



# Camera Attributes Manual

v1.0.9

June 11, 2020



# Table of Contents

<b>Contact</b>	<b>3</b>
<b>Legal</b>	<b>3</b>
<b>Overview</b>	<b>4</b>
<b>Attribute Reference</b>	<b>5</b>
DeviceInformation	5
ImageFormatControl	6
AcquisitionControl	9
AnalogControl	13
TransportLayerControl	19
GPIO	21
<b>Document History</b>	<b>29</b>



## Contact

### Emergent Vision Technologies Canada (Headquarters)

3135-580 Nicola Ave

Port Coquitlam, BC

V3B 0P2

Canada

[info@emergentvisiontec.com](mailto:info@emergentvisiontec.com)

[www.emergentvisiontec.com](http://www.emergentvisiontec.com)

### Technical Support

[info@emergentvisiontec.com](mailto:info@emergentvisiontec.com)

## Legal

### Trademarks

All trademarks appearing in this document are protected by law.

### Warranty

The information provided is supplied without any guarantees or warranty.

### Copyright

All texts, pictures, files, and graphics are protected by copyright and other laws protecting intellectual property. It is not permitted to copy or modify them for and use.



## Overview

This document is the Camera Attributes manual for Emergent Vision Technologies cameras.

It covers all cameras so not all attributes are available on all cameras or firmware versions. For example, a monochrome camera will not have color image format modes.

Attributes are programmable through the Emergent API documented in the Emergent Programmer's Reference Manual.

Emergent cameras are GenICam compliant so XML files are used to determine attribute functionality. This document puts this attribute information in a more readable format.



## Attribute Reference

### DeviceInformation

See eSDK example “EVT\_DeviceInformation” for usage and functional example.

#### DeviceVendorName

*Access Mode:* RO

*Node Type:* String

Name of the camera vendor: ie. “EVT”

#### DeviceModelName

*Access Mode:* RO

*Node Type:* String

Name of the camera model: ie. “HR-12000SC”

#### DeviceVersion

*Access Mode:* RO

*Node Type:* String

The camera version: ie. “1.0”

#### DeviceSerialNumber

*Access Mode:* RO

*Node Type:* String

The camera serial number: ie. “01200231”

#### DeviceFirmwareVersion

*Access Mode:* RO

*Node Type:* String

The camera firmware version: ie. “3.24”

#### DeviceUserName

*Access Mode:* RW

*Node Type:* String

Name that User may assign. ie. “Camera10”



## SensTemp

*Access Mode:* RO

*Node Type:* Int32

Temperature reading of the sensor in Celsius: ie. 34

## ImageFormatControl

See eSDK example “EVT\_ImageFormatControl” for usage and functional example.

### Width

*Access Mode:* RW

*Node Type:* Uint32

The horizontal size of the ROI.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes such as OffsetX so rechecking limits after other dependent settings changes is recommended.

### Height

*Access Mode:* RW

*Node Type:* Uint32

The vertical size of the ROI.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes such as OffsetY so rechecking limits after other dependent settings changes is recommended.



## PixelFormat

*Access Mode:* RW

*Node Type:* Enumeration

The pixel output mode for the camera. Follows standard naming.

Supported pixel formats vary based on camera model but examples are:

MONO: Mono8  
Mono10  
Mono10Packed  
Mono12  
Mono12Packed

COLOR: BayerGB8/BayerGR8/BayerRG8/BayerBG8  
BayerGB10/BayerGR10/BayerRG10/BayerBG10  
BayerGB10Packed/BayerGR10Packed/BayerRG10Packed/BayerBG10Packed  
BayerGB12Packed/BayerGR12Packed/BayerRG12Packed/BayerBG12Packed  
RGB8Packed  
RGB10Packed  
BGR8Packed  
BGR10Packed  
YUV411Packed  
YUV422Packed  
YUV444Packed

## OffsetX

*Access Mode:* RW

*Node Type:* Uint32

The horizontal offset of the ROI.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes such as Width so rechecking limits after other dependent settings changes is recommended.

