CXP-12 Interface Card 2C

Applet Feature Reference Manual for

FrameGrabberTest

Functional Description
For pylon or GenTL Usage

Document Number: AW001885
Part Number: 000 (English)
Document Version: 02
Release Date: 25 February 2025
Applet Version 3.5.6.1

Contacting Basler Support Worldwide

Europe, Middle East, Africa
Tel. +49 4102 463 515
support.europe@baslerweb.com

The Americas
Tel. +1 610 280 0171
support.usa@baslerweb.com

Asia-Pacific
Tel. +65 6367 1355
support.asia@baslerweb.com

Singapore
Tel. +65 6367 1355
support.asia@baslerweb.com

Taiwan
Tel. +886 3 558 3955
support.asia@baslerweb.com

China
Tel. +86 10 6295 2828
support.asia@baslerweb.com

Korea
Tel. +82 31 714 3114
support.asia@baslerweb.com

Japan
Tel. +81 3 6672 2333
support.asia@baslerweb.com

https://www.baslerweb.com/en/sales-support/support-contact

Supplemental Information

Acquisition Card Documentation:
https://docs.baslerweb.com/acquisition-cards
Frame Grabber Documentation:
https://docs.baslerweb.com/frame-grabbers
Framegrabber SDK Documentation:
https://docs.baslerweb.com/frame-grabbers/framegrabber-sdk-overview.html

Table of Contents

Chapter 1. Introduction

This document provides detailed information on the Silicon Software Test Applet "FrameGrabberTest" for CXP-12 Interface Card 2C frame grabbers.

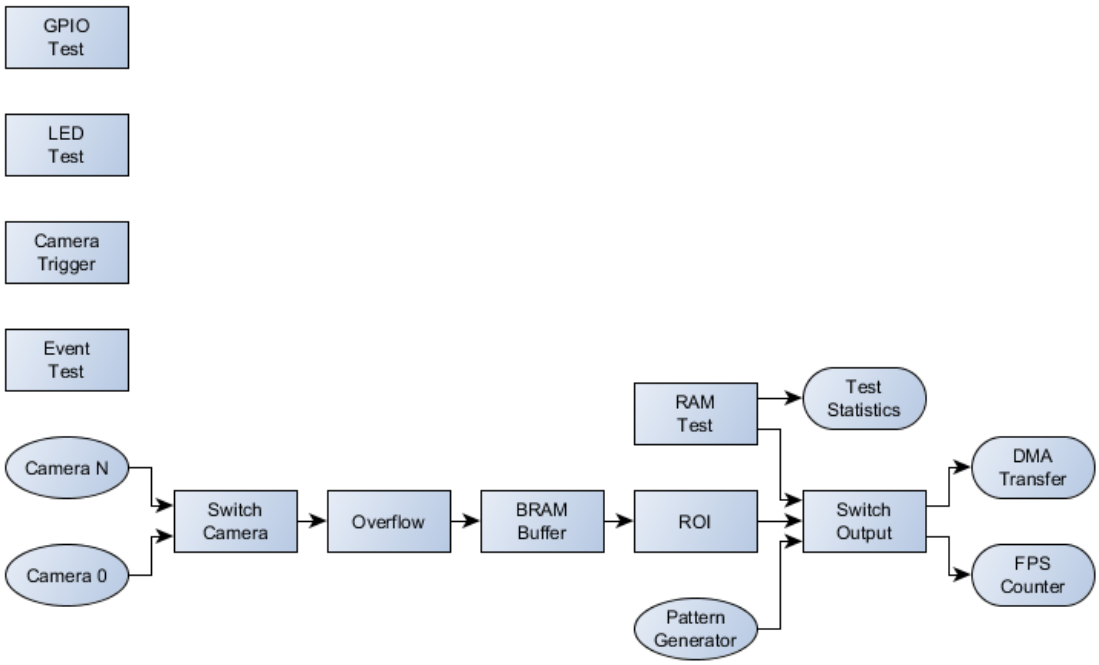
This applet is a frame grabber test applet. Its intention is to test the hardware. You shall not use this applet for your final image processing application. Use AcquisitionApplets or VA Applets instead.

The applet comprises the following functions:

- DMA Performance test: Different image dimensions for varying PC memory sizes and interrupt rates.
- RAM Test: Check for errors and bandwidth of the on board DRAM.
- Camera: Check camera port image acquisition
- Camera Trigger: Send trigger signals to camera
- GPIO: Monitor the GPIs and set the GPOs
- Event test: Generate a sofware callback event
- Monitoring: FPGA Temperature, Power, PoCL, ... (See Chapter 13, '*Miscellaneous*')

The following diagram shows the functional blocks of the applet.

Figure 1.1. Block Diagram of the applet



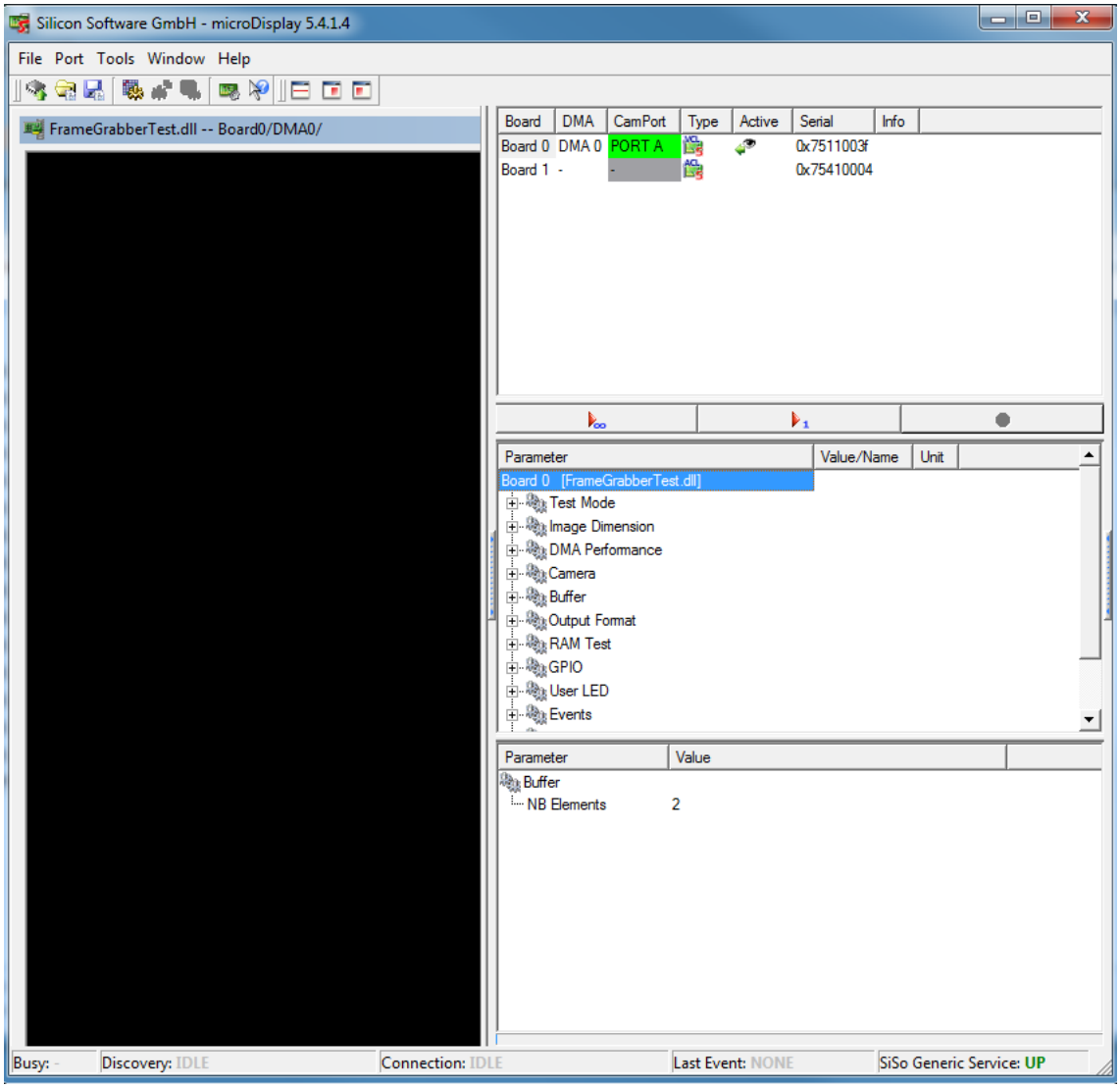
Chapter 2. Test Procedure in microDisplayX

