoint, DampOver affects the ratio calculation of the last frame exposure to the next frame exposure, i.e., it affects the reduction adjustment speed.

```
./vvext 2 "AEC DampOver" 0.5
```

Note: *AEC DampOver takes effect only when AEC is enabled.*

- AEC DampUnder: sets auto-exposure control damping lower limit; range: [0.0 .. 1.0]. When actual scene luminance value is under the SetPoint, DampOver affects the ratio calculation of last frame exposure to the next frame exposure, i.e., it affects the increment adjustment speed.

```
./vvext 2 "AEC DampUnder" 0.5
```

Note: *AEC DampUnder takes effect only when AEC is enabled.*

- $\frac{\left|\text{Luminance}_{(\text{actual})} - \text{SetPoint}\right|}{\text{SetPoint}} \leq \frac{\text{Tolerance}}{100}$
- AEC Tolerance: sets auto-exposure control calculation accuracy; range: [0.0 .. 100.0].

```
./vvext 2 "AEC Tolerance" 20
```

Note: *AEC Tolerance takes effect only when AEC is enabled.*

- AEC Gain: sets exposure gain within a sensor-specific range.

```
./vvext 2 "AEC Gain" 1.0
./vvext 2 "AEC Gain" 4.0
```

Note: *AEC Gain can be called and takes effect only when AEC is disabled.*

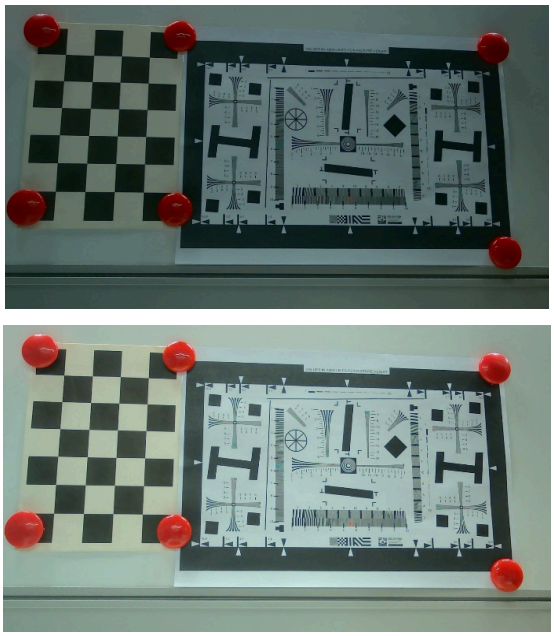


Figure 29. AEC Gain 1.0 vs AEC Gain 4.0

- AEC ExposureTime: sets exposure time within a sensor-specific range.

```
./vvext 2 "AEC ExposureTime" 0.005
./vvext 2 "AEC ExposureTime" 0.01
```

Note: *AEC ExposureTime can be called and takes effect only when AEC is disabled.*

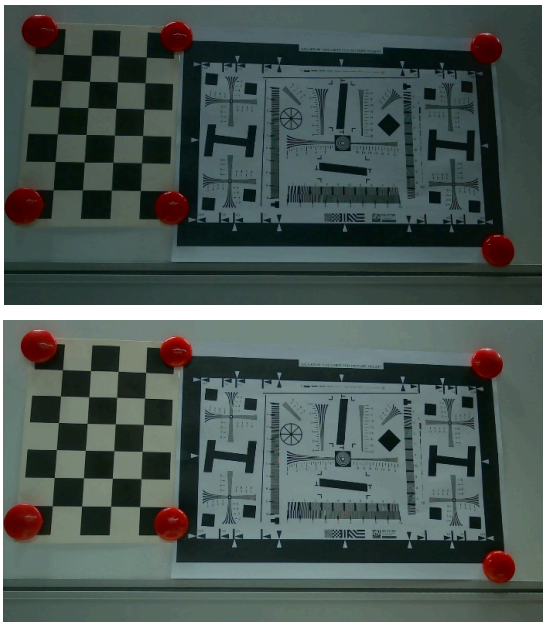


Figure 30. AEC ExposureTime Examples

- AEC Sensitivity: sets exposure sensitivity (ISO) within a sensor-specific range. Typical range: [100 .. 1600].

```
./vvext 2 "AEC Sensitivity" 100
./vvext 2 "AEC Sensitivity" 400
```

Note: *AEC Sensitivity can be called and takes effect only when AEC is disabled.*

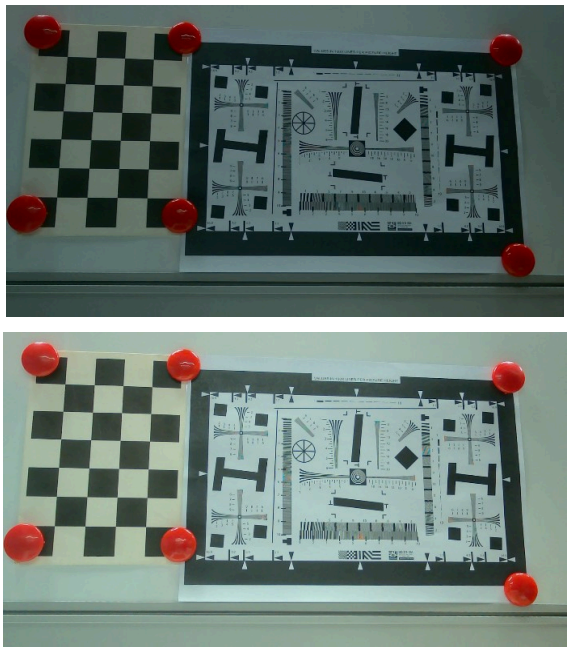


Figure 31. AEC Sensitivity (ISO) 100 vs 400

- AEC weight: sets auto exposure weights of a 5x5 block, 25 integer numbers; range: [0 .. 16]. Increase the weight value of a block to select the ROI (region of interest) and set AEC based on the ROI statistics to calculate appropriate exposure.

```
./vvext 2 "AEC weight" "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
./vvext 2 "AEC weight" "16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1"
./vvext 2 "AEC weight" "1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1"
```

Note: *AEC Weight takes effect only when AEC is enabled.*

- ROI: total image. All default values of the 5x5 block are 1, which indicates that the ROI is the total image.

```
./vvext 2 "AEC weight" "1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1"
```

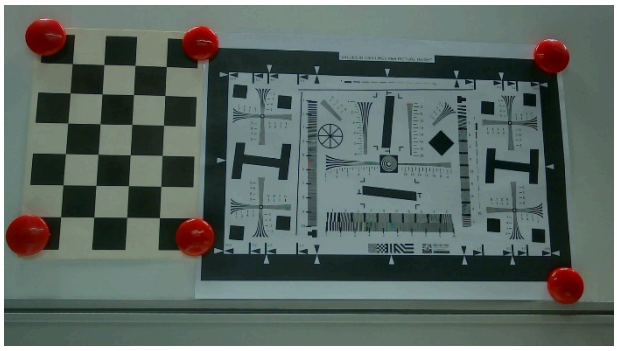


Figure 32. ROI: Total Image

A 5x5 block statistic is shown in the following figure.

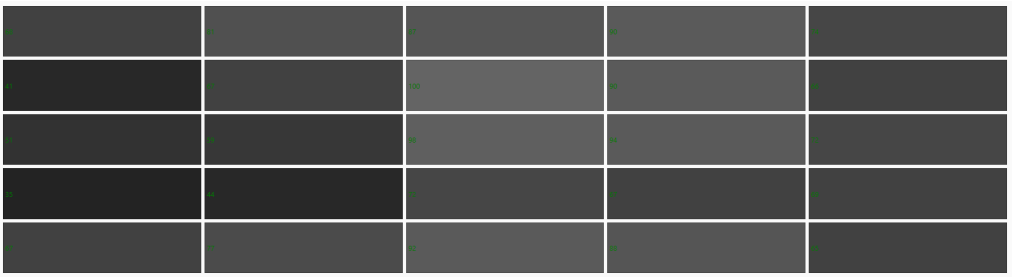


Figure 33. 5x5 Block Statistic

- ROI: left region. From the figure above, we can see that the left region of the image is dark. We select the left 5 blocks as the ROI and set the corresponding weight to the maximum value 16.

```
./vvext 2 "AEC weight" "16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1"
```

- ROI: middle region. From the figure above, we can see that the middle region of the image is bright. We select the middle 5 blocks as the ROI and set the corresponding weight to the maximum value 16.

```
./vvext 2 "AEC weight" "1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1,1,1,16,1,1"
```

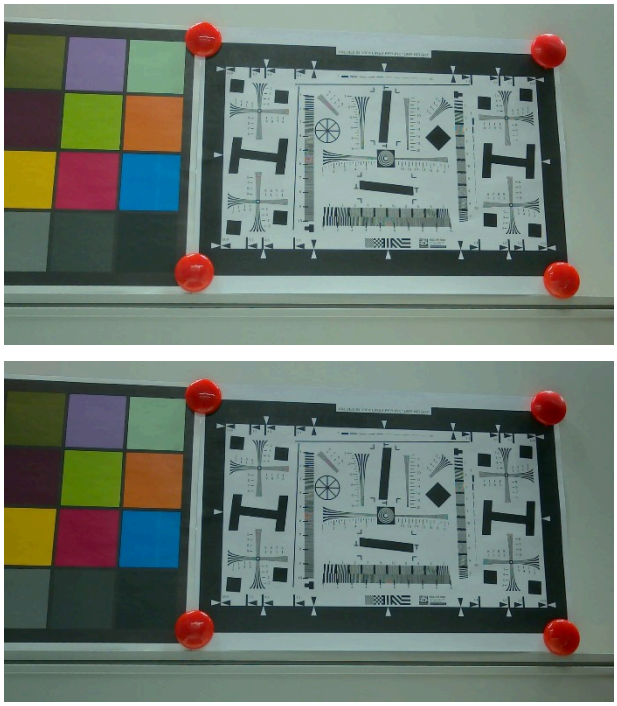


Figure 34. ROI Left Region and Middle Region

The ROI Left Region and Middle Region examples can be seen from the comparison of the figure above:

- ROI: Left Region is dark - after setting the AEC weight, the AEC increases the exposure and the image became brighter.
- ROI: Middle Region is bright - after setting the AEC weight, the AEC reduces the exposure and the image became darker.