## OffsetY

*Access Mode:* RW

*Node Type:* Uint32

The vertical offset of the ROI.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes such as Height so rechecking limits after other dependent settings changes is recommended.



## FlipX

*Access Mode:* RW

*Node Type:* Bool

Flips the image in the X direction when true.

## Iris

*Access Mode:* RW

*Node Type:* Uint32

For cameras using a Birger EF adapter, this value is defined as follows.

0 = initialize the iris mechanism and set iris wide open.

1+ = set the number of stops

Note that if mechanism is confused by values which are too large or if lens is powered/swapped/etc after last initialization then may need to initialize again with Iris = 0.

## Focus

*Access Mode:* RW

*Node Type:* Uint32

For cameras using a Birger EF adapter, this value is defined as follows.

0 = initialize the focus mechanism and set focus to minimum absolute value.

1+ = set the focus to the absolute value

Note that if mechanism is confused by values which are too large or if lens is powered/swapped/etc after last initialization then may need to initialize again with Focus = 0.

## SubSample

*Access Mode:* RW

*Node Type:* Enumeration

Allows user to maintain same field of but skip pixels for lower resolution and data rate.

Higher frame rates can be achieved in subsampled images on some models.

Note that this is not binning as in traditional CCD cameras in that pixels are not summed but rather skipped.

Supported subsample options are:

1x1

2x2

4x4

8x8





## AcquisitionControl

See eSDK example “EVT\_AcquisitionControl” for usage and functional example.

### AcquisitionMode

*Access Mode:* RW

*Node Type:* Enumeration

Continuous: For all modes except when MultiFrame used.

MultiFrame: Use for hardware or software trigger modes where multiple frames are to be triggered by a single hardware pulse or single software trigger.

Used in conjunction with AcquisitionFrameCount.

### AcquisitionStart

*Access Mode:* WO

*Node Type:* Command

The standard GenICam command. In TriggerMode “Off” this command starts streaming images until the AcquisitionStop command is executed. In TriggerMode “On”, this command prepares the camera for triggers selected by TriggerSource and when

AcquisitionStop command is executed the camera is put on standby from such trigger events.

### AcquisitionStop

*Access Mode:* WO

*Node Type:* Command

The standard GenICam command. See AcquisitionStart.

### TriggerMode

*Access Mode:* RW

*Node Type:* Enumeration

Enable or disable non-continuous (ie. Not free-run) trigger modes.

Off: Disabled

On: Enabled

### TriggerSoftware

*Access Mode:* WO

*Node Type:* Command

Command to send a software trigger request to camera

applicable when TriggerMode is enabled and TriggerSource is Software.



## TriggerSource

*Access Mode:* RW

*Node Type:* Enumeration

The trigger sources applicable when TriggerMode is "On".

Software: Will use TriggerSoftware command to trigger captures or when PTPMode is not "Off."

Hardware: Will use GPIO to trigger captures.

## Exposure

*Access Mode:* RW

*Node Type:* Uint32

Sets the camera exposure in micro-seconds.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.

## AutoExposure

*Access Mode:* RW

*Node Type:* Bool

Use to enable auto exposure control. Use AutoExpSet and AutoExpIGain to program set point and integral gain (auto exposure rate) respectively.

## AutoExpSet

*Access Mode:* RW

*Node Type:* Uint32

Auto exposure set point in 10bit grey levels.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## AutoExpIGain

*Access Mode:* RW

*Node Type:* Uint32

Auto exposure rate.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.



## FrameRate

*Access Mode:* RW

*Node Type:* Uint32

Sets the desired frame rate.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.

## FrameRateMHz

*Access Mode:* RW

*Node Type:* Uint32

Sets the desired frame rate in milli-Hz. Convenient for those needing fractional frame rates: ie. 29.97Hz.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.