In the following, the steps to test the hardware with the applet FrameGrabberTest in microDisplayX are explained. Of course, you can also integrate the tests in your own programs with the Framegrabber SDK, GenTL or pylon API.

2.1. Choose Your Test Procedure

In the following we describe how you can choose the single test procedures, which are listed in the introduction text. In Fig. 2.1 you see a list of parameters ('TestMode' to 'Events'). To set the test procedures we use these parameters. In chapter 3 to 13 their functionality and settings are explained in detail.

Figure 2.1. Parameters of the applet 'FrameGrabbertest.dll' in 'microDisplay'



2.1.1. DMA Performance Test

To test the DMA performance set the parameter 'OutputSelect' under 'Test Mode' with Right-Mouse-Click to 'DMA Performance'. You can choose the image dimensions for the test with the parameters 'Width' and

'Height' (under 'ImageDimensions' (see Chapter 4)). With Right-Mouse-Click on 'DMA Performance Output Mode' under 'DMA Performance' (see Chapter 5) you can choose between maximum DMA performance ('DMA Performance Maximum') and user defined DMA framerate ('DMA Performance Custom Framerate'). For the latter set the frame rate with the parameter 'DMA Performance Framerate'. In addition you have the possibility to stop completely the DMA output in setting 'DMA Performance Output Mode' to 'DMA Performance Off'. You can monitor the current DMA framerate with the read only parameter 'FPS' under 'Output Format'.

2.1.2. RAM Test

To test the RAM performance of the RAM modules set the parameter 'OutputSelect' under 'Test Mode' with Right-Mouse-Click to 'RAM Difference' or 'RAM Errors' for the corresponding RAM module. In 'RAM Difference' mode (difference between expected and read value from RAM) you can see RAM defects in output values, which are not zero. In 'RAM Errors' mode a white pixel indicates an error (see also chapter 3). Output image size is does not exceed 1 MiB. Suggested display width in 'RAM Difference' mode is to 1024 pixels (parameter 'Width' under 'ImageDimensions' (see Chapter 4)). You can choose display height with parameter'Height' under 'ImageDimensions' (see Chapter 4)). If display size exceeds output image size the output images are split to several displayed images. You can detect RAM errors, when RAM data processing is enabled, but the read-only parameter 'Image Count' of the corresponding RAM module does not increase. Defects of RAM modules can also be observed with the read-only parameters 'Error Occurred', 'Error COUNT_RAM0' to 'Error COUNT_RAM3'. You can also check the total RAM bandwidth using parameter *RamBandwidth*. You should set the output to **`RAM Differences`** or another test mode to esure that the DMA does not limit the DRAM.

2.1.3. Camera Test/Camera Trigger Test

To test the camera port image acquisition set the parameter 'OutputSelect' under 'Test Mode' with Right-Mouse-Click to 'Camera'. You can choose the image ROI dimensions for the test with the parameters 'Width' and 'Height' (under 'ImageDimensions' (see Chapter 4)). Select your camera port with the parameter 'Camera Port' (under 'Camera') and choose your 'Camera Input Format'(see also Chapter 6). The read-only parameters 'Buffer fill level' and 'Buffer overflow' indicate the fill level and overflow of the BRAM between camera and DMA output (see also Chapter 7). It helps to identify problems during image acquisition. You have the possibility to send trigger signals to the the camera by setting the parameters 'FG_CCSEL0' to 'FG_CCSEL3' (under parameter 'Camera').

2.1.4. GPIO

You can monitor the digital inputs with the parameters 'GPI Status bitmask' and 'Front GPI Status bitmask' (under parameter 'GPIO'). Bit 0 to bit N represent digital inputs 0 to N. Find more information on these parameters in sections 10.1 and 10.2. You can set the digital outputs of the frame grabber with the parameters 'Output bitmask' and 'Front Output bitmask'. Values between 0 to 255 and 0 to 37 are possible. Here also bit 0 to bit N represent digital outputs 0 to N. You find further information on these parameters in sections 10.3 and 10.4.

2.1.5. Event Test

With the parameter 'Generate a Test Event' you can start a software callback event for test purposes. More information you find in Chapter 12.

2.1.6. Monitoring

You have the possibility to monitor several Applet and frame grabber parameters under 'Miscellaneous'. There you find e.g. information on the 'Applet version', 'Applet revision', 'Build time' and several more. Also current FPGA temperature, voltage and link speed information are located there.

Chapter 3. CoaXPress

3.1. SystemmonitorStreamPacketSize

Returns the stream packet size in bytes. Range: between 4 and 65535 bytes in steps of 4 bytes.

Table 3.1. Parameter properties of SystemmonitorStreamPacketSize

Property	Value
Name	**SystemmonitorStreamPacketSize**
Display Name	**Systemmonitor Stream Packet Size**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 3.1. Usage of SystemmonitorStreamPacketSize

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorStreamPacketSizeSelector = i;
    value_ = SystemmonitorStreamPacketSize;
}
```

3.2. SystemmonitorCxpStandard

Returns the version of the used CXP standard.

Table 3.2. CXP Standard Version

CXP Standard Version		
CXP_1_0		
CXP_1_1_1		
CXP_2_0		
Unknown		

Table 3.3. Parameter properties of SystemmonitorCxpStandard

Property	Value
Name	**SystemmonitorCxpStandard**
Display Name	**Systemmonitor CXP Standard**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Beginner**

Example 3.2. Usage of SystemmonitorCxpStandard

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorCxpStandardSelector = i;
    value_ = SystemmonitorCxpStandard;
}
```

3.3. CxpStreamPacketCount

This parameter counts the amount of received stream packets. Bits [29:0] count the number of packets. Bit [30] is set when a counter overflow occurs. Range: 0 to 4294967295 (32 bit).