5.4.6 AEC Measuring Window Set

This function sets the position and size of the measuring window in an image for AE adjustment. The parameters are in the order of [left, top, width, height]:

- `left`: The x coordinate of the top left point of the measuring window. The value range is [0, 65535].
- `top`: The y coordinate of the top left point of the measuring window. The value range is [0, 65535].
- `width`: The width of the measuring window. The value range is [0, 65535].
- `height`: The height of the measuring window. The value range is [0, 65535].

Where:

- left + width <= sensor width
- top + height <= sensor height

Note: *The AEC Measuring Window Set function can be called only when AE is disabled and the function takes effect when AE is enabled again.*

Example:

```
./vvext 2 "AEC Measuring Window Set" "0, 0, 1920, 1080"
```

5.4.7 AWB

- AWB ON: enables auto-white balance.

```
./vvext 2 "AWB On/Off" 1
```

- AWB OFF: disables auto-white balance.

```
./vvext 2 "AWB On/Off" 0
```

- AWB Reset: resets auto-white balance.

```
./vvext 2 "AWB Reset" 1
```

Note: *AWB Reset takes effect only when AWB is enabled.*

- AWB Auto Ctrl item: sets the following items:
  - AWB mode (1: manual; 2: auto)
  - The index of illumination profile of calibration data (0: A; 1: D50; 2: D65; 3: F2 (CWF); 4: F11 (TL84))
  - Damping data, changes white balance smoothly through temporal damping (0: False; 1: True)

```
./vvext 2 "AWB Auto Ctrl item" "1,0,1"
./vvext 2 "AWB Auto Ctrl item" "1,2,1"
```

Note:

- *AWB Auto Ctrl items can be set when AWB is disabled and takes effect only when AWB is enabled.*
- *If AWB mode is set to auto, it will automatically calculate the appropriate illumination profile index independent of the illumination index that is set.*

- Set AWB Illumination Profile A:

```
./vvext 2 "AWB On/Off" 0
./vvext 2 "AWB Auto Ctrl item" "1,0,1"
./vvext 2 "AWB On/Off" 1
```

- Set AWB Illumination Profile D65:

```
./vvext 2 "AWB On/Off" 0
./vvext 2 "AWB Auto Ctrl item" "1,2,1"
./vvext 2 "AWB On/Off" 1
```

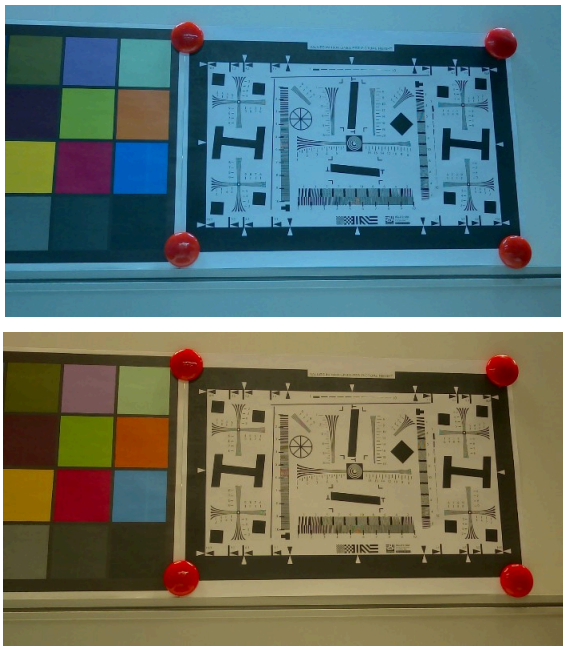


Figure 35. AWB Illumination Profile A and D65

- GAIN RED to 3: sample to gain red channel, sets the AWB red channel gain to 3.

```
./vvext 2 "GAIN RED to 3" 1
```

Note: *GAIN RED to 3 can be called and takes effect only when AWB is disabled.*

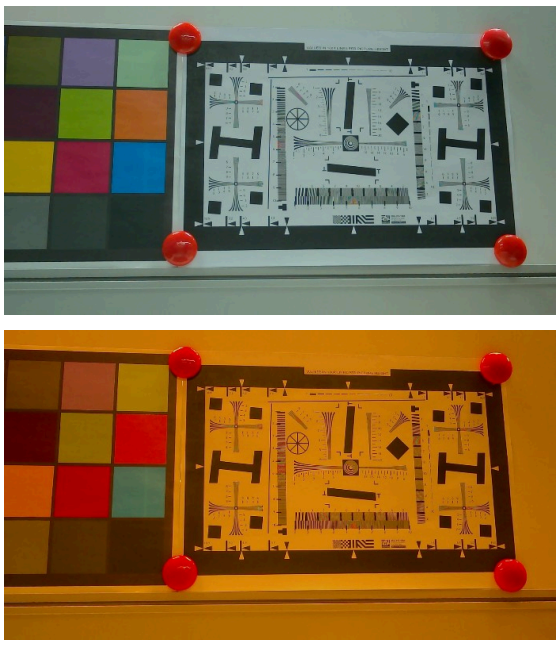


Figure 36. AWB Red Channel Gain 3

- GAIN BLUE to 3: sample to gain blue channel, sets the AWB blue channel gain to 3.

```
./vvext 2 "GAIN BLUE to 3" 1
```

Note: *GAIN BLUE to 3 can be called and takes effect only when AWB is disabled.*

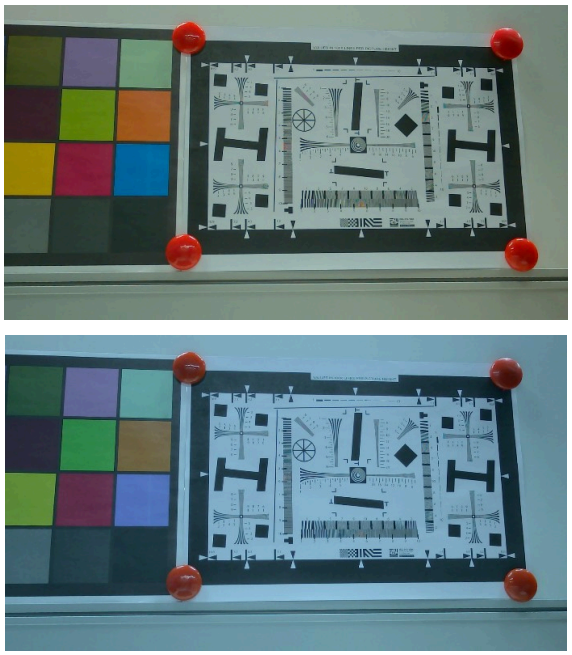


Figure 37. AWB Blue Channel Gain 3

- GAIN INPUT: adjust AWB gain [r, gr, gb, b]; range: [1.0 .. 3.999].

```
./vvext 2 "GAIN INPUT" "1.4, 1, 1, 2.3"
./vvext 2 "GAIN INPUT" "1, 1, 1, 1"
```

Note: *GAIN INPUT can be called and takes effect only when AWB is disabled.*

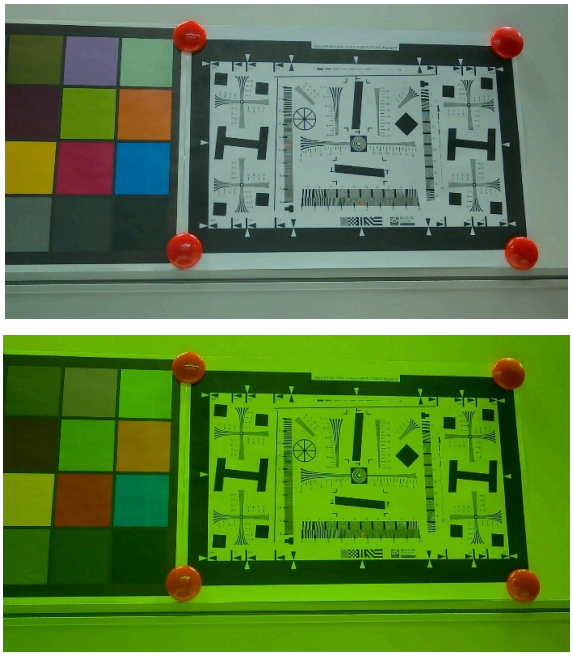


Figure 38. AWB Gain Input [1.4, 1, 1, 2.3] and [1, 1, 1, 1]