## LineTime

*Access Mode:* RW

*Node Type:* Uint32

Sets the desired line time to read images out of on-camera memory. This can be used to slow down the data transmission on a line-by-line basis for those softwares that cannot keep up. The parameter is in 150MHz clock cycles but since this parameter is used to slow the continuous data rate it is only necessary to set frame rate to the desired value and then set the line-time parameter to the minimum. At max frame rate the line time min is the same as max and thus cannot be adjusted. At lower frame rates the line time range opens up. The lower the frame rate, the greater the range of adjust in the max line time.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.

## AcquisitionFrameCount

*Access Mode:* RW

*Node Type:* Uint32

The number of frames to capture for a MultiFrame trigger sequence. Setting this to a number greater than 1 enables a multi-frame capture.

Use API functions as in example to get, set, and determine minimum, maximum. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.



## Sync

*Access Mode:* WO

*Node Type:* Command

For using with Myricom Sync NICs w/ IRIG timesource for achieving multicamera synchronization to micro-second levels. This command makes the SyncOffset time offset be applied to the current in-camera timer to push out the start of the next exposure for this particular camera.

Use API functions as in example to get, set, and determine minimum, maximum.

## SyncOffset

*Access Mode:* RW

*Node Type:* Uint32

For using with Myricom Sync NICs w/ IRIG timesource for achieving multicamera synchronization to micro-second levels. This parameter determines the time offset to be applied to the current in-camera timer to push out the start of the next exposure for this particular camera as executed by the Sync command.

Use API functions as in example to get, set, and determine minimum, maximum.

## PtpMode

*Access Mode:* RW

*Node Type:* Enumeration

The mode options for PTP support

Off: PTP is not used

OneStep: PTP is enabled and only PTP "Sync" is processed in calculating the camera internal PTP clock.

TwoStep: PTP is enabled and PTP "Sync" and "Follow-up" are processed in calculating the camera internal PTP clock.

## PtpStatus

*Access Mode:* RO

*Node Type:* Enumeration

The current PTP status.

Disabled: PTPMode is set to off so not enabled.

Listening: PTPMode is enabled in camera but does not have a lock on any incoming PTP clock.

Calibrating: PTP clock has been found and rate calibration being performed if needed.

Slave: PTP clock has been acquired and camera now has a valid and reliable PTP clock.



### PtpAcquisitionGateTimeHigh

*Access Mode:* RW

*Node Type:* Uint32

This is the time set in the future to begin an acquisition. This register is the upper 32 bits of the gate time.

Use API functions as in example to get, set, and determine minimum, maximum. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.

### PtpAcquisitionGateTimeLow

*Access Mode:* RW

*Node Type:* Uint32

This is the time set in the future to begin an acquisition. This register is the lower 32 bits of the gate time.

Use API functions as in example to get, set, and determine minimum, maximum. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is recommended.

### PtpOffset

*Access Mode:* RO

*Node Type:* Int32

This is the offset or worst-case error in nano-seconds between the grandmaster PTP clock and the camera clock upon reception of the updated clock from the grandmaster.

## AnalogControl

See eSDK example “EVT\_AnalogControl” for usage and functional example.

### Gain

*Access Mode:* RW

*Node Type:* Uint32

Gain to apply to the image before transmission. Default value is 256 which equates to 0dB or gain of 1.

The formula for gain is:  $GV/V = Gslider/256$ , or  $GdB = 20\log_{10}(Gslider/256)$ .

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

### PGAGain

*Access Mode:* RW

*Node Type:* Uint32

Sensor analog gain if available.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.



## Offset

*Access Mode:* RW

*Node Type:* Uint32

Offset to apply to the image before transmission.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## OffsetSigned

*Access Mode:* RW

*Node Type:* Int32

Offset to apply to the image before transmission but allows for negative offsets.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## AutoGain

*Access Mode:* RW

*Node Type:* Bool

Use to enable auto gain control. Use AutoGainSet and AutoGainIGain to program set point and integral gain (auto gain rate) respectively.

## AutoGainSet

*Access Mode:* RW

*Node Type:* Uint32

Auto gain set point in 10bit grey levels.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## AutoGainIGain

*Access Mode:* RW

*Node Type:* Uint32

Auto gain rate.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## LUTEnable

*Access Mode:* RW

*Node Type:* Bool

Use to enable LUT. Use LUTIndex and LUTValue to program the LUT (look-up table).