Table 3.4. Parameter properties of CxpStreamPacketCount

Property	Value
Name	**CxpStreamPacketCount**
Display Name	**CXP Stream Packet Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 3.3. Usage of CxpStreamPacketCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpStreamPacketCountSelector = i;
    value_ = CxpStreamPacketCount;
}
```

Chapter 4. Test Mode

4.1. OutputSelect

The frame grabber test applet offers several test modes

- DMA Performance Output
- Camera Image Output
- RAM Test Output

The DMA performance output uses a pattern generator which is directly connected to the DMA and can support the full bandwidth. Use parameters *Width* and parameter *Height* set the generator and DMA output size. In this mode data will always be output at the maximum possible datarate which is capable by the PCIe interface and PC.

If you select camera output, the camera images are forwarded to the output. Again use parameters *Width* and *Height* to set the output size.

If you select the RAM test you need to note the following

- RAM Difference output:

  Will output the absolute difference between the expected and read value from RAM. This should always be 0. Otherwise there is a RAM defect.

- RAM Error output:

  Will output a white pixel for any error.

  In this mode, the RAM data width is used so that the output is not 8 bit pixel. Instead for each RAM data one pixel is output. For example if your RAM has a data width of 128 bit, 16 8 bit pixel are merged together.

- The output image size will always be the size of the RAM. For example 512MiB or 256MiB.

  Parameter *Width* will set a display width. The width is constant depending on difference or error output. In difference output the width should always be 4096.

  Parameter *Height* will set a display height. If the actual image height exceeds the height of the RAM, the image is split into many several images.

Table 4.1. Parameter properties of OutputSelect

Property	Value	
Name	**`OutputSelect`**	
Display Name	**`Output Select`**	
Interface	**`IEnumeration`**	
Access policy	**`Read/Write/Change`**	
Visibility	**`Beginner`**	
Allowed values	**`DmaPerformance`**	DMA Performance
	`Camera`	Camera
	`Ram0Difference`	RAM 0 Difference
	`Ram0Errors`	RAM 0 Errors
Default value	**`DmaPerformance`**	

Example 4.1. Usage of OutputSelect

```
/* Set */ OutputSelect = DmaPerformance;
/* Get */ value_ = OutputSelect;
```

Chapter 5. Image Dimension

5.1. Width

Set the output width using this parameter. The width setting defines the size for DMA test and camera ROI.

The DMA output is defined using parameter *OutputSelect*.

Note that for RAM test output the width and height settings simply define the display size.

Table 5.1. Parameter properties of Width

Property	Value
Name	**Width**
Display Name	**Width**
Interface	**IInteger**
Access policy	**Read/Write**
Visibility	**Beginner**
Allowed values	**Minimum 32** **Maximum 16384** **Stepsize 32**
Default value	**1024**
Unit of measure	**pixel**

Example 5.1. Usage of Width

```
/* Set */ Width = 1024;
/* Get */ value_ = Width;
```

5.2. Height

Set the output height using this parameter. The height setting defines the size for DMA test and camera ROI.

The DMA output is defined using parameter *OutputSelect*.

Note that for RAM test output the width and height settings simply define the display size.

Table 5.2. Parameter properties of Height

Property	Value
Name	**Height**
Display Name	**Height**
Interface	**IInteger**
Access policy	**Read/Write**
Visibility	**Beginner**
Allowed values	**Minimum 1** **Maximum 65536** **Stepsize 1**
Default value	**1024**
Unit of measure	**pixel**

Example 5.2. Usage of Height

```
/* Set */ Height = 1024;
/* Get */ value_ = Height;
```

Chapter 6. DMA Performance

6.1. DmaPerformanceOutputMode

The DMA Performance test can be used in several modes.

- Off: No data will be output
- Maximum: The image generator will run at maximum speed and data is output as fast as the DMA transfer allows. To obtain the maximum possible bandwidth of the DMA use this mode.
- Custom Framerate: Allows you to specify any framerate in the allowed range. Use parameter *DmaPerformanceFramerate* to define the framerate.

Table 6.1. Parameter properties of DmaPerformanceOutputMode

Property	Value	
Name	**DmaPerformanceOutputMode**	
Display Name	**DMA Performance Output Mode**	
Interface	**IEnumeration**	
Access policy	**Read/Write/Change**	
Visibility	**Beginner**	
Allowed values	**DmaPerformanceOff**	DMA Performance Off
	DmaPerformanceMaximum	DMA Performance Maximum
	DmaPerformanceCustomFramerate	DMA Performance Custom Framerate
Default value	**DmaPerformanceMaximum**	

Example 6.1. Usage of DmaPerformanceOutputMode

```
/* Set */ DmaPerformanceOutputMode = DmaPerformanceMaximum;
/* Get */ value_ = DmaPerformanceOutputMode;
```

6.2. DmaPerformanceFramerate

For the DMA test you can specify a custom framerate. Set parameter *DmaPerformanceOutputMode* to **FG_DMA_PERFORMANCE_CUSTOM_FRAMERATE** so that this parameter is enabled.

You can use any framerate. However, if the defined framerate exceeds the maximum possible by the DMA, the framerate is decreased.

Table 6.2. Parameter properties of DmaPerformanceFramerate

Property	Value
Name	**DmaPerformanceFramerate**
Display Name	**DMA Performance Framerate**
Interface	**IFloat**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Minimum 0.931323** **Maximum 1.25E8** **Stepsize 8.0E-9**
Default value	**100.0**
Unit of measure	**fps**

Example 6.2. Usage of DmaPerformanceFramerate

```
/* Set */ DmaPerformanceFramerate = 100.0;
/* Get */ value_ = DmaPerformanceFramerate;
```

Chapter 7. Camera

7.1. CameraPort

Select the camera port index.

Table 7.1. Parameter properties of CameraPort

Property	Value
Name	**CameraPort**
Display Name	**Camera Port**
Interface	**IInteger**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Minimum** 0 **Maximum** 1 **Stepsize** 1
Default value	**0**

Example 7.1. Usage of CameraPort

```
/* Set */ CameraPort = 0;
/* Get */ value_ = CameraPort;
```

7.2. TriggercameraOutSelect

Table 7.2. Parameter properties of TriggercameraOutSelect

Property	Value
Name	**TriggercameraOutSelect**
Display Name	**CXP Trigger Select**
Interface	**IEnumeration**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**On** On **Off** Off
Default value	**Off**

Example 7.2. Usage of TriggercameraOutSelect

```
/* Set */ TriggercameraOutSelect = Off;
/* Get */ value_ = TriggercameraOutSelect;
```

Chapter 8. Buffer

8.1. Filllevel

Indicates the buffer filllevel of the BRAM based buffer between the camera interface and DMA. Use this value if you output camera images to the DMA.

Table 8.1. Parameter properties of Filllevel

Property	Value
Name	**Filllevel**
Display Name	**Buffer fill level**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 0** **Maximum 100** **Stepsize 1**
Unit of measure	**%**

Example 8.1. Usage of Filllevel

```
/* Get */ value_ = Filllevel;
```

8.2. Overflow

Indicates a buffer overflow. The parameter is automatically reset when read. Note that microDisplay continuously reads all parameters so that you might not see the occurrence of an overflow. Have a look at the event counter in this case.

The overflow shows buffer overflows of the BRAM based buffer between the camera interface and DMA.

Table 8.2. Parameter properties of Overflow

Property	Value
Name	**Overflow**
Display Name	**Buffer overflow**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 0** **Maximum 1** **Stepsize 1**

Example 8.2. Usage of Overflow

```
/* Get */ value_ = Overflow;
```

Chapter 9. Output Format

9.1. Format

Table 9.1. Parameter properties of Format

Property	Value
Name	**Format**
Display Name	**Output Format**
Interface	**IEnumeration**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Mono8** Mono 8
Default value	**Mono8**

Example 9.1. Usage of Format

```
/* Set */ Format = Mono8;
/* Get */ value_ = Format;
```

9.2. Fps

This read only parameter shows the current DMA framerate. It measures the number of frames which are output in one second. Only integer values i.e. completed frames are considered.

Table 9.2. Parameter properties of Fps

Property	Value
Name	**Fps**
Display Name	**Fps**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum** 0 **Maximum** 125000000 **Stepsize** 1

Example 9.2. Usage of Fps

