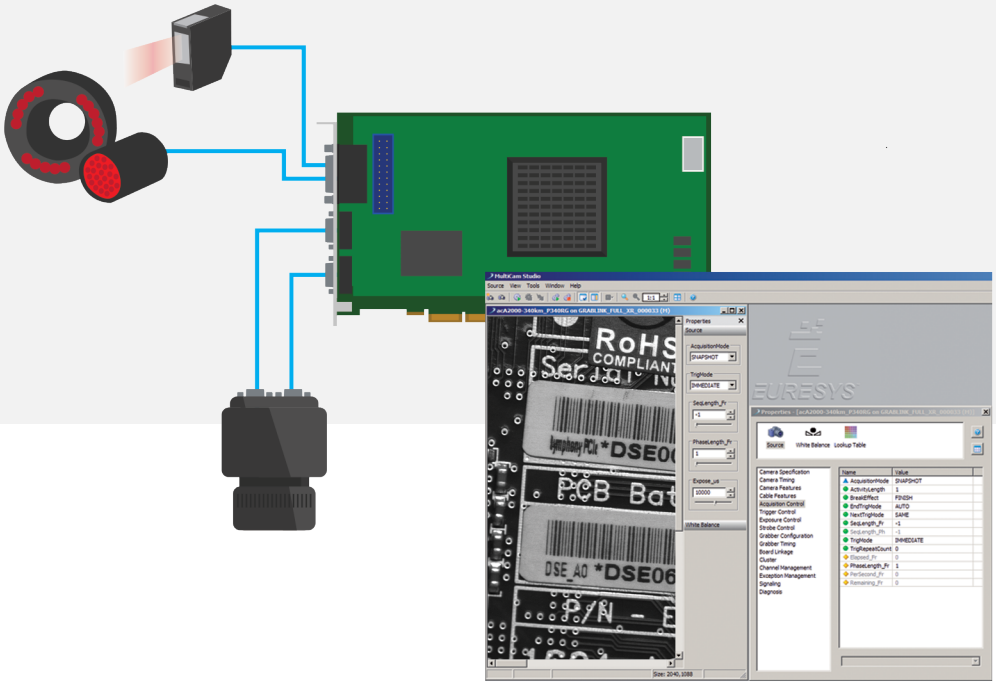
EURESYS™
Excellence in vision

USER GUIDE

MultiCam

MultiCam User Guide



Terms of Use

EURESYS s.a. shall retain all property rights, title and interest of the documentation of the hardware and the software, and of the trademarks of EURESYS s.a.

All the names of companies and products mentioned in the documentation may be the trademarks of their respective owners.

The licensing, use, leasing, loaning, translation, reproduction, copying or modification of the hardware or the software, brands or documentation of EURESYS s.a. contained in this book, is not allowed without prior notice.

EURESYS s.a. may modify the product specification or change the information given in this documentation at any time, at its discretion, and without prior notice.

EURESYS s.a. shall not be liable for any loss of or damage to revenues, profits, goodwill, data, information systems or other special, incidental, indirect, consequential or punitive damages of any kind arising in connection with the use of the hardware or the software of EURESYS s.a. or resulting of omissions or errors in this documentation.

This documentation is provided with MultiCam 6.17.0 (doc build 4033).

Contents

PART I
GENERAL TOPICS

1. Cameras Topics

1.1. Common to Area-Scan and Line-Scan

What is Exposure?

In order to build an electrical signal representing the light intensity at a given point of a scene, the CCD sensor photosite should be exposed to the light during a certain amount of time. This amount of time is known as the exposure time, also called the integration time.

The quantity of electrical charge built during the exposure process is proportional to the incoming light intensity and to the exposure time.

In the CCD sensors that can be used for industrial imaging, all the pixels experience simultaneously the exposure condition. This means that the exposure starting and stopping instants are common to all photosites.

The accurate control of the exposure time is a feature that applies to all line-scan sensors, and to most area-scan sensors. The area-scan CCD sensor type that performs best in this respect is the interline transfer CCD sensor.

Analog vs. Digital

An analog camera delivers the signal representing the observed image in the form of an analog signal, named the video signal. The analog signal incorporates several features aimed at providing timing information for the frame grabber to synchronize on it.

Domino frame grabbers are able to interface to industrial analog cameras, while Picolo frame grabbers are able to interface to standard analog cameras.

A digital camera delivers the signal representing the observed image in the form of a digital signal. The luminance signal can be coded on 8 to 12 bits, and transmitted to the frame grabber through a digital data link.

Grablink frame grabbers are able to interface to industrial digital cameras based on the Camera Link standard.

Fundamental Synchronization Modes

Fundamental modes to synchronize a camera and a frame grabber

Synchronization mode	Description
ANALOG	The only timing information available from the camera is the composite video signal.
DIGITAL	The camera delivers the timing information through a set of digital lines.
MASTER	The camera is due to receive its timing information from the part of the board. The board is the timing master of the camera.

The three fundamental synchronization modes are applicable to the analog cameras attached to a board of the *Domino* series. For a detailed description of the synchronization mode on Domino series, refer to the **Hmode**, **Hreference**, **Vmode**, and **Vreference** parameters.

The digital cameras on *Grablink* series use exclusively the DIGITAL fundamental synchronization mode.

1.2. Area-Scan Camera Topics

Area-scan Exposure and Readout

Let us consider the case of an area-scan camera equipped with a CCD sensor of the interline transfer or frame transfer type. It is useful to consider two successive conditions: frame exposure, followed by frame readout.

Additionally, the area-scan CCD sensor can be temporarily set in a reset condition. In this condition, the light has no effect on the sensitive area of the sensor. Any electrical charge that could be contained in the photosites is cleared.

At some instant, the sensor leaves the reset condition to enter into the exposure condition (also known as integration). During this time, every photosite builds an electrical charge growing at a rate proportional to the light intensity it receives. The longer the exposure time, the larger the electrical charge.

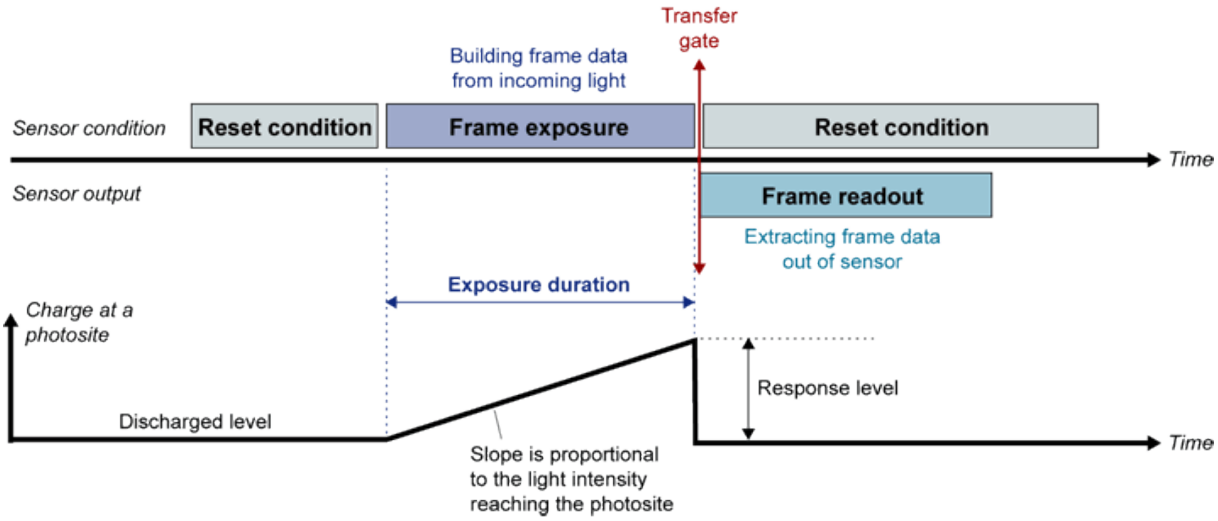
Consequently, increasing the exposure time is a mean to increase the light sensitivity of the camera.

The end of the exposure time is marked by a special event usually referred to as the "transfer gate". At this instant, the individual electrical charges built by the photosites of the entire frame are set aside and made ready for transport towards the CCD sensor output.

Simultaneously, the photosites get emptied of any electrical charge. This sets the photosites in the same state as the reset feature does.

The transfer gate duration is short, virtually instantaneous for interline transfer type area-scan CCD sensors.

After the transfer gate, the CCD sensor enters the readout period. This takes a fixed amount of time to extract the individual electrical charges of the entire frame set aside at the transfer gate instant. The charges are converted to a voltage, serially conveyed outside the CCD sensor and made available at the camera output as a video signal.



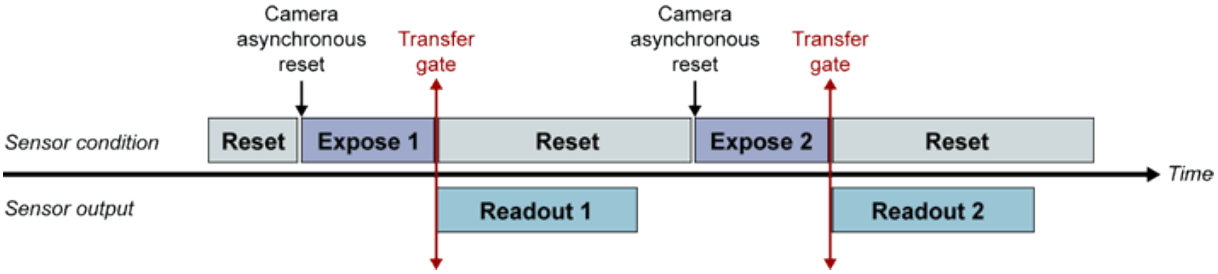
Operational sequence

Exposure Control and Asynchronous Reset

Many industrial grade cameras allow for asynchronous reset while enabling exposure control. This means that, at some asynchronous instant, the camera is forced to enter into an "Expose-Readout" sequence, called an acquisition phase.

The image seen by the camera during the exposure condition is sent out of the camera during the readout condition. It is often advisable to keep the exposure duration short to remove any blurring effect if the observed scene is in movement.

The figure shows two successive camera cycles. Quite often, it is not possible to overlap the exposure condition of a given cycle with the readout condition of the previous cycle.



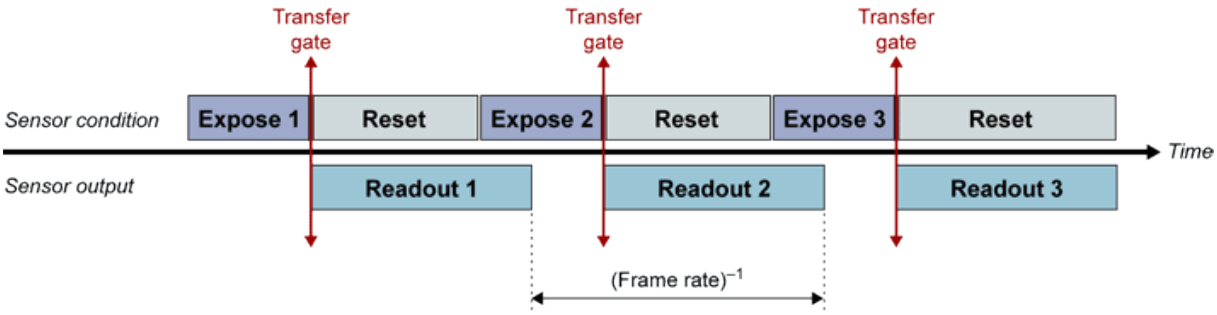
Two successive camera cycles

Exposure Control and Synchronous Scanning

Non-industrial grade cameras do not support the asynchronous reset feature. This means that the frame readout condition repeats itself periodically at some permanent frequency called the "frame rate". This is called the synchronous scanning mode.

This does not preclude some exposure control capability. Most synchronous cameras make possible to overlap the exposure and readout conditions. This feature is often referred to as "electronic shutter".

In the following figure, the exposure condition before a given transfer gate is responsible for the building of a video image which is extracted out of the camera during the readout condition that follows this transfer gate.

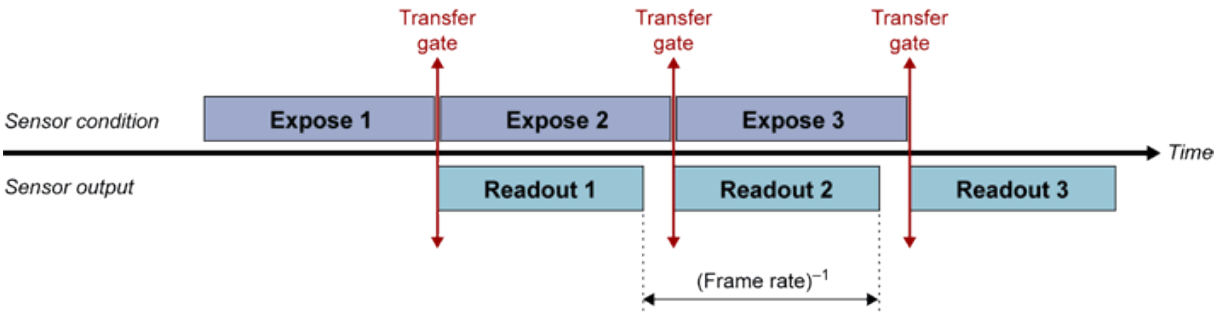


Exposure control and synchronous scanning

The depicted cycling repeats indefinitely.

Permanent Exposure

The permanent exposure feature allows for the greatest sensitivity of a synchronously scanned area-scan camera. It is made possible because the transfer gate removes all charge out of the sensor pixels, establishing the right situation for a new exposure condition to take place.



Permanent exposure

The cycling depicted in the above figure repeats indefinitely.

1.3. Line-Scan Camera Topics

Line-Scan Exposure and Readout

Let us consider the case of a camera equipped with a line-scan CCD sensor. It is useful to consider two successive conditions: line exposure, followed by line readout.

Additionally, the line-scan CCD sensor can be temporarily set in a reset condition. In this condition, the light has no effect on the sensitive area of the sensor. Any electrical charge that could be contained in the photosites is cleared.

At some instant, the sensor leaves the reset condition to enter into the exposure condition (also known as integration). During this time, every photosite builds an electrical charge growing at a rate proportional to the light intensity it receives. The longer the exposure time, the larger the electrical charge.

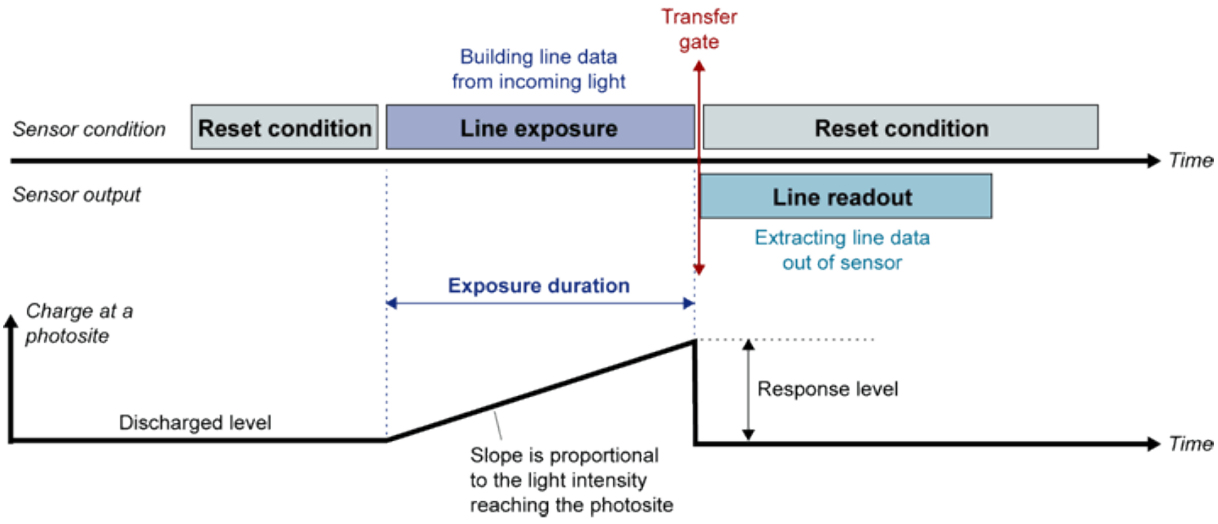
Consequently, increasing the exposure time is a mean to increase the light sensitivity of the camera.

The end of the exposure time is marked by a special event usually referred to as the "transfer gate". At this instant, the individual electrical charges built by the photosites of the entire line are set aside and made ready for transport towards the CCD sensor output.

Simultaneously, the photosites are emptied of any electrical charge. This sets the photosites in the same state as the reset feature does.

The transfer gate duration is short, virtually instantaneous.

After the transfer gate, the CCD sensor enters the readout period. This takes a fixed amount of time to extract the individual electrical charges of the entire line set aside at the transfer gate instant. The charges are converted to a voltage, serially conveyed outside the CCD sensor and made available at the camera output as a video signal.



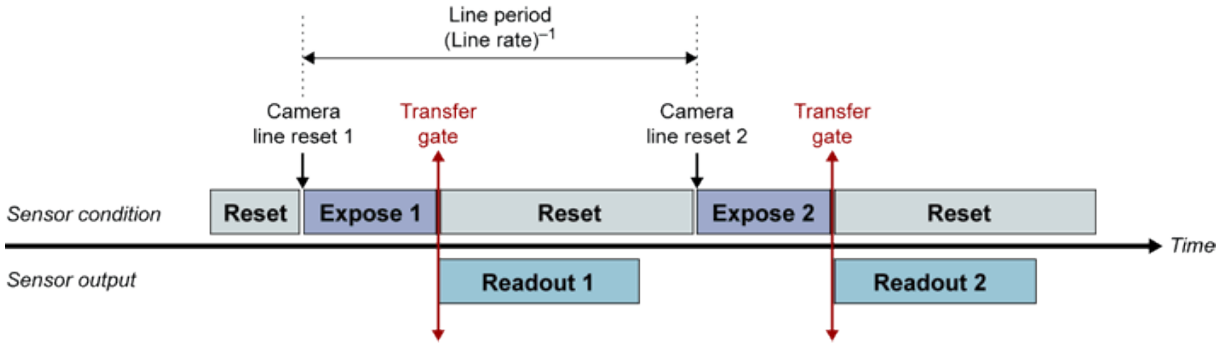
Operational sequence

Controlled Exposure

In the controlled exposure situation, the line-scan CCD sensor is brought into the reset condition before entering the exposure condition. It is often said that an "electronic shutter" is used.

The line-scan camera hosting the sensor experiences a periodical cycle paced by successive instants we will refer to as "line reset". The line reset pulses occur at the line frequency. Each line reset causes the following sequence:

1. Exposure condition of some known duration.
2. Transfer gate event.
3. Readout process.
4. Return to reset condition.



Operational sequence

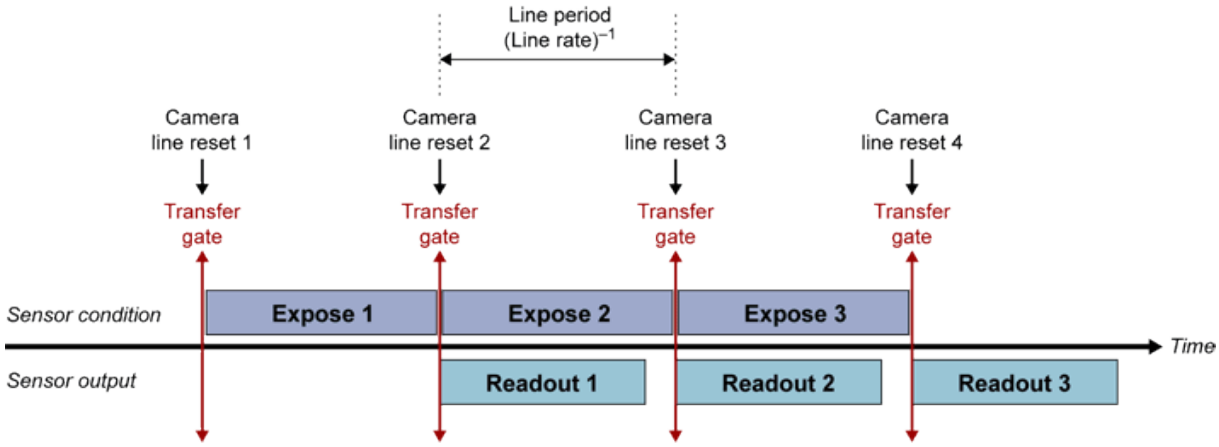
It can be seen that the exposure duration is not equal to the line period. It can be made independent of the line rate.

Permanent Exposure

In the permanent exposure situation, the line-scan CCD sensor is never brought into the reset condition.

The line-scan camera hosting the sensor experiences a periodical cycle paced by successive instants we will refer to as "line reset". The line reset pulses occur at the line frequency. Each line reset causes a transfer gate event followed by a readout process.

The line reset pulses are issued by the frame grabber in a programmable way.



Permanent exposure operational sequence

Between two successive line reset pulses, the sensor integrates the light. The sensitivity of the camera depends on the line period separating the pulses. The lower the line frequency, the higher the response to a given scene.

Camera Operating Modes

There are six operating modes, each of them being identified by a symbol as follows:

First letter of the symbol:

- **R** (Reset) : the line-scanning process starts upon receiving a signal from the frame grabber.
- **S** (Synchronous) : the line-scanning process is free-running.

Second letter of the symbol:

- **G** (Grabber) : the exposure control is exercised from the frame grabber.
- **C** (Camera) : the exposure control is exercised from within the camera.
- **P** (Permanent) : there is no exposure control, i.e. the exposure is permanent.

Note: *Some cameras may be configured to behave according to more than one mode from the list before. However, in a given application, only one mode is in use.*

Controlled cameras

Symbol	Meaning
RG	Grabber-controlled line-scanning, grabber-controlled exposure, single signal
	Exposure duration defined as the active duration of a pulse over a single line issued by the frame grabber.

Symbol	Meaning
RG2	Grabber-controlled line-scanning, grabber-controlled exposure, dual signal
	Exposure duration defined as the active duration of a pulse over a dual line issued by the frame grabber.
RC	Grabber-controlled line-scanning, camera-controlled exposure
	Exposure duration set through camera switches or serial control. Line-scanning is triggered by a pulse over a line issued by the frame grabber.
RP	Grabber-controlled line-scanning, permanent exposure
	No exposure control capability, resulting in permanent exposure. Line-scanning is triggered by a pulse over a line issued by the frame grabber.

Free-running cameras

Symbol	Meaning
SC	Free-running, camera-controlled exposure
	Exposure duration set through camera switches or serial control. Line-scanning is free-running.
SP	Free-running, permanent exposure
	No exposure control capability, resulting in permanent exposure. Line-scanning is free-running.

2. Frame Grabber Topics

2.1. Common to Area-Scan and Line-Scan

What is a Frame Grabber?

A frame grabber is the usual name given to an electronic board to be installed into a computer and aimed at interfacing a video camera to this computer.

This name is quite adequate for area-scan cameras, which generate image frames. When line-scan cameras are involved, lines instead of frames are generated. Nevertheless, the interface board is still commonly called a frame grabber.

The frame grabber can be categorized in several kinds, according to the internal storage structure:

- Frame buffer-based frame grabbers
- FIFO-based frame grabbers

Another kind of frame grabber is the frame processor, which includes some image processing means.

What is a Grabber?

A grabber is a set of hardware resources owned by a frame grabber.

Many Euresys frame grabbers are able to incorporate several grabbers operating simultaneously. This effectively results into several independent frame grabbers within a single board.

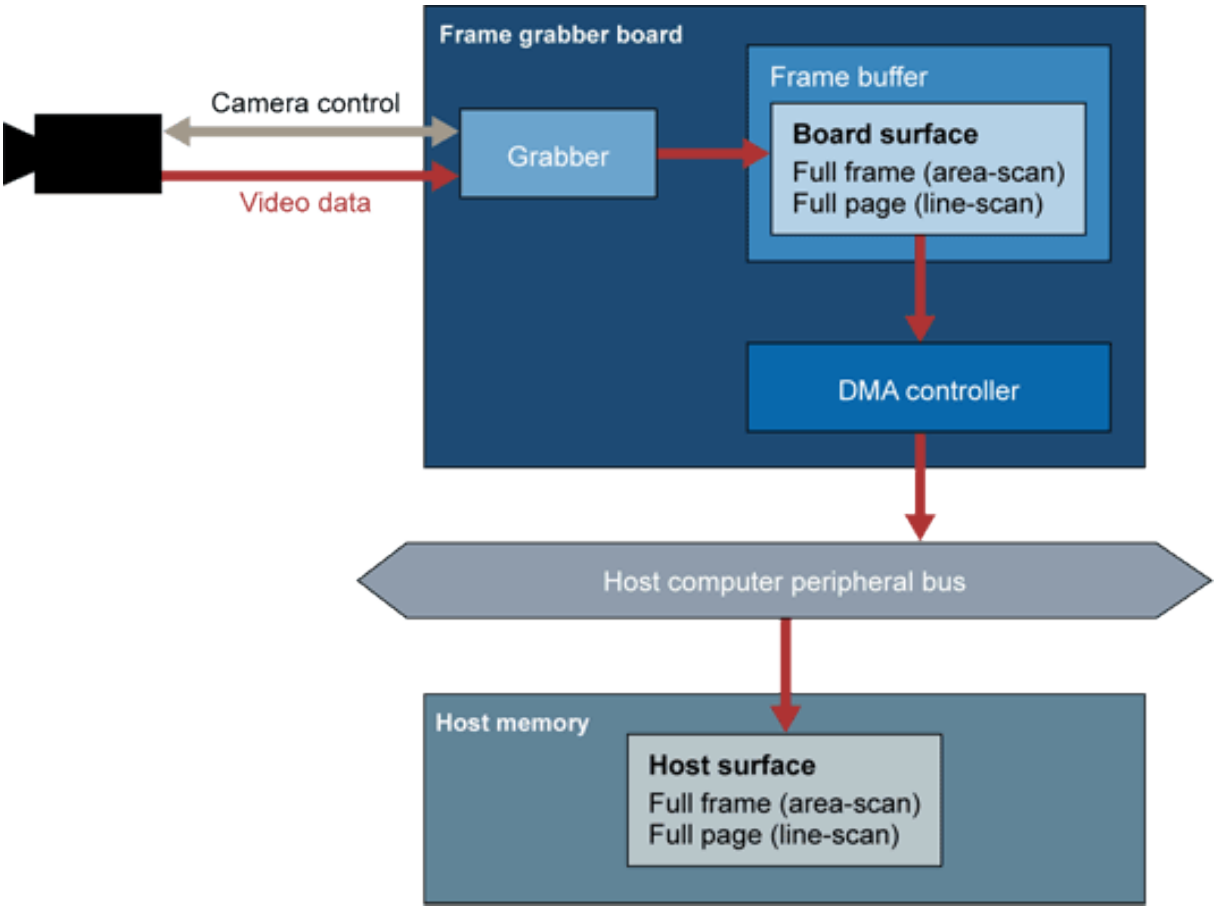
When several grabbers cannot be operated simultaneously, it is still possible to handle several cameras. The grabber (or set of grabbers) is used in a time-multiplexed fashion. This is called "grabber switching". The grabber is successively allocated to each camera to take control over it and acquire from it a video frame or page.

A frame buffer-based grabber will deposit frames or pages into the on-board frame buffer, and subsequently into the host memory.

A FIFO-based grabber will deposit frames or pages directly into the host memory.

See specific definitions of a grabber in the documentation topics related to Domino and Grablink series.

Frame Buffer-Based Frame Grabber



Frame buffer-based frame grabber

The **grabber** is a set of hardware resources taking in charge all timing and control task required by the camera, and conditioning the video data (analog or digital) provided by the camera.

The **frame buffer** is an internal storage area large enough to hold a full frame image issued by the camera. In case of line-scan operation, the frame is actually made of a set of contiguous lines, and this set is called a page.

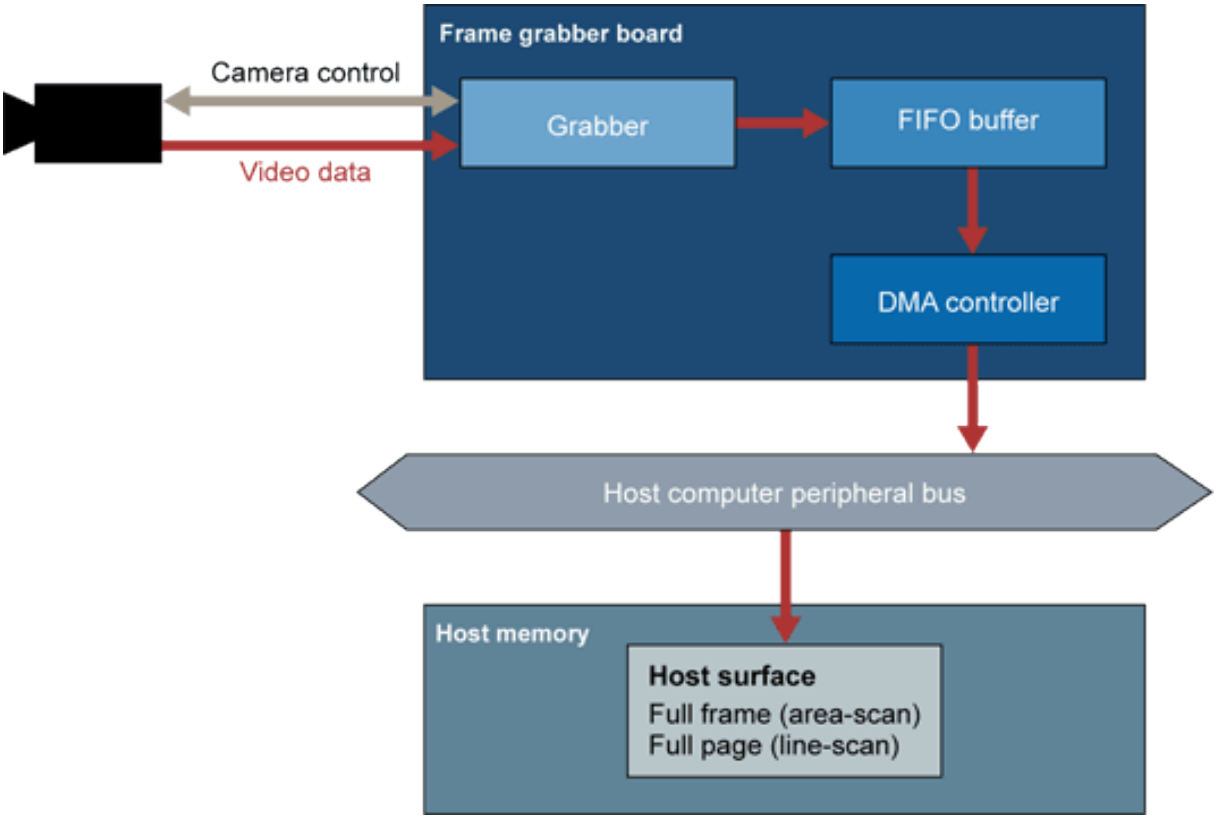
The **DMA controller** is a device able to transfer the stored image from the frame buffer into the host computer memory in a DMA (Direct Memory Access) fashion. This transfer does not require any host CPU intervention.

Usually, the host computer is a PC and the peripheral bus is a PCI bus.

The destination area in the on-board or host memory is called a "surface". A surface is a memory container able to store a bi-dimensional image corresponding to a frame (area-scan) or a page (line-scan).

In the frame buffer-based frame grabber, the camera-to-buffer transfer can be decoupled from the buffer-to-surface transfer. The goal of the grabber is to feed the frame buffer. Emptying the frame buffer to the PC can optionally obey to specific rules, such as area of interest transfers.

FIFO-Based Frame Grabber



FIFO-based frame grabber

The **grabber** is a set of hardware resources taking in charge all timing and control task required by the camera, and conditioning the video data (analog or digital) provided by the camera.

The **FIFO buffer** is an internal storage area able to hold a part of the image issued by the camera, usually a few video lines. FIFO means "First In, First Out".

The **DMA controller** is a device able to transfer the buffered data from the FIFO to the host computer memory in a DMA (Direct Memory Access) fashion. This transfer does not require any host CPU intervention.

Usually, the host computer is a PC and the peripheral bus is a PCI bus.

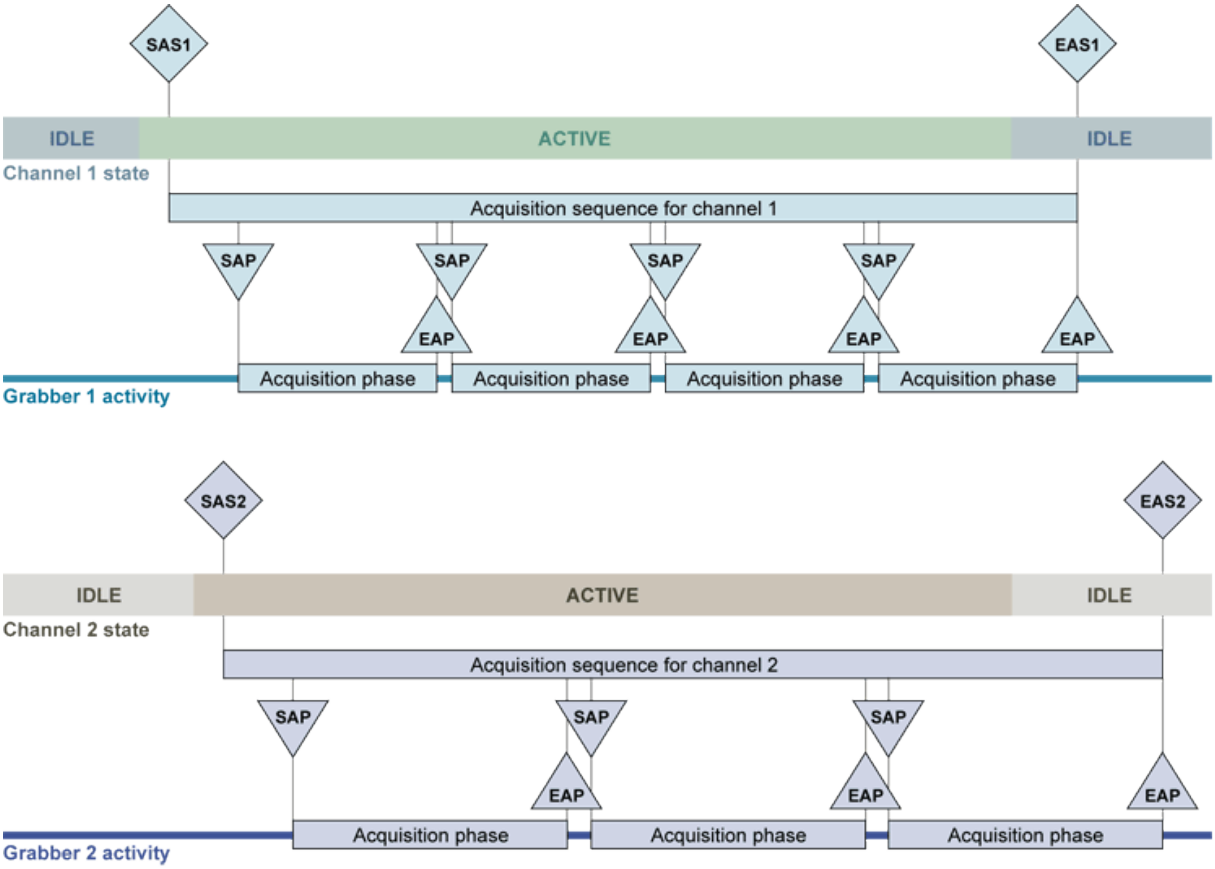
The destination area in the memory is called a "surface". A surface is a memory container able to store a bi-dimensional image corresponding to a frame (area-scan) or a page (line-scan).

In the FIFO-based frame grabber, the camera-to-buffer transfer cannot be decoupled from the buffer-to-surface transfer. The goal of the grabber is to directly feed the host memory.

Concurrent Acquisition Modes

The names **DuoCam** and **TrioCam** are given to the concurrent acquisition modes.

All hardware resources involved in the image acquisition process exist in two instances within the frame grabber. This allows for full independent and simultaneous operation of two acquisition channels.

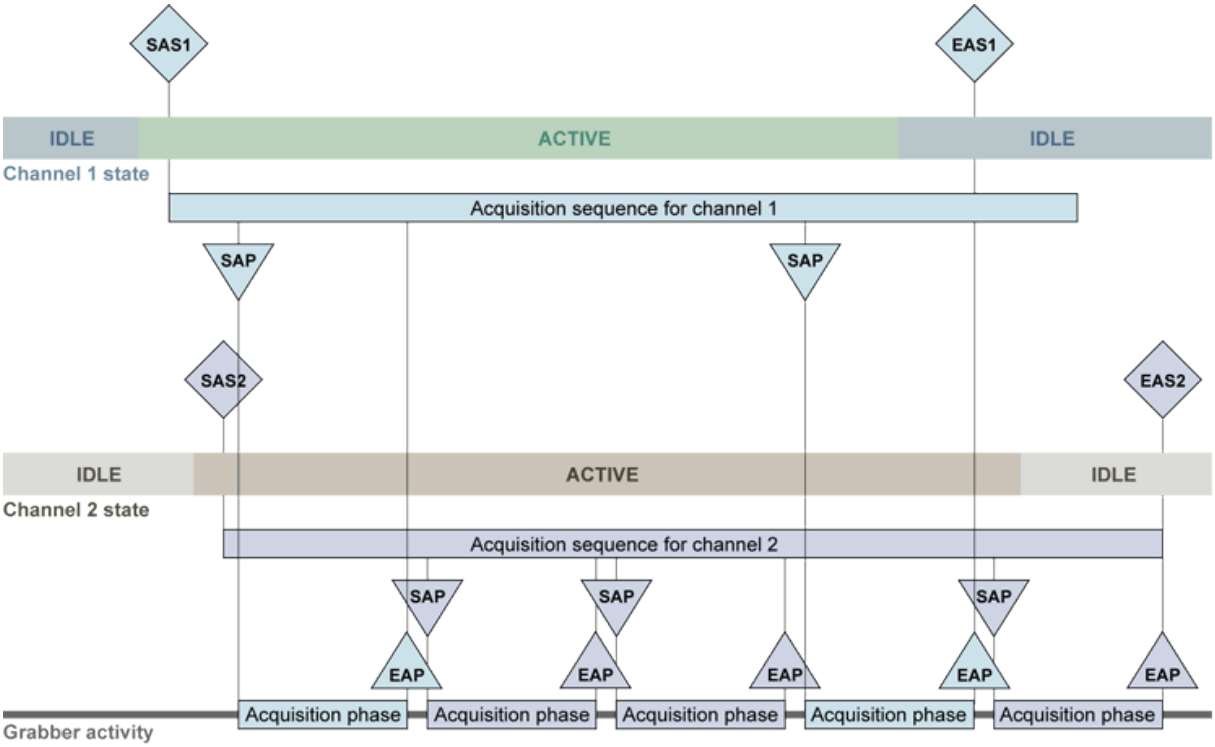


DuoCam acquisition mode

The TrioCam mode is similar to this, with three acquisition channels instead of two.

Switched Acquisition Mode

One frame at a time is extracted out of an individual camera, and all cameras are sequentially scanned according to controlled rules.



Switched acquisition mode for two cameras and a single grabber

The switched acquisition mode is in no manner limited to two cameras.

2.2. Area-Scan Frame Grabber Topics

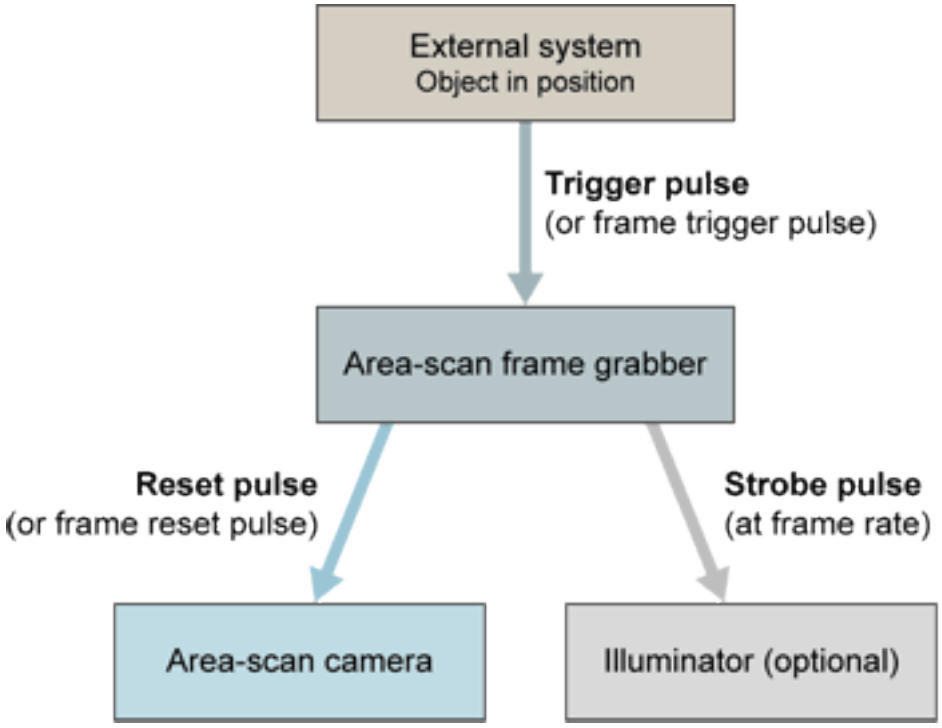
Trigger, Reset and Strobe in Area-Scan

In order to ensure the terminology consistency throughout the MultiCam documentation, the following terminology conventions have been taken.

A **trigger pulse**, or frame trigger pulse, is an electrical signal sent by the external system to instruct a frame grabber to take control over the camera, including exposure control, and to perform a frame acquisition. This is usually used when an asynchronous capture of a moving object is involved. The trigger pulse is issued by a position sensor indicating when the observed object is adequately located in the field of view.

A **reset pulse**, or frame reset pulse, is an electrical signal sent by the frame grabber to instruct an area-scan camera to start its frame acquisition cycle.

A **strobe pulse** is an electrical signal sent by the frame grabber to control an external illumination device. The strobe pulse timing is controlled to adequately fit into the frame timing of the area-scan camera.

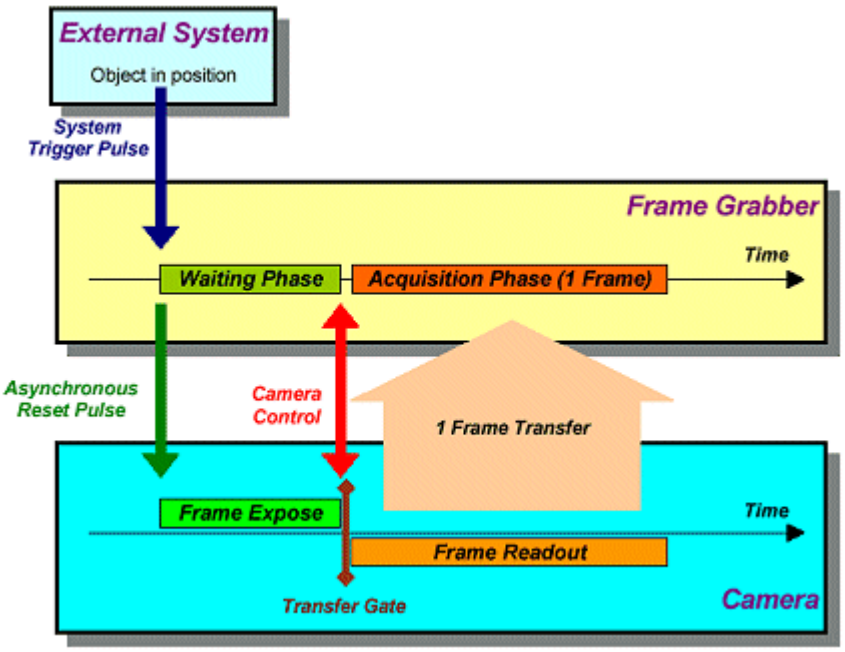


Terminology conventions in area-scan

Area-Scan camera and System Relationship

The following figure is a conceptual view of the task an industrial frame grabber is in charge of. An area-scan camera and some part of the external system are considered.

This drawing addresses the most complex case. It may happen that the actual application involves only a part of the figured resources.



The external system determines that a scene is to be captured. It can be a moving object found to have reached the right location thanks to judicious position sensors.

The system sends a trigger pulse to the frame grabber. As a reaction, the frame grabber has the following mission:

- To adequately instruct the camera to capture the image as quickly as possible.
- To retrieve the image produced by the camera.
- To deposit the image in digital format into a memory location inside the computer hosting the frame grabber.

In response to the external trigger, the frame grabber sends an asynchronous reset command to the camera, and enters a waiting phase. Simultaneously, the camera enters the frame exposure condition.

Thereafter, an intense data transfer takes place to transport a video frame from the camera into the frame grabber. This is the readout condition for the camera, closely matching the acquisition phase for the frame grabber The rationale for the acquisition phase is to bring a video frame from the camera into the frame grabber.

Finally, the frame grabber applies a rich set of data conditioning actions to the acquired image, and sends it through the peripheral bus to the host computer memory.

Area-Scan Operational Modes

Four fundamental camera modes are applicable to the area-scan camera and grabber association.

MultiCam provides two expert-level parameters named **Expose** and **Readout** to declare the camera operational mode. Their possible combinations are the following:

Operational mode	Expose	Readout	Camera mode description
SC	**INTCTL**	**INTCTL**	The camera operates in the "synchronous scanning" modality. The light exposure duration is set by the camera following some form of earlier configuration.
SP	**INTPRM**	**INTCTL**	The camera operates in the "synchronous scanning" modality. The exposure is permanent.
RC	**PLSTRG**	**INTCTL**	The camera operates in the "asynchronous reset" modality. The light exposure duration is set by the camera following some form of earlier configuration.
RG	**WIDTH**	**INTCTL**	The camera operates in the "asynchronous reset" modality. The frame grabber positively controls the camera light exposure duration through the specific MultiCam parameter `Expose_us`.

Area-Scan Acquisition Modes

Fundamental acquisition modes applicable to the area-scan camera and grabber association

AcquisitionMode	Description
VIDEO	Automated acquisition of several video sequences from a standard area-scan camera.
SNAPSHOT	Controlled capture of one or several pictures of one of several objects.
HFR	Acquisition of snapshot images from a camera delivering the frames at a very high frame rate.

For detailed information, refer to the **AcquisitionMode** parameter.

White Balance

What Is White Balance?

Color image acquisition

A color image acquisition involves the use of three color filters on the camera sensor. Each color filter restricts the light source to one wavelength of the light spectrum, either red (R), green (G), or blue (B).

An ideal capture system renders a white object as a white image. A white stimulation should yield the same signal for R, G and B filters. But practically, there are always unavoidable defects on the signals that introduce a **white imbalance**.

White imbalance factors

Several factors, due to the camera and to the capture conditions, are responsible for the white imbalance:

- Object illumination. The color of an object is a combination of its reflectivity and the spectral contents of the illuminating light.
- Camera optical filters response.
- Sensor sensitivity, which is not the same for the three ranges of wavelength.
- Different gain coefficients applied to each color signal before digitization.

White balance correction

MultiCam can correct the white imbalance of the capture system. The operation is called the **white balance**:

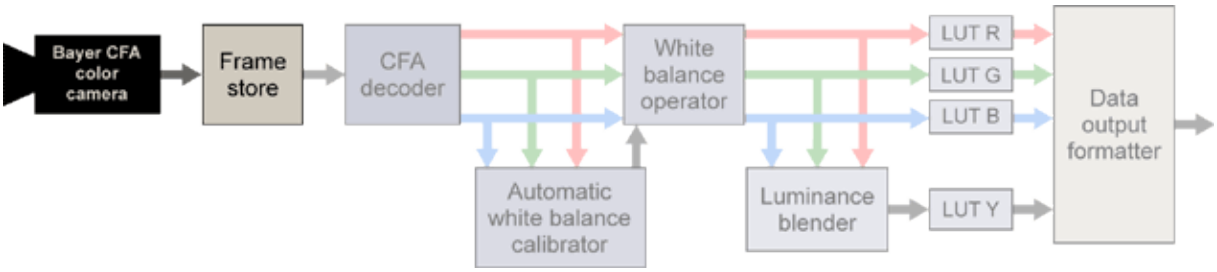
- The **white balance operator** applies correcting coefficients (R, G, and B gains) to each color signal, so, for a white object, the sum of the R, G, and B signals renders a white image.
- The **white balance calibration** is the computation of the three R, G, and B gains. It is performed on a representative image area, prior to the image capture. It can be automatic or manual.

Automatic Calibration

The white balance calibration can be performed during the image acquisition process. If enabled, this acquisition is performed in 3 steps.

Step 1: image acquisition

During the step 1, the frame store collects the raw image issued by the camera. This step is initiated by the MultiCam controller and its duration is a camera characteristic.

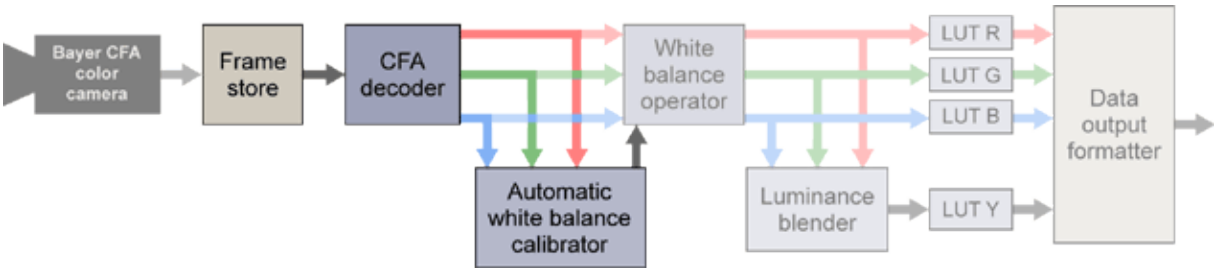


Step 1

Step 2: gain coefficients computation

A rectangular region of interest or AWB ROI is extracted from the frame store, decoded and analyzed by the automatic white balance calibrator. During this analysis, the 3 gain coefficients are computed, one for each color component.

The step 2 starts when the step 1 is complete, that is when the image acquisition is completed. The duration of the step 2 depends on the size of the AWB ROI. See the AWB performance specification for detailed informations in the board handbook.

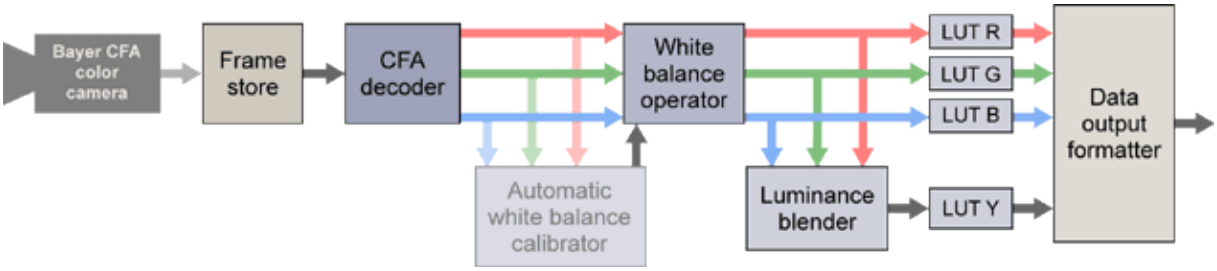


Step 2

Step 3: image color correction

The image is extracted from the frame store, decoded, white balanced with the gain coefficients computed in the step 2, and sent to the host memory through the LUT. Simultaneously, the luminance blender generates the Y component of the balanced image.

The step 3 begins immediately after the completion of the step 2 and its duration depends on the size of the image transferred, the data output format, and the PCI performance.



Step 3

Operating Modes

There are 4 operating modes for the white balance calibration. The mode is selected with the **WBO_Mode** parameter that can take the following values :

- **NONE** (default value)
- **MANUAL**
- **CONTINUOUS**
- **ONCE**

The automatic white balance calibration is effective only in the 2 latest modes.

No calibration

MultiCam parameter: **WBO_Mode** = **NONE**

In this mode, the white balance operator is disabled.

Manual calibration

MultiCam parameters: **WBO_Mode** = **MANUAL**, **WBO_GainR**,**WBO_GainG** and **WBO_GainB**must be defined

The 3 gains to apply to the image can be obtained with one of the following methods:

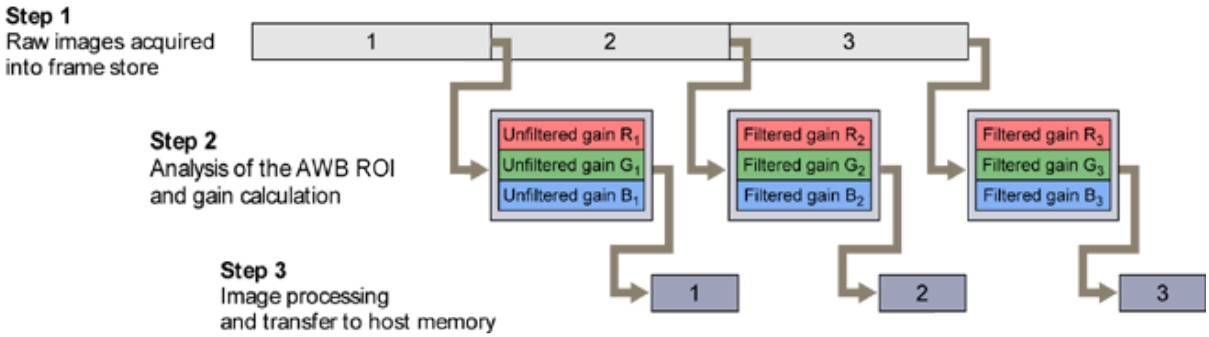
- User-owned wite balance analysis tool performed on an image previously acquired without calibration.
- Re-use gains previously computed by Grablink Quickpack CFA.

This is the fastest mode to obtain the corrected image in the computer memory.

Automatic CONTINUOUS calibration

MultiCam parameters: **WBO_Mode** = **CONTINUOUS**, **WBO_OrgX**, **WBO_OrgY**, **WBO_Width** and **WBO_Height** should be defined

In this mode, the gains are computed on all acquisitions. The first image of the sequence is corrected with the first measured gain. The subsequent images are corrected with time-filtered gains computed from the previous gains and the newly computed gains.



Timing diagram of CONTINUOUS calibration

As all images including the first one are adequately corrected, this method is recommended when the color imbalance varies with time. It is applicable only when the calibration can be done on the image of the inspected object.

The default AWB ROI is the whole image. So, it is recommended to redefine this ROI when the calibration area is located in a specific area. The use of a smaller ROI will reduce the gains computing time.

Following features of Grablink Quickpack CFA are effective:

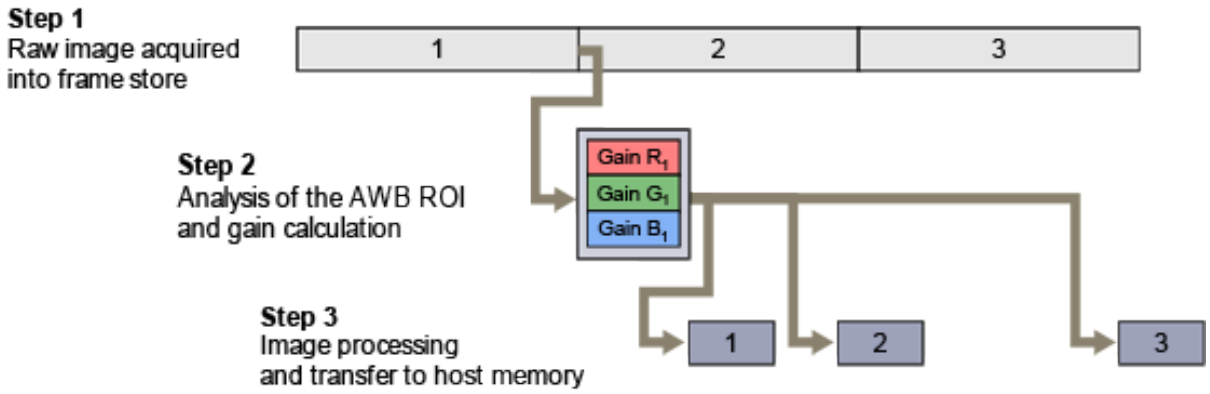
- The gains computation is performed quickly.
- The search area is programmable.

