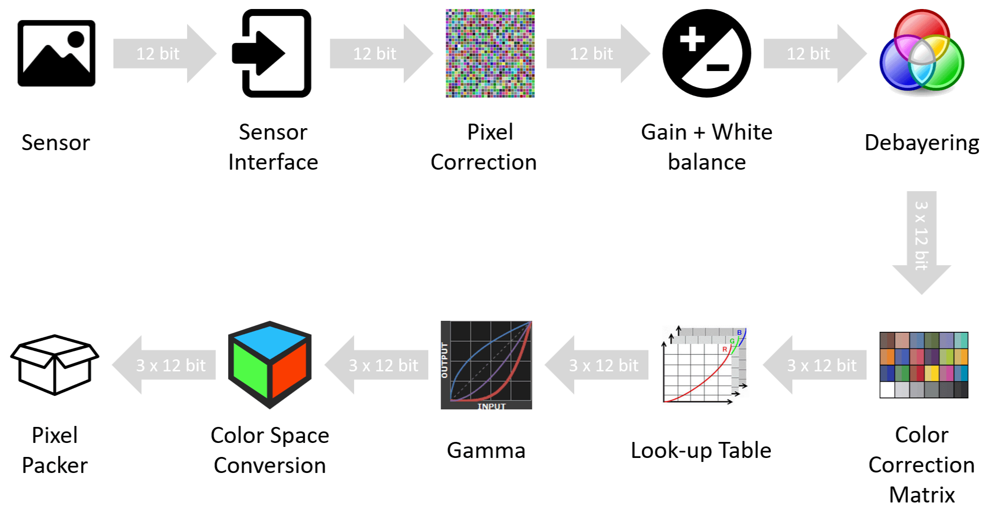
Camera Features

Image Processing Controls

The PHX081S camera has the following image processing control flow. However, the color-related image processing controls (i.e., Debayering, Color Correction Matrix, and Color Space Conversion Conversion) are not available for monochrome cameras.



The details of each of the image processing controls are described below.

Defect Pixel Correction

This feature corrects defective pixels such as dead pixels and hot pixels in the camera. From a preloaded list of defective pixels, the PHX081S replaces each of these pixels with a value interpolated from its neighboring pixels. The list of defective pixels is loaded on the camera during the camera manufacturing process. You can update the pixel correction list at any time using the steps shown below.

Defective pixels are an inevitable part of the semiconductor manufacturing process. As the camera operates for extended periods at elevated temperatures or is exposed to radiation, more defective pixels may appear. Hot pixels can intensify with greater exposure time.

Steps to add a new pixel to the correction list

1. Set OffsetX and OffsetY to zero. Set Width and Height to the maximum allowed value.
2. Set Gain to zero and note the coordinates of any bright pixels in the image. Ensure the camera is not exposed to light by covering it with a lens cap and placing in a dark box.
3. Fire the DefectCorrectionGetNewDefect command.
4. Enter the X-coordinate noted in step 2 into DefectCorrectionPositionX.
5. Enter the Y-coordinate noted in step 2 into DefectCorrectionPositionY.
6. Fire the DefectCorrectionApply command.
7. Repeat steps 3-6 as needed and fire the DefectCorrectionSave command when done.

Note

Pixel correction is still applied if the image geometry changes (e.g. applying ReverseX, ReverseY, a region of interest, or binning).

The following pseudocode demonstrates how to add a defective pixel to the correction list:

```
// Connect to camera
// Get device node map
// Set maximum width and height
OffsetX = 0;
OffsetY = 0;
Width = Max Width;
Height = Max Height;
// Set constant ExposureTime and Gain

DefectCorrectionGetNewDefect();
DefectCorrectionPositionX = 25;  // The X-coordinate of the blemish pixel
DefectCorrectionPositionY = 150; // The Y-coordinate of the blemish pixel
DefectCorrectionApply();

// Repeat the above four steps as needed. When complete:
DefectCorrectionSave();
```

Gain

Gain is a feature that allows the camera user to increase the brightness of an image. More specifically, gain is a multiplication factor applied to a signal to increase the strength of that signal. On Lucid cameras, gain can be either manually adjusted or automatically controlled.

Some cameras feature gain that is purely digital while others allow for analog gain control up to a certain value, beyond which the gain becomes digital. Depending on the camera family and sensor model, the specific gain control can vary.

Analog Gain

Analog Gain refers to amplification of the sensor signal prior to A/D conversion.

Digital Gain

Digital Gain refers to amplification of the signal after digitization.

Model	Analog	Digital
PDH050S	0-24 dB	24-48 dB
PHX004S	0-24 dB	24-48 dB
PHX016S	0-24 dB	24-48 dB
PHX023S	0-24 dB	24-48 dB
PHX032S	0-24 dB	24-48 dB
PHX050S	0-24 dB	24-48 dB
PHX050S-(P/Q)	0-24 dB	24-48 dB
PHX050S1-(P/Q)	0-24 dB	24-48 dB
PHX064S	0-24 dB	24-48 dB
PHX089S	0-24 dB	24-48 dB
PHX120S	0-24 dB	24-48 dB
PHX122S	0-27 dB	NA
PHX200S	0-27 dB	NA

The following pseudocode demonstrates how to set Gain to 12dB:

```
// Connect to camera
// Get device node map
GainAuto = Off;
Gain = 12;
```

Color Processing

The Debayering core in the image processing pipeline is a color processing core that enables the camera to output a color processed image format in addition to the unprocessed Bayer-tiled image. The camera supports the RGB8 pixel format, which outputs 8 bits of data per color channel for a total of 24 bits per pixel. Due to the number of bits per pixel, the total image size for RGB8 is three times larger than an 8-bit image. This increase in image data size per frame reduces the average frame rate of the camera.

The following pseudocode demonstrates how to configure the camera to RGB8 pixel format:

```
// Connect to camera
// Get device node map
PixelFormat = PixelFormat_RGB8;
```

White Balance

This feature balances the Red, Green, and Blue channels so that a white object appears white in acquired images. Lucid cameras allow for manual white balance adjustment by the user, or automatic white balance adjustment based on statistics of previously acquired frames. Different external illuminations and different sensors may render acquired images with color shift. The White Balance module allows the user to correct for the color shift by adjusting gain value of each color channel.

Lucid offers two types of white balance algorithms as described below. Each algorithm below allow for user controlled anchor points or reference points, from which multipliers are computed for each channel. The different anchor points are summarized below.

Anchors	Information
Min	The lowest luminance channel is used as reference while other channels are adjusted to match it. There is no chance of overflowing the pixels, but the image is darkened.
Max	The highest luminance channel is used as reference while other channels are adjusted to match it. There is a chance of overflowing the pixels.
Mean	The mean value of all channels is used as reference while all channels are adjusted to match the mean. There is a smaller chance of overflowing.
Green	Green channel is used as the reference while the Red and Blue are adjusted.

Grey World

The Grey World algorithm assumes that the average of all colors in an image is a neutral grey.

White Patch

The White Patch algorithm has the same idea as Grey World, but only considers a section of the image (i.e. the section being the white patches). A simple way to determine such section(s) of the image is to indicate a pixel as white when R+G+B is greater than the threshold pixel value. Determining the threshold can be done using a 90% percentile of previous image. There is also a need for an additional threshold to exclude saturated pixels for better white balance adjustment.