## LUTIndex

*Access Mode:* RW

*Node Type:* Uint32

The address to the given LUT entry. Some cameras are 10bit max and others 12bit which changes how many elements are used: 1024 vs 4096.

The LUT is expected to be programmed sequentially so the following process should be used:

LUTIndex=0

LUTValue=v0

LUTIndex=1

LUTValue=v1

LUTIndex=2

LUTValue=v2

...

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## LUTValue

*Access Mode:* RW

*Node Type:* Uint32

The value to program to the given LUT address LUTIndex. Some cameras are 10bit max and others 12bit which changes how many elements are used: 1024 vs 4096.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## WB\_R\_GAIN\_Value

*Access Mode:* RW

*Node Type:* Uint32

The value to program for red channel white balance gain. Its use is the same as the Gain parameter.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## WB\_GR\_GAIN\_Value

*Access Mode:* RW

*Node Type:* Uint32

The value to program for green channel (Green on the Bayer pattern red line) white balance gain.

Its use is the same as the Gain parameter.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.



### WB\_GB\_GAIN\_Value

*Access Mode:* RW

*Node Type:* Uint32

The value to program for green channel (Green on the Bayer pattern blue line) white balance gain.

Its use is the same as the Gain parameter.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

### WB\_B\_GAIN\_Value

*Access Mode:* RW

*Node Type:* Uint32

The value to program for blue channel white balance gain.

Its use is the same as the Gain parameter.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

### WB\_Enable

*Access Mode:* RW

*Node Type:* Bool

Use to enable Auto White Balance mode. Based on image means for all bayer channels being equal.

### WB\_Hold

*Access Mode:* RW

*Node Type:* Bool

Use to enable Auto White Balance Hold mode. Once an acceptable white balance is achieved through the auto white balance function one can enable this to hold the current white balance gains. Disable this to continue the auto white balance function.

### CCMIndex

*Access Mode:* RW

*Node Type:* Uint32

The address to the given CCM entry. The CCM matrix and the operation is defined as:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} Ro \\ Go \\ Bo \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} Ri \\ Gi \\ Bi \end{bmatrix}$$





The CCM is expected to be programmed sequentially so the following process should be used:

```
CCMIndex=0
CCMValue=A11
CCMIndex=1
CCMValue=A12
CCMIndex=2
CCMValue=A13
CCMIndex=3
CCMValue=A21
CCMIndex=4
CCMValue=A22
CCMIndex=5
CCMValue=A23
CCMIndex=6
CCMValue=A31
CCMIndex=7
CCMValue=A32
CCMIndex=8
CCMValue=A33
```

This is a rarely used parameter since we have factory calibrated CCM tables for different color temperatures which can be used through the ColorTemp parameter.

For more information on the CCM process, one can visit: <http://www.imatest.com/docs/colormatrix/>

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## CCMValue

*Access Mode:* RW

*Node Type:* Uint32

Please see CCMIndex.

## ColorTemp

*Access Mode:* RW

*Node Type:* Enumeration

Select the color temperature of the lighting to utilize the factory calibrated CCMs.

CT\_Off: CCM matrix is the identity matrix (ie. CCM feature not in use)

CT\_2800K: 2800 Kelvin

CT\_3000K: 3000 Kelvin

CT\_4000K: 4000 Kelvin

CT\_5000K: 5000 Kelvin

CT\_6500K: 6500 Kelvin

CT\_Custom: Values used with user setting of CCMIndex/CCMValue.



## AutoSizeX

*Access Mode:* RW

*Node Type:* Uint32

AutoSizeX, AutoSizeY, AutoOffsetX, AutoOffsetY define the ROI for all auto functionality including auto exposure, auto gain, and auto white balance.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is needed.

## AutoSizeY

*Access Mode:* RW

*Node Type:* Uint32

AutoSizeX, AutoSizeY, AutoOffsetX, AutoOffsetY define the ROI for all auto functionality including auto exposure, auto gain, and auto white balance.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is needed.