```
/* Get */ value_ = Fps;
```

Chapter 10. RAM Test

10.1. NumberOfRams

Number of logic RAM modules the applet is using. The frame grabber might allow more but the applet might not use all of them.

Table 10.1. Parameter properties of NumberOfRams

Property	Value
Name	**NumberOfRams**
Display Name	**Number of RAMs**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 1** **Maximum 4** **Stepsize 1**
Unit of measure	

Example 10.1. Usage of NumberOfRams

```
/* Get */ value_ = NumberOfRams;
```

10.2. RamSize

Size of one RAM module. Unit is Mebibyte i.e. Byte times 2^20.

Table 10.2. Parameter properties of RamSize

Property	Value
Name	**RamSize**
Display Name	**RAM Size**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 1** **Maximum 8192** **Stepsize 1**
Unit of measure	**MiB**

Example 10.2. Usage of RamSize

```
/* Get */ value_ = RamSize;
```

10.3. ErrorOccurred

Is set if an error in any of the RAM modules is detected. This value should always be at FG_NO.

Table 10.3. Parameter properties of ErrorOccurred

Property	Value
Name	**ErrorOccurred**
Display Name	**Erorr Occured**
Interface	**IEnumeration**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Yes** Yes **No** No

Example 10.3. Usage of ErrorOccurred

```
/* Get */ value_ = ErrorOccurred;
```

10.4. RamBandwidth

Shows the throughput of the DRAM in MB/s. (10^6 byte). Ensure to not block the DRAM speed by the DMA. You can ensure this by setting the test output (parameter *OutputSelect*) mode to DMA performance or camera output.

Table 10.4. Parameter properties of RamBandwidth

Property	Value
Name	**RamBandwidth**
Display Name	**RAM Bandwidth MBs**
Interface	**IFloat**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum** **0.0** **Maximum** **40000.0** **Stepsize** **1.0**

Example 10.4. Usage of RamBandwidth

```
/* Get */ value_ = RamBandwidth;
```

10.5. EnableRam0

You can stop the processing of data for each RAM module.

For frame grabbers with non shared memory this has no effect. However, for frame grabbers with shared memory, RAM modules can get more bandwidth if others are disabled.

Check the RAM image counter parameters *ImageCountRam0* to see if a RAM module processes data or not. If processing is enabled, but the counter value does not change, the RAM module might have a defect.

Table 10.5. Parameter properties of EnableRam0

Property	Value
Name	**EnableRam0**
Display Name	**Enable RAM 0**
Interface	**IEnumeration**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Yes** Yes **No** No
Default value	**Yes**

Example 10.5. Usage of EnableRam0

```
/* Set */ EnableRam0 = Yes;
/* Get */ value_ = EnableRam0;
```

10.6. ErrorCountRam0

This parameter shows the number of errors detected for the respecive RAM module. One error indicates that in a RAM data cell at least one bit is not equal to the expected value. The RAM data cell size corresponds to the RAM data width and can be for example 128Bit or 256Bit.

Table 10.6. Parameter properties of ErrorCountRam0

Property	Value
Name	**ErrorCountRam0**
Display Name	**Error Count RAM 0**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum** 0 **Maximum** 4294967295 **Stepsize** 1
Unit of measure	**pixel errors**

Example 10.6. Usage of ErrorCountRam0

```
/* Get */ value_ = ErrorCountRam0;
```

10.7. ImageCountRam0

This value is incremented when the RAM module has been fully written and read. If this value does not increase it might show a defect in a RAM module.

Table 10.7. Parameter properties of ImageCountRam0

Property	Value
Name	**ImageCountRam0**
Display Name	**Image Count RAM 0**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum** 0 **Maximum** 4294967295 **Stepsize** 1

Example 10.7. Usage of ImageCountRam0

```
/* Get */ value_ = ImageCountRam0;
```

10.8. InjectErrorsRam0

For self-test you can inject errors to the current processing.

Table 10.8. Parameter properties of InjectErrorsRam0

Property	Value
Name	**InjectErrorsRam0**
Display Name	**Inject Errors on RAM0**
Interface	**IEnumeration**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Yes** Yes **No** No
Default value	**No**

Example 10.8. Usage of InjectErrorsRam0

```
/* Set */ InjectErrorsRam0 = No;
/* Get */ value_ = InjectErrorsRam0;
```

Chapter 11. Gpio

11.1. FrontGpi

Parameter *FrontGpi* is used to monitor the digital inputs of the frame grabber.

You can read the current state of these inputs using parameter *FrontGpi*. Bit 0 of the read value represents input 0, bit 1 represents input 1 and so on. For example, if you obtain the value 10 or hexadecimal 0xA the frame grabber will have high level on it's digital inputs 1 and 3.

Table 11.1. Parameter properties of FrontGpi

Property	Value
Name	**FrontGpi**
Display Name	**Front GPI Status bitmask**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 0** **Maximum 3** **Stepsize 1**

Example 11.1. Usage of FrontGpi

```
/* Get */ value_ = FrontGpi;
```

11.2. FrontGpo

You can use this parameter to set the state of the front digital outputs.

Bit 0 of the read value represents output 0, bit 1 represents output 1 and so on. For example, if you set the value to 37 or hexadecimal 0x25 the frame grabber will have high level on it's digital outputs 0, 2 and 5.

Table 11.2. Parameter properties of FrontGpo

Property	Value
Name	**FrontGpo**
Display Name	**Front Output bitmask**
Interface	**IInteger**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Minimum 0** **Maximum 15** **Stepsize 1**
Default value	**15**

Example 11.2. Usage of FrontGpo

```
/* Set */ FrontGpo = 15;
/* Get */ value_ = FrontGpo;
```

Chapter 12. User LED

12.1. LedMode

The applet has several user LEDs. You can either define the state of this LEDs manual using parameter *LedPattern* or use an automatic pattern. Use this parameter to set the desired mode.

Table 12.1. Parameter properties of LedMode

Property	Value
Name	**LedMode**
Display Name	**LED Mode**
Interface	**IEnumeration**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Manual** Manual **Counter** Counter
Default value	**Manual**

Example 12.1. Usage of LedMode

```
/* Set */ LedMode = Manual;
/* Get */ value_ = LedMode;
```

12.2. LedPattern

The applet has several user LEDs. Set the state of the user LEDs using this parameter. Use a bitmask. For example, if you set the parameter to value 5, LEDs 0 and 2 will be switched on. Note that the number of user LEDs depends on the frame grabber used.

Table 12.2. Parameter properties of LedPattern

Property	Value
Name	**LedPattern**
Display Name	**LED pattern bitmask**
Interface	**IInteger**
Access policy	**Read/Write/Change**
Visibility	**Beginner**
Allowed values	**Minimum** 0 **Maximum** 255 **Stepsize** 1
Default value	**0**

Example 12.2. Usage of LedPattern

```
/* Set */ LedPattern = 0;
/* Get */ value_ = LedPattern;
```

Chapter 13. Miscellaneous

This category summarizes other read and write parameters such as the camera status, buffer fill levels, DMA transfer lengths, and time stamps.

13.1. Version

FIXME_CategoryDocumentation_Missing_Miscellaneous::Version

13.1.1. AppletVersion

This parameter indicates the version number of the applet. Report this value when contacting the Basler support.

Table 13.1. Parameter properties of AppletVersion

Property	Value
Name	**AppletVersion**
Display Name	**Applet Version**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 0** **Maximum 256** **Stepsize 1**

Example 13.1. Usage of AppletVersion

```
/* Get */ value_ = AppletVersion;
```

13.1.2. AppletRevision

This parameter indicates the revision number of the applet. Report this value when contacting the Basler support.

Table 13.2. Parameter properties of AppletRevision

Property	Value
Name	**AppletRevision**
Display Name	**Applet Revision**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Beginner**
Allowed values	**Minimum 0** **Maximum 256** **Stepsize 1**

Example 13.2. Usage of AppletRevision