- The computed gains are applied immediately.
- The gains are adaptive.
- The calibration status is monitored.

Automatic ONCE calibration

MultiCam parameters: **WBO_Mode** = **ONCE**, **WBO_OrgX**, **WBO_OrgY**, **WBO_Width** and **WBO_Height** can be defined

In this mode, the gains are computed on the first acquisition and used for all images of the sequence.



Timing diagram of ONCE calibration

This method is recommended when the color imbalance does not vary with time. It is applicable only when the calibration can be done on the image of the inspected object.

The default AWB ROI is the whole image. So, it is recommended to redefine this ROI when the calibration target is located in a specific area. The use of a smaller ROI will reduce the gains computing time.

The gains computation is executed only on the first image of every sequence. Subsequent images in the sequence are processed without any delay, thus reducing the time to obtain the image in computer memory.

Following features of Grablink Quickpack CFA are effective:

- The gains computation is performed quickly.
- The search area is programmable.
- The computed gains are applied immediately.
- The calibration status is monitored.

2.3. Line-Scan Frame Grabber Topics

Trigger, Reset and Strobe in Line-Scan

In order to ensure the terminology consistency throughout the MultiCam documentation, the following conventions have been taken.

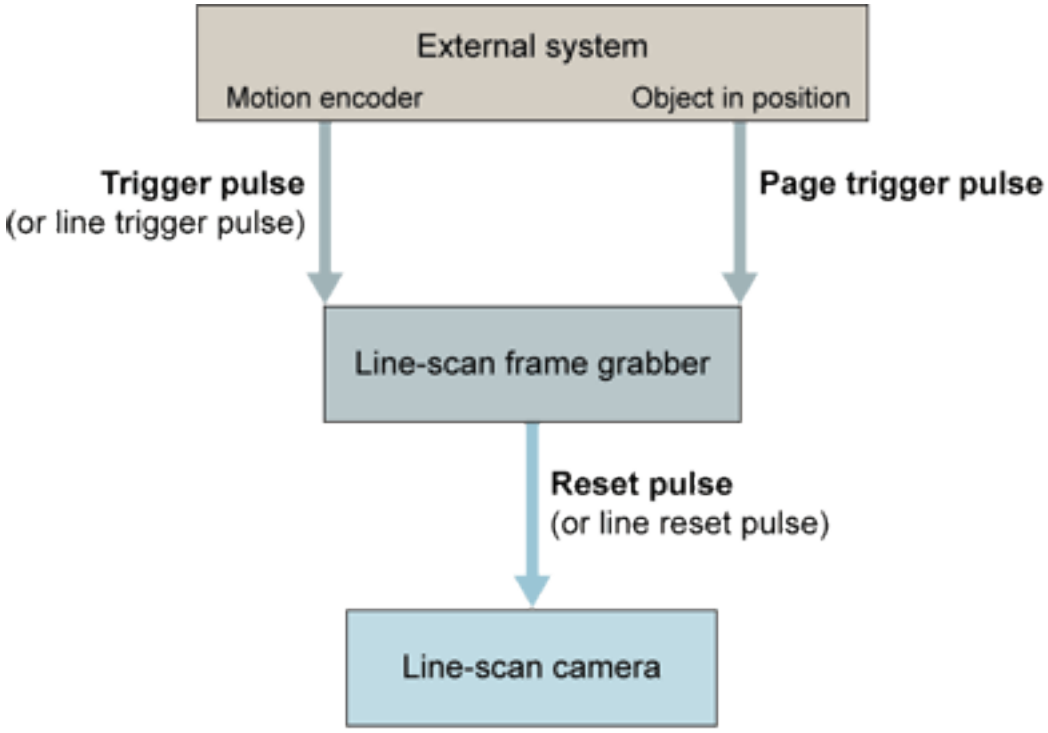
A **trigger pulse** is an electrical signal sent by the external system to synchronize the line rate of the camera to some external reference. This is needed to preserve the geometric aspect of a line-scanned object moving at a non-uniform speed. The trigger pulse is often referred to as a line trigger pulse, although the rate conversion feature results in a trigger frequency that may not be equal to the camera line frequency.

A **page trigger pulse** is an electrical signal sent by the external system to instruct a frame grabber to perform the acquisition of a set of several successive lines. This is usually used when a moving object is about to enter the field of view of the line-scan camera.

In the case of line-scan, an adjacent set of scanned lines is called a page. The name page is borrowed to the document scanning application, and has been chosen to denote the case of line-scan cameras. The area-scan corresponding item is a frame. The length of the page is chosen to adequately cover the extension of the moving object.

Due to the similarities of the frame and page concepts, a page trigger is sometimes referred to as a frame trigger. The line-scan or area-scan context makes the difference.

A **reset pulse** is an electrical signal sent by the frame grabber to instruct a line-scan camera to start its line acquisition cycle. The reset pulse is often referred to as a line reset pulse.

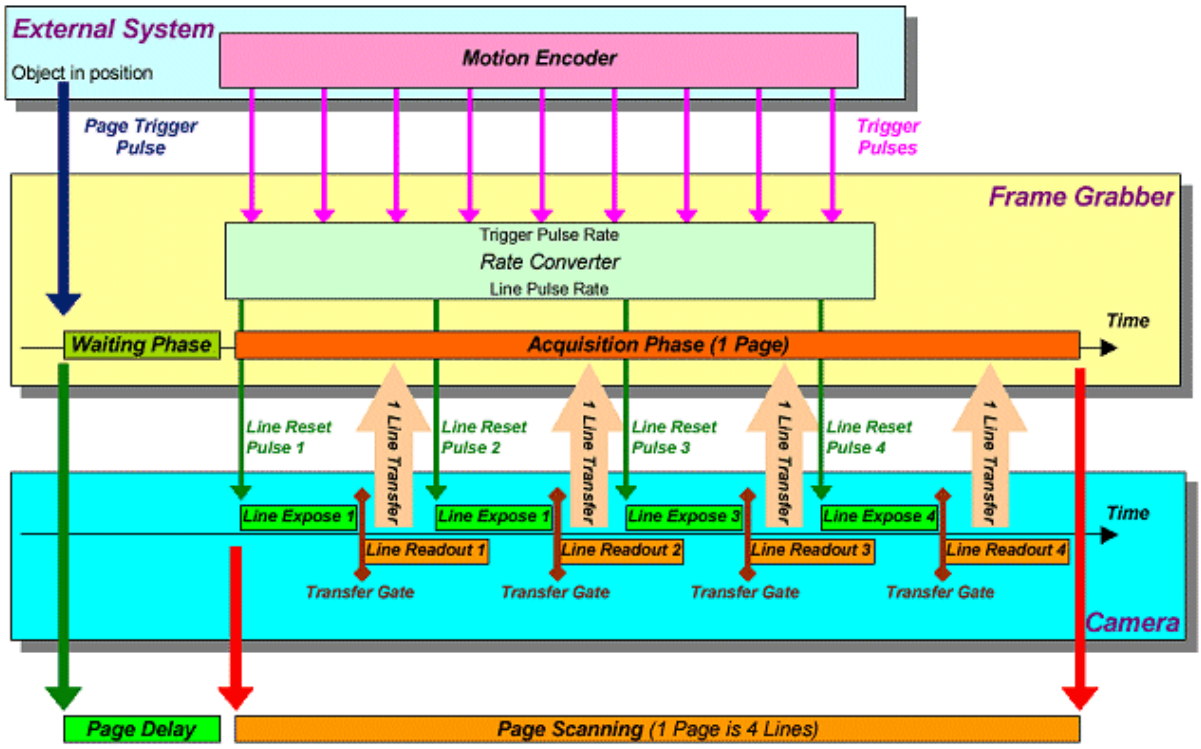


Terminology conventions in line-scan

Line-Scan Camera and System Relationship

The following figure is a conceptual view of the task that an industrial frame grabber is in charge of. A line-scan camera and some part of the external system are considered.

This drawing addresses the most complex case. It may happen that the actual application involves only a part of the figured resources.



At any time, the frame grabber controls the line sequencing of the camera. To achieve this, a succession of line-reset pulses is sent to the camera at a frequency proportional to the repetition frequency of a succession of trigger pulses sent by a motion encoder. This is the effect of the built-in rate converter.

The external system determines that a moving object is about to reach the camera field of view. The position detector can be located well in front of the actual observed line. A page trigger pulse is issued to inform the frame grabber on this instant.

As a reaction, the frame grabber enters a programmable delay, called the page delay. This page delay can be expressed as a number of scan lines. At the end of the page delay, the object to be scanned effectively enters the field of view.

The page delay corresponds to the waiting phase inside the frame grabber. Following the waiting phase, the acquisition phase takes place. The rationale for the acquisition phase is to bring a video page from the camera into the frame grabber.

The page is made of a number of video lines. At each line, an intense data transfer takes place to transport data from the camera into the frame grabber. Each transfer corresponds to the readout condition for the camera.

When all the lines are elapsed, the page is fully acquired. The frame grabber applies a rich set of data conditioning actions to the acquired image, and sends it through the peripheral bus to the host computer memory.

Line-Scan Operational Modes

Six fundamental camera operational modes are applicable to the line-scan camera and grabber association.

MultiCam provides two expert-level parameters named **Expose** and **Readout** to declare the camera operational mode. Their possible combinations are the following:

Operational mode	Expose	Readout	Camera mode description
SC	**INTCTL**	**INTCTL**	The camera operates in the "synchronous scanning" modality. The light exposure duration is set by the camera following some form of earlier configuration.
SP	**INTPRM**	**INTCTL**	The camera operates in the "synchronous scanning" modality. The exposure is permanent.
RP	**INTPRM**	**PLSTRG**	The camera operates in the "asynchronous reset" modality. The exposure is permanent.
RC	**PLSTRG**	**INTCTL**	The camera operates in the "asynchronous reset" modality. The light exposure duration is set by the camera following some form of earlier configuration.
RG2	**PLSTRG**	**PLSTRG**	The camera operates in the "asynchronous reset" modality. The light exposure duration is set by the camera following some form of earlier configuration.
RG	**WIDTH**	**INTCTL**	The camera operates in the "asynchronous reset" modality. The frame grabber positively controls the camera light exposure duration through the specific MultiCam parameter **Expose_us**.

Line-Scan Acquisition Modes

Fundamental acquisition modes applicable to the line-scan camera and grabber association

AcquisitionMode	Description
PAGE	The PAGE acquisition mode makes the line-scan operation similar to an area-scan operation. The line-scan camera is expected to provide a set of contiguous line constituting the image of the object moving in front of it. This mode is intended for image acquisition of multiple discrete objects.
WEB	The WEB acquisition mode is dedicated to applications where one object of indeterminate size has to be observed by a line-scan camera.
LONGPAGE	The LONGPAGE acquisition mode is also dedicated to the observation of multiple discrete objects, but of larger size than for the Page acquisition mode. Moreover, this mode has the capability to acquire variable size objects.

For detailed information, refer to the **AcquisitionMode** parameter and to Using Line-Scan Acquisitions.

Line Capture Modes

Three fundamental line capture modes are applicable to the line-scan camera and grabber association:

LineCaptureMode	Description
ALL	**Take All Lines** – Each delivered camera line results into a line acquisition. If the downweb motion speed is varying, the line-scanning process of the camera should be rate-controlled accordingly.
PICK	**Pick A Line** – The line-scanning process of the camera is running at a constant rate. Each pulse occurring at the *Downweb Line Rate* determines the acquisition of the next line delivered by the camera.
ADR	**Advanced Downweb Resampling** – The line-scanning process of the camera is running at a constant rate. Each pulse occurring at the *Downweb Line Rate* determines the computation of a line from a set of presently recorded lines delivered by the camera.

When **LineCaptureMode** = **ALL**, the *Downweb Line Rate* and the *Camera Line Rate* are the same. The requested resolution and the effective motion speed uni-vocally dictate the *Downweb Line Rate*. Then the camera has to be chosen to operate at an exactly matching Camera Line Rate, even if the speed of motion is varying. This imposes a requirement for a rate-controllable camera.

Using *Downweb Resampling* offers a way to eliminate the requirement for this exact match. The *Camera Line Rate* may be chosen at a fixed value, and the acquisition will still acquire lines at the expected downweb resolution, even when the speed of motion is varying.

The ADR – *Advanced Downweb Resampling* – technology re-samples the camera data along the vertical direction using multi-tap interpolation filter while the Pick-A-Line technology uses a rough resampling method, namely the Nearest neighbor method. (The ADR technology is exclusively available on Grablink Avenue and Grablink Express.)

For technical information, refer to the **LineCaptureMode** reference.

Synchronization modes, in function of the camera operation mode

Imaging	CamConfig	LineCaptureMode									
		ALL					PICK			ADR	
		LineRateMode									
		CAMERA	PERIOD	PULSE	CONVERT	EXPOSE	PERIOD	PULSE	CONVERT	PERIOD	CONVERT
LINE	Lxxxx SP	✓					✓	✓	✓	✓	✓
	Lxxxx RP			✓ (*)	✓ (*)	✓	✓	✓	✓	✓	✓
	Lxxxx SC	✓					✓	✓	✓	✓	✓
	Lxxxx RC		✓	✓	✓						
	Lxxxx RG		✓	✓	✓						
	Lxxxx RG2		✓	✓	✓						
TDI	Lxxxx SP	✓					✓	✓	✓	✓	✓
	Lxxxx RP			✓ (*)	✓ (*)	✓	✓	✓	✓	✓	✓

Note: *(*) These settings are not recommended since the camera sensitivity is varying with the line rate.*

Line Rate Modes

Line Rate Mode expresses how the *Downweb Line Rate* is determined in a line-scan acquisition system.

The user specifies the *Line Rate Mode* by means of MultiCam parameter **LineRateMode**. Five *Line Rate Modes* are identified in MultiCam:

LineRateMode	Description
CAMERA	**Camera** – The *Downweb Line Rate* is originated by the camera.
PULSE	**Trigger Pulse** – The *Downweb Line Rate* originates from a train of pulses applied on the line trigger input belonging to the grabber.
CONVERT	**Rate Converter** – The *Downweb Line Rate* originates from a train of pulses applied on the line trigger input and processed by a rate converter belonging to the grabber.
PERIOD	**Periodic** – The *Downweb Line Rate* originates from an internal periodic generator belonging to the grabber
EXPOSE	**Exposure Time** – The *Downweb Line Rate* is identical to the camera line rate and established by the exposure time settings

LineRateMode = CAMERA

This mode is applicable exclusively for free-run permanent exposure – **LxxxxSP** – class of line scan cameras when **LineCaptureMode** = **ALL**. The grabber does not perform any sampling in the downweb direction; the *Downweb Line Rate* is equal to the camera line rate. The camera line rate is entirely under control of the camera. Notice that most of the line scan cameras provide an internal line rate adjustment.

LineRateMode = PULSE

When the speed of motion is varying, the *Downweb Line Rate* should be slaved to this motion. To achieve this, a motion encoder is a good solution.
The motion encoder delivers an electrical pulse each time the moving web advances by a determined amount of length. The continuous motion results in a train of pulses the frequency of which is proportional to the web speed.
There exists another way to take knowledge of the web speed. In some applications, the motion is caused by a stepping motor controlled by pulses. The controlling train of pulses is also a measure of relative motion.
In both cases, the pulses are called line trigger pulses, and their repetition rate is the Line Trigger Rate. The line trigger pulses are applied to the frame grabber to determine the *Downweb Line Rate* .
Each line trigger pulse may result into the generation of one line in the acquired image. This means that the *Downweb Line Rate* is equal to the Trigger Rate.

LineRateMode = CONVERT

Alternatively to the "PULSE" mode, for more flexibility, the Line Trigger Rate may be scaled up or down to match the required *Downweb Line Rate*. The proportion between the two rates is freely programmable to any value lower or greater than unity, with high accuracy. This makes possible to accommodate a variety of mechanical setups, and still maintain a full control over the downweb resolution. The hardware device responsible for this rate conversion is called the

rate converter. This device is a unique characteristic of Euresys line-scan frame grabbers.

LineRateMode = PERIOD

Other circumstances necessitate the *Downweb Line Rate* to be hardware-generated by a programmable timer, called the "periodic generator".

LineRateMode = EXPOSE

Applies to: Express | Base | DualBase | Full | FullXR

This mode is applicable exclusively for line rate controlled permanent exposure – **LxxxxRP** – class of line scan cameras when **LineCaptureMode** = **ALL**. The grabber does not perform any sampling in the downweb direction; the *Downweb Line Rate* is equal to the camera line rate. The camera line rate is entirely under control of the grabber through the exposure time settings.

This mode is the default and recommended mode for **LxxxxRP** class of cameras on 1621 Grablink Express.

> **Note:** *This mode is not available on 1191 Grablink Value.*
> *Instead, use **LineRateMode** = **PERIOD**, specify the desired exposure time through **Period_us** and assign a value to **Expose_us** that is smaller than the desired exposure time.*

Line-Scan Synchronization Modes

The following table summarizes all synchronization mechanisms.

Imaging	CamConfig	LineCaptureMode	LineRateMode	
			Default value	Alternate value(s)
LINE	LxxxxSP LxxxxSC	ALL	CAMERA	-
		PICK	CONVERT	PERIOD, PULSE
		ADR	CONVERT	PERIOD
	LxxxxRP	ALL	EXPOSE	PULSE (*), CONVERT (*)
		PICK	CONVERT	PERIOD, PULSE
		ADR	CONVERT	CONVERT
		ALL	PERIOD	PULSE, CONVERT
TDI	LxxxxSP	ALL	CAMERA	-
		PICK	CONVERT	PERIOD, PULSE
		ADR	CONVERT	PERIOD
	LxxxxRP	ALL	EXPOSE	PULSE (*), CONVERT (*)
		PICK	CONVERT	PERIOD, PULSE
		ADR	CONVERT	PERIOD

Note: *(*) These settings are not recommended since the camera sensitivity is varying with the line rate.*

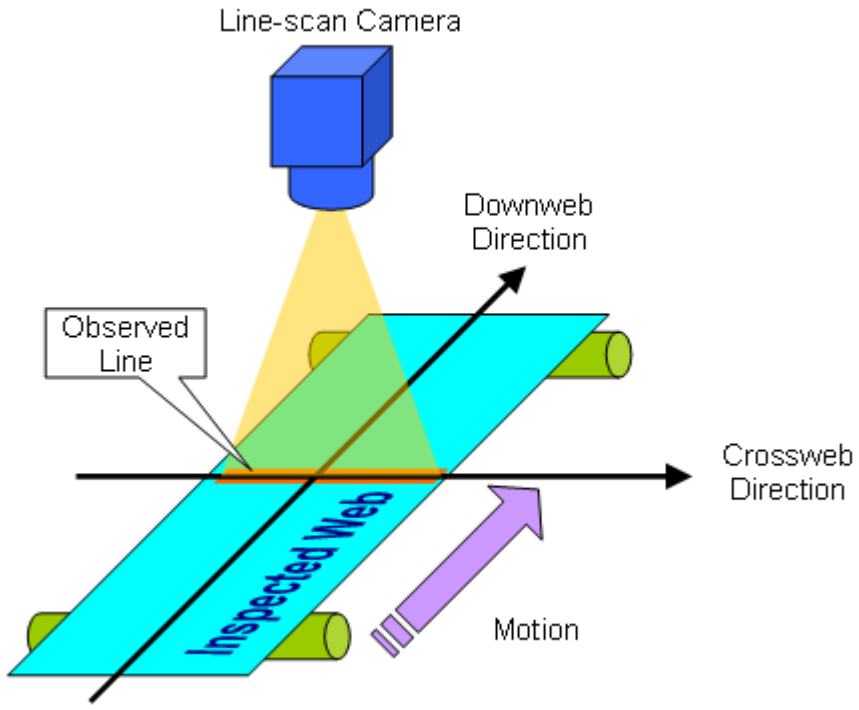
3. Line-Scan Inspection Topics

3.1. A Simplified View of a Typical Setup

A line-scan camera transforms the light intensity along a line into a time varying video signal.

A line-scan based system usually observes a continuous material or set of objects exhibiting a regular shifting movement. We will refer to this material as the "inspected web".

Conceptually, the system reconstructs a 2D representation of the moving scene from a 1D analysis.



Two directional references are introduced:

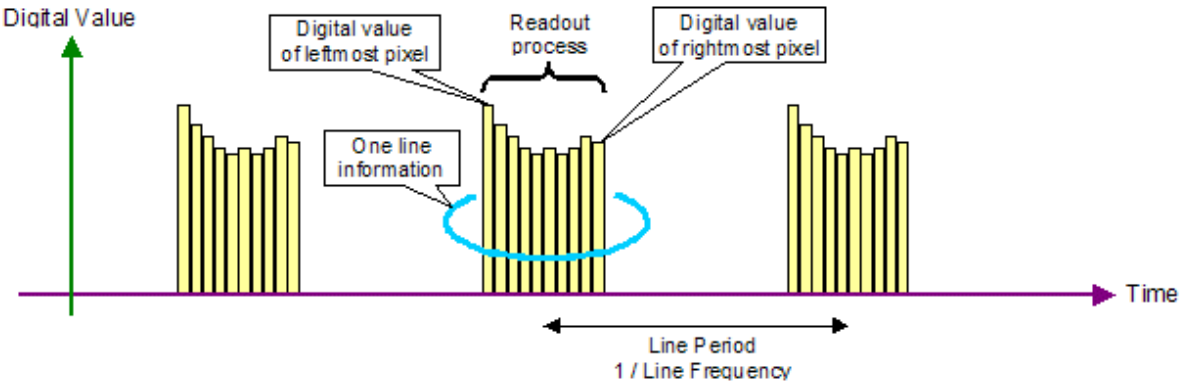
- the **downweb direction**: the motion direction of the inspected web. It may be also named the axial direction.
- the **crossweb direction**: the axis of the observed line. It may be also called the transverse direction.

Those directions are perpendicular to each other.

3.2. The Video Line

The observed line is divided into a set of aligned pixels. In the case of a digital line-scan camera, the produced video signal is a set of digital values corresponding to the light intensity measured for each consecutive pixel.

On a timing diagram, the output signal of a digital line-scan camera can be represented as follows.



The observed line is repetitively scanned pixel after pixel from left to right, and the corresponding light intensity is translated into a set of digital numbers. The digital numbers are quickly output over a digital bus at a speed known as the pixel frequency.

A fixed number of pixels is output for each line. The operation consisting in outputting a full line information is called the readout process.

The readout process is periodically repeated. The recurrence period is called the line period, associated to its inverse value the line frequency.

It should be understood that the line information of two successive readout processes is not identical. Because the inspected web moves, the observed light intensity pattern under the camera changes, and so changes the digital output pattern.

> **Note:** *The left to right order presented in this explanation is for example only. Mechanical and optical considerations may lead to the opposite order as well.*

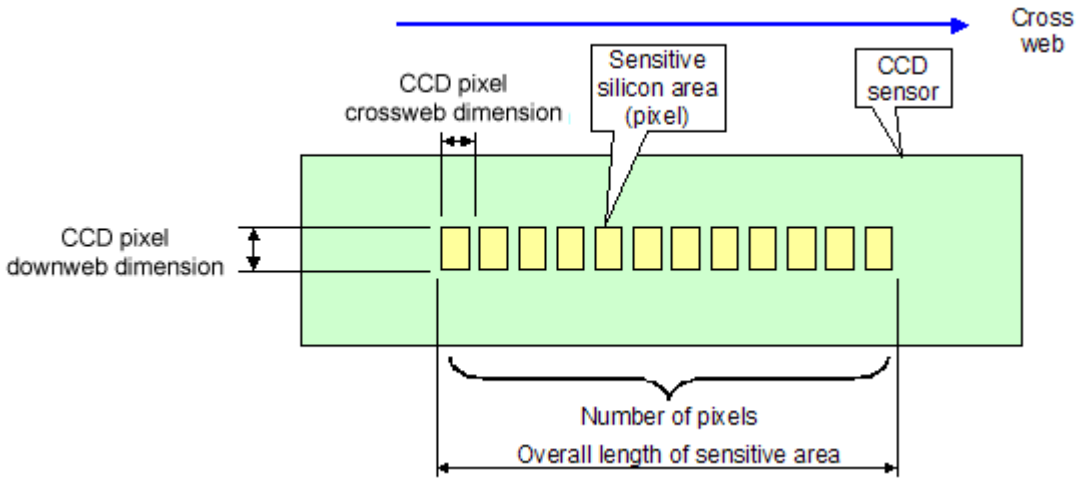
3.3. The CCD Sensor

The heart of the line-scan camera is an opto-electronic semiconductor device called a CCD (Charge Coupled Device). More precisely, a line CCD is involved, as opposed to the area CCD used in the area scan cameras.

A line CCD sensor is a set of light sensitive elements aligned on a small piece of silicon. Each element correspond to a pixel. Some dedicated electronic circuitry is associated to extract the data measured by the individual elements and send it outside the device during the readout process.

In the line-scan based system, the CCD is aligned along the crossweb direction.

The following figure highlights some important dimensional features of the CCD sensor.



The CCD pixel crossweb dimension is also called the CCD pitch.

Usually, the CCD pixel has the same crossweb and downweb dimension, i.e. the sensitive area for an individual pixel is square.

A typical size for a CCD pixel is in the order of magnitude of 10 by 10 µm.

The number of pixels for currently available CCD sensors and line-scan cameras is in the range of 200 to 12,000.

As an example, a typical CCD sensor could have 2048 pixels, each sized 14x14 µm. In this case, the overall length of the CCD sensitive area is 28.67 mm.

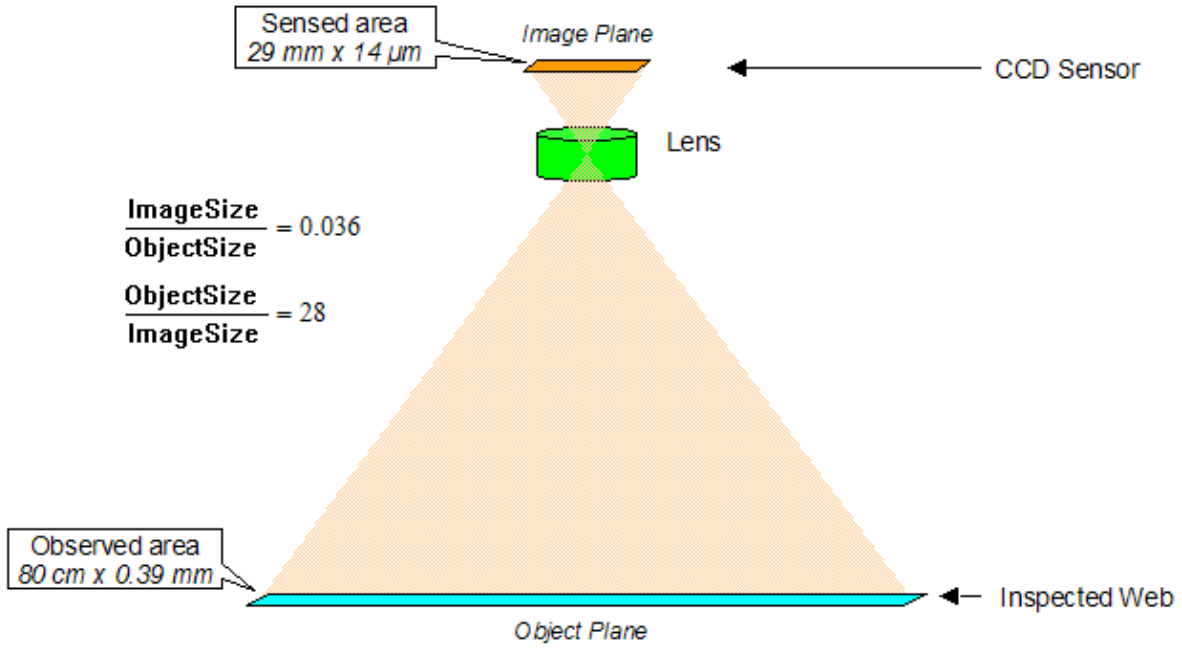
3.4. The Optical Setup

A lens is used to project the image of the observed line over the sensitive part of the CCD sensor.

For the sake of this introduction, the most prevalent attribute of the optical setup is the magnification ratio.

For instance, a particular line-scan based system is being designed to inspect a web over a 80cm width. The camera used is equipped with a CCD sensor the useful length of which is 28.67 mm.

If we define the magnification as ratio of the optical image to the real object, the required magnification ratio is 0.03584. Actually, this corresponds to a reduction in size from the real object to the optical image formed on the sensor. This reduction can be quantified as 27.90.



Let us assume that the CCD sensor size is 14x14 µm. The size of the corresponding pixel on the web is 0.39x0.39 mm.

Throughout this introduction, we will keep these numbers as an example of a representative application.

3.5. Basic Resolution Issues

Transverse Resolution

The crossweb resolution of the image system is determined by the distance between two adjacent pixels on the inspected web in the crossweb direction. Let us call this distance the "crossweb pitch".

It only depends on the CCD sensor geometry and the optical setup.

In the case of the example of a 2048 pixel camera observing a 80cm wide web, the crossweb pitch is 0.39 mm.

Axial Resolution

The downweb resolution of the image system is determined by the distance between two adjacent pixels on the inspected web in the downweb direction . Let us call this distance the "line pitch".

Contrary to the crossweb resolution, the downweb resolution is not dependent on the CCD sensor geometry nor on the optical setup.

The downweb resolution corresponds to the gap between two successively scanned lines.

The time between two video line readout processes is the line period. Considering the motion speed of the web, the distance traveled by the web during one line period is easily computed. This distance is the line pitch. It purely reflects the downweb resolution.

$$\mathrm{LinePitch} = \mathrm{WebSpeed} \times \mathrm{LinePeriod}$$

or equivalently

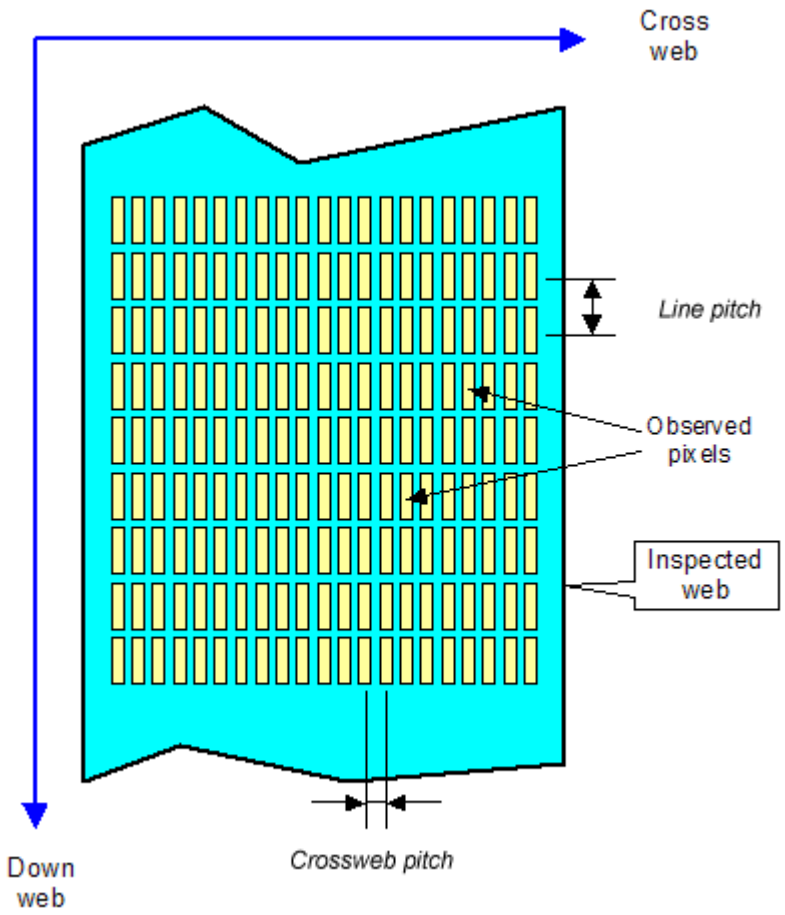
$$\mathrm{LinePitch} = \frac{\mathrm{WebSpeed}}{\mathrm{LineFrequency}}$$

For instance, consider a camera working at the line frequency of 1 kHz. The line period is 1 ms.

Consider that the transport mechanical system moves the web at 50 cm/s.

In this situation, the downweb resolution, i.e. the line pitch, is 0.5 mm.

The Observed Pixels



Spatial distribution of the observed pixels on the inspected web

This is the 2D analysis of the inspected web. The crossweb scanning process is performed by the CCD sensor itself. The downweb scanning process is performed thanks to the web motion.

The line pitch and the crossweb pitch are not necessarily equal.

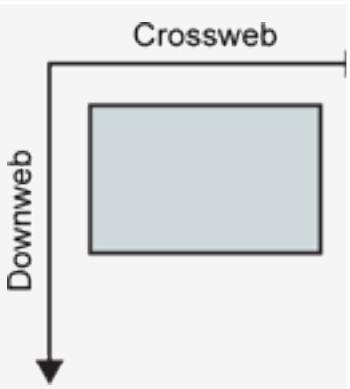
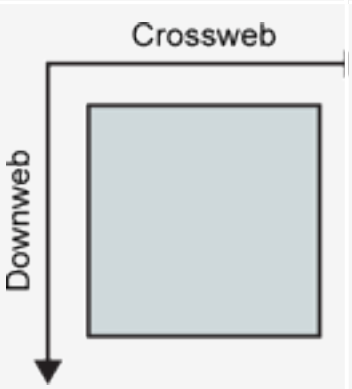
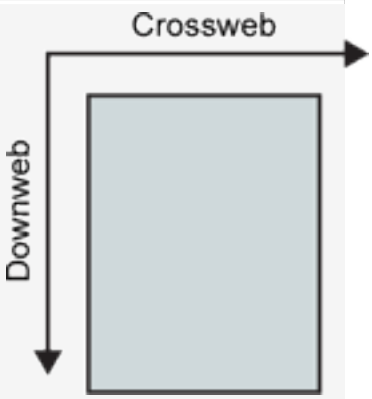
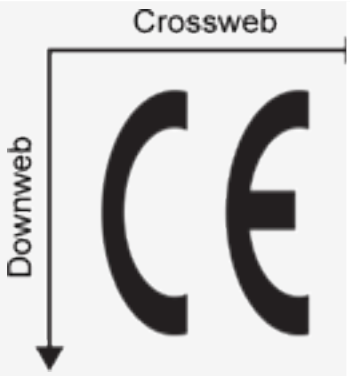
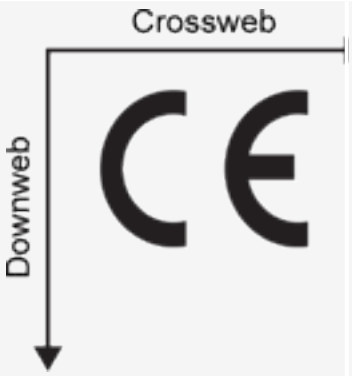
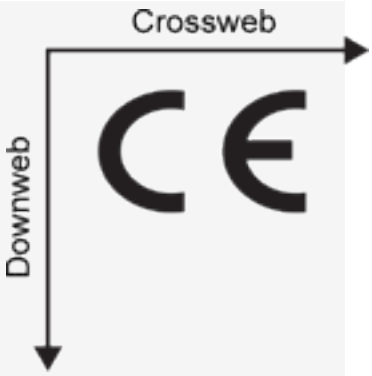
Aspect Ratio

In general, the line pitch is not equal to the crossweb pitch. The result is that the digital image provided by the frame grabber is not geometrically exact.

For example, if the web speed and the camera line frequency are such that the line pitch is larger than the crossweb pitch, the resulting image will appear somehow compressed in the downweb direction.

The following table displays the result of varying the web speed for our representative example.

	Slow web	Correct speed	Fast web
Web speed	25 cm/s	39 cm/s	50 cm/s
Camera line frequency	1 kHz	1 kHz	1 kHz
Crossweb pitch	0.39 mm	0.39 mm	0.39 mm
Line pitch	0.25 mm	0.39 mm	0.5 mm
Aspect ratio	1.56	1	0.78

	Slow web	Correct speed	Fast web
Single pixel look			
Image look			

In this table, we define the aspect ratio as:

$$\text{AspectRatio} = \frac{\text{CrosswebPitch}}{\text{LinePitch}}$$

This shows that, if the aspect ratio has to be controlled for the application (and usually it has to), the web speed and the camera line frequency should be coupled together somehow in order to maintain the line pitch to a known value.

3.6. Exposure Issues

Some additional knowledge on the way the CCD sensor operates is needed to understand the goals of the camera control means offered by MultiCam.

In order to build an electrical signal representing the light intensity of a point, the CCD pixel area should be exposed to the light during a certain amount of time. This amount of time is known as the exposure time, or the integration time.

In a line CCD sensor, all the pixels experience simultaneously the exposure period. This means that the exposure starting and stopping instants are common to all pixels.

Two cases are to be considered:

- The permanent exposure,
- The controlled exposure.

All line-scan cameras support the permanent exposure scheme.

Not all line CCD sensors and not all line-scan cameras are able to support the controlled exposure scheme.

In order to support the controlled exposure scheme, a line-scan camera should be fitted with a so-called electronic shutter feature.

As this will become clear later in the document, the industrial vision integrator will prefer a line-scan camera equipped with the electronic shutter.

Electronic Shutter

The CCD sensor can be temporarily set in a "pixel reset" condition. In this condition, the light has no effect on the sensitive area of the sensor. Any electrical charge that could be contained in the pixels is cleared.

At some instant, the camera leaves the pixel reset condition to enter into the exposure or integration condition. During this time, every pixel builds an electrical charge growing at a rate proportional to the light intensity it receives. The longer the exposure time, the bigger the electrical charge.

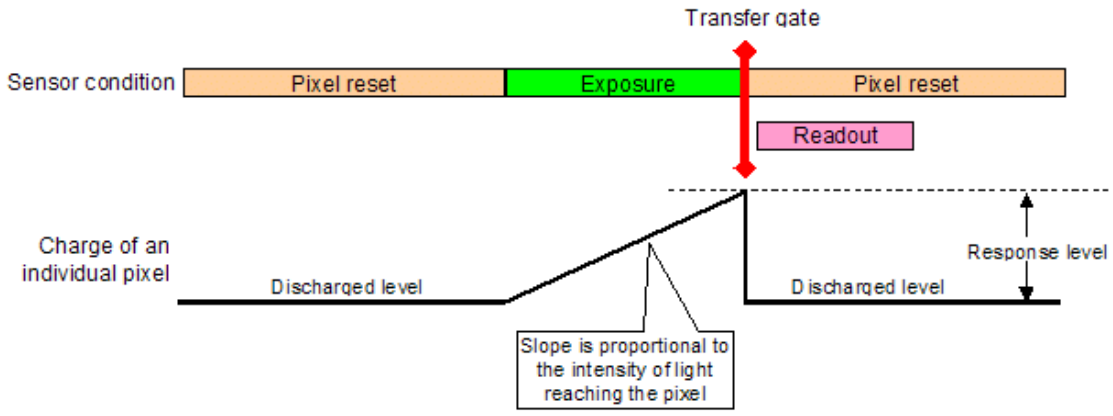
Consequently, increasing the exposure time is a mean to increase the light sensitivity of the camera.

The end of the exposure time is marked by a special event usually referred to as the "transfer gate". At this instant, the individual electrical charges built by the pixels are set aside and made ready for transport towards the CCD sensor output.

Simultaneously, the pixels gets emptied of any electrical charge. This sets the pixels in the same state as the pixel reset feature does.

After the transfer gate, the CCD sensor enters the readout period. This takes a fixed amount of time to extract the individual electrical charges set aside at the transfer gate instant. The charges are converted to voltage and serially conveyed outside the CCD sensor. Subsequently, the voltage is converted into digital values, and serially conveyed outside the line-scan camera.

Graphically, the operational sequence can be presented as follows:



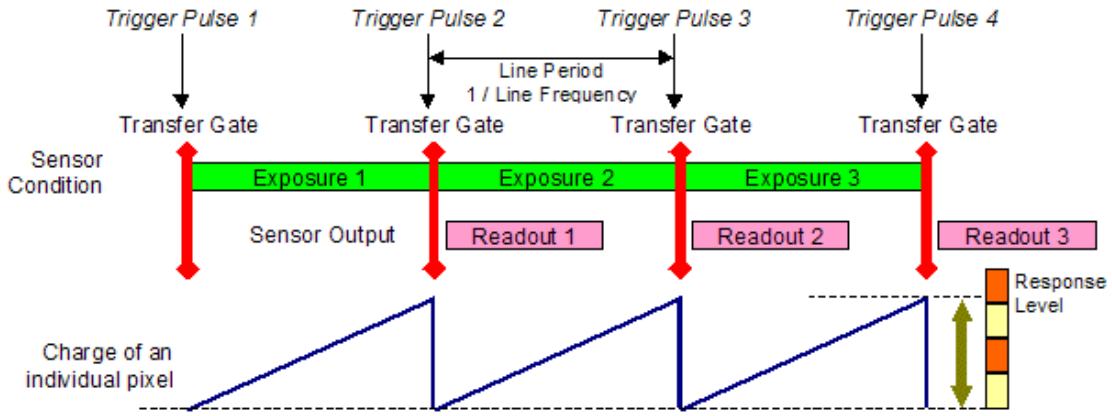
Permanent Exposure

In the permanent exposure situation, the CCD sensor is never brought into the pixel reset condition.

The line-scan camera hosting the sensor experiences a periodical cycle paced by successive instants we will refer to as "trigger pulses". The trigger pulses occur at the line frequency. Each trigger pulse causes a transfer gate event followed by a readout process.

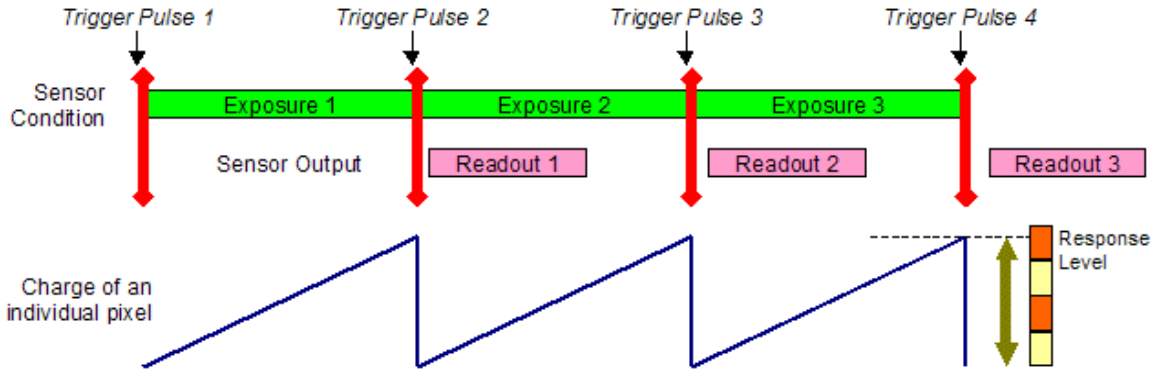
The trigger pulses are issued by the frame grabber according to rules that will become clear later.

The following figure shows the operational sequence at some line frequency.



Between two successive trigger pulses, the sensor integrates the light. The sensitivity of the camera depends on the line period separating the pulses. The lower the line frequency, the higher the response to a given scene.

The following figure shows the operation sequence at some slower line frequency.



It is seen that the pixel response is higher.

Suppose that we are willing to adjust the aspect ratio by varying the line frequency in order to keep it adequately coupled to the web speed. With the permanently exposed sensor, we are faced to some response variation when the web speed is changing.

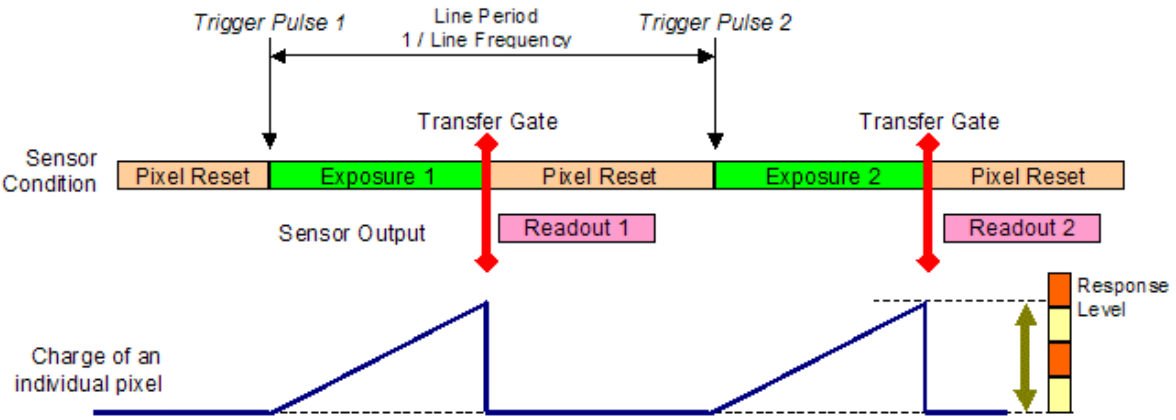
Controlled Exposure

In the controlled exposure situation, the CCD sensor is brought into the pixel reset condition before entering the exposure condition. In other words, the electronic shutter is used.

The line-scan camera hosting the sensor experiences a periodical cycle paced by successive instants we will refer to as "trigger pulses". The trigger pulses occur at the line frequency. Each trigger pulse causes the following sequence:

- Exposure condition of some known duration,
- Transfer gate event,
- Readout process,
- Return to pixel reset condition.

The trigger pulses are issued by the frame grabber according to rules that will become clear later.



Operational sequence

It can be seen that the exposure duration is not equal to the line period. It can be made independent of the line frequency.

Suppose that we are willing to adjust the aspect ratio by varying the line frequency in order to keep it adequately coupled to the web speed. With the controlled exposure sensor, we can do it without any response variation.

This is the fundamental reason why the industrial vision integrator should specify the electronic shutter feature when selecting a line-scan camera.

3.7. Line Frequency Limits

Duration of the Readout Process

The readout process lasts a predictable amount of time depending on the pixel frequency and the number of pixels.

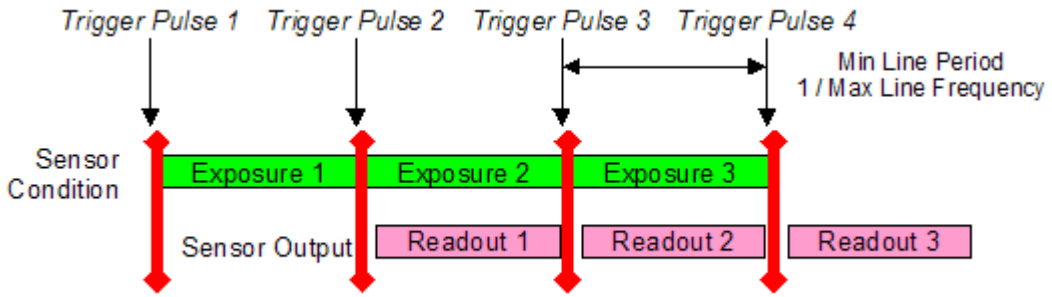
$$\mathrm{ReadoutDuration} = \frac{\mathrm{NumberOfPixels}}{\mathrm{PixelFrequency}}$$

Coming back to our representative application, consider that the 2048 pixels are read out of the sensor at a speed of 4 MHz.

The readout duration is 512 µs in this case.

Line Period for Permanent Exposure

In the permanent exposure operational mode, the maximum line frequency is dictated by the readout duration. The successive readout processes cannot overlap. This is suggested by the following figure.



In fact, the minimum line period is slightly larger than the readout duration.

In the case of the representative application, the minimum line period is about 512 µs, corresponding to a maximum line frequency of about 1950 Hz.

The maximum line frequency in the permanent exposure mode is a characteristic indicated in the camera data sheet. It is a significant performance indicator. The following formulae can be used.

$$\mathrm{MaxLineFrequency} \approx \frac{\mathrm{PixelFrequency}}{\mathrm{NumberOfPixels}}$$

$$\mathrm{MinLinePeriod} \approx \frac{\mathrm{NumberOfPixels}}{\mathrm{PixelFrequency}}$$

It should be understood that operating the camera at the maximum line frequency inherently means that the exposure time is as short as possible. Consequently, the camera exhibits the minimal sensitivity. This implies that the maximum camera rate as specified by the data sheet can not be achievable for illumination reasons.

Line Period for Controlled Exposure

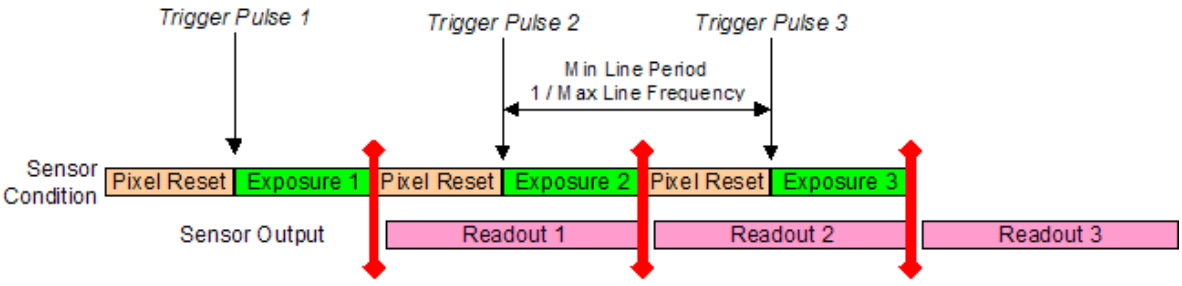
In the controlled exposure operational mode, the maximum line frequency can be dictated by the camera performance, or by system constraints.

Two cases are to be considered:

- The exposure time is shorter than the readout duration,
- The exposure time is longer than the readout duration

Short exposure time

The maximum line frequency is dictated by the camera, exactly as it is in the case of the permanent expose situation.



The maximum line frequency is tied to the readout process duration. The following formulae can be used.

$$\mathrm{MaxLineFrequency} \approx \frac{\mathrm{PixelFrequency}}{\mathrm{NumberOfPixels}}$$

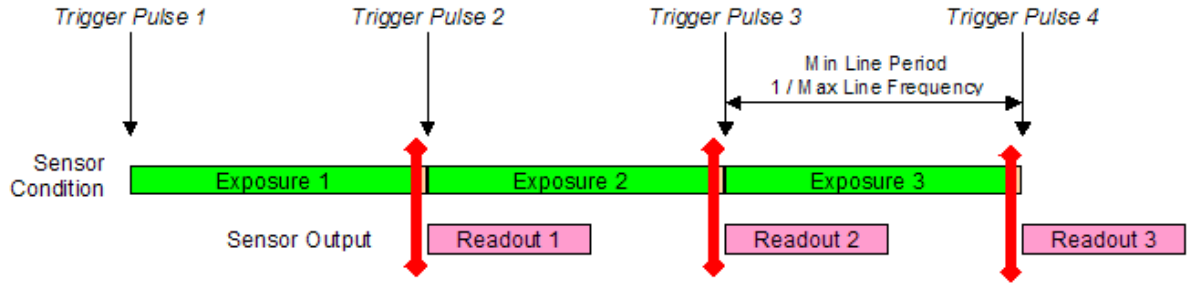
$$\mathrm{MinLinePeriod} \approx \frac{\mathrm{NumberOfPixels}}{\mathrm{PixelFrequency}}$$

Operating the camera at the maximum rate as described here implies an exposure time even shorter than in the case of the permanent exposure situation. The sensitivity is usually so low that an intense illumination of the inspected web is needed.

Long exposure time

Shortening the exposure time drastically reduces the sensitivity of the camera. Sometimes, it is simply not possible to achieve an adequate response with a short exposure time and a reasonable illumination device.

If the needed exposure is longer than the readout duration, then the maximum line frequency is dictated by the exposure time rather than by the camera performance.



The maximum line frequency is tied to the exposure duration. The following formulae can be used.

$$\text{MaxLineFrequency} \approx \frac{1}{\text{ExposureTime}}$$

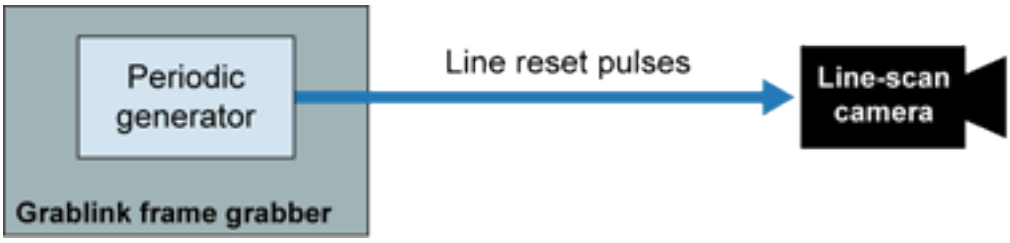
$$\text{MinLinePeriod} \approx \text{ExposureTime}$$

3.8. Triggering a Line-Scan Camera

Internal Line Triggering

This mode is effective when the MultiCam parameter **LineRateMode** is set to **PERIOD**.

In this mode, the trigger pulse sequence is issued by the Grablink frame grabber internally at a pre-defined rate.



Internal line triggering

The MultiCam parameter **Period_us** should be used to set the desired rate. This parameter simply sets the line period and is expressed in microseconds.

To reach a line period of 1 ms (i.e. a line frequency of 1 kHz), you have to set **Period_us** parameter to **1000**.

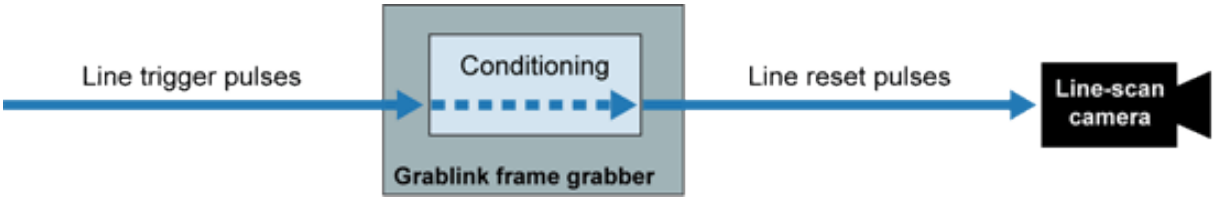
In addition, the MultiCam parameter **PeriodTrim** may be used to modify the periodic frequency using a logarithmic scale from –6 dB up to +12 dB. It applies following correction:

Setting	Effective trimmed period
-6	**Period_us** x 0.5
-3	**Period_us** x 0.7
0	**Period_us**
+3	**Period_us** x 1.4
+6	**Period_us** x 2
+9	**Period_us** x 2.8
+12	**Period_us** x 4

Take into account the line frequency limits introduced above to avoid programming a too short period.

External Line Triggering

This mode is effective when the MultiCam parameter **LineRateMode** is set to **PULSE**.



External line triggering

In this mode, the trigger pulse generated by the frame grabber and reaching the camera is a copy of a system trigger pulse applied to the frame grabber trough a dedicated hardware line.

The MultiCam parameter **LineTrigCtl** is used to select the appropriate signal style.

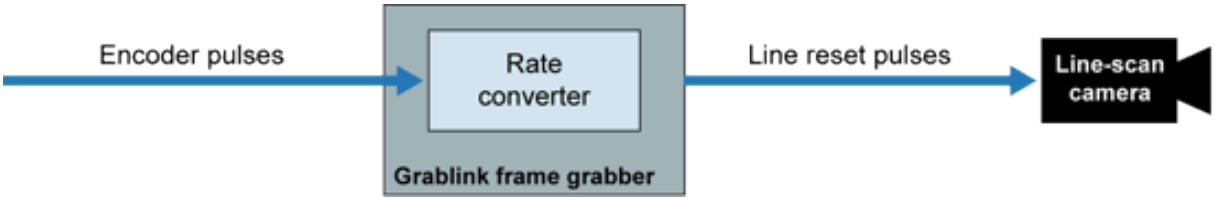
The hardware line used is selected with MultiCam parameter **LineTrigLine**, and the style of the signal applied to this line can be chosen among a set of variants.