Look-Up Table (LUT)

A Look-Up Table (LUT) lets users relate raw sensor pixel values to other values that they specify.

Users input values for the even indices including the last index 4095 while averaging is used to calculate the rest of the odd indices. This results in a total of 2049 effective input entries: 2048 even (e.g., 0, 2, 4, ..., 4092, 4094) + 1 odd (4095). Index value 0 corresponds to black while the index value 4095 corresponds to white.

To build a LUT, users input 12-bit index values (e.g., 0, 2, 4, ..., 4092, 4094, 4095) that need to be replaced in the LUTIndex field and the corresponding new value in the LUTValue field. For the odd index values in the gap (e.g., 1,3,5 ...,4089, 4091, 4093), their mapped value is calculated by taking the average of their neighbor mapped values (e.g., If the input mappings are LUTIndex = 1090 -> LUTValue = 10 and LUTIndex = 1092 -> LUTValue = 20, then the mapped value of LUTIndex = 1091 will be LUTValue = 15)

Note

To reset the Look-Up Table, execute the **LUTReset** command

The following pseudocode demonstrates how to replace black pixel values with white pixel values:

```
// Connect to camera
// Get device node map

LUTEnable = true;
LUTIndex = 0; //Pixel value to be replaced, in this case black
LUTValue = 4095; //New pixel value, in this case white
LUTSave();
```

Gamma

The gamma control allows the optimization of brightness for display. Lucid implements Gamma using the GenICam standard as shown below.

$$X = Y^{Gamma}$$

- X = New pixel value; 0 <= X <=1
- Y = Old pixel value; 0 <= Y <=1
- Gamma = Pixel intensity; 0.2 <= Gamma <= 2

Y in the Gamma formula is scaled down to [0-1] from the original pixel range, which results in a pixel range of [0-1] for X. Thus, for 12-bit pixel formats, this means scaling down the pixel range from [0-4095] to [0-1] and for 16-bit pixel formats, this means scaling down the pixel range from [0-65535] to [0-1].

The camera applies gamma correction values to the intensity of each pixel. In general, gamma values can be summarized as follows:

- Gamma = 1: brightness is unchanged.
- 1 <= Gamma <= 2: brightness decreases.

- 0.2 <= Gamma <= 1: brightness increases.

Color Space Conversion and Correction

The Color Space Conversion control allows you to convert from RGB color space to another color space such as YUV. The conversion is done in a linear manner as shown in the following equation.

$$\begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} Gain_{00} & Gain_{01} & Gain_{02} & Offset_0 \\ Gain_{10} & Gain_{11} & Gain_{12} & Offset_1 \\ Gain_{20} & Gain_{21} & Gain_{22} & Offset_2 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \\ 1 \end{bmatrix}$$

The color correction function lets you select from several preset values or configure your own matrix values. The color correction enables the multiplication of a 3x3 matrix to the 3x1 matrix containing R, G and B pixel values to achieve more desirable R', G' and B' values. The specific mathematical procedure can be represented by the following.

$$\begin{bmatrix} Gain_{00} & Gain_{01} & Gain_{02} \\ Gain_{10} & Gain_{11} & Gain_{12} \\ Gain_{20} & Gain_{21} & Gain_{22} \end{bmatrix} \begin{bmatrix} R_{in} \\ G_{in} \\ B_{in} \end{bmatrix} + \begin{bmatrix} Offset_0 \\ Offset_1 \\ Offset_2 \end{bmatrix} = \begin{bmatrix} R_{out} \\ G_{out} \\ B_{out} \end{bmatrix}$$

- The ColorTransformationEnable node indicates whether the conversion matrix of the color space conversion module is used or bypassed.
- When the pixel format is YUV or YCbCr, ColorTransformationSelector is displayed as RGBtoYUV. When the pixel format is Mono, ColorTransformationSelector is displayed as RGBtoY.
- ColorTransformationValueSelector configures which coefficient in the conversion matrix and the coefficient value is shown in ColorTransformationValue. In Mono, YUV or YCbCr pixel formats, ColorTransformationValue is read only.

Pixel Packer

The pixel packer is a module on the ISP (image signal processor) pipeline that processes image data into the configured pixel format for the output from the camera.

Note

For more information on pixel formats, see our related knowledge base article.

QOI (Quite OK Image Format)

The QOI format is an open-source specification for lossless image compression. One or more QOI image formats, which are each prefixed with “QOI”, may be available on your camera model.

To configure your camera to use the QOI format:

1. Configure the PixelFormat node to a choose a QOI pixel format (e.g., QOI_Mono8).
2. Set the AcquisitionFrameRateLinkLimitEnable node to False. (This node configures whether the acquisition frame rate is constrained to prevent the device bandwidth from being exceeded. This node must be set to False so to enable the frame rate to to be increased past this constraint.)
3. Set the ISPClockSpeed node to Fast.

Note

Using the above configuration with non-QOI pixel formats may lead to dropped frames if the resulting bandwidth is beyond 1 Gbps. The host is notified of these dropped frames with a Frame Dropped event.

Triggering

In trigger mode, images are captured asynchronously in response to an external signal or event, enabling precise control over the timing of each image capture. Triggering is often used in applications where timing is critical, such as when objects move through a camera’s field of view at unpredictable intervals. By relying on external triggers like sensors or software signals, the system ensures that images are captured exactly when needed for accurate analysis.

Overview

Triggering a camera includes putting the camera in Acquisition mode, sending the trigger to the camera, and capturing the image.

Entering Acquisition mode

Before a camera can accept triggers, the Acquisition Start command is issued to put the camera in a state where it can accept trigger signals. Depending on the configuration, one or more triggers can be issued after the camera receives the Acquisition Start command.

For more details, see the Acquisition Control page.

Triggering the Camera

A trigger is fired externally that is received by the camera. In response, the camera initiates image capture. Configuring the trigger is described in more detail below.

Capturing the Image

When the trigger is received, the camera does the following to capture an image:

- Resets the electrical charge on the image sensor's pixels to prepare the sensor for a new image capture. This process ensures that the exposure starts exactly when the trigger occurs, preventing any residual charge from affecting the accuracy of the image and and to better capture fast-moving scenes.
- Exposes the image sensor in the camera to light from the scene. This step is called *exposure* or *integration*.
- Reads out the image data from the image sensor for further processing.
- Transmits the image data to the host.

Note

Unless otherwise stated, all timing diagrams below are for global shutter cameras.

Note

You cannot trigger a camera faster than its frame rate while in Free-run mode.