```
/* Get */ value_ = AppletRevision;
```

13.1.3. VisualAppletsBuildVersion

Returns the VisualApplets version used to build the applets.

Table 13.3. Parameter properties of VisualAppletsBuildVersion

Property	Value
Name	**VisualAppletsBuildVersion**
Display Name	**Visual Applets Build Version**
Interface	**IString**
Access policy	**Read-Only**
Visibility	**Beginner**

Example 13.3. Usage of VisualAppletsBuildVersion

```
/* Get */ value_ = VisualAppletsBuildVersion;
```

Chapter 14. Boardstatus

14.1. SystemmonitorMappedToFgPort

Indicates the frame grabber port mapping. Range: between 0 and 3.

Table 14.1. Parameter properties of SystemmonitorMappedToFgPort

Property	Value
Name	**SystemmonitorMappedToFgPort**
Display Name	**Systemmonitor Mapped to Fg Port**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 14.1. Usage of SystemmonitorMappedToFgPort

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorMappedToFgPortSelector = i;
    value_ = SystemmonitorMappedToFgPort;
}
```

14.2. SystemmonitorCurrentLinkSpeed

Returns the current link width of the frame grabber representing the number of PCIe lanes that are used for data transfer. This is a value that should correspond to the number of hardware lanes the frame grabber is requiring, otherwise the possible maximum of DMA bandwidth can be reduced drastically.

Table 14.2. Parameter properties of SystemmonitorCurrentLinkSpeed

Property	Value
Name	**SystemmonitorCurrentLinkSpeed**
Display Name	**Systemmonitor Current Link Speed**
Interface	**IFloat**
Access policy	**Read-Only**
Visibility	**Expert**
Allowed values	**Minimum 0.0** **Maximum 1000.0** **Stepsize 0.5**
Unit of measure	**Gb/s**

Example 14.2. Usage of SystemmonitorCurrentLinkSpeed

```
/* Get */ value_ = SystemmonitorCurrentLinkSpeed;
```

14.3. SystemmonitorPcieTrainedPayloadSize

Returns the PCIe packet size that was evaluated during the training period at boot-time.

Table 14.3. Parameter properties of SystemmonitorPcieTrainedPayloadSize

Property	Value
Name	**SystemmonitorPcieTrainedPayloadSize**
Display Name	**Systemmonitor PCIe Trained Payload Size**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Expert**
Allowed values	**Minimum 0** **Maximum 1024** **Stepsize 1**
Unit of measure	**byte**

Example 14.3. Usage of SystemmonitorPcieTrainedPayloadSize

```
/* Get */ value_ = SystemmonitorPcieTrainedPayloadSize;
```

14.4. SystemmonitorPcieTrainedRequestSize

Returns the size (in bytes) of the PCIe packets payload that are used for the data transmission between the frame grabber and the PCIe bridge.

Table 14.4. Parameter properties of SystemmonitorPcieTrainedRequestSize

Property	Value
Name	**SystemmonitorPcieTrainedRequestSize**
Display Name	**Systemmonitor PCIe Trained Request Size**
Interface	**IInteger**
Access policy	**Read-Only**
Visibility	**Expert**
Allowed values	**Minimum 0** **Maximum 4096** **Stepsize 1**
Unit of measure	**byte**

Example 14.4. Usage of SystemmonitorPcieTrainedRequestSize

```
/* Get */ value_ = SystemmonitorPcieTrainedRequestSize;
```

14.5. CxpInputMappedToFWPortPort

This parameter returns the firmware CXP channel, which is currently monitored by the module. There is not necessarily a one-by-one mapping between firmware port (i.e. the camera port resource) and frame grabber port (i.e. the physical connector). Instead, the mapping can be any permutation. The software discovery process reorders the channels and ports to achieve correct virtual interconnect. Range: 0 to 3 (2 bit).

Table 14.5. Parameter properties of CxpInputMappedToFWPortPort

Property	Value
Name	**CxpInputMappedToFWPortPort**
Display Name	**CXP Input Mapped to Firmware Port Port**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 14.5. Usage of CxpInputMappedToFWPortPort

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpInputMappedToFWPortPortSelector = i;
    value_ = CxpInputMappedToFWPortPort;
}
```

Chapter 15. Errors

15.1. SystemmonitorDecoder8b10bError

Link stability counter. It is incremented when the number of measured symbols received by the channel transceiver are not in 8b10b encoding or/and have wrong disparity. Range: 0 to (2^48 - 1) (48 bit).

Table 15.1. Parameter properties of SystemmonitorDecoder8b10bError

Property	Value
Name	**SystemmonitorDecoder8b10bError**
Display Name	**Systemmonitor Decoder 8b10b Error**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.1. Usage of SystemmonitorDecoder8b10bError

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorDecoder8b10bErrorSelector = i;
    value_ = SystemmonitorDecoder8b10bError;
}
```

15.2. SystemmonitorByteAlignment8b10bLocked

Monitors whether the clock recovery has worked and valid 8b/10b signals are recognized.

Table 15.2. Parameter properties of SystemmonitorByteAlignment8b10bLocked

Property	Value
Name	**SystemmonitorByteAlignment8b10bLocked**
Display Name	**Systemmonitor Byte Alignment 8B 10 B Locked**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.2. Usage of SystemmonitorByteAlignment8b10bLocked

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorByteAlignment8b10bLockedSelector = i;
    value_ = SystemmonitorByteAlignment8b10bLocked;
}
```

15.3. SystemmonitorRxStreamIncompleteCount

Returns the number of received incomplete stream counts. Range: between 0 and 8191 in steps of 1.

Table 15.3. Parameter properties of SystemmonitorRxStreamIncompleteCount

Property	Value
Name	**SystemmonitorRxStreamIncompleteCount**
Display Name	**Systemmonitor Rx Stream Incomplete Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.3. Usage of SystemmonitorRxStreamIncompleteCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorRxStreamIncompleteCountSelector = i;
    value_ = SystemmonitorRxStreamIncompleteCount;
}
```

15.4. SystemmonitorRxUnknownDataReceivedCount

Returns the number of received unknown data, i.e. packets received that aren't defined in the CXP standard. Range: between 0 and 8191 in steps of 1.

Table 15.4. Parameter properties of SystemmonitorRxUnknownDataReceivedCount

Property	Value
Name	**SystemmonitorRxUnknownDataReceivedCount**
Display Name	**Systemmonitor Rx Unknown Data Received Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.4. Usage of SystemmonitorRxUnknownDataReceivedCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorRxUnknownDataReceivedCountSelector = i;
    value_ = SystemmonitorRxUnknownDataReceivedCount;
}
```

15.5. CxpOvertriggerRequestPulseCount

This parameter counts the trigger requests that were skipped, because the transmitter was still busy by sending the previous trigger packet. See CXP 2.0 standard, chapter 9.3.2. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range: 0 to 4095 (12 bit). Bit 12 indicates an overflow.

Table 15.5. Parameter properties of CxpOvertriggerRequestPulseCount

Property	Value
Name	**CxpOvertriggerRequestPulseCount**
Display Name	**CXP Overtrigger Request Pulse Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.5. Usage of CxpOvertriggerRequestPulseCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpOvertriggerRequestPulseCountSelector = i;
    value_ = CxpOvertriggerRequestPulseCount;
}
```

15.6. CxpTriggerAckMissingCount

This parameter counts the situations in which a trigger packet was sent, but no acknowledgment packet was received for it yet, which then led to a timeout (480ns for 1-6Gb/s, 240ns for 10-12.5Gb/s). See CXP 2.0 standard, chapter 9.3.2. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range: 0 to 8191 (13 bit).

