pytelicam (Python Library)

User Guide

Version 1.1.1 (2024/12/13)

Toshiba Teli Corporation

Information contained in this document is subject to change without prior notice.

DAA01621E

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Contents

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

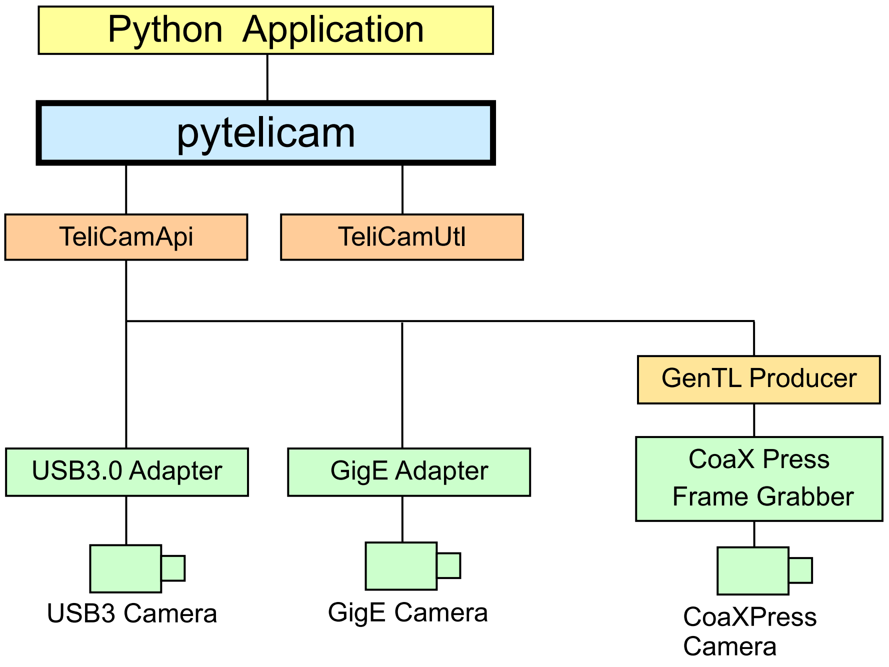
1. Introduction

pytelicam is a Python wrapper package for TeliCamSDK.
pytelicam allows you to use almost all of functions provided by TeliCamSDK in the Python language.

Software engineers who build the systems using cameras are assumed to be readers of this document.

2. Configuration

The following figure indicates software configuration of pytelicam.



Module	Description
pytelicam	Python library for designing python applications.
TeliCamApi	Function library for designing native applications.
TeliCamUtl	Utility function library for handling images.
GenTL Producer	GenTL Producer library provided by frame grabber manufacturer.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

3. Operation Environment

In order to use pytelicam, the following software must be installed.

Windows 64bit (win_amd64)	TeliCamSDK v4.0.4.1 or later
Linux 64bit (linux_x86_64) Linux ARM64 (linux_aarch64)	TeliCamSDK for Linux v4.0.4.1 or later

To install TeliCamSDK, please refer to "TeliCamSDK Start-up Guide" of TeliCamSDK.

4. Installation

pytelicam is distributed as a Python wheel (.whl) package. Please download and install the necessary pytelicam package from Toshiba Teli website.

The wheel file name is defined as follows:
{distribution}-{version}-{python tag}-{abi tag}-{platform tag}.whl

{distribution} is ”pytelicam”.
{version} is the version of pytelicam.
The relationship between the Python versions and {python tag}-{abi tag} is as follows:
For example: In case Python 3.9.x
- cp39-cp39

{platform tag} is the platform as follows.

win_amd64	:	Windows 64bit
linux_x86_64	:	Linux 64bit

To install pytelicam, use pip command.
For example: In case Python 3.9.x

```
pip install pytelicam-1.1.1-cp39-cp39-win_amd64.whl
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

5. Uninstallation

To uninstall pytelicam, use pip command.
For example:

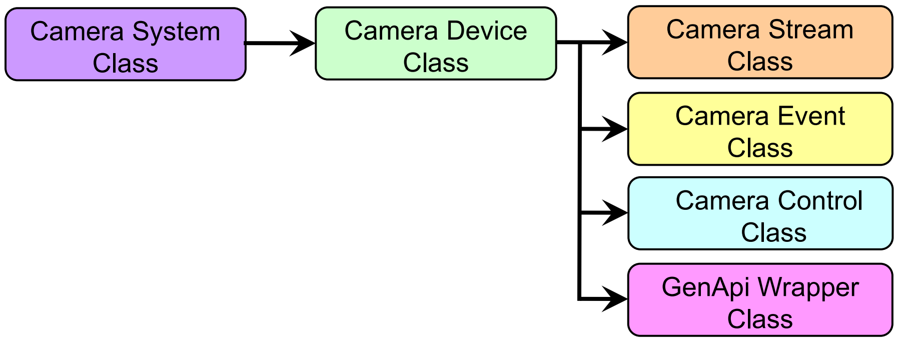
```
pip uninstall pytelicam
```

6. Programmer's Guide

pytelicam is a library that wraps TeliCamAPI function library. Refer to section "4.1 Usage" in "TeliCamAPI Library Manual Eng.pdf" about basic usage of APIs.

6.1. Class diagram

The following diagram shows class hierarchy of pytelicam.



Class	Description
Camera System Class	Class for controlling pytelicam system itself.
Camera Device Class	Class for controlling a camera. This class contains CameraStream class, camera event class, and GenApiWrapper class objects as its members.
Camera Stream Class	Class for controlling an image stream interface.
Camera Event Class	Class for notifying events of the camera.
Camera Control Class	Class for controlling individual features of the camera.
GenApi Wrapper Class	Class for controlling individual features of the camera using GenICam GenApi module.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

6.2. General process flow

1. Imports pytelicam

```
import pytelicam
```

2. Initializes pytelicam and get the object of CameraSystem class.

```
cam_system = pytelicam.get_camera_system()
```

3. Enumerates the connected cameras.

```
cam_num = cam_system.get_num_of_cameras()
```

4. Gets the CameraDevice object of the camera.

```
cam_device = cam_system.create_device_object()
```

5. Opens the camera.

```
cam_device.open()
```

6. Opens a stream of the camera.