Also the MultiCam parameter **LineTrigFilter** is used to remove any parasitic electrical noise that could alter the shape of the system trigger pulse.

The system device generating the system trigger pulse should be such that the maximum line frequency is never reached.

Rate Converter-Based Triggering

This mode is effective when the MultiCam parameter **LineRateMode** is set to **CONVERT**.



Rate converter-based triggering

The hardware line used is selected with MultiCam parameter **LineTrigLine**. The MultiCam parameters **LineTrigCtl** and **LineTrigFilter** are used to select the appropriate signal style and filter.

However, in the rate converter mode, the trigger pulses sent towards the frame grabber are not a copy of the system trigger pulses applied to the frame grabber.

The Grablink frame grabber includes a special device called a "rate converter".

The rate (or frequency) of system pulses sensed at the selected hardware line is converted into a pulse train towards the camera at a different rate (or frequency).

The system trigger rate and the camera line trigger rate are electronically maintained in a constant ratio. This ratio can be programmed to suit the application needs.

PART II
MULTICAM BASICS

1. MultiCam as a Driver

Multiple Grabber Servicing

A grabber is a set of hardware resources able to handle the task of grabbing a frame from a camera. A board can incorporate one or several simultaneous grabbers.

A computer equipped with several frame grabbers can represent a fairly high number of grabbers. Up to eight boards can be used simultaneously in MultiCam. Only one MultiCam driver must be installed in this computer to give an application control over all the grabbers.

This application can freely select a specific grabber in a specific board and create a channel acquisition object using this grabber.

A channel is a set of hardware and software resources able to condition and transport the image from the camera into a PC memory surface. Once created, the channel is effectively perceived by the user as a uniform image acquisition chain controlled by its own set of MultiCam parameters.

The way the channel is operated is largely independent of the board effectively hosting the grabber, greatly simplifying the learning curve in case of new boards or new cameras.

Multiple Application Servicing

Several applications can be launched, each of them having the ability to interact with the common MultiCam driver installed in the host computer.

Any application can create as many channels as needed, implying the following resources:

- Any MultiCam compliant board within the system
- Any grabber within the selected board
- Any camera connected to the selected grabber
- Any memory surface in the computer memory

In addition, memory surfaces can be shared by several channel, and cameras can feed several channels.

Connecting and Disconnecting the Driver

Before using any MultiCam function, the communication between the application process and the MultiCam driver should be established. This is done with a C language API function `McOpenDriver`.

Before terminating the application, the user should terminate the communication of the application process with the MultiCam driver. This is done with a C language API function `McCloseDriver`.

Example

//Connecting to driver

```
MCSTATUS Status = McOpenDriver(NULL);
```

//...
//Application code
//...

//Disconnecting from driver

```
Status = McCloseDriver();
```

The `Status` variable can be used for error checking.

2. Parameters

A MultiCam parameter is a **control item**, its **value** can be written and read by the software application.

A **"set"** function writes a value into a parameter. This results into a predictable modification of some hardware or software aspect of the system.

A **"get"** function reads the value of a parameter. This is a way to obtain information on the present state of some hardware or software aspect of the system.

All MultiCam parameters are named objects. The parameters serving a similar goal are grouped in classes. A parameter name is unique within a given class.

2.1. Parameter Types

A parameter holds a value, and MultiCam requires the **type** of this value to be consistently known and used.

Parameter types

Type	Description
Integer	An integer parameter is coded as a 32-bit signed integer value.
Floating-point	A floating-point parameter is coded as a 64-bit floating-point value.
String	A string parameter is an ordered set of ASCII characters.
Enumerated	An enumerated parameter holds a particular value chosen among a set of known possibilities.
Instance	An instance parameter is able to host a handle designating the instance of a class object.

Some MultiCam parameters exist as a **collection**. This applies to any parameter type.

A collection parameter is a container able to host several values of the same type. In other words, it provides a way to reach several parameters of the same type using a single parameter name.

Collection parameter types

Type	Description
Integer collection	Set of integer parameters
Floating-point collection	Set of floating-point parameters
String collection	Set of string parameters
Enumerated collection	Set of enumerated parameters
Instance collection	Set of instance parameters

2.2. Parameter Name and Parameter Identifiers

Each MultiCam parameter has a **name**, that is a string of alphanumeric characters, without space, freely including letters, figures and a few special characters. Conventionally, a parameter name uses upper- and lower-case letters to improve the readability. Nevertheless, the MultiCam database is not case-sensitive.

Each MultiCam parameter has also (language-dependent) **identifiers** that are defined in the header file `McParams.h`. This file is included by the `MultiCam.h` header file. The header files are maintained by Euresys and are part of the MultiCam releases. The parameter identifiers are built from the parameter name, according to constant rules.

Moreover, each parameter has also a numerical identifier that is an integer value unambiguously designating the parameter in MultiCam.

Example: BoardTopology name and identifiers

String	**BoardTopology**
C, C++	`MC_BoardTopology`
.NET	`MC.BoardTopology`
Numerical	59

In the same way, each possible **value of an enumerated parameter** has a name and an identifier.

Example: some possible values of the enumerated parameter BoardTopology

Name	Identifier
MONO	`MC_BoardTopology_MONO`
DUO	`MC_BoardTopology_DUO`

2.3. By-Name vs. By-Identifier Access

Two ways are provided to refer to parameters and enumerated parameter values:

- The **by-identifier** access (also called the **by-ident** access or **by-handle** access). The items are referred to by their identifier.
- The **by-name** access. The items are referred to by their name (a character string).

See also Parameter Name and Parameter Identifiers.

Advantages and disadvantages of using by-ident access or by-name access

By-ident access	**By-name access**
Static method All item identifiers are prepared offline by Euresys, and made known to the user application source code by means of a special header file.	**Dynamic method** The reference to an item can be dynamically established at runtime. For example, a dialog box can invite the user to type an item name.
Syntax checking The compiler can check the consistency of the identifier spelling.	**No syntax checking** The compiler has no way to control the correctness of a quoted character string.
Faster The MultiCam driver directly accesses the required item when the access function is executed. This can be valuable when a lot of items have to be consecutively updated under application control.	**Slower** The MultiCam driver has to interpret the string when the parameter access function is executed. This involves a search in a list of names, and consumes time at run-time.

2.4. Parameter Levels

With MultiCam, any hardware or software feature that could profitably benefit of some user controllability is associated to a parameter.

Some of these features are related to the fundamental behavior of the system. There are controlled by **high-level** parameters. As a general rule, the user acts on these high-level parameters in order to implement the expected behavior.

Other features are controlled by **low-level** parameters. These parameters will be exercised by the experienced user to fine-tune some functional aspects of the application.

Any parameter possesses one of the following three levels.

Parameter level	Definition
Select	The parameter addresses a **fundamental feature** of the hardware or software behavior.
Adjust	The parameter addresses an operative mode for the fundamental feature specified by the select-level parameters.
Expert	The parameter can be used by experienced users to modify the operative mode of the fundamental feature ruled by the higher-level parameters.

An application is able to change the value of any parameter regardless to its level.

2.5. Inference Rules

The **Channel** class parameters exhibit a powerful behavior.

The value of any channel parameter is made dependent on one or several higher-level channel parameters. This means that each time the user changes the value of a parameter at any level, all parameters that are designed to depend on the modified parameter are automatically and instantly updated.

A sophisticated **inference rule** is associated to any parameter that has to be automatically adjusted.

A parameter without an inference rule is called an **entry parameter**. It should receive a value from a deliberate setting action. The highest-level parameters are entry parameters.

A parameter controlled by an inference rule is called a **ruled parameter**. Its value is automatically updated according to values of other parameters, but it can still receive at any time a value from a deliberate setting action.

The set of all inference rules implements the powerful intelligence of the MultiCam system.

2.6. Code Examples for Parameters Management

How to Set an Enumerated Parameter?

The following code sets the **Camera** enumerated parameter to the value **CAM2000**. The parameter as well as the value can be both accessed by identifier and by name. So, there are four ways to proceed:

[C]

//Setting the parameter using the by-ident parameter
//and by-ident enumerated method

```
Status = McSetParamInt(ObjectHandle, MC_Camera, MC_Camera_CAM2000);
```

//Setting the parameter using the by-name parameter
//and by-ident enumerated method

```
Status = McSetParamNmInt(ObjectHandle, "Camera", MC_Camera_CAM2000);
```

//Setting the parameter using the by-ident parameter
//and by-name enumerated method

```
Status = McSetParamStr(ObjectHandle, MC_Camera, "CAM2000");
```

//Setting the parameter using the by-name parameter
//and by-name enumerated method

```
Status = McSetParamNmStr(ObjectHandle, "Camera", "CAM2000");
```

How to Get a Collection Parameter?

The following code gets the second element of the **SignalEvent** integer collection parameter into an integer variable. The collection parameter can be accessed by identifier and by name. So, there are two ways to proceed:

[C]

//Declaring an integer variable

```
INT32 MyEvent;
```

//Getting the element using the by-ident method

```
Status = McGetParamInt(ObjectHandle, MC_SignalEvent+1, & MyEvent);
```

//Getting the element using the by-name method

```
Status = McGetParamNmInt(ObjectHandle, "SignalEvent:1", & MyEvent);
```

3. Classes

A MultiCam class is a container for MultiCam parameters.

Each class is divided into categories, containing sets of MultiCam parameters serving a common goal.

Those categories and parameters are fully described in the Parameters Reference Manual

MultiCam classes

Class	Definition
`Configuration`	The configuration parameters control the common features of the MultiCam system.
`Board`	The board parameters control the common features of each board.
`Channel`	The channel parameters define and control every individual acquisition path.
`Surface`	The surface parameters define every image container defined inside the MultiCam system.

Creating an object is equivalent to instantiating a class. Creating an object of a specified class has the following effects:

- A new object belonging to the class effectively exists.
- This object has its own designating handle.
- An additional set of parameters owned by the object is created.

One board object (a single instance of the `Board` class) exists for each Euresys board installed inside the host computer. One configuration object (a single instance of the `Configuration` class) exists within the host system. The board and configuration objects cannot be created by a user application. They natively exist when the application connects itself to the MultiCam driver.

The `Channel` and `Surface` classes can be instantiated by a user application as many times as needed, creating one or several objects sharing the common characteristics of the class. For example, the channel object will be created for every acquisition path that the user needs in the vision system to handle his application. For each channel instance, the application gets an independent set of control parameters.

3.1. Code Examples for Objects Management

How to Create and Delete Channels

The following code shows how to create two MultiCam channels for a 1623 Grablink DualBase board with driver index 0, using by-handle and by-name methods. At the end, after application code, both channels are deleted. The `Status` variable can be used for error checking.

[C]

```
//Connecting to driver
MCSTATUS Status = McOpenDriver(NULL);
//Instantiating a first channel and associating it with connector A using the by-
handle method
MCHANDLE MyChannel1;
Status = McCreate(MC_CHANNEL &MyChannel1);
Status = McSetParamInt(MyChannel1, MC_DriverIndex, 0);
Status = McSetParamInt(MyChannel1, MC_Connector, MC_Connector_A);
//Instantiating a second channel and associating it with connector B using the by-name
method
MCHANDLE MyChannel2;
Status = McCreateNm("CHANNEL" &MyChannel2);
Status = McSetParamNmInt(MyChannel2, "DriverIndex", 0);
Status = McSetParamNmStr(MyChannel2, "Connector", "B");
//...
//Application code
//...
//Deleting the channels
Status = McDelete(MyChannel1);
Status = McDelete(MyChannel2);
//Disconnecting from driver
Status = McCloseDriver();
```

How to Create and Delete Surfaces

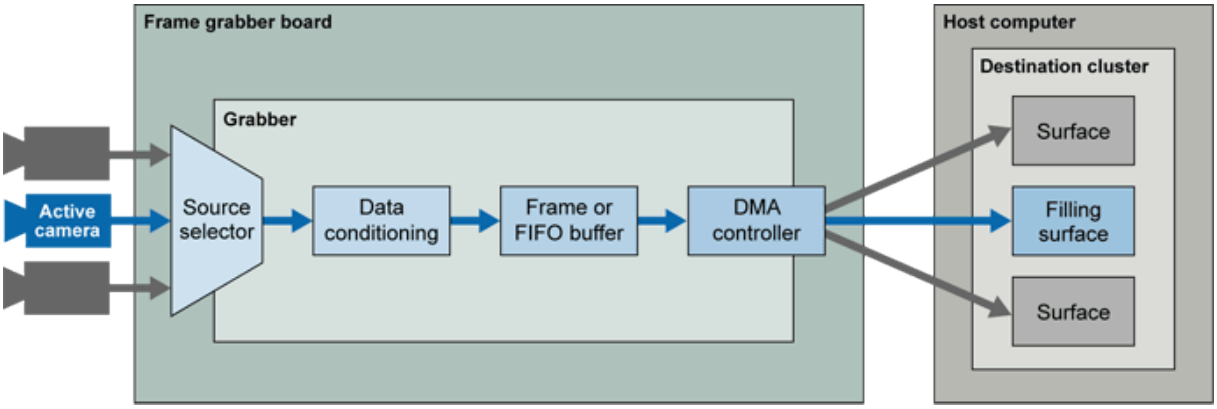
The following code shows how to create three MultiCam surfaces. At the end, after application code, the surfaces are deleted. The `Status` variable can be used for error checking.

[C]

```
//Connecting to driver
MCSTATUS Status = McOpenDriver(NULL);
//Instantiating a first surface
MCHANDLE MySurface1;
MCSTATUS Status = McCreate(MC_DEFAULT_SURFACE_HANDLE, &MySurface1);
//Instantiating a second surface
MCHANDLE MySurface2;
Status = McCreate(MC_DEFAULT_SURFACE_HANDLE, &MySurface2);
//Instantiating a third surface
MCHANDLE MySurface3;
Status = McCreate(MC_DEFAULT_SURFACE_HANDLE, &MySurface3);
//...
//Application code
//...
//Deleting all surfaces
Status = McDelete(MySurface1);
Status = McDelete(MySurface2);
Status = McDelete(MySurface2);
//Disconnecting from driver
Status = McCloseDriver();
```

4. Acquisition

4.1. Overview of a Simplified Acquisition Model



Simplified model for an acquisition chain

The **grabber** can be a part of the frame grabber in case of boards supporting multiple cameras simultaneously. Boards with simpler architecture embed a single grabber.

As represented in the picture, a grabber can be sourced by a set of cameras, among which only one is active at a time. The source selector is a programmable device that establishes the connection to one of the incoming cameras. This is typical of the **switched acquisition** situation.

Some grabbers can only be connected to a single camera, whereas there may be other grabbers in the board connected to other cameras. This is typical of the **concurrent acquisition** situation.

The image data extracted out of the active camera is processed by a set of grabber-specific devices. Refer to the frame grabber documentation for details.

Generally speaking, the rationale for the acquisition path is to leave in some part of the host computer memory a digital representation of the image produced by the camera. The format of the stored image is directly usable by the application software.

The MultiCam system provides a specific object to represent a memory buffer able to receive an image. This object is called the **surface**, which is an instance of the Surface class, and owns a set of defining parameters.

A high degree of flexibility is provided for implementing multiple-buffer image destination structures. To achieve this, the surfaces are grouped into a **cluster** of surfaces.

A **channel** is the temporary association of a grabber connected to a camera delivering data to a destination cluster. Each channel should be linked to a cluster. The channel is able to transport an image from the camera towards one of the surfaces constituting its cluster. In the above picture, the surface receiving the image is noted as "Filling surface".

4.2. Acquisition Phase

An **acquisition phase** is the constitutive element of the **acquisition sequence**.

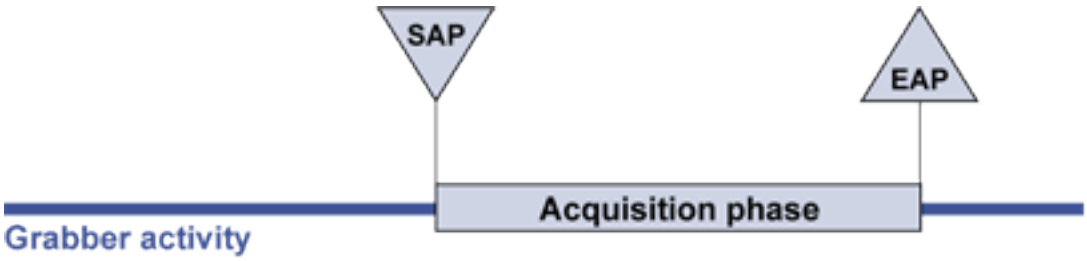
The rationale for an acquisition phase is to fill one surface of the destination cluster.

To achieve this result, the channel locks the use of a set of hardware resources for its own benefit, preventing any other channel to use these resources at the same time. By definition, this set of hardware resources (connector, selector, data conditioner, timing generator...) is collectively called a **grabber**.

- In case of normal speed **area-scan** cameras, the image data provided by the acquisition phase is a **frame**. For high-speed area-scan cameras, multiples frames are stored in a single surface during an acquisition phase.
- In case of **line-scan**, the image data provided by the acquisition phase is called a **page**. The number of lines constituting a page is chosen by the **PageLength_Ln** parameter.

An acquisition phase starts at **SAP** event (Start of Acquisition Phase). It occurs either at TE (Trigger Event), or automatically according to the chosen acquisition mode.

When completing the acquisition phase, the grabber hardware issues an event called **EAP** (End of Acquisition Phase). The usage of the EAP event is programmable.



Acquisition phase

For more information, refer to the MultiCam Acquisition Principles application note.

4.3. Acquisition Sequence

An **acquisition sequence** is a succession of **acquisition phases**.

The successive acquisition phases are not necessarily contiguous in time. Intervening gaps of various duration can be present between acquisition phases.

An acquisition sequence starts at **SAS** event (Start of Acquisition Sequence). It occurs either at TE (Trigger Event), or automatically according to the chosen acquisition mode.

When completing the acquisition sequence, the channel issues an event called **EAS** (End of Acquisition Sequence).

Acquisition sequence

For more information, refer to the MultiCam Acquisition Principles application note.

Speeding-Up the Start of Acquisition Sequence

Normally, starting an acquisition sequence involves no delay. The time interval between the software activation and the effective SAS event is less than one millisecond.

However, under certain circumstances, the activation time can be longer (several tens of milliseconds).

The circumstances for a slowed down channel activation are as follows:

- First activation after channel creation
- One or several format-related MultiCam parameter have been updated

A format-related parameter is a channel- or surface-class parameter the update of which induces a change in the format characteristics of the destination surface. Specifically, any change of size, bit-width or plane structure in the acquisition or processing destination surface will be taken into account at the next channel activation, and this will take a noticeable amount of time.

MultiCam provides a way to speed up the channel activation to alleviate this restriction.

When a channel is in the **IDLE** state, as reflected by the channel parameter **ChannelState**, the application software is allowed to set this parameter to the pseudo-state **READY**.

In doing this, the MultiCam system will try to make ready everything inside the channel in such a way that the next time the **ChannelState** parameter is set to **ACTIVE**, the channel activation is immediate (less than one millisecond).

Setting **ChannelState** to **READY** takes a short time to complete. The channel's state changes to **READY** if all required grabber resources are available. Otherwise, **ChannelState** remains **ORPHAN**.

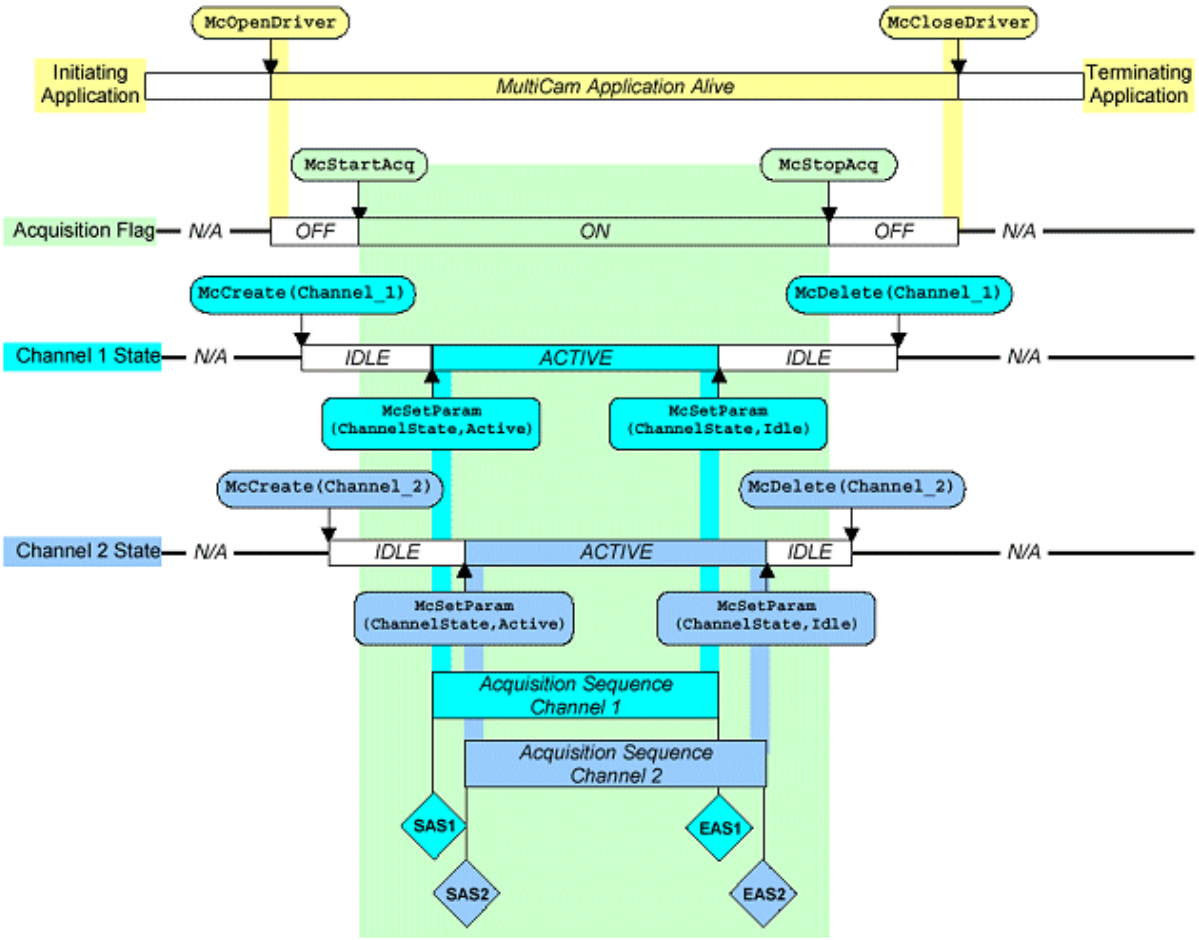
For more information, see also Understanding Automatic Switching.

4.4. Acquisition Method

To be able to perform an image acquisition, a given application should create at least one channel. A given application can create as many channels as required.

The usual method is the channel-specific acquisition method where each channel is controlled individually.

The following diagram shows how to control the acquisition sequence individually per channel defined inside the application.



Control the acquisition sequence individually

About switched acquisitions, an automatic switching mechanism is also available in the channel-specific acquisition method. It allows an optimized time sharing of grabber resource due to multiple channel acquisitions. For more information, see Understanding Automatic Switching.

4.5. Multiple Acquisition Buffer Management

Cluster of Surfaces

A cluster of surfaces is associated to any channel object. The surfaces are chosen among a pool of surfaces previously instantiated by the application. A large number of surfaces can be registered to a cluster.

The cluster linked to a channel is called the **destination cluster** of the channel.

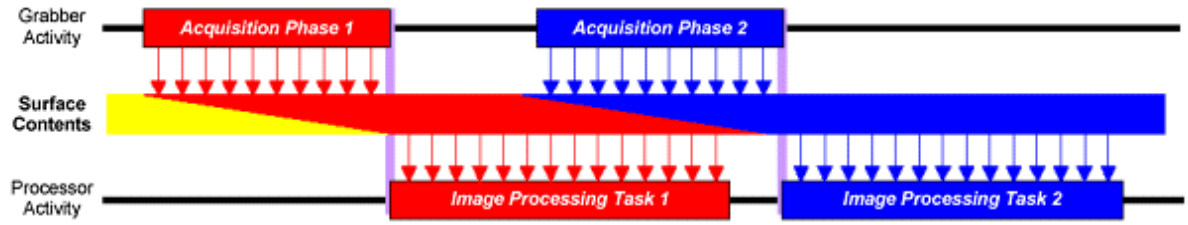
Several channels can designate the same surface as a member of their destination clusters.

The rationale for the cluster is to allow for an unified management scheme when the user wishes to implement a multiple-buffer acquisition structure.

The cluster mechanism offered by MultiCam unifies the control mechanism for a single, double or triple buffer, as well as the acquisition of image sequences implying more than three buffers.

Single Buffer Acquisition

If the user chooses to define a single surface cluster, the surface will be written to by the grabber at each new acquisition phase, irrespective of any possible processing task applied to the image contained in the buffer. This can be satisfactory for simple applications.



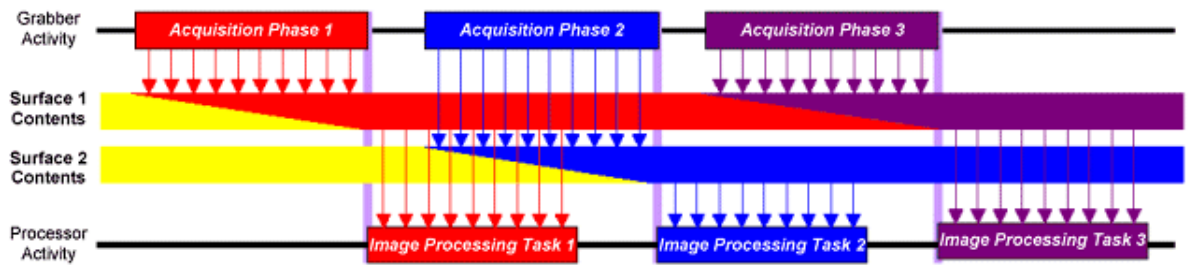
Single buffer acquisition

The image processing task 1 operates on data from image issued at acquisition phase 1 and at acquisition phase 2.

This adverse effect is called **surface alteration**. This can be acceptable or not depending on the application.

Double Buffer Acquisition

To improve the chance of processing an unaltered image buffer, the user may wish to use the **double buffer** technique. He simply chooses to define a cluster with two surfaces, which automatically implements this technique.



Double buffer acquisition

The technique is useful as long as the processing task is shorter than the period separating successive acquisition phases.

In the case of a processing task taking more time, there still exists a risk of surface alteration sourced by two distinct acquisition phases.

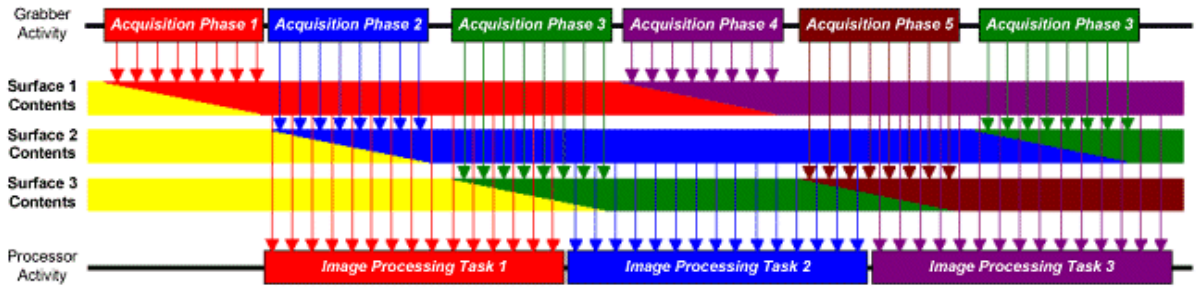
Triple Buffer Acquisition

The radical way to guarantee an unaltered surface in the cluster for robust image processing is to use a **triple buffer** scheme.

If a processing task operates for a fairly long time, the more probable condition will be as follows:

- One surface is being processed.
- Another surface is experiencing acquisition.
- The third one is ready for processing after the latest completed acquisition phase.

As soon as the processing task is finished on a specific surface, the cluster immediately submits another surface to the processor with the most recently acquired full image. This method maximizes the host processor occupation, while providing the optimal way to schedule acquisition.



Triple buffer acquisition

The surface alteration phenomenon is suppressed.

Image Sequence Acquisition

The user is allowed to create as many surfaces as wanted, and to include them to the destination cluster of a channel.

The cluster management mechanism offered by MultiCam will sequentially store successive images into the set of surfaces provided that way.

Registering Surfaces to the Destination Cluster

Registering a surface to the destination cluster of a channel is done with the MultiCam channel parameter **Cluster**.

This parameter is an instance collection giving access to the handles of the surfaces belonging to the destination cluster.

The application should use the by-identifier parameter access method with consecutive numeric values starting at **MC_Cluster**, followed by **MC_Cluster+1**, **MC_Cluster+2**, etc. Each item of the **Cluster** parameter is sequentially loaded with the handle of the constitutive surface, until the last surface.

It is not allowed to use an alternative registration order.

Surface Index

Registering the surfaces to the destination cluster implicitly assigns a numeric zero-base index to each surface of the channel. This index is called the *surface index*. This index can be seen as the serial number of the registered surface, or equivalently as the displacement inside the collection parameter. The surface index is made available to the application through the MultiCam channel parameter **SurfaceIndex**.

Obviously, all the surfaces have to be created beforehand.

It is allowed to assign the same surface to more than one cluster belonging to different channels.

Refer to the MultiCam release notes for restriction on the capability of multiple assignment of a surface to several clusters, and to the maximum number of surfaces that can be created.

4.6. Cluster Mechanism

Description of the MultiCam surface cluster mechanism

Surface States

To implement the cluster mechanism, MultiCam uses the state of the surface, which is available through the MultiCam parameter **SurfaceState**.

Any instantiated surface is necessarily in one of the five following states:

SurfaceState	Description
FREE	The surface is unconditionally able to receive image data from the grabber.
FILLING	The surface is presently receiving or ready to receive image data from the grabber.
FILLED	The surface has finished receiving image data from the grabber, and thus is ready for processing.
PROCESSING	The surface is being processed by the host processor.
RESERVED	The surface is removed from the standard state transition mechanism.

The state of the surface is unique in the sense that, at a given instant, a surface belonging to several clusters is perceived in a consistent state by all associated channels.

Cluster State

The cluster mechanism description uses an internal state associated to the cluster. The destination cluster of a channel is necessarily in one of the four following states:

Cluster state	Meaning
OFF	The cluster if **OFF** when the acquisition is not alive.
READY	The cluster is **READY** when the acquisition is alive, no surface is **PROCESSING**, and the cluster can accept a new acquisition.

Cluster state	Meaning
BUSY	The cluster is **BUSY** when the acquisition is alive and one of its surfaces is **PROCESSING**.
UNAVAILABLE	The cluster is **UNAVAILABLE** when the acquisition is alive but the cluster cannot accept a new acquisition. This is an exceptional situation.

The goal of the cluster mechanism is to make available to the application an unalterable surface for image processing. The callback mechanism offers a separate thread to realize the image processing task.

As long as the callback thread is processing a surface, this surface is in the state **PROCESSING**. As the thread is unique, no other surface can be processed at this time.

When the processing callback function is in operation, the cluster is said to be **BUSY**. Otherwise, the cluster is said to be **READY**. In other words, the cluster is **BUSY** when one of its surfaces is **PROCESSING**.

Surface Allocation Rules

This section summarizes all the rules applicable to the allocation of MultiCam surfaces in the host PC memory for Grablink Base, Grablink DualBase, and Grablink Full exclusively.

MultiCam Limitations

The MultiCam driver exhibits the following limitations concerning the number of surfaces:

- The maximum number of surfaces instantiated within an application is **4096**.
- The maximum number of surfaces assigned to a channel is **4096**.
- The maximum number of surfaces per board is **4096**.

Dynamic DMA

Note: *On 1623 Grablink DualBase, when using two channels, a practical limit would be 2048 surfaces per channel.*

Operating System Limitations

The Windows operating systems exhibit the following limitations for the maximum buffer size allowed per MultiCam surface:

- About **64 Megabytes** under Windows XP x86;
- About **32 Megabytes** under Windows XP x64;
- About **2 Gigabytes** under Windows Vista and Windows Server 2008;
- About 4 **Gigabytes** under Windows 7 and Windows Server 2008 R2.

Note: *If a MultiCam surface exceeds those limits, MultiCam returns **MC_IO_ERROR** at channel activation.*

Board Resources Limitations - Dynamic DMA

Applies to: Base DualBase Full FullXR

These Grablink boards implement "dynamic DMA" meaning that, if required, the descriptors lists are stored in the host PC memory.

When this happens, prior to the transfer of any surface into the host PC memory, the MultiCam driver copies the corresponding descriptors list into the on-board memory reserved for that purpose.

The "dynamic DMA" feature relaxes the constraints about the total amount of descriptors for all the surfaces belonging to the channel. However, the descriptors of one surface must fit within the on-board being board dependent. Consequently, the rule becomes:

- For 1622 Grablink Full and 1626 Grablink Full XR, the maximum number of descriptors per surface is about **4,000,000**.
- For each channel of 1623 Grablink DualBase, the maximum number of descriptors per surface is about **2,000,000**.
- For 1624 Grablink Base, the maximum number of descriptors per surface is about **2,000,000**.

The number of descriptors required for a surface is depending on multiple factors:

- The fragmentation of the memory allocated by the operating systems; usually, the memory is allocated by non-contiguous blocks of 4096 bytes forcing a new descriptor for every block boundary.
- The alignment of the MultiCam surface on the memory blocks.
- The usage of the **ImageFlipY** function requires a descriptor boundary at each line boundary.
- The usage of RGB planar formats requires a descriptor boundary at each line boundary for each color component.

A practical, but very conservative, rule to estimate the number of descriptors for a surface is:

$$N = \text{SurfaceSize [Bytes]} * \left(\frac{1}{\text{BlockSize [Bytes]}} + \frac{1}{\text{LineSize [Bytes]}} \right)$$

with BlockSize = 4096 Bytes.

> **Note:** *The rule used to decide to enable "dynamic DMA" is based on the above formulae. If the descriptors for all surfaces cannot fit in the on-board memory, the dynamic DMA is activated.*

Board Resources Limitations - Cropping in Hardware

Applies to: Base DualBase Full FullXR

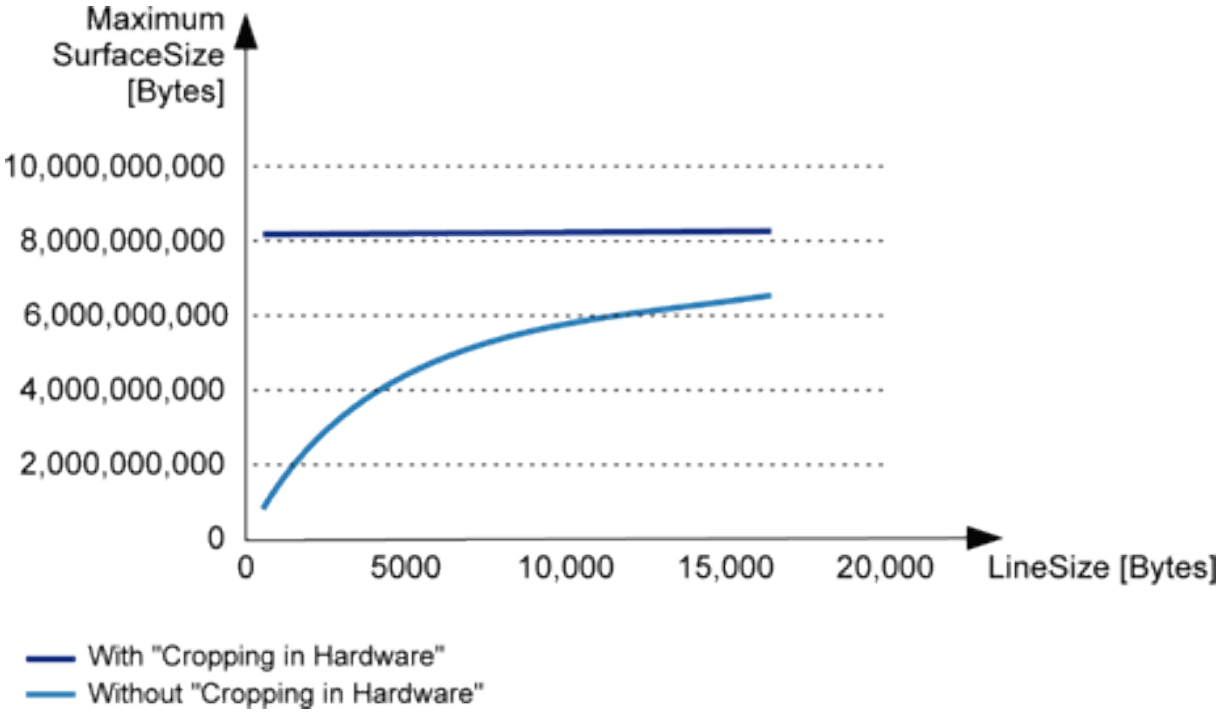
These Grablink boards activates the "Cropping in Hardware" feature to avoid descriptor boundaries at line boundaries when following conditions are met:

- Monochrome or packed RGB image formats
- **ImageFlipY** is **OFF**

When "Cropping in Hardware" is active, the following rule is applicable:

$$N = \frac{\text{SurfaceSize [Bytes]}}{\text{BlockSize [Bytes]}}$$

Note: *"Cropping in Hardware" is particularly useful for small line sizes. The following diagram shows the maximum SurfaceSize vs. LineSize for a channel having a capacity of 2,000,000 descriptors (Grablink Base or Grablink DualBase).*



Maximum SurfaceSize vs. LineSize

Note: *This rule is often less restrictive than the OS rule.*

Controlling Events

The state of a surface is modified according to events received by the cluster mechanism. The following events are relevant:

Event	**Meaning**
Creation	This event is issued when a surface is instantiated.
SAS	This event marks the beginning of the acquisition sequence.
EAP	This event marks the end of any acquisition phases.
Exit Callback	This event is generated when the channel-proprietary callback function returns.

Event	Meaning
Cluster Ready	The state of the cluster determines the surface state transition towards **PROCESSING**.
Software Action	This kind of event corresponds to a deliberate user action on the **SurfaceState** parameter.

Sourced Signals

The cluster mechanism issues three MultiCam signals that can be used by the regular signaling mechanisms:

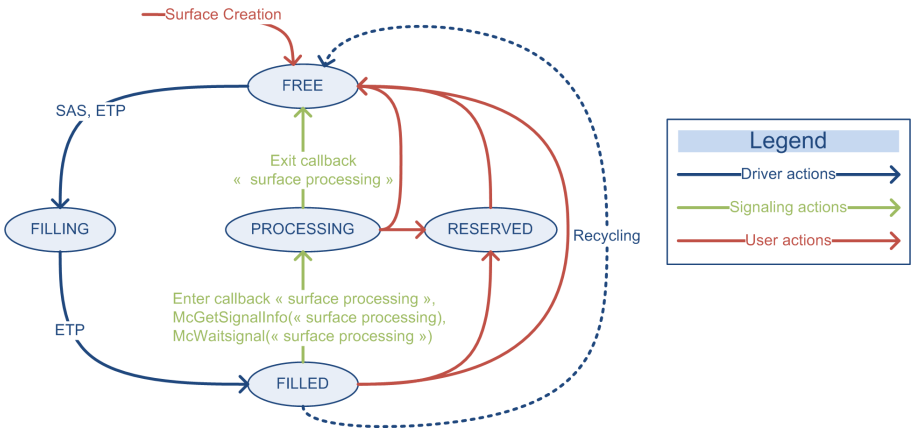
Signal	Meaning
Surface Processing	This signal is issued when a surface of the destination cluster enters the state **PROCESSING**.
Surface Filled	This signal is issued when a surface of the destination cluster enters the state **FILLED**.
Cluster Unavailable	This signal is issued when the acquisition phase terminates without having been assigned to a destination surface.

State Diagram

State origin	State destination	Initiator	Occurrence and applicability
Not applicable	**FREE**	User application	On creation of a new surface.
FREE	**FILLING**	MultiCam driver	On Start Acquisition Sequence and End of Transfer Phase events: • When **MaxFillingSurfaces** = **MAXIMUM**: applies to all (up to 512) FREE surfaces in the cluster • Otherwise applies to only one FREE surface
FILLING	**FILLED**	MultiCam driver	On End of Transfer Phase event: • When **MaxFillingSurfaces** = **MAXIMUM**: applies to all (up to 512) FILLING surfaces in the cluster that have finished receiving image data from the grabber, and thus are ready for processing Otherwise applies to the unique FILLING surface

State origin	State destination	Initiator	Occurrence and applicability
FILLED	**PROCESSING**	User application or Operating System	Applies to the oldest FILLED surface (if any) when the cluster contains no more PROCESSING surface and ... • on Execution of the `McGetSignalInfo` function or • on release (exit) of the `McWaitSignal (SurfaceProcessing)` function or • on entry of the "Surface Processing" callback function.
PROCESSING	**FREE**	User application or Operating System	Applies to the unique PROCESSING surface (if any) when: • setting the parameter **MC_SurfaceState** to the value **MC_SurfaceState_FREE** or • automatically when exiting the "Surface Processing" callback function.
FILLED **RESERVED**	**FREE**	User application	Applies to any FILLED or RESERVED surface when setting the parameter **MC_SurfaceState** to the value **MC_SurfaceState_FREE**.
FILLED **PROCESSING**	**RESERVED**	User application	Applies to any FILLED or PROCESSING surface when setting the parameter **MC_SurfaceState** to the value **MC_SurfaceState_RESERVED**.
FILLED	**FREE**	MultiCam driver	Applies to the oldest FILLED surface (if any) when the cluster contains no more FREE surfaces

The following drawing shows a simplified state transition diagram applying to any surface in the cluster:



Surface state diagram

For a cluster having a total of N registered surfaces:

- 0 up to N surfaces can be in the FREE state
- 0 up to N (limited to 512) surfaces can be in the FILLING state when the parameter **MaxFillingSurfaces** = **MAXIMUM**
- 0 or 1 surface can be in the FILLING state when the parameter **MaxFillingSurfaces** = **MINIMUM**
- 0 up to N surfaces can be in the FILLED state
- 0 or 1 surface can be in the PROCESSING state
- 0 to (N-2) surfaces can be in the RESERVED state

  Note: *There is at most one surface in the PROCESSING state per cluster!*

  Note: *At least 2 surfaces should be left outside the RESERVED state to maintain a minimal operability of the cluster mechanism.*

Filling Index

The cluster mechanism uses a internal variable called the filling index. During a transfer, this variable designates the surface of the cluster which is currently receiving an image. Otherwise, it designates the surface which has been elected to receive the next image.

In other words, when the acquisition sequence is alive, the filling index designates the **FILLING** surface in the cluster.

The designation uses the zero-based surface index associated to each surface of the cluster.

Next Index Evaluation

After each acquisition, the cluster mechanism invokes an internal evaluation function which determines the index of the next surface to be elected for acquisition.

MultiCam looks at each surface in the cluster, starting with the current filling index and wrapping-around to zero, and selects the first **FREE** surface it finds.

If it cannot find a **FREE** surface in the cluster, MultiCam tries to recycle a **FILLED** surface: if it can find one or several **FILLED** surfaces, the oldest one becomes **FREE** (it is recycled) and is used as the destination for the next acquisition. If it cannot recycle a surface, the acquisition is skipped.

In some exceptional situation, the evaluation may fail to find a surface suitable for election. In this case, the signal Cluster Unavailable is issued.

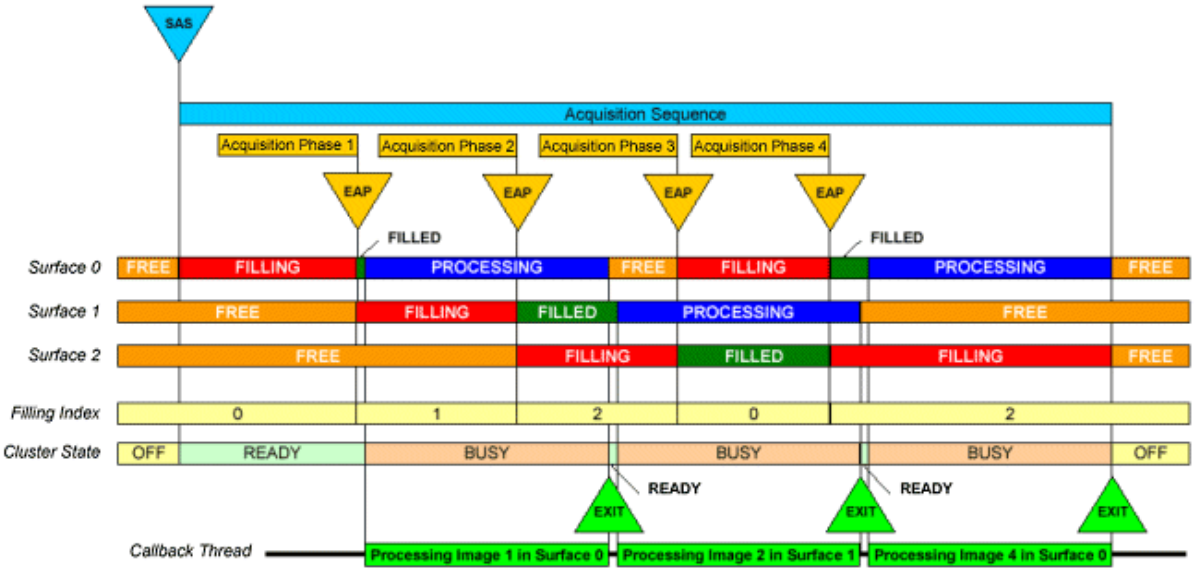
Manual selection of the destination surface

Parameter **SurfaceIndex** allows an application to select the next filling index, i.e., the index of the next destination surface.

Manipulating this parameter is usually unnecessary and should only be used as an expert setting.

Timing Diagram for Triple Buffer

The following diagram shows the details of the cluster mechanism operation when the triple buffer technique is used.



Timing diagram for triple buffer

4.7. Acquisition Code Sample

The following sample code manages the acquisition of two channels with a 1623 Grablink DualBase frame grabber with driver index 0. The `Status` variable can be used for error checking.

[C]

```
//Connecting to driver
MCSTATUS Status = McOpenDriver(NULL);
//Instantiating first channel
MCHANDLE MyChannel1;
Status = McCreateNm("CHANNEL", &MyChannel1);
Status = McSetParamInt(MyChannel1, MC_DriverIndex, 0);
Status = McSetParamInt(MyChannel1, MC_Connector, MC_Connector_A);
//Assign grabber and camera to first channel
//Configure first channel including triggering mode
//Instantiating second channel
MCHANDLE MyChannel2;
Status = McCreateNm("CHANNEL", &MyChannel2);
Status = McSetParamInt(MyChannel2, MC_DriverIndex, 0);
Status = McSetParamInt(MyChannel2, MC_Connector, MC_Connector_B);
//Assign grabber and camera channel
//Configure second channel including triggering mode
//Activating acquisition sequence for first channel
Status = McSetParamInt(MyChannel1, MC_ChannelState, MC_ChannelState_ACTIVE);
//Acquisition sequence for first channel is now active
//...
//Application code
//...
//Acquisition sequence usually terminates naturally
//Activating acquisition sequence for second channel
Status = McSetParamInt(MyChannel2, MC_ChannelState, MC_ChannelState_ACTIVE);
//Acquisition sequence for second channel is now active
//...
//Application code
//...
//Acquisition sequence usually terminates naturally
//Deleting the channels
Status = McDelete(MyChannel1);
Status = McDelete(MyChannel2);
//Disconnecting from driver
Status = McCloseDriver();
```

5. Signaling

5.1. MultiCam Signals

What Is a MultiCam Signal?

A signal is an entity representing a particular event issued from a channel and able to interact with the application.

Three mechanisms are provided to synchronize the application operation with the MultiCam system.

Mechanism	Description
Callback signaling	Mechanism involving a user-written function automatically called when a pre-defined signal occurs.
Waiting signaling	Dedicated mechanism allowing for a thread to wait for the occurrence of a pre-defined signal.
Advanced signaling	User-defined mechanism involving standard Windows wait functions.

Signal Identifier

The MultiCam signals are not named with a character string as parameters are.

In the body of C functions used to interact with the MultiCam system, they are referred to by a **signal identifier** defined in the header file `MultiCam.h`.

Throughout the documentation, the MultiCam signals are designated by a descriptive text.

```
Example: Surface Processing
```

The signal identifier is build from the descriptive text according to the following rules.

- The descriptive text is prefixed with `MC_SIG_`
- Individual words of the descriptive text are separated by underscores

```
Example:MC_SIG_SURFACE_PROCESSING
```

The signal identifier is an integer value unambiguously designating the signal.

The MultiCam signals have the MultiCam-defined type `MCSIGNAL`.

List of MultiCam Signals Issued by Channels

Signal identifier	Description	Applicable Products
MC_SIG_ FRAMETRIGGER_ VIOLATION	*Frame Trigger Violation*. This signal is issued when a frame or page trigger has been detected too early for correct hardware handling.	Alert-e, AlertC-e, Alpha2, Value, Express, Base, DualBase, Full, FullXR
MC_SIG_START_ EXPOSURE	*Start exposure* This signal is issued at the beginning of the frame exposure condition.	Alpha2, Melody, Harmony, Symphony, Value, Express, Base, DualBase, Full, FullXR
MC_SIG_END_ EXPOSURE	*End exposure* This signal is issued at the end of the frame exposure condition.	Alpha2, Melody, Harmony, Symphony, Value, Express, Base, DualBase, Full, FullXR
MC_SIG_ RELEASE	*Release* This signal is issued when the object may be moved away from the camera, at the end of the exposure. With interlaced cameras, this signal is issued at the end of the second field's exposure.	Alpha2, Melody, Harmony, Symphony
MC_SIG_ SURFACE_ FILLED	*Surface filled* This signal is issued when a surface of the destination cluster enters the state **FILLED**.	All products
MC_SIG_ SURFACE_ PROCESSING	*Surface processing*. This signal is issued when a surface of the destination cluster enters the state **PROCESSING**.	All products
MC_SIG_ CLUSTER_ UNAVAILABLE	*Cluster unavailable* This signal is issued when the destination cluster is not able to receive the acquired data.	All products
MC_SIG_ ACQUISITION_ FAILURE	*Acquisition failure* This signal is issued when the channel acquisition time-out timer expires before the end of the acquisition phase. (*)	All products
MC_SIG_START_ ACQUISITION_ SEQUENCE	*Start of acquisition sequence* This signal is issued when the acquisition sequence begins.	All products

Signal identifier	Description	Applicable Products
MC_SIG_END_ ACQUISITION_ SEQUENCE	*End of acquisition sequence* This signal is issued when the acquisition sequence terminates.	All products
MC_SIG_END_ CHANNEL_ ACTIVITY	*End of channel activity* This signal is issued when the channel leaves the active state.	All products

(*) On Picolo boards, in case of a loss of video signal during an acquisition, the video digitizer does not provide the loss of signal information immediately, and the last acquired image may contain invalid data.

Enabling Signals

To designate one or several signals as responsible for signaling operation, the MultiCam system provides an adjust-level parameter called `SignalEnable`.

One such parameter exists for the channel class. It has the MultiCam type "enumerated collection".

Each item of the collection allows for enabling or disabling a specific signal. The value of the item is **ON** or **OFF**.

The set of all **ON** signals constitute the selection of signals enabling the relevant channel to perform one of the following:

- Calling a callback function
- Releasing a waiting thread
- Causing a Windows event

To address a specific signal, the by-ident parameter access method is used with the `SignalEnable` parameter belonging to the desired channel object. The parameter setting function `McSetParamInt` or `McSetParamStr` is used with a parameter identifier established as follows:

To reach signal...	Use parameter identifier...
Frame Trigger Violation	**MC_SignalEnable** + **MC_SIG_FRAME_TRIGGER_VIOLATION**
Start Exposure	**MC_SignalEnable** + **MC_SIG_START_EXPOSURE**
End Exposure	**MC_SignalEnable** + **MC_SIG_END_EXPOSURE**
Release (*)	**MC_SignalEnable** + **MC_SIG_RELEASE**
Surface Filled	**MC_SignalEnable** + **MC_SIG_SURFACE_FILLED**
Surface Processing	**MC_SignalEnable** + **MC_SIG_SURFACE_PROCESSING**
Cluster Unavailable	**MC_SignalEnable** + **MC_SIG_CLUSTER_UNAVAILABLE**
Acquisition failure	**MC_SignalEnable** + **MC_SIG_ACQUISITION_FAILURE**

To reach signal...	Use parameter identifier...
End of acquisition	**MC_SignalEnable** + **MC_SIG_END_ACQUISITION_SEQUENCE**
Start of acquisition	**MC_SignalEnable** + **MC_SIG_START_ACQUISITION_SEQUENCE**
End of channel activity	**MC_SignalEnable** + **MC_SIG_END_CHANNEL_ACTIVITY**

Example

The following code enables the "Surface Filled" signal with the channel designated by `my_Channel`:

```
Status = McSetParamInt (
  my_Channel,
  MC_SignalEnable + MC_SIG_SURFACE_FILLED,
  MC_SignalEnable_ON
);
```

The `Status` variable can be used for error checking.

Signal Information Structure

A dedicated C structure of the type `PMCSIGNALINFO` is specified to provide information on a specific MultiCam signal issued by a specific channel object.

The *signal information* type is declared in the MultiCam system `MultiCam.h` header file.

There are three usages:

- Usage with the callback function
- Usage with the waiting function
- Usage with the signal information retrieving function

The user-written code implementing a callback function is informed on the signal that caused the callback with an argument of the signal information type.

The waiting function returns the information of the thread releasing signal with an argument of the signal information type.

This signal information structure can be retrieved with the MultiCam function `McGetSignalInfo`, which specifies a target channel. In that case, the structure holds the information associated to the last signal issued by the target object.

Signal Info

`SignalInfo` is a member of the signal information structure available in the callback function or returned by the signal information retrieving function.

Its value only makes sense when the relevant signal is the Surface Processing or Surface Filled signal.

In that case, `SignalInfo` contains the handle of the surface that experienced the designated state transition, i.e. the surface presently in the state **PROCESSING** or **FILLED**.

This handle enable the user to retrieve the reference of the surface where the acquisition phase just stored the acquire image. The application can advantageously use the surface parameter **SurfaceContext** accessed using this surface handle.

Surface Context

A MultiCam integer parameter called **SurfaceContext** is available. This parameter is not written or read by the MultiCam driver. It is included for user's convenience.

Typically, the user assigns to this parameter an indexing information of its own. This assignment occurs after the surface instantiation and before the acquisition process.

After acquisition of a frame or a page, the `SignalInfo` member of the signal information structure directly designates the surface that experienced the acquisition. It is just a matter of getting the corresponding **SurfaceContext** parameter to get back the indexing information previously provided.

This greatly eases any surface filling sequence control the user may need to implement.