## AutoOffsetX

*Access Mode:* RW

*Node Type:* Uint32

AutoSizeX, AutoSizeY, AutoOffsetX, AutoOffsetY define the ROI for all auto functionality including auto exposure, auto gain, and auto white balance.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is needed.

## AutoOffsetY

*Access Mode:* RW

*Node Type:* Uint32

AutoSizeX, AutoSizeY, AutoOffsetX, AutoOffsetY define the ROI for all auto functionality including auto exposure, auto gain, and auto white balance.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps. Note that limits are influenced by other nodes so rechecking limits after other dependent settings changes is needed.

## DarkFPNEnable

*Access Mode:* RW

*Node Type:* Bool

Use to enable dark level FPN correction with Line Scan cameras.



### DarkFPNCal

*Access Mode:* WO

*Node Type:* Command

With lens cap on or no light we execute this command to calibrate the dark level FPN.

When this command is executed, the image height must be  $\geq 1024$

### LitFPNEnable

*Access Mode:* RW

*Node Type:* Bool

Use to enable lit level FPN correction with Line Scan cameras.

### LitFPNCal

*Access Mode:* WO

*Node Type:* Command

With uniform and non-static lighting we execute this command to calibrate the lit level FPN.

When this command is executed, the image height must be  $\geq 1024$

## TransportLayerControl

See eSDK example “EVT\_TransportLayerControl” for usage and functional example.

### GevSCPSPacketSize

*Access Mode:* RW

*Node Type:* Uint32

The Ethernet/IP (MTU) packet size to use for image transmission.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

### HTimeout

*Access Mode:* RW

*Node Type:* Uint32

This defines the timeout in milli-seconds during which the camera must receive a heartbeat command from the host application to remain available and connected for this primary host application.

The default value is 3000 (3 seconds as defined by the GigEVision specification).

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.



## **HBDisable**

*Access Mode:* RW

*Node Type:* Bool

This defines whether the camera is in heartbeat mode or not. When set to FALSE, the camera is expecting a heartbeat command within every HBTimeout seconds else it will disconnect and be available for host applications to connect to. When set to FALSE, the eCapture or eSDK software creates a heartbeat thread in the background to send these heartbeats. If the camera disconnects due to absence of heartbeats, then the API will return error when eventually a command is sent from the then controlling application which is a sign that an application needs to re-open the camera.

This feature is useful in the cases of PC or software crashes as well as cases where link has gone down because of broken cable or other hardware malfunction without the need then for power cycling the camera.

Use to enable auto exposure control. Use AutoExpSet and AutoExpIGain to program set point and integral gain (auto exposure rate) respectively.

## **GevTimestampControlLatch**

*Access Mode:* WO

*Node Type:* Command

The standard GigEVision/GenICam command which tells the camera upon receiving this command to latch the current timestamp counter for subsequent reading via GevTimestampValueHigh and GevTimestampValueLow.

## **GevTimestampControlReset**

*Access Mode:* WO

*Node Type:* Command

The standard GigEVision/GenICam command which tells the camera upon receiving this command to reset or zero the current timestamp counter.

## **GevTimestampValueHigh**

*Access Mode:* RW

*Node Type:* Uint32

This is the upper 32 bits of the currently latched value of the Timestamp counter.

In PTP modes, this is the PTP clock in the camera.

## **GevTimestampValueLow**

*Access Mode:* RW

*Node Type:* Uint32

This is the lower 32 bits of the currently latched value of the Timestamp counter.

In PTP modes, this is the PTP clock in the camera.



### **GevTimestampTickFrequencyHigh**

*Access Mode:* RO

*Node Type:* Uint32

This is the upper 32 bits of the frequency in Hz of the clock in the camera used to define what each tick of GevTimestampValue means. ie. 150000000 means 150MHz so that each count of GevTimestampValue is  $1/150\text{MHz} = 6.66\text{ns}$ .