- CCM INPUT: auto-white balance Color Correction Matrix; 9 float values; range: [-8 .. 7.992].

```
./vvext 2 "CCM INPUT" "1, 0, 0, 0, 1, 0, 0, 0, 1"
./vvext 2 "CCM INPUT" "1.875000, -0.640625, -0.179688, -0.328125, 1.562500,
```

```
-0.148438,0.054688, -0.851562,1.906250"
```

Note: *CCM INPUT can be called and takes effect only when AWB is disabled.*

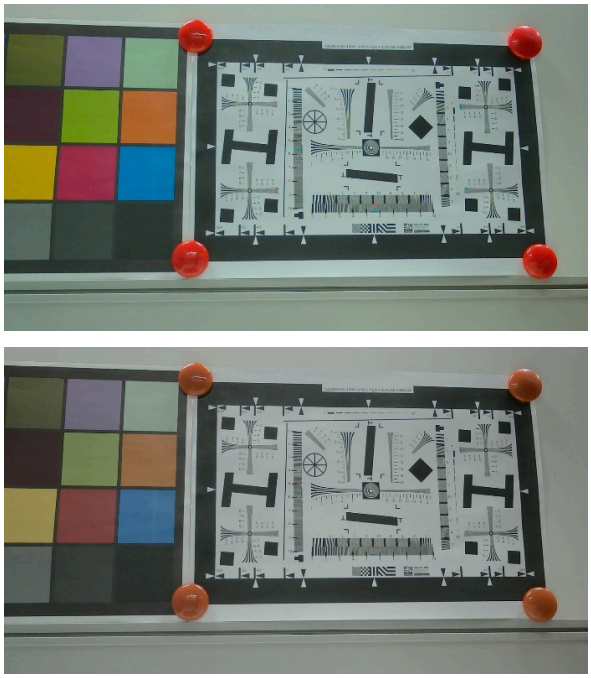
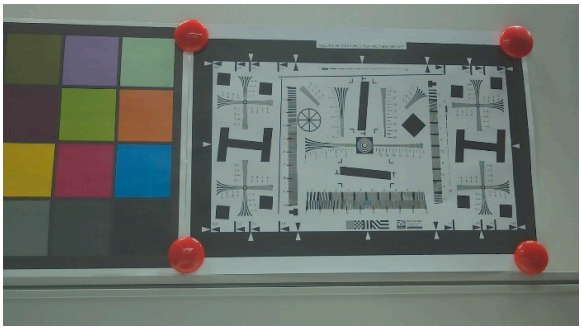


Figure 39. CCM Input Examples

- Offset INPUT: auto-white balance color [R, G, B] offset; 3 integer values; range: [-2048 .. 2047].

```
./vvext 2 "Offset INPUT" "0, 0, 0"
./vvext 2 "Offset INPUT" "1000, 0, 0"
```

Note: *Offset INPUT can be called and takes effect only when AWB is disabled.*



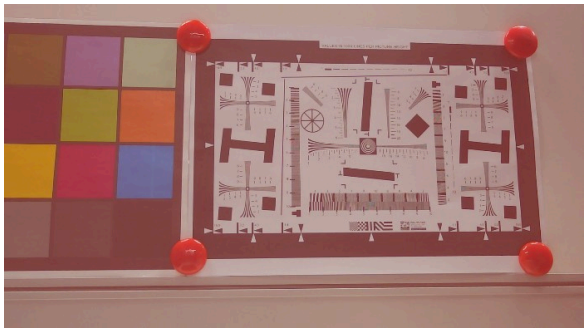


Figure 40. Offset INPUT Examples

- Measuring Window Set: sets the auto-white balance measuring window [left, top, width, height].

```
./vvext 2 "Measuring Window Set" "0, 0, 1920, 1080"
```

Note: *Measuring Window Set can be called only when AWB is disabled and takes effect when AWB is enabled again.*

– left + width <= sensor width

– top + height <= sensor height

5.4.8 AF

- AF OFF: disables auto-focus.

```
./vvext 2 "AF On/Off" 1
```

Note: *When the "AF On/Off" feature is used, the value is set to the opposite value of the previous value, and it is independent of the value of the "params string."*

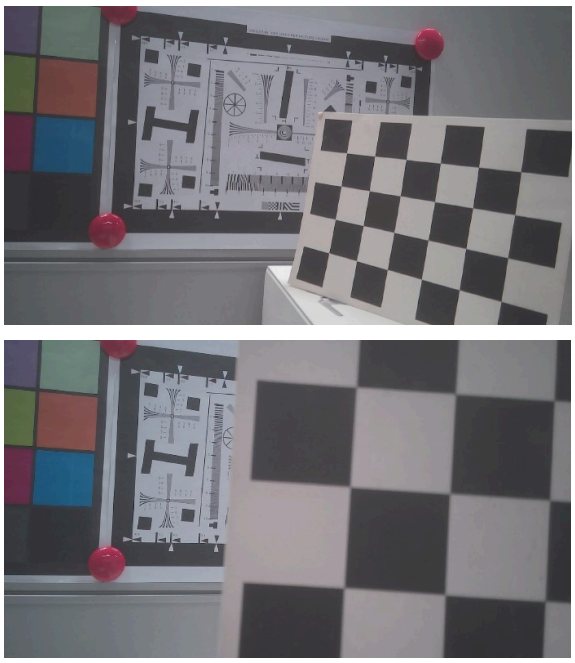


Figure 41. Auto Focus OFF

- AF ON: enables auto-focus.

```
./vvext 2 "AF On/Off" 1
```

Note: *When the "AF On/Off" feature is used, the value is set to the opposite value of the previous value, and it is independent of the value of the "params string."*

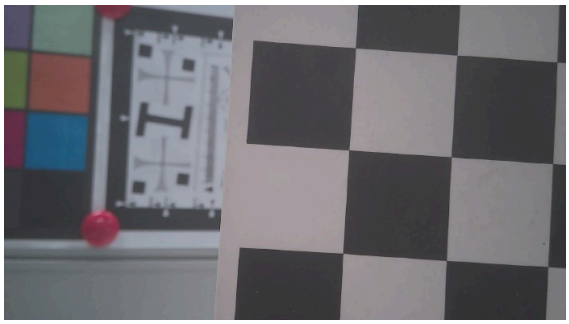


Figure 42. Auto Focus ON

- AF WorkingMode: selects auto-focus working mode; 1: Manual, 2: Auto. It will automatically calculate the appropriate length when in auto mode.

```
./vvext 2 "AF WorkingMode" 1
./vvext 2 "AF WorkingMode" 2
```

Note: *AF WorkingMode takes effect only when AF is enabled.*

- AF OneShot: oneshot means only auto focus once and keep the configuration. Parameters: set the one shot enable variable (0: Disable, 1: Enable) and measure window [startX, startY, width, height], where startX and width range are 5 to (sensor width – 5), startY and height range: [2 .. (sensor height – 4)].

```
./vvext 2 "AF OneShot" "1, 860, 440, 200, 200"
./vvext 2 "AF OneShot" "0, 860, 440, 200, 200"
```

Note: *AF OneShot takes effect only when AF is enabled.*

- AF SearchAlgorithm: selects the AF algorithm; 1: Full search, 2: Adaptive range search, 3: Hill climbing.

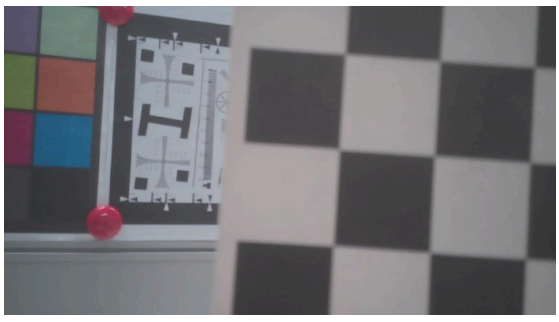
```
./vvext 2 "AF SearchAlgorithm" "2"
```

Note: *AF SearchAlgorithm takes effect only when AF is enabled.*

- AF Length: sets the focus position; range: [0 .. 100].

```
./vvext 2 "AF Length" "10"
./vvext 2 "AF Length" "28"
./vvext 2 "AF Length" "55"
./vvext 2 "AF Length" "85"
```

Note: *AF Length takes effect only when AF is enabled and working mode is manual.*