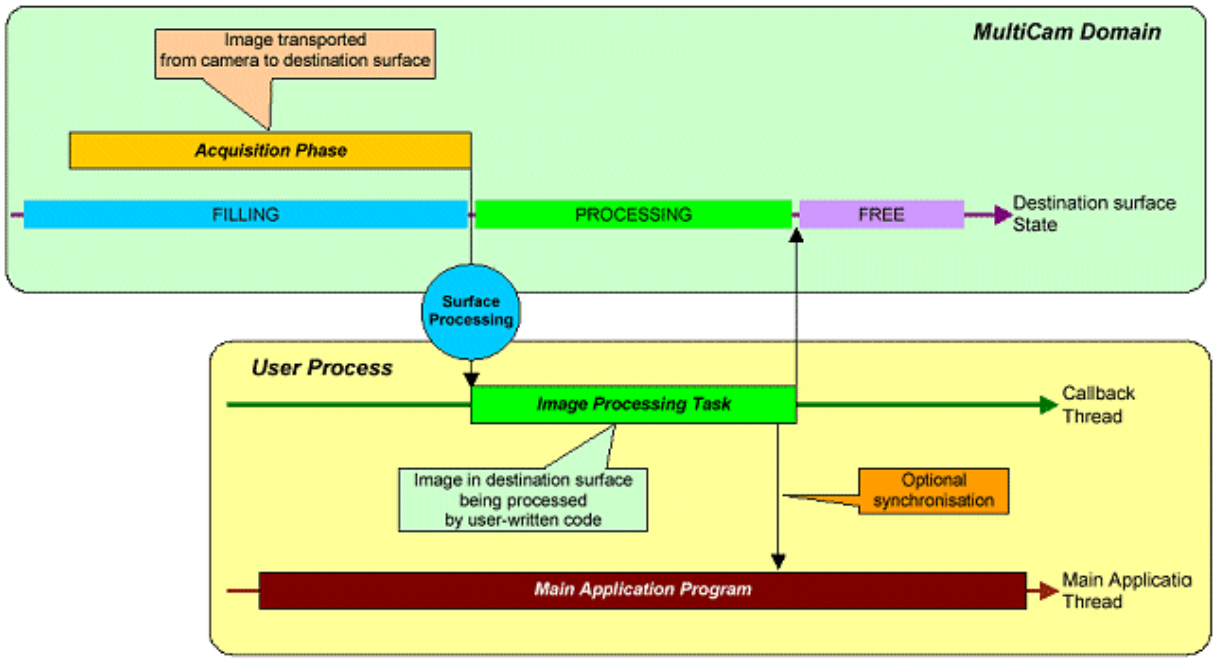
5.2. Callback Signaling

Callback Signaling Mechanism

The callback mechanism implies an event driven behavior. The following description uses the Surface Processing signal as an example of callback generating event.

The Surface Processing signal occurs when a transfer phase terminates. It is issued by a channel to indicate that the destination memory surface has been filled with an image coming from the source camera, and that this surface is available for image processing (see **SurfaceState**).

The image processing task is performed on this event by a special function called the callback function.



Callback mechanism

The callback function is called by the MultiCam driver, not by the user application. This ensures that the image-processing task is realized at the ideal instant, exactly when the surface becomes ready for processing.

MultiCam benefits from several built-in features to ease the implementation of the callback function.

- A dedicated thread is created for the callback function execution.
- The callback function prototype is declared in the MultiCam system C header file.
- Means are provided to designate the channel and the signal(s) issuing the callback function calls.
- The callback function argument provides all relevant information to the user-written code.

The MultiCam function to register a callback function to a channel is `McRegisterCallback`.

Callback Signaling Information

Callback Function Prototype

The callback function prototype is declared in the MultiCam system's `MultiCam.h` header file as follows:

```
typedef void (MCAPI *PMCCALLBACK)(PMCSIGNALINFO SignalInfo);
```

Item	Type	Description
Function	`PMCCALLBACK`	Callback function
`SignalInfo`	`PMCSIGNALINFO`	Argument providing the signal information structure.

The user should define the callback function in the application code in accordance with this prototype.

The callback function is called by the MultiCam driver when a channel issues a pre-defined signal.

The pre-defined signal should be enabled with the **SignalEnable** parameter. It is allowed to enable several signals.

If more than one enabled signals are issued simultaneously from an object, the callback function is successively called for each signal occurrence.

When the signal occurs, the callback dedicated thread is released, and the callback function is automatically invoked. The thread is restored to an idle condition when the callback function is exited.

The function has a single argument, which is a structure passing information on the signal that caused the callback function. This structure has the *signal information* type.

If the callback signaling mechanism is used, the waiting and advanced signaling mechanisms cannot be used.

Registration of Callback Function

A callback function should be registered to a channel object before use. Only one callback function per object is supported.

Registering the callback function results into the creation in the application process of a separate thread dedicated to the callback function. This thread is maintained in a idle state until a signal occurs. There can be only one dedicated thread per channel object.

A dedicated MultiCam function is provided -for callback registration: `McRegisterCallback`.

Context

`Context` is an argument of the callback registration function as well as a member of the signal information structure available to the callback function.

The user is free to use this item at the registration time to hold any identifying information he may find useful.

When the callback function is executed, the user gets back the context information as it was passed to the registration function.

Code Example of Callback Mechanism

The following code uses the callback mechanism to process images grabbed during an acquisition sequence. One or several surfaces have to be created and assigned to the cluster owned by the channel. At the end of each acquisition phase, the surface is filled and made available to the callback function. The `Status` variable can be used for error checking.

[C]

```
void MyApplication()
{
//Application level initializing code
MCSTATUS Status = McOpenDriver(NULL);
//Application level initializing code
MCHANDLE MyChannel;
Status = McCreateNm("CHANNEL", &MyChannel);
Status = McSetParamInt(MyChannel, MC_DriverIndex, 0);
Status = McSetParamInt(MyChannel, MC_Connector, MC_Connector_M);
//Assign grabber and camera to channel
//Configure channel including triggering mode
//Assign to channel a destination cluster of surfaces
//Registering the callback function
Status = McRegisterCallback(MyChannel, MyFunction, NULL);
//Activating acquisition sequence
Status = McSetParamInt(MyChannel, MC_ChannelState, MC_ChannelState_ACTIVE);
//Acquisition sequence is now active
//A callback is automatically generated after each acquisition phase
//Deleting the channels
Status = McDelete(MyChannel);
//Disconnecting from driver
Status = McCloseDriver();
}
void MCAPI MyFunction(PMCSIGNALINFO SignalInfo){
//...
//Image processing code
//Image to be processed is available in the destination cluster of surfaces
//...
}
```

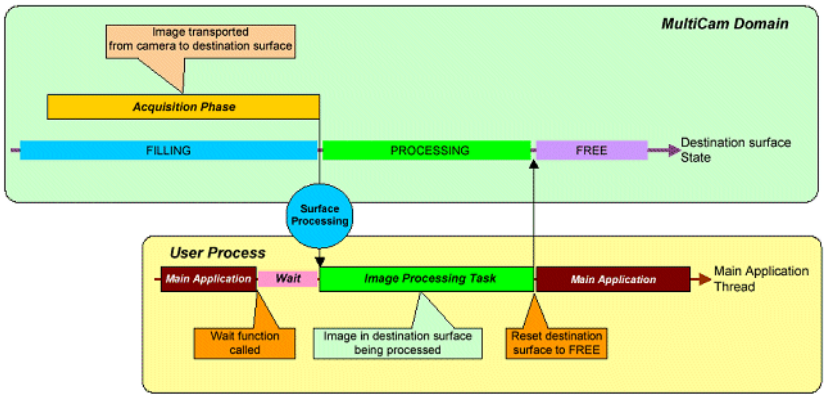
5.3. Waiting Signaling

Waiting Signaling Mechanism

The waiting mechanism implies an event being waited for. The following description uses the Surface Processing signal as an example of the waited event.

The Surface Processing signal occurs when a transfer phase terminates. It is issued by a channel to indicate that the destination memory surface has been filled with an image coming from the source camera, and that this surface is available for image processing.

The main application thread has to wait for a specified event to occurs. Subsequently, the image processing task is performed.



Waiting mechanism

The MultiCam wait function is `McWaitSignal`.

Waiting Signaling Information

The main application thread can be forced to wait for a specific MultiCam signal. The signal should be enabled with the **SignalEnable** parameter.

Only one signal can be waited for in a given thread at a given time.

The dedicated MultiCam function to control the waiting mechanism is called `McWaitSignal`.

If the expected signal does not occur within the specified timeout, the function returns `MC_TIMEOUT`.

When waiting for the Surface Processing signal, it is the application responsibility to reset the **SurfaceState** parameter of the **PROCESSING** surface to **FREE** when done. Failure to do so will prevent the surface from being used by subsequent acquisitions.

If the waiting signaling mechanism is used, the callback signaling mechanism cannot be used. However, the advanced signaling mechanism can be used as long as it is not used for the particular event involved in the waiting function.

5.4. Advanced Signaling

Advanced Signaling Mechanism

In order to achieve the highest level of control over the event flow between the user process and the MultiCam operation, a linkage to the regular Windows events is provided.

The experienced user can use one of the following Windows' API functions.

- `WaitForSingleObject`
- `WaitForMultipleObject`
- `MsgWaitForMultipleObjects`

The application calls these functions with handles to events of any nature, including the MultiCam generated events.

A dedicated MultiCam parameter is included, called **SignalEvent**. This parameter is a collection of handles to events corresponding to all MultiCam signals.

The events pointed by these handles are set to the signaled state on each corresponding MultiCam signal occurrence. Restoring the events to the non-signaled state is automatically managed by the MultiCam driver.

A MultiCam function called `McGetSignalInfo` is specifically provided to retrieve all relevant information on a specified signal owned by a specified channel object.

Linkage of MultiCam Signals to Windows Events

To implement an advanced signaling mechanism of his own, the user has to associate a signal to a regular Windows event.

This is realized with an expert-level parameter called **SignalEvent**.

One such parameter exists for the channel class. It has the MultiCam type "integer collection".

Each item of the collection allows for retrieving the operating system event object associated to the addressed signal.

To address a specific signal, the by-ident parameter access method is used with the **SignalEnable** parameter belonging to the desired channel object. The parameter getting function is `McSetParamInt` with a parameter identifier established as follows:

To reach signal...	Use parameter identifier...
Start Exposure	**MC_SignalEvent** + **MC_SIG_START_EXPOSURE**
End Exposure	**MC_SignalEvent** + **MC_SIG_END_EXPOSURE**
Release (*)	**MC_SignalEvent** + **MC_SIG_RELEASE**
Surface Filled	**MC_SignalEvent** + **MC_SIG_SURFACE_FILLED**
Surface Processing	**MC_SignalEvent** + **MC_SIG_SURFACE_PROCESSING**
Cluster Unavailable	**MC_SignalEvent** + **MC_SIG_CLUSTER_UNAVAILABLE**
Acquisition failure	**MC_SignalEvent** + **MC_SIG_ACQUISITION_FAILURE**

The retrieved value may be cast into an operating system handle (`HANDLE`) and subsequently used in any of the following wait functions:

- `WaitForSingleObject`
- `WaitForMultipleObject`
- `MsgWaitForMultipleObjects`

The operating system event is signaled each time an enabled MultiCam signal occurs. Enabling a MultiCam signal is done with the **SignalEnable** parameter. It is allowed to enable several signals.

When waiting for the Surface Processing signal, it is the application responsibility to reset the **SurfaceState** parameter of the **PROCESSING** surface to **FREE** when done. Failure to do so would prevent the surface from being used by subsequent acquisition phases.

The MultiCam signal information associated with the event may be retrieved by calling the `McGetSignalInfo` function.

If the advanced signaling mechanism is used, the callback signaling mechanism cannot be used. However, the waiting signaling mechanism can be used as long as the waiting function is not used to wait for an event used by the advanced signaling mechanism.

Example

```
HANDLE MyHandle;
McSetParamInt(hChannel, MC_SignalEnable + MC_SIG_SURFACE_FILLED, MC_SignalEnable_ON);
McGetParamInt(hChannel, MC_SignalEvent + MC_SIG_SURFACE_FILLED, (int*)&MyHandle);
WaitForSingleObject(MyHandle, INFINITE);
```

Retrieving Signal Information

A dedicated MultiCam function is provided to retrieve the signal information for one or several specified signals issued by a specified object at any time. The retrieving function is called `McGetSignalInfo`. The function should be called several times if information from several signals is to be retrieved.

Any signal the information of which is to be retrieved should be enabled with the **SignalEnable** parameter.

6. Pixel Formats Conversion

In order to use software pixel formats conversion, the user must first create a MultiCam surface (refer to "How to Create and Delete Surfaces" on page 65) then configure its parameters.

The following code example shows how to convert a **BAYER8** surface into an **RGB24** surface. We assume that `surfaceIn` is the `MCHANDLE` to an acquired surface to convert which is in the **PROCESSING** state.

```
// Get surfaceIn properties
INT32 width, height, pitchIn;
McGetParamInt(surfaceIn, MC_SurfaceSizeX, &width);
McGetParamInt(surfaceIn, MC_SurfaceSizeY, &height);
McGetParamInt(surfaceIn, MC_SurfacePitch, &pitchIn);

// Create surfaceOut and allocate buffer
MCHANDLE surfaceOut = 0;
McCreate(MC_DEFAULT_SURFACE_HANDLE, &surfaceOut);
INT32 pitchOut = pitchIn * 3;
INT32 sizeOut = pitchOut * height;
unsigned char *buffer = new unsigned char[sizeOut];

// Configure surfaceOut to hold the RGB24 converted data
McSetParamPtr(surfaceOut, MC_SurfaceAddr + 0, buffer);
McSetParamInt(surfaceOut, MC_SurfaceSize + 0, sizeOut);
McSetParamInt(surfaceOut, MC_SurfacePitch + 0, pitchOut);
McSetParamInt(surfaceOut, MC_SurfaceSizeX, width);
McSetParamInt(surfaceOut, MC_SurfaceSizeY, height);
McSetParamInt(surfaceOut, MC_SurfaceColorFormat, MC_ColorFormat_RGB24);
McSetParamInt(surfaceOut, MC_SurfaceColorComponentsOrder, MC_ColorComponentsOrder_
BGR);

// Convert surfaceIn into surfaceOut
McConvertSurface(surfaceIn, surfaceOut);
```

Later, in cleanup code:

```
// Delete surfaceOut and associated buffer when no longer used
if (surfaceOut != 0)
    McDelete(surfaceOut);
if (buffer != NULL)
   delete[] buffer;
```

7. Exceptions

7.1. API Errors

Error Reporting Behaviors

Four behaviors are supported when an error occurs during the execution of a MultiCam API function.

The wished behavior is selected with the **ErrorHandling** parameter, which can assume one of the four following values:

ErrorHandling	Behavior designation	Effect
NONE	Return	The MultiCam function returns the error code.
MSGBOX	Message	A dialog box is displayed.
EXCEPTION	Exception	A Windows structured exception is issued.
MSGEXCEPTION	Message and exception	A dialog box is displayed, allowing for a Windows structured exception to be issued.

Return Behavior

This behavior is assumed when the MultiCam configuration parameter **ErrorHandling** is set to **NONE**.

All C language API functions have the type `MCSTATUS` and return an integer value.

Normally, the returned value is `MC_OK`, which is equal to zero, indicating that no error occurred during the function execution.

In case of error, a negative value is returned. This value is an error code associated to an error identifier.

Message Behavior

This behavior is assumed when the MultiCam configuration parameter **ErrorHandling** is set to **MSGBOX**.

When a MultiCam API function encounters an error, it displays an error dialog giving more information about the problem. The user closes the dialog by selecting one of three buttons:

Selecting OK simply pass the error as a negative number (MultiCam error code) to the application, allowing the program to handle the error.

Selecting ABORT immediately terminates the application. All MultiCam associated resources are cleaned up excepted the application resources.

Selecting IGNORE forces the function to return MC_OK, allowing the program to ignore the error.

Important: *The message behavior is available on Windows operating systems only.*

Exception Behavior

This behavior is assumed when the MultiCam configuration parameter **ErrorHandling** is set to **EXCEPTION**.

When a MultiCam API function encounters an error, it throws a WIN32 structured exception. The exception code is the MultiCam error code.

Exception handling is usually performed by special compiler-dependent keywords.

The user will refer to the Windows SDK documentation and to the relevant compiler documentation for more information about WIN32 structured exceptions and how to handle them.

Important: *The exception behavior is available on Windows operating systems only.*

Message and Exception Behavior

This behavior is assumed when the MultiCam configuration parameter **ErrorHandling** is set to **MSGEXCEPTION**.

When a MultiCam API function encounters an error, it displays an error dialog giving more information about the problem. The user closes the dialog by selection on of three buttons:

Selecting OK simply throws a WIN32 structured exception. The exception code is the MultiCam error code.

Exception handling is usually performed by special compiler dependent keywords.

Selecting ABORT immediately terminates the application. All MultiCam associated resources are cleaned up excepted the application resources.

Selecting IGNORE forces the function to return MC_OK without throwing an exception, allowing the program to ignore the error.

The user will refer to the Windows SDK documentation and to the relevant compiler documentation for more information about WIN32 structured exceptions and how to handle them.

Important: *The message and exception behavior is available on Windows operating systems only.*

7.2. MultiCam Error Codes

Error codes returned by MultiCam functions

Return value	Error identifier	Description
0	**MC_OK**	No Error
-1	**MC_NO_BOARD_FOUND**	No Board Found
-2	**MC_BAD_PARAMETER**	Bad Parameter
-3	**MC_IO_ERROR**	I/O Error
-4	**MC_INTERNAL_ERROR**	Internal Error
-5	**MC_NO_MORE_RESOURCES**	No More Resources
-6	**MC_IN_USE**	Object still in use
-7	**MC_NOT_SUPPORTED**	Operation not supported
-8	**MC_DATABASE_ERROR**	Parameter database error
-9	**MC_OUT_OF_BOUND**	Value out of bound
-10	**MC_INSTANCE_NOT_FOUND**	Object instance not found
-11	**MC_INVALID_HANDLE**	Invalid Handle
-12	**MC_TIMEOUT**	Timeout
-13	**MC_INVALID_VALUE**	Invalid Value
-14	**MC_RANGE_ERROR**	Value not in range
-15	**MC_BAD_HW_CONFIG**	Invalid hardware configuration
-16	**MC_NO_EVENT**	No Event
-17	**MC_LICENSE_NOT_GRANTED**	License not granted
-18	**MC_FATAL_ERROR**	Fatal error
-19	**MC_HW_EVENT_CONFLICT**	Hardware event conflict
-20	**MC_FILE_NOT_FOUND**	File not found
-21	**MC_OVERFLOW**	Overflow
-22	**MC_INVALID_PARAMETER_SETTING**	Parameter inconsistency
-23	**MC_PARAMETER_ILLEGAL_ACCESS**	Illegal operation
-24	**MC_CLUSTER_BUSY**	Cluster busy
-25	**MC_SERVICE_ERROR**	MultiCam service error
-26	**MC_INVALID_SURFACE**	Invalid surface

7.3. Operational Exceptions

Frame Trigger Violation

This exception is signaled with a MultiCam signal identified by **MC_SIG_FRAMETRIGGER_VIOLATION**.

This signal is issued when a frame or page trigger has been received which could not be handled because an acquisition phase was still in progress. The trigger is lost.

Cluster Unavailable

This exception is signaled with a MultiCam signal identified by **MC_SIG_CLUSTER_UNAVAILABLE**.

This signal is issued when the cluster mechanism has not been able to designate one of its surface as the destination of frame or page acquisition.

Acquisition Failure

This exception is signaled with a MultiCam signal identified by **MC_SIG_ACQUISITION_FAILURE**.

This signal is issued when the channel acquisition timeout timer expires before the end of the acquisition. In that case the channel is disabled and must be deleted.

8. CAM Files

What Is a CamFile?

The CamFile can be seen as a script of MultiCam setting functions that are played when the **CamFile** parameter is written to.

After the CamFile is played, the channel is ready to operate according to the parameter settings specified in the file. Generally speaking, it means that the channel is ready to start an acquisition for a specified camera in a specified fundamental mode.

CAM stands for Camera. In the computer file system, the CamFile exhibits the `.cam` extension.

A CamFile is a readable ASCII file having the following structure:

- An identification header (optional)
- A pair of assignments for the **Camera** and **CamConfig** parameters (mandatory)
- A list of assignments for all relevant channel parameters (optional)

CamFile Identification Header

The identification header is an optional section that includes directives used exclusively by MultiCam Studio.

Example of a CamFile header

```
;*************************************************************
; Camera Manufacturer: My Cameras
; Camera Model: ProgressiveFR
; Camera Configuration: Progressive Free-Run Scanning, Analog synchronization
;*************************************************************
```

The MultiCam Studio CamFile directives have the simple format:

```
; <DirectiveName>: <DirectiveValue> <EOL>
```

All values are string of characters terminated by an end of line.

Directive name	Value meaning
Board	Restricts the visibility of the camera in the camera selection wizard of MultiCam Studio. When value is Domino, the CamFile is listed only when the channel is created on a a Domino board.When value is Grablink, the CamFile is listed only when the channel is created on a a Grablink board. Other values are simply ignored. If more than one board directive is present, only the first one is considered
Camera Manufacturer	Declare the manufacturer name to display in the camera selection wizard of MultiCam Studio
Camera Model	Declare the camera model name to display in the camera selection wizard of MultiCam Studio
Revision	Declare the revision number and/or date of the CamFile

CamFile Parameter Assignment

A parameter assignment line has the following format:

<ParameterName> = <ParameterValue> [;<Comment>] <EOL>

where:

ParameterName is a valid MultiCam parameter name for the targeted board.

ParameterValue is a valid MultiCam parameter value for the parameter.

- Only one parameter assignment per line is allowed.
- An optional comment can be appended to the assignment; it must be preceded by a semi-column.
- Every line containing a parameter assignment must be terminated by a valid End-Of-Line: a CR or a pair of CR and LF characters.
- Spaces or Tab characters can be freely inserted anywhere.
- Empty lines, lines containing only comments, are allowed.

Example of parameters assignment lines

```
Camera = ProgressiveFR
CamConfig = PxxSA      ;
Gain=1000
TargetFrameRate_Hz = 0.5; 1 frame every two seconds
```

Example of comment lines

```
; Camera Specification category
;------------------------------
; Gain=1000
```

Loading the CamFile

The loading of a CamFile into a MultiCam channel is a matter of setting the **CamFile** parameter

of a MultiCam channel to the value of the CamFile name (without the .cam extension)

When a CamFile is loaded, it is simply interpreted by the MultiCam driver as a series of "set parameter" function calls.

Examples

The following lines of code implement possible **CamFile** parameter assignment to a MultiCam channel defined in a Domino board (depends of the camera).

```
MCSTATUS Status = McSetParamStr(MyChannel, MC_CamFile, "VCC-870A_P15RA");
MCSTATUS Status = McSetParamStr(MyChannel, MC_CamFile, "KP-F3_I60SM");
MCSTATUS Status = McSetParamStr(MyChannel, MC_CamFile, "XC-ES30CE_I50SM_R");
```

The following lines of code implement possible camera assignment to a MultiCam channel defined in a Grablink board (depends of the camera).

```
MCSTATUS Status = McSetParamStr(MyChannel, MC_CamFile, "4000m_P16RG");
MCSTATUS Status = McSetParamStr(MyChannel, MC_CamFile, "Colibri2048CL_L2048RG");
```

CamFile Templates

A CamFile template is a file intended to be customized by the MultiCam user willing to interface a particular camera with a Domino or a Grablink board.

The MultiCam driver is delivered with a collection of templates. The MultiCam driver installation tool installs the CamFile templates as follows:

The CamFile templates applicable to the the Grablink boards are stored in the `<InstallDir>\Cameras\_TEMPLATES\Grablink\` directory.

Refer to Analog Cameras Documentation for additional information about CamFile templates for Domino boards.

Refer to Interfacing Camera Link Cameras for additional information about CamFile templates for Grablink boards.

Camera Interface Packages Library

A Camera Interface Package is a set of files that contains all the information needed by a MultiCam user to configure a MultiCam channel for a particular camera model. A Camera Interface Package is a ZIP file that includes:

- Ready-to-use CamFiles with the exhaustive set of relevant parameters. One for each of the recommended operating modes
- A documentation explaining how to use this particular camera model with Euresys frame grabbers

When unzipped on the target machine, the CamFiles and the documentation are extracted in the `<InstallDir>\Cameras\<Manufacturer>\` folder.

The library of Camera Interface Packages contains a large amount of packages for both analog and Camera Link digital camera models. Furthermore, this library is regularly updated with new packages and constantly growing.

There are 2 ways to access the library:

1. Automatic update with MultiCam Studio

MultiCam Studio provides a convenient way to download and update all the available CamFiles. MultiCam Studio automatically downloads and installs on the MultiCam install directory, from the website, a ZIP file containing the CamFiles and the associated PDF documentation files.

2. Free downloads from the Euresys website

The library directory is available online on https://www.euresys.com/Support/Supported-cameras. The directory can be easily browsed using interactive filters. Each entry in the directory provides the following fields:

- Camera manufacturer name
- Camera model name
- Compatible Euresys boards
- Link to the Camera Interface Package ZIP file

PART III
THE CONFIGURATION OBJECT

1. The Configuration Object

The configuration object groups all MultiCam parameters dedicated to the **control of system-wide features**.

The system should be basically understood as the set of Euresys boards installed inside a host computer. The configuration object also addresses any hardware or software element of the host computer requesting some degree of control for the MultiCam system operation.

The configuration object does not belong to a true class, as it is unique within the system. There is no need for the user to instantiate the `Configuration` class. The configuration object is natively made available to the application when the MultiCam driver is connected to it.

PART IV
THE BOARD OBJECT

1. The Board Object

The Board object groups all MultiCam parameters dedicated to the *control of features specific to a board*.

The board parameters also address the access of I/O lines from an application program, implementing the general-purpose I/O functionality.

The board object does not belong to a true class, as it is unique for each Euresys board installed inside a host computer. There is no need for the user to instantiate the `Board` class. One board object is natively made available to the application for each installed Euresys board when the MultiCam driver is opened.

2. Board Information Parameters

2.1. Board Identification: Addressing a Board

The MultiCam driver is capable of multiple grabbers servicing. This means that there can be more than one MultiCam-compliant frame grabber board in the host computer running under the MultiCam environment.

The board object encompasses a set of parameters aimed at controlling and describing the full set of features associated to these several boards.

Each board is described by a set of similar parameters. In order to specifically reach the parameters concerning a pointed board, the board object features a board addressing mechanism.

Four different ways are provided to address a particular board:

- Index-addressing
- PCI-addressing
- Name-addressing
- Identifier-addressing

The board-class parameters constituting the "Board Information" category are similar to the board addressing mechanism available in the channel class. However, as board-class parameters, they can only get information.

The number of boards recognized by the MultiCam driver can be returned through the configuration-object **BoardCount** parameter.

Index-Addressing

The board information integer MultiCam parameter implementing the index-addressing mechanism is **DriverIndex**.

When the MultiCam driver is launched, the hardware system is scanned and all present MultiCam compliant boards are reported.

The driver establishes a list of the boards, starting at index zero and incrementing by one up to the last available board. The order in which the boards are effectively scanned cannot be predicted, as it is BIOS dependent. However, for a given host computer and a identical insertion of the boards in the PCI slots, the numbering order will be consistently repeated.

Setting **DriverIndex** to a value between **0** and **BoardCount–1** uniquely designates one of the boards. Any set or get function involving a "board information" parameter will address the designated board.

Getting **DriverIndex** returns the index of the presently selected MultiCam board.

PCI-Addressing

The board information integer MultiCam parameter implementing the PCI-addressing mechanism is **PciPosition**.

Each PCI slot in the PC can be identified by means of a number different from slot to slot. The numbering system can differ from PC to PC, but is consistent for a given model of PC running the same BIOS revision. This number is independent of the board type effectively inserted in the slot.

Setting **PciPosition** to a value previously known as valid uniquely designates one of the boards. Any set or get function involving a "board information" parameter will address the designated board.

Getting **PciPosition** returns the position of the presently selected MultiCam board.

Name-Addressing

The board information string MultiCam parameter implementing the name-addressing mechanism is **BoardName**.

A MultiCam compliant board is given a name. The name is a string of maximum 16-ASCII characters. It can be altered by the user if desired, and can be made unique for each board in a system.

The name is written in a non-volatile memory residing in the board.

Getting **BoardName** returns the name of the presently selected MultiCam board.

Identifier-Addressing

The board information enumerated MultiCam parameter implementing the identifier-addressing mechanism is **BoardIdentifier**.

Any Euresys board can be uniquely designated by a combination of its type and its serial number, called the board's identifier. The board identifier is an ASCII character string resulting from the concatenation of the board type (refer to **BoardType**) and the serial number, with an intervening underscore.

The serial number is a 6-digit string made of characters 0 to 9.

Getting **BoardIdentifier** returns the identifier of the presently selected MultiCam board, for instance **COLORSCAN_000123**.

Code Examples to Access Board Information

Code Example: How to Gather Board Information?

The following code scans all installed MultiCam-compliant boards, and builds a database containing their information relative to name, serial number and type.

`MC_CONFIGURATION` is the C identifier used as a handle to the configuration object. This object has not to be explicitly instantiated.

`MC_BOARD` is the C identifier used as a handle to the board object. This object has not to be explicitly instantiated.

The `Status` variable can be used for error checking.

[C]

//Defining the database structure type

```
typedef struct
{
  char BoardName[17];
  INT32 SerialNumber;
  INT32 BoardType;
} MULTICAM_BOARDINFO;
```

//Variables declaration

```
MULTICAM_BOARDINFO BoardInfo[10];
INT32 BoardCount;
INT32 i;
MCSTATUS Status;
```

//Connecting to driver

```
Status = McOpenDriver(NULL);
```

//Getting number of boards

```
Status = McGetParamInt(MC_CONFIGURATION, MC_BoardCount, &BoardCount);
```

//Scanning across MultiCam boards

```
for (i=0; i<BoardCount; i++)
{
```

//Fetching the board name (String MultiCam parameter)

```
  Status = McGetParamStr(
    MC_BOARD+i,
    MC_BoardName,
```

```
    BoardInfo[i].BoardName,
    17);
```

//Fetching the board serial number (Integer MultiCam parameter)

```
  Status = McGetParamInt(
    MC_BOARD+i,
    MC_SerialNumber,
    &BoardInfo[i].SerialNumber);
```

//Fetching the board type (Enumerated MultiCam parameter)

```
  Status = McGetParamInt(
    MC_BOARD+i,
    MC_BoardType,
    &BoardInfo[i].BoardType);
}
```

//Disconnecting from driver

```
Status = McCloseDriver();
```

Code Example: How to Name a MultiCam Board?

The following code assigns the name `MYBOARD` to the MultiCam board with `DriverIndex = 5`.

[C]

//Connecting to driver

```
MCSTATUS Status = McOpenDriver(NULL);
```

//Renaming the board

```
Status = McSetParamStr(MC_BOARD, MC_NameBoard+5, "MYBOARD");
```

//Disconnecting from driver

```
Status = McCloseDriver();
```

2.2. Board Security Feature

A security feature is incorporated in all MultiCam-compliant boards.

The general idea is that the OEM application programmer is able to engrave in the board a secret proprietary key.

The security key is an 8-bytes string of ASCII characters. Any character is allowed. A null character acts as the termination character of the key.

The security key is stored in the non-volatile memory of the board and cannot be read back.

There is no way to obtain this security key number back from the board. However, it is possible to verify that a given board currently holds a security key equal to a given one.

Using this simple mechanism, it is easy to lock an application to a board or to a set of boards.

Code Examples to Manage Security Features

Writing a Security Key Into a Board

Setting the **OemSafetyLock** parameter to an 8-bytes string activates the recording of the security key.

The following code engraves `MY_NUM18` in the board with `DriverIndex = 1`. The `Status` variable can be used for error checking.

[C]

//Connecting to driver

```
MCSTATUS Status = McOpenDriver(NULL);
```

//Entering the key number

```
Status = McSetParamStr(MC_BOARD +1, MC_OemSafetyLock, "MY_NUM18");
```

//Disconnecting from driver

```
Status = McCloseDriver();
```

Checking the Security Key Validity for a Board

Setting the **OemSafetyKey** parameter to a an 8-bytes string instructs the driver that this security key is to be checked.

Subsequently, getting the **OemSafetyLock** parameter returns a string **TRUE** if the security key recorded in the board matches the value entered with the parameter **OemSafetyKey**. Otherwise, the returned string is **FALSE**.

The targeted board is selected by `DriverIndex = 1`. The security key is `MY_NUM18`.

The `Status` variable can be used for error checking.

[C]

//Variables declaration

```
char Match[6];
```

```
MCSTATUS Status;
```

//Connecting to driver

```
Status = McOpenDriver(NULL);
```

//Entering the reference key number

```
Status = McSetParamStr(MC_BOARD + 1, MC_OemSafetyKey, "MY_NUM18");
```

//Checking for correspondence

```
McGetParamStr(MC_BOARD + 1, MC_OemSafetyLock, &Match, 6);
```

```
if (Match=="FALSE")
{
```

//...
//Rejection code
//...

```
}
```

//Disconnecting from driver

```
Status = McCloseDriver();
```

2.3. Board Topology

The MultiCam system design is such that, for some frame grabbers, it is necessary to declare beforehand how the cameras are wired to the frame grabber.

This information should be entered *before the creation of any channel* for the targeted board.

The board-object parameter **BoardTopology** is provided for this effect.

3. Input/Output Control Parameters

See also How to Work With Input/Output Lines?

PART V
THE CHANNEL CLASS

1. The Channel Class

The Channel class groups all MultiCam parameters dedicated to the control of image acquisition related features.

A Channel object is an instance of the Channel class, represented by a dedicated set of such parameters.

Typically, the following items are defined and controlled by the Channel object:

- The camera feeding the channel, including reset and exposure control
- The connector and cable linking the camera to the frame grabber
- The switching structures routing the analog or digital video signal inside the frame grabbe
- In case of analog camera, the analog-to-digital converter and the associated signal conditioning devices
- In case of digital camera, the digital receiving or de-serializing devices
- The timing generator and controller associated to the camera, and the video signal conditioning
- All digital devices affecting the signal during acquisition, performing tasks such as lookup tables, byte alignment, data channel merging...
- The data buffer receiving the images
- The DMA devices extracting images out of the data buffer for transfer into host memory
- The destination cluster of host memory surfaces
- The hardware resources managing the external system trigger

The channel is the association of an individual grabber connected to a camera delivering data to a set of surfaces, called a cluster. The channel is able to transport an image from the camera towards a surface belonging to the cluster and usually located in the host memory.

2. Introduction to MultiCam Channels

2.1. The Channel Concept

A MultiCam channel can be thought as an acquisition path between a defined camera and a defined cluster of surfaces.

The camera is an off-the-shelf device chosen, bought and installed by the user. Euresys supports many cameras from many third-party manufacturers.

A cluster of surfaces is a MultiCam item describing several memory buffers resident in the host computer.

The application author sees the MultiCam channel as a set of adjustable parameters. Any resource participating in the acquisition process is placed under control of one or several parameters.

A channel is an instantiable class. This means that there exist several MultiCam channels, exhibiting similar functionality, but representing different acquisition paths.

Usually, for each camera installed in the MultiCam system, the user will create a corresponding channel.

However, it may be useful to create several channels sharing the same camera. The different channels would differ by an aspect other than the camera, for instance the destination cluster.

2.2. Grabber and Camera Association

A MultiCam channel has the fundamental mission of transferring images from the camera into the host computer memory. To achieve this, the channel uses some hardware resources belonging to the frame grabber board.

The set of needed resource to perform acquisition is called a grabber. Some high-end frame grabbers have enough built-in resources to perform several acquisition tasks simultaneously. It is equivalent to say that such frame grabbers incorporate several grabbers.

When performing acquisition, a channel appropriates a grabber. The grabber appropriation lasts as long as needed to acquire a frame (area-scan) or a page (line-scan).

The tenure of time the channel appropriates a grabber is called an acquisition phase.

During an acquisition phase, the MultiCam channel realizes the temporary association of a camera and a grabber to deliver an image to a surface belonging to the destination cluster.

2.3. Concurrent Acquisition

For a frame grabber owning several grabbers, the concurrent acquisition is possible.

To implement the concurrent acquisition mode, the user will create a number of channels equal to the number of connected cameras, and assign each camera to each channel. All channels will appropriate one grabber each, and this resource appropriation should not surpass the frame grabber capabilities.

After the creation of channels, the application has to adequately configure them, and to activate an acquisition sequence for each channel. Subsequently, the channel effectively grabs images from its assigned camera as soon as the triggering conditions are met.

An acquisition sequence is, by definition, a succession of acquisition phases. Because the concurrent acquisition is possible, each activated channel can immediately enter into an acquisition phase at any time, independently of other channels' activity.

The total independence of channels is a characteristic of the concurrent acquisition mode.

2.4. Switched Acquisition

When the number of cameras connected to a board is larger than the number of available grabbers, the switched acquisition is required.

To implement the switched acquisition mode, the user will create a number of channels equal to the number of connected cameras, and assign each camera to each channel. However, the grabbers are not enough in quantity to satisfy the need of all channels simultaneously. The grabbers have to be considered as a resource common to several channels, and rules of appropriation are needed.

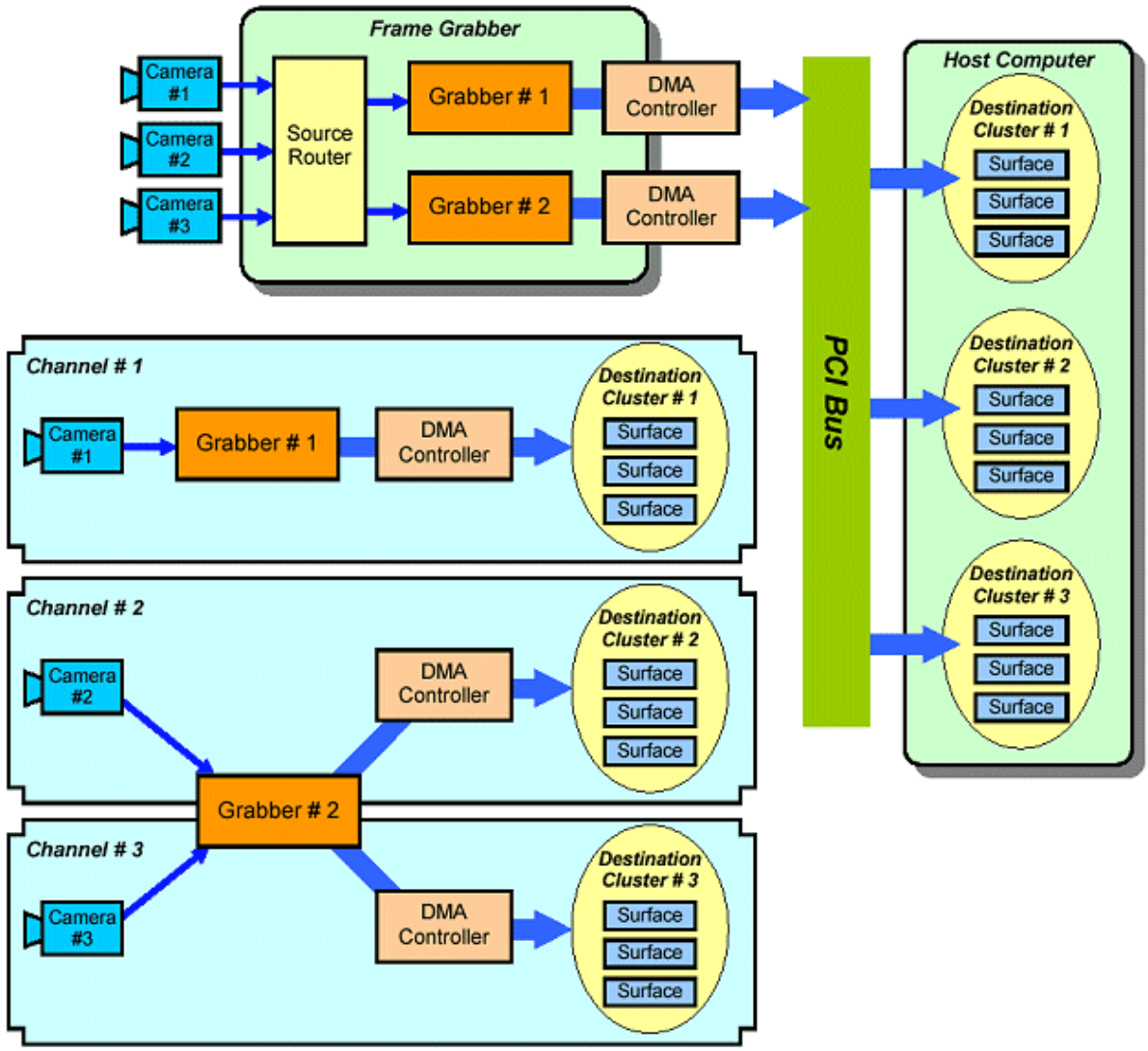
After the creation of channels, the application has to adequately configure them, and to activate an acquisition sequence for each channel. Subsequently, the channel becomes ready to grab images from its assigned camera as soon as the triggering conditions are met.

An acquisition phase does not necessarily occur immediately. The channel may have to wait for another channel to release the grabber. Actually, a grabber is appropriated by a channel for an acquisition phase, and, as long as the acquisition phase is running, the grabber is busy for other channels.

An acquisition sequence is, by definition, a succession of acquisition phases. Because the switched acquisition is involved, each activated channel cannot enter into an acquisition phase at any time, depending on the activity of other channels. The channel has to wait for its expected grabber to get free.

When multiple channels compete for the same hardware resources, MultiCam handles the resource allocation automatically. For more details, refer to understanding automatic switching.

2.5. A Pictorial View of a MultiCam System



This drawing shows three channels, numbered #1, #2 and #3.

The channel #1 realizes an acquisition path from the camera #1 to the cluster #1. This channel appropriates the grabber #1 for its sole usage. Consequently, the channel #1 can operate in a concurrent fashion, without interfering with the other channels.

The channel #2 realizes an acquisition path from the camera #2 to the cluster #2. The channel #3 realizes an acquisition path from the camera #3 to the cluster #3.

This pair of channels needs the grabber #2 for operation, as only two grabbers are available in the frame grabber. This means that the grabber #2 is a shared resource, and channels #2 and #3 will operate in a switched fashion.

3. Configuring a Channel

3.1. Declaring a Topology

The MultiCam system design is such that, for some frame grabbers, it is necessary to declare beforehand how the cameras are wired to the frame grabber.

A board-class MultiCam parameter called **BoardTopology** is provided for this effect.

3.2. Creating the Channel

A channel object is implemented with a special creation function provided by the MultiCam API.

The created channel is defined by a creation model specific to a given frame grabber. For detailed information about these channel creation models, refer to Channel Creation.

The designation of the location where the camera is connected is entered at the creation through the model argument.

Based on the information provided by the topology parameter and by the creation model, the MultiCam system is able to locate the needed grabber resources owned by the frame grabber.

See also Code Examples for Objects Management.

3.3. Assigning a Board to a Channel

A physical board must be assigned to a channel instance. The board to channel assignment must be performed immediately after the channel creation before applying any other channel parameter setting

To specify the linkage of a board to a channel, four different ways are provided to address a particular board:

- Index-addressing
- PCI-addressing
- Name-addressing
- Identifier-addressing

The board linkage mechanisms are similar to the board addressing mechanism available in the board object. Refer to Board Information.

Four parameters constitute the "board linkage" category of channel-class MultiCam parameters:

- **DriverIndex** (integer)
- **PciPosition** (integer)
- **BoardName** (string)
- **BoardIdentifier** (string)

To assign a board to a channel, a linkage parameter belonging to this channel has to be set with a valid value, using one of the four provided methods. After that assignment, the grabber associated to the channel is known as residing in the designated board.

> **Note:** *To assign a Picolo board to a channel, the first instruction following channel creation must set **DriverIndex**. Other mechanisms are not available in this release.*

3.4. Assigning a Camera to a Channel

Assigning a camera to a MultiCam channel is a matter of setting the string channel-class select-level parameter **CamFile** to a value designating a released CAM file.

Refer to CAM Files.

The CAM file contents are basically a set of parameter setting instructions, stated in a legible way according to a specified syntax. It can be thought as a macro or script files.

When setting the parameter **CamFile** to a CAM file name, this file is "played". All parameter settings are realized, and the MultiCam channel is now operable for the combination of camera and operational mode corresponding to the CAM file.

3.5. Code Example to Configure a Channel

The following code fully creates and configures a channel for a 1622 Grablink Full board that is designated by **DriverIndex** = **1**.

The channel configuration requires the **DECA_10T8** tap configuration, which is consistent with the chosen **MONO_DECA** board topology.

The `Status` variable can be used for error checking.

[C]

```
//Connecting to driver
MCSTATUS Status = McOpenDriver(NULL);
//Declaring the topology
Status = McSetParamStr(MC_BOARD + 1, MC_BoardTopology, "MONO_DECA");
//Creating a channel
MCHANDLE MyChannel;
Status = McCreate(MC_CHANNEL, &MyChannel);
//Assigning the board to the channel
Status = McSetParamInt(MyChannel, MC_DriverIndex, 1);
Status = McSetParamStr(MyChannel, MC_Connector, "M");
//Playing the CAM file
Status = McSetParamStr(MyChannel, MC_CamFile, "P4-CC-04K07T_L12240RG");
//... Application code ...
//Deleting the channel
Status = McDelete(MyChannel);
//Disconnecting from driver
Status = McCloseDriver();
```

4. Understanding MultiCam Acquisition Phases

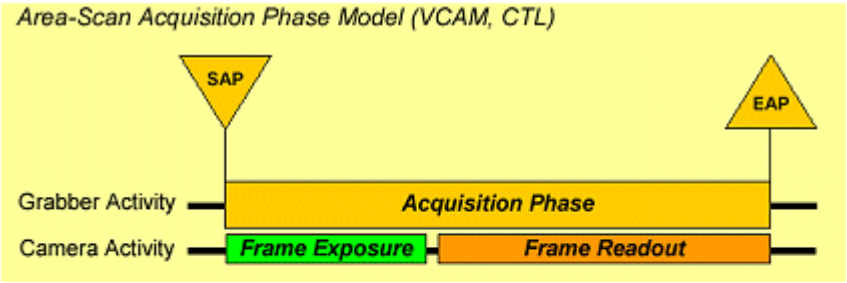
4.1. Acquisition Phase

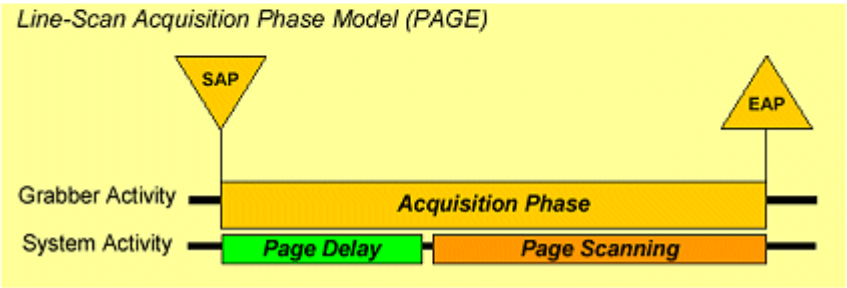
A MultiCam grabber is designed in such a way that it should operate during a non-interruptible amount of time in order to deposit an image into a destination surface of the host computer.

This non-interruptible amount of time is called the acquisition phase.

When an area-scan camera is operated in the VCAM or CTL mode, the acquisition phase is composed of a succession of "frame exposure" and "frame readout" sub-phases.

When a line-scan camera is operated in the PAGE mode, the acquisition phase is composed of a succession of "page delay" and "page scanning" sub-phases.





Considering the acquisition process in this manner provides a smart way to unify the description and the control in both situations.

In case of area-scan, the image data provided by the acquisition phase are called a frame. The number of lines constituting a frame is automatically inferred from the selected camera characteristics.

In case of line-scan, the image data provided by the acquisition phase are called a page. The number of lines constituting a page is chosen by the PageLength parameter.

The image data provided by an acquisition phase fits into the surface of the cluster defined in the host computer memory. There is a match between the size of the frame or page and the size of the destination surface.

Two events are introduced to account for the acquisition phase:

- **SAP** is a shortcut for the event marking the beginning of the acquisition phase (Start of Acquisition Phase)
- **EAP** is a shortcut for the event marking the end of the acquisition phase (End of Acquisition Phase)

4.2. Vanishing Initial Sub-Phase

Under certain circumstances, the subdivision of the area-scan acquisition phase into "frame exposure" and "frame readout" does not make sense.

When an area-scan camera is operated in the SYNC mode, the camera delivers frames continuously in a synchronous way. If an electronic shutter is used, the frame exposure occurs systematically during the previous frame readout condition, according to modalities established by the camera itself, not by the frame grabber.

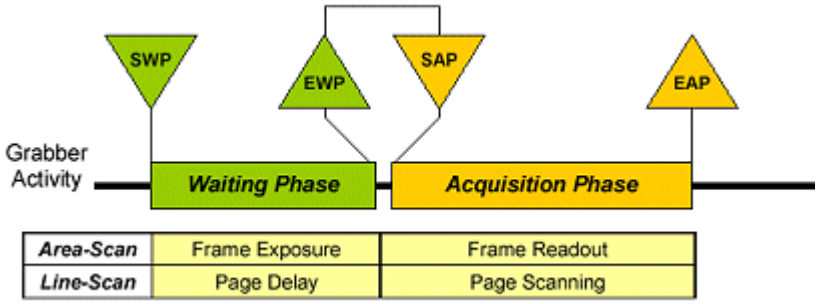
Consequently, it is advisable to identify the acquisition phase with the frame readout condition.

When a line-scan camera is operated in the WEB mode, the elementary pages forming the acquisition sequence are successively issued without intervening gap. The page delay feature is simply irrelevant.

These considerations are summarized in the following drawing:

4.3. Waiting Phase

It is sometimes useful to conceptually split the acquisition phase into its two constituting sub-phases. When this has to be done, the first sub-phase is called the waiting phase, while the second sub-phase keeps the name of acquisition phase.

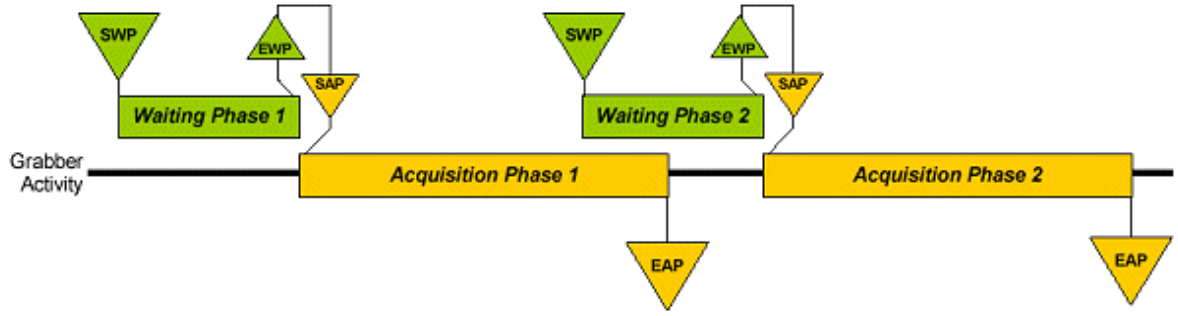


Two additional events are introduced to account for the waiting phase:

- **SWP** is a shortcut for the event marking the beginning of the waiting phase (Start of Waiting Phase)
- **EWP** is a shortcut for the event marking the end of the waiting phase (End of Waiting Phase)

The above drawing suggests that the EWP event unconditionally provokes the SAP event.

The reason for introducing the waiting phase concept is the functional requirement for a waiting sub-phase starting before the end of the previous acquisition sub-phase.



The above drawing shows that the definition of the acquisition phase as an interruptible amount of time is maintained despite the occurrence of the starting event SWP during the acquisition phase tenure.

This functional requirement is called phase overlapping.

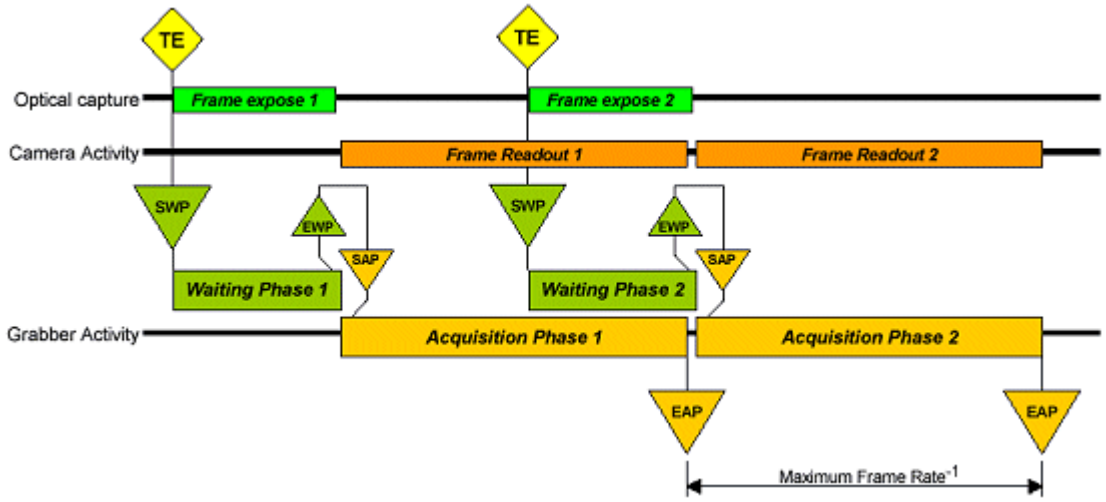
> **Note:** *The waiting phase events EWP and SWP are not available. They will be added in a future release.*

4.4. Phase Overlapping in Area-Scan

Area-scan phase overlapping is allowed when the following conditions are met:

MultiCam parameter	Value	Description
Expose	**PLSTRG** **WIDTH**	The area-scan camera feeding the channel operates in the "asynchronous reset" modality
Readout	**INTCTL**	
ExposeOverlap	**ALLOW**	

It is possible to trigger the optical image capture while the previous image is read out of the camera, as shown in the following drawing:



TE is a shortcut for Trigger Event. In this particular case, TE represents the frame trigger pulse.

The waiting phase duration is set by the adjust-level MultiCam parameter **Expose_us**.

Thanks to the phase overlapping feature, the theoretical maximum frame rate, usually only feasible is the "synchronous scanning" modality, can be reached in asynchronous reset applications.

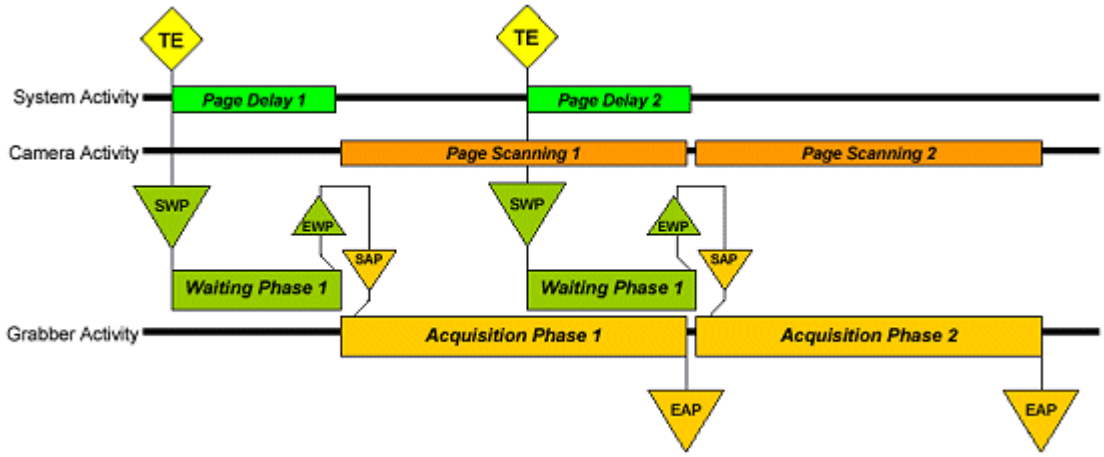
This allows for the highest frame rate in "on-the-fly" grabbing acquisition.

4.5. Phase Overlapping in Line-Scan

Line-scan phase overlapping is allowed when the following conditions are met:

MultiCam Parameter	Value	Description
AcquisitionMode	**PAGE**	The line-scan channel is indented for acquisition of moving discrete object

It is possible to trigger the scanning of the page before the end of the previous one. The scanning effectively starts after a programmable delay subsequent to the page trigger event.



TE is a shortcut for Trigger Event. In this particular case, TE represents the page trigger pulse.

The waiting phase duration is set by the adjust-level MultiCam parameter **PageDelay_Ln**.

The page delay corresponds to the distance the moving object has to travel to reach the axial position under the camera where it enters the field of view.

Successive objects to be scanned on the conveyor belt can be located close each other, while the position sensor generating the page trigger pulse can be installed way in front the scanning line.

5. MultiCam Triggering

5.1. Trigger Event (TE)

The trigger event is an item specifically related to a MultiCam channel. It is abbreviated as TE. The trigger event should be considered as the instant when the channel decides to request a grabber in order to fulfill an acquisition phase.