```
cam_device.cam_stream.open()
```

7. Starts the stream.

```
cam_device.cam_stream.start()
```

8. Gets an ImageData object that stores image data from the camera.

   There are several ways to get an ImageData object. The following code gets the next arriving image.

```
image_data = cam_device.cam_stream.get_next_image()
```

9. Gets the image data as a NumPy array in BGR24 format.

```
np_arr = image_data.get_ndarray(pytelicam.OutputImageType.Bgr24)
```

   Release the ImageData object immediately after using it.
   If you repeatedly get images without releasing ImageData objects, there will be no unused ImageData object in the stream ring buffer inside this API, and you will not be able to get images.

```
image_data.release()
```

10. Stops the stream.

```
cam_device.cam_stream.stop()
```

11. Closes the stream.

```
cam_device.cam_stream.close()
```

12. Closes the camera.

```
cam_device.close()
```

13. Executes the termination process of the CameraSystem object and terminate the application.

```
cam_system.terminate()
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

6.3. Error handling

pytelicam has two types of functions that report errors.

The first type is a function that reports an error with the return value of the status code.
Even if the status code is not Success, there may be no error for pytelicam.
For example, when the camera is changed ROI using pytelicam during image transfer, the camera returns a NotWritable error. pytelicam returns pytelicam.CamApiStatus.NotWritable as the status code, but pytelicam has completed processing without error. The application is expected to check the status code and execute the branch processing.

Another type is a function that raises an exception and reports the error.
Errors that do not occur when the camera and API are operating normally are output as exception.
When an error occurs, this API will raise pytelicam.PytelicamError.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7. Classes reference

The following table shows classes that pytelicam provides.

[Classes]

	Name	Description
	CameraSystem	The root system class of pytelicam.
	CameraDevice	Class for controlling a camera.
	CameraStream	Class for controlling a stream (image).
	CameraEvent	Class for controlling camera event of a camera.
	CameraControl	Class for controlling features of a camera.
	GenApiWrapper	Class for controlling features of a camera using GenICam GenApi module.
	SignalHandle	Class for managing a signal.
	CamSystemInfo	Class for providing system information for pytelicam.
	CameraInfo	Class for providing camera information.
	ImageData	Class for managing image data.
	EventData	Class for managing camera event data.
	PytelicamError	An exception class that is raised when an error occurs in this API.

[Remarks]

Classes, functions, and properties that are not described in this manual are for use inside pytelicam only.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.1. Basic functions

7.1.1. get_camera_system

This function initializes pytelicam and get the object of CameraSystem class.

[Syntax]

pytelicam.get_camera_system(camera_type)

[Parameters]

Parameters	Description
camera_type (int)	Interface type of target cameras. Specifies type of pytelicam.CameraType as type of int. If no argument is given, “int(pytelicam.CameraType.All)” is specified. ※ GenTL interface is not included.

[Returns]

A CameraSystem object

[Return type]

pytelicam.CameraSystem

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

User application must call this function once, before calling any other functions in pytelicam.

To use all interfaces using the device drivers provided by Toshiba Teli, do the following:
(GenTL interface is not included.)

```
cam_system = pytelicam.get_camera_system()
```

To use only a specific interface, do the following:

```
cam_system = pytelicam.get_camera_system(int(pytelicam.CameraType.U3v))
```

To use multiple interfaces, do the following:

```
cam_system = pytelicam.get_camera_system( \
    int(pytelicam.CameraType.U3v) | \
    int(pytelicam.CameraType.Gev) | \
    int(pytelicam.CameraType.GenTL))
```

Excluding unused interface types may make processing time of functions shorter.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2. CameraSystem class (Root class)

This class is the root system class of pytelicam.

[Syntax]

pytelicam.CameraSystem

[Functions]

	Name	Description
◆	terminate	Executes the termination processing of the CameraSystem object.
◆	get_information	Gets system information of pytelicam.
◆	get_num_of_cameras	Creates list of connected cameras in TeliCamAPI and returns number of detected cameras.
◆	get_camera_information	Gets camera information.
◆	create_device_object	Creates and get a CameraDevice object for the camera with the specified camera index.
◆	create_device_object_from_info	Creates and get a CameraDevice object of the camera that has the information specified in Parameters.
◆	create_device_object_from_ip_address	Creates and get a CameraDevice object of the camera that is assigned the specified IP address.
◆	create_signal	Creates a signal object to detect a signal.
◆	close_signal	Closes a signal object.
◆	wait_for_signal	Checks the current state of the specified signal object.
◆	set_signal	Sets the specified signal object to the signaled state.
◆	reset_signal	Sets the specified signal object to the nonsignaled state.
◆	register_cti_file	Registers the GenTL Producer(cti file) to be used.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.1. terminate

This function executes the termination processing of the CameraSystem object.

[Syntax]

pytelicam.CameraSystem.terminate(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

User application must call this function once before destroying the CameraSystem object.

7.2.2. get_information

This function gets system information of pytelicam.

[Syntax]

pytelicam.CameraSystem.get_information(self)

[Returns]

system information

[Return_type]

pytelicam.CamSystemInfo

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.3. get_num_of_cameras

This function creates list of connected cameras in TeliCamAPI and returns number of detected cameras.

[Syntax]

pytelicam.CameraSystem.get_num_of_cameras(self)

[Returns]

The number of detected cameras.

[Return type]

int

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

User application cannot get CameraDevice object using create_device_object() or create_device_object_from_info() function until this function is called, because camera list inside this API is not ready.

If the connection status of cameras changes, this function should be executed again.

7.2.4. get_camera_information

This function gets camera information.

[Syntax]

pytelicam.CameraSystem.get_camera_information(self, camera_index)

[Parameters]

Parameters	Description
camera_index (int)	Index of camera. Index value should be up to number of detected cameras minus 1, from 0.

[Returns]

Camera information.

[Return type]

pytelicam.CameraInfo

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.5. create_device_object

This function creates and gets a CameraDevice object for the camera with the specified camera index.

[Syntax]

pytelicam.CameraSystem.create_device_object(self, camera_index=0)

[Parameters]

Parameters	Description
camera_index (int)	Index of camera. Index value should be up to number of detected cameras minus 1, from 0.