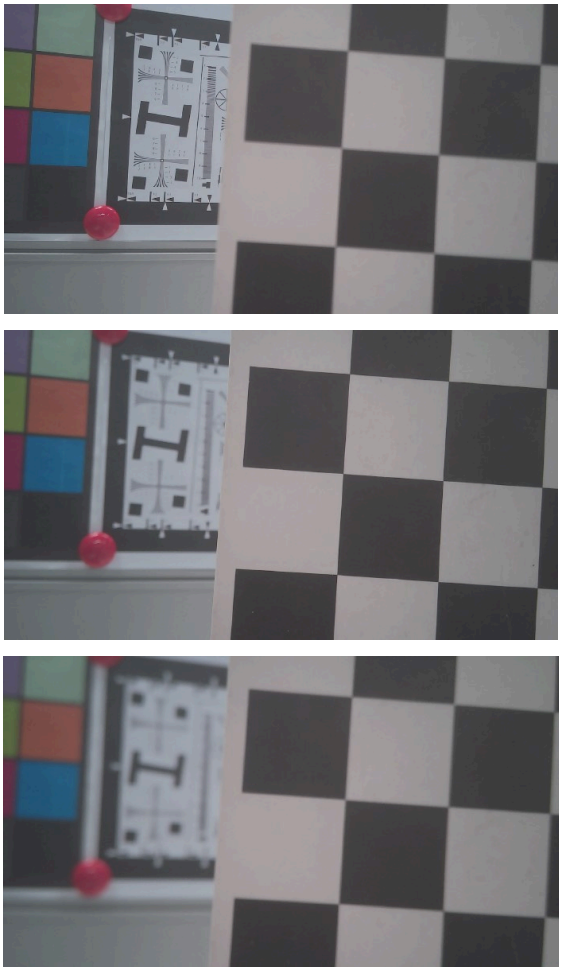
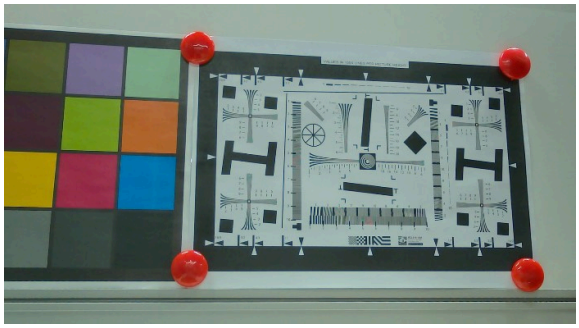


Figure 43. AF Lengths 10, 28, 55 and 85

5.4.9 BLS

- BLS: adjusts black level subtraction parameters [R, Gr, Gr, B]; range: [0 .. 4095].

```
./vvext 2 "BLS" "168, 168, 168, 168"
./vvext 2 "BLS" "0, 168, 168, 168"
```



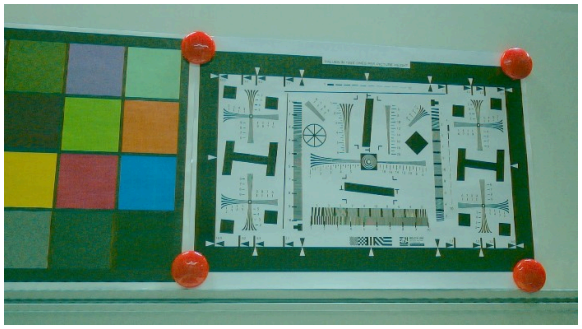


Figure 44. BLS Examples

5.4.10 LSC

- LSC ON: enables Lens Shade Correction.

```
./vvext 2 "LSC ON/OFF" 1
```

- LSC OFF: disables Lens Shade Correction.

```
./vvext 2 "LSC ON/OFF" 0
```

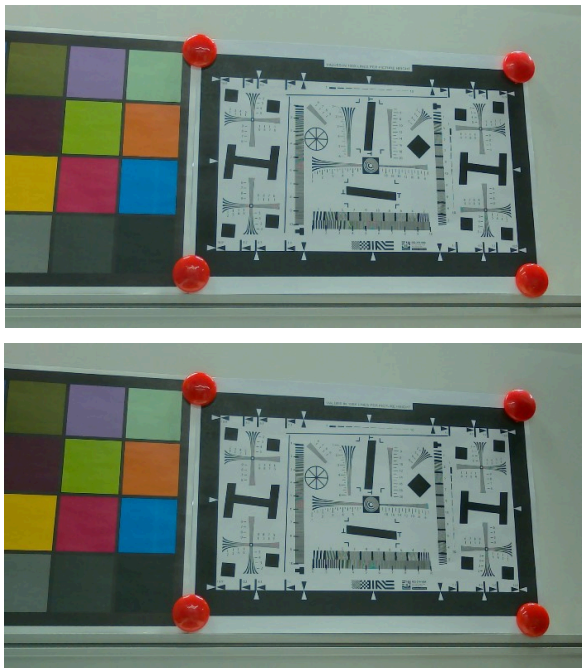


Figure 45. LSC ON vs. OFF

5.4.11 CPROC

- CPROC ON: enables color processing.

```
./vvext 2 "CPROC ON/OFF" 1
```

- CPROC OFF: disables color processing.

```
./vvext 2 "CPROC ON/OFF" 0
```

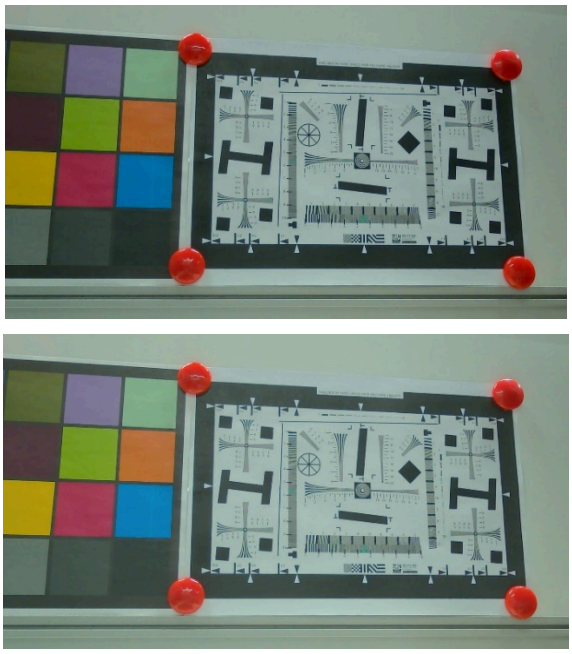


Figure 46. CPROC ON vs. OFF

- Adjust brightness: range: [-127 .. 127]

```
./vvext 2 "Adjust brightness" -127
./vvext 2 "Adjust brightness" 127
```

Note: *Adjust brightness takes effect only when CPROC is enabled.*

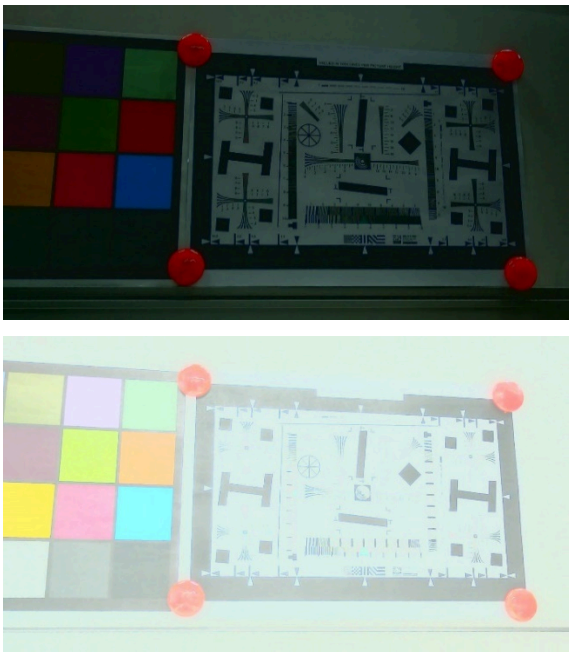


Figure 47. Adjust Brightness Examples

- Adjust contrast: range: [0 .. 1.99]

```
./vvext 2 "Adjust contrast" 0.5
./vvext 2 "Adjust contrast" 1.5
```

Note: *Adjust contrast takes effect only when CPROC is enabled.*

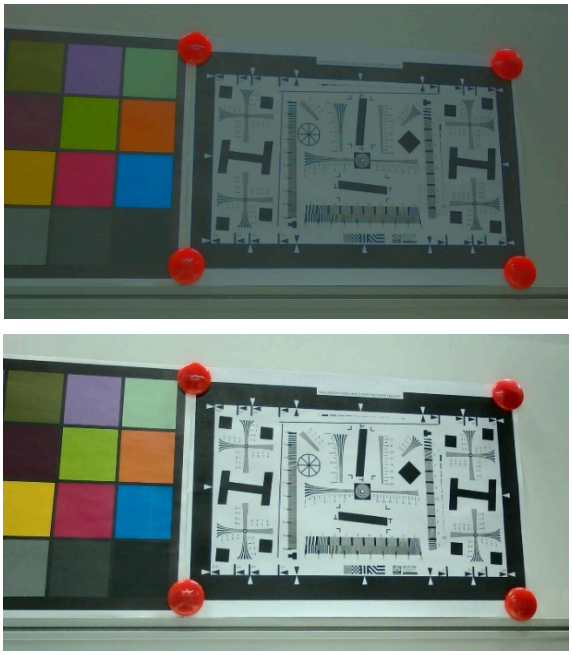


Figure 48. Adjust Contrast Examples

- Adjust saturation: range: [0 .. 1.99]

```
./vvext 2 "Adjust saturation" 0.5
./vvext 2 "Adjust saturation" 1.5
```