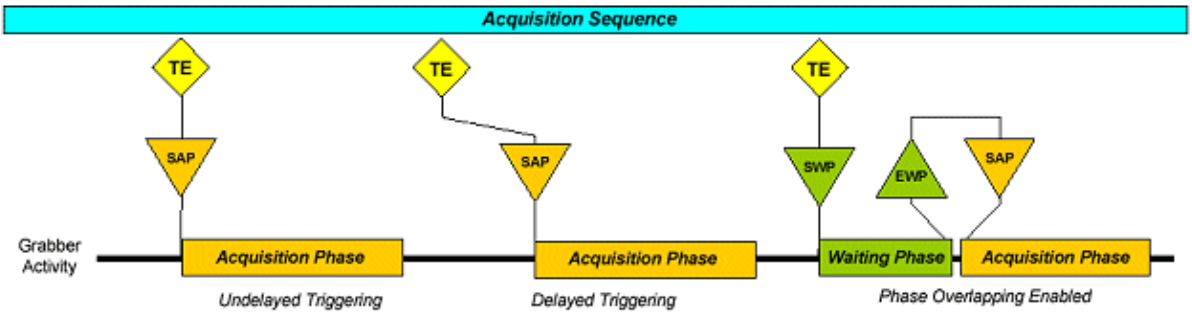
When some specified occurrence appears, a channel trigger event is generated. The origin of the trigger event is either external or internal according to the selected triggering modality. In addition, a number of channel parameters are needed to uniquely define the source of the trigger.

As a rule, when the trigger event occurs, an acquisition phase immediately starts. However, this is not always true because some internal circumstances can prevent the trigger event to be immediately served. The rules governing the behavior during those special circumstances are referred to as "trigger event delaying mechanisms".

When the grabber triggering mode is installed, the channel is allowed to use its own grabber at any time. Consequently, there is no resynchronization delay between the TE and the event starting the acquisition phase (SAP) or the waiting phase (SWP). However, when a synchronously scanned camera is used, the acquisition phase is delayed until the start of the next camera frame.

When the **timer triggering mode** or **system triggering mode** is installed, a delay can occur between the TE and the expected SAP.

Typical trigger event situations are pictured below:



5.2. Grabber Triggering Mode

In this mode, the trigger event associated to the channel can come from three origins:

- A specified logic transition on a hardware line managed by the grabber
- A special software control
- Another event

The first acquisition phase appearing in the acquisition sequence is discriminated and called the initial acquisition phase. It is caused by the initial trigger event.

The subsequent acquisition phases, if any, are caused by the trigger event subsequently occurring.

There is only one designated hardware-line able to supply trigger events, serving undistinguishingly the initial or subsequent acquisition phases.

The MultiCam adjust-level parameters establishing the triggering details are `TrigMode` and `NextTrigMode`.

Trigger Event Sources

The parameters `TrigMode` and `NextTrigMode` establish rules attributing a source to the trigger events (TE). The event triggering the first phase of an acquisition sequence is called the initial TE. The events triggering the subsequent phases of an acquisition sequence are called the subsequent TE.

`TrigMode`	Initial TE	`NextTrigMode`	Subsequent TE
IMMEDIATE	SAS	**REPEAT**	Previous EAP
HARD	Hardware line edge detection	**HARD**	Hardware line edge detection
SOFT	`ForceTrig` set to **TRIG**	**SOFT**	`ForceTrig` set to **TRIG**
COMBINED	As **HARD** or **SOFT** which occurs first	**COMBINED**	As **HARD** or **SOFT** which occurs first

The channel is armed when the following conditions are met:

- The acquisition sequence is running
- The previous acquisition phase is completed
- The next acquisition phase is due to come at the next TE

Grabber Triggering Examples

Many useful grabber triggering combinations are possible. Some of them are exemplified.

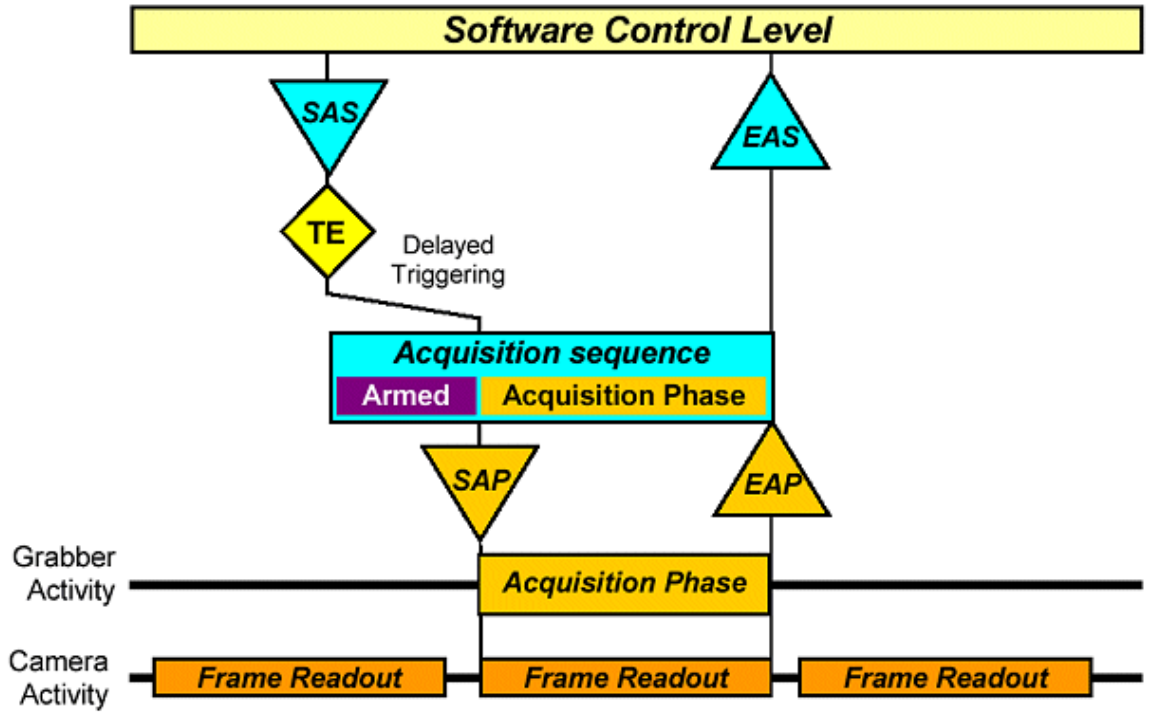
Trigger Modality	Operation			
	SYNC	VCAM CTL	WEB	PAGE
Immediate single-phase	Example 1	Example 2 without phase overlapping	N/A	No example
Armed single-phase	No example	Example 3 without phase overlapping	N/A	Example 4 with phase overlapping
Immediate multiple-phase	N/A	Example 5 without phase overlapping	N/A	No example
		Example 6 with phase overlapping		
		Example 7 without phase overlapping		
Immediate sustained	Example 9	No example	Example 10	No example
Armed sustained	No example	Example 11	Example 12	No example

Triggering Example 1

Immediate single-phase Area-scan, synchronous scanning

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	Irrelevant
`TrigMode`	**IMMEDIATE**
`NextTrigMode`	Irrelevant
`SeqLength_Fr`	**1**



This example shows the simplest software-controlled way to grab an image from a synchronous camera.

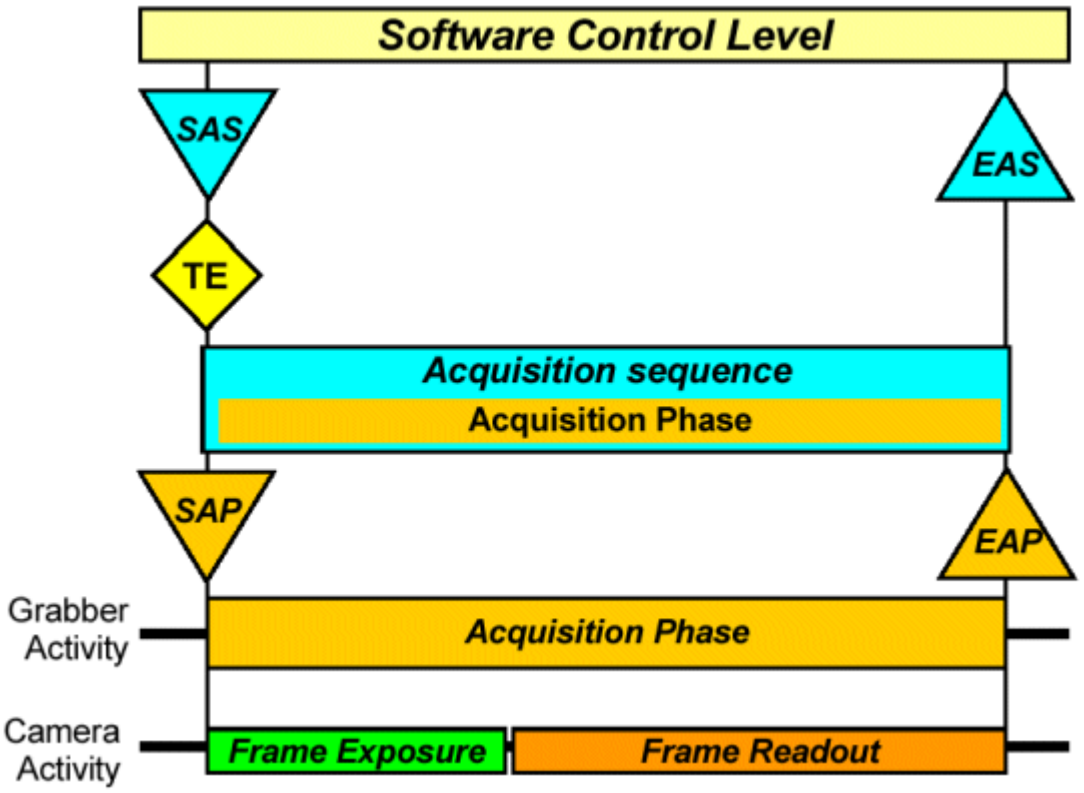
The trigger event (TE) is directly caused by the software-issued SAS event. Due to the synchronous nature of the camera, there is an uncontrolled delay between the SAS and SAP events.

Triggering Example 2

Immediate single-phase
Area-scan, asynchronous reset
Without phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	**FORBID**
`TrigMode`	**IMMEDIATE**
`NextTrigMode`	Irrelevant
`SeqLength_Fr`	**1**



This example shows the simplest software-controlled way to grab an image from an asynchronously reset camera.

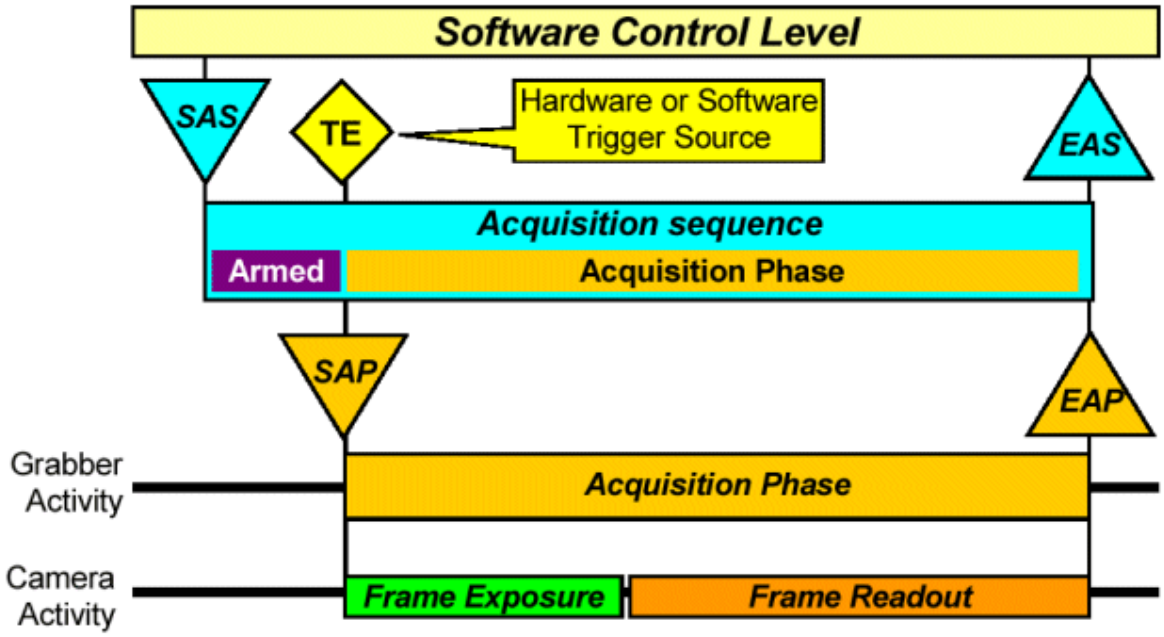
The trigger event (TE) is directly caused by the software-issued SAS event. There can be a substantial delay between the SAS and SAP events due to the software overhead implied in opening the acquisition sequence.

Triggering Example 3

Armed single-phase
Area-scan, asynchronous reset
Without phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	**FORBID**
`TrigMode`	**HARD** **SOFT** **COMBINED**
`NextTrigMode`	Irrelevant
`SeqLength_Fr`	**1**



This example shows the way to grab an image from an asynchronously reset camera using an external frame trigger.

The trigger event (TE) is sourced by a hardware- or software-issued event. There is no substantial delay between the source and SAP events.

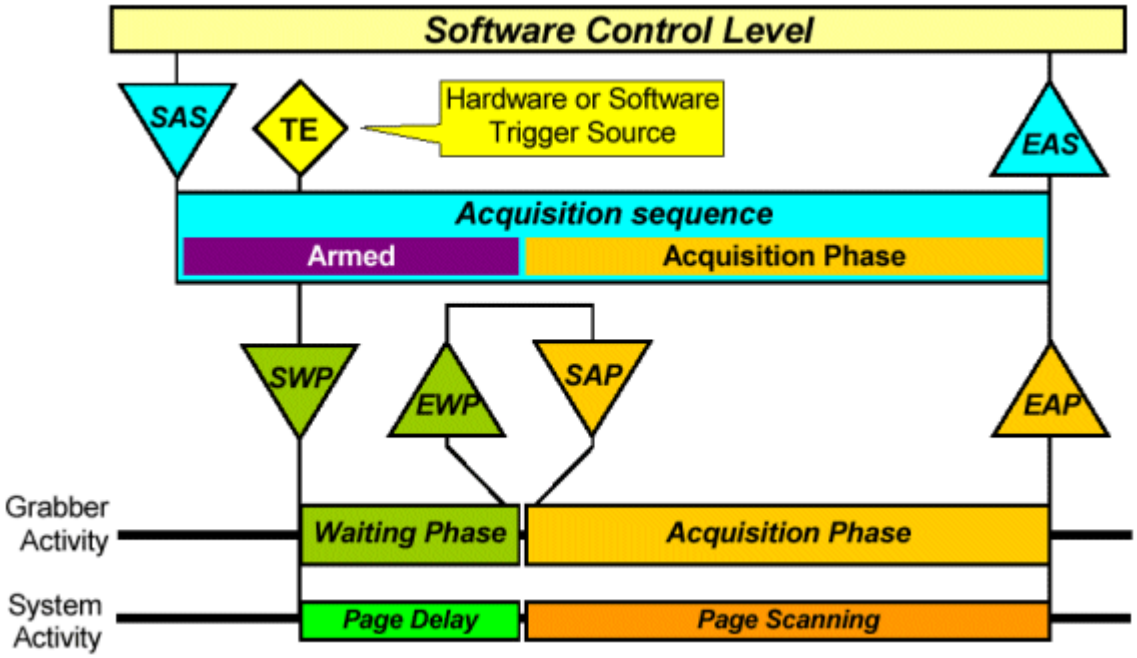
Use this triggering mode with `TrigMode` set to **SOFT** for software- controlled grab if a tight timing is to be met. The acquisition sequence has to be previously armed.

Triggering Example 4

Armed single-phase
Line-scan, page mode
With phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	Irrelevant
`TrigMode`	**HARD** **SOFT** **COMBINED**
`NextTrigMode`	Irrelevant
`SeqLength_Fr`	**1**



This example shows the way to grab the image of a discrete object through a line-scan camera using an external page trigger.

The trigger event (TE) is sourced by a hardware- or software-issued event. There is no substantial delay between the source and SWP events.

Use this triggering mode with `TrigMode` set to **SOFT** for software-controlled grab if a tight timing is to be met. The acquisition sequence has to be previously armed.

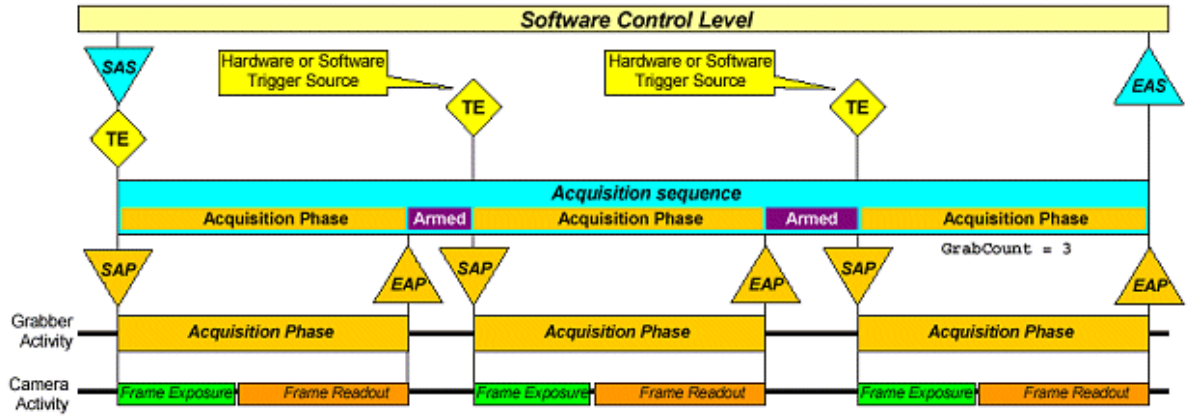
Set the `PageDelay_Ln` parameter to zero when the page delay is not necessary.

Triggering Example 5

Immediate multiple-phase Area-scan, asynchronous reset Without phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	**FORBID**
`TrigMode`	**IMMEDIATE**
`NextTrigMode`	**HARD** **SOFT** **COMBINED**
`SeqLength_Fr`	**>1**



This example shows the way to grab several successive images from an asynchronously reset camera using an external frame trigger.

The initial trigger event (TE) is directly caused by the software-issued SAS event. There can be a substantial delay between the SAS and SAP events due to the software overhead implied in opening the acquisition sequence.

The subsequent trigger events are sourced by a hardware- or software-issued event. There is no substantial delay between the source and subsequent SAP events.

Use this triggering mode with `NextTrigMode` set to **SOFT** for software-controlled subsequent grabs if a tight timing is to be met.

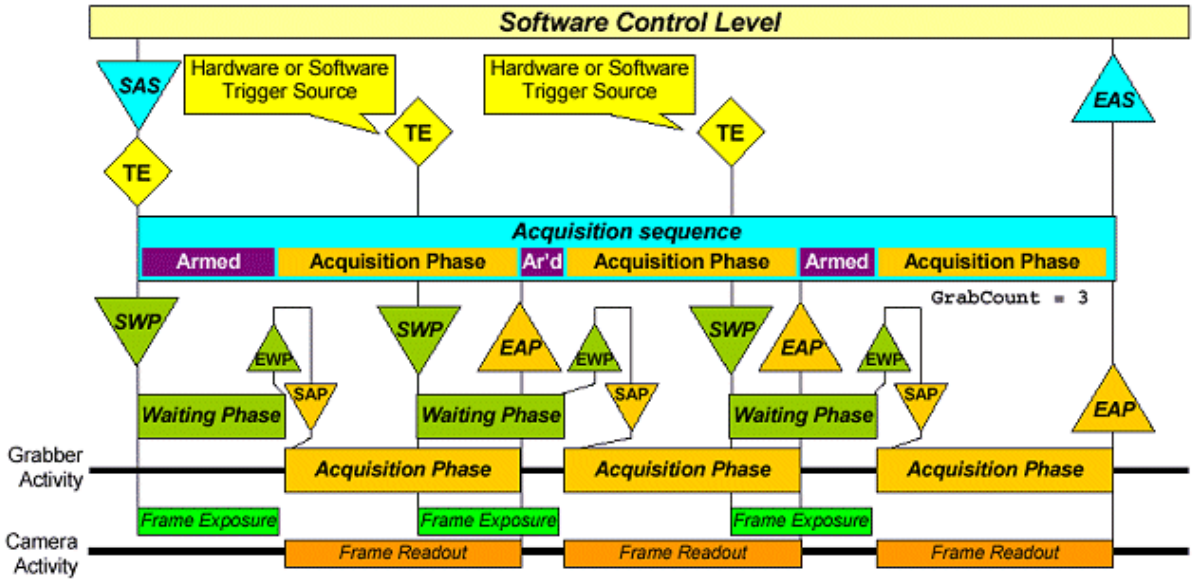
Specify the number of images to be acquired in a row with the parameter `SeqLength_Fr`. To acquire an undefined number of images, set `SeqLength_Fr` to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

Triggering Example 6

Immediate multiple-phase
Area-scan, asynchronous reset
With phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	**ALLOW**
`TrigMode`	**IMMEDIATE**
`NextTrigMode`	**HARD** **SOFT** **COMBINED**
`SeqLength_Fr`	**MC_INDETERMINATE, >1**



This example shows the way to grab several successive images from an asynchronously reset camera using an external frame trigger.

The initial trigger event (TE) is directly caused by the software-issued SAS event. There can be a substantial delay between the SAS and SWP events due to the software overhead implied in opening the acquisition sequence.

The subsequent trigger events are sourced by a hardware- or software-issued event. There is no substantial delay between the source and subsequent SAP events.

Use this triggering mode with `NextTrigMode` set to **SOFT** for software-controlled subsequent grabs if a tight timing is to be met.

Specify the number of images to be acquired in a row with the parameter `SeqLength_Fr`. To acquire an undefined number of images, set `SeqLength_Fr` to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

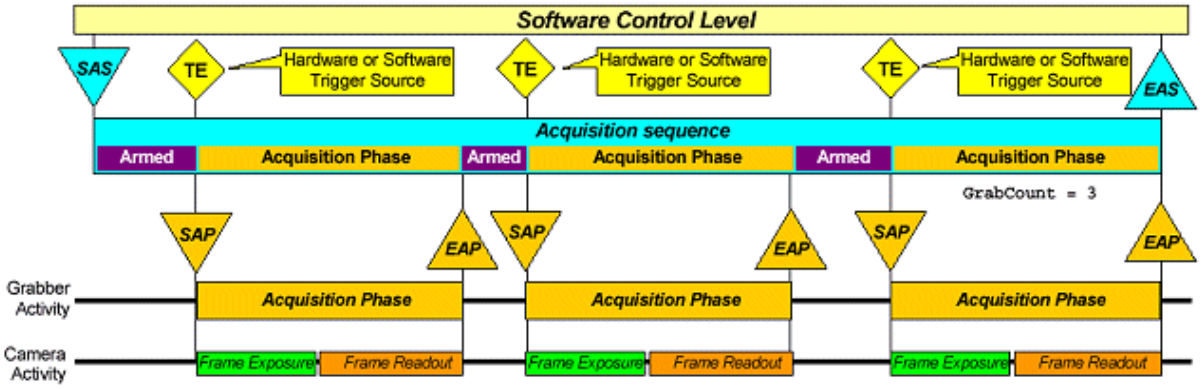
Triggering Example 7

Armed multiple-phase
Area-scan, asynchronous reset
Without phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	**FORBID**
`TrigMode`	**HARD** **SOFT** **COMBINED**

NextTrigMode	**HARD** **SOFT** **COMBINED**
SeqLength_Fr	**MC_INDETERMINATE, >1**



This example shows the way to grab several successive images from an asynchronously reset camera using an external frame trigger.

All trigger events (TE) are sourced by a hardware- or software-issued event. There is no substantial delay between the source and SAP events.

Use this triggering mode with TrigMode and NextTrigMode set to **SOFT** for software-controlled grabs if a tight timing is to be met.

Specify the number of images to be acquired in a row with the parameter SeqLength_Fr. To acquire an undefined number of images, set SeqLength_Fr to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

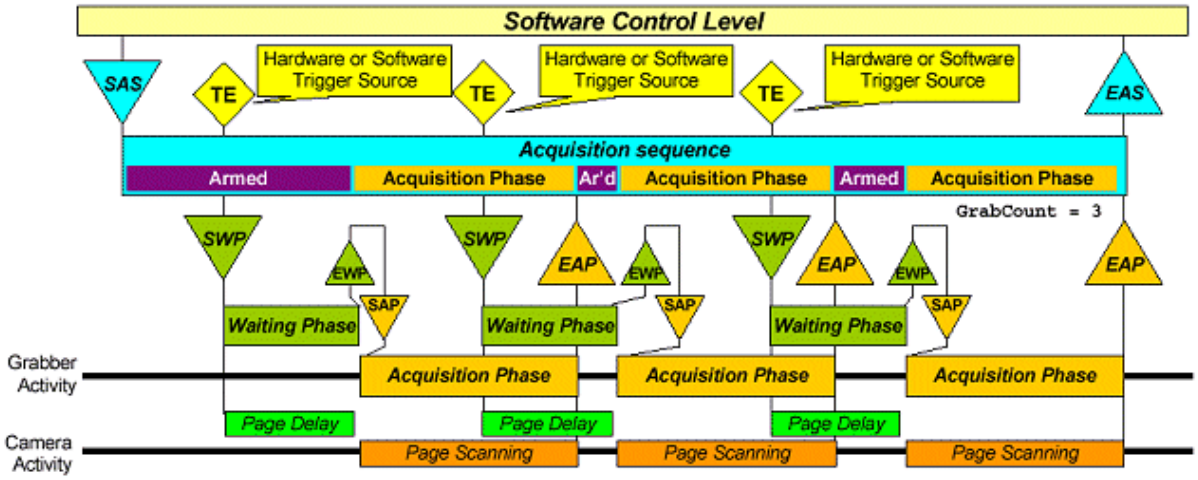
Triggering Example 8

Armed multiple-phase
Line-scan, page mode
With phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
ExposeOverlap	Irrelevant
TrigMode	**HARD** **SOFT** **COMBINED**

NextTrigMode	**HARD** **SOFT** **COMBINED**
SeqLength_Fr	**MC_INDETERMINATE, >1**



This example shows the way to grab several successive images of discrete objects through a line-scan camera using an external page trigger.

All trigger events (TE) are sourced by a hardware- or software-issued event. There is no substantial delay between the source and SWP events.

Use this triggering mode with `TrigMode` and `NextTrigMode` set to **SOFT** for software-controlled grabs if a tight timing is to be met.

Set the `PageDelay_Ln` parameter to zero when the page delay is not necessary.

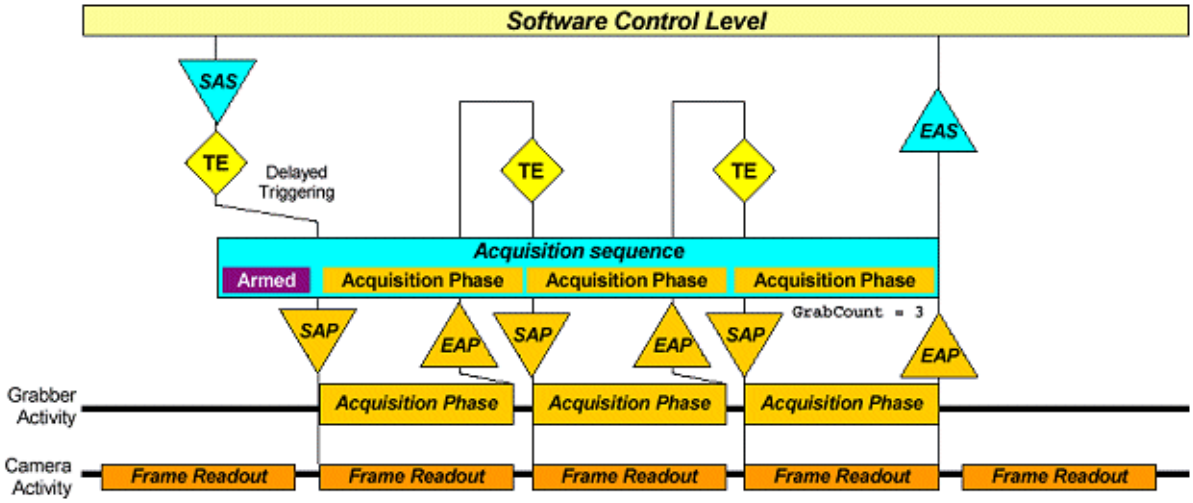
Specify the number of images to be acquired in a row with the parameter `SeqLength_Fr`. To acquire an undefined number of images, set `SeqLength_Fr` to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

Triggering Example 9

Immediate sustained
Area-scan, synchronous scanning

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
ExposeOverlap	Irrelevant
TrigMode	**IMMEDIATE**
NextTrigMode	**REPEAT**
SeqLength_Fr	**MC_INDETERMINATE, >1**



This example shows the simplest software-controlled way to grab several successive images image from a synchronous camera.

The initial trigger event (TE) is directly caused by the software-issued SAS event. Due to the synchronous nature of the camera, there is an uncontrolled delay between the SAS and SAP events.

Each subsequent TE event is conceptually sourced by the end of the preceding acquisition phase.

Specify the number of images to be acquired in a row with the parameter `SeqLength_Fr`. To acquire an undefined number of images, set `SeqLength_Fr` to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

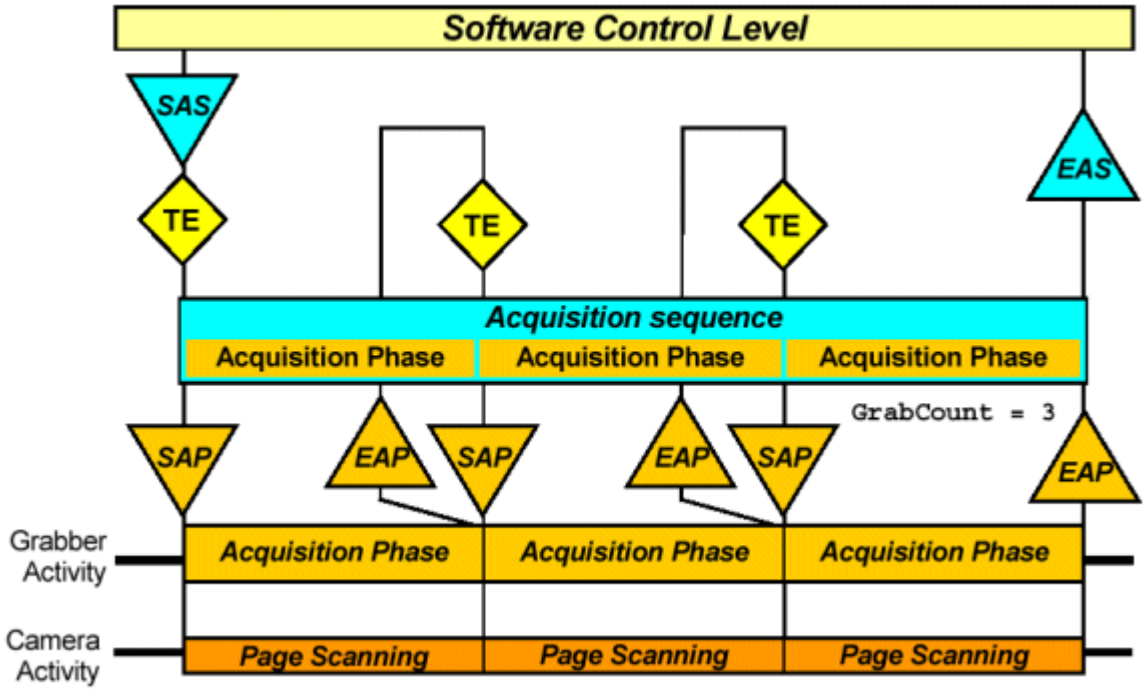
This mode of operation is often called "live acquisition".

Triggering Example 10

Immediate sustained Line-scan, WEB mode

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	Irrelevant
`TrigMode`	**IMMEDIATE**
`NextTrigMode`	**REPEAT**
`SeqLength_Fr`	**MC_INDETERMINATE, >1**



This example shows the software-controlled way to grab an arbitrarily long image of a continuous web through a line-scan camera.

The initial trigger event (TE) is directly caused by the software-issued SAS event. There can be a substantial delay between the SAS and SAP events due to the software overhead implied in opening the acquisition sequence.

Each subsequent TE event is conceptually sourced by the end of the preceding acquisition phase.

Specify the number of pages to be acquired in a row with the parameter `SeqLength_Fr`. To acquire an undefined number of pages, set `SeqLength_Fr` to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

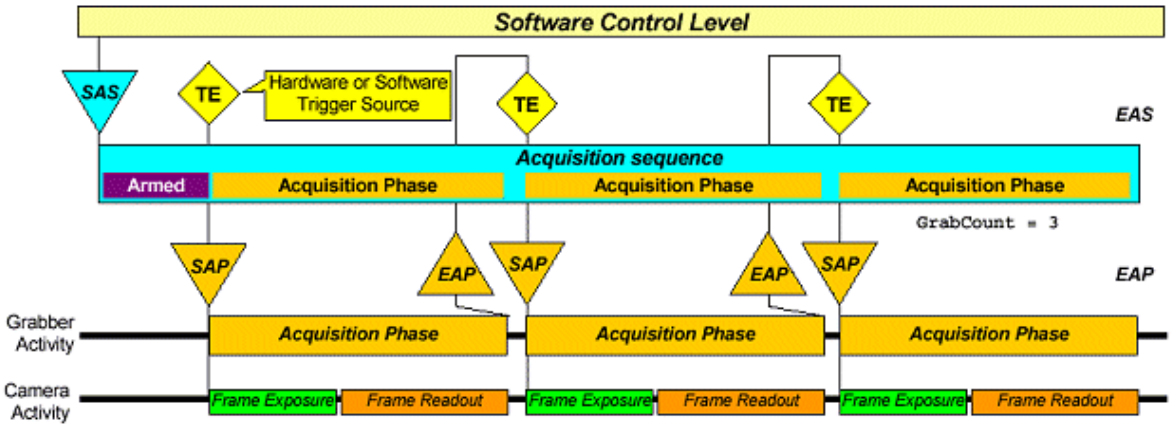
Triggering Example 11

Armed sustained
Area-scan, asynchronous reset
Without phase overlapping

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
`ExposeOverlap`	**FORBID**

TrigMode	**HARD** **SOFT** **COMBINED**
NextTrigMode	**REPEAT**
SeqLength_Fr	**MC_INDETERMINATE, >1**



This example shows the way to grab several successive images from an asynchronously reset camera using a single initial external frame trigger.

The trigger event (TE) is sourced by a hardware- or software-issued event. There is no substantial delay between the source and SAP events.

Each subsequent TE event is conceptually sourced by the end of the preceding acquisition phase.

Use this triggering mode with TrigMode set to **SOFT** for software-controlled grab if a tight timing is to be met.

Specify the number of images to be acquired in a row with the parameter SeqLength_Fr. To acquire an undefined number of images, set SeqLength_Fr to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

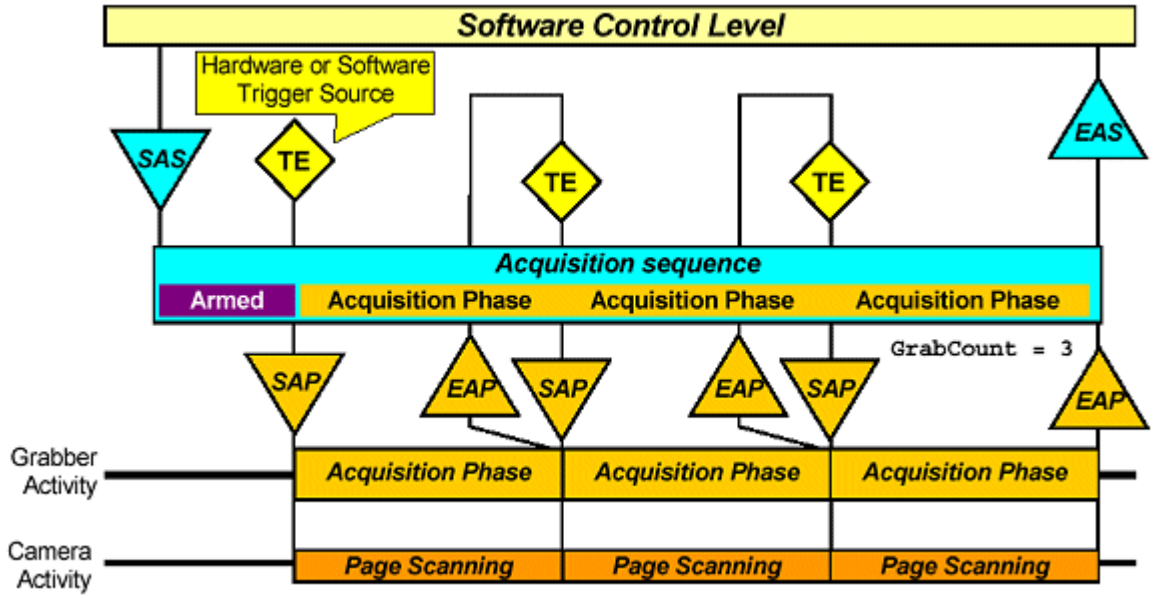
Triggering Example 12

Armed sustained
Line-scan, WEB mode

This triggering modality is reached when the following combination of parameters is set:

MultiCam parameter	Value
ExposeOverlap	Irrelevant

TrigMode	**HARD** **SOFT** **COMBINED**
NextTrigMode	**REPEAT**
SeqLength_Fr	**MC_INDETERMINATE, >1**



This example shows the way to grab an arbitrarily long image of a continuous web through a line-scan camera using an external trigger

The trigger event (TE) is sourced by a hardware- or software-issued event. There is no substantial delay between the source and SAP events.

Each subsequent TE event is conceptually sourced by the end of the preceding acquisition phase.

Use this triggering mode with TrigMode set to **SOFT** for software- controlled grab if a tight timing is to be met.

Specify the number of pages to be acquired in a row with the parameter SeqLength_Fr. To acquire an undefined number of pages, set SeqLength_Fr to **MC_INDETERMINATE**. In this case, the sequence will be stopped by returning the MultiCam channel to the idle state.

Frame Trigger Violation

It relates to the fact that a trigger pulse cannot be immediately served under some circumstances. A trigger pulse is used as TE source when TrigMode or NextTrigMode are set to **HARD**, **SOFT** or **COMBINED**.

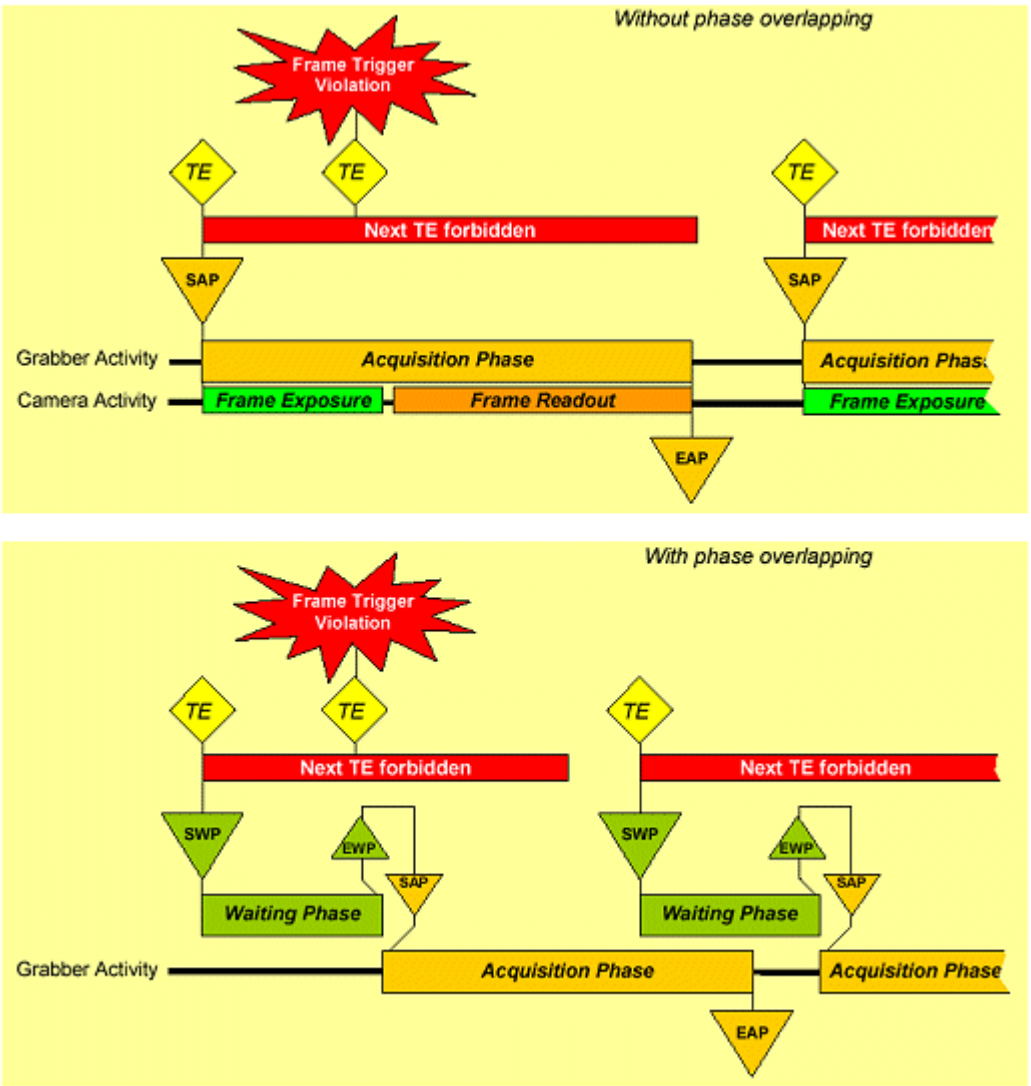
The mechanism equally applies to area-scan and to line-scan. In the latter case, "frame trigger" should be understood as "page trigger".

The mechanism applies only when the acquisition sequence is activated.

The cases when a trigger pulse cannot be served are as follows:

- If phase overlapping is not allowed, a violation is detected when the trigger pulse occurs during the acquisition phase
- If phase overlapping is allowed, a violation is detected when the trigger pulse occurs too early during the acquisition phase for the waiting phase to complete before the end of the acquisition phase

This is illustrated by the following drawing:



When a trigger pulse occurs at a forbidden time, the Frame Trigger Violation MultiCam signal is issued.

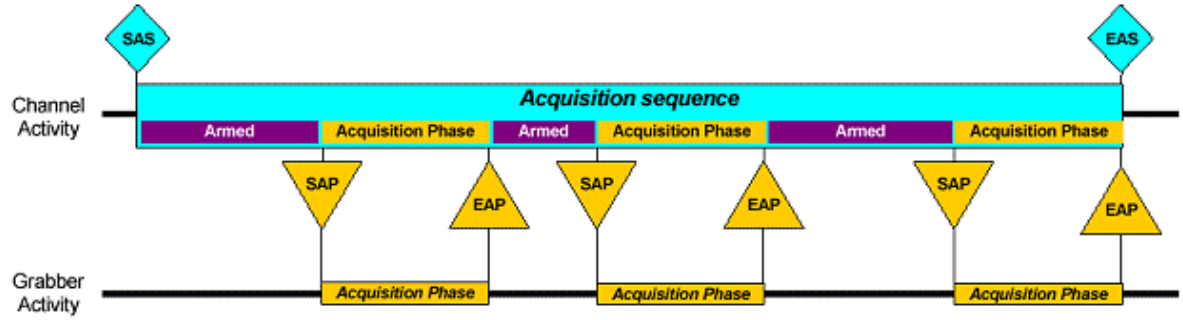
Refer to Signaling and Exceptions to learn how to handle a MultiCam signal.

An informative MultiCam parameter called `FrameTriggerViolation` is also provided. It is incremented when violations occur.

6. Understanding MultiCam Acquisition Sequences

6.1. Acquisition Sequence

An acquisition sequence is a succession of acquisition phases. Each acquisition phase corresponds to a period of time when a channel uses a grabber to accept an individual image from a camera, and to deliver it to the destination cluster.



The circumstances directing the occurrence of acquisition phases are explained in MultiCam Triggering.

The acquisition sequence fundamentally belongs to a channel. This means that each channel is allowed to activate one acquisition sequence for its own purpose. When a channel has activated one acquisition sequence, it is not allowed to activate an additional one.

If two acquisition sequences are needed for a same camera, two channels will be created. This makes possible to acquire images from an identical source under different modalities, such as different refresh rates or destinations.

The acquisition phases are closely matched to a grabber. This means that several acquisition sequences can simultaneously post request to a common grabber for acquisition phases to occur. In this situation, MultiCam uses the automatic switching mechanism.

6.2. Activating the Acquisition Sequence

The acquisition sequence is activated when the channel is activated while the acquisition flag is started, whichever occurs last. This event is referred to as the **SAS** (Start Acquisition Sequence).

Activating the channel is a matter of assigning the value **ACTIVE** to the `ChannelState` parameter. This is a channel-specific operation.

As long as the conjunction "channel is active" and "acquisition is started" is realized, the acquisition sequence remains alive. The channel parameters allow defining a number of useful ways to organize the occurrence of acquisition phases. In particular, repetitive or triggered acquisition phases are possible, as well as "live" operation of the channel.

6.3. Deactivating the Acquisition Sequence

There are several ways to stop an acquisition sequence. They all imply that the `ChannelState` parameter returns to **IDLE**. This event is referred to as the **EAS** (End Acquisition Sequence).

The application can stop the acquisition sequence on a per-channel basis by setting the `ChannelState` parameter to **IDLE**.

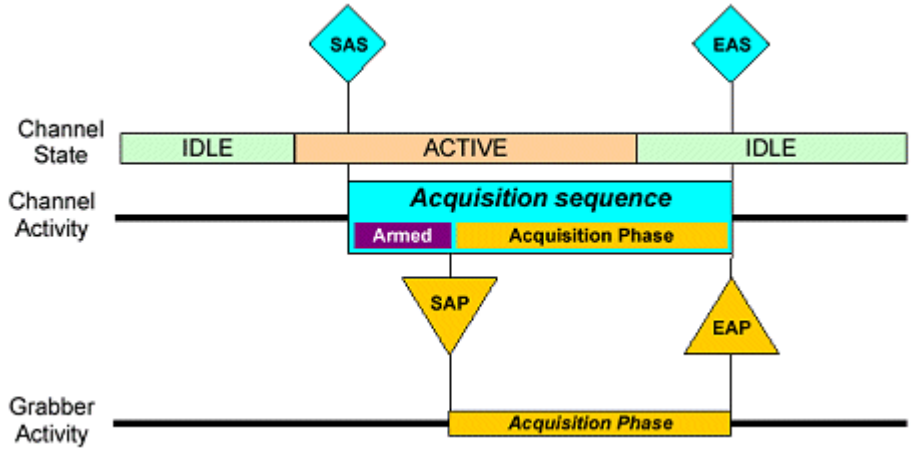
Exception signals can also force the `ChannelState` to reset to **IDLE**, namely the "Acquisition Failure".

Once the `ChannelState` is set to **IDLE**, the acquisition sequence remains alive until the completion of the current acquisition phase, if any is running. This rule is true for all stopping conditions with two exceptions:

- The stopping condition is an exception signal.
- The fundamental operational mode is the line-scan WEB mode.

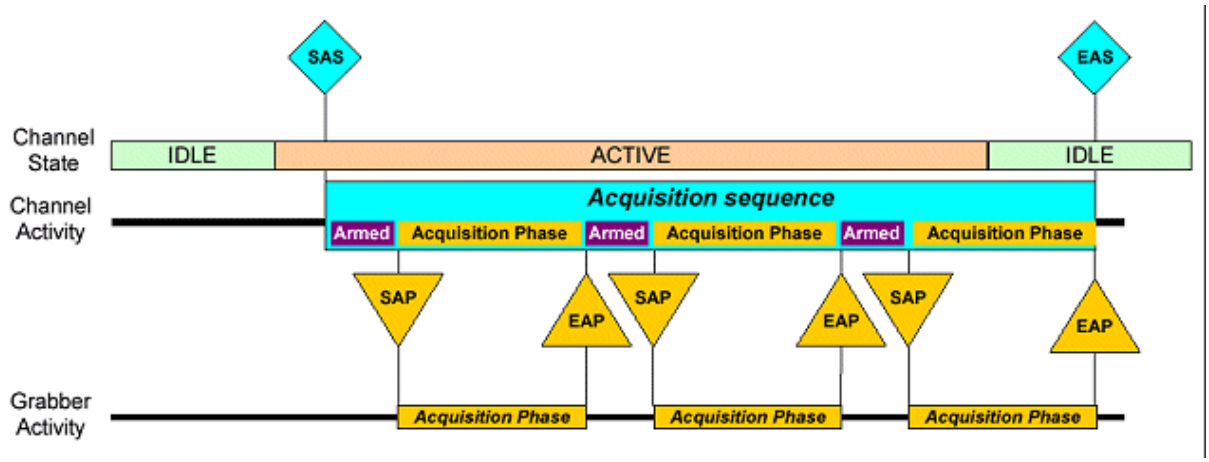
In these two cases, the acquisition sequence immediately stops, abruptly interrupting any ongoing acquisition phase.

6.4. Single-Phase Acquisition Sequence



The SAP occurrence condition is directed by the choice of the triggering mode, and by the availability of the particular grabber the channel intents to appropriate to perform the frame or page acquisition.

6.5. Multiple-Phase Acquisition Sequence



The SAP occurrence condition is directed by the choice of the triggering mode, and by the availability of the particular grabber the channel intents to appropriate to perform the frame or page acquisition.

6.6. Sustained Acquisition Sequence

In many circumstances, it is possible to derive the SAP of an acquisition phase from the EAP of the previous one.

This results in a sustained or live acquisition, with no intervening between acquisition phases.

In the area-scan case, this conceptual model represents the way to implement the acquisition of a synchronous-mode camera frame sequence.

In the line-scan case, this conceptual model represents the way to implement the acquisition of a continuous flow of video lines. There is no gap between acquisition phases.

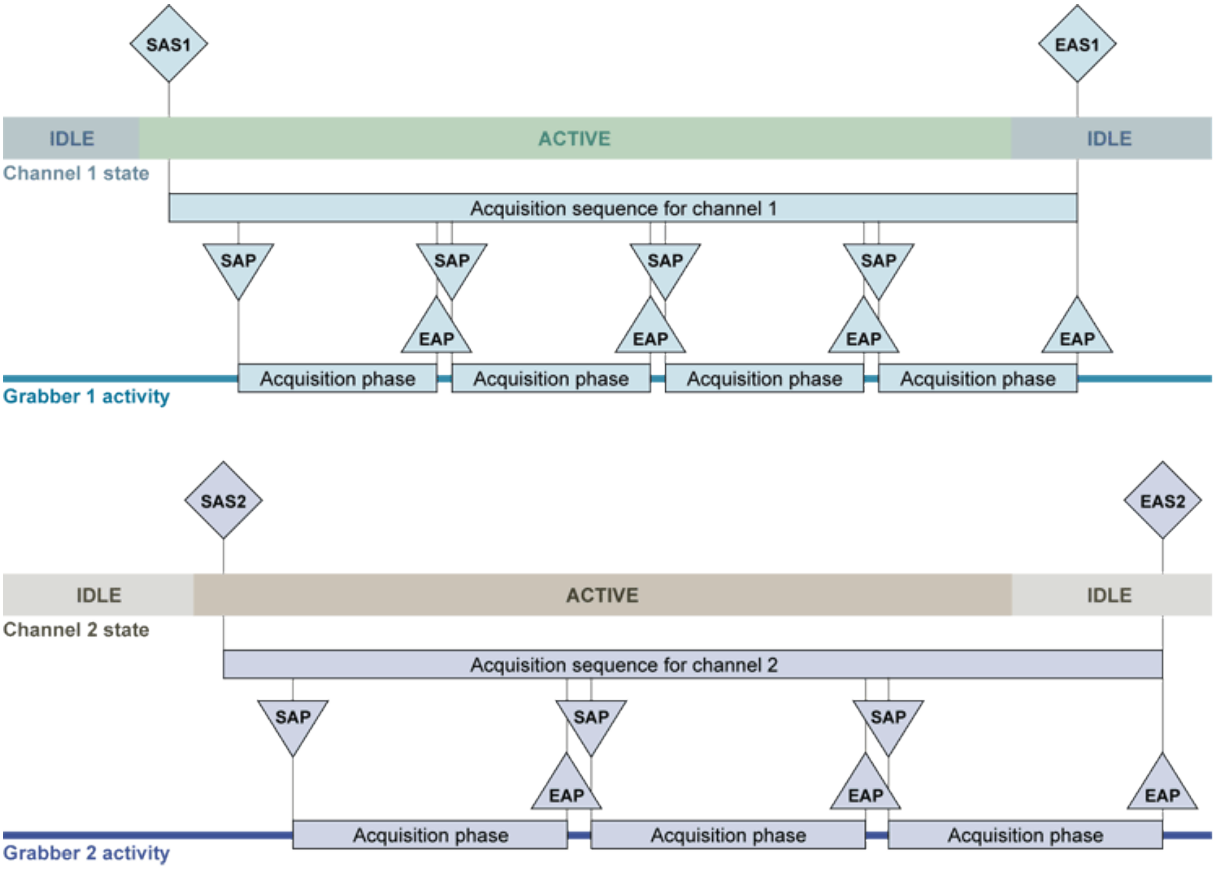
6.7. Concurrent Acquisition Mode

There can be enough resources in a frame grabber board to simultaneously support more than one camera.

For instance, consider the case of a board used in the 2-2-0 topology. Two dual-channel cameras can be connected to the board, and there are enough hardware resources to operate both cameras simultaneously.

This mode is referred to as DuoCam, because two cameras are operable at the same time. Note that the TrioCam mode also exists.

One channel will be created for each camera, and a specific grabber is permanently assigned to each channel. There is no need for switching, because there is no resource competition.



DuoCam acquisition mode

Each channel operates at a frame rate which is set by the camera capabilities and the external triggering requirement. There is no impeding interaction from another channel.

6.8. Switched Acquisition Mode

There can be too few resources in a frame grabber board to simultaneously support more than one camera.

For instance, consider the case of a Gamma board used in the 3-3-0 topology. Two RGB triple-channel can be connected to the board, but there are not enough hardware resources to operate both cameras simultaneously.

Then the frame grabber board operates in the switched mode. The cameras are alternatively time-multiplexed in the best way to satisfy the application needs.

Another way to express the situation is to say that the channels compete for a common grabber. Two channels are created, one for each camera, but only one grabber is available.

The MultiCam system embeds an automatic switching mechanism to attribute the grabber to any channel requesting an acquisition in a programmable fair way. For more details, refer to understanding automatic switching.

It should be understood that the switching operation implies that occasionally a requesting channel will have to wait its turn before the acquisition phase be served by the grabber.

7. Understanding Automatic Switching

The automatic switching mechanism is available for all Domino and Grablink frame grabbers.

It allows:

- Creation of more than one channel with different settings, per camera.
- Creation of more than one channel per grabber.
- Usage of all the 4 video inputs of the Domino Alpha 2.

The channels are activated as usual by the `ChannelState` parameter and MultiCam handles the resource allocation automatically, without the need to delete and create the channels at each time.

7.1. Getting ChannelState

Getting the parameter `ChannelState` returns one of the four following values: **ORPHAN**, **IDLE**, **READY** or **ACTIVE**.

Orphan

When a channel is in the ORPHAN state, it has no grabber associated with it. Hence, image acquisition is not possible immediately.

However the channel exists and all its parameters can be set or get freely.

Idle

When a channel is in the IDLE state, it has a grabber associated with it. Hence, image acquisition is possible immediately.

In this state, MultiCam may reassign the grabber resources automatically to another channel.

Ready

When a channel is in the READY state, it has a grabber associated with it. Hence, image acquisition is possible immediately.