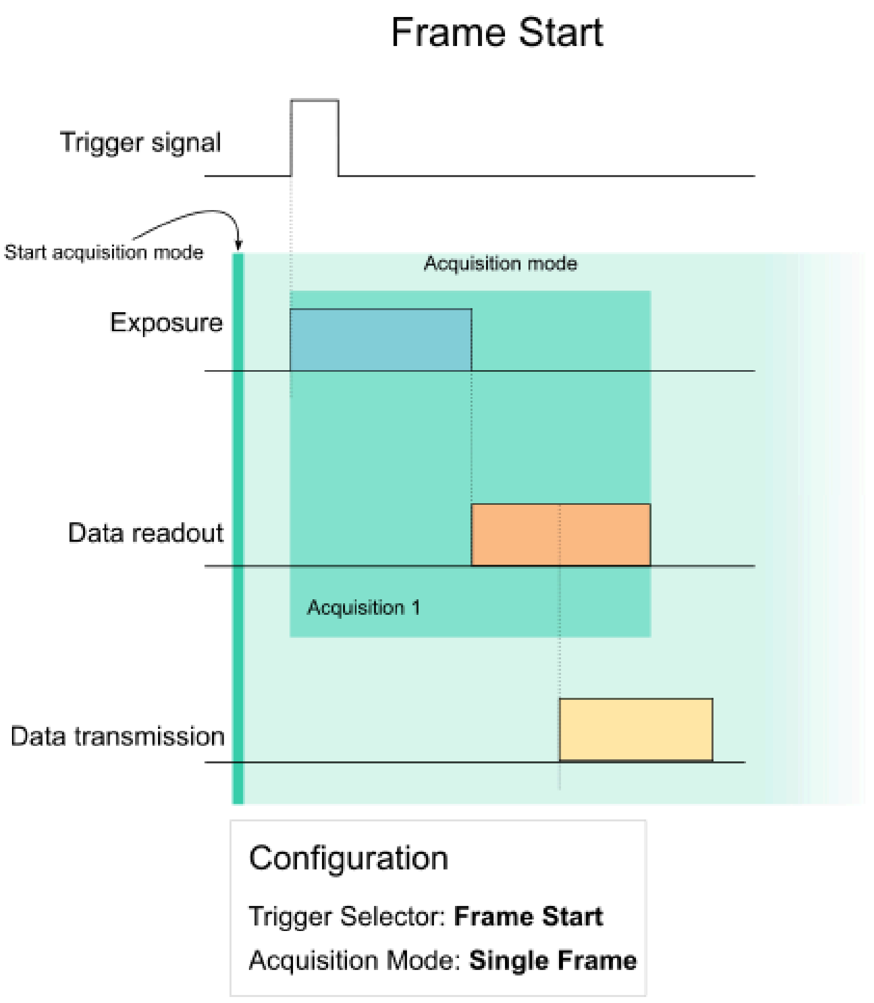
Trigger Configuration Details

Different trigger modes can be configured using the various nodes on your camera, including the Trigger Selector and Trigger source nodes.

Trigger Selector

The Trigger Selector node determines which trigger is active for the camera.

- **Frame Start**: This trigger initiates the capture of one frame. It is used with the Acquisition Mode node set to Single Frame.



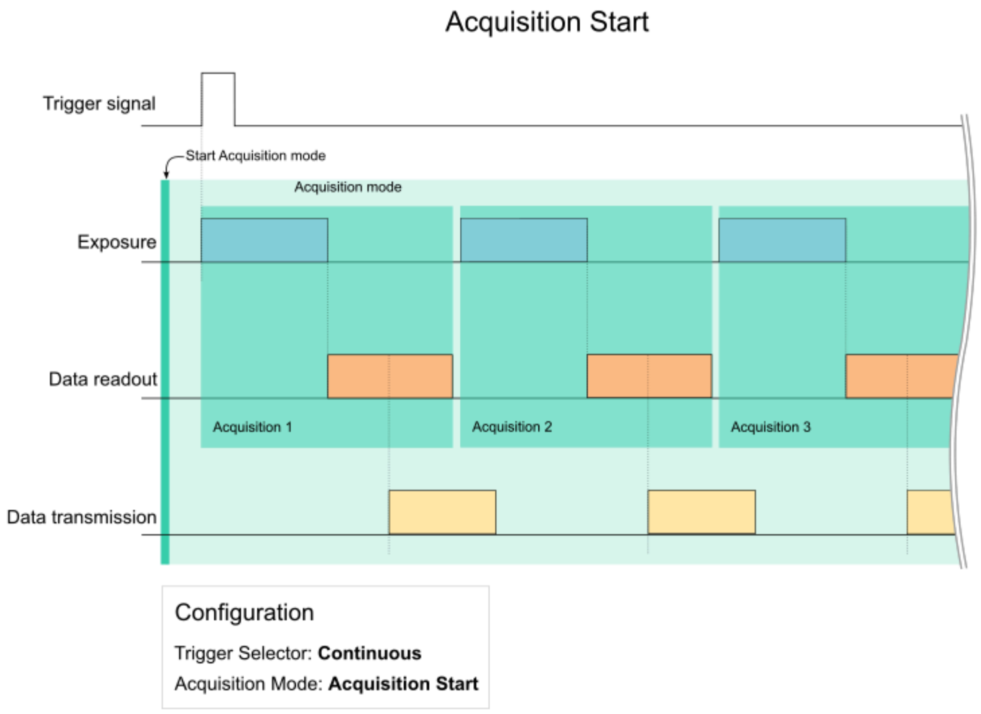
Configuration

Trigger Selector: **Frame Start**

Acquisition Mode: **Single Frame**

After entering Acquisition mode, the camera detects a trigger signal and acquires a single image. Another trigger would make the camera acquire another image.

- **Acquisition Start**: This trigger makes the camera start streaming in Free-run mode, which means allowing the camera to acquire images using its own internal clock. Each frame after the first one is acquired at the frequency defined by AcquisitionFrameRate node. Capturing images in this way gives you a slightly faster frame rate than the most rapid possible triggering.



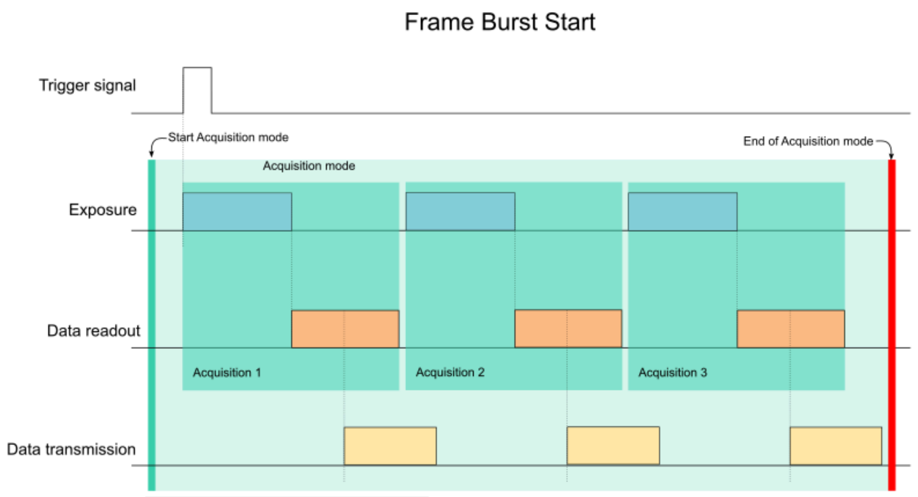
Configuration

Trigger Selector: **Continuous**

Acquisition Mode: **Acquisition Start**

After entering Acquisition mode, the camera detects a trigger and goes into Free-run mode. The timing of the subsequent images is determined by the camera settings similarly to images captured in Free-run mode.

- **Frame Burst Start**: This trigger initiates the capture of multiple frames. The number of acquired frames is defined by the Acquisition Burst Frame Count node. Each frame after the first one is acquired at the frequency defined by AcquisitionFrameRate node.