Note: *Adjust saturation takes effect only when CPROC is enabled.*

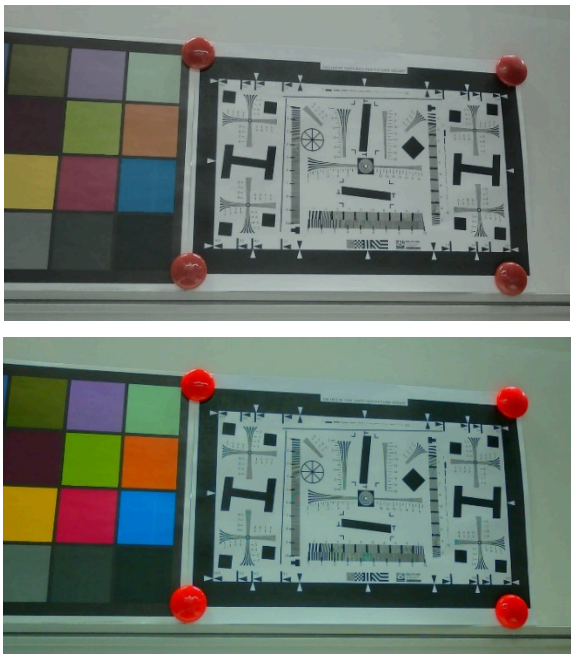


Figure 49. Adjust Saturation Examples

- Adjust HUE: range: [-90 .. 89]

```
./vvext 2 "Adjust HUE" -90
```

```
./vvext 2 "Adjust HUE" 89
```

Note: *Adjust HUE takes effect only when CPROC is enabled.*

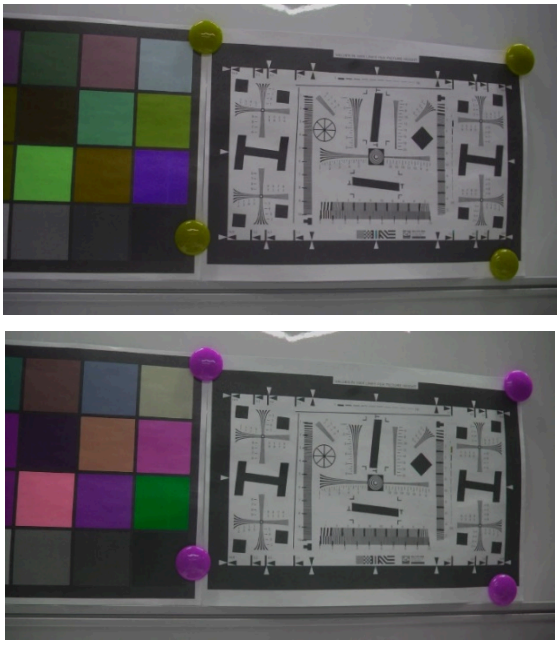


Figure 50. Adjust Hue Examples

– CC coefficient: provides 2 sets of parameters based on the specified index.
– Index 1: 0.257812, 0.5, 0.101562, -0.148438, -0.289062, 0.4375, 0.4375, -0.367188, -0.070312.
– Index 2: 0.296875, 0.585938, 0.117188, -0.171875, -0.328125, 0.5, 0.5, -0.421875, -0.078125

```
./vvext 2 "CC coefficient" 1
./vvext 2 "CC coefficient" 2
```

Note: *CC coefficient takes effect only when CPROC is enabled.*

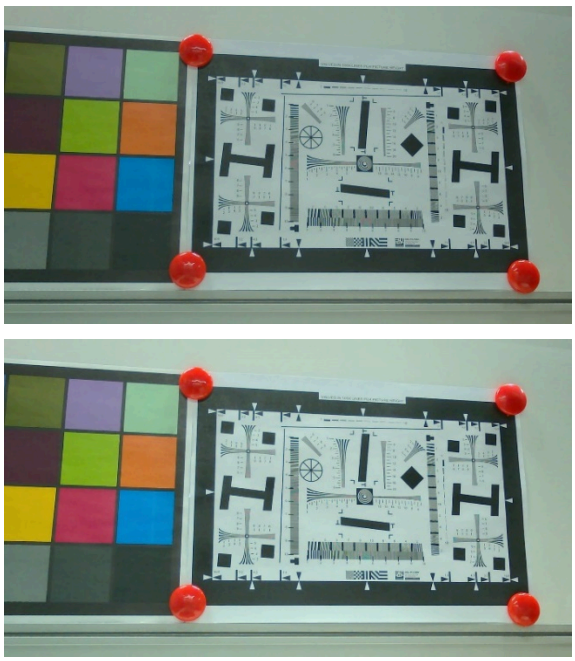


Figure 51. CC Coefficient: Index 1 and Index 2

5.4.12 GAMMA

- GAMMA ON: enables gamma.

```
./vvext 2 "GAMMA ON/OFF" 1
```

- GAMMA OFF: disables gamma.

```
./vvext 2 "GAMMA ON/OFF" 0
```

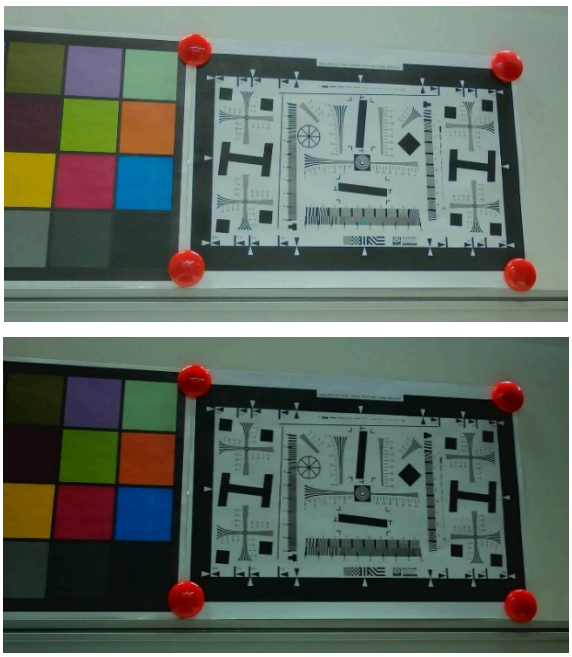
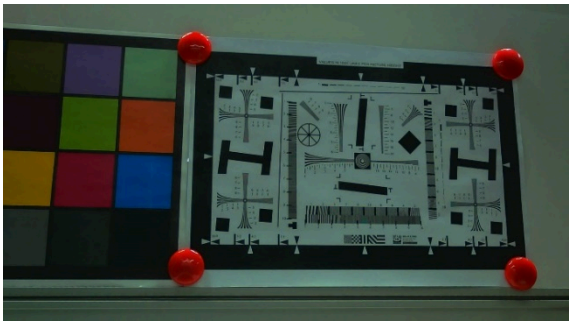


Figure 52. Gamma ON vs. Gamma OFF

- SWITCH GAMMA SCALING MODE: supports 0: EQU mode and 1: LOG mode.
  EQU mode: equidistant segmentation from 0 to 4095, (256, 256, ...), all 16 segments are 256.
  LOG mode: logarithmic segmentation from 0 to 4095,
  (64,64,64,64,128,128,128,128,256,256,256,512,512,512,512,512).

```
./vvext 2 "SWITCH GAMMA SCALING MODE" 1
./vvext 2 "SWITCH GAMMA SCALING MODE" 0
```

Note: *SWITCH GAMMA SCALING MODE takes effect only when gamma is enabled.*



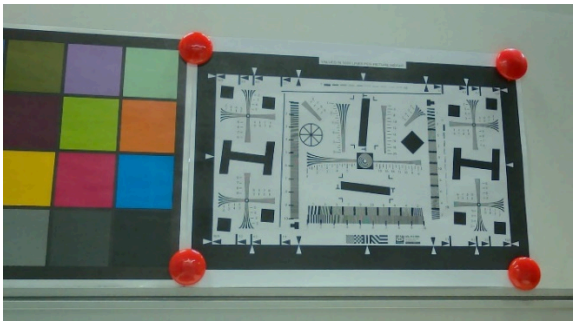


Figure 53. SWITCH GAMMA SCALING MODE 1 (LOG) and 0 (EQU)

- GAMMA CURVE: adjusts gamma factor; range: [0.1 .. 4.0].

```
./vvext 2 "GAMMA CURVE" 0.5
./vvext 2 "GAMMA CURVE" 4.0
```