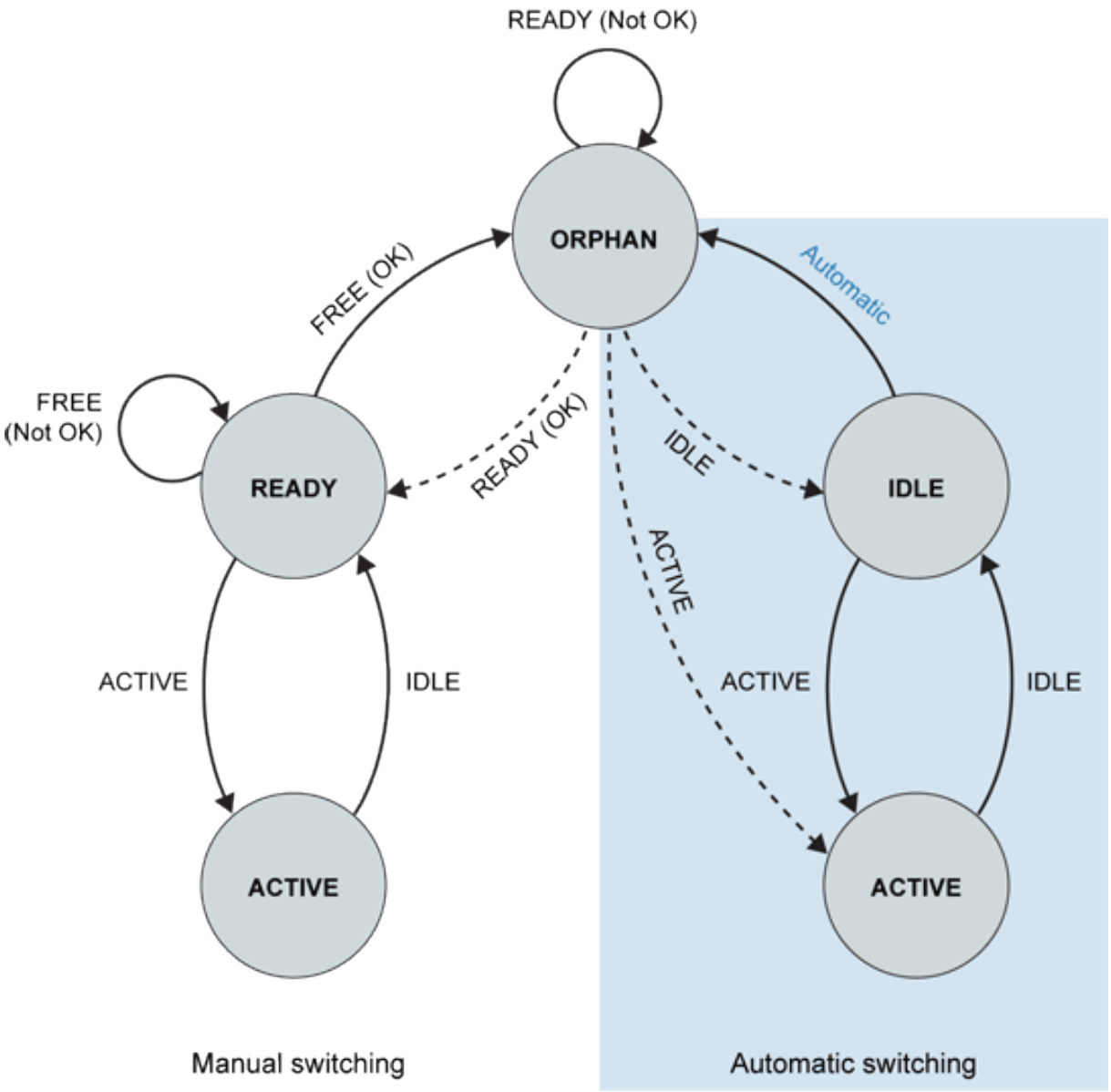
In this state, MultiCam cannot reassign the grabber resources automatically to another channel.

Active

When a channel is in the ACTIVE state, it has a grabber associated with it and performs an image acquisition sequence.

In this state, MultiCam cannot reassign the grabber resources automatically to another channel.

7.2. ChannelState Transitions



ChannelState transitions

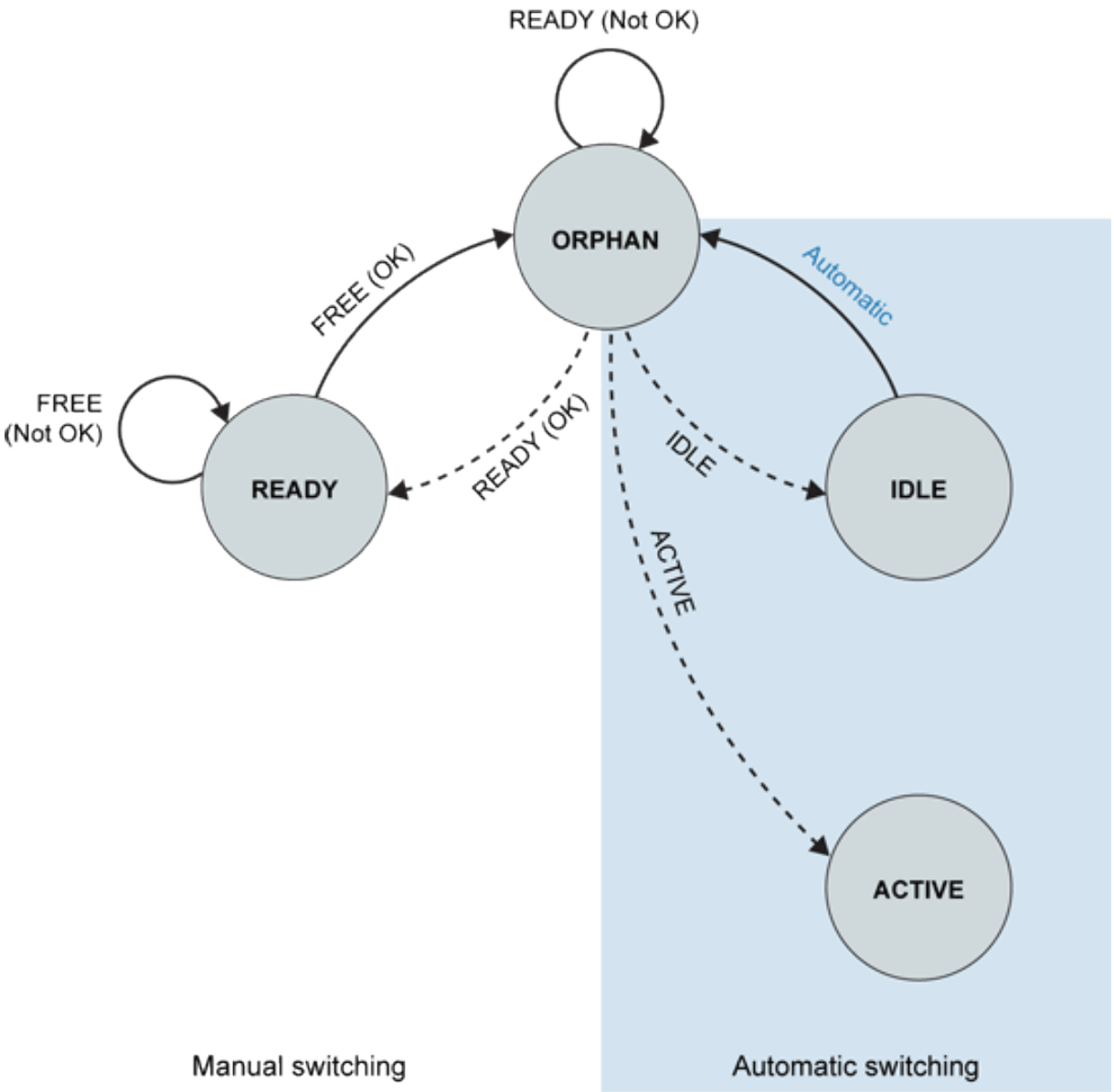
A dashed arrow represents a transition that may last until all required grabber resources are available.

A continuous arrow represents an immediate transition.

Initial State

At channel creation, `ChannelState` is **ORPHAN**.

To Leave ORPHAN State



To acquire images, a channel needs a grabber.

To associate a grabber to a channel, the parameter `ChannelState` should be set to one of the following: **IDLE**, **READY** or **ACTIVE**.

Setting ChannelState to IDLE

On setting the `ChannelState` parameter to **IDLE**, MultiCam associates the required grabber resources to the channel. `ChannelState` changes to **IDLE**.

The command will terminate successfully when all required grabber resources are available. It might take some time to complete.

This command implicitly selects the automatic switching mechanism for this channel.

Setting ChannelState to READY

On setting the `ChannelState` parameter to **READY**, MultiCam associates the required grabber resources to the channel.

`ChannelState` changes to **READY** if all required grabber resources are available. Otherwise, `ChannelState` remains **ORPHAN**. This action takes a short time to complete. By simply reading back `ChannelState`, the application is informed of the success or failure of the setting.

As long as `ChannelState` remains **ORPHAN**, the application can retry, as many times as necessary, to set `ChannelState` to **READY**.

This command implicitly selects the manual switching mechanism for this channel.

Setting ChannelState to ACTIVE

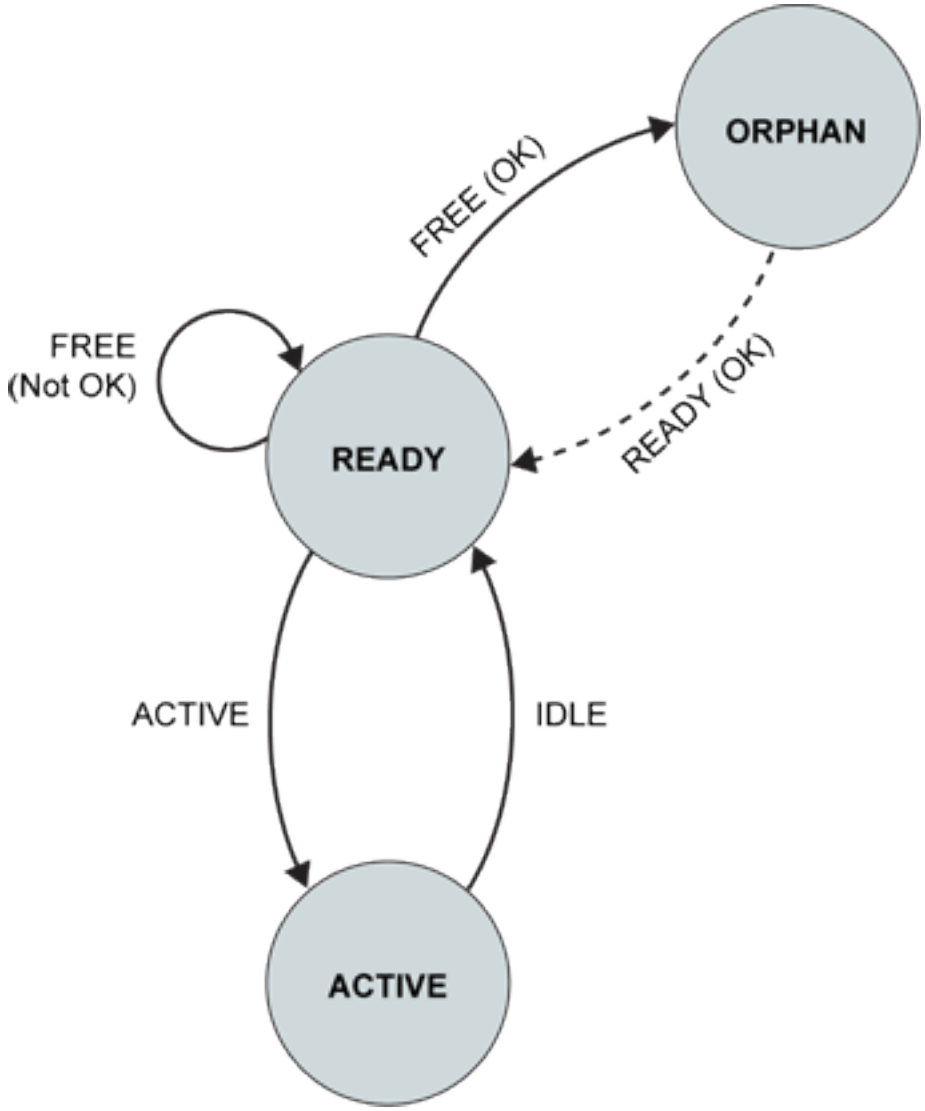
On setting the `ChannelState` parameter to **ACTIVE**, MultiCam associates the required grabber resources to the channel, and starts the image acquisition sequence. `ChannelState` changes to **ACTIVE**.

The command will terminate successfully when all required grabber resources are available. It might take some time to complete.

This command implicitly selects the automatic switching mechanism for this channel.

Manual Switching

To Leave READY State



Setting ChannelState to ACTIVE

Setting the parameter `ChannelState` to **ACTIVE** starts an acquisition sequence.

Because all required grabber resources are available, the command takes effect immediately.

Setting ChannelState to FREE

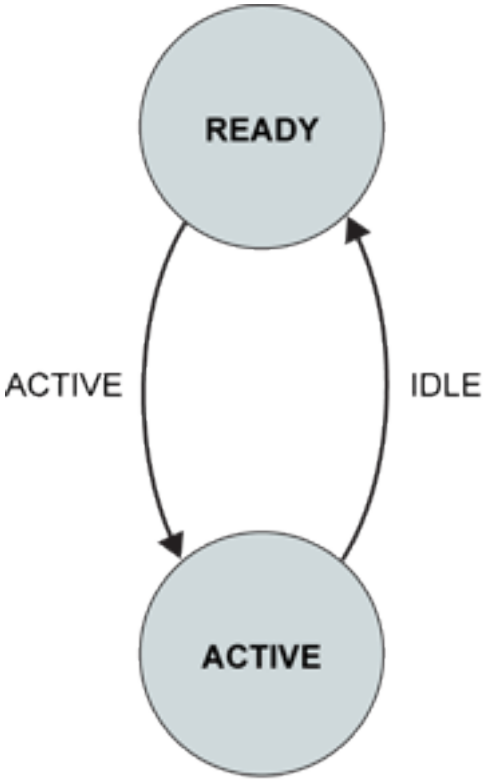
Setting the parameter `ChannelState` to **FREE** releases the associated grabber resources.

If the resources release is possible, the command takes effect immediately and `ChannelState` becomes **ORPHAN**.

It may happen, on Domino Alpha 2 with two cameras in master synchronization mode, that `ChannelState` remains **READY**. By simply reading back `ChannelState`, the application is informed of the success or failure of the setting.

As long as `ChannelState` remains **READY**, the application can retry as many times as necessary to set `ChannelState` to **FREE**.

To Leave ACTIVE State



Automatically

The end of a sequence (EAS) occurs automatically after `SeqLength_Fr` acquisition phases.

Consequently, `ChannelState` changes to **READY**.

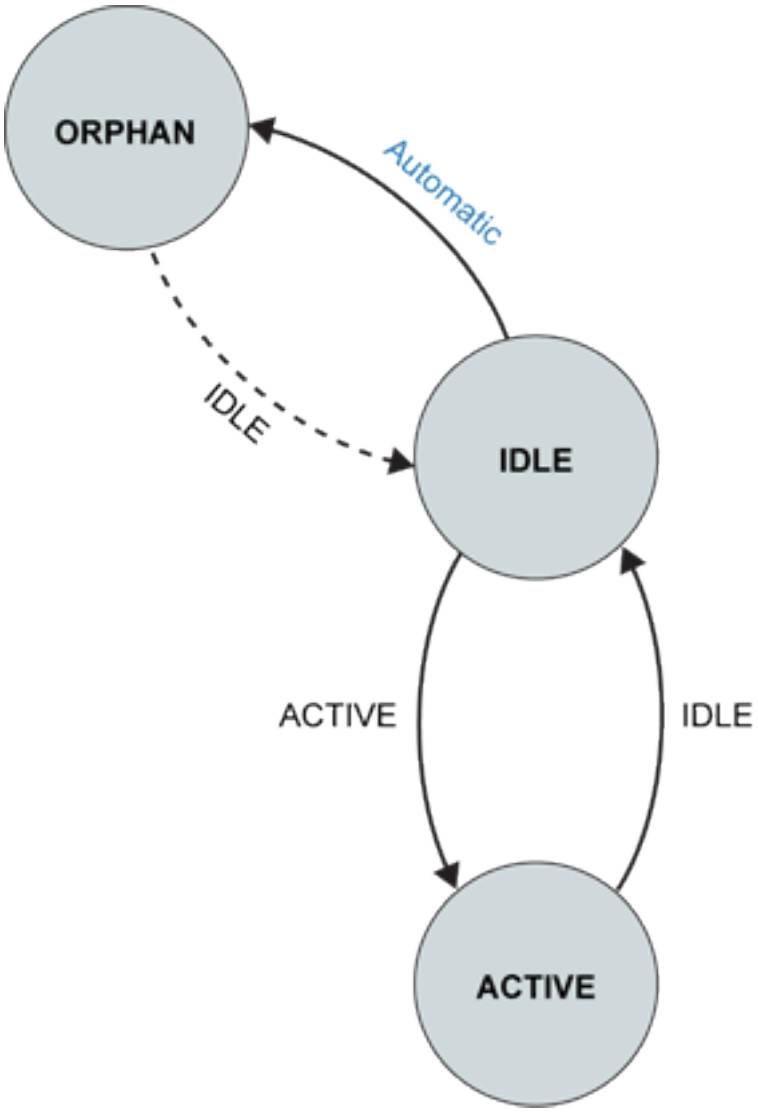
Setting ChannelState to IDLE

The end of an acquisition sequence (EAS) can be forced by setting `ChannelState` to **IDLE**. According to the channel settings, the last acquisition phase finishes and afterwards `ChannelState` is **READY**.

Automatic Switching

Refer to the "Automatic Switching" above section in *D402EN-MultiCam User Guide* PDF document.

To Leave IDLE State



Automatically

MultiCam automatically makes the transition IDLE to ORPHAN, when needed.

This occurs when another channel requires some involved grabber resources.

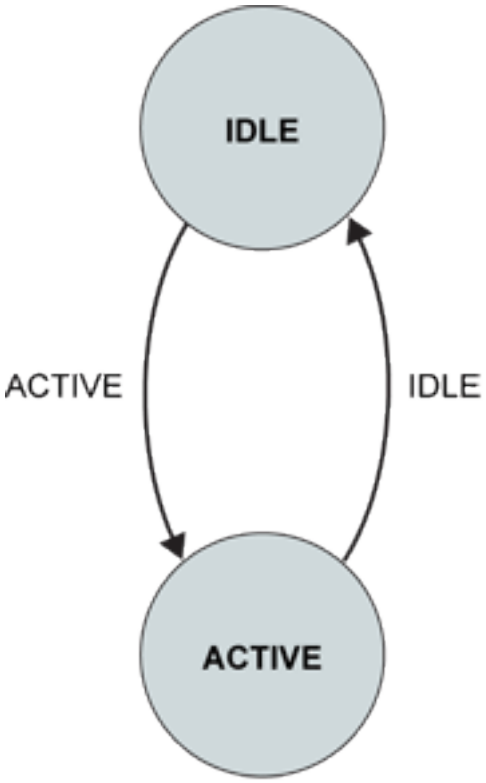
This is a part of the automatic switching mechanism.

Setting ChannelState to ACTIVE

On setting the parameter `ChannelState` to **ACTIVE**, MultiCam immediately starts the image acquisition sequence. The channel's state changes to **ACTIVE**.

This is a part of the automatic switching mechanism.

To Leave ACTIVE State



Automatically

The end of a sequence (EAS) occurs automatically after `SeqLength_Fr` acquisition phases.

Consequently, `ChannelState` changes to **IDLE**.

Setting ChannelState to IDLE

The end of an acquisition sequence (EAS) can be forced by setting `ChannelState` to **IDLE**. According to the channel settings, the last acquisition phase finishes and afterwards `ChannelState` is **IDLE**.

8. Understanding Camera Specification

8.1. Camera Class Specification

Camera Imaging Basic Geometry

The **Imaging** parameter declares the basic geometry of the camera feeding the channel. MultiCam supports three basic geometries:

- **AREA**: The area-scan cameras are based on 2D imager(s) and deliver 2D data frames
- **LINE**: The non-TDI line-scan cameras are based on 1D imager(s) and deliver 1D data lines
- **TDI**: The TDI line-scan cameras are based on 2D imager(s) and deliver 1D data lines

TDI stands for Time Delay Integration. TDI line-scan cameras exhibit an increased sensitivity since the light integration spans over multiple line periods.

MultiCam distinguishes TDI and non-TDI line-scan cameras since TDI line-scan cameras have specific requirements for their control. However, both are line-scan cameras and share a common set of acquisition modes.

Camera Spectral Sensitivity

The **Spectrum** parameter declares the spectral sensitivity of the camera feeding the channel. MultiCam supports three spectral sensitivities:

- **BW**: The black/white cameras are delivering a monochrome video signal built from an imager having a spectral response covering the visible light spectrum
- **IR**: The infrared cameras are delivering a monochrome video signal built from an imager having a spectral response covering the infra-red light spectrum
- **COLOR**: The color cameras are delivering a multi-component video signal built from either a single imager having Color Filter Arrays or from multiple imagers having different spectral responses

For the frame grabber point of view, BW and IR are equivalent. The wording "monochrome cameras" designates both classes of cameras.

The class of color cameras is further divided into several sub-classes. See Color Camera Specification.

Camera Data Transfer Method

The **DataLink** parameter declares the data transfer method of the camera feeding the channel. MultiCam supports three data transfer methods:

- **COMPOSITE**: The "composite video" cameras deliver the video data as an analog composite video signal. The signal can be:
  - CVBS including Color, Video, Blanking, and Sync
  - VBS including Video, Blanking, and Sync
- **ANALOG**: The "analog industrial" cameras deliver the video data as an analog video signal. The signal can be:
  - Single lane VBS including Video, Blanking, and Sync
  - Single lane VB including Video, Blanking
  - Three lane analog RGB with Sync on Green
- **CAMERALINK**: The "Camera Link" cameras deliver digital video data complying with the Camera Link standard.

  Note: *There is a 1-to-1 match between the values of **DataLink** and the Euresys frame grabber series: **COMPOSITE** for Picolo series, **ANALOG** for Domino series and **CAMERALINK** for Grablink series.*

8.2. Color Camera Specification

Camera Color Analysis Method

The **ColorMethod** parameter declares the color analysis method of the camera feeding the channel. MultiCam supports the following color analysis methods:

- **NONE**: The "monochrome" cameras have no color analysis method.
- **RGB**: The "RGB" cameras deliver the video data as three separate color components respectively named Red, Green, Blue.
- **BAYER**: The "Bayer CFA" cameras deliver the raw video obtained from a Bayer CFA imager.
- **PRISM**: The "PRISM " cameras are a sub-class of RGB cameras using a 3-CCD prism assembly ensuring a perfect registration of all color components of a pixel.
- **TRILINEAR**: The "trilinear" cameras are a sub-class of non-TDI line-scan RGB color cameras using a triple line-array imager and delivering un-registered color components.

Camera Color Pattern Filter Alignment

The **ColorRegistration** parameter declares the alignment of the color pattern filter of the camera feeding the channel.

MultiCam supports the following filter alignments for **Bayer CFA cameras**: **GB**, **BG**, **RG**, **GR**.

MultiCam supports the following filter alignments for **trilinear cameras**: **RGB**, **GBR**, **BRG**.

Color Gap

The **ColorGap** parameter declares the gap between adjacent sensing lines of the trilinear camera feeding the channel.

This gap is expressed as a number of pixel pitches along the line. It is an unchangeable geometrical feature of the trilinear sensor.

8.3. Camera Timing Specification

Camera Active Window

The select-level channel parameter `Camera` belonging to the Camera Specification category declares the unique name of the camera feeding the channel.

Expert-level parameters constituting the Camera Timing category are automatically adjusted to a consistent set of values each time the `Camera` parameter is updated.

These values define the timing behavior of the signals issued or received by the camera as long as the video sequence is concerned. The camera timing parameters specify the temporal size and location of the Camera Active Window inside the scanning format.

The camera active window is a rectangular region for area-scan cameras and a linear region for line-scan cameras. By definition, the camera active window is the timing area where luminance data are due to be provided by the camera.

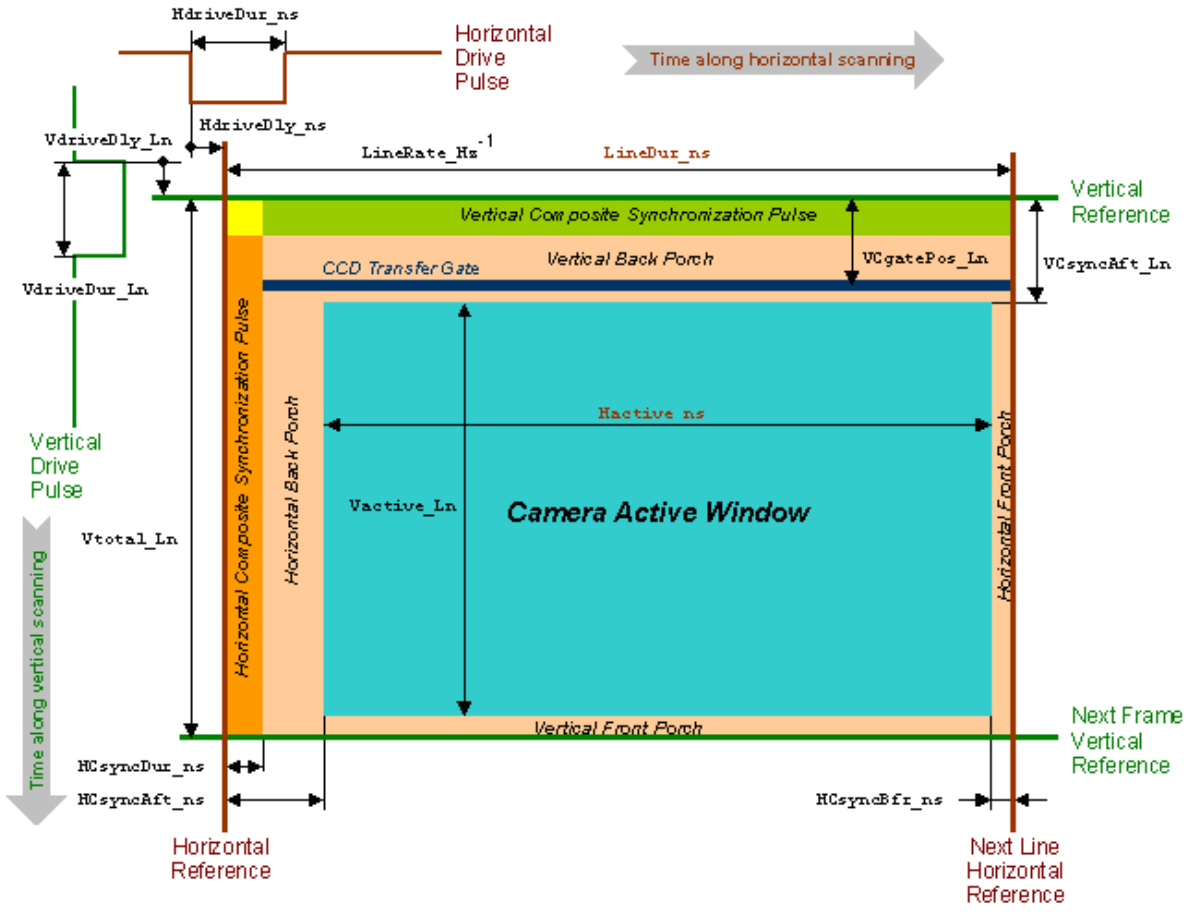
Occasionally, a user wishes to adapt the MultiCam operation to a non-officially supported camera. The recommended way to achieve this is to set the `Camera` parameter to a value specifying a supported camera having characteristics close to the targeted camera. The next step is to adjust a few timing parameters to reflect the actual behavior of this camera. The MultiCam system will automatically adapt all grabber-related parameters to acquire video frames adapted to the newly defined camera active window.

See also How to Define the Grabbing Window?

Area-Scan Analog Camera

The following diagram depicts the Camera Active Window of an analog area-scan camera used in analog or master synchronization mode. For the sake of simplicity, a progressive video format is represented. The camera timing parameter structure supports the interlaced format as well.

The diagram includes the possible use of horizontal and vertical drive pulses sent to the camera (genlocking the camera).



Camera active window of an analog area-scan camera

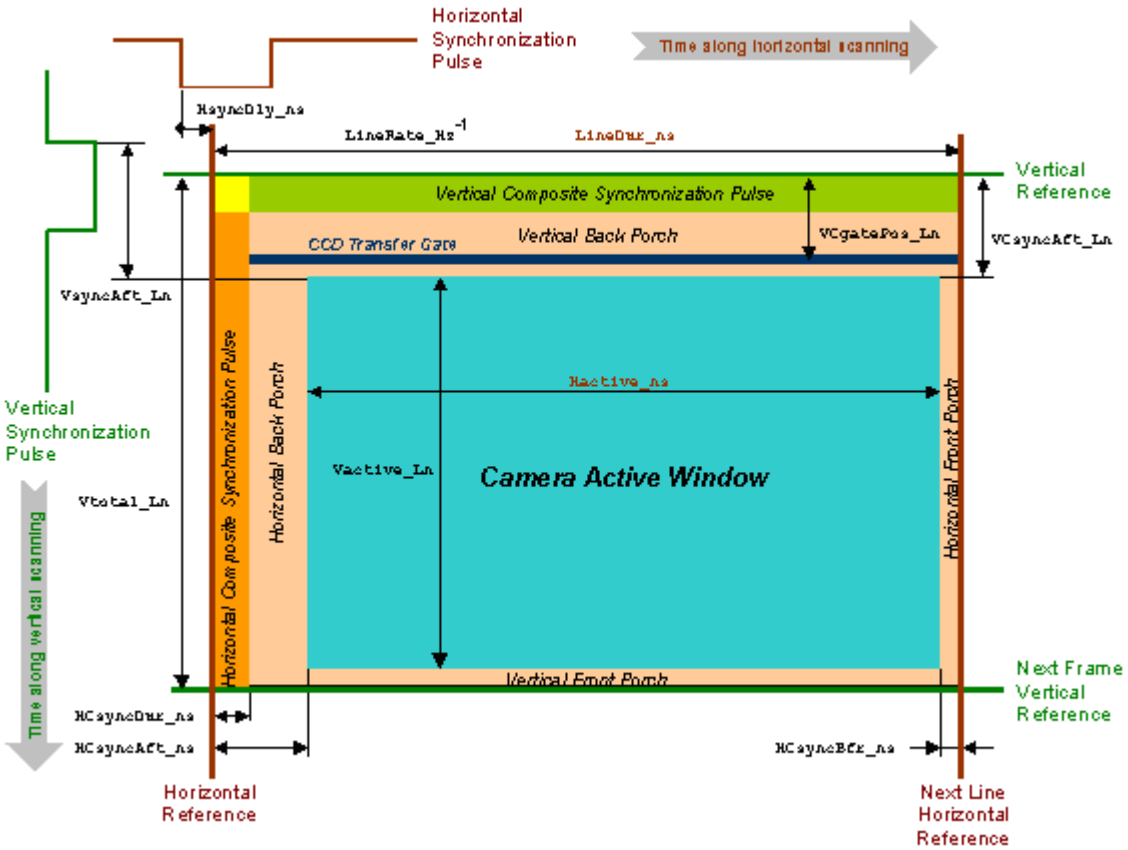
	Controlling parameters	Feedback parameters
Sizing scanning format	`LineRate_Hz` `Vtotal_Ln`	`LineDur_ns`
Locating active window	`HCsyncAft_ns` `VCsyncAft_Ln`	
Sizing active window	`HCsyncBfr_ns` `Vactive_Ln`	`Hactive_ns`

Defining composite horizontal sync	HCsyncDur_ns	
Defining horizontal drive	HdriveDur_ns HdriveDly_ns	
Defining vertical drive	VdriveDur_Ln VdriveDly_Ln	

Area-Scan Analog Camera with Pixel Clock

The following diagram depicts the Camera Active Window of an analog area-scan camera used in digital synchronization mode.

The diagram includes the specification of the horizontal and vertical separate synchronization pulses delivered by the camera, which are needed along with the pixel clock to support the digital synchronization.



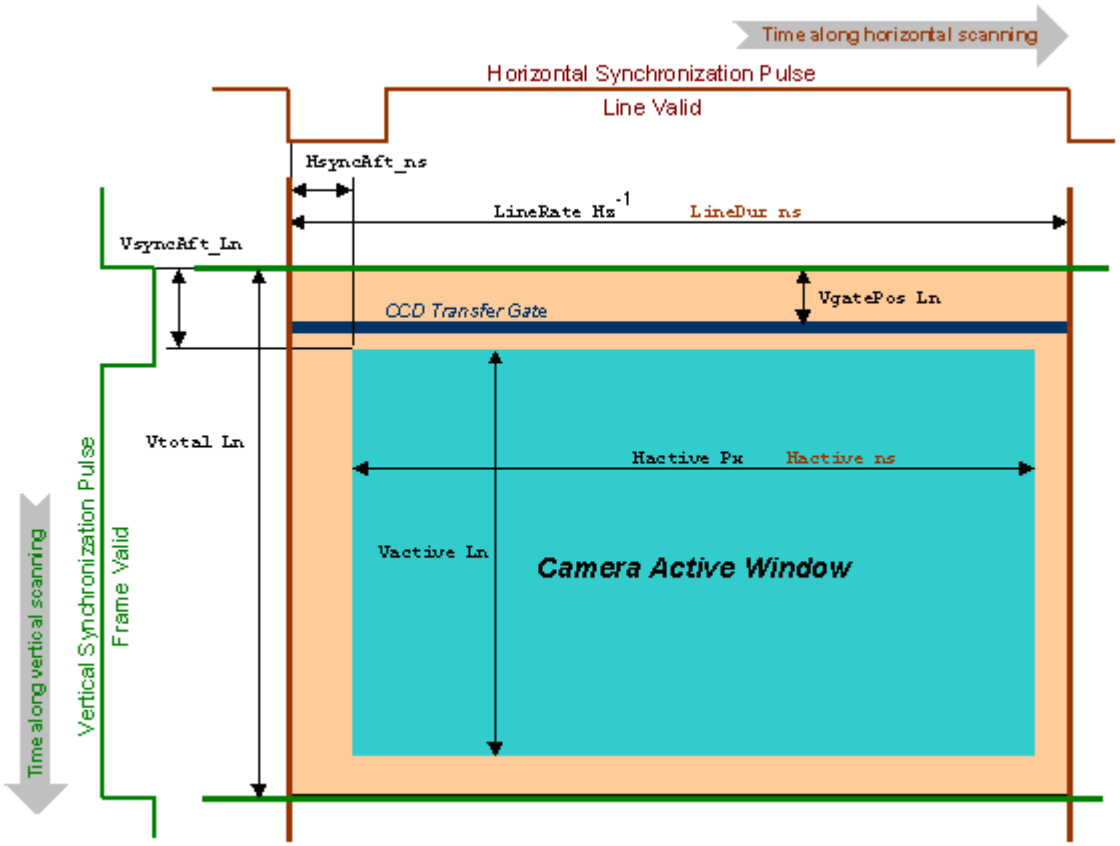
Camera active window of an analog area-scan camera with pixel clock

	Controlling parameters	Feedback parameters
Sizing scanning format	LineRate_Hz Vtotal_Ln	LineDur_ns
Locating active window	HsyncDly_ns HCsyncAft_ns	
Sizing active window	HCsyncBfr_ns Vactive_Ln	Hactive_ns
Defining composite horizontal sync	HCsyncDur_ns	
Locating transfer gate	VCsyncAft_Ln	
Defining pixel drive	PixelClk_Hz	

Area-Scan Digital Camera

The following diagram depicts the Camera Active Window of a digital area-scan camera.

The diagram includes the specification of the horizontal and vertical separate synchronization pulses delivered by the camera, which accompany the pixel clock to synchronize the frame grabber.



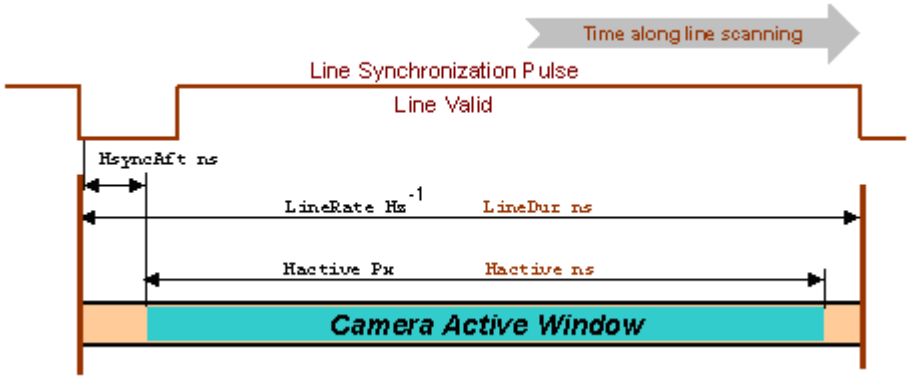
Camera active window of a digital area-scan camera

	Controlling parameters	Feedback parameters
Sizing scanning format	LineRate_Hz Vtotal_Ln	LineDur_ns
Locating active window	HCsyncAft_ns VCsyncAft_Ln	
Sizing active window	Hactive_Px Vactive_Ln	Hactive_ns
Defining pixel drive	PixelClk_Hz	

Line-Scan Digital Camera

The following diagram depicts the Camera Active Window of a digital line-scan camera.

The diagram includes the specification of the horizontal synchronization pulse, often called "Line Valid", delivered by the camera, which accompanies the pixel clock to synchronize the frame grabber.



Camera active window of a digital line-scan camera

	Controlling parameters	Feedback parameters
Line scanning rate	LineRate_Hz	LineDur_ns
Active window width	Hactive_Px	Hactive_ns
Active window position	HsyncAft_Tk	
Pixel rate	PixelClk_Hz	

8.4. Camera Upstream Specification

There are two distinct functions inside the camera, referred to as `Expose` and `Readout`. Each function is described by a corresponding MultiCam parameter.

One or two upstream lines control both functions.

- When one line is enough to control both functions, it is designated by `Reset`.
- When two lines are enough to control both functions, they are designated by `Reset` and `AuxReset`.

Line-Scan Cameras Upstream Specification

Six combinations of `Expose` and `Readout` are supported:

Expose	Readout	Description
INTCTL	**INTCTL**	Free-running, camera controlled exposure time
INTPRM	**INTCTL**	Free-running, permanent exposure
PLSTRG	**INTCTL**	Grabber-controlled rate, camera-controlled exposure time
PLSTRG	**PLSTRG**	Grabber-controlled rate, grabber-controlled exposure, dual signal
WIDTH	**INTCTL**	Grabber-controlled rate, grabber-controlled exposure, single signal
INTPRM	**PLSTRG**	Grabber-controlled rate, permanent exposure

Line-Scan, Free-Run

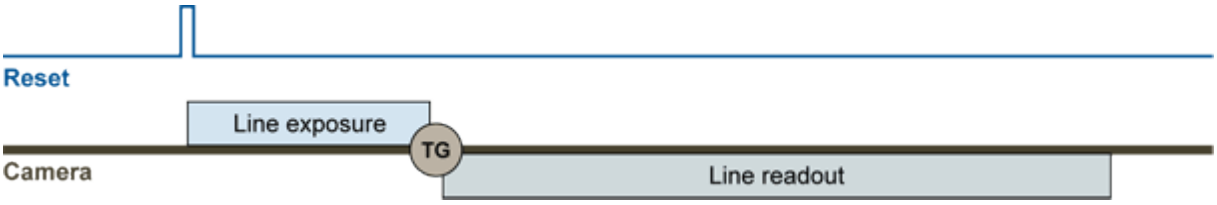
Parameter	Value	Meaning
`Expose`	**INTCTL**	The line exposure condition is totally controlled by the camera.
`Readout`	**INTCTL**	The line readout condition is totally controlled by the camera.

Line-Scan, Permanent Exposure, Free-Run

Parameter	Value	Meaning
`Expose`	**INTPRM**	The camera is such that the exposure is permanently enabled.
`Readout`	**INTCTL**	The line readout condition is totally controlled by the camera.

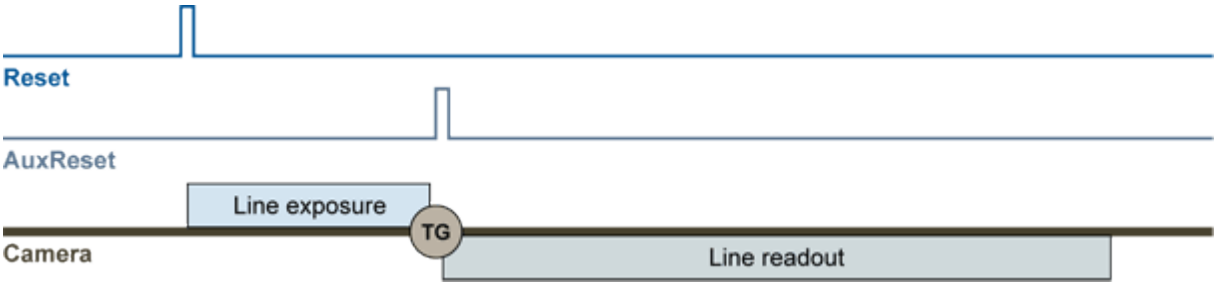
Line-Scan, Triggered

Parameter	Value	Meaning
Expose	**PLSTRG**	The line exposure condition starts upon receiving a pulse from a frame grabber issued signal.
Readout	**INTCTL**	The line readout condition is totally controlled by the camera.



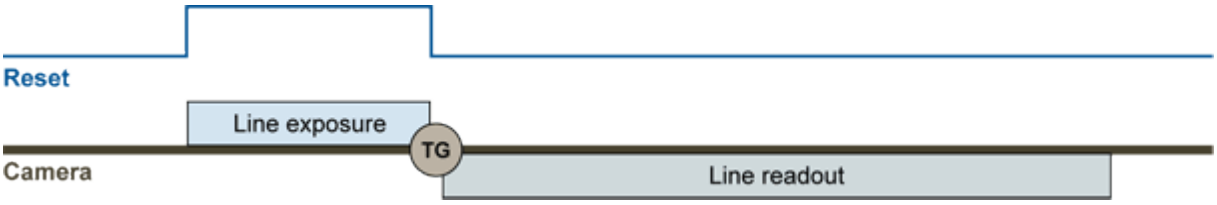
Line-Scan, Exposure Control, Dual Signal

Parameter	Value	Meaning
Expose	**PLSTRG**	The line exposure condition starts upon receiving a pulse from a frame grabber issued signal.
Readout	**PLSTRG**	The line readout condition starts upon receiving a pulse from a frame grabber issued signal.



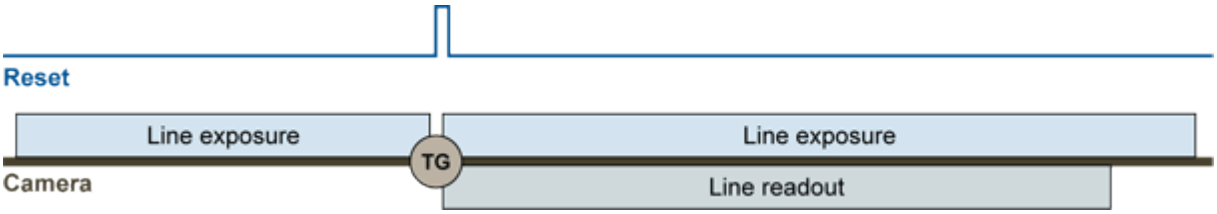
Line-Scan, Exposure Control

Parameter	Value	Meaning
Expose	**WIDTH**	The line exposure condition is controlled by the duration of a logic state issued by the frame grabber.
Readout	**INTCTL**	The line readout condition is totally controlled by the camera.



Line-Scan, Permanent Exposure, Triggered

Parameter	Value	Meaning
Expose	**INTPRM**	The camera is such that the exposure is permanently enabled
Readout	**PLSTRG**	The line readout condition starts upon receiving a pulse from a frame grabber issued signal



Area-Scan Cameras Upstream Specification

Three combinations of Expose and Readout are supported:

Expose	Readout	Description
INTCTL	**INTCTL**	Free-running, camera-controlled exposure
PLSTRG	**INTCTL**	Grabber-controlled rate, camera-controlled exposure
WIDTH	**INTCTL**	Grabber-controlled rate, grabber-controlled exposure

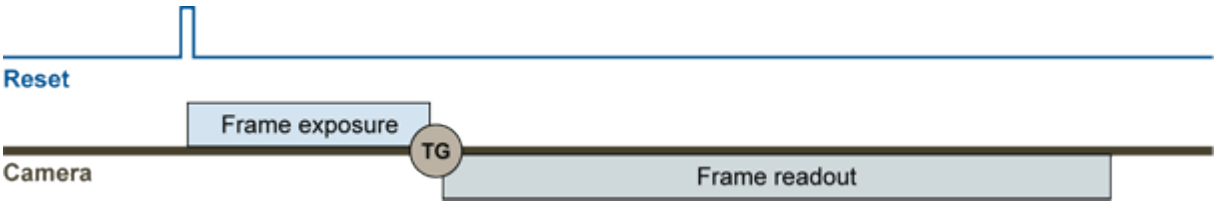
Area-Scan, Free-Run

Parameter	Value	Meaning
Expose	**INTCTL**	The frame exposure condition is totally controlled by the camera.
Readout	**INTCTL**	The frame readout condition is totally controlled by the camera.

This camera implements a SYNC operational mode.

Area-Scan, Triggered

Parameter	Value	Meaning
Expose	**PLSTRG**	The frame exposure condition starts upon receiving a pulse from a frame grabber issued signal.
Readout	**INTCTL**	The line readout condition is totally controlled by the camera.



This camera implements a VCAM operational mode.

8.5. Camera Output Structure with Grablink

The output structure of Camera Link cameras is characterized by a pair of MultiCam channel parameters:

- The Camera Link configuration, the number of taps and the bit depth is characterized by the parameter TapConfiguration
- The geometrical relationship between taps is characterized by TapGeometry

Camera Link Tap Configuration

Naming Convention

A tap configuration is designated by: **<ConfigurationClass>-<TapCount>T<BitDepth> (B<TimeSlots>)**

- **<ConfigurationClass>** is {Base, Medium, Full, Deca};
- **<TapCount>** is an integer number in range {1, 2, 3, 4, 8, 10} specifying the total number of camera taps;
- **<BitDepth>** is an integer number in range {8, 10, 12, 14, 16, 24, 30, 36, 42, 48} specifying the tap bit depth.
- **<TimeSlots>** is an integer number in range {2, 3} specifying the number of consecutive time slots required to transfer one data of each tap.
- **B<TimeSlots>** is omitted when single time slot is sufficient to serve all the taps.

Examples

Base-1T8, Medium-4T10, Base-4T8B2

Non-multiplexed tap configurations for monochrome cameras

Configuration name	Configuration class	Tap count	Bit depth	Pixels per clock cycle
BASE-1T8	Base	1	8	1
BASE-1T10	Base	1	10	1
BASE-1T12	Base	1	12	1
BASE-1T14	Base	1	14	1
BASE-1T16	Base	1	16	1
BASE-2T8	Base	2	8	2
BASE-2T10	Base	2	10	2
BASE-2T12	Base	2	12	2
MEDIUM-2T14	Medium	2	14	2
MEDIUM-2T16	Medium	2	16	2
MEDIUM-4T8	Medium	4	8	4
MEDIUM-4T10	Medium	4	10	4
MEDIUM-4T12	Medium	4	12	4
FULL-8T8	Full	8	8	8
DECA-10T8	10-tap	10	8	10

Non-multiplexed tap configurations for RGB color cameras

Configuration name	Configuration class	Tap count	Bit depth	Pixels per clock cycle
BASE-1T24	Base	1	24	1
MEDIUM-1T30	Medium	1	30	1
MEDIUM-1T36	Medium	1	36	1
MEDIUM-1T42	Medium	1	42	1
MEDIUM-1T48	Medium	1	48	1
MEDIUM-2T24	Medium	2	24	2

Multiplexed tap configurations for monochrome cameras

Configuration name	Configuration class	Tap count	Bit depth	Time slots	Pixels per clock cycle
BASE-2T14B2	Base	2	14	2	1
BASE-2T16B2	Base	2	16	2	1
BASE-4T8B2	Base	4	8	2	2
BASE-4T10B2	Base	4	10	2	2
BASE-4T12B2	Base	4	12	2	2

Multiplexed tap configurations for RGB color cameras

Configuration name	Configuration class	Tap count	Bit depth	Time slots	Pixels per clock cycle
BASE-1T24	Base	1	24	1	1
BASE-1T24B3	Base	1	24	3	0,3333
BASE-1T30B2	Base	1	30	2	0,5
BASE-1T30B3	Base	1	30	3	0,3333
BASE-1T36B2	Base	1	36	2	0,5
BASE-1T36B3	Base	1	36	3	0,3333
BASE-1T42B2	Base	1	42	2	0,5
BASE-1T42B2	Base	1	42	3	0,3333
BASE-1T48B2	Base	1	48	2	0,5
BASE-1T48B3	Base	1	48	3	0,3333
BASE-2T24B2	Base	2	24	2	1

Camera Link Tap Geometry

Image Geometrical Properties

The relevant geometrical properties required to reconstruct the image are:

- **Vantage point**: an enumerated type of parameter that specifies the position of the pixel with coordinate X=1, Y=1 in the scene. {Top-Left, Top-Right, Bottom-Left, Bottom-Right}. Default is Top-Left.
- **ImageWidth**: an integer type of parameter declaring the image width expressed in pixels.
- **ImageHeight**: an integer type of parameter declaring the image height expressed in pixels. This parameter is irrelevant in case of line-scan or TDI cameras.
- **TapGeometry**: an enumerated type of parameter that summarizes following properties for each tap:
  - **X Start**: X-coordinate of the first pixel column
  - **Y Start**: Y-coordinate if the first pixel row
  - **X End**: X-coordinate of the last pixel column
  - **Y End**: Y-coordinate of the last pixel row
  - **X Step**: difference of X-coordinates between consecutive pixel columns; X-step is positive when X-coordinates are increasing along a row; it is negative otherwise.
  - **Y Step**: difference of Y-coordinates between consecutive pixel rows; Y-step is positive when Y-coordinates is increasing at the end of a line; it is negative otherwise.

Naming Convention

A tap geometry for area-scan camera is designated by: **<TapGeometry>-<TapGeometry>**

A tap geometry for line-scan or TDI-line-scan camera is designated by : **<TapGeometry>**

TapGeometryX is designated by **<RegionX>X(<TapX>)(<ExtX>)**

- **<RegionX>**: an integer in range {1, 2, 3, 4, 8, 10} declaring the number of regions encountered across horizontal direction. A region is a rectangular area of adjacent pixels that are transferred in a raster-scan order through one or several taps. In case of several taps, adjacent pixels are transferred simultaneously.
- **<TapX>**: an integer in range {2, 4, 8, 10} declaring the number of consecutive pixels across horizontal direction that are output simultaneously from a region.
  This field is omitted when all pixels are in the same column.
- **<ExtX>**: a letter declaring the location of the pixels extractors across horizontal direction.
  This field is omitted when all pixel extractors are all at the left of each region.
  Letter E indicates that pixel extractors are at both ends of the line.
  Letter M indicates that pixel extractors are at middle of the line.
  Letter R indicates that the pixel extractors are all at the right of each region

For other values, refer to the tabular description.

TapGeometryY is designated by **<RegionY>Y(<TapY>)(<ExtY)>)**

- **<RegionY>**: an integer in range {1, 2} declaring the number of regions encountered across vertical direction.

- **<TapY>**: an integer declaring the number of consecutive pixels across vertical direction that are output simultaneously from a region.
  This field is omitted when all pixels are in the same line.
- **<ExtY>**: a letter declaring the location of the pixels extractors across vertical direction. Value E indicates that pixel extractors are at top and bottom lines.
  This field is omitted when all pixel extractors are at all the top line.

Examples

- **1X-1Y** designates an area-scan camera having 1 region across X and 1 region across Y. The pixels are delivered one at a time on a single tap beginning with the leftmost pixel of the top line.
- **1X4** designates a line-scan camera having 1 region across X. Four adjacent pixels are delivered simultaneously on 4 taps beginning with the leftmost pixels.
- **4X1** designates a line-scan camera having 4 regions across X. Four non-adjacent pixels are delivered simultaneously on 4 taps; all taps begins with the leftmost pixel.

Allocation of Taps to Ports

The camera taps are indexed using following conventional sorting rule: first by increasing values of **YStart**, then by increasing value of **XStart**. The tap **T1** is the camera tap that exhibits the smallest **XStart** for the smallest **YStart**.

Restrictions

- All zones have the same size.
- Zones are not overlapping.
- All zones have the same number of taps.
- All taps are carrying the same amount of pixels.

Tap Geometrical Properties

X properties - Tabular definitions

The following tables define the geometrical properties along the X direction.

One tap - 1X

Tap#	X Start	X End	X Step
Tap1	1	W	1

Two taps - 1X2

Tap#	X Start	X End	X Step
Tap1	1	W-1	2
Tap2	2	W	2

Two taps - 2X

Tap#	X Start	X End	X Step
Tap1	1	W/2	1
Tap2	W/2+1	W	1

Two taps - 2XE

Tap#	X Start	X End	X Step
Tap1	1	W/2	1
Tap2	W	W/2+1	-1

Two taps - 2XM

Tap#	X Start	X End	X Step
Tap1	W/2	1	-1
Tap2	W/2+1	W	1

Two taps - 2XR

Tap#	X Start	X End	X Step
Tap1	W/2	1	-1
Tap2	W	W/2+1	-1

Four taps - 1X4

Tap#	X Start	X End	X Step
Tap1	1	W-3	4
Tap2	2	W-2	4
Tap3	3	W-1	4
Tap4	4	W	4

Four taps - 2X2

Tap#	X Start	X End	X Step
Tap1	1	W/2-1	2
Tap2	2	W/2	2
Tap3	W/2+1	W-1	2
Tap4	W/2+2	W	2

Four taps – 2X2E

Tap#	X Start	X End	X Step
Tap1	1	W/2-1	2
Tap2	2	W/2	2
Tap3	W-1	W/2+1	-2
Tap4	W	W/2+2	-2

Four taps – 2X2M

Tap#	X Start	X End	X Step
Tap1	W/2-1	1	-2
Tap2	W/2	2	-2
Tap3	W/2+1	W-1	2
Tap4	W/2+2	W	2

Four taps – 4X

Tap#	X Start	X End	X Step
Tap1	1	W/4	1
Tap2	W/4+1	W/2	1
Tap3	W/2+1	3W/4	1
Tap4	3W/4+1	W	1

Four taps – 4XE

Tap#	X Start	X End	X Step
Tap1	1	W/4	1
Tap2	W/4+1	W/2	1
Tap3	3W/4	W/2+1	-1
Tap4	W	3W/4+1	-1

Four taps – 4XR

Tap#	X Start	X End	X Step
Tap1	W/4	1	-1
Tap2	W/2	W/4+1	-1
Tap3	3W/4	W/2+1	-1
Tap4	W	3W/4+1	-1

Eight taps – 1X8

Tap#	X Start	X End	X Step
Tap1	1	W-7	8
Tap2	2	W-6	8
Tap3	3	W-5	8
Tap4	4	W-4	8
Tap5	5	W-3	8
Tap6	6	W-2	8
Tap7	7	W-1	8
Tap8	8	W	8

Eight taps – 4X2

Tap#	X Start	X End	X Step
Tap1	1	W/4-1	2
Tap2	2	W/4	2
Tap3	W/4+1	W/2-1	2
Tap4	W/4+2	W/2	2
Tap5	3W/2+1	3W/4-1	2
Tap6	3W/2+2	3W/4	2
Tap7	3W/4+1	W-1	2
Tap8	3W/4+2	W	2

Eight taps – 4X2E

Tap#	X Start	X End	X Step
Tap1	1	W/4-1	2
Tap2	2	W/4	2
Tap3	W/4+1	W/2-1	2
Tap4	W/4+2	W/2	2
Tap5	3W/4-1	W/2+1	-2
Tap6	3W/4	W/2+2	-2
Tap7	W-1	3W/4+1	-2
Tap8	W	3W/4+2	-2

Eight taps – 8X

Tap#	X Start	X End	X Step
Tap1	1	W/8	1
Tap2	W/8+1	2W/8	1
Tap3	2W/8+1	3W/8	1
Tap4	3W/8+1	4W/8	1
Tap5	4W/8+1	5W/8	1
Tap6	5W/8+1	6W/8	1
Tap7	6W/8+1	7W/8	1
Tap8	7W/8+1	W	1

Eight taps – 8XR

Tap#	X Start	X End	X Step
Tap1	W/8	1	-1
Tap2	2W/8	W/8+1	-1
Tap3	3W/8	2W/8+1	-1
Tap4	4W/8	3W/8+1	-1
Tap5	5W/8	4W/8+1	-1
Tap6	6W/8	5W/8+1	-1
Tap7	7W/8	6W/8+1	-1
Tap8	W	7W/8+1	-1

Ten taps – 1X10

Tap#	X Start	X End	X Step	Y Start*	Y End*	Y Step*
Tap1	1	W-9	8	1	H	1
Tap2	2	W-8	8	1	H	1
Tap3	3	W-7	8	1	H	1
Tap4	4	W-6	8	1	H	1
Tap5	5	W-5	8	1	H	1
Tap6	6	W-4	8	1	H	1
Tap7	7	W-3	8	1	H	1
Tap8	8	W-2	8	1	H	1
Tap9	9	W-1	8	1	H	1
Tap10	10	W	8	1	H	1

Y properties - Tabular definitions

The following tables define the geometrical properties along the Y direction.