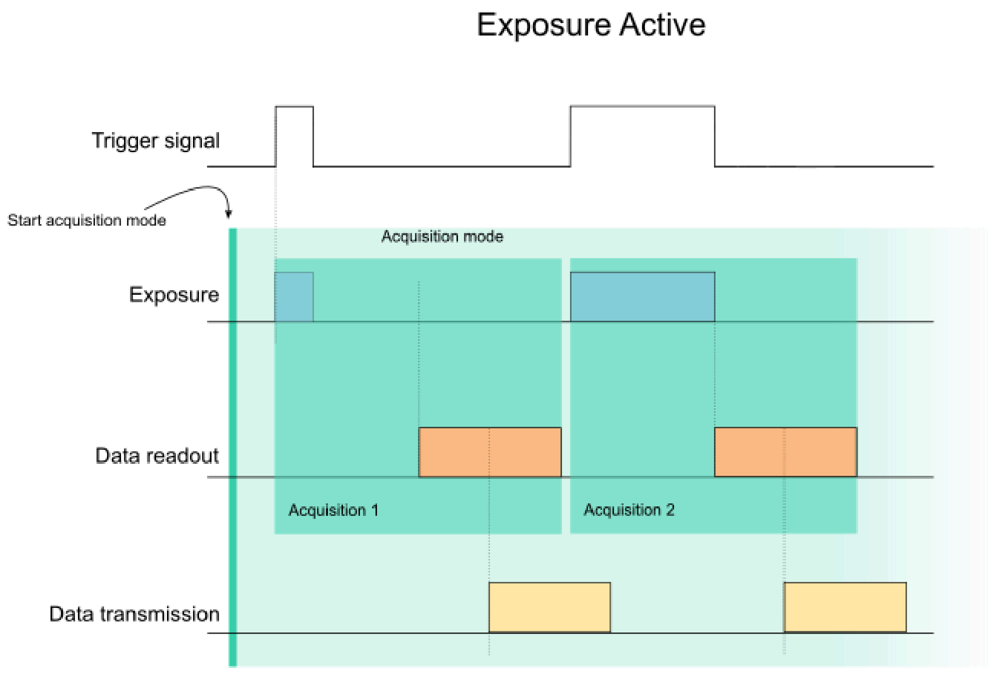


Configuration

Trigger Selector: **Frame Burst Start**
Acquisition Mode: **Multi Frame**
Acquisition Frame Count: **3**
Acquisition Burst Frame Count: **3**

After entering Acquisition mode, the camera detects a trigger and acquires the configured number of images. The timing of the subsequent images is determined by the camera settings similarly to images captured in Free-run mode. After the end of Acquisition Frame Count (3 frames), the camera exits Acquisition mode. Before accepting another trigger to take more images, the camera must be put in Acquisition mode again.

- **Exposure Active**: The exposure starts when the trigger signal becomes active, and lasts until the trigger signal is deactivated. In this mode, the duration of the exposure is controlled by the width of the trigger pulse.

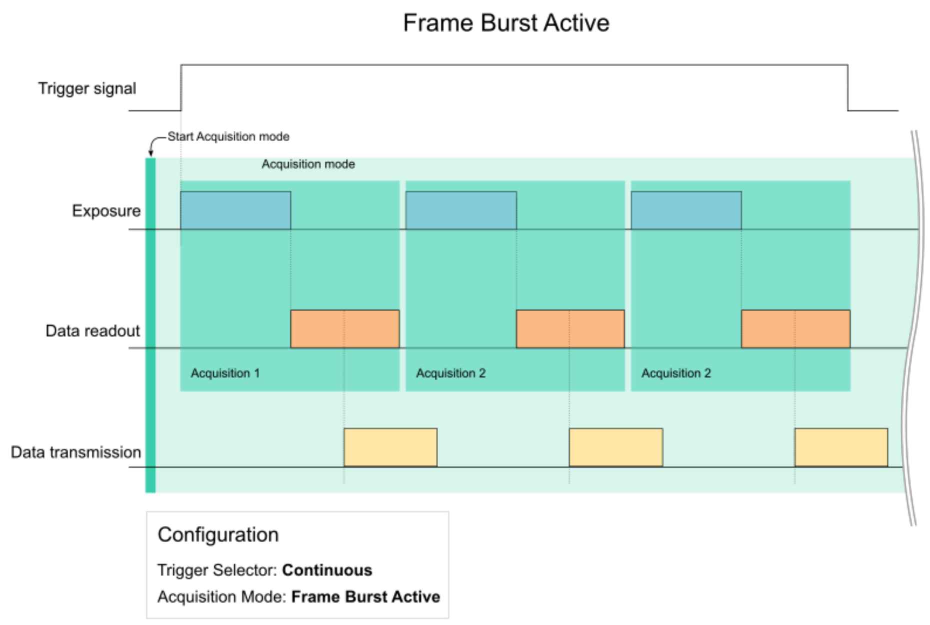


Configuration

Trigger Selector: **Exposure Active**
Acquisition Mode: **Single Frame**

After entering Acquisition mode, the camera detects a trigger and exposes the image sensor according to the duration of the trigger signal.

- **Frame Burst Active**: The camera starts streaming in Free-run mode when the trigger signal is received. Each frame after the first one is acquired at the frequency defined by AcquisitionFrameRate node.



After entering Acquisition mode, the camera detects a trigger and acquires images as long as the trigger signal is active. The timing of the subsequent images is determined by the camera settings similarly to images captured in Free-run mode.

To use any of the trigger modes above, turn it on with the Trigger Mode node. Only one triggers mode can be turned on at once.

Trigger Source

You can trigger via hardware, software triggering, and via action commands. You can also trigger via timer. The trigger source is configured via the Trigger Source node.

Hardware triggering

Hardware triggering gives the greatest level of precision in timing, with the least latency and jitter. The PHX081S camera can be triggered through one of several GPIO pins. On the Trigger Source node, these nodes are indicated by one or more entries prefixed by “Line” (e.g., Line0.) Further configuration of these lines can be done in the Digital IO control.

Note

- Latency: The length of time between when the signal is received and when exposure starts.
- Jitter: The variability in latency.

When working with input voltages greater than 5 V on an opto-isolated GPIO input, set the Line Activation Voltage node to High. (This High value is not available in Phoenix models.)

See more information on configuring hardware triggering, see our related app note.

Software triggering

You can trigger the camera through software commands via the GigE Vision interface. Known as software triggers, these triggers have additional latencies including those from the host OS.

Triggering via action commands

Action commands can also be used to trigger the camera. Triggering via action commands also relies on the GigE Vision interface, but makes use of network-based commands to improve reliability and latency.

When used with PTP (Precision Time Protocol), action commands are an effective way to synchronize multiple cameras in a way that is less dependent on the host computer than in software triggering.