### **GevTimestampTickFrequencyLow**

*Access Mode:* RO

*Node Type:* Uint32

This is the lower 32 bits of the frequency in Hz of the clock in the camera.

## **GPIO**

See eSDK example “EVT\_GPIO” for usage and functional example.

### **GPO\_n\_Mode**

*Access Mode:* RW

*Node Type:* Enumeration

The mode to use for the nth GPO output. ie. GPO\_0\_Mode. IO is defined in the User’s Manual.

Exposure: Output a pulse that is active during exposure.

Readout: Output a pulse that is active during readout.

GPO: Use as general purpose output set by polarity.

Test\_Generator: To generate frame time clock of varying duty cycle. See TG\_Frame\_time, TG\_High\_time

### **GPO\_n\_Polarity**

*Access Mode:* RW

*Node Type:* Bool

The polarity of the nth GPO output. Positive or negative. ie. GPO\_0\_Polarity

### **GPI\_n\_Polarity**

*Access Mode:* RO

*Node Type:* Bool

Read this parameter to determine the current state of the nth GPI input. ie. GPI\_4\_Polarity.



### GPI\_Start\_Exp\_Mode

*Access Mode:* RW

*Node Type:* Enumeration

The GPIO input to use for start exposure from external trigger.

It may be the same input as for end exposure. ie.

GPI\_4

GPI\_5

### GPI\_Start\_Exp\_Event

*Access Mode:* RW

*Node Type:* Enumeration

The event that triggers the start of exposure.

*Rising\_Edge:* The input signal rising edge triggers.

*Falling\_Edge:* The input signal falling edge triggers.

### GPI\_End\_Exp\_Mode

*Access Mode:* RW

*Node Type:* Enumeration

The GPIO input to use for end exposure from external trigger.

It may be the same input as for start exposure.

GPI\_4

GPI\_5

Internal : exposure start based on trigger input but exposure end based off Exposure register.

### GPI\_End\_Exp\_Event

*Access Mode:* RW

*Node Type:* Enumeration

The event that triggers the end of exposure.

*Rising\_Edge:* The input signal rising edge triggers.

*Falling\_Edge:* The input signal falling edge triggers.

### GPI\_n\_Debounce\_Count

*Access Mode:* RW

*Node Type:* Uint32

Value in 50MHz clock cycle counts. For example, a count of 1000 will provide a debounce time of  $1000/50\text{MHz} = 20\mu\text{s}$ . Debounce is used when triggering from external signals via GPI\_n allows the camera to compensate for sampling input trigger signals with different signal rise and fall times. Generally, setting this time to be greater than the worst-case input rise and fall time should ensure proper operation.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.



## Trigger\_Delay

*Access Mode:* RW

*Node Type:* Uint32

Value in us to set for trigger delay time. The externally triggered exposure will be delayed relative to the external triggering signal by the amount specified in this register.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## TG\_Frame\_Time

*Access Mode:* RW

*Node Type:* Uint32

Value in us to set as period of frame time clock out of any GPO. Care must be taken to ensure that reasonable values are set for this parameter with consideration to exposure, readout time, etc. The safest way is to determine the frame rate/time limits using the simple internal continuous mode.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## TG\_High\_Time

*Access Mode:* RW

*Node Type:* Uint32

Value in us to set as high time of frame time clock out of any GPO. Care must be taken to ensure that reasonable values are set for this parameter with consideration to exposure, readout time, etc. The safest way is to determine the frame rate/time limits using the simple internal continuous mode.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## UartEnable

*Access Mode:* RW

*Node Type:* Bool

Use to enable/disable Uart mode. In Uart mode, the Uart receive data come into the camera on GPI5 and the Uart transmit data goes out GPO3. These IO are reserved for Uart functionality in this mode.

Many customers use the Uart functionality to control devices such as the Birger Canon EF mount module



## UartBaud

*Access Mode:* RW

*Node Type:* Emuneration

The baud rate of the Uart interface is set through this parameter.

Higher baud rates may have bandwidth restrictions depending how the IO are configured.