One tap - 1Y

Tap#	X Start	X End	X Step
Tap1	1	H	1

Two taps - 1Y2

Tap#	X Start	X End	X Step
Tap1	1	H-1	2
Tap2	2	H	2

Two taps - 2YE

Tap#	X Start	X End	X Step
Tap1	1	H/2	1
Tap2	H	H/2+1	-1

9. Understanding Grabber Specification

9.1. How to Define the Grabbing Window?

Camera Active Window versus Grabbing Window

The expert-level channel parameters belonging to the Camera Timing category uniquely define the Camera Active Window.

For area-scan cameras, this is a rectangular array of adjacent pixels. For line-scan cameras, this is a linear set of contiguous pixel.

Because of the definition of the camera active window, it is guaranteed that no luminance information appears on the camera video signal outside this window. However, it may happen that some pixels inside the camera active window are not bearing significant luminance information.

Here are two examples:

- The classical interlaced video standard incorporates starting and ending half-lines. For such video signals, the first and the last half-line within the active camera window are black.
- Camera may be specified by the manufacturer for a given effective width, but can exhibit a poor video response for the pixels located on the boundaries.

The frame grabber goal is to acquire a set of significant pixels from the camera active window. However, the frame grabber can impose some rules on the acquisition process. For example, a given frame grabber can restrict the number of pixels constituting a grabbed line to a multiple of four. This may be required to comply with data alignment issues inside the host computer.

Following the above considerations, it is necessary to define a Grabbing Window, which is a rectangular or linear array of pixel having a size and a position matching the camera active window, but in general not totally identical.

The `GrabWindow` parameter provides several pre-established methods to derive the grabbing window from the camera active window definition.

The NOBLACK Method

This is the recommended method to build the grabbing window.

Upon setting the `GrabWindow` parameter to the **NOBLACK** value, the grabbing window adjusts itself in size and position in such a way that:

- Only pixels bearing luminance information are acquired.
- The possible grabber restrictions are taken into account.

The `WindowX_Px` and `WindowY_Ln` parameters are adjusted to the effective size of the grabbing window.

Thereafter, the user is allowed to readjust the grabbing window position by amending `OffsetX_Px` and `OffsetY_Ln` to non-zero values.

The grabber's sampling frequency, which can differ from the camera's pixel clock frequency when analog or master synchronization is used, remains unchanged when `GrabWindow` is set to **NOBLACK**.

The NOLOSS Method

Upon setting the `GrabWindow` parameter to the **NOLOSS** value, the grabbing window adjusts itself in size and position in such a way that:

- The grabbing window automatically assumes the smallest size encompassing all camera active pixels.

The `WindowX_Px` and `WindowY_Ln` parameters are adjusted to the effective size of the grabbing window.

Thereafter, the user is allowed to readjust the grabbing window position by amending `OffsetX_Px` and `OffsetY_Ln` to non-zero values.

The STD Method

Upon setting the parameter to the STD value, the grabbing window is adjusted to a size expressed by numbers taken from a set of definite values.

The number of active columns is computed from the camera timing definition and the grabber's sampling frequency.

The number of active lines is computed from the camera timing definition and from the possible frame or field acquisition modality.

These numbers are "rounded" to standard values according to the following rules:

Active columns	Grabbing window width	Active lines	Grabbing window height
304 to 485	320	236 to 243	240

486 to 607	512	244 to 475	288
608 to 729	640	476 to 571	480
730 to 972	768	572 to 595	576
973 to 1215	1024	596 to 995	600
1216 and more	1280	996 to 1019	1000
		1020 and more	1024

Do not use STD for smaller grabbing windows.

The `WindowX_Px` and `WindowY_Ln` parameters are adjusted to the above-mentioned values.

Thereafter, the user is allowed to readjust the grabbing window position by amending `OffsetX_Px` and `OffsetY_Ln` to non-zero values.

The grabber's sampling frequency, which can differ from the camera's pixel clock frequency when analog or master synchronization is used, remains unchanged when `GrabWindow` is set to **STD**.

The MAN Method

Upon setting the parameter to the **MAN** value, the grabbing window firstly remains in the size and position previously defined, usually the default condition installed by `NOBLACK`. This method allows the user to subsequently freely adjust the grabbing window.

The size is adjusted with `WindowX_Px` and `WindowY_Ln` and the position with `OffsetX_Px` and `OffsetY_Ln`.

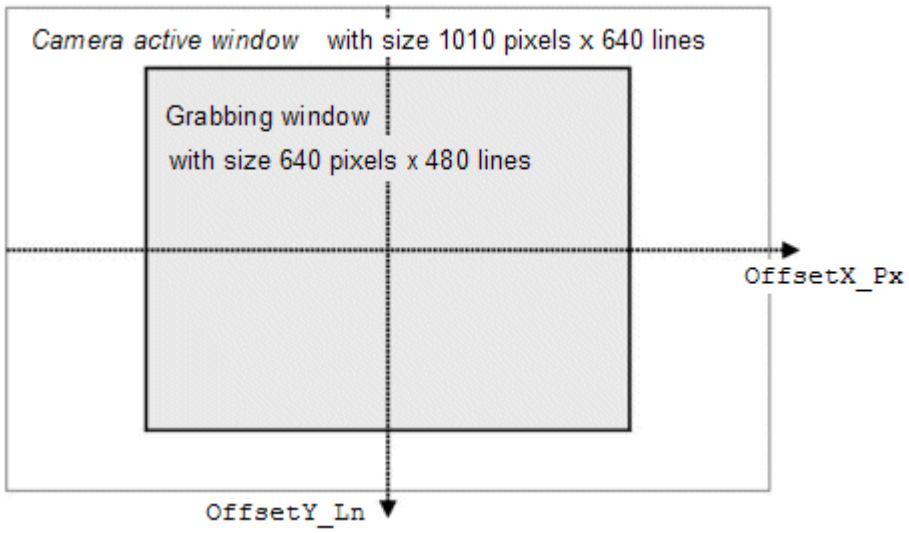
The offset parameters measure the displacement of the center of the grabbing window respective to the center of the camera active window.

The grabber's sampling frequency, which can differ from the camera's pixel clock frequency when analog or master synchronization is used, remains unchanged when `GrabWindow` is set to **MAN**. This means that the pixel aspect ratio remains unchanged when varying the size of the grabbing window.

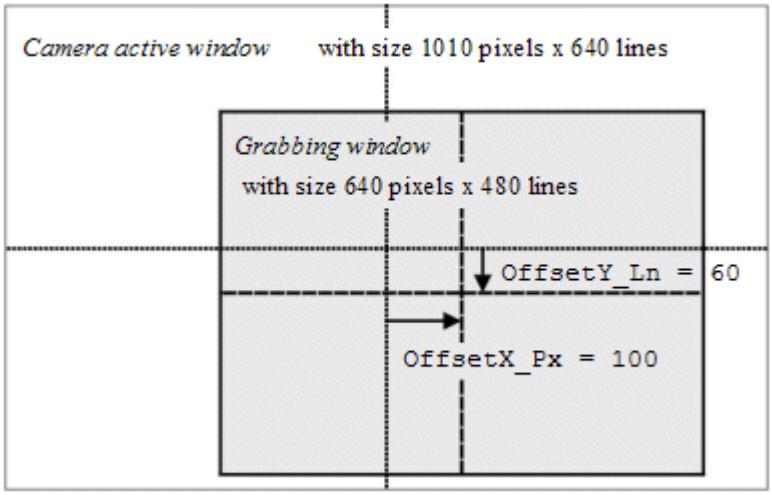
On a reduced window, the user can adjust `OffsetX_Px` or `OffsetY_Ln` provided that the grabbing window remains inside the maximal active window. For other values, an internal protection limits their effect.

As an example, on the above picture, the camera delivers an image composed of 1010 pixels and 640 lines as defined using `Hactive_Px` and `Vactive_Ln`. After setting `GrabWindow` to **MAN**, a smaller window of `WindowX_Px`=**640** pixels and `WindowY_Ln`=**480** lines is selected.

When both `OffsetX_Px` and `OffsetY_Ln` are zero, the centered image is transferred to the surface. As illustrated, the image is locked on the center of the camera active window.

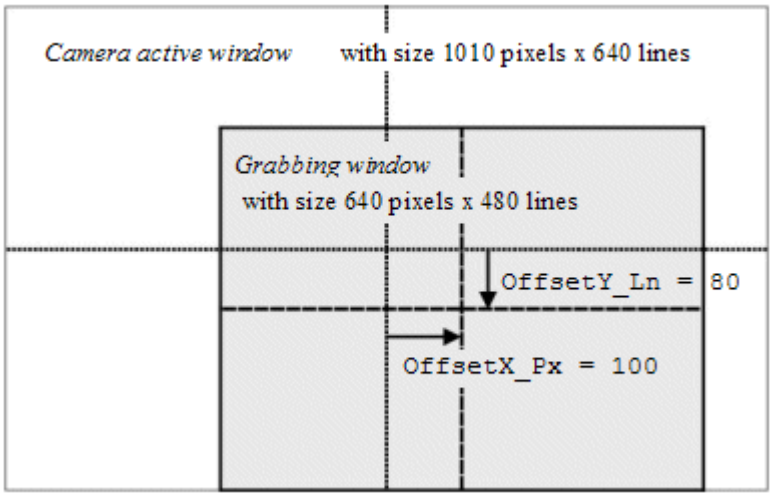


To obtain another grabbing window, if setting `OffsetX_Px` to **100** and `OffsetY_Ln` to **60**, the below image will then be transferred to the surface.



Any value of `OffsetX_Px` and `OffsetY_Ln` are accepted but an internal protection will always maintain the grabbing window inside the camera active window.

If trying `OffsetX_Px`=**100** and `OffsetY_Ln`=**200**, the internal protection limits the offset on Y. The below image will be transferred to the surface.



9.2. How to Control the Analog Gain on Domino Boards?

Gain Control Parameters

A set of channel-class adjust-level MultiCam parameters is provided to control the analog gain of the acquisition units.

Parameter	Effect
GainCtl	Gain control mode for all camera channels
Gain	Gain adjustment for all camera channels
GainTrim1 GainTrim2 GainTrim3	Gain adjustment for each camera channel individually

A MultiCam channel may require one, two or three digitizing units for operation, according to the camera structure sourcing the MultiCam channel.

Camera structure	Camera channels	Gain control parameters	
		All camera channels	Per camera channels

Single-A	1	GainCtl Gain	GainTrim1
Dual-A	2		GainTrim1 GainTrim2
Triple-A	3		GainTrim1 (Red) GainTrim2 (Green) GainTrim3 (Blue)

Nominal Gain

The nominal gain is the one that causes a full-scale response at the digitizer output when the luminance part of the applied video signal is exactly 0.7 V.

The table below fully defines the nominal gain condition:

Analog video signal	Condition	Digitized value	
		8-bit digitizer	10-bit digitizer
Black	Luminance level is equal to blanking level	0	0
White	Luminance level is 0.7 V above blanking level	255	1023

To establish the gain of all camera taps to the nominal value, set the GainCtl parameter to **0DB** and all involved GainTrim1, GainTrim2 and GainTrim3 parameters to zero.

An alternative way is to set the GainCtl parameters to **LIN**, the Gain parameter to **1000** and all involved GainTrim1, GainTrim2 and GainTrim3 parameters to **0**.

Logarithmic Gain Adjustment

This gain control scheme allows establishing one of seven gain steps logarithmically distributed.

Simply set the GainCtl parameter to one of the seven values tabulated below, and maintain all involved GainTrim1,GainTrim2 and GainTrim3 parameters to zero.

GainCtl value	Gain for all camera channels	
	Logarithmic expression	Linear expression
+3 dB	Nominal gain + 3dB	Nominal gain x 1.413
+2 dB	Nominal gain +2dB	Nominal gain x 1.259
+1 dB	Nominal gain +1d**B**	Nominal gain x 1.122
0 dB	Nominal gain	Nominal gain
-1 dB	Nominal gain -1dB	Nominal gain x 0.891

-2 dB	Nominal gain -2dB	Nominal gain x 0.794
-3 dB	Nominal gain - 3dB	Nominal gain x 0.708

When these values are used for GainCtl, the Gain parameter is not applicable.

Once the common gain is chosen for all camera taps (1, 2 or 3), it is still possible to trim an individual tap through the GainTrim1, GainTrim2 and GainTrim3 parameter.s

The control resolution for GainTrim1, GainTrim2 and GainTrim3 is not specified. Each unit increment or decrement correspond to the minimal amount of gain alteration the board hardware is supporting.

Setting the GainTrim1, GainTrim2 and GainTrim3 parameters to a positive value increases the gain for the relevant camera tap above the value established by the GainCtl common parameter. The correction induced by GainTrim1, GainTrim2 and GainTrim3 is simply added to the common gain after the logarithmic conversion of Gain.

Setting the GainTrim1, GainTrim2 and GainTrim3 parameters to a negative value decreases the gain for the relevant camera tap below the value established by the GainCtl common parameter.

The minimal and maximal values for GainTrim1, GainTrim2 and GainTrim3 depend on the value assigned to GainCtl. The limits are in the order of magnitude of ±500.

Linear Gain Adjustment

This gain control scheme allows establishing the gain according to a continuous linear scale.

Simply set the GainCtl parameter to **LIN** and maintain all involved GainTrim1,GainTrim2 and GainTrim3 parameters to zero.

The gain adjustment uses the Gain parameter and performs as follows:

Gain value	Gain for all camera channels	
	Linear expressions	Logarithmic expressions
700	Nominal gain x 0.70	Nominal gain -3.1 dB
1000	Nominal gain	Nominal gain
1400	Nominal gain x 1.40	Nominal gain +2.9 dB

Once the common gain is chosen for all camera taps (1, 2 or 3), it is still possible to trim an individual tap through the GainTrim1,GainTrim2 and GainTrim3 parameters.

The control resolution for GainTrim1,GainTrim2 and GainTrim3 is not specified. Each unit increment or decrement correspond to the minimal amount of gain alteration the board hardware is supporting.

Setting the GainTrim1,GainTrim2 and GainTrim3 parameters to a positive value increases the gain for the relevant camera tap above the value established by the GainCtl and Gain common parameters. The correction induced by GainTrim1,GainTrim2 and GainTrim3 is simply added to the common gain.

Setting the GainTrim1,GainTrim2 and GainTrim3 parameters to a negative value decreases the gain for the relevant camera tap below the value established by the GainCtl and Gain common parameters.

The minimal and maximal values for GainTrim1,GainTrim2 and GainTrim3 depend on the value assigned to GainCtl and Gain. The limits are in the order of magnitude of ±500.

9.3. How to Control the Analog Offset on Domino Boards?

Offset Control Parameters

A set of channel-class adjust-level MultiCam parameters is provided to control the analog offset of the acquisition units.

Parameter	Effect
Offset	Offset adjustment for all camera channels
OffsetTrim1 OffsetTrim2OffsetTrim3	Offset adjustment for each camera channel individually

A MultiCam channel may require one, two or three digitizing units for operation, according to the camera structure sourcing the MultiCam channel.

Camera structure	Camera channels	Offset control parameters	
		All camera channels	Per camera channel
Single- A	1	Offset	OffsetTrim1
Dual- A	2		OffsetTrim1 OffsetTrim2
Triple- A	3		OffsetTrim1 (Red) OffsetTrim2 (Green) OffsetTrim3 (Blue)

Nominal Offset

The nominal offset is the one that causes the blanking level in the analog video signal to be digitized to zero.

To establish the offset of all camera taps to the nominal value, set the `Offset` parameter to zero and all `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` parameters to zero.

Offset Adjustment

The offset is applied after the gain. This means that as a first step the gain control establishes a digitizing span, then as a second step the offset control moves this span over the clamped video signal.

The offset is adjusted through the `Offset` parameter, the value of which should be in a range depending of the hardware.

Increasing the offset value results into a brighter image, and reversely, decreasing the offset value results into a darker image.

An offset adjustment of 500 results in a digitizing span shift, which amounts to 50 % of the full digitizing span.

Once the common offset is chosen, it is still possible to trim an individual tap through the `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` parameters.

The control resolution for `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` is not specified. Each unit increment or decrement corresponds to the minimal amount of offset alteration the board hardware is supporting.

Setting the `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` parameters to a positive value increases the offset for the relevant camera tap above the value established by the `Offset` common parameter. The correction induced by `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` is simply added to the common `Offset`.

Setting the `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` parameters to a negative value decreases the offset for the relevant camera tap below the value established by the `Offset` common parameter.

The minimal and maximal values for `OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3` depend on the value assigned to `Offset`.

10. Using Look-Up Tables

Refer to the "Using Look-Up Tables" above section in *D402EN-MultiCam User Guide* PDF document.

10.1. Definitions

LUT device: a device transforming a single input vector into a single output vector using a transformation law defined by a look-up table.

LUT dimension: input vector width x output vector width.

LUT (device) set: a group of LUT devices operating concurrently in a MultiCam channel.

LUT image: the data contents of one LUT device.

LUT set image: the data contents of a LUT device set.

LUT buffer: an area of the frame grabber memory used as cache buffer for storage of one LUT image.

LUT set buffer: an area of the frame grabber memory used as cache buffer for storage of one LUT set image.

LUT array: the area of frame grabber memory used for LUT buffers

LUT_RAM: a type of LUT device based on a RAM block of 2 "input vector width" locations of "output vector width" bits.

HD_LUT: a type of LUT device that emulates a high-dynamic LUT using a EURESYS proprietary algorithm.

10.2. LUT Characteristics per Board

The following table summarizes the characteristics of the LUT operators on every board:

Board	Number of LUT devices	LUT dimension	Number of LUT devices per set	Total number of LUT buffers	Number of LUT set buffers per channel
Domino Iota	1	8 x 8	1	32	**LUT_RAM**

Domino Alpha 2	2	8 x 8	1 or 2	32	**LUT_RAM**
Domino Melody	1	8 x 8 or 10 x 10	1	5	**LUT_RAM**
Domino Symphony	4	8 x 8 or 10 x 10	1	5	**LUT_RAM**
Grablink Value Grablink Value cPCI	3	8 x 8	1, 2 or 3	0	**LUT_RAM**
Grablink Quickpack ColorScan	3	8 x 8	3	5	**LUT_RAM**
Grablink Quickpack CFA	4	16 x 16	4	5	**HD_LUT**
Grablink Full	3	12 x 16	3	4	**LUT_RAM**

On **Domino Iota** and **Domino Alpha 2**, each digitizing unit includes an 8 x 8 **LUT_RAM** LUT device; a LUT array capable to store 32 LUT images serves as a data pool for the LUT sets.

Note. The number of devices per set is depending on the `TapStructure` of the camera:

TapStructure	Number of LUT devices per set
SINGLE_A	1
DUAL_A	2

On **Domino Melody**, the dimension of the unique **LUT_RAM** LUT device is depending on the selected value of `ColorFormat` parameter:

- 8 x 8 when `ColorFormat` = **Y8**
- 10 x 10 when `ColorFormat` = **Y10** or **Y16**

Each channel benefits from a LUT array capable to store 4 user-defined LUT set images and 1 pre-defined LUT set image.

On **Domino Symphony**, each of the 4 channels has a LUT operator offering the same characteristics as the LUT operator of Domino Melody.

On **Grablink Value** and **Grablink Value cPCI**, each Camera Link tap includes an 8 x 8 LUT device; there is no LUT cache buffer.

Note. The number of devices per set is depending on the `TapStructure` of the camera:

TapStructure	Number of LUT devices per set
SINGLE_B	1
DUAL_B	2
TRIPLE_B	3

Note. The LUT operator is not available for other values of `TapStructure`.

On **Grablink Quickpack ColorScan**, the LUT device set is composed with 3 8x8 **LUT_RAM** LUT devices, one per color component; the LUT array is capable to store 4 user-defined LUT set images and 1 pre-defined LUT set image.

On **Grablink Quickpack CFA**, the LUT device set is composed with 4 16x16 **HD_LUT** LUT devices, one per color component plus one for the luminance component; the LUT array is capable to store 4 user-defined LUT set images and 1 pre-defined LUT set image.

Note. The number of devices per set is independent of the selected `ColorFormat`; it is always 4.

10.3. LUT APIs

MultiCam provides two Application Programming Interfaces:

- The classic LUT API
- The advanced LUT API

The following table shows the availability of the LUT APIs for the boards featuring a LUT operator:

Board	Classic LUT API	Advanced LUT API
Domino Iota Domino Alpha 2	Applicable	N/A
Domino Melody Domino Symphony	Applicable	Applicable
Grablink Value Grablink Value cPCI	Applicable	N/A
Grablink Quickpack ColorScan	N/A	Applicable
Grablink Quickpack CFA	N/A	Applicable
Grablink Full	N/A	Applicable

The advanced LUT API is systematically available on latest generation of boards featuring a LUT operator since **Grablink Quickpack ColorScan**.

The classic LUT API is available on all Domino boards featuring a LUT operator (all Domino's but **Domino Symphony**).

The following table compares the features of both APIs:

Feature	Classic LUT API	Advanced LUT API
Parametric LUT definition	NO	YES
High dynamic LUT support	NO	YES
User-defined LUT data	YES	YES

The advanced LUT API provides 3 different parametric methods to define the transformation law of the LUT operator. It enables also the high dynamic LUT operators. Both APIs allow for the user to define its own look-up table data.

10.4. How to Operate LUTs?

Following operations must be done successively to set up the LUT operator:

1. LUT definition: select a definition method and defines the LUT contents
2. LUT loading: load the LUT into frame grabber memory
3. LUT activation: activate a selected LUT

LUT Definition

This step defines the contents of the LUT.

MultiCam provides two methodologies:

1. **Table method**: the transformation law is defined in a tabular form. The table method is available for both classic and advanced APIs; see Table LUT Definition Method for a detailed description.
2. **Parametric methods**: where the transformation law is defined by a few parameters. The parametric methods are available exclusively in the advanced API; see Parametric LUT Definition Methods for a detailed description.

   In the advanced LUT API only: the LUT definition method is defined by setting the LUT_Method parameter to the desired value.

LUT Loading

Advanced API

When the LUT is defined using one of the parametric method, its contents is calculated and loaded into the LUT array using a single-step procedure. This involves the `LUT_StoreIndex` parameter:

1. Choose an index to designate the **LUT set buffer**. This is done by setting the integer parameter `LUT_StoreIndex` to a value between 1 and 4.

   Note: ***Note.*** *The advanced LUT API doesn't allow reading back the table contents.*

When the LUT is defined using the table method, transferring the LUT surface contents into the LUT array is a two-step procedure. This involves the `LUT_Table` and `LUT_StoreIndex` parameters:

1. Set the instance parameter `LUT_Table` to the handle of the LUT surface.
2. Choose an index to designate the **LUT set buffer**. This is done by setting the integer parameter `LUT_StoreIndex` to a value between 1 and 4.

Classic API

Transferring the LUT surface contents to a LUT image inside the LUT array belonging to the frame grabber is a two-steps process. This involves the `LutIndex` and `InputLut` parameters:

1. Set the instance parameter `InputLut` to the handle of the LUT surface.
2. Choose an index to designate the **LUT buffer**. This is done by setting the integer parameter `LutIndex` to a value between 1 and 32.

`LutIndex` is a channel-class parameter; each channel owns its own independent set of designating indexes.

> **Note:** ***Note.*** *In the classic API, loading the LUT into the LUT array triggers automatically the activation of this LUT.*

LUT Activation

Advanced API

The user designates the storage buffer to activate by setting the `LUT_UseIndex` parameter with its index.

Classic API

Setting the `LutIndex` parameter to a defined value has the effect of applying the contents of the designated LUT image to the channel LUT set.

The effective application occurs at the next SAS event (start of acquisition sequence). This involves an internal data transfer of great speed (duration less than 100 microseconds on Domino boards).

10.5. Parametric LUT Definition Methods

The advanced LUT API offers three parametric methods to specify the transfer function of the LUT operator:

- Response control method
- Emphasis method
- Threshold method

LUT_Method

The user selects the LUT construction method by means of the `LUT_Method` parameter.

Note: ***Note.*** *The same LUT definition method applies for all LUT devices belonging to a channel.*

Controlling parameters associated with each parametric definition method

LUT_Method	Applicable parametric LUT building control parameters

RESPONSE_CONTROL	`LUT_Contrast`
	`LUT_Brightness`
	`LUT_Visibility`
	`LUT_Negative`

EMPHASIS	`LUT_Emphasis` `LUT_Negative`

THRESHOLD	`LUT_SlicingLevel`
	`LUT_SlicingBand`
	`LUT_LightResponse`
	`LUT_BandResponse`
	`LUT_DarkResponse`

All controlling parameters are of the "float collection" type. The dimension of the collection is equal to the number of LUT devices per LUT set. Refer to 1.1.2 for the number of LUT devices per LUT set.

On boards with 1 LUT device per set, such as **Domino Melody**, the dimension is 1:

Collection member #	0
Associated color component	Luminance

On boards with 3 LUT devices per set, such as **Grablink Quickpack ColorScan**, the dimension is 3:

Collection member #	0	1	2
Associated color component	Red	Green	Blue

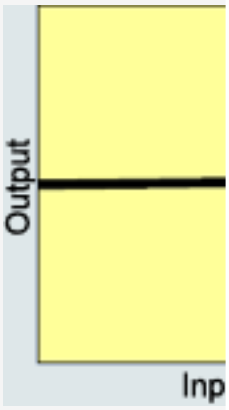
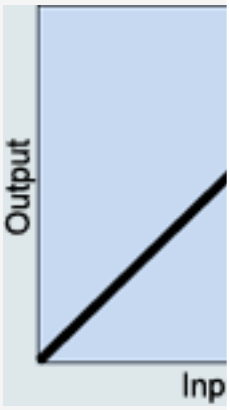
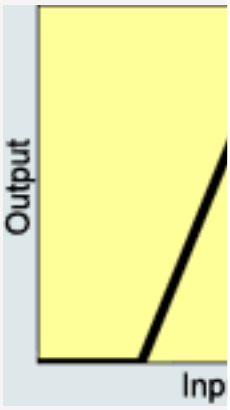
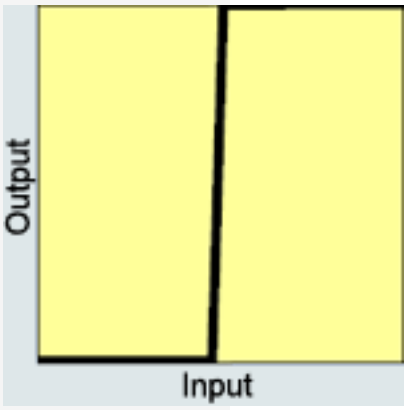
On boards with 4 LUT devices per set, such as **Grablink Quickpack CFA**, the dimension is 4:

Collection member #	0	1	2	3
Associated color component	Red	Green	Blue	Luminance

LUT_Contrast

This control is applicable exclusively with the Response Control parametric LUT definition method. This implements a control similar to the contrast control of a television monitor.

The following charts explain the contrast effect when all other controls are set to the default value: LUT_Brightness = **0.00**, LUT_Visibility = **0.00**, LUT_Negative = **FALSE**.

LUT_Contrast = 0.00 Minimum value	LUT_Contrast = 0.80	LUT_Contrast = 1.00 Default value	LUT_Contrast = 1.20	LUT_Contrast = 2.00 Maximum value
				

The slope of the transformation law is the gain, which is non-linearly controlled from the Contrast parameter.

Mathematically, the relationship is:

$$\text{Gain} = 10^{2\times(\text{Contrast}-1)}$$

- The smallest gain achieved for Contrast = 0.00 is Gain = 0.01.
- The unity gain is achieved for Contrast = 1.00.
- The largest gain achieved for Contrast = 2.00 is Gain = 100.

To achieve a required given gain, the contrast control should be set to:

$$\text{Contrast} = 1 + \frac{\log_{10}\text{Gain}}{2}$$

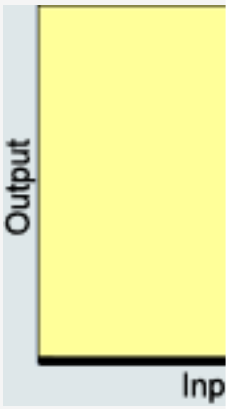
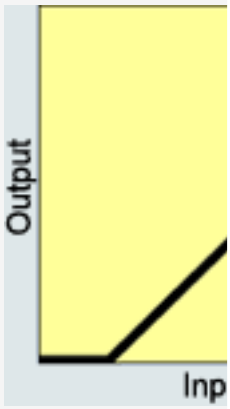
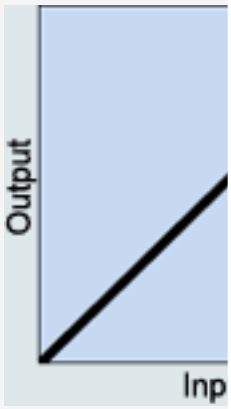
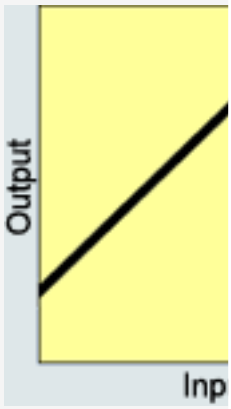
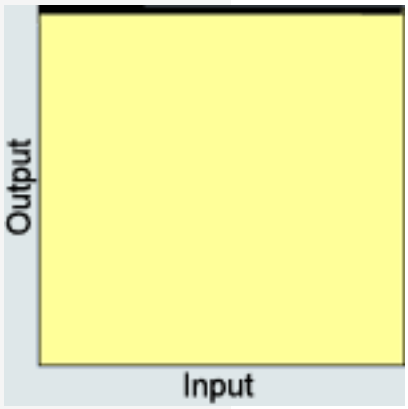
If the required gain is expressed in decibels (dB):

$$\text{Contrast} = 1 + \frac{\text{Gain (dB)}}{40}$$

LUT_Brightness

This control is applicable exclusively with the **Response Control** parametric LUT definition method. This implements a control similar to the brightness control of a television monitor.

The following charts explain the brightness effect when all other controls are set to the default value: LUT_Contrast = 1.00, LUT_Visibility = 0.00, LUT_Negative = **FALSE**.

LUT_Brightness = -1.00 (minimum value)	LUT_Brightness = -0.20	LUT_Brightness = 0.00 (default value)	LUT_Brightness = +0.20	LUT_Brightness = +1.00 (maximum value)
				

When the brightness is set to zero, the mid-level input level of 0.5 is transformed as the same output level of 0.5. This is true for any value of the other control parameters.

Any increase in the brightness towards +1.00 results into a lighter output. Any decrease of the brightness towards -1.00 results into a darker output.

The +1.00 limit corresponds to the situation where the whole input range data gets transformed into the full white. The -1.00 limit corresponds to the situation where the whole input range data gets transformed into the full black.

This rule applies for any chosen contrast value.

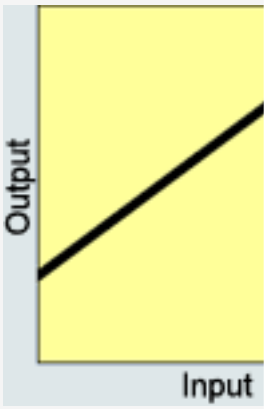
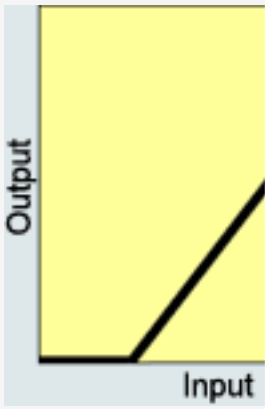
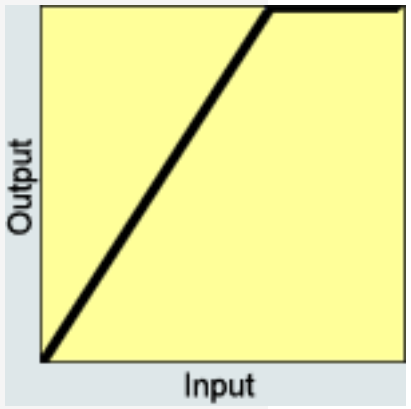
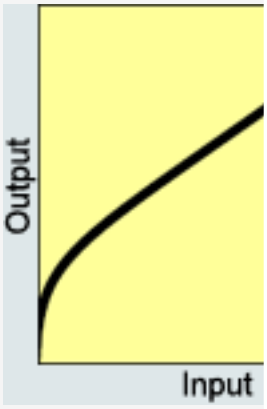
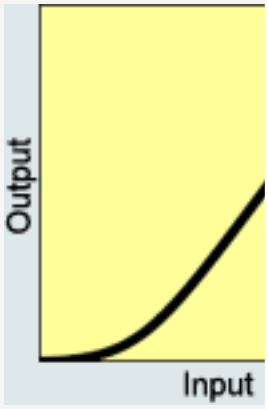
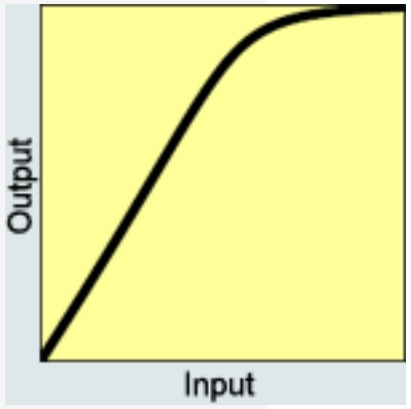
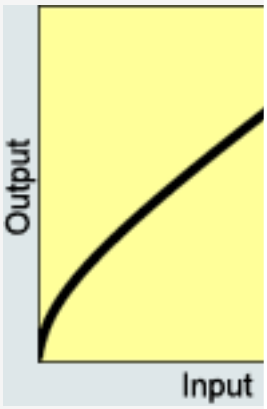
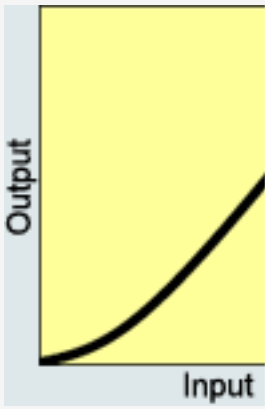
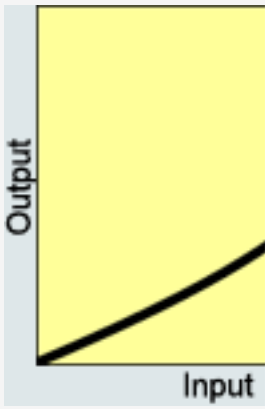
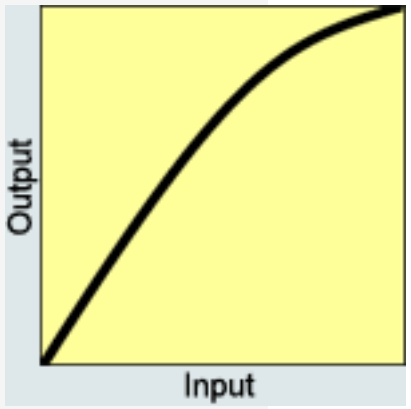
LUT_Visibility

This control is applicable exclusively with the **Response Control** parametric LUT definition method.

The operation of contrast and brightness controls as explained above occasionally removes some part of the input dynamics. Significantly dark regions of the image can be transformed into full black, and become invisible. This holds true for significantly illuminated regions, clipping to full white.

The visibility control has been incorporated to smoothly reveal these hidden parts in the image. It operates in a range 0.00 to +1.00.

Visibility effect for typical values of LUT_Contrast and LUT_Brightness controls, assuming that LUT_Negative = **FALSE**

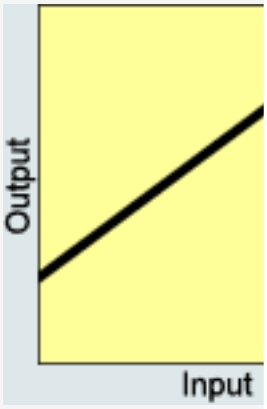
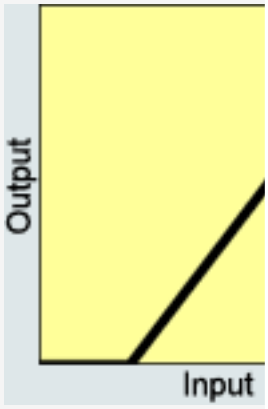
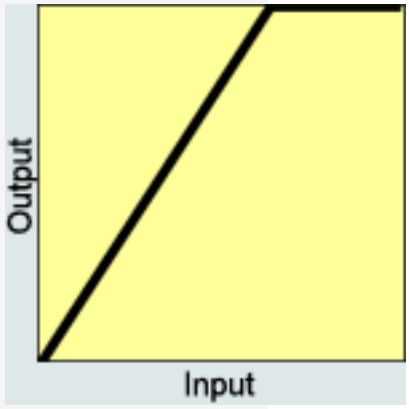
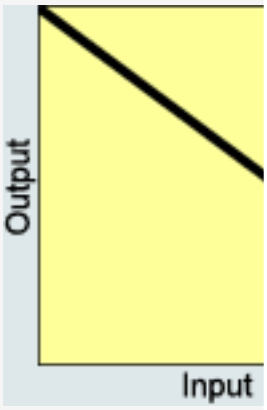
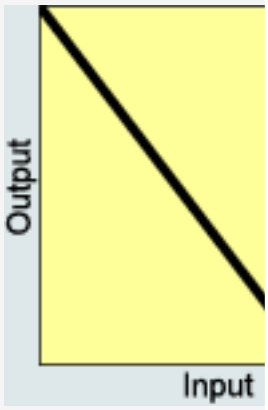
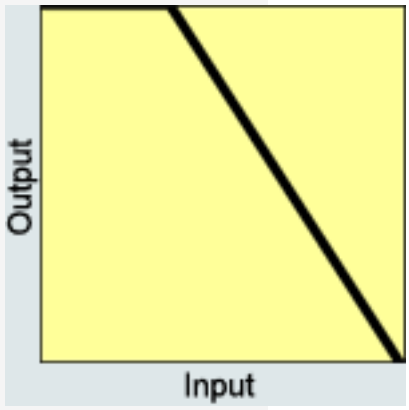
	LUT_Contrast = 0.94 LUT_ Brightness = +0.14	LUT_Contrast = 1.06 LUT_ Brightness = -0.14	LUT_Contrast = 0.82 LUT_ Brightness = -0.38	LUT_Contrast = 1.10 LUT_ Brightness = +0.22
LUT_ Visibility = 0.00 (default value)				
LUT_ Visibility = 0.25				
LUT_ Visibility = 1.00				

The default value is 0 which generates the piecewise linear transformation curves. Choosing values closer to +1 generates smoother curves.

LUT_Negative

This control is applicable with both the **Response Control** and the **Emphasis** parametric LUT definition methods. This control allows transforming an image into its negative counterpart, where black is substituted to white and conversely.

The following charts explain the negative effect for typical values of other controls. When the parameter `Negative` is set to **TRUE**, the transformation table is mirrored around a vertical axis in the graphs. This swaps the black and white values, and gives rise to a photographic negative effect. The default value is **FALSE**.

	LUT_Contrast = 0.94 LUT_ Brightness = +0.14	LUT_Contrast = 1.06 LUT_ Brightness = -0.14	LUT_Contrast = 0.82 LUT_ Brightness = -0.38	LUT_Contrast = 1.10 LUT_ Brightness = +0.22
LUT_Negative = **FALSE** (default value)				
LUT_Negative = **TRUE**				

LUT_Emphasis

This control is applicable exclusively with the **Emphasis** parametric LUT definition method.

This control allows transforming an image using a power-law expression:

$$\text{Output} = \text{Input}^{\gamma}$$

The g (gamma) exponent is mathematically linked to LUT_Emphasis by:

$$\gamma = 10^{-\text{Emphasis}}$$

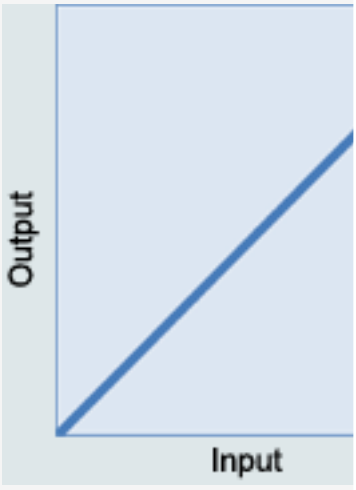
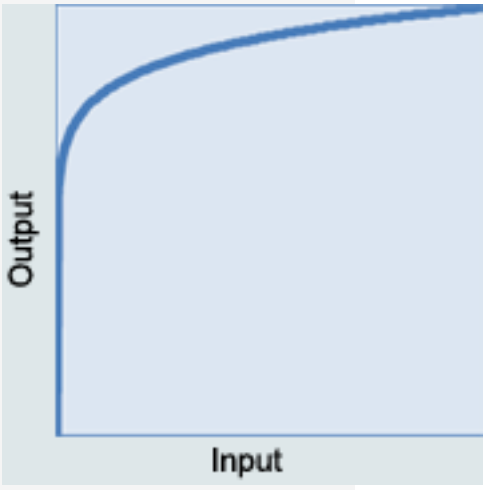
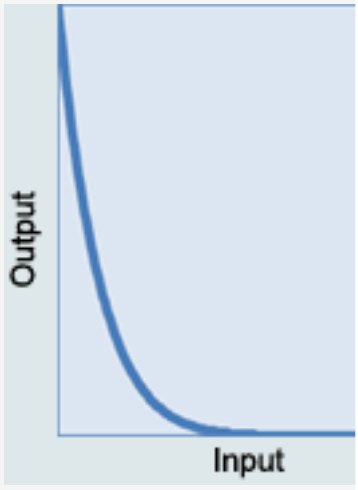
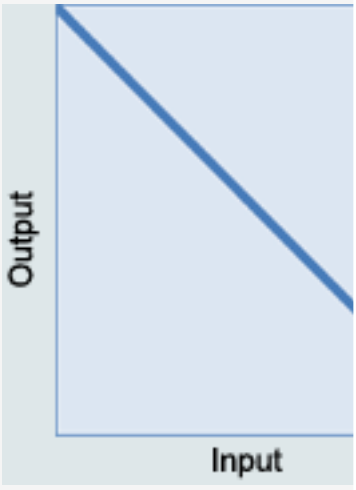
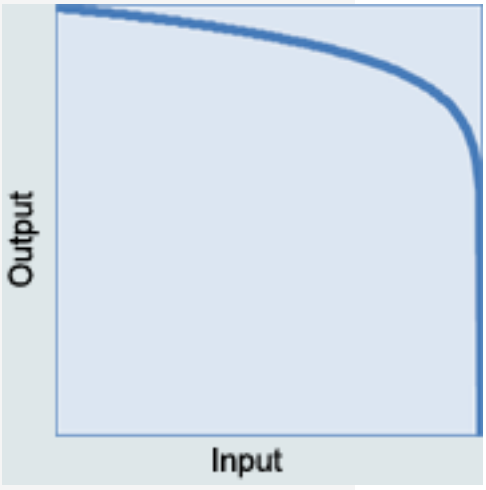
- The smallest g achieved for LUT_Emphasis = 1.00 is: g = 0.1.
- The linear law is obtained with for LUT_Emphasis = 0.00.
- The largest g achieved for LUT_Emphasis = -1.00 is: g = 10.

To achieve a required given g, the LUT_Emphasis control should be set to:

$$\text{Emphasis} = -\log_{10}\gamma$$

The default value of LUT_Emphasis is 0.

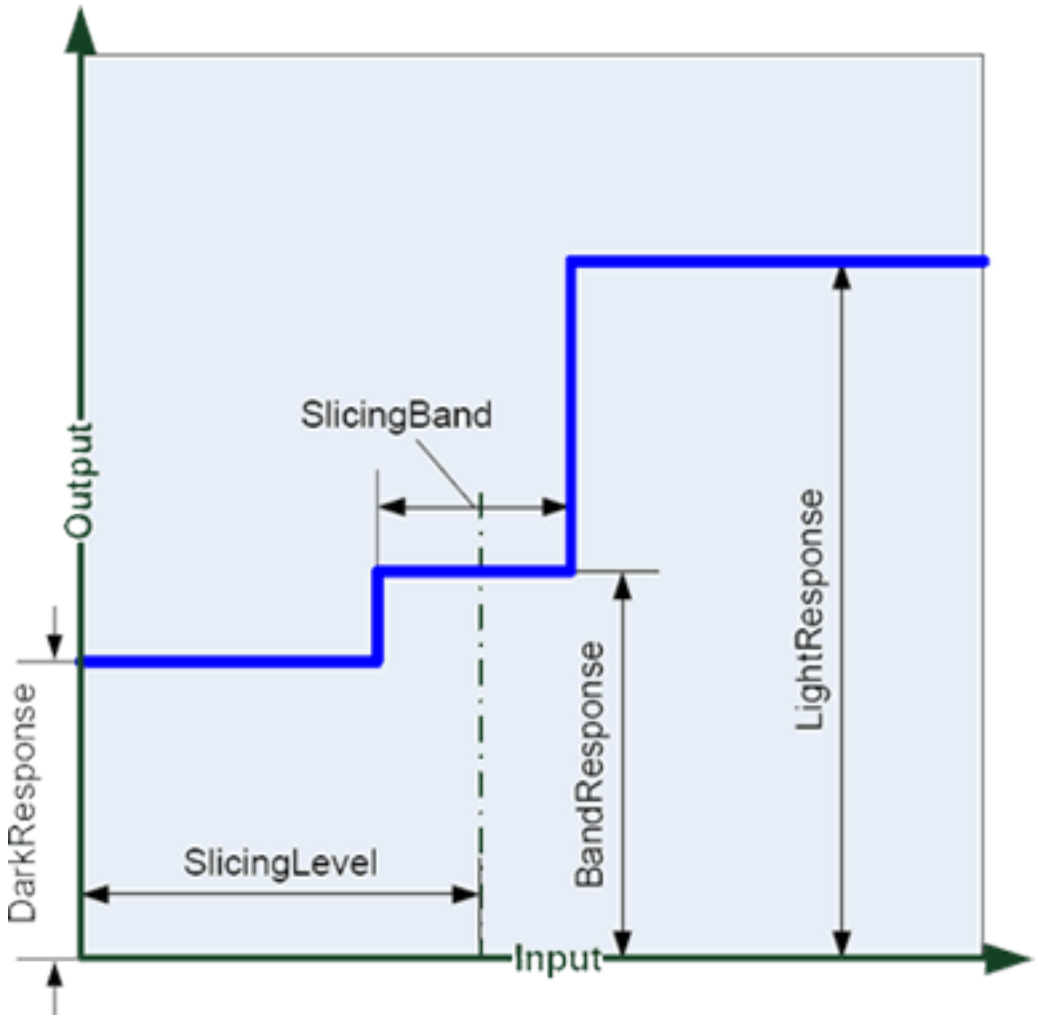
Emphasis effect for typical values of LUT_Emphasis and LUT_Negative

	LUT_Emphasis = -1.00 [g = 10]	LUT_Emphasis = 0.00 (default value) [g = 1]	LUT_Emphasis = 1.00 [g = 0.1]
LUT_Negative = **FALSE** (default value)			
LUT_Negative = **TRUE**			

Threshold

As shown on next figure, a set of five parameters —LUT_SlicingLevel, LUT_SlicingBand, LUT_LightResponse, LUT_BandResponse, and LUT_DarkResponse— defines a double threshold transformation law. All parameters are "float collection" parameters, ranging from 0.00 up to 1.00.

These controls are applicable exclusively with the threshold parametric LUT definition method.



Threshold method

LUT_SlicingLevel specifies the mean value of both thresholds in the input range. The default value of 0.5 specifies the middle of the input range.

The following table lists the value ranges and default values:

Parameter	Value range	Default value
LUT_SlicingLevel	0.00 up 1.00	0.50

LUT_SlicingBand	0.00 up 1.00	0.50
LUT_LightResponse	0.00 up 1.00	0.75
LUT_BandResponse	0.00 up 1.00	0.50
LUT_DarkResponse	0.00 up 1.00	0.25

10.6. Table LUT Definition Method

In the Advanced LUT API only: set value TABLE to parameter LUT_Method.

When Table method is selected, the user must fill a temporary buffer located in the host memory. The container for this temporary buffer is a MultiCam surface belonging to the surface class normally involved as a destination of camera acquisition. However, the surface used in the process of look-up table programming contains no image. This item is referred as a **LUT surface**.

Three surface-class parameters are significant: SurfaceSize, SurfaceAddr, and PlaneCount. SurfaceSize and SurfaceAddr are of the "float collection" type.

Number of Planes

The dimension of the collection is equal to the number of LUT devices per LUT set and to the number of planes in the LUT surface. The Surface-class parameter PlaneCount must be set to this value.

On boards with 1 LUT device per set, such as **Domino Melody**, the dimension is 1:

Plane #	0
Associated color component	Luminance

On boards with 2 LUT devices per set, such as **Domino Alpha 2** or **Grablink Value** with dual-tap cameras, the dimension is 2 and the components are assigned as follows:

Plane #	0	1
Associated color component	Tap0	Tap1

On boards with 3 LUT devices per set, such as **Grablink Quickpack ColorScan**, the dimension is 3 and the components are assigned as follows:

Plane #	0	1	2
Associated color component	Red	Green	Blue

On boards with 4 LUT devices per set, such as **Grablink Quickpack CFA**, the dimension is 4 and the components are assigned as follows:

Plane #	0	1	2	3
Associated color component	Red	Green	Blue	Luminance

Surface Size

The LUT structure is defined by the `LUT_InDataWidth` and `LUT_OutDataWidth` parameters.

The following table shows the values of both parameters:

Board	LUT_ InDataWidth	LUT_ OutDataWidth	Conditions
Domino Melody	8	8	`ColorFormat` = **Y8**
	10	10	`ColorFormat` = **Y10** `ColorFormat` = **Y16**
Grablink Quickpack ColorScan	8	8	N/A
Grablink Quickpack CFA	16	16	N/A
Grablink Full	12	16	N/A

Note: ***Note.Grablink Quickpack CFA*** *implements high-dynamic LUTs.*

- With the **advanced LUT API**, each entry in a LUT surface plane is 32-bit wide, independently of the `LUT_OutDataWidth`.
- With the **classic API**, an entry in a LUT surface plane is 8-bit when `LUT_OutDataWidth` = **8** and 16-bit when `LUT_OutDataWidth` = **10**.

The following table lists the `SurfaceSize` requirement for each plane of the LUT surface as a function of `LUT_InDataWidth`, `LUT_OutDataWidth` for both the classic and advanced LUT APIs.

LUT_ InDataWidth [Bit]	LUT_ OutDataWidth [Bit]	SurfaceSize [Bytes] advanced API	SurfaceSize [Bytes] classic API
8	8	1,024	256
10	10	4,096	2,048
12	16	16,384	N/A
16	16	262,144	N/A

Data Alignment

For both API, the significant bits must be right-aligned in the container.

LUT Surface Creation

The user creates a LUT surface corresponding to the above described characteristics. It receives a handle that designates this surface instance. Once instantiated, the application software fills the LUT surface with the required data.

11. Using Line-Scan Acquisitions

This section describes the capabilities of MultiCam to acquire images from line-scan cameras. MultiCam provides an elaborate set of triggering capabilities and system interfaces.

The latest enhancements of MultiCam includes:

- LONGPAGE mode with fixed length up to 2,147,483,648 (= 2^{31}) lines.
- LONGPAGE mode with variable length up to 2^{31} lines.
- Enhanced rate converter with overflow detection and activity detector.

Three fundamental operation modes are available for line-scan cameras: WEB, PAGE and LONGPAGE.

The WEB operation mode is intended for image acquisition of single continuous objects.

The PAGE and LONGPAGE operation modes are both intended for image acquisition of multiple discrete objects.

The LONGPAGE operation mode supports objects up to 2,147,483,648 lines while the classic PAGE operation mode is limited to objects up to 65,535 lines.

The LONGPAGE operation mode has the capability to acquire variable size objects as defined by a "Page Cover" signal.

The PAGE operation mode has the capability to terminate the Page acquisition sequence after a pre-defined number of objects. This feature is not available in LONGPAGE operation mode.

11.1. Web Operation

The camera observes a single object moving continuously in front of the line-scan camera. The length of the object is undefined or infinite. The end of the acquisition is under user application control (see below).

This operation mode is available for line-scan cameras only, when `Imaging`=**LINE** or `Imaging`=**TDI**. It is selected by setting Operation to **WEB**.

Controlling the Web operation

In Web operation mode, a unique acquisition sequence can be executed within the channel activity period. The Web acquisition sequence is intended for the capture of a single object.

Preparing and Activating the Channel

The first action is to define all MultiCam channel parameters before channel activation. More specifically the trigger parameters defining the sequence start and stop conditions are required to control the acquisition sequence.

Setting `ChannelState` parameter to **ACTIVE** activates the channel and arms the trigger circuit.

The Web Acquisition Sequence will start after the first trigger event occurring after channel activation.

Starting a Web Acquisition Sequence

The origin of the trigger event is determined by the trigger condition as specified by `TrigMode` parameter.

Usually, the trigger event is immediate (`TrigMode`=**IMMEDIATE**) in order to start the acquisition sequence immediately.

Alternatively soft trigger (`TrigMode`=**SOFT**) or hard trigger signal (`TrigMode`=**HARD**) or both can be selected (`TrigMode`=**COMBINED**).

When hard trigger is specified, additional parameters `TrigCtl`, `TrigEdge`, `TrigFilter` and `TrigLine` define the trigger input.

To summarize the usage of trigger parameters, see Trigger Control Category.

Web Acquisition Sequence

Once the Web acquisition sequence is started, the frame grabber acquires data lines continuously until the acquisition sequence is stopped.

The acquisition sequence is composed of one or more acquisition phases. During an acquisition phase the frame grabber stores data lines into one destination surface.

Destination surfaces contain an arbitrary number of lines as defined by parameter `PageLength_Ln`. For more information, see "Selecting optimal `PageLength_Ln`".

When a surface is filled, the acquisition continues automatically into the next available surface without any missing lines. This process repeats until a stop condition occurs.

Every time a surface is filled, the "MC_SIG_SURFACE_FILLED" signal is reported to the user. Parameter `LineIndex` reflects the number of lines already written in a partially filled surface. If SeqLength_Ln is not a multiple of `PageLength_Ln`, the last surface will be partially filled. A new surface is selected at each begin of acquisition phase.

Lines are acquired at a rate defined by the line trigger condition. When object speed is variable, it is convenient to generate a line trigger derived from a motion encoder. For more information, see "Line trigger condition".

Stopping a Web Acquisition Sequence

The Web operation allows two methods to stop the sequence:

- **Manual stop** (recommended method)
- **Automatic stop**

If scanned object has a finite length, the automatic stop method can be used to automatically stop the sequence after a pre- defined number of lines. This variant of Web operation is named WEB Finite. In this case, the number of lines to be acquired needs to be specified before Channel activation.