Note: *GAMMA CURVE takes effect only when gamma is enabled.*

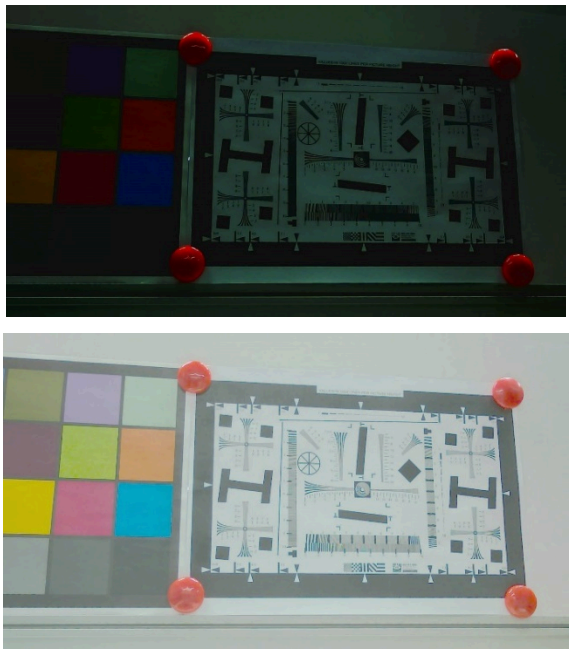


Figure 54. GAMMA CURVE 0.5 and 4.0

5.4.13 DEMOSAIC

- DEMOSAIC ENABLE:

```
./vvext 2 "DEMOSAIC SWITCH MODE" 1
```

- DEMOSAIC BYPASS:

```
./vvext 2 "DEMOSAIC SWITCH MODE" 0
```

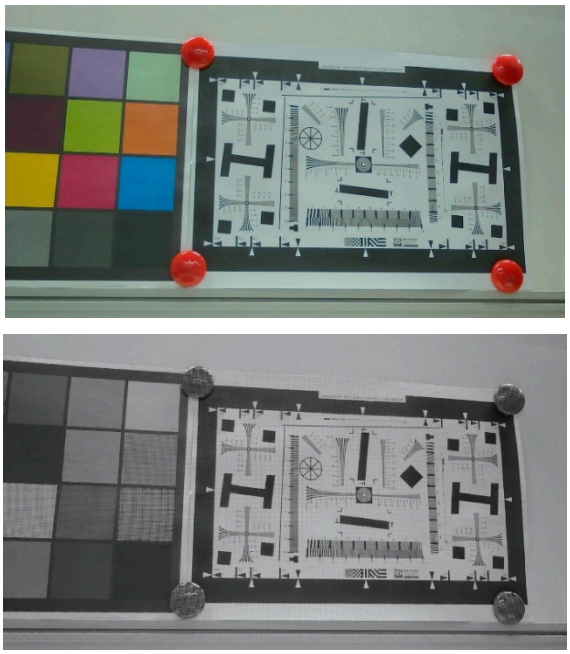
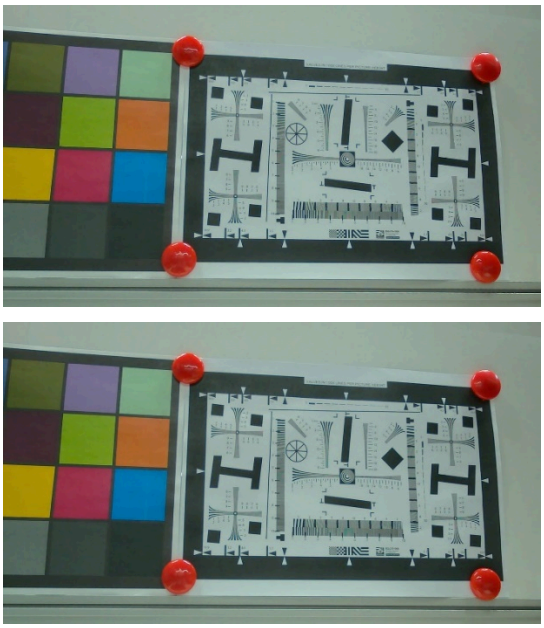


Figure 55. DEMOSAIC ENABLE and BYPASS

- DEMOSAIC THRESHOLD: adjusts demosaic texture detection threshold; range: [0 .. 255]. 0: maximum edge sensitivity. 255: no texture detection.

```
./vvext 2 "DEMOSAIC THRESHOLD" 0
./vvext 2 "DEMOSAIC THRESHOLD" 255
```

Note: *DEMOSAIC THRESHOLD takes effect only when demosaic is enabled.*



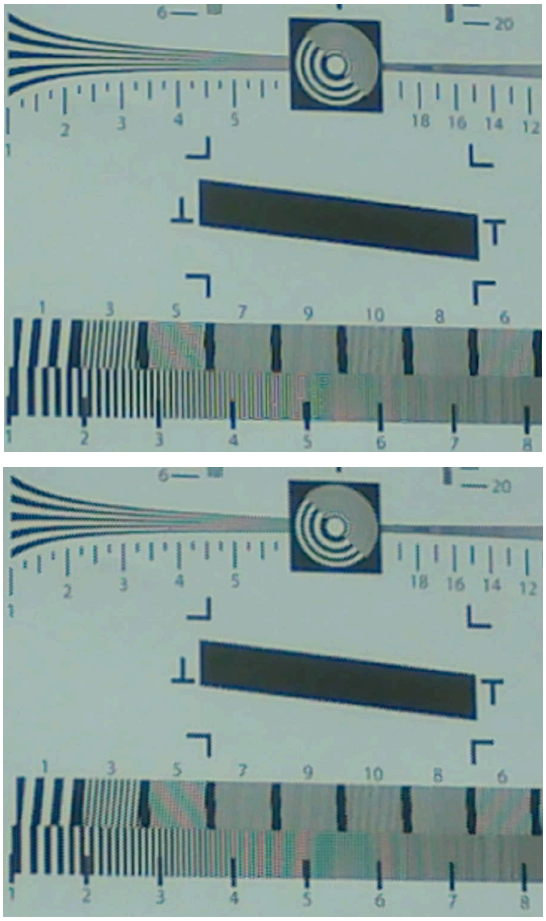


Figure 56. DEMOSAIC THRESHOLD Examples

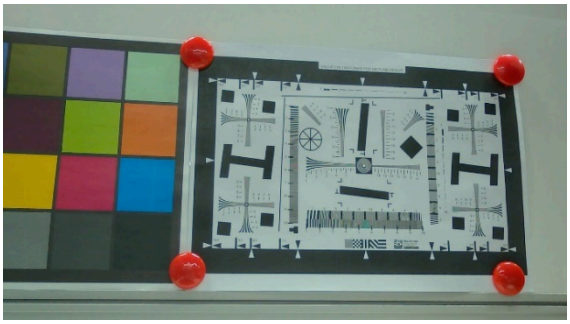
5.4.14 FILTER

- FILTER ON:

```
./vvext 2 "FILTER ON/OFF" 1
```

- FILTER OFF:

```
./vvext 2 "FILTER ON/OFF" 0
```



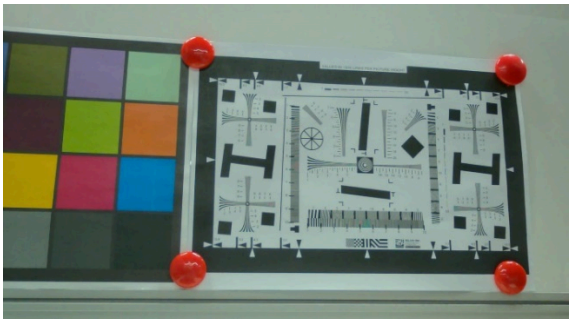


Figure 57. FILTER ON vs. OFF

- FILTER Mode: switches filter manual/auto mode; 0: Manual, 1: Auto; it will automatically calculate the appropriate value of denoise and sharpen.

```
./vvext 2 "FILTER Mode" 0
./vvext 2 "FILTER Mode" 1
```

Note: *FILTER MODE can be set when filter is OFF and takes effect only when filter is ON.*

- Adjust denoise/sharpen level: inputs new denoise/sharpen levels; 2 values, range: [1 .. 10].

```
./vvext 2 "Adjust denoise/sharpen level" "1, 1"
./vvext 2 "Adjust denoise/sharpen level" "10, 10"
```

Note: *Adjust denoise/sharpen level can be set when filter is OFF and takes effect only when filter is ON, and mode is set to Manual.*

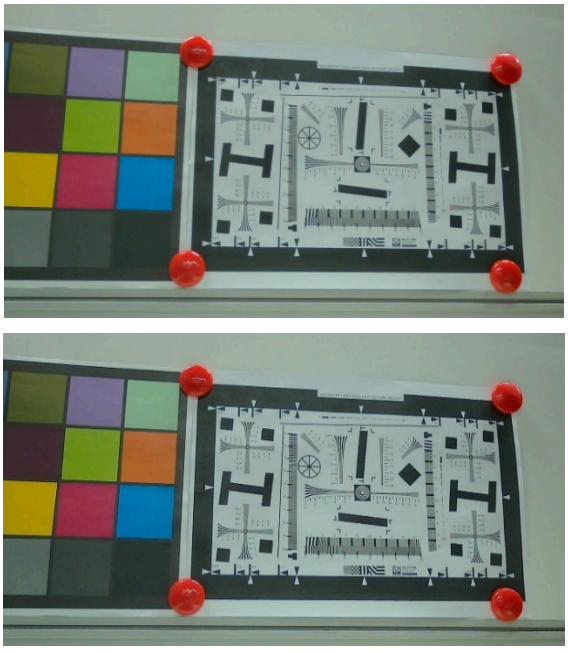


Figure 58. Adjust Denoise/Sharpen Level Examples

- Adjust chroma horizontal mode: inputs new chroma horizontal mode; range: [0 .. 3].

```
./vvext 2 "Adjust chroma horizontal mode" 0
./vvext 2 "Adjust chroma horizontal mode" 3
```

Note: *Adjust chroma horizontal mode can be set when filter is disabled and takes effect only when filter is ON.*

- Adjust chroma vertical mode: inputs new chroma vertical mode; range: [0 .. 3].

```
./vvext 2 "Adjust chroma vertical mode" 0
./vvext 2 "Adjust chroma vertical mode" 3
```

Note: *Adjust chroma vertical mode can be set when filter is disabled and takes effect only when filter is ON.*