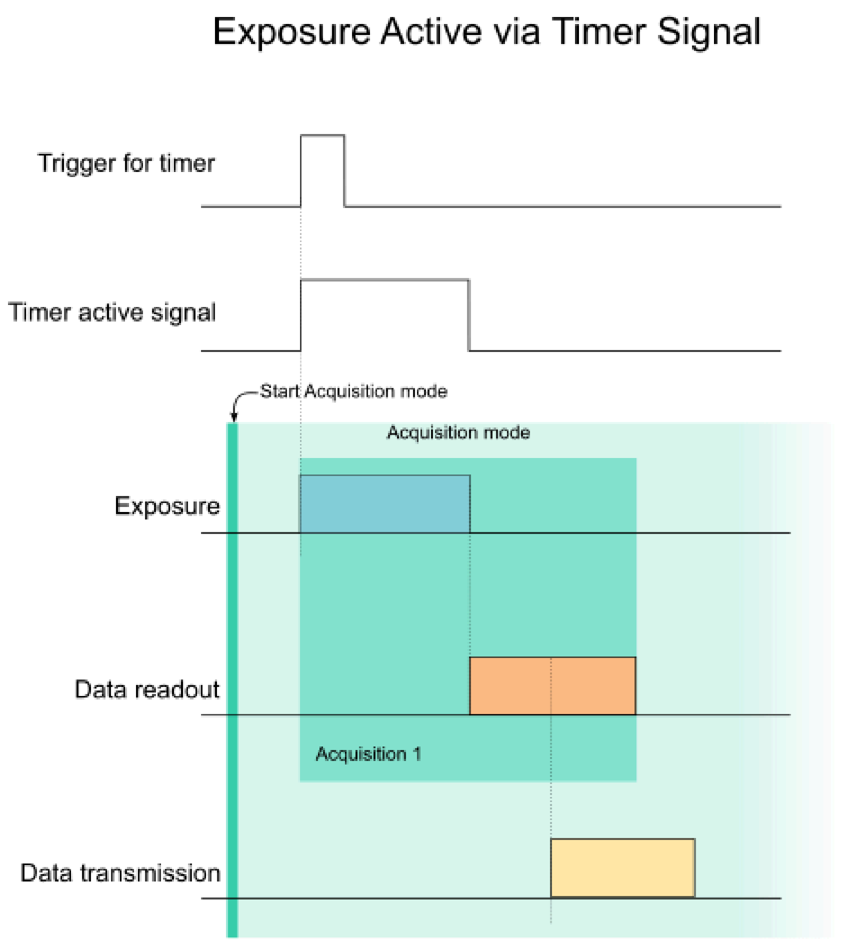
For more information on triggering via action commands, see our related knowledge base article.

Triggering via timer

This camera may be equipped with a trigger that can be activated in response to a timer, which is in turn activated by another trigger.

The following code block demonstrates how to configure a camera to use a timer duration pulse as an Exposure Active trigger source. (Your camera may not have this feature.)

```
// Use a timer duration pulse as an Exposure Active trigger source

TriggerSelector = ExposureActive
TriggerSource = Timer0Active
TriggerActivation = LevelHigh     // Camera exposes while trigger is
high
TriggerMode = On

// Timer setup

SoftwareSignalSelector = SoftwareSignal0
TimerTriggerSource = SoftwareSignal0
TimerTriggerActivation = RisingEdge
TimerDuration = 55000 us          // Chose a random time for the
timer duration
```



An external signal initiates a timer. While the timer is high, the camera exposes the sensor. The configuration is shown in the code block above.

Acquisition Mode

The configurations in the Acquisition Mode node are frequently used with those used with the Trigger Selector. For instance, the Single Frame setting is often used with Frame Start trigger selector; the Continuous setting is often used with Acquisition Start trigger selector.

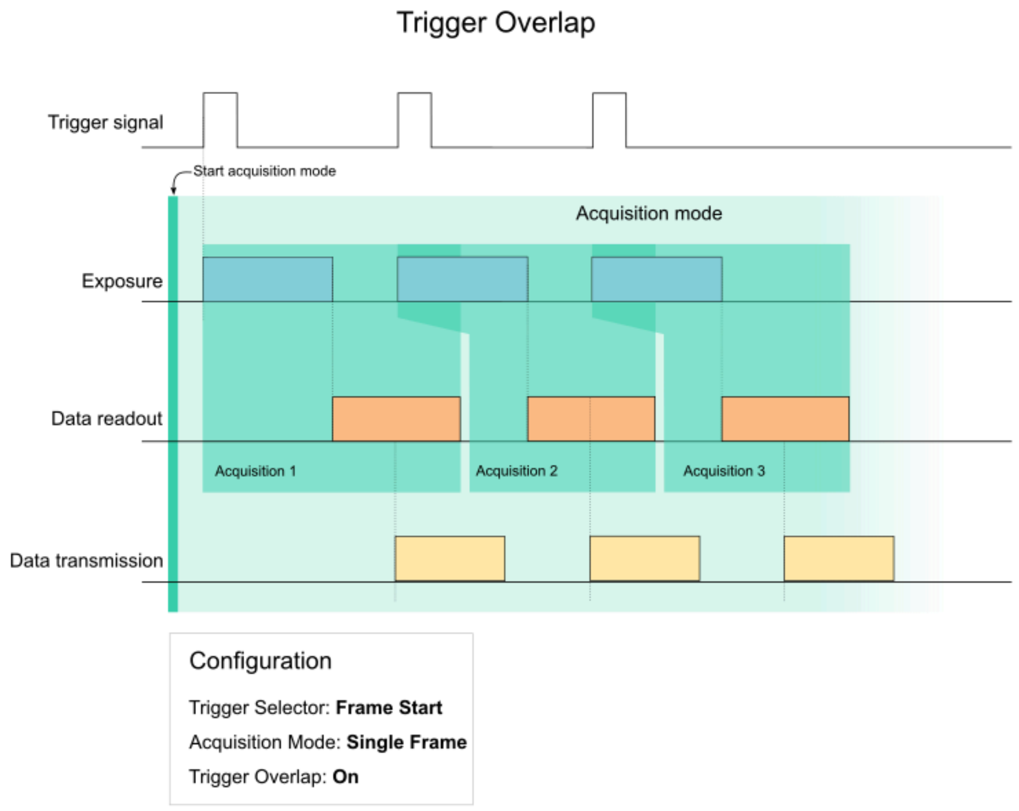
For more details, see the Acquisition Control page.

Trigger Overlap

Trigger overlap improves frames rates by allowing the camera to start exposure while the previous image is being read out. This lets you to achieve almost the same frame rates as you would be able to achieve using Free-run mode.

Note

Trigger Overlap is primarily supported in rolling shutter cameras.



Configuration

Trigger Selector: **Frame Start**

Acquisition Mode: **Single Frame**

Trigger Overlap: **On**

After entering Acquisition mode, the camera detects a trigger and exposes the image sensor. The camera can be triggered during the readout step for the previous image.

Best Practices

- The theoretical maximum trigger rate is indicated by the AcquisitionFrameRate node value.
- Avoid overtriggering the camera, which means triggering a camera faster than its ability to accept triggers. This can cause the frame rate to drop to half of the AcquisitionFrameRate node value.
- Choose an appropriate trigger for your application. Hardware triggers give the most accuracy, while software triggers provide the least

accuracy. Triggering via action commands provides a medium level of accuracy.

Image Format Controls

The PHX081S camera is equipped with the following image format control capabilities.

Region of Interest (ROI)

The Region of Interest feature allows you to specify which region of the sensor is used for image acquisition. This feature allows a custom width and height for image size and a custom X and Y offset for image position. The width and height must be a multiple of the minimum width and height values allowed by the camera.