B\_9600

B\_19200

B\_38400

B\_57600

B\_115200

## UartDataBits

*Access Mode:* RW

*Node Type:* Uint32

Number of data bits for the Uart interface.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## UartStopBits

*Access Mode:* RW

*Node Type:* Uint32

Number of stop bits for the Uart interface.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

## UartTxData

*Access Mode:* WO

*Node Type:* Uint32

Writing this parameter writes a word to the Uart transmit FIFO which will be shortly followed by its transmission on the Uart transmit interface.

## UartTxFifoCnt

*Access Mode:* RO

*Node Type:* Uint32

Read this parameter to find out how many words are in the Uart transmit FIFO (ie. Not yet transmitted).





### UartRxData

*Access Mode:* RO

*Node Type:* Uint32

Read this parameter to read a word out of the receive FIFO.

### UartRxFifoCnt

*Access Mode:* RO

*Node Type:* Uint32

Read this parameter to find out how many words are in the Uart receive FIFO (ie. Not yet read out).

### GPI\_Start\_Frame\_Mode

*Access Mode:* RW

*Node Type:* Enumeration

The GPIO input to use for start frame from external trigger.

GPI\_1

GPI\_2

Applicable to Line Scan cameras. See User's Manual for more details.

### GPI\_Start\_Frame\_Event

*Access Mode:* RW

*Node Type:* Enumeration

The event that triggers the start of frame.

*Rising\_Edge:* The input signal rising edge triggers.

*Falling\_Edge:* The input signal falling edge triggers.

*Encoder\_Frame\_Divider:* The internal encoder frame divider counter triggers.

*Pulse\_High:* The camera triggers by encoder frame divider while GPI input is high.

*Pulse\_Low:* The camera triggers by encoder frame divider while GPI input is low.

Applicable to Line Scan cameras. See User's Manual for more details.

### GP\_ENC\_MODE

*Access Mode:* RW

*Node Type:* Bool

Set to True to enable use of external encoder in triggering.

Applicable to Line Scan cameras. See User's Manual for more details.



## GP\_ENC\_TERMINATION

*Access Mode:* RW

*Node Type:* Bool

Set to True to enable use of 120 Ohms termination on the RS422 differential encoder input.

Applicable to Line Scan cameras. See User's Manual for more details.

## GP\_ENC\_DIRECTION

*Access Mode:* RW

*Node Type:* Bool

True or false determines the direction of the encoder rotation.

Applicable to Line Scan cameras. See User's Manual for more details.

## GP\_ENC\_LINE\_DIVIDER

*Access Mode:* RW

*Node Type:* Uint32

This value determines how many encoder pulses are needed to trigger a single line.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

Applicable to Line Scan cameras. See User's Manual for more details.

## GP\_ENC\_FRAME\_DIVIDER

*Access Mode:* RW

*Node Type:* Uint32

This value determines how many encoder ticks will pass before another frame is started.

Use API functions as in example to get, set, and determine minimum, maximum, and quantization/increment steps.

Applicable to Line Scan cameras. See User's Manual for more details.



## GP\_SPAT\_CORR

*Access Mode:* RW

*Node Type:* Enumeration

Allows spatial correction which accounts for line gaps in tri-linear sensors.

p8 : 8 lines in positive direction.