[Returns]

A CameraDevice object

[Return type]

pytelicam.CameraDevice

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.2.6. create_device_object_from_info

This function creates and gets a CameraDevice object of the camera that has the information specified in Parameters.

[Syntax]

```
pytelicam.CameraSystem.create_device_object_from_info(
                                                      self,
                                                      serial_no,
                                                      model_name,
                                                      user_defined_name)
```

[Parameters]

Parameters	Description
serial_no (str)	Serial number of the camera.
model_name (str)	Model name of the camera.
user_defined_name (str)	User defined name of the camera.

[Returns]

A CameraDevice object

[Return type]

pytelicam.CameraDevice

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Set to '' for the arguments that are not used to specify the camera.

```
cam_device = cam_system.create_device_object_from_info('0210001', 'BU406MC', '')
```

If multiple cameras have specified parameters, this function returns a camera detected earliest.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.7. create_device_object_from_ip_address

This function creates and gets a CameraDevice object of the camera by specified IP address that is already applied to a GigE camera that is existing on your network environment.

[Syntax]

```
pytelicam.CameraSystem.create_device_object_from_ip_address(
                                                        self,
                                                        ip_string)
```

[Parameters]

Parameters	Description
ip_string (str)	IPv4 address of the camera. IPv4 address is specified as a string. For example, specify '192.168.0.16'.

[Returns]

A CameraDevice object

[Return type]

pytelicam.CameraDevice

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function is available only for GigE cameras.

A sample code is as follows:

```
cam_device = cam_system.create_device_object_from_ip_address('192.168.0.16')
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.8. create_signal

This function creates a signal object to detect a signal.
A signal object is used to detect a signal such as the completion of image acquisition from the camera.

[Syntax]

pytelicam.CameraSystem.create_signal(self)

[Returns]

A signal object

[Return type]

pytelicam.SignalHandle

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.2.9. close_signal

This function closes a signal object.

[Syntax]

pytelicam.CameraSystem.close_signal(self, signal_object)

[Parameters]

Parameters	Description
signal_object (pytelicam.SignalHandle)	A signal object.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.10. wait_for_signal

This function checks the current state of the specified signal object.

If the signal object is signaled, it returns immediately. At this time, the signal object will return to the nonsignaled state.

If the signal object is nonsignaled, the calling thread enters the wait state until the signal object is signaled or the time-out interval elapses.

[Syntax]

pytelicam.CameraSystem.wait_for_signal(self, signal_object, milliseconds=5000)

[Parameters]

Parameters	Description
signal_object (pytelicam.SignalHandle)	A signal object.
milliseconds (int)	The time-out interval, in milliseconds. If this value is nonzero value is specified, this function waits until the specified signal object is signaled or the interval elapses. If this value is zero, this function does not enter a wait state if the specified signal object is not signaled. (It always returns immediately.) If this value is -1 or 0xFFFFFFFF, this function will return only when the specified object is signaled.

[Returns]

A status code indicating the result of the operation.

The return value is as follows:

CamApiStatus.Success :	The state of the specified signal object is signaled.
CamApiStatus.Timeout :	The time-out interval elapsed, and the state of the specified signal object is nonsignaled.
CamApiStatus.Unsuccessful :	The function has failed.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.11. set_signal

This function sets the specified signal object to the signaled state.

[Syntax]

pytelicam.CameraSystem.set_signal(self, signal_object)

[Parameters]

Parameters	Description
signal_object (pytelicam.SignalHandle)	A signal object.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.2.12. reset_signal

This function sets the specified signal object to the nonsignaled state.

[Syntax]

pytelicam.CameraSystem.reset_signal(self, signal_object)

[Parameters]

Parameters	Description
signal_object (pytelicam.SignalHandle)	A signal object.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.2.13. register_cti_file

This function registers the GenTL Producer(cti file) to be used.

Only valid when the GenTL interface is specified in pytelicam.get_camera_system() function.

Only registered GenTL Producers are loaded, which speeds up the camera enumeration process.

Also, you can exclude specific GenTL Producers that have problems.

[Syntax]

pytelicam.CameraSystem.register_cti_file(self, file_name)

[Parameters]

Parameters	Description
file_name (str)	Absolute path of the GenTL Producer file.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function must be executed immediately after executing pytelicam.get_camera_system() function. If this function is executed after the camera is opened, the processing may not be executed normally.

When this function is executed, GenTL Producer (cti files) existing in the directory specified by the environment variable GENICAM_GENTL64_PATH (GENICAM_GENTL32_PATH for 32-bit version) will not be loaded.

It is possible to register multiple GenTL Producers.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3. CameraDevice class (Camera Control)

This class is a class for controlling a camera.

Use create_device_object() or create_device_object_from_info() function of CameraSystem class to get CameraDevice class object.

[Syntax]

pytelicam.CameraDevice

[Functions]

	Name	Description
◆	get_information	Gets camera information.
◆	open	Opens the camera.
◆	close	Closes the camera.
◆	read_register	Reads data from the specified camera register.
◆	write_register	Writes data to the specified camera register.
◆	reset_port	Resets port of host adapter / host controller that the camera is connected.
◆	get_heartbeat	Gets the current Heartbeat setting of the camera.
◆	set_heartbeat	Sets the new Heartbeat setting to the camera.
◆	save_parameter	Saves camera parameters.
◆	load_parameter	Loads camera parameters.
◆	get_last_genicam_error	Gets information of GenICamError.

[Members]

	Name	Description
◆	cam_stream (pytelicam.CameraStream)	A CameraStream object for receiving images.
◆	cam_event (pytelicam.CameraEvent)	A CameraEvent object for receiving camera event.
◆	cam_control (pytelicam.CameraControl)	A CameraControl object for controlling camera features.
◆	genapi (pytelicam.GenApiWrapper)	A GenApiWrapper object for controlling camera features.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

[Properties]

	Name	Description
	is_open (bool)	Gets a value indicating whether the camera is opened or not. This property will return True when the camera is opened. Otherwise, this property will return False.
	is_support_iidc2 (bool)	Gets a value indicating whether the camera complies with IIDC2 standard or not. This property will return True when the camera complies with IIDC2 standard. Otherwise, this property will return False.
	camera_type (pytelicam.CameraType)	Gets camera interface type.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3.1. get_information

This function gets camera information.

[Syntax]

pytelicam.CameraDevice.get_camera_information(self)

[Returns]

Camera information.

[Return type]

pytelicam.CameraInfo

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.3.2. open

This function opens the camera.

On Windows, user application can open cameras used by other applications.
However, multiple applications cannot use stream interface or event interface at the same time.

[Syntax]