The following pseudocode demonstrates how to configure the camera to use a region of interest of 200x200 at offset (250,150):

```
// Connect to camera
// Get device node map
OffsetX = 250;
OffsetY = 150;
Width = 200;
Height = 200;
```

Note

If the camera is acquiring images, AcquisitionStop must be called before changing region of interest settings.

Binning

This feature combines columns and/or rows of pixels to reduce the overall image size without changing the image's field of view. This feature may increase the camera's frame rate.

The binning factor indicates how many pixels in the horizontal and vertical axes are combined. For example, when applying 2x2 binning, which is two pixels in the horizontal axis and two pixels in the vertical axis, 4 pixels combine to form one pixel. The resultant pixel values can be summed or averaged.

Binning affects the settings for the image width and height. For example, if you are using a camera with sensor resolution of 2448 x 2048 and apply 2x2 binning, the effective resolution of the resultant image is reduced to 1224 x 1024. You can verify this by checking the Width and Height nodes.

The following pseudocode demonstrates how to configure binning on the camera:

```
// Connect to camera
// Get device node map

BinningSelector = Digital;    // Digital binning is performed by
FPGA; some cameras will support BinningSelector = Sensor
BinningHorizontalMode = Sum;  // Binned horizontal pixels will be
summed (additive binning)
BinningVerticalMode = Sum;    // Binned vertical pixels will be
summed  (additive binning)

BinningHorizontal = 2; // Set Horizontal Binning to a factor of 2
BinningVertical = 2;   // Set Vertical Binning to a factor of 2

// Resulting image is 1/4 of original size
```

Note

When horizontal binning is used, horizontal decimation (if supported) is not available. When vertical binning is used, vertical decimation (if supported) is not available.

Note

If the camera is acquiring images, AcquisitionStop must be called before adjusting binning settings.

Decimation

The PHX081S camera supports decimation, which is a feature that skips columns and/or rows of pixels to reduce the overall image size without changing the image’s field of view. This feature is also known as “subsampling” due to the smaller sample size of pixels the camera transmits. This feature may result in an increase of camera’s frame rate.

Decimation affects the settings for the image width and height. For example, suppose that you are using a camera with sensor resolution of 2448 x 2048. When horizontal and vertical decimation are both set to 2, the effective resolution of the resultant image is reduced to 1224 x 1024. You can verify this by checking the Width and Height nodes.

The following pseudocode demonstrates how to configure decimation on the camera:

```
// Connect to camera
// Get device node map

DecimationHorizontal = 2; //Set Horizontal Decimation to 2
DecimationVertical = 2; // Set Vertical Decimation to 2

// Resulting image is 1/4 of original size
```

Note

When horizontal decimation is used, horizontal binning (if supported) is not available. When vertical decimation is used, vertical binning (if supported) is not available.

Note

If the camera is acquiring images, AcquisitionStop must be called before adjusting decimation settings.

Horizontal and Vertical Flip

This feature allows the camera to flip the image horizontally and vertically. The flip action occurs on the camera before transmitting the image to the host.

The following pseudocode demonstrates how to configure the camera to flip both horizontal and vertical axes:

```
// Connect to camera
// Get device node map
ReverseX = True;
ReverseY = True;
```

Note

If the camera is acquiring images, AcquisitionStop must be called before changing horizontal or vertical flip.

Test Pattern

The camera is able to output one of several FPGA-generated test patterns that you can choose.

Digital IO

The PHX081S's Digital IO controls input and output lines that can be utilized with external circuitry for synchronization with other devices. An example use of an input line is to allow the camera to take an image upon receipt of an internal software signal or an external pulse (rising or falling edge). An example use of an output line is to fire a pulse when the camera starts integration for the duration of the current ExposureTime value.

The Digital IO lines correspond to the PHX081S's GPIO pins. Please consult the GPIO Cable section for more information on the required cable for the GPIO connector and the GPIO Characteristics section for a GPIO pinout diagram.

Configuring an Input Line

When a Digital IO line is set to Input, the line can accept external pulses. To trigger the camera upon receipt of an external pulse, the camera must also have trigger mode enabled.

The following pseudocode demonstrates how to enable trigger mode and setting Line0 as the trigger input source.

```
// Connect to camera
// Get device node map

// Choose Line0 and set it to Input
LineSelector = Line0;
LineMode = Input;

TriggerMode = On;
TriggerSelector = FrameStart;           // Trigger signal starts a frame capture
TriggerSource = Line0;                  // External signal to be expected on Line0
TriggerActivation = FallingEdge;        // Camera will trigger on the falling edge of the input signal

AcquisitionStart();

// Acquire images by sending pulses to Line0
```

It is also possible to set software as the input source. This will enable the camera to trigger upon a software signal. Note this mechanism may not be as accurate as using an external trigger source.

The following pseudocode demonstrates how to enable trigger mode and set a software trigger source.

```
// Connect to camera
// Get device node map

TriggerMode = On;
TriggerSelector = FrameStart;           // Trigger signal starts a frame capture
TriggerSource = Software;               // Software signal will trigger camera

AcquisitionStart();

TriggerSoftware();                      // Execute TriggerSoftware to signal the camera to acquire an image
```

Note

Some Digital IO lines may be Input only (e.g., an opto-isolated input). Consult the GPIO Characteristics section for a GPIO pinout diagram.

Note

When a rolling shutter sensor is triggered, the sensor needs to take and read out an extra frame to reset the sensor. This typically reduces the maximum achievable trigger frequency to about half of the maximum value of the AcquisitionFrameRate node on most rolling shutter cameras.

TriggerOverlap

By default, the PHX081S rejects input pulses until the last triggered image has completed the readout step on the sensor. This may limit the maximum achievable trigger frequency on some cameras when compared to the maximum non-triggered FrameRate.

To address this situation, the PHX081S also supports TriggerOverlap functionality. When TriggerOverlap is enabled, the camera is able to accept an input pulse before the readout step is complete. This allows the camera to be triggered at frequencies closer to the maximum non-triggered FrameRate.

Configuring an Output Line

When a Digital IO line is set to Output, the line can fire pulses.

Note

Opto-isolated outputs will require external circuitry to be properly signaled. Some Digital IO lines may be Output only (e.g., an opto-isolated output). Consult the GPIO Characteristics section for a GPIO pinout diagram.

Turning on GPIO Voltage Output

The PHX081S is capable of supplying external circuits with power through the V_{DD} line. By default this line is turned off. The following pseudocode demonstrates how to enable the GPIO V_{DD} line to output voltage.

```
// Connect to camera
// Get device node map

// Choose Line4 and set it to output external voltage
LineSelector = Line4;
VoltageExternalEnable = True;

// Continue with the rest of the program...
```

Note

Consult the GPIO Characteristics section for a GPIO pinout diagram.

Chunk Data

Chunk data is additional tagged data that can be used to identify individual images. The chunk data is appended after the image data.

For more details on chunk data, see the Device Nodes section.

Extracting the Image CRC checksum with ChunkCRC

When the CRC chunk data property is enabled, the camera tags a cyclic-redundancy check (CRC) checksum that is calculated against the image payload, which is useful for error detection.