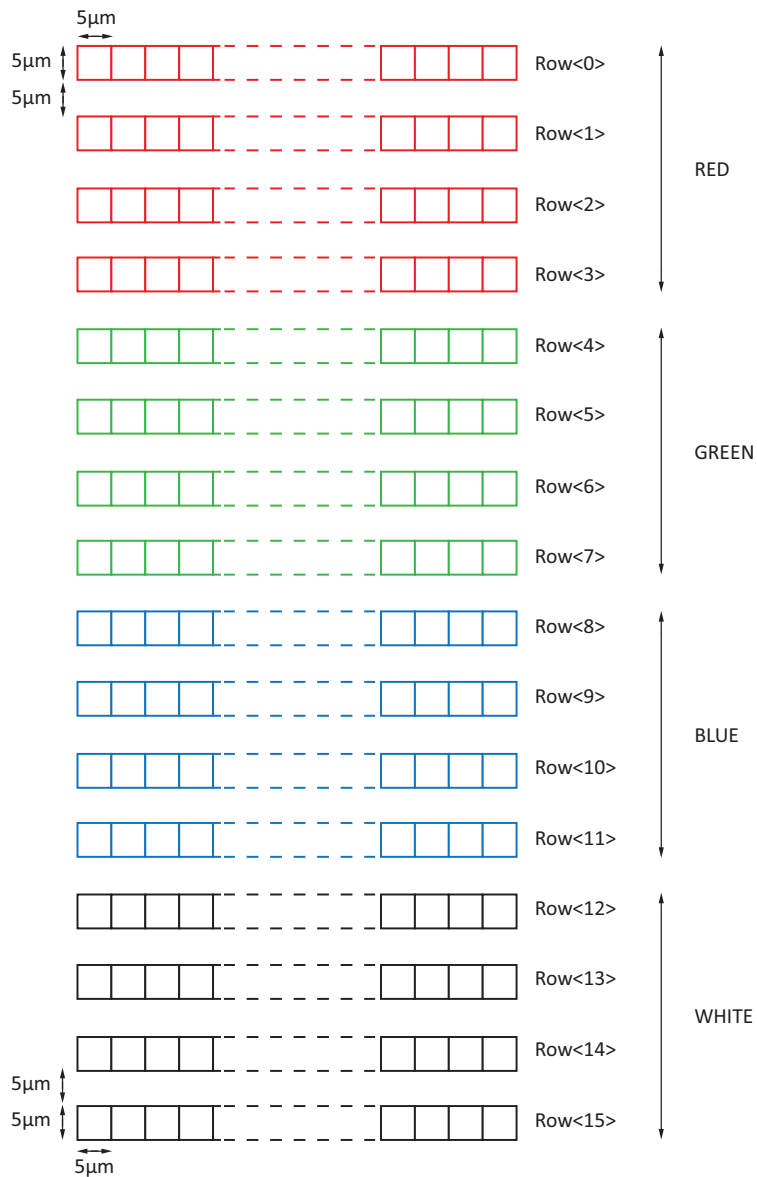
Off : Spatial correction disabled.

n8 : 8 lines in negative direction.

Applicable to Line Scan cameras.

The value of 8 is chosen based on the 8x5 $\mu$ m spacing between the different colors.

One color is chosen from each color subgroup to create the overall RGB pixels.





### GP\_ENC\_TIMEOUT

*Access Mode:* RW

*Node Type:* Uint32

This value is used to determine how long the internal logic should wait before considering the current frame aborted due to absence or slowing of encoder pulses. Measured in 150MHz cycles.

### GP\_ENC\_LINE\_DIV\_MEAS

*Access Mode:* RO

*Node Type:* Uint32

This value is the measured period of the line trigger pulses at the output of the divider set by GP\_ENCODER\_LINE\_DIVIDER.

This measurement helps in determining the appropriate value of GP\_ENC\_TIMEOUT. Measured in 150MHz cycles.

### GP\_ENC\_FRAME\_DIV\_MEAS

*Access Mode:* RO

*Node Type:* Uint32

This value is the measured period of the frame trigger pulses at the output of the divider set by GP\_ENCODER\_FRAME\_DIVIDER.

Measured in 150MHz cycles.



## Document History

Version	Date	Description
1.01	29 May 2012	Initial Version
1.02	8 March 2013	Added HDR Mode
1.03	25 June 2013	Added GPI Debounce Option
1.04	20 July 2013	Added Auto white balance and hold functions
1.05	10 December 2013	Clarified Gain operation
1.06	2 May 2014	Added Uart, Sync, Line Time, GPI4/5 Polarity parameters
1.07	1 December 2014	Added Subsample parameters
1.08	15 July 2019	Rewrite
1.09	11 June 2020	Updated line-scan information