Table 15.6. Parameter properties of CxpTriggerAckMissingCount

Property	Value
Name	**CxpTriggerAckMissingCount**
Display Name	**CXP Lost Trigger ACK Missing Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.6. Usage of CxpTriggerAckMissingCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpTriggerAckMissingCountSelector = i;
    value_ = CxpTriggerAckMissingCount;
}
```

15.7. CxpControlAckLostCount

This parameter counts situations in which a control packet was sent but no acknowledgment packet was received for it yet and the timeout of 200 ms is reached. See CXP 2.0 standard, chapter 9.6.1.1. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.7. Parameter properties of CxpControlAckLostCount

Property	Value
Name	**CxpControlAckLostCount**
Display Name	**CXP Control ACK Lost Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.7. Usage of CxpControlAckLostCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpControlAckLostCountSelector = i;
    value_ = CxpControlAckLostCount;
}
```

15.8. CxpControlTagErrorCount

This parameter counts situations in which an acknowledgment for a control packet was received with a tag that doesn't match the expected tag sent in the corresponding request control packet. See CXP 2.0 standard, chapter 9.6.1.2. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.8. Parameter properties of CxpControlTagErrorCount

Property	Value
Name	**CxpControlTagErrorCount**
Display Name	**CXP Control Tag Error Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.8. Usage of CxpControlTagErrorCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpControlTagErrorCountSelector = i;
    value_ = CxpControlTagErrorCount;
}
```

15.9. CxpControlAckIncompleteCount

This parameter counts situations in which an incorrectly formatted acknowledgment for a control packet was received. Incorrectly formatted means that e.g. the end of packet indicator is missing etc. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.9. Parameter properties of CxpControlAckIncompleteCount

Property	Value
Name	**CxpControlAckIncompleteCount**
Display Name	**CXP Control ACK Incomplete Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.9. Usage of CxpControlAckIncompleteCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpControlAckIncompleteCountSelector = i;
    value_ = CxpControlAckIncompleteCount;
}
```

15.10. CxpHeartbeatIncompleteCount

This parameter counts situations in which the received heart beat packet is incomplete, e.g. it misses the end of the packet indicator. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.10. Parameter properties of CxpHeartbeatIncompleteCount

Property	Value
Name	**CxpHeartbeatIncompleteCount**
Display Name	**CXP Heartbeat Incomplete Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.10. Usage of CxpHeartbeatIncompleteCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpHeartbeatIncompleteCountSelector = i;
    value_ = CxpHeartbeatIncompleteCount;
}
```

15.11. CxpHeartbeatMaxPeriodViolationCount

The heartbeat period is defined in CXP 2.0 standard as 100ms maximum, i.e. within that time at least 1 heartbeat packet must be sent by the camera. This parameter counts the situations in which heartbeat packets exceeded this timeout (100ms). Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.11. Parameter properties of CxpHeartbeatMaxPeriodViolationCount

Property	Value
Name	**CxpHeartbeatMaxPeriodViolationCount**
Display Name	**CXP Hearbeat Max Period Violation Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.11. Usage of CxpHeartbeatMaxPeriodViolationCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpHeartbeatMaxPeriodViolationCountSelector = i;
    value_ = CxpHeartbeatMaxPeriodViolationCount;
}
```

15.12. Crc

FIXME_CategoryDocumentation_Missing_Errors::CRC

15.12.1. SystemmonitorRxPacketCrcErrorCount

Returns the number of received packet CRC errors. Range: between 0 and 8191 in steps of 1.

Table 15.12. Parameter properties of SystemmonitorRxPacketCrcErrorCount

Property	Value
Name	**SystemmonitorRxPacketCrcErrorCount**
Display Name	**Systemmonitor Rx Packet CRC Error Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.12. Usage of SystemmonitorRxPacketCrcErrorCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorRxPacketCrcErrorCountSelector = i;
    value_ = SystemmonitorRxPacketCrcErrorCount;
}
```

15.12.2. CxpStreamPacketCrcError

This parameter returns information whether there were CRC errors in received stream packets. Range 0 (NO) to 1 (YES).

Table 15.13. Parameter properties of CxpStreamPacketCrcError

Property	Value
Name	**CxpStreamPacketCrcError**
Display Name	**CXP Stream Packet CRC Error**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.13. Usage of CxpStreamPacketCrcError

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpStreamPacketCrcErrorSelector = i;
    value_ = CxpStreamPacketCrcError;
}
```

15.12.3. CxpControlAckPacketCrcError

This parameter returns information whether there were CRC errors in received control acknowledgement packets. Range 0 (NO) to 1 (YES).

Table 15.14. Parameter properties of CxpControlAckPacketCrcError

Property	Value
Name	**CxpControlAckPacketCrcError**
Display Name	**CXP Control ACK Packet CRC Error**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.14. Usage of CxpControlAckPacketCrcError

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpControlAckPacketCrcErrorSelector = i;
    value_ = CxpControlAckPacketCrcError;
}
```

15.13. LengthErrors

FIXME_CategoryDocumentation_Missing_Errors::LengthErrors

15.13.1. SystemmonitorRxLengthErrorCount

This parmeter counts how often the length of a CXP packet doesn't correspond to what is specified in the header and returns the number of length errors. Range: between 0 and 8191 in steps of 1.

Table 15.15. Parameter properties of SystemmonitorRxLengthErrorCount

Property	Value
Name	**SystemmonitorRxLengthErrorCount**
Display Name	**Systemmonitor Rx Length Error Count**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.15. Usage of SystemmonitorRxLengthErrorCount

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorRxLengthErrorCountSelector = i;
    value_ = SystemmonitorRxLengthErrorCount;
}
```

15.13.2. CxpStreamPacketLengthError

This parameter returns information whether a length error in the stream packets was detected. Range: 0 (NO) to 1 (YES).

Table 15.16. Parameter properties of CxpStreamPacketLengthError

Property	Value
Name	**CxpStreamPacketLengthError**
Display Name	**CXP Stream Packet Length Error**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.16. Usage of CxpStreamPacketLengthError

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpStreamPacketLengthErrorSelector = i;
    value_ = CxpStreamPacketLengthError;
}
```

15.14. ReceivedPacketsCorrected

FIXME_CategoryDocumentation_Missing_Errors::ReceivedPacketsCorrected

15.14.1. CxpErrorCorrected

This parameter counts errors received in packet headers and trailers that were corrected. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.17. Parameter properties of CxpErrorCorrected

Property	Value
Name	**CxpErrorCorrected**
Display Name	**CXP Error Corrected**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.17. Usage of CxpErrorCorrected

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedSelector = i;
    value_ = CxpErrorCorrected;
}
```

15.14.2. CxpErrorCorrectedTrigger

This parameter returns the information whether errors were corrected in received trigger packets. Range 0 (NO) to 1 (YES).

Table 15.18. Parameter properties of CxpErrorCorrectedTrigger

Property	Value
Name	**CxpErrorCorrectedTrigger**
Display Name	**CXP Error Corrected Trigger**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.18. Usage of CxpErrorCorrectedTrigger

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedTriggerSelector = i;
    value_ = CxpErrorCorrectedTrigger;
}
```

15.14.3. CxpErrorCorrectedTriggerAck

This parameter returns the information whether errors were corrected in received trigger acknowledge packets. Range 0 (NO) to 1 (YES).

Table 15.19. Parameter properties of CxpErrorCorrectedTriggerAck

Property	Value
Name	**CxpErrorCorrectedTriggerAck**
Display Name	**CXP Error Corrected Trigger ACK**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.19. Usage of CxpErrorCorrectedTriggerAck

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedTriggerAckSelector = i;
    value_ = CxpErrorCorrectedTriggerAck;
}
```

15.14.4. CxpErrorCorrectedStream

This parameter returns the information whether errors were corrected in received stream packets. Range 0 (NO) to 1 (YES).