The following pseudocode demonstrates how to enable the CRC property in chunk data and extracting the ChunkCRC chunk from acquired images:

```
// Connect to camera
// Get device node map
ChunkModeActive = True;
ChunkSelector   = CRC;
ChunkEnable     = True;

AcquisitionStart();

// Acquire image into buffer
// Read chunk data from received buffer
bufferCRC   = ChunkCRC;        // CRC checksum for image payload
```

Note

If the camera is acquiring images, AcquisitionStop must be called before enabling or disabling chunk data.

Timestamp

The timestamp for the image is available from the chunk data. This timestamp is recorded the moment the camera starts exposure.

Transfer Control

Transfer Control allows the device to accumulate images on the on-camera buffer in a queue. The data stored in the queue, referred to as blocks, can be transmitted to the host application at a later time. The host application will be able to request the device to transmit one or more blocks. By default, this control is disabled on the PHX081S and acquired images are automatically transmitted.

Automatic Transfer Control

When using Automatic Transfer Control mode, the transfer of blocks to the host is controlled by the device's acquisition controls.

In Automatic Transfer Control mode, the TransferOperationMode is read only and set to Continuous. Using Continuous TransferOperationMode is similar to when Transfer Control is not enabled, except the host application can stop data transmission without stopping image acquisition on the device.

The following pseudocode demonstrates how to enable Automatic Transfer Control mode on the camera.

```
// Connect to camera
// Get device node map

// Set desired Pixel Format, Width, Height

// Optional: Read TransferQueueMaxBlockCount to determine the number
of blocks
// that can be stored in the on-camera buffer with the current
device settings

TransferControlMode = Automatic;
TransferQueueMode = FirstInFirstOut;

AcquisitionStart();              // When in Automatic Transfer
Control Mode, the
                                 // AcquisitionStart command
automatically executes the TransferStart command

// Acquire images
// Images will be accumulating into the on-camera buffer
// Blocks will be transmitted to the host system automatically until
TransferPause or AcquisitionStop is executed

// Retrieve images on host system
```

Note

In Automatic Transfer Control mode, the following commands are available:

- **TransferPause** pauses the transfer of blocks without executing TransferStop or AcquisitionStop.
- **TransferResume** resumes the transfer of blocks after TransferPause has been called.
- **TransferAbort** aborts the transfer of blocks.
- **TransferStop** stops the transfer of blocks.

Note

When blocks are not transmitted from the on-camera buffer to the host quickly enough and the number of blocks to be stored in the buffer exceeds TransferQueueMaxBlockCount, the new images will be dropped. Set TransferStatusSelector to **QueueOverflow** and read TransferStatus to determine if a block is lost.

UserControlled Transfer Control

When using UserControlled Transfer Control mode, the transfer of blocks to the host is controlled by the host application. Using MultiBlock TransferOperationMode with UserControlled Transfer Control allows the host application to specify when to transmit data from the device.

The following pseudocode demonstrates how to enable UserControlled Transfer Control mode with MultiBlock TransferOperationMode on the camera. It also demonstrates transferring 2 blocks from the on-camera buffer for each transmission operation.

```
// Connect to camera
// Get device node map

// Set desired PixelFormat, Width, Height

// Optional: Read TransferQueueMaxBlockCount to determine the number of blocks
// that can be stored in the on-camera buffer with the current device settings

TransferControlMode = UserControlled;
TransferQueueMode = FirstInFirstOut;

TransferOperationMode = MultiBlock;    // Transmit the number of blocks in TransferBlockCount
TransferBlockCount = 2;                // This should be less than or equal to TransferQueueMaxBlockCount

AcquisitionStart();

// Acquire images
// Images will be accumulating into the on-camera buffer

// Read TransferQueueCurrentBlockCount
If (TransferQueueCurrentBlockCount > 0)
{
    // Transmit blocks to host system
    TransferStart();

    // Retrieve block 1 on host system
    // Retrieve block 2 on host system

    TransferStop();
}
```

Note

In UserControlled Transfer Control mode, the following commands are available:

- **TransferPause** to pause transfer of blocks without executing TransferStop or AcquisitionStop.
- **TransferResume** to resume transfer of blocks after TransferPause has been called.
- **TransferAbort** to abort transfer of blocks.
- **TransferStop** to stop transfer of blocks.

Note

When blocks are not transmitted from the on-camera buffer to the host quickly enough and the amount of blocks to be stored in the buffer exceeds TransferQueueMaxBlockCount, the new images are dropped. Set TransferStatusSelector to **QueueOverflow** and read TransferStatus to determine if a block is lost.

Note

To turn off the Transfer Control mechanism, set the TransferControlMode to Basic.

Note

If the camera is acquiring images, AcquisitionStop must be executed before changing Transfer Control properties.

Event Control

Events are notifications generated by the camera to inform the host application about internal updates. Event Control is a mechanism used to synchronize the camera with host application using these events.

Note

The events described in this section are separate from EVS events, which are small packets of information that are generated when there is a contrast change in one or more pixels in the vision sensor of event-based cameras.

EventSelector can be used to select the event for turning notifications on or off. By default, EventNotification for all the events are set to Off except for Test events. This is because EventNotification for Test events is always set to On and cannot be changed.

The following pseudocode demonstrates how to enable Event Control on the camera.

```
// Connect to camera
// Get device node map

// Initialize host software to receive events

// Register event callback within host software

// Wait on event

// Encounter event from camera

// De-register event callback within host software

// De-initialize host software to receive events
```

Events available on PHX081S are as follows:

Exposure Start

The Exposure Start event occurs when the camera starts exposing the sensor to capture a frame. When this event occurs, the following data is sent by the camera in an Exposure Start event:

- **EventExposureStart:** Unique identifier of the Exposure Start type of Event.
- **EventExposureStartTimestamp:** Unique timestamp of the Exposure Start Event.
- **EventExposureStartFrameID:** Unique Frame ID related to Exposure Start Event.

Exposure End

The Exposure End event occurs when the camera has finished exposing the sensor. The following data is sent by the camera in an Exposure End event:

- **EventExposureEnd:** Unique identifier of the Exposure End type of Event.
- **EventExposureEndTimestamp:** Unique timestamp of the Exposure End Event.
- **EventExposureEndFrameID:** Unique Frame ID related to Exposure End Event.

Test

The Test event occurs when TestEventGenerate node is executed. Test event is mainly used to confirm that camera is generating events. The following data is sent by the camera in a Test event:

- **EventTest:** Unique identifier of the Test type of Event.
- **EventTestTimestamp:** Unique timestamp of the Test Event.

Counter and Timer Control

Counter

Counters can be used to keep a running total of an internal event. The Counter value can be reset, read, or written at any time.

The following nodes are available to configure the Counter increment:

- **CounterSelector**: The Counter control to be used
- **CounterEventSource**: The event source that increments the Counter
- **CounterEventActiviation**: The activation mode used for **CounterEventSource** (e.g., RisingEdge)
- **CounterTriggerSource**: The source that starts the Counter

The following nodes are available to configure the Counter reset:

- **CounterResetSource**: The signal that resets the Counter
- **CounterResetActivation**: The signal used to activate **CounterResetSource**
- **CounterReset**: The command resets CounterValue and sets the last value to **CounterValueAtReset**

The following nodes are available to read or control the Counter:

- **CounterValue**: The current value of the Counter
- **CounterValueAtReset**: The value of the Counter before it was reset
- **CounterDuration**: The value to count before setting **CounterStatus = CounterCompleted**

- **CounterStatus**: The status of the Counter

The following **CounterStatus** modes are available:

- **CounterIdle**: Counter control is not enabled
- **CounterTriggerWait**: Counter is waiting to be activated
- **CounterActive**: Counter is counting until the value specified in **CounterDuration**
- **CounterCompleted**: Counter has incremented to value set in **CounterDuration**
- **CounterOverflow**: **CounterValue** has reached its maximum possible value

Note

When **CounterValue = CounterDuration**, the counter stops counting until a **CounterReset** or a new **CounterTriggerSource** occurs.