More often the length of the object is unknown. Therefore, the manual stop method is required. In this case, the parameter SeqLength_Ln should be set to **-1**, in order to define an infinite Web acquisition sequence.

Manual Web acquisition sequence stop

The Web acquisition sequence stops by setting `ChannelState` to **IDLE**.

Two flavors of stop are selectable with parameter `BreakEffect`:

- When `BreakEffect`=**FINISH**, the Web acquisition sequence stops after the completion of the surface filling.
- When `BreakEffect`=**ABORT**, the Web acquisition sequence stops immediately, interrupting also the filling of the current surface. The last surface might be partially filled.

Automatic Web acquisition sequence stop

An automatic stop is available for the Web acquisition sequence. Two methods are available.

Automatic Stop after specified number of lines

Use parameter SeqLength_Ln to specify the total number of lines to be acquired. The sequence will automatically stop after the last acquired line. The last surface might be partially filled.

Range of values for LineCount is:

- SeqLength_Ln > 0
- SeqLength_Ln < (`PageLength_Ln` * 65,535)

Automatic Stop after specified number of surfaces filled

Use parameter `SeqLength_Fr` to specify the number of surfaces to be filled. The sequence will automatically stop after the filling of the last surface.

Range of values for `SeqLength_Fr` is:

- `SeqLength_Fr` > 0
- `SeqLength_Fr` < 65,535

11.2. Page Operation

The camera observes a train of multiple objects moving in front of the line-scan camera. Page operation is capable to handle objects requiring up to 65,534 lines.

This operation mode is available for line-scan cameras only, when `Imaging`=**LINE** or `Imaging`=**TDI**.

Page operation is selected by setting Operation=PAGE.

Controlling the Page operation

In Page operation mode, a unique Page acquisition sequence takes place within the Channel activity period.

A Page acquisition sequence is composed of a repetitive sequence of Page acquisition phases. Each Page acquisition phase is responsible for the image capture of one object.

Preparing and Activating the Channel

The first action is to define all MultiCam Channel parameters before channel activation. More specifically the trigger parameters defining the sequence start and stop conditions are required to control the acquisition sequence.

Setting `ChannelState` parameter to **ACTIVE** activates the channel and arms the trigger circuit.

The unique sequence will effectively start after the first trigger event occurring after channel activation when the first acquisition phase is triggered.

Starting the First Page Acquisition Phase

The trigger source is determined by the Start Trigger condition. Usually, the start trigger event is a hard trigger; however a soft trigger signal can also be selected. Usually, the trigger is derived from a position detector. The trigger condition is defined by parameter `TrigMode`.

The start trigger event will produce after a programmable delay a trigger event. This delay can be used to compensate an advanced trigger delivered by a position detector placed away from the camera field of view. The trigger delay can be specified with parameter `PageDelay_Ln`.

At first trigger event, the object scanning starts effectively; a "Start of Sequence" signal is reported to the user.

To summarize the usage of trigger parameters, see"Trigger Control Category" on page 224.

Starting the Subsequent Page Acquisition Phase

The trigger source for all subsequent acquisition phases is determined by the Next Start Trigger condition. Usually, the Next start trigger event is the same as the Start Trigger condition. The next trigger condition is defined by parameter `NextTrigMode`.

The start trigger event will produce after a programmable delay a trigger event. This delay can be used to compensate an advanced trigger delivered by a position detector placed away from the camera field of view. The trigger delay can be specified with parameter `PageDelay_Ln`.

At trigger event, the object scanning starts effectively; however no "Start of Sequence" signal is reported to the user.

Page Acquisition Phase

Once the page acquisition phase is started, the frame grabber acquires data lines and stores them into one surface.

Destination surface receives an arbitrary number of lines as defined by parameter `PageLength_Ln`. See "Selecting Optimal `PageLength_Ln`" for more info.

Parameter `LineIndex` reflects the number of lines already written in a partially filled surface. If SeqLength_Ln is not a multiple of `PageLength_Ln`, the surface will be partially filled. A new surface is selected at each begin of Page acquisition phase.

Lines are acquired at a rate defined by the line trigger condition. When object speed is variable, it is convenient to generate a line trigger derived from a motion encoder. See "line trigger condition" for more info.

Stopping a Page Acquisition Phase

Within a Page acquisition sequence, the acquisition phase stops automatically when the surface is filled after the acquisition of `PageLength_Ln` lines.

The Page acquisition sequence supports objects up to highest possible value of `PageLength_Ln`, namely 65,535 lines.

When acquisition phase is stopped, a "Surface filled" signal is reported immediately to the user. The "Surface filled" signal will be reported as soon as all the acquired data lines are transferred to the host memory.

Stopping the Acquisition Sequence

Manual sequence stop

The Page acquisition sequence stops by setting `ChannelState`=**IDLE**.

Two flavors are selectable with parameter `BreakEffect`:

When BreakEffect=**FINISH**, the Page acquisition sequence stops after normal completion of the current Page acquisition phase if the Object scanning is already started. Otherwise the Page acquisition sequence stops immediately.

When BreakEffect=**ABORT**, the Page acquisition sequence stops immediately, interrupting also a Page acquisition phase in progress. The last object acquisition is not usable.

Automatic sequence stop

An automatic sequence stop is available if following conditions are satisfied:

- Number of objects is known before starting the sequence.
- Number of objects is smaller than 65,534.

Therefore, use parameter SeqLength_Fr to specify the number of objects to be acquired. The sequence will automatically stops after SeqLength_Fr phases.

11.3. LongPage Operation

The camera observes a train of multiple objects moving in front of the line-scan camera. LongPage operation is capable to handle long objects requiring up to 2,147,483,648 lines.

This operation mode is available for line-scan cameras only, when Imaging=**LINE** or Imaging=**TDI**.

Setting Operation to LONGPAGE enables LongPage operation.

Controlling the LongPage operation

In Web operation mode, a unique acquisition sequence can be executed within the channel activity period. The Web acquisition sequence is intended for the capture of a single object.

Preparing and Activating the Channel

The first action is to define all MultiCam channel parameters before channel activation. More specifically, the trigger parameters defining the sequence start and stop conditions are required to control the acquisition sequence.

Setting ChannelState parameter to **ACTIVE** activates the channel and arms the trigger circuit.

The first sequence will start after the first trigger event occurring after channel activation.

Starting an Acquisition Sequence

The trigger source is determined by the Start Trigger condition. Usually, the start trigger event is a hard trigger; however a soft trigger signal can also be selected. Usually, the trigger is derived from a position detector. The trigger condition is defined by parameter TrigMode.

The start trigger event will produce after a programmable delay a trigger event. This delay can be used to compensate an advanced trigger delivered by a position detector placed away from the camera field of view. The trigger delay can be specified with parameter `PageDelay_Ln`.

At trigger event, the acquisition sequence starts effectively.

To summarize the usage of trigger parameters, see "Trigger Control Category" on page 224.

Long Page Acquisition Sequence

Once the Long Page acquisition sequence is started, the frame grabber acquires data lines continuously until the acquisition sequence is stopped.

The acquisition sequence is composed of one or more acquisition phases. During an acquisition phase the frame grabber stores data lines into one destination surface.

Destination surfaces contain an arbitrary number of lines as defined by parameter `PageLength_Ln`. For more information, see "Selecting optimal `PageLength_Ln`".

When a surface is filled, the acquisition continues automatically into the next available surface without any missing lines. This process repeats until a stop condition occurs.

Every time a surface is filled, the "Surface Filled" signal is reported to the user. Parameter `LineIndex` reflects the number of lines already written in a partially filled surface. If SeqLength_Ln is not a multiple of `PageLength_Ln`, the last surface will be partially filled. A new surface is selected at each begin of acquisition phase.

Lines are acquired at a rate defined by the line trigger condition. When object speed is variable, it is convenient to generate a line trigger derived from a motion encoder. See "line trigger condition" for more info.

Stopping a Long Page Acquisition Sequence

MultiCam provides solutions for both fixed size and variable size objects.

When acquisition sequence is stopped, the "Surface Filled" signal will be reported as soon as all the acquired data lines are transferred to the host memory. The "MC_SIG_EAS" signal is also reported to the user.

Fixed size objects

When objects have fixed size, it is convenient to specify the number of lines to be acquired during a LongPage acquisition sequence with parameter SeqLength_Ln.

The acquisition sequence will automatically stop after the last acquired line. The last surface might be partially filled. All following conditions must be satisfied:

- SeqLength_Ln > 0
- SeqLength_Ln < (PageLength * 65 535)
- SeqLength_Ln < 2^{31}

This method does not require any specific signal or user actions.

Variable size objects

When objects have variable size, it is convenient to use the signal delivered by a position detector to generate an end trigger condition.

Parameter `EndTrigLine` specify the input pin. Polarity and electrical style are specified by parameters `EndTrigEdge` and `EndTrigCtl`.

In case a single signal is used to indicate the start and the stop object positions, the `EndTrigLine` input pin is the same as the `TrigLine` input pin with opposite polarities.

A programmable page delay can optionally be specified with parameter `EndPageDelay_Ln`. It compensates a trigger advance delivered by a position detector placed away from the camera field of view.

Desactivation of the Channel

The Channel is deactivated by setting `ChannelState` to **IDLE**.

Two flavors are selectable with parameter `BreakEffect`:

- When `BreakEffect`=**FINISH**, the Channel deactivates after normal completion of the current LongPage acquisition sequence. If the trigger event has not yet occurred, the Channel deactivates immediately.
- When `BreakEffect`=**ABORT**, the LongPage acquisition sequence stops immediately.

An automatic channel de-activation is not available for the LongPage operation mode.

12. Understanding the Rate Converter

12.1. How to Use an Encoder?

Encoder Characteristics

An encoder is an electro-mechanical device aimed at delivering a measurement of the web speed.

Usually this device is attached to the rotational axis of one of the rolls transporting the web.

The speed measurement is made known thanks to a pulse train the frequency of which is proportional to the speed.

When the web is traveling in the axial direction at some given speed, the pulses provided by the encoder exhibit a predictable frequency.

We will characterize the encoder operation with a value called the "encoder pitch".

The encoder pitch is simply the distance traveled by the web between two successive pulses provided by the encoder. This distance is not a function of the web speed.

The frequency of the pulses delivered by the encoder while the web is moving will be called the encoder rate. The encoder rate obviously varies with the web speed. The following applies:

$$\mathrm{RateConversionRatio} = \frac{\mathrm{EncoderPitch}}{\mathrm{LinePitch}}$$

Consider that an encoder is provided with an encoder pitch of 0.75 mm.

If the transport mechanical system moves the web at 50 cm/s, the encoder generates a pulse train with a frequency of 667 Hz.

Encoder and Line Triggering

The goal of the industrial vision integrator is to achieve a predictable aspect ratio within the acquired digital image. Usually, but not necessarily, a visual aspect ratio of 1 is desired.

This aspect ratio should be maintained even if the web speed is varying.

The unitary aspect ratio means that the line pitch is equal to the cross web pitch.

We will recall that the line pitch is a function of the web speed and the camera line frequency according to:

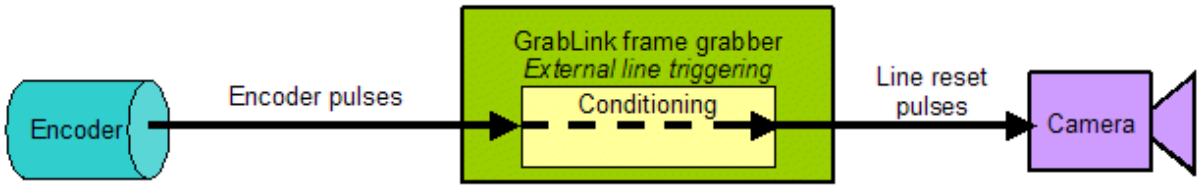
$$\mathrm{LinePitch} = \frac{\mathrm{WebSpeed}}{\mathrm{LineFrequency}}$$

Combining this formula with the encoder frequency expression, we easily obtain:

$$\frac{\mathrm{LineFrequency}}{\mathrm{EncoderFrequency}} = \frac{\mathrm{EncoderPitch}}{\mathrm{LinePitch}}$$

If we can manage to select an encoder having a characteristic pitch equal to the desired line pitch, this simply mean that we just have to impose to the camera a line frequency equal to the encoder frequency.

This is quite simply done by selecting the external line triggering mode and applying the encoder pulses as the line trigger pulses.



For the representative example, we want the line pitch to be 0.39 mm. Consequently, we just have to select an encoder providing a characteristic pitch having the same size.

This impose some mechanical adjustment of the encoder, such as gear wheels swapping.

This burden imposed to the industrial vision integrator can be costly to achieve.

Encoder and Rate Conversion

Instead of constraining the encoder to adapt its performance to the aspect ratio requirements, it is possible to use a built-in feature of the Grablink frame grabber.

Recall that we have to achieve the following:

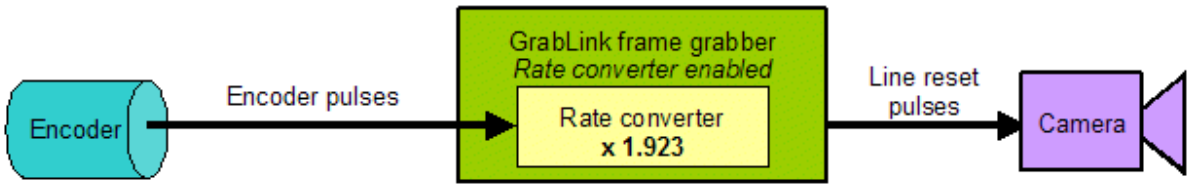
$$\frac{\mathrm{LineFrequency}}{\mathrm{EncoderFrequency}} = \frac{\mathrm{EncoderPitch}}{\mathrm{WebAxialPitch}}$$

Suppose that we have a non adjustable encoder, happening to have a characteristic pitch of 0.75 mm. We still want the line pitch to be 0.39 mm.

This means that we want the following relation to hold true:

$$\frac{\mathrm{LineFrequency}}{\mathrm{EncoderFrequency}} = 1.923$$

The rate converter makes possible to achieve a frequency multiplication by a non integer ratio with an extremely high accuracy. The ratio can be smaller or larger than the unity.



It can be seen that the rate converter feature provides a smart way to solve the problem of maintaining a constant aspect ratio in line scan inspection systems with varying web speed.

Computing the Rate Conversion Ratio

We will define the rate conversion ratio as:

$$RateConversionRatio = \frac{LineFrequency}{EncoderFrequency} = \frac{EncoderPitch}{LinePitch}$$

On another hand, we will define the aspect ratio as:

$$AspectRatio = \frac{CrosswebPitch}{LinePitch}$$

The cross web pitch is a geometrical feature given by:

$$CrossWebPitch = \frac{WebWidth}{NumberOfPixels}$$

The line pitch is a system dependent feature given by:

$$LinePitch = \frac{WebSpeed}{LineFrequency}$$

We well also remember that:

$$EncoderFrequency = \frac{WebSpeed}{EncoderPitch}$$

Combining all this, and assuming we want to achieve a specified aspect ratio not necessarily 1, we obtain:

$$RateConversionRatio = \frac{EncoderPitch \times NumberOfPixels \times AspectRatio}{WebWidth}$$

This formula probably provides the more direct way to evaluate the requirement of the rate converter knowing the basic geometry of the application.

Testing the Formula

Let us apply this formula for our representative application.

We assumes that the width of the observed area on the web is 80 cm, and that the CCD sensor is a 2048 pixel type. The pitch of the encoder in our possession is 0.75 mm. We want a visual aspect ratio of 1.

The calculation yields 1.92.

When the speed web varies between 25 and 50 cm/s, the following table shows the system behavior.

	Slow web speed	Medium web speed	Fast web speed
Web speed	25 cm/s	39 cm/s	50 cm/s
Encoder frequency	$\frac{250}{0.75} = 333\,Hz$	$\frac{390}{0.75} = 520\,Hz$	$\frac{500}{0.75} = 667\,Hz$
Rate conversion ratio	$\frac{0.75 \times 2048 \times 1}{800} = 1.92$		
Camera line frequency	333 ´ 1.92 = 640 Hz	520 ´ 1.92 = 1000 Hz	667 ´ 1.92 = 1280 Hz
Line pitch	$\frac{250}{640} = 0.39\,mm$	$\frac{390}{1000} = 0.39\,mm$	$\frac{500}{1280} = 0.39\,mm$
Cross web pitch	$\frac{800}{2048} = 0.39\,mm$		
Aspect ratio	$\frac{0.39}{0.39} = 1$	$\frac{0.39}{0.39} = 1$	$\frac{0.39}{0.39} = 1$

This clearly proves that the aspect ratio is maintained to 1 independently of the web speed.

12.2. Programming the Rate Converter

Setting the Rate Conversion Ratio in the MultiCam Environment

MultiCam proposes two integer parameters to program the rate conversion ratio, namely `LinePitch` and `EncoderPitch`

`LinePitch` declares the distance between two successively scanned lines on the moving web. The `LinePitch` parameter must be expressed in the same length unit used with `EncoderPitch`.

`EncoderPitch` declares the distance traveled by the web between two successive pulses provided by the encoder.

We recall the formula to evaluate the web axial pitch for a desired aspect ratio:

$$\text{LinePitch} = \frac{\text{CrossWebPitch}}{\text{AspectRatio}} = \frac{\text{WebWidth}}{\text{NumberOfPixels} \times \text{AspectRatio}}$$

The two pitches should be scaled to some integer values in such a way the ratio is as accurate as possible. Those MultiCam parameters should have an integer contents in the range 1 to 10000.

To make this clear, consider the representative example.

The encoder pitch is 0.75 mm, as imposed by the mechanics.

The web axial pitch for a unitary aspect ratio is 0.391 mm.

Enter the figures in an arbitrary unit that will bring the pitches to an integer value while maintaining their ratio.

In this case, correct settings are:

`LinePitch`=**391**

`EncoderPitch`=**750**

This is equivalent to entering the pitches in micrometers.

The rate conversion ratio will be programmed to:

$$\text{RateConversionRatio} = \frac{\text{EncoderPitch}}{\text{WebAxialPitch}} = \frac{\text{EncoderPitch}}{\text{LinePitch}}$$

In the representative application, `RateConversionRatio` = 1.918

This procedure is merely a means to enter a programmable ratio into the MultiCam system with an accuracy in the order of magnitude of 1/1000.

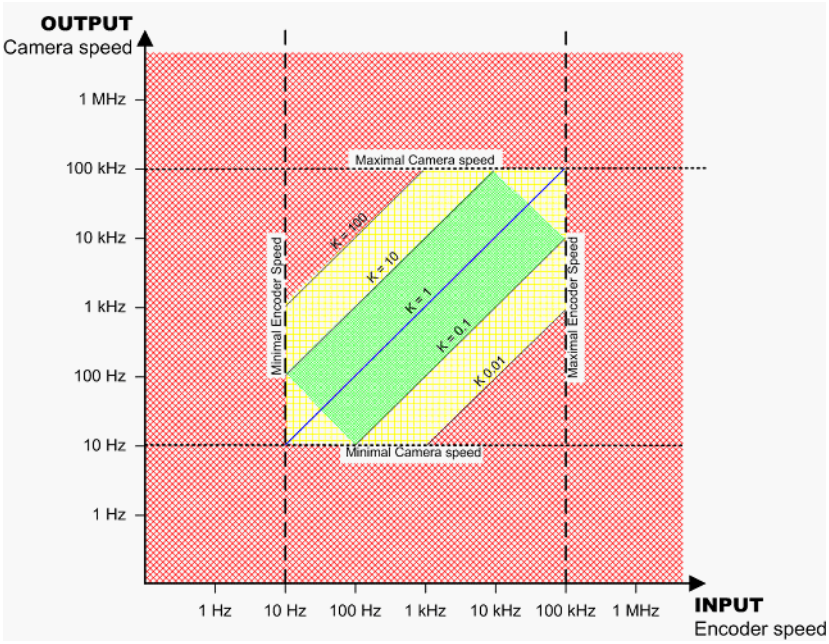
The Operating Range of the Rate Converter

The purpose of the rate converter is to provide trigger pulses to the camera reflecting the frequency of the encoder pulse train multiplied by a fixed amount.

The rate converter will operate in a certain frequency range. We can introduce three equivalent ways to express the operating range of the rate converter.

- The minimum and maximum operating web speed,
- The minimum and maximum operating encoder frequency,
- The minimum and maximum operating camera line frequency.

The third item (camera line frequency in Hertz) has been chosen to represent the speed performance of the line scan system fitted with an encoder and using the Grablink rate converter.



As seen on the picture, the user should use the rate converter in the recommended working area. Inside the valid working area, the rate converter will loose its precision if approaching from the prohibited area. The values are too much different:

- too small input (< 10 Hz) means that the encoder gives too few pulses to obtain any accuracy at the output.
- too high input (> 100 kHz) cannot be generated by an encoder. It must be anyway divided by a too large number to give correct camera pulses.
- too small output (< 10 Hz) means that the line-scan camera receives not enough pulses.
- too high output (> 100 kHz) cannot be followed by a line-scan camera.

As the user can check, those limits are quite exceptional.

The operating range is defined as follows:

$$\mathtt{OperatingRange} = \frac{\mathtt{MaximumSpeed}}{\mathtt{MinimumSpeed}}$$

It is usually above 200, but depends on the value actually entered to program the rate converter.

The user should ensure that the operating range adequately covers the need of his web inspection application. MultiCam parameters are provided to achieve this check.

The speed limit is imposed by the camera readout process or by the minimal exposure time.

Evaluating the Operating Range

The three ways able to represent the speed performance are the web speed, the encoder frequency, the line frequency.

Evaluate the line frequency for a known web speed with the following formula:

$$\mathrm{LineFrequency} = \frac{\mathrm{WebSpeed}}{\mathrm{DownWebPitch}}$$

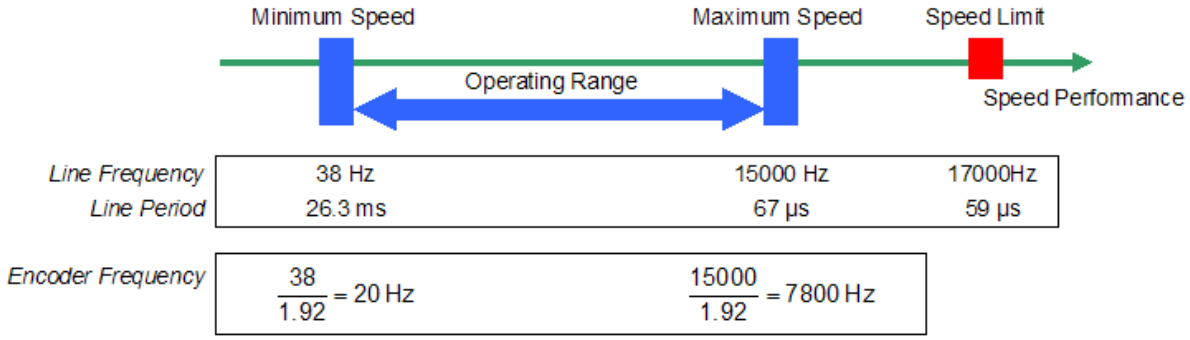
The line frequency and the encoder frequency are linked together with:

$$\mathrm{RateConversionRatio} = \frac{\mathrm{LineFrequency}}{\mathrm{EncoderFrequency}}$$

To have an example, consider a line-scan camera that can generate a top line frequency of 17,000 Hz.

For the sake of security, we will impose a larger maximum frequency of 15,000 Hz.

The operating range with this camera is described below.



The minimum line frequency and encoder frequency figures are returned by the MultiCam driver. The operating range is always above 200.

Dealing with the Operating Range in the MultiCam Environment

Besides the `LinePitch` and `EncoderPitch` MultiCam parameters, you may have to take care of two additional parameters: `PixelClk_Hz` and `MaxSpeed`.

`PixelClk_Hz` is an inherent attribute of the camera you are using. You will not have to change it, unless you are defining a brand new camera, with a non standard clock frequency. This parameter corresponds to the pixel frequency of the camera.

MaxSpeed is set by default to the speed limit of the camera. The limit line frequency is linked to the camera attributes in the following way:

$$LimitLineFrequency \approx \frac{PixelFrequency}{NumberOfPixels}$$

You will change the MaxSpeed parameter only if the default value does not suit your needs.

Once a maximum line frequency is chosen, the MultiCam environment computes for you the corresponding minimum line frequency.

This value is returned through the read-only parameter MinSpeed.

You will use this value to appreciate the operating range of the rate encoder, and determine if it covers your requirements in term of web speed variation.

If your application is working significantly lower than the speed limitation imposed by the camera, it may happen that you have to lower the maximum line frequency.

All the MultiCam parameters representing frequencies are expressed as an integer number of Hertz.

12.3. Programming the Exposure Time

Choosing the Controlled Exposure Mode

Any line scan camera fitted with an electronic shutter is by default controlled through a set of exposure control parameters.

The user can disable the controlled exposure scheme and force the permanent exposure scheme by acting on the hardware line controlling the exposure at the camera level.

This is achieved with specific MultiCam parameters, but is beyond the scope of this document.

Control Parameters

The basic MultiCam parameter to control the exposure duration is Expose_us. The exposure time is expressed in microseconds (10-6 s).

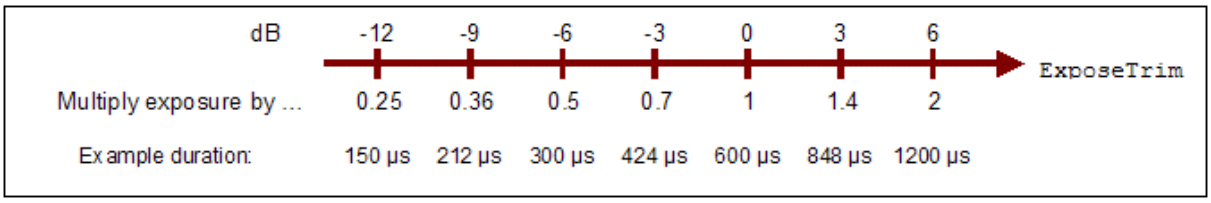
As an example, to program the exposure time to a value of 600 µs, simply set Expose_us to **600**.

A second parameter called ExposeTrim is available. This parameter provides a way to trim, or modify, the exposure duration as originally set by the Expose_us parameter.

The trimming parameter works in the following range:

- The exposure duration can be lowered up to one fourth of the original value,
- The exposure duration can be increased up to two times the original value,

It has been chosen to provide the trimming control on a logarithmic scale. Consequently, the `ExposeTrim` unit is the decibel with a range -12 dB to +6 dB.



Feedback Parameter

The MultiCam parameter `TrueExp_us` can be used to get the effective exposure duration resulting from the combination of the control parameters `Expose_us` and `ExposeTrim`.

The exposure time is expressed in microseconds (10^{-6} s).

A very special care should be taken to avoid to program the exposure time to a value causing a line period limitation.

More specifically, if the exposure time is longer than the readout process duration, the gap between two successive trigger pulses cannot be shorter than the exposure duration.

This situation could happen in the case when the trigger pulses are generated from an external system trigger pulse (external line triggering or rate converted-based triggering). If the trigger pulse rates becomes too large, the line protection mechanisms will enter into action.

13. Channel Parameter User Notes

13.1. Interactivity of Parameters

General rule

Updating a Channel-class parameter takes effect at the next Start of Acquisition Sequence (SAS).

Improved interactivity of parameters for Domino

Concerning parameters shown in next table, the update has a more immediate effect at the next Start of Acquisition Phase (SAP):

Category	Parameter
Exposure control	`Expose_us`
	`ExposeTrim`
Strobe control	`StrobeDur`
	`StrobePos`
Grabber conditioning	`VideoFilter`
	`GainCtl`
	`Gain`
	`GainTrim1`,`GainTrim2` and `GainTrim3`
	`Offset`
	`OffsetTrim1`, `OffsetTrim2` and `OffsetTrim3`
	`InputLut`
	`LutIndex`
	`ColorFormat`

Improved interactivity of parameters for Grablink

Concerning parameters shown in next table, the update has a more immediate effect at the next Start of Acquisition Phase (SAP):

Category	Parameter
Exposure control	Expose_us
	ExposeTrim
Strobe control	StrobeDur
	StrobePos
Grabber conditioning	Period_us
	PeriodTrim
	LinePitch
	EncoderPitch
	LineTrigCtl
	LineTrigEdge
	LineTrigLine
	ConverterTrim
	MaxSpeed

13.2. Camera Specification Category

CamFile

Upon loading a CamFile by setting the CamFile parameter to a character string designating a file name, the following file searching rules are applied:

1. If the given file name is a full pathname, no other location is searched.
2. Otherwise, the following other locations are searched in the specified order:
    1. The application's current working folder
    2. The MultiCam "Camera" folder
    3. All subfolders in the "Camera" folder

CamConfig

CamConfig is an enumerated parameter, the syntax of which summarizes some essential features of the camera.

In some cases, the camera's behavior depends on the camera configuration established through specific hardware settings. (jumpers, switches or remote control serial line).

The combination of a given camera specified by the parameter Camera and the specification of a particular camera configuration with the parameter CamConfig fully describes the cameras sourcing the MultiCam channel.

13.3. Camera Timing Category

PixelClk_Hz

The pixel clock is a signal internal to the camera, according to which pixels are sequentially extracted out of the CCD sensor.

All line-scan cameras and some area-scan cameras are equipped with a pixel clock output. In this case, the frame grabber should be able to sense the pixel clock frequency.

An analog area-scan camera synchronizing the frame grabber by means of the pixel clock is said to work in digital synchronization mode. This is reflected by the parameter `SyncMode` valued to **DIGITAL**.

When an analog area-can camera does not use the pixel clock synchronization, the frame grabber builds an internal sampling frequency, which can differ from the pixel frequency inside the camera. This internal sampling frequency is reflected by the parameter `SampleClk_Hz`.

See also

`PixelClk_Hz`
Area-scan digital camera
Line-scan digital camera

LineRate_Hz

See

`LineRate_Hz`
Area-scan analog camera
Area-scan analog camera with pixel clock
Area-scan digital camera
Line-scan digital camera

LineDur_ns

See

`LineDur_ns`
Area-scan analog camera
Area-scan analog camera with pixel clock
Area-scan digital camera
Line-scan digital camera

Vtotal_Ln

See

Vtotal_Ln
Area-scan analog camera
Area-scan analog camera with pixel clock
Area-scan digital camera

Vactive_Ln

See

Vactive_Ln
Area-scan analog camera
Area-scan analog camera with pixel clock
Area-scan digital camera

Hactive_ns

See

Hactive_ns
Area-scan analog camera
Area-scan analog camera with pixel clock
Area-scan digital camera
Line-scan digital camera

Hactive_Px

See

Hactive_Px
Area-scan digital camera
Line-scan digital camera

VsyncAft_Ln

See

VsyncAft_Ln
Area-scan analog camera with pixel clock
Area-scan digital camera

VCsyncAft_Ln

See

VCsyncAft_Ln

Area-scan analog camera

Area-scan analog camera with pixel clock

VCsyncAft_Ln

See

VCsyncAft_Ln

Area-scan digital camera

Line-scan digital camera

HCsyncAft_ns

See

HCsyncAft_ns

Area-scan analog camera

Area-scan analog camera with pixel clock

VdriveDur_Ln

See

VdriveDur_Ln

Area-scan analog camera

HdriveDur_ns

See

HdriveDur_ns

Area-scan analog camera

HCsyncDur_ns

See

HCsyncDur_ns

Area-scan analog camera

Area-scan analog camera with pixel clock

HCsyncBfr_ns

See

`HCsyncBfr_ns`

Area-scan analog camera

Area-scan analog camera with pixel clock

VdriveDly

See

Area-scan analog camera

HsyncDly_ns

See

`HsyncDly_ns`

Area-scan analog camera with pixel clock

HdriveDly

See

Area-scan analog camera

VsyncPst_Ln

See

Area-scan digital camera

HsyncPst_ns

See

Area-scan digital camera

Line-scan digital camera

VgatePos_Ln

See

Area-scan digital camera

VCgatePos_Ln

See

Area-scan digital camera
Line-scan digital camera

13.4. Camera Features Category

13.5. Trigger Control Category

TrigMode Parameter

The IMMEDIATE option means that the initial TE is equivalent to SAS. As soon as the acquisition sequence is awakened (generally, when the application sets the `ChannelState` parameter to **ACTIVE**), the first acquisition phase takes place.

The HARD option means that the TE occurs upon reception of a valid transition on a hardware line while the channel is armed. The hardware trigger line involved is specified by `TrigLine`, `TrigCtl`, `TrigEdge` and `TrigFilter`.

The SOFT option means that the TE occurs when the application sets the `ForceTrig` parameter while the channel is armed.

The COMBINED option means that the TE occurs indifferently for conditions described for HARD and SOFT.

The `TrigMode` parameter can be modified under application control without returning the channel to the IDLE state.

NextTrigMode Parameter

The REPEAT option means that any subsequent TE is equivalent to the EAP of the previous acquisition phase. As soon as the acquisition phase terminates, the next acquisition phase takes place, including a live-type acquisition mode.

The HARD option means that the TE occurs upon reception of a valid transition on a hardware line while the channel is armed. The hardware trigger line involved is specified by `TrigLine`, `TrigCtl`, `TrigEdge` and `TrigFilter`.

The SOFT option means that the TE occurs when the application sets the `ForceTrig` parameter while the channel is armed.

The COMBINED option means that the TE occurs indifferently for conditions described for HARD and SOFT.

The NextTrigMode parameter can be modified under application control without returning the channel to the IDLE state.

PageLength_Ln Parameter

MultiCam parameter PageLength_Ln specifies the number of lines to acquire in a single surface.

The MultiCam user has to specify the page length according to:

- User application requirements
- Functional limits of MultiCam and Hardware implementations

In addition, following recommendations to select the optimal values should be taken into account.

Rule 1

Current hardware implementations limit PageLength_Ln to a 16-bit value.

$$PageLength\_Ln < 65536$$

Rule 2

The maximum surface transition rate should not exceed 1 kHz.

With MaximumLineAcquisitionRate being the highest line acquisition rate of the application expressed in Hertz:

$$PageLength\_Ln > \frac{MaximumLineAcquisitionRate}{1000}$$

Rule 3

In order to reach large SeqLength_Ln above 65535, it is required to select a PageLength_Ln that satisfies this rule:

$$PageLength\_Ln \geq \frac{SeqLength\_Ln}{65535}$$

Recommendation

For optimal usage of the on-board buffer, the amount of data in a single surface should be in the range 1..4 Mbytes.

Therefore:

$$PageLength\_Ln > \frac{1048576}{LineSize\_bytes}$$

$$PageLength\_Ln < \frac{4194304}{LineSize\_bytes}$$

This recommendation becomes significant only when the average data rate on the PCI interface is approaching the board limits. Namely:

- 90 Mbyte/s on Grablink Value
- 240 Mbytes/s on Grablink Expert 2 and Grablink Quickpack ColorScan

TrigLine Parameter

The NOM option selects one of the three nominal input trigger lines TRX, TRY or TRZ available at the TR-ST connector. The selected hardware port will automatically assumes the X,Y,Z index of the connector linked to the relevant camera for the channel.

See also

`TrigLine`

13.6. Strobe Control Category

StrobeMode Parameter

This parameter establishes the method according to which the illumination control pulse is generated.

- For area-scan cameras, `StrobeMode` relates to the illumination during the **frame** exposure period.
- For line-scan cameras, `StrobeMode` relates to the illumination during the **line** exposure period.

`StrobeMode`	Meaning
NONE	The strobe function is disabled.
AUTO	The strobe function is enabled with an automatic timing control feature. (available only with RG - Grabber Controlled Exposure Mode)
MAN	The strobe function is enabled with a manual timing control feature. (available only with RC - Camera Controlled Exposure Mode)
OFF	The designed `StrobeLine` is set to the inactive level; no more strobe pulses are issued.

The **OFF** value is available only for Grablink boards. When `StrobeMode` is set to **OFF**, no strobe pulses are issued, while when `StrobeMode` is set to **NONE**, no `StrobeLine` is allocated to the channel.

When `StrobeMode` is set to **NONE**, the hardware line dedicated to issuing the strobe pulse is available for general-purpose usage.

When `StrobeMode` is set to **AUTO**, two cases are to be considered:

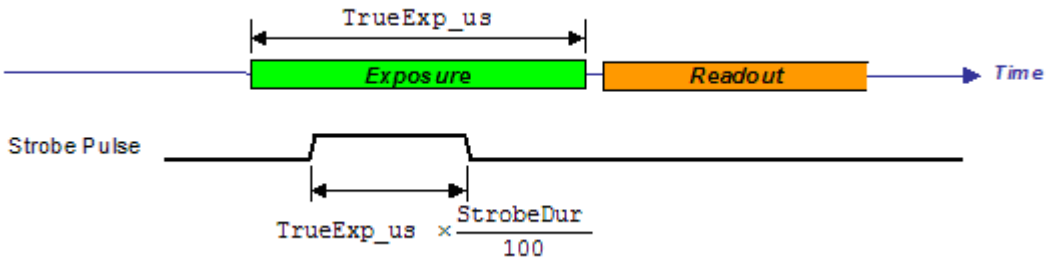
- The grabber positively controls the camera exposure function. The parameter TrueExp_us duration is reported by the grabber. Consequently, StrobeDur and StrobePos are used as specified to define the strobe pulse.
- The exposure function depends on camera settings. The strobe pulse starts upon sending the asynchronous reset pulse to the camera, and ends upon receiving the vertical validation pulse from the camera. No other strobe pulse is possible. The StrobeDur and StrobePos parameters are not used.

When StrobeMode is set to **MAN**, the parameters StrobeDur and StrobePos are used as specified to define the strobe pulse, using the TrueExp_us duration. However, the application is requested to write the exposure duration into TrueExp_us beforehand.

An equivalent scheme applies to line-scan operation.

StrobeDur Parameter

StrobeDur is expressed as a percentage of the expose width pulse.



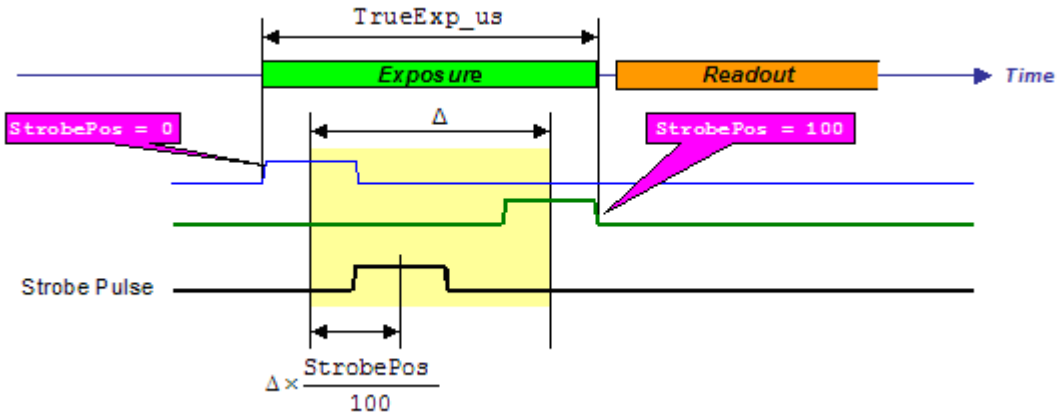
StrobePos Parameter

StrobePos is expressed as a percentage of the expose width pulse.

A value of 0 % establishes the earliest position. The leading edge of strobe pulse is simultaneous with the beginning of exposure.

A value of 100 % establishes the latest position. The trailing edges of strobe pulse and exposure are simultaneous.

A value of 50 % means that the strobe pulse is located in the middle of the exposure period.



In any situation, the strobe pulse is issued during the exposure period.

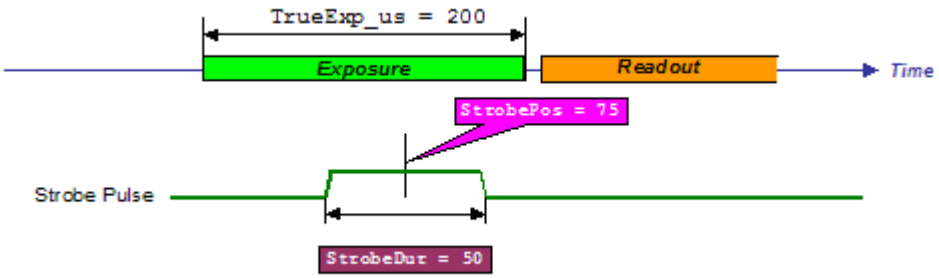
PreStrobe_us Parameter

`PreStrobe_us` is expressed in microseconds. Its function is to advance the leading edge of strobe pulse.

In following examples, consider the values:

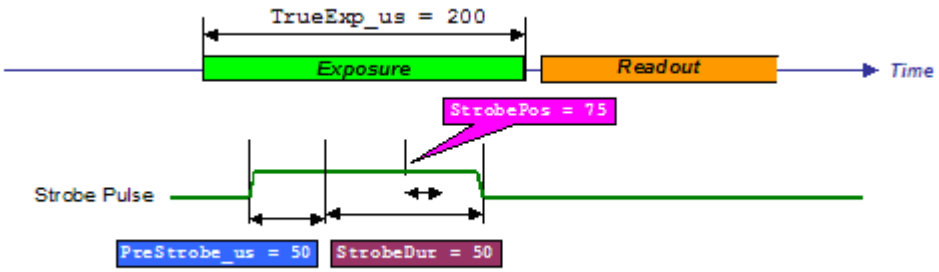
- `StrobeMode`=**MAN**
- `TrueExp_us`=**200**
- `StrobeDur`=**50**
- `StrobePos`=**75**

This situation is drawn below. Consequently, the duration of strobe pulse is 100 µs. The leading edge of strobe pulse appears 75 µs after the beginning of exposure.



The usage of `PreStrobe_us` parameter allows to start earlier the strobe pulse.

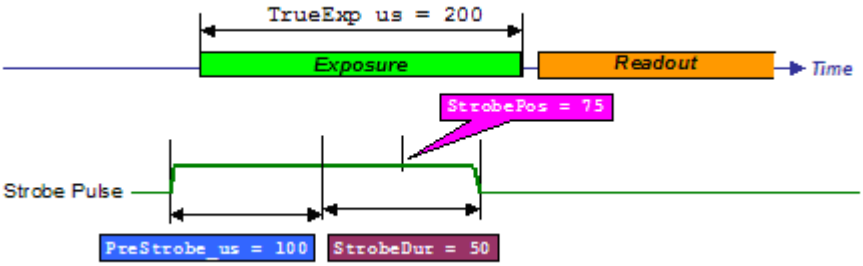
A first example with `PreStrobe_us`=**50** is drawn below.



The leading edge of strobe pulse appears 25 µs after the beginning of exposure.

Increasing `PreStrobe_us` can lead to begin the strobe pulse before the exposure starts.

A second example with `PreStrobe_us`=**100** is drawn below. In this case, the strobe pulse begins 25 µs before the exposure starts.



StrobeLine Parameter

The NOM option selects the of the three nominal output strobe lines STX, STY or STZ available at the TR-ST connector. The selected hardware port will automatically assumes the X,Y,Z index of the connector linked to the relevant camera for the channel.

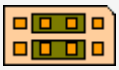
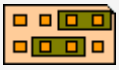
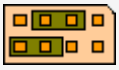
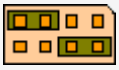
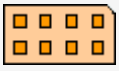
See also

`StrobeLine`

13.7. Grabber Configuration Category

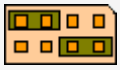
JumperCK Parameter

Depending on the `Camera` and `CamConfig` presently selected, the `JumperCK` parameter indicates how to configure the CK jumper block for the relevant channel.

JumperCK	CK block strapping	Description
CKDPOS		The channel receives a non-inverted differential pixel clock from the camera through pin 9 (CK+) and pin 10 (CK-)
CKDNEG		The channel receives an inverted differential pixel clock from the camera through pin 9 (CK+) and pin 10 (CK-)
CKSPOS		The channel receives a non-inverted single-ended pixel clock from the camera through pin 9 (CK+) with ground return at pin 3 (GND)
CKSNEG		The channel receives an inverted single-ended pixel clock from the camera through pin 9 (CK+) with ground return at pin 3 (GND)
ZLANE		Pin 9 and pin 10 of the connector are feeding the channel with a third video lane borrowed to the Z connector
EMPTY		Pin 9 and pin 10 of the channel connector are unused

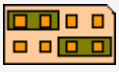
JumperH Parameter

Depending on the `Camera` and `CamConfig` presently selected, the `JumperH` parameter indicates how to configure the H jumper block for the relevant channel.

JumperH	H block strapping	Description
TTL		The pin 14 (HIO) and pin 15 (GATE) of the connector feeding the channel can be used as input or output in TTL format
DPOS		The pin 14 (HIO) and pin 15 (GATE) of the connector are used to sense a non inverted differential horizontal reference from the camera feeding the channel
DNEG		The pin 14 (HIO) and pin 15 (GATE) of the connector are used to sense an inverted differential horizontal reference from the camera feeding the channel

JumperV Parameter

Depending on the `Camera` and `CamConfig` presently selected, the `JumperV` parameter indicates how to configure the V jumper block for the relevant channel.

JumperV	V block strapping	Description
TTL		The pin 4 (VIO) and pin 5 (EXP) of the connector feeding the channel can be used as input or output in TTL format

DPOS		The pin 4 (VIO) and pin 5 (EXP) of the connector are used to sense a non inverted differential vertical reference from the camera feeding the channel
DNEG		The pin 4 (VIO) and pin 5 (EXP) of the connector are used to sense an inverted differential vertical reference from the camera feeding the channel

SyncMode Parameter

ANALOG

This operating mode is installed when the only timing information available from the camera is the composite video signal.

Consider a MultiCam channel owning such a camera. The timing unit controlling the channel automatically assumes the analog synchronization mode. This means that the timing unit extracts the synchronization out of the video signal supplied through the relevant camera connector.

The timing unit locks itself to the camera synchronization with a phase lock loop operating at the camera horizontal frequency.

A special care is taken to keep the phase jitter inherent to this operational mode at a minimum.

DIGITAL

This operating mode is installed when the camera delivers the timing information through a set of digital lines.

Consider a MultiCam channel owning such a camera. The timing unit controlling the channel automatically assumes the digital synchronization mode. This means that the timing unit exploits the set of digital timing signals supplied by the camera through the associated connector.

Namely, the timing unit uses a pixel clock, a horizontal reference and a vertical reference issued by the camera.

MASTER

This operating mode is installed when the camera is due to receive its timing information from the part of the board. The board is the timing master of the camera.

Consider a MultiCam channel owning a camera working in this manner. The timing unit controlling the channel automatically assumes the master synchronization mode. This means that the timing units delivers horizontal and vertical drive synchronizing signals to the camera through the associated connector.

The basic timing reference for a channel working in the master synchronization mode is an internal resource called the clock synthesizer.

13.8. Grabber Timing Category

GrabWindow Parameter

See

`GrabWindow`

How to Define the Grabbing Window?

OffsetX_Px Parameter

See

`OffsetX_Px`

How to Define the Grabbing Window?

OffsetY_Ln Parameter

See

`OffsetY_Ln`

How to Define the Grabbing Window?

WindowX_Px Parameter

See

`WindowX_Px`

How to Define the Grabbing Window?

WindowY_Ln Parameter

See

`WindowY_Ln`

How to Define the Grabbing Window?

SampleClk_Hz Parameter

The sample clock is a signal internal to the frame grabber, according to which pixels are sampled out of the video signal delivered by the camera.

When the camera is fitted with a pixel clock output, the frame grabber sampling clock exactly aligns itself to the camera pixel clock (digital synchronization scheme).

Otherwise, the frame grabber uses the analog synchronization scheme, and builds a sampling frequency of its own through a phase locking mechanism.

The camera pixel clock frequency can be declared with the parameter `PixelClk_Hz`. The sampling frequency is not necessarily equal to the pixel frequency.

The sampling clock is also used in the Gamma frame grabber to operate all timing dependant devices. This explains why the sampling clock is also referred to as the timing clock, or Tk.

The duration of the timing clock period is often called the TCU (Timing Clock Unit). This is a customary way to quantify duration items within the frame grabber.

See also

`SampleClk_Hz`

13.9. Cluster Category

ImageSizeX Granularity

With Grablink series

Following restrictions apply on Grablink Value and Grablink Value cPCI:

- For 8-bit pixels, the `ImageSizeX` granularity is 4.
- For 16-bit pixels, the `ImageSizeX` granularity is 2.
- For 24-bit pixels, the `ImageSizeX` granularity is 4.

Following restrictions apply on Grablink Expert 2 and Grablink Expert 2 cPCI:

- For 8-bit pixels, the `ImageSizeX` granularity is 8.
- For 16-bit pixels, the `ImageSizeX` granularity is 4.
- For 24-bit pixels, the `ImageSizeX` granularity is 8.

Following restrictions apply on Grablink Quickpack ColorScan and Grablink Quickpack ColorScan cPCI:

- For 24-bit pixels, the `ImageSizeX` granularity is 8.

13.10. Selecting the Pixel Data Output Format

In MultiCam, the pixel data output format is entirely characterized by the adjust-level Channel parameter of the Cluster category ColorFormat.

Refer to the MultiCam Storage Formats topic for detailed information on the pixel data formats.

Refer to the ColorFormat reference topic "for the list of applicable values for each board individually.

PICOLO Series

In the Picolo boards implementation, the pixel data output format is selectable independently from the type of camera.

For instance, it is allowed to select color output formats with monochrome cameras.

DOMINO Series

In the Domino boards implementation, the pixel data output format is linked to the type of camera.

RGB planar and packed formats are available only when a RGB camera is attached to the board.

Y formats are available only when a monochrome camera is attached to the board.

Domino Melody, Domino Harmony, and Domino Symphony implement a 10-bit ADC's and pixel data interpolation. They provide the choice between three bit depth for the pixel data output: 8-bit (default), 10-bit, and 16-bit.

Available values of ColorFormat according to the camera class

Camera class	Domino Melody Domino Symphony	Domino Harmony
Analog monochrome	Y8, Y10, Y16	Y8, Y10, Y16
Analog RGB	-	RGB24, RGB32, RGB24PL, RGB30PL, RGB48PL

GRABLINK Series

Available values of ColorFormat according to the camera class

Camera class	Grablink Value	Grablink Express	Grablink Full
Monochrome 8-bit	Y8	Y8	Y8
Monochrome 10-bit	Y8, Y12, Y16	Y8, Y12, Y16	Y8, Y12, Y16
Monochrome 12-bit	Y8, Y12, Y16	Y8, Y12, Y16	Y8, Y12, Y16
Monochrome 14-bit	Y8, Y14, Y16	Y8, Y14, Y16	-
Monochrome 16-bit	Y8, Y16	Y8, Y16	Y8, Y16
RGB 24-bit	RGB24, RGB32 RGB24PL	RGB24, RGB32 RGB24PL	RGB24, RGB32 RGB24PL
RGB 30-bit	-	RGB24, RGB32 RGB24PL, RGB30PL RGB48PL	RGB24, RGB32 RGB24PL, RGB30PL RGB48PL
RGB 36-bit	-	RGB24, RGB32 RGB24PL, RGB36PL RGB48PL	RGB24, RGB32 RGB24PL, RGB36PL RGB48PL
Bayer CFA 8-bit	BAYER8	BAYER8	RGB24, RGB32 RGB24PL, RGB36PL RGB48PL
Bayer CFA 10-bit	BAYER8, BAYER10 BAYER16	BAYER8, BAYER10 BAYER16	BAYER8, BAYER10 BAYER16 RGB24, RGB32 RGB24PL, RGB30PL RGB48PL
Bayer CFA 12-bit	BAYER8, BAYER12 BAYER16	BAYER8, BAYER12 BAYER16	BAYER8, BAYER12 BAYER16 RGB24, RGB32 RGB24PL, RGB36PL RGB48PL
Bayer CFA 14-bit	BAYER8, BAYER14 BAYER16	BAYER8, BAYER14 BAYER16	-
Bayer CFA 16-bit	BAYER8, BAYER16	BAYER8, BAYER16	BAYER8, BAYER16

13.11. SignalEnable Parameter

To control a specific signal, designate the corresponding item of the collection parameter. See MultiCam Collection Parameter and Enabling Signals for more information.

Only enabled signals will trigger a callback or satisfy any of the waiting functions.

Compatibility issue

Before MultiCam 2.3 release, the `SignalEnable` parameter was not a collection. It accepted three predefined values:

`SignalEnable`	Description
NONE	No signal is enabled
PROCESSING	The Surface Processing signal is enabled
FILLED	The Surface Processing and Surface Filled signals are enabled

Element #0 of the collection accepts any of the above values to ensure backward compatibility with releases before MultiCam 2.3.

13.12. AcqTimeOut_ms Parameter

When an acquisition does not complete within the specified timeout duration, the **Acquisition Failure** signal is fired and the channel is disabled. The channel must be deleted and created again to allow for further acquisitions.

14. The Surface Class

15. The Surface Class

The surface is a container where a 2D image can be stored.

In most situations, the surface is a buffer in the host memory. Other types of surfaces may be defined, such as the hardware frame buffer located inside a frame grabber. In the particular case of a line-scan camera, the surface can be used as a circular buffer. This implies that, although the surface is 2D-limited, the incoming data flow is continuous and virtually unlimited.

Regarding the acquisition process, the surface is the destination where the grabbed images from the cameras are recorded. The overall goal of the MultiCam driver is to provide flexible channels to route images coming from a camera towards a specified surface.

15.1. Surface Creation

The Surface class groups all MultiCam parameters dedicated to the definition of memory buffers for image or data storage. A Surface object is an instance of the Surface class represented by a dedicated set of such parameters that uniquely describe the surface.

Several surfaces can exist simultaneously. A process called "surface creation" is applied to define a new surface. A created surface is entirely characterized by a corresponding instance of the Surface class in the MultiCam environment.

Surfaces can be deleted by their owning application with an appropriate API function.

15.2. Surface Conversion

Delete this text and replace it with your own content.

PART VI

COMMAND-LINE INSTALLATION PROCEDURE

1. Command-Line Installation Procedure

You may want to integrate the boards drivers and MultiCam tools installation into your own application distribution.

MultiCam setup program can be called in command-line mode with your installation options. In this mode, the MultiCam installation program does not prompt for user action and does not display any dialog box.

The Response File

A command-line installation is created in two steps.

1. Use the graphical user interface to record a response file.
   - Call the setup program with the **/r** flag
   - Use the **/f1** flag to specify the response file name and path. By default, the "Setup.iss" file is created inside the Windows system folder.
2. Recall the response file.
   - Call the setup program with the **/s** flag. The installation is launched in the silent mode, that is no window nor dialog box will appear.
   - Use the **/f1** flag to specify the response file name and path. By default, the "Setup.iss" file is recalled from the Windows system folder.
   - Use the **/z"ForceInstall"** flag to forces the removal of an already installed version before executing the setup file, even if the already installed version is newer.

Note: *There is no space between the flag and its argument.*
/f1"Setup.iss" *will work.*
/f1 "Setup.iss" *will not work.*

Installation Removal

To automatically remove installed tools, call the setup program with the **/removeonly** flag.

Use the /s flag to launch the removal program in the silent mode. In this mode, no window nor dialog box will appear.

Reboot during Installation

If during the record of the response file, you accepted the reboot at the end of the installation, this reboot is stored in the response file and it will take place automatically during the

installations using this response file.

If you did not accept the reboot, the reboot will not take place automatically. The reboot must be performed by your application. In this case, the `[HKEY_LOCAL_MACHINE\SOFTWARE\Euresys\Common] "RebootNeeded"` registry entry should be checked. If it exists and is set to 1, then it should be replaced by 0, and the system must be rebooted.

Error Reporting

After the command line installation, the following registry key is updated and holds the installation status: `[HKEY_LOCAL_MACHINE\SOFTWARE\Euresys\Common\LastInstallError]`.

- The `ErrorCode` DWORD identifies the error:
  - **0** - There is no error.
  - **1** - The user is not administrator.
  - **2** - There is not enough space on the target disk.
  - **3** - The command line is invalid.
  - **4** - There is a newer product version already installed.
- The `Cause` string gives a wording of the error.
- The `Source` string identifies the installer that caused the error.
- The `ErrorTime` string gives the time and date of the error.