```
pytelicam.CameraDevice.open(
                                self,
                                removed_signal=None,
                                access_mode=pytelicam.CameraAccessMode.Control)
```

[Parameters]

Parameters	Description
removed_signal (pytelicam.SignalHandle)	A signal object for notifying the disconnection of the camera. When the disconnection of the camera is detected, this object is set to the signaled state. Set to None if you do not need to know the camera disconnect.
access_mode (pytelicam.CameraAccessMode)	Access mode (Control Channel Privilege) of the camera. This parameter is used only for GigE cameras. If there is no special reason, please set to pytelicam.CameraAccessMode.Control.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

The removed_signal is set to the signaled state when:

- Detect disconnection of communication cable
- Detect disconnection due to communication error
- Execute DeviceReset command of the camera
- Execute reset_port() function

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

For GigE cameras, a heartbeat timeout error will occur when communication timeout occures consecutively within a certain period of time.
When a heartbeat timeout error occurs, pytelicam detects disconnection.
When the camera is opened, heartbeat is enabled with the 15 second heartbeat timeout setting.
Heartbeat settings can be changed using set_heartbeat() function.

7.3.3. close

This function closes the camera.

[Syntax]

pytelicam.CameraDevice.close(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Calling this function during the other thread is using the camera may cause error in the other thread. Call this function after all threads finished using the camera.

7.3.4. read_register

This function reads data from the specified camera register.

[Syntax]

pytelicam.CameraDevice.read_register(self, address)

[Parameters]

Parameters	Description
address (int)	Address of the register to read.

[Returns]

The value read from the specified camera register.

[Return_type]

int

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3.5. write_register

This function writes data to the specified camera register.

[Syntax]

pytelicam.CameraDevice.write_register(self, address, value)

[Parameters]

Parameters	Description
address (int)	Address of the register to write.
value (int)	The value to write to the register.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.NotWritable :	The target is not writable.
CamApiStatus.NotImplemented :	The feature is not implemented in the camera.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3.6. reset_port

This function resets port of host adapter / host controller on the adapter that the camera is connected.

[Syntax]

pytelicam.CameraDevice.reset_port(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function is available only for USB3 cameras.

CameraDevice object should be opened before calling this function.

When a port reset is executed, *removed_signal* is set to the signaled state.

After this function is executed, Plug and Play service will search USB controller and cameras and attach device driver again.

It will take for a while for CameraSystem object to recognize all cameras again.

Calling get_num_of_cameras() before obtaining a CameraDevice object is recommended, to check that all connected cameras are recognized by confirming number of recognized cameras.

Most problems will be recovered by calling this function when problem occurred. But there may be problems which will not be recovered by calling this function.

7.3.7. get_heartbeat

This function gets the current Heartbeat setting of the camera.

[Syntax]

pytelicam.CameraDevice.get_heartbeat(self)

[Returns]

A tuple type data (enable, timeout)

enable :	The current Heartbeat enable state of the camera. If True, Heartbeat process is enabled, otherwise disabled.
timeout :	The current Heartbeat timeout value of the camera, in milliseconds.

[Return type]

tuple(bool, int)

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function is available only for GigE cameras.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3.8. set_heartbeat

This function sets the new Heartbeat setting to the camera.

[Syntax]

pytelicam.CameraDevice.set_heartbeat(self, enable, timeout)

[Parameters]

Parameters	Description
enable (bool)	The Heartbeat setting value to set. If True, Heartbeat process is enabled, otherwise disabled.
timeout (int)	The Heartbeat timeout value to set, in milliseconds. Minimum available value is 500 milliseconds.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function is available only for GigE cameras.
There are some cameras whose the Heartbeat enable state cannot be set to disable. For example, GiantDragon series. .

When opening the GigE camera, pytelicam will set the Heartbeat enable state of the camera to enable and set the Heartbeat timeout value to 15 sec.

It will not be necessary to change the Heartbeat settings, usually. Change the Heartbeat settings using this function when user application may be interrupted more than a few seconds, for example debugging user application.

Never change "GevHeartbeatTimeout" register or "GevGCCPHeartbeatDisable" register using write_register() function or functions in GenApiWrapper class, because pytelicam manages the Heartbeat settings.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3.9. save_parameter

This function saves camera parameters using GenICam GenAPI functions.

[Syntax]

pytelicam.CameraDevice.save_parameter(self, file_full_path)

[Parameters]

Parameters	Description
file_full_path (str)	File name (absolute path).

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

The parameters that are saved are defined in the camera description file (XML file) .
Normally, the camera description file is loaded from the camera.
The saved file is a text file.

7.3.10. load_parameter

This function loads camera parameters using GenICam GenAPI functions.

[Syntax]

pytelicam.CameraDevice.load_parameter(self, file_full_path)

[Parameters]

Parameters	Description
file_full_path (str)	File name (absolute path).

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Only files saved by executing save_parameter() function can be loaded.

DPC (Defective Pixel Correction) data is data adjusted for each individual camera.
The DPC data stored in the file is not loaded into the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.3.11. get_last_genicam_error

This function gets information of GenICamError.

[Syntax]

pytelicam.CameraDevice.get_last_genicam_error(self)

[Returns]

GenICam error message.

[Return type]

str

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

GenICamError is an error that may occur during executing functions in GenApiWrapper class, that will return CamApiStatus.GenICamError.

pytelicam collects error information from all threads it managed,and store them into one variable.
If another GenICamError occurs in another thread before or during the execution of this function, the error information returned form this function may be different from the desired error information.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4. CameraStream class (Stream Control)

This class is a class for controlling a stream (image).
This class uses high-level stream functions of TeliCamAPI.

This class manages ImageData objects for passing image data received from the camera to the user application. ImageData objects are managed in the stream ring buffer inside this API.

User applications can get image data using the functions in this class.
The class also provides callback functions to simplify the handling of each event as it occurs.

The object of this class is created and managed as a member of CameraDevice class. User application does not need to create or destroy this object.

[Syntax]

pytelicam.CameraStream

[Functions]

	Name	Description
	open	Opens a stream interface for receiving image data from a camera.
	close	Closes the stream interface for receiving image data from the camera.
	start	Sends the AcquisitionStart command to the camera and requests it to start outputting image data.
	stop	Sends the AcquisitionStop command to the camera and requests it to stop outputting image data.
	abort	Sends the AcquisitionAbort command to the camera and requests it to abort outputting image data.
	get_current_buffer_index	Gets the buffer index of the stream ring buffer inside this API that is storing the latest image data.
	get_next_image	Waits until the image data that receives next is stored in the stream ring buffer inside this API, and get the image data when the store process is completed.
	get_next_image_with_trigger	Sends a software trigger command to the camera and gets the next image data to arrive.
	get_current_buffered_image	Gets the latest stored image data in the stream ring buffer inside this API.
	get_buffered_image	Gets the image data stored in the stream ring buffer inside this API with the specified buffer index.
	chunk_attach_buffer	Attaches the buffer to the chunk adapter of GenICam GenApi.
	chunk_update_buffer	Updates the buffer attached to the chunk adapter of GenICam GenApi.
	chunk_check_buffer_layout	Checks if the buffer contains the chunk data in a known format.
	set_callback_image_acquired	Registers a callback function to be called when new image data is stored in the stream ring buffer inside this API.
	reset_callback_image_acquired	Unregisters a callback function to be called when new image data is stored in the stream ring buffer inside this API.
	set_callback_image_error	Registers a callback function to be called when new abnormal image data is stored in the stream ring buffer inside this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	reset_callback_image_error	Unregisters a callback function to be called when new abnormal image data is stored in the stream ring buffer inside this API.
	set_callback_buffer_busy	Registers the callback function to be called when a buffer busy error occurs.
	reset_callback_buffer_busy	Unregisters the callback function to be called when a buffer busy error occurs.

[Properties]

	Name	Description
	is_open (bool)	Gets a value indicating whether the stream interface is opened or not. This property will return True when the stream interface is opened. Otherwise, this property will return False.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.1. open

This function opens a stream interface for receiving image data from a camera.

This function creates ImageData objects for passing image data received from the camera to the user application. ImageData objects are managed in the stream ring buffer inside this API.

[Syntax]