Note

Changing **CounterDuration** issues a **CounterReset** to **CounterValue**.

Use Counter to Count Rising Edges

The following pseudocode demonstrates how to use a Rising Edge counter event to track the number of rising edge signals received on Line 0. A second counter is enabled to track the number of Exposure Start events that occurred to indicate the number of images captured.

```
// Connect to camera
// Get device node map

// Use Counter0 to count rising edges received on Line0
CounterSelector = Counter0;
CounterEventSource = Line0;
CounterEventActivation = RisingEdge;
CounterTriggerSource = AcquisitionStart;     // Start this counter
at the camera's AcquisitionStart
CounterDuration = 1;
CounterValue = 1;                            // Start at
CounterValue = CounterDuration (e.g., increment the counter until we
reset it)

// Use Counter1 to count ExposureStart events occurring on camera
CounterSelector = Counter1;
CounterEventSource = ExposureStart;
CounterEventActivation = RisingEdge;
CounterTriggerSource = AcquisitionStart;     // Start this counter
at the camera's AcquisitionStart
CounterDuration = 1;
CounterValue = 1;                            // Start at
CounterValue = CounterDuration (e.g., increment the counter until we
reset it)

// Enable Rising Edge trigger on Line0
TriggerSelector = FrameStart;
TriggerSource = Line0;
TriggerActivation = RisingEdge;
TriggerMode = On;

AcquisitionStart();

// Acquire images by sending pulses to Line0

AcquisitionStop();

// To read rising edge pulses received:
// CounterSelector = Counter0
// Read CounterValue

// To read images captured:
// CounterSelector = Counter1
// Read CounterValue

 // CounterValue for Counter0 should equal CounterValue for
Counter1. If not we missed a trigger signal.
```

Timer

Timers can be used to measure the duration of internal or external signals.

The following nodes are available to configure the Timer:

- **TimerSelector**: The Timer control to be used
- **TimerTriggerSource**: The source that will trigger the Timer
- **TimerTriggerActivation**: The activation mode used for **TimerTriggerSource** (e.g., RisingEdge)
- **TimerDuration**: The duration of the Timer pulse in microseconds
- **TimerDelay**: The duration of delay to apply before starting the Timer upon receipt of a TimerTriggerActivation
- **TimerReset**: Resets the Timer control
- **TimerValue**: Reads or writes the current value of the Timer
- **TimerStatus**: The status of the Timer

The following TimerStatus modes are available:

- **TimerIdle**: Timer control is not enabled
- **TimerTriggerWait**: Timer is waiting to be activated
- **TimerActive**: Timer is counting for the specified duration in **TimerDuration**
- **TimerCompleted**: Timer has reached the duration specified in **TimerDuration**

Note

When **TimerStatus = TimerCompleted**, the Timer signal becomes active low until a **TimerReset** or a new **TimerTriggerSource** occurs.

Use Timer to Delay a Line Output Pulse

The following pseudocode demonstrates how to use a Timer to delay a Line output pulse relative to the ExposureStart event of the image.

```
// Connect to camera
// Get device node map

// Set up Line2 for Timer pulse output:
LineSelector = Line2;
LineMode = Output;
LineSource = Timer0Active;                // The Line output will be
the duration of the Timer

// Use Timer0 to configure your Line2 pulse:
TimerSelector = Timer0;
TimerTriggerSource = ExposureStart;
TimerTriggerActivation = RisingEdge;
TimerDelay = 2000;                             // The Line output
delay will be 2000 microseconds from the ExposureStart event
TimerDuration = 8000;                          // The Line output
duration will be 8000 microseconds

AcquisitionStart();
// Acquire images
// Monitor Line2 output
```

Use Timer to Output a Custom Pulse Width

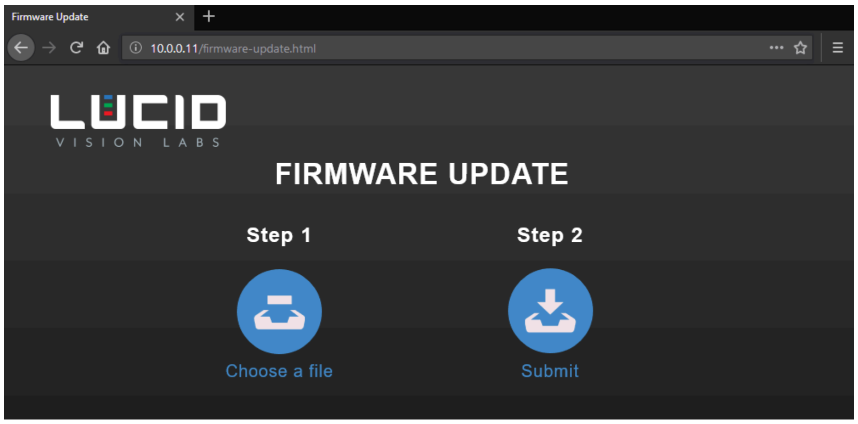
The following pseudocode demonstrates how to use a Timer output a pulse of a custom width. This can be used for a single pulse width or pulse width modulation.

```
// Connect to camera
// Get device node map

// Set up Line2 for Timer pulse output:
LineSelector = Line2;
LineMode = Output;
LineSource = Timer0Active;                // The Line output will be
the duration of the Timer

// Use Timer0 to configure your Line2 pulse:
TimerSelector = Timer0;
TimerTriggerSource = SoftwareSignal0;
TimerTriggerActivation = RisingEdge;
TimerDelay = 0;
TimerDuration = 2000;                          // The Line output
duration will be 2000 microseconds

// Use SoftwareSignal0 to trigger the Timer pulse
SoftwareSignalSelector = SoftwareSignal0;
SoftwareSignalPulse();
```

Firmware Update

Use the camera's Firmware Update page to update your camera's firmware. Updating your camera's firmware is easy with our firmware (fwa) file. The firmware fwa files for each camera are located on our Downloads page.

Every Lucid camera comes with an onboard firmware update page that you can directly access in a web browser. If you already know your camera's IP address, you can access it at http://{your-camera-ip-address}/firmware-update.html.



Updating your Camera Firmware

1. Navigate to your camera's firmware update page in your web browser (http://{your-camera-ip-address}/firmware-update.html).
2. Click **Choose a file** on the Firmware Update page and select your fwa file.
3. Click **Submit** on the Firmware Update page. This will start the firmware update process.

The Firmware Update page will show the progress of the update. During the update, the camera will not be accessible for control or image capture. The update may take a few minutes to complete. Please refrain from turning off power to the camera during the update.

When the firmware update is complete, the camera will reboot. Once the update is finished, you can close the Firmware Update page and access the camera again.