Figure 59. Adjust Chroma Horizontal and Vertical Mode Examples

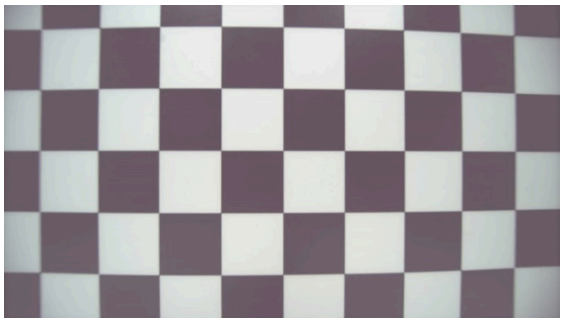
5.4.15 CAC

- CAC ON: enables chromatic aberration correction

```
./vvext 2 "CAC ON/OFF" 1
```

- CAC OFF: disables chromatic aberration correction

```
./vvext 2 "CAC ON/OFF" 0
```



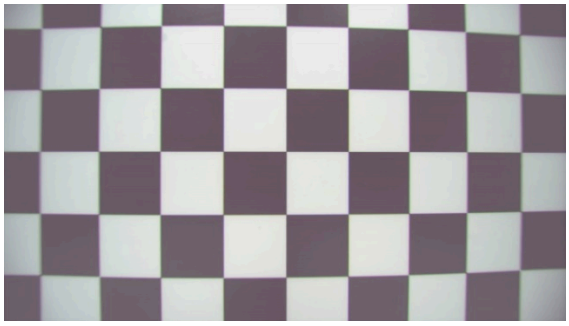


Figure 60. CAC ON vs. OFF

5.4.16 CNR

- CNR ON: enables color noise reduction

```
./vvext 2 "CNR ON/OFF" 1
```

- CNR OFF: disables color noise reduction

```
./vvext 2 "CNR ON/OFF" 0
```

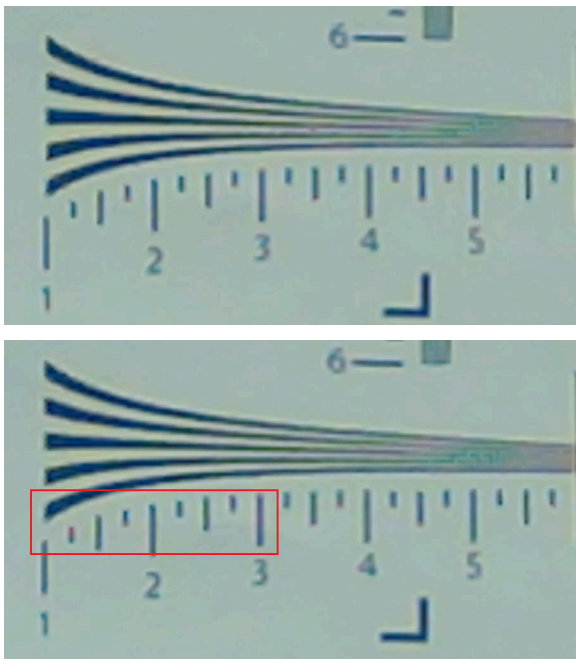


Figure 61. CNR ON vs. OFF

- Adjust CNR threshold: adjusts the color noise reduction thresholds of channel 1 and 2; range: [0 … 32767]. The larger the value, the stronger the noise reduction

```
./vvext 2 "Adjust CNR threshold" "0, 0"
./vvext 2 "Adjust CNR threshold" "100, 100"
```

Note: *Adjust CNR threshold takes effect only when CNR is enabled.*

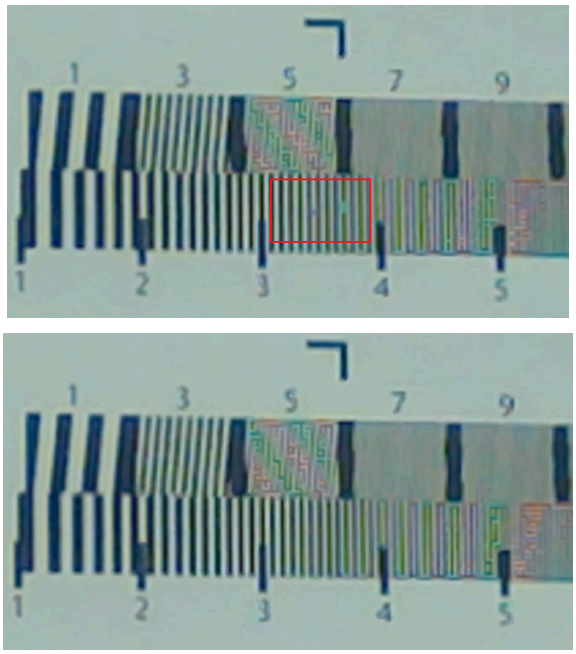


Figure 62. Adjust CNR Threshold Examples

5.4.17 DPCC

- DPCC ON: enables defect pixel cluster correction

```
./vvext 2 "DPCC ON/OFF" 1
```

- DPCC OFF: disables defect pixel cluster correction

```
./vvext 2 "DPCC ON/OFF" 0
```

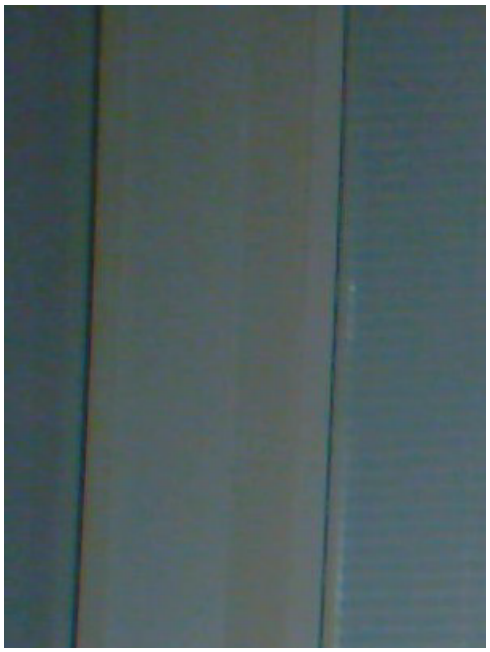


Figure 63. DPCC ON vs. DPCC OFF

5.4.18 DPF

- DPF ON: enables denoising prefilter

```
./vvext 2 "DPF ON/OFF" 1
```

- DPF OFF: disables denoising prefilter

```
./vvext 2 "DPF ON/OFF" 0
```

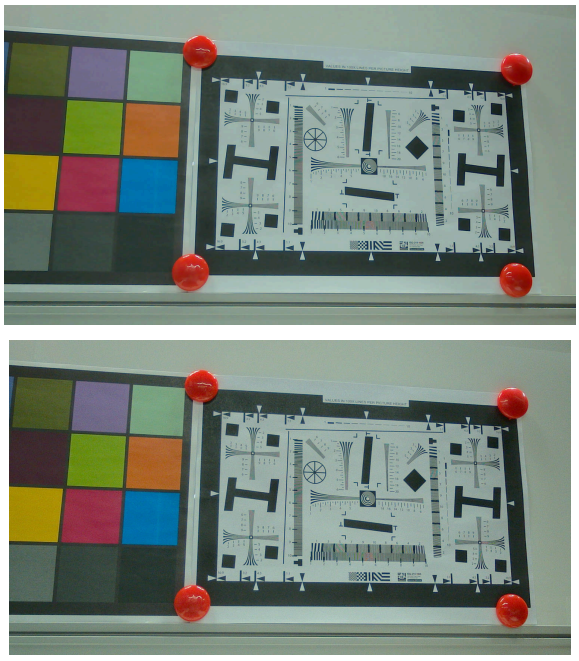


Figure 64. DPF ON vs.DPF OFF

5.4.19 WDR3

- WDR3 ON: enables Wide Dynamic Range 3

```
./vvext 2 "WDR3 ON/OFF" 1
```

- WDR3 OFF: disables Wide Dynamic Range 3

```
./vvext 2 "WDR3 ON/OFF" 0
```

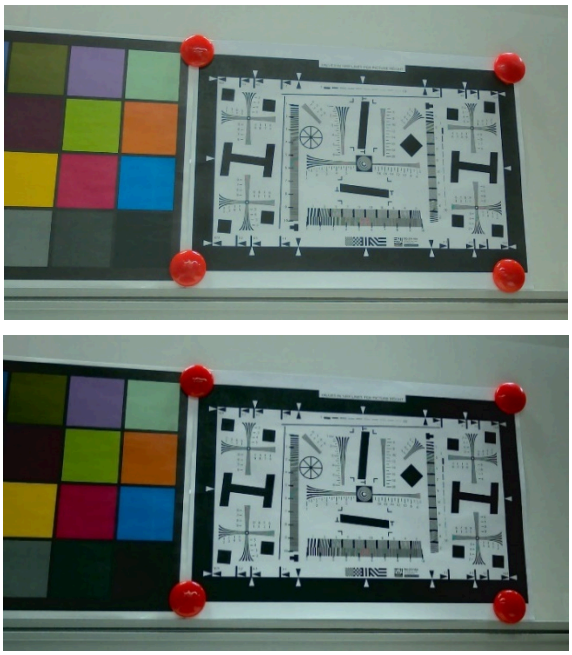


Figure 65. WDR3 ON vs. WDR3 OFF

- WDR3 Auto/Manual: switches WDR3 manual/auto mode; 0: Manual, 1: Auto. It will automatically calculate the appropriate strength, maxGain, and globalStrength parameters based on the auto table.