Table 15.20. Parameter properties of CxpErrorCorrectedStream

Property	Value
Name	**CxpErrorCorrectedStream**
Display Name	**CXP Error Corrected Stream**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.20. Usage of CxpErrorCorrectedStream

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedStreamSelector = i;
    value_ = CxpErrorCorrectedStream;
}
```

15.14.5. CxpErrorCorrectedControlAck

This parameter returns the information whether errors were corrected in received stream acknowledge packets. Range 0 (NO) to 1 (YES).

Table 15.21. Parameter properties of CxpErrorCorrectedControlAck

Property	Value
Name	**CxpErrorCorrectedControlAck**
Display Name	**CXP Error Corrected Control ACK**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.21. Usage of CxpErrorCorrectedControlAck

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedControlAckSelector = i;
    value_ = CxpErrorCorrectedControlAck;
}
```

15.14.6. CxpErrorCorrectedLinkTest

This parameter returns the information whether errors were corrected in received link test packets. Range 0 (NO) to 1 (YES).

Table 15.22. Parameter properties of CxpErrorCorrectedLinkTest

Property	Value
Name	**CxpErrorCorrectedLinkTest**
Display Name	**CXP Error Corrected Link Test**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.22. Usage of CxpErrorCorrectedLinkTest

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedLinkTestSelector = i;
    value_ = CxpErrorCorrectedLinkTest;
}
```

15.14.7. CxpErrorCorrectedHeartbeat

This parameter returns the information whether errors were corrected in received heartbeat packets. Range 0 (NO) to 1 (YES).

Table 15.23. Parameter properties of CxpErrorCorrectedHeartbeat

Property	Value
Name	**CxpErrorCorrectedHeartbeat**
Display Name	**CXP Error Corrected Heartbeat**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.23. Usage of CxpErrorCorrectedHeartbeat

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorCorrectedHeartbeatSelector = i;
    value_ = CxpErrorCorrectedHeartbeat;
}
```

15.15. ReceivedPacketsUncorrected

FIXME_CategoryDocumentation_Missing_Errors::ReceivedPacketsUncorrected

15.15.1. CxpErrorUncorrected

This parameter counts errors received in packet headers and trailers that haven't been corrected. Bits [11:0] count the amount of violations. Bit [12] is set when a counter overflow occurs. Range 0 to 8191 (13 bit).

Table 15.24. Parameter properties of CxpErrorUncorrected

Property	Value
Name	**CxpErrorUncorrected**
Display Name	**CXP Error Uncorrected**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.24. Usage of CxpErrorUncorrected

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedSelector = i;
    value_ = CxpErrorUncorrected;
}
```

15.15.2. CxpErrorUncorrectedTrigger

This parameter returns the information whether there were errors in received trigger packets that haven't been corrected. Range 0 (NO) to 1 (YES).

Table 15.25. Parameter properties of CxpErrorUncorrectedTrigger

Property	Value
Name	**CxpErrorUncorrectedTrigger**
Display Name	**CXP Error Uncorrected Trigger**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.25. Usage of CxpErrorUncorrectedTrigger

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedTriggerSelector = i;
    value_ = CxpErrorUncorrectedTrigger;
}
```

15.15.3. CxpErrorUncorrectedTriggerAck

This parameter returns the information whether there were errors in received trigger acknowledgement packets that haven't been corrected. Range 0 (NO) to 1 (YES).

Table 15.26. Parameter properties of CxpErrorUncorrectedTriggerAck

Property	Value
Name	**CxpErrorUncorrectedTriggerAck**
Display Name	**CXP Error Uncorrected Trigger ACK**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.26. Usage of CxpErrorUncorrectedTriggerAck

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedTriggerAckSelector = i;
    value_ = CxpErrorUncorrectedTriggerAck;
}
```

15.15.4. CxpErrorUncorrectedStream

This parameter returns the information whether there were errors in received stream packets that haven't been corrected. Range 0 (NO) to 1 (YES).

Table 15.27. Parameter properties of CxpErrorUncorrectedStream

Property	Value
Name	**CxpErrorUncorrectedStream**
Display Name	**CXP Error Uncorrected Stream**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.27. Usage of CxpErrorUncorrectedStream

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedStreamSelector = i;
    value_ = CxpErrorUncorrectedStream;
}
```

15.15.5. CxpErrorUncorrectedControlAck

This parameter returns information whether there were errors in received control acknowledgement packets that haven't been corrected. Range 0 (NO) to 1 (YES).

Table 15.28. Parameter properties of CxpErrorUncorrectedControlAck

Property	Value
Name	**CxpErrorUncorrectedControlAck**
Display Name	**CXP Error Uncorrected Control ACK**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.28. Usage of CxpErrorUncorrectedControlAck

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedControlAckSelector = i;
    value_ = CxpErrorUncorrectedControlAck;
}
```

15.15.6. CxpErrorUncorrectedLinkTest

This parameter returns information whether there were errors in received link test packets that haven't been corrected. Range 0 (NO) to 1 (YES).

Table 15.29. Parameter properties of CxpErrorUncorrectedLinkTest

Property	Value
Name	**CxpErrorUncorrectedLinkTest**
Display Name	**CXP Error Uncorrected Link Test**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.29. Usage of CxpErrorUncorrectedLinkTest

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedLinkTestSelector = i;
    value_ = CxpErrorUncorrectedLinkTest;
}
```

15.15.7. CxpErrorUncorrectedHeartbeat

This parameter returns information whether there were errors in received heartbeat packets that haven't been corrected. Range 0 (NO) to 1 (YES).

Table 15.30. Parameter properties of CxpErrorUncorrectedHeartbeat

Property	Value
Name	**CxpErrorUncorrectedHeartbeat**
Display Name	**CXP Error Uncorrected Heartbeat**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.30. Usage of CxpErrorUncorrectedHeartbeat

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpErrorUncorrectedHeartbeatSelector = i;
    value_ = CxpErrorUncorrectedHeartbeat;
}
```

15.16. UnsupportedPackets

FIXME_CategoryDocumentation_Missing_Errors::UnsupportedPackets

15.16.1. SystemmonitorRxUnsupportedPacketUnit

This parameter returns the number of received unsupported packets. Range: between 0 and 8191 in steps of 1.

Table 15.31. Parameter properties of SystemmonitorRxUnsupportedPacketUnit

Property	Value
Name	**SystemmonitorRxUnsupportedPacketUnit**
Display Name	**Systemmonitor Rx Unsupported Packet Unit**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.31. Usage of SystemmonitorRxUnsupportedPacketUnit

```
/* Get */ for (i = 0; i < 2; ++i)
{
    SystemmonitorRxUnsupportedPacketUnitSelector = i;
    value_ = SystemmonitorRxUnsupportedPacketUnit;
}
```

15.16.2. CxpUnsupportedGpioReceived

This parameter returns information whether a GPIO packet was received while using a CXP standard higher than 1.0. Range: 0 (NO) to 1 (YES).

Table 15.32. Parameter properties of CxpUnsupportedGpioReceived

Property	Value
Name	**CxpUnsupportedGpioReceived**
Display Name	**CXP Unsupported GPIO Received**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.32. Usage of CxpUnsupportedGpioReceived

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpUnsupportedGpioReceivedSelector = i;
    value_ = CxpUnsupportedGpioReceived;
}
```

15.16.3. CxpUnsupportedEventReceived

This parameter returns information whether an event packet was received while using a CXP standard less than 2.0. Range: 0 (NO) to 1 (YES).