```
pytelicam.CameraStream.open(
                                self, acquired_signal=None, api_buffer_count=0, max_packet_size=0)
```

[Parameters]

Parameters	Description
acquired_signal (pytelicam.SignalHandle)	A signal object for notifying that image data has been received. When new image data is received, it changes to the signaled state. Specify None if notification is not necessary.
api_buffer_count (int)	The size of the stream ring buffer managed in this API. The available value range is from 1 up to 128. If 0 is specified or if no value is specified, this parameter will be set as shown in the following table according to the payload size.
max_packet_size (int)	The maximum packet size that camera driver can receive, in bytes. This argument is optional. If 0 is specified, the following default value will be used. USB3 camera : 65536 bytes GigE camera : Byte value calculated from jumbo frame (MTU) settings When decreasing overhead of streaming is required, enable Jumbo-Frame of network card that GigE camera is connected, and specify the Jumbo-Frame size to maxPacketSize parameter. Note that maxPacketSize value is packet size excluding Ethernet header (14 bytes). If the network card uses packet size value including Ethernet header as Jumbo-Frame size, specify (Jumbo-Frame size – 14). Also, the value specified must be a multiple of 4. For example, when Jumbo-Frame size is 9014, which usually includes Ethernet header size, specify 9000 instead of 9014. Unless you have a specific reason, we recommend that this argument is specified to 0 or not specified.

Payload Size (byte)	Buffer Count
Less than 16 MBytes	8
16MByte or more and less than 24MByte	7
24MByte or more and less than 32MByte	6
32MByte or more and less than 40MByte	5
40MByte or more and less than 48MByte	4
48MByte or more	3

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

If other application is opening the stream interface of this camera, this function will throw an exception.

When CameraStream object has finished storing an image data to a buffer in the stream ring buffer, CameraStream object will set *acquired_signal* to the signaled state, then call callback function set using set_callback_image_acquired() function.

CameraStream class provides three ways for getting received image data in the stream ring buffer inside this API.

１. using get_next_image() function

Get the next image data to arrive from the camera.

２. using get_current_buffered_image() function

Get the latest stored image data from the stream ring buffer inside this API.

３. using get_current_buffer_index()、get_buffered_image() functions

This way can get the image data from any buffer(ImageData object) in the streaming buffer.
Follow these steps to get the image data:

A. Call get_current_buffer_index() function to get the index of the stream ring buffer that stores the latest image data.
B. Calculate the index of the buffer storing the image data to be get, using the size of the stream ring buffer and the index of the current buffer.
C. Call get_buffered_image() function to get the ImageData object, and get the image data.
D. After using the image data, release the ImageData object using the release() function and return the ImageData object to the stream ring buffer.

If high-priority threads are running in parallel, multiple new image data might have stored in the stream ring buffer when *acquired_signal* was set to the signaled state or callback functions were called.
Note that in this case, *acquired_signal* is set to the signaled state only once and the callback function is executed only once.
If you need all the received image data, get all the image data from the stream ring buffer by following the procedure in 3 above.

Note that stream ring buffer in TeliCamAPI is not image buffer in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.2. close

This function closes the stream interface for receiving image data from the camera.

[Syntax]

pytelicam.CameraStream.close(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

User application has to close the opened stream interface using this function when finished using the CameraStream object.

Calling this function during the other thread is using the CameraStream object may cause error in the other thread.
Call this function after all threads finished using the CameraStream object.

7.4.3. start

This function sends the AcquisitionStart command to the camera and requests it to start outputting image data.

[Syntax]