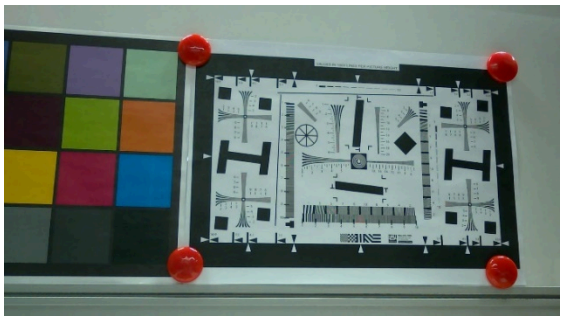
```
./vvext 2 "WDR3 Auto/Manual" 0
./vvext 2 "WDR3 Auto/Manual" 1
```

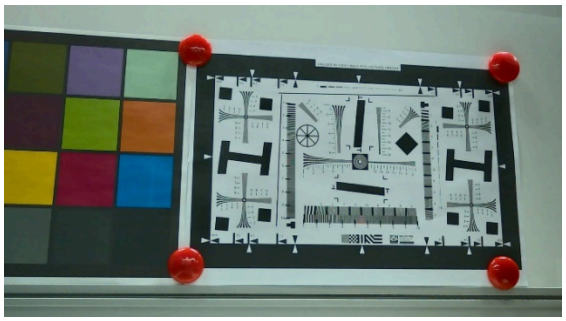
Note: *WDR3 Auto/Manual can be set when WDR3 is disabled and takes effect only when WDR3 is enabled.*

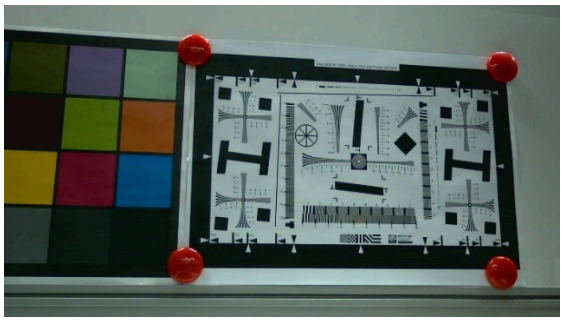
- WDR3 STRENGTH INPUT: adjusts the following 3 WDR3 parameters; 3 integer values; range: [0 .. 128]:
  - strength input
  - maxGain
  - globalStrength

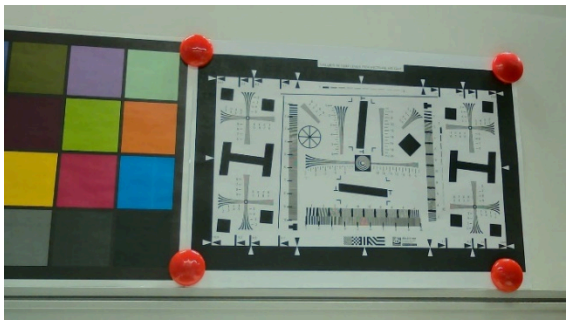
```
./vvext 2 "WDR3 STRENGTH INPUT" "0, 64, 128"
./vvext 2 "WDR3 STRENGTH INPUT" "128, 64, 128"
./vvext 2 "WDR3 STRENGTH INPUT" "64, 0, 128"
./vvext 2 "WDR3 STRENGTH INPUT" "64, 128, 128"
./vvext 2 "WDR3 STRENGTH INPUT" "64, 64, 0"
./vvext 2 "WDR3 STRENGTH INPUT" "64, 64, 128"
```

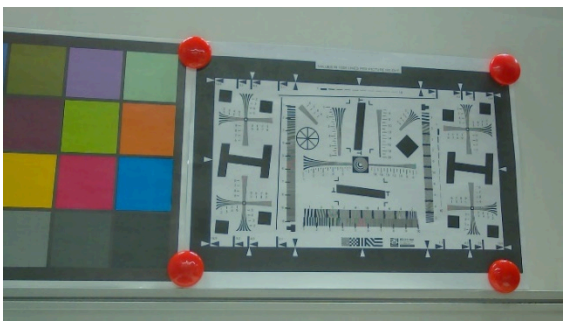
Note: *WDR3 STRENGTH INPUT can be set when WDR3 is disabled and takes effect only when WDR3 is enabled, and WDR3 mode is manual.*











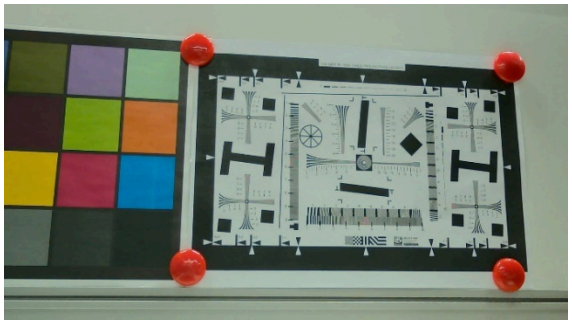


Figure 66. WDR3 STRENGTH INPUT Examples

5.4.20 HDR

- HDR RATIO INPUT: sets extension bit* (0 to 4) and HDR ratio (1.0 to 64.0)

```
./vvext 2 "HDR" "3, 1.0"
./vvext 2 "HDR" "3, 16.0"
```

Note: *the extension bit parameter is not valid for NXP. This function can be called and takes effect when the sensor mode is in HDR mode.*

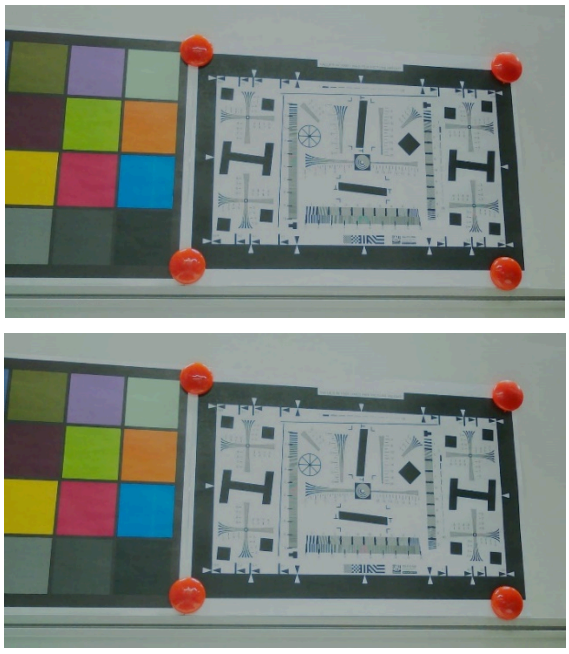


Figure 67. HDR RATIO INPUT Examples

5.4.21 REGGET

- REGGET: gets a register value (1 hexadecimal value)

```
./vvext 2 "REGGET" "addr"
```

5.4.22 REGSET

- REGSET: sets a register value (2 hexadecimal values)

```
./vvext 2 "REGSET" "addr, value"
```

5.4.23 PRELOAD

- PRELOAD: loads the sensor calibration file.

```
./vvext 2 "PRELOAD" 1
```

Note: *This function can only be called before turning the stream on.*

6 Revision History

This table provides the revision history.

Revision history

Document ID	Release date	Description
IMX8MPCDUG v.LF6.6.3_1.0.0	29 March 2024	Upgraded to the 6.6.3 kernel.
IMX8MPCDUG v.LF6.1.55_2.2.0	12/2023	Upgraded to the 6.1.55 kernel.
IMX8MPCDUG v.LF6.1.36_2.1.0	09/2023	Upgraded to the 6.1.36 kernel.
IMX8MPCDUG v.LF6.1.22_2.0.0	06/2023	Upgraded to the 6.1.22 kernel.
IMX8MPCDUG v.LF6.1.1_1.1.0	03/2022	Updated Section 3 and added Section 5.
IMX8MPCDUG v.LF5.15.71_2.2.0	12/2022	Upgraded to the 5.15.71 kernel.
IMX8MPCDUG v.LF5.15.52_2.1.0	09/2022	Upgraded to the 5.15.52 kernel, and added the i.MX 93.
IMX8MPCDUG v.LF5.15.32_2.0.0	06/2022	Upgraded to the 5.15.32 kernel, U-Boot 2022.04, and Kirkstone Yocto.
IMX8MPCDUG v.LF5.15.5_1.0.0	03/2022	Added Chapter 4 and some minor updates.
IMX8MPCDUG v.LF5.10.72_2.2.0	12/2021	Updated for the LF5.10.72_2.2.0 release.
IMX8MPCDUG v.LF5.10.52_2.1.0	10/2021	Major content update for the Linux LF5.10.52_2.1.0 release.
IMX8MPCDUG v.LF5.10.35_2.0.0	06/2021	Upgraded to the 5.10.35 kernel.
IMX8MPCDUG v.L5.4.70_2.3.2	05/2021	Initial release.

Legal information

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.

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 https://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 document 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) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

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.

i.MX — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

Contents

For more information, please visit: https://www.nxp.com

Date of release: 29 March 2024
Document identifier: IMX8MPCDUG