Table 15.33. Parameter properties of CxpUnsupportedEventReceived

Property	Value
Name	**CxpUnsupportedEventReceived**
Display Name	**CXP Unsupported Event Received**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.33. Usage of CxpUnsupportedEventReceived

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpUnsupportedEventReceivedSelector = i;
    value_ = CxpUnsupportedEventReceived;
}
```

15.16.4. CxpUnsupportedHeartbeatReceived

This parameter returns information whether a heartbeat packet was received while using a CXP standard less than 2.0. Range: 0 (NO) to 1 (YES).

Table 15.34. Parameter properties of CxpUnsupportedHeartbeatReceived

Property	Value
Name	**CxpUnsupportedHeartbeatReceived**
Display Name	**CXP Unsupported Hearbeat Received**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.34. Usage of CxpUnsupportedHeartbeatReceived

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpUnsupportedHeartbeatReceivedSelector = i;
    value_ = CxpUnsupportedHeartbeatReceived;
}
```

15.16.5. CxpUnsupportedGpioAckReceived

This parameter returns information whether a GPIO acknowledgment was received while using a CXP standard higher than 1.0. Range: 0 (NO) to 1 (YES).

Table 15.35. Parameter properties of CxpUnsupportedGpioAckReceived

Property	Value
Name	**CxpUnsupportedGpioAckReceived**
Display Name	**CXP Unsupported GPIO ACK Received**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.35. Usage of CxpUnsupportedGpioAckReceived

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpUnsupportedGpioAckReceivedSelector = i;
    value_ = CxpUnsupportedGpioAckReceived;
}
```

15.16.6. CxpUnsupportedGpioRequestReceived

This parameter returns information whether a GPIO request from VisualApplets was received while using a CXP standard higher than 1.0. Range: 0 (NO) to 1 (YES).

Table 15.36. Parameter properties of CxpUnsupportedGpioRequestReceived

Property	Value
Name	**CxpUnsupportedGpioRequestReceived**
Display Name	**CXP Unsupported GPIO Request Received**
Interface	**IInteger (Field)**
Field Size	**2**
Access policy	**Read-Only**
Visibility	**Expert**

Example 15.36. Usage of CxpUnsupportedGpioRequestReceived

```
/* Get */ for (i = 0; i < 2; ++i)
{
    CxpUnsupportedGpioRequestReceivedSelector = i;
    value_ = CxpUnsupportedGpioRequestReceived;
}
```

Glossary

Area of Interest (AOI)	See Region of Interest.
Board	A Basler hardware. Usually, a board is represented by a frame grabber. Boards might comprise multiple devices.
Board ID Number	An identification number of a Basler board in a PC system. The number is not fixed to a specific hardware but has to be unique in a PC system.
Camera Index	The index of a camera connected to a frame grabber. The first camera will have index zero. Mind the difference between the camera index and the frame grabber camera port. See also Camera Port.
Camera Port	The Basler frame grabber connectors for cameras are called camera ports. They are numbered {0, 1, 2, ...} or enumerated {A, B, C, ... }. Depending on the interface one camera could be connected to multiple camera ports. Also, multiple cameras could be connected to one camera port.
Camera Tap	See Tap.
Device	A board can consist of multiple devices. Devices are numbered. The first device usually has number one.
Direct Memory Access (DMA)	A DMA transfer allows hardware subsystems within the computer to access the system memory independently of the central processing unit (CPU). Basler uses DMAs for data transfer such as image data between a board e.g. a frame grabber and a PC. Data transfers can be established in multiple directions i.e. from a frame grabber to the PC (download) and from the PC to a frame grabber (upload). Multiple DMA channels may exist for one board. Control and configuration data usually do not use DMA channels.
DMA Channel	See DMA Index.
DMA Index	The index of a DMA transfer channel. See also Direct Memory Access.
Event	In programming or runtime environments, a callback function is a piece of executable code that is passed as an argument, which is expected to call back (execute) exactly that time an event is triggered. These events are not related to a special camera functionality and based on frame grabber internal functionality. Basler uses hardware interrupts for the event transfer and processing is absolutely optimized for low latency. These interrupts are only produced by the frame grabber if an event is registered and activated by software. If an event is fired at a very high frequency this may influence the system performance. For example these events can be used to check the reliability between a frame trigger input and the resulting and expected camera frame. The Basler Framegrabber SDK enables an application to get these event notifications about certain state changes at the data flow from camera to RAM and the image and trigger processing as well. Please consult the Basler Framegrabber SDK documentation for more details concerning the implementation of this functionality. Some events are enabled to produce additional data, which is described for the event itself.

Frame Grabber	Usually a PC hardware using PCI express to interface the camera and grab camera images. The frame grabber will grab, buffer, pre-process and forward the images to the PC memory. Moreover, the frame grabber performs the trigger signal processing to trigger the camera, external lights and controllers. On V-series frame grabber custom processing can be implemented using VisualApplets. See also Direct Memory Access, Interface Card, VisualApplets.
GenICam	Generic Interface for Cameras is a generic programming interface for machine vision (industrial) cameras.
GenTL	GenICam Transport Layer. This is the transport layer interface for enumerating cameras, grabbing images from the camera, and moving them to the user application.
Interface Card	Usually a PC hardware using PCI express to interface the camera and grab camera images. The interface card will grab, buffer and forward the images to the PC memory. Moreover, the interface card performs the trigger signal processing to trigger the camera, external lights and controllers. See also Direct Memory Access, Frame Grabber.
Port	See Camera Port.
Process	An image or signal data processing block. A process can include one or more cameras, one or more DMA channels and modules.
Region of Interest (ROI)	Represents a part of a frame. Mostly rectangular and within the original image boundaries. Defined by source coordinates and its dimension. The frame grabber cuts the region of interest from the camera image. A region of interest might reduce or increase the required bandwidth and the corresponding image dimension.
Sensor Tap	See Tap.
Software Callback	See Event.
Tap	Some cameras have multiple taps. This means, they can acquire or transfer more than one pixel at a time which increses the camera's acquisition speed. The camera sensor tap readout order varies. Some cameras read the pixels interlaced using multiple taps, while some cameras read the pixel simultaneously from different locations on the sensor. The reconstruction of the frame is called sensor readout correction. The Camera Link interface is also using multiple taps for image transfer to increase the bandwidth. These taps are independent from the sensor taps.
Trigger	In machine vision and image processing, a trigger is an event which causes an action. This can be for example the initiation of a new line or frame acquisition, the control of external hardware such as flash lights or actions by a software applications. Trigger events can be initiated by external sources, an internal frequency generator (timer) or software applications. The event itself is mostly based on a rising or falling edge of a electrical signal.
Trigger Input	A logic input of a trigger IO. The first input has index 0. Check mapping of input pins to logic inputs in the hardware documentation.
Trigger Output	A logic output of a trigger IO. The first output has index 1. Please check the mapping of output pins to logic outputs in the hardware documentation. The electrical characteristics and specification can be found related to the selected or used trigger board/connector.
Trigger Reliability	See Event.

User Interrupt	See Event.
VisualApplets	Simple programming of FPGA-based image processing devices. VisualApplets enables access to the FPGA processors in the image processing hardware, such as frame grabbers, industrial cameras and image processing devices, to implement individual image processing applications.

Index

F

G

H

I

L

M

N

O

R

S

T

U

V

W