```
pytelicam.CameraStream.start(
                          self,
                          acquisition_mode=pytelicam.CameraAcquisitionMode.Continuous)
```

[Parameters]

Parameters	Description
acquisition_mode (pytelicam.CameraAcquisitionMode)	Image acquisition mode.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Refer to instruction manual of the camera about the available image acquisition mode.

Set the value of the camera's ImageBufferMode register to On when using the CameraAcquisitionMode.ImageBufferRead mode.
Set it to Off to use in other modes.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.4. stop

This function sends the AcquisitionStop command to the camera and requests it to stop outputting image data.

[Syntax]

pytelicam.CameraStream.stop(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.4.5. abort

This function sends the AcquisitionAbort command to the camera and requests it to abort outputting image data.

[Syntax]

pytelicam.CameraStream.abort(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

The Aborted images are given an error status.

When image data transfer is aborted by this function, the callback function set by the set_callback_image_error() function is not called.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.6. get_current_buffer_index

This function gets the buffer index of the stream ring buffer inside this API that is storing the latest image data.

[Syntax]

pytelicam.CameraStream.get_current_buffer_index(self)

[Returns]

The buffer index of the stream ring buffer

[Return type]

int

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

The range of the buffer index is 0 to the value -1 specified by the *api_buffer_count* argument of open() function of this CameraStream object.
If no images have been stored in the stream ring buffer, -1 will be returned.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.7. get_next_image

This function waits until the image data that receives next is stored in the stream ring buffer inside this API, and get the image data when the store process is completed.

[Syntax]

pytelicam.CameraStream.get_next_image(self, timeout=5000)

[Parameters]

Parameters	Description
timeout (int)	The time-out interval to wait until the next image data to be acquired, in milliseconds. If this value is zero, the function does not enter a wait state. (It always returns immediately.) If this value is -1 or 0xFFFFFFFF, this function will return only when the next image data is stored in the buffer.

[Returns]

An ImageData object

[Return type]

pytelicam.ImageData

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

A time-out error may occur or abnormal image data may be stored in the buffer.

Abnormal image data is as follows.

- An image with an error information. An error information is set by the camera.
- An image for which all image data could not be captured normally due to packet loss, etc.

By checking the *status* value of the returned ImageData object, you can judge whether normal image data has been acquired or not.

The returned ImageData object is locked. If you repeatedly get images without releasing ImageData objects, there will be no unused ImageData object in the stream ring buffer inside this API, and you will not be able to get images. Release the ImageData object immediately after using it. Use release() function or 'with' syntax of python to release it.

```
image_data = cam_device.cam_stream.get_next_image()
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

or

```
with cam_device.cam_stream.get_next_image() as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.8. get_next_image_with_trigger

This function sends a software trigger command to the camera and gets the next image data to arrive.

TriggerMode of the camera should be set to “On” and TriggerSource should be set to “Software”.

[Syntax]

pytelicam.CameraStream.get_next_image_with_trigger(self, timeout=5000)

[Parameters]

Parameters	Description
timeout (int)	The time-out interval to wait until the next image data to be acquired, in milliseconds. If this value is zero, the function does not enter a wait state. (It always returns immediately.) If this value is -1 or 0xFFFFFFFF, this function will return only when the next image data is stored in the buffer.

[Returns]

An ImageData object

[Return type]

pytelicam.ImageData

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

A time-out error may occur or abnormal image data may be stored in the buffer.

Abnormal image data is as follows.

- An image with an error information. An error information is set by the camera.
- An image for which all image data could not be captured normally due to packet loss, etc.

By checking the *status* value of the returned ImageData object, you can judge whether normal image data has been acquired or not.

The returned ImageData object is locked. If you repeatedly get images without releasing ImageData objects, there will be no unused ImageData object in the stream ring buffer inside this API, and you will not be able to get images. Release the ImageData object immediately after using it. Use release() function or 'with' syntax of python to release it.

```
cam_device.genapi.set_enum_str_value('TriggerMode', 'On')
cam_device.genapi.set_enum_str_value('TriggerSource', 'Software')

image_data = cam_device.cam_stream.get_next_image()
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

or

```
cam_device.genapi.set_enum_str_value('TriggerMode', 'On')
cam_device.genapi.set_enum_str_value('TriggerSource', 'Software')

with cam_device.cam_stream.get_next_image() as image_data:
   if image_data.status != pytelicam.CamApiStatus.Success:
       print('Grab error! status = {0}'.format(image_data.status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.9. get_current_buffered_image

This function gets the latest stored image data in the stream ring buffer inside this API.

[Syntax]

pytelicam.CameraStream.get_current_buffered_image(self)

[Returns]

An ImageData object

[Return type]

pytelicam.ImageData

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Abnormal image data may be stored in the buffer.

Abnormal image data is as follows.

- An image with an error information. An error information is set by the camera.
- An image for which all image data could not be captured normally due to packet loss, etc.

By checking the *status* value of the returned ImageData object, you can judge whether normal image data has been acquired or not.

The returned ImageData object is locked. If you repeatedly get images without releasing ImageData objects, there will be no unused ImageData object in the stream ring buffer inside this API, and you wil not be able to get images. Release the ImageData object immediately after using it. Use release() function or 'with' syntax of python to release it.

```
image_data = cam_device.cam_stream.get_current_buffered_image()
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

or

```
with cam_device.cam_stream.get_current_buffered_image() as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.10. get_buffered_image

This function gets the image data stored in the stream ring buffer inside this API with the specified buffer index.

[Syntax]

pytelicam.CameraStream.get_buffered_image(self, buffer_index)

[Parameters]

Parameters	Description
buffer_index (int)	The buffer index of the stream ring buffer. The range of this value is 0 to the value -1 specified in the *api_buffer_count* argument of open() function of this CameraStream object.

[Returns]

An ImageData object

[Return type]

pytelicam.ImageData

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

abnormal image data may be stored in the buffer.

Abnormal image data is as follows.

- An image with an error information. An error information is set by the camera.
- An image for which all image data could not be captured normally due to packet loss, etc.

By checking the *status* value of the returned ImageData object, you can judge whether normal image data has been acquired or not.

The returned ImageData object is locked. If you repeatedly get images without releasing ImageData objects, there will be no unused ImageData object in the stream ring buffer inside this API, and you will not be able to get images. Release the ImageData object immediately after using it. Use release() function or 'with' syntax of python to release it.

```
image_data = cam_device.cam_stream.get_buffered_image(0)
if image_data.status != pytelicam.CamApiStatus.Success:
    print('Grab error! status = {0}'.format(image_data.status))

image_data.release()
```

or

```
with cam_device.cam_stream.get_buffered_image(0) as image_data:
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.11. chunk_attach_buffer

This function attaches the buffer to the chunk adapter of GenICam GenApi.

This function does not support for GenTL interface (CoaXPress camera).

When acquiring chunk data using GenICam GenApi, it is necessary to attach the buffer managed by ImageData object to the chunk adapter of GenICam GenApi.

[Syntax]

pytelicam.CameraStream.chunk_attach_buffer(self, image_data)

[Parameters]

Parameters	Description
image_data (pytelicam.ImageData)	An ImageData object that manages the buffer that should be attached to the chunk adapter of GenICam GenApi.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

If no chunk data is attached to the image data, an exception will be raised.
The chunk data is got using the GenApiWrapper class.

```
image_data = cam_device.cam_stream.get_current_buffered_image()
cam_device.cam_stream.chunk_attach_buffer(image_data)

res, frame_id = cam_device.genapi.get_int_value('ChunkFrameID')
res, exposure_time = cam_device.genapi.get_float_value('ChunkExposureTime')
res, gain = cam_device.genapi.get_float_value('ChunkGain')

print('  ChunkFrameID      : ', frame_id)
print('  ChunkExposureTime : ', exposure_time)
print('  ChunkGain         : ', gain)

image_data.release()
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.12. chunk_update_buffer

This function updates the buffer attached to the chunk adapter of GenICam GenApi.

This function does not support for GenTL interface (CoaXPress camera).

[Syntax]

pytelicam.CameraStream.chunk_attach_buffer(self, image_data)

[Parameters]

Parameters	Description
image_data (pytelicam.ImageData)	An ImageData object that manages the buffer that should be attached to the chunk adapter of GenICam GenApi.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

If no chunk data is attached to the image data, an exception will be raised.
The chunk data is got using the GenApiWrapper class.

If the buffer has not been attached to chunk adapter of GenICam GenApi, or if the layout of the chunk data has changed, an exception will be raised.

It can process faster than chunk_attach_buffer() function.

7.4.13. chunk_check_buffer_layout

This function checks if the buffer contains the chunk data in a known format.

This function does not support for GenTL interface (CoaXPress camera).

[Syntax]

pytelicam.CameraStream.chunk_check_buffer_layout(self, image_data)

[Parameters]

Parameters	Description
image_data (pytelicam.ImageData)	An ImageData object that manages the buffer to check.

[Returns]

The result value.
If the value is True, the buffer contains the chunk data.
If the value is False, the buffer does not contain the chunk data.

[Return type]

bool

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.14. set_callback_image_acquired

This function registers a callback function to be called when new image data is stored in the stream ring buffer inside this API.

[Syntax]

pytelicam.CameraStream.set_callback_image_acquired(self, func)

[Parameters]

Parameters	Description
func	A callback function to be called.

The function that is registered with the func argument must have the following format:

func(image_data)

Parameters	Description
image_data (pytelicam.ImageData)	An ImageData object

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Only one function can be registered.

The buffer managed by *image_data* is locked. Processing of the callback function should finish as soon as possible.

```
def callback_image_acquired(image_data):
    if image_data.status != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(image_data.status))


cam_system = pytelicam.get_camera_system()
cam_num = cam_system.get_num_of_cameras()
cam_device = cam_system.create_device_object(cam_no)
cam_device.open()
cam_device.cam_stream.open()
cam_device.cam_stream.set_callback_image_acquired(callback_image_acquired)
cam_device.cam_stream.start()
…
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.15. reset_callback_image_acquired

This function unregisters a callback function to be called when new image data is stored in the stream ring buffer inside this API.

[Syntax]

pytelicam.CameraStream.reset_callback_image_acquired(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.4.16. set_callback_image_error

This function registers a callback function to be called when new abnormal image data is stored in the stream ring buffer inside this API.

[Syntax]

pytelicam.CameraStream.set_callback_image_error(self)

[Parameters]

Parameters	Description
func	A callback function to be called.

The function that is registered with the func argument must have the following format:

func(status, buffer_index)

Parameters	Description
status (pytelicam.CamApiStatus)	Error status code
buffer_index (int)	The buffer index of the stream ring buffer inside this API that stores abnormal image data.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Only one function can be registered.

Abnormal image data is as follows.

- An image with an error information. An error information is set by the camera.
- An image for which all image data could not be captured normally due to packet loss, etc.

```
def callback_image_error(status, buffer_index):
    print('ImageError Callback! , status={0}, buffer_index={1}' \
           .format(status, buffer_index))

cam_system = pytelicam.get_camera_system()
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

```
cam_num = cam_system.get_num_of_cameras()
cam_device = cam_system.create_device_object(cam_no)
cam_device.open()
cam_device.cam_stream.open()
cam_device.cam_stream.set_callback_image_acquired(callback_image_acquired)
cam_device.cam_stream.set_callback_image_error(callback_image_error)
…
```

7.4.17. reset_callback_image_error

This function unregisters a callback function to be called when new abnormal image data is stored in the stream ring buffer inside this API.

[Syntax]

pytelicam.CameraStream.reset_callback_image_error(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.18. set_callback_buffer_busy

This function registers the callback function to be called when a buffer busy error occurs.

[Syntax]

pytelicam.CameraStream.set_callback_buffer_busy(self, func)

[Parameters]

Parameters	Description
func	A callback function to be called.

The function that is registered with the func argument must have the following format:

func(buffer_index)

Parameters	Description
buffer_index (int)	The buffer index of the stream ring buffer inside this API that is locked and caused the error.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Only one function can be registered.

Buffer busy error indicates that the buffer in the streaming buffer inside this API was locked and new image data has been discarded without being stored in the buffer.

If buffer busy error occurs, you may be able to avoid the error in the following ways:

- Modify the code to release the ImageData object early.
- Increase the number of buffers.
  The number of buffers can be set with the *api_buffer_count* argument of open() function of this CameraStream object.

```
def callback_buffer_busy(buffer_index):
    print('BufferBusy Callback! , buffer_index={0}'.format(buffer_index))

cam_system = pytelicam.get_camera_system()
cam_num = cam_system.get_num_of_cameras()
cam_device = cam_system.create_device_object(cam_no)
cam_device.open()
cam_device.cam_stream.open(None, 32)
cam_device.cam_stream.set_callback_image_acquired(callback_image_acquired)
cam_device.cam_stream.set_callback_buffer_busy(callback_buffer_busy)
…
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.4.19. reset_callback_buffer_busy

This function unregisters the callback function to be called when a buffer busy error occurs.

[Syntax]

pytelicam.CameraStream.reset_callback_buffer_busy(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.5. CameraEvent class (Event Control)

This class is a class for controlling camera event (message) of a camera.
This class uses low-level event functions of TeliCamAPI.

This class manages EventData objects for passing camera event data received from the camera to the user application. EventData objects are managed in the event FIFO buffer inside this API.

User applications can get camera event data using the functions in this class.

Camera event data is not delivered to GenICam GenApi. Therefore, event information such as Timestamp cannot be got using the functions of GenApiWrapper object.

The object of this class is created and managed as a member of CameraDevice class. User application does not need to create or destroy this object.

[Syntax]

pytelicam.CameraEvent

[Functions]

	Name	Description
	open	Opens an event interface for receiving camera event data from a camera.
	close	Closes the event interface for receiving camera event data from the camera.
	activate	Activates the specified camera event of the camera.
	deactivate	Deactivates the specified camera event of the camera.
	get_event_data	Gets stored camera event data from the event FIFO buffer inside this API.
	clear_event_data	Clears stored camera event data from the event FIFO buffer inside this API.
	get_queueing_count	Gets the number of camera event data stored in the event FIFO buffer inside this API.
	get_lost_count	Gets the number of lost event data.

[Properties]

	Name	Description
	is_open (bool)	Gets a value indicating whether the event interface is opened or not. This property will return True when the event interface is opened. Otherwise, this property will return False.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.5.1. open

This function opens an event interface for receiving camera event data from a camera.

This function creates EventData objects for passing camera event data received from the camera to the user application. EventData objects are managed in the event FIFO buffer inside this API.

[Syntax]

pytelicam.CameraEvent.open(self, api_buffer_count=32)

[Parameters]

Parameters	Description
api_buffer_count (int)	The size of the event FIFO buffer managed in this API. The available value range is from 1 up to 128.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

If other application is opening the event interface of this camera, this function will throw an exception.

Use the get_event_data() function to get the camera event data in the order they were stored in the event FIFO buffer inside this API.

```
cam_device.cam_event.open()
cam_device.cam_event.activate(pytelicam.CameraEventType.ExposureEnd)

acquired_signal = cam_system.create_signal()
cam_device.cam_stream.open(acquired_signal)
cam_device.cam_stream.start()

for i in range(10):
    res = cam_device.genapi.execute_command('TriggerSoftware')
    if res != pytelicam.CamApiStatus.Success:
        print("Can't execute TriggerSoftware.")
        break

    # If you don't use 'with' statement, event_data.release() must be
    # called after using event_data.
    with cam_device.cam_event.get_event_data(1000) as event_data :
        if event_data.status == pytelicam.CamApiStatus.Success:
            print('\nrequest_id : {0}, event_id : {1}, timestamp  : {2}, ' \
               .format( \
                   hex(event_data.request_id), \
                   hex(event_data.event_id), \
                   event_data.timestamp))

        else:
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

```
            print('Error : event_data was not received.')

    res = cam_system.wait_for_signal(acquired_signal)
    if res != pytelicam.CamApiStatus.Success:
        print('Grab error! status = {0}'.format(res))
        break
    else:
        print('Image Acquired.')

cam_device.cam_stream.stop()
cam_device.cam_stream.close()
cam_device.cam_event.deactivate(pytelicam.CameraEventType.ExposureEnd)
cam_device.cam_event.close()
```

When multiple camera event data are activated for one camera, multiple camera event data are transferred in a short period of time. Therefore, processing may not be completed according to the usage environment and application. At this time, one or more received camera event data may be discarded without being stored in the event FIFO buffer inside this API.

If some camera event data is not received, you may be able to avoid this problem in the following ways:

- Modify the code to release the EventData object got by get_event_data() function early.
- Increase the number of buffers.
  The number of buffers can be set with the *api_buffer_count* argument of this function.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.5.2. close

This function closes the event interface for receiving camera event data from the camera.

[Syntax]

pytelicam.CameraEvent.close(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

User application has to close the opened event interface using this function when finished using the CameraEvent object.

Calling this function during the other thread is using the CameraEvent object may cause error in the other thread.
Call this function after all threads finished using the CameraEvent object.

7.5.3. activate

This function activates the specified camera event of the camera.

[Syntax]

pytelicam.CameraEvent.activate(self, event_type)

[Parameters]

Parameters	Description
event_type (pytelicam.CameraEventType)	Type of camera event to activate.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function sets the EventSelector and EventNotification nodes of the camera.

To activate camera events that are not defined by CameraEventType, set them using GenApiWrapper object.

An exception may be thrown depending on the order to set features to the camera.
It depends on the model or firmware version of the camera.
For example, on some GigE cameras, the FrameTrigger event can only be set when TriggerMode is On.

Since the types of events that can be notified depends on the camera model, refer to the user's manual of the camera for more details.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.5.4. deactivate

This function deactivates the specified camera event of the camera.

[Syntax]

pytelicam.CameraEvent.deactivate(self, event_type)

[Parameters]

Parameters	Description
event_type (pytelicam.CameraEventType)	Type of camera event to deactivate.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.5.5. get_event_data

This function gets stored camera event data from the event FIFO buffer inside this API.
Each time this function is executed, the event data is got one by one in the order in which it was stored in the buffer.

[Syntax]

pytelicam.CameraEvent.get_event_data(self, timeout=0)

[Parameters]

Parameters	Description
timeout (int)	The time-out interval to wait until a new camera event data arrives if there is no camera event data in the event FIFO buffer inside this API, in milliseconds. If this value is zero, the function does not enter a wait state. (It always returns immediately.) If this value is -1 or 0xFFFFFFFF, this function will return only when a new camera event data is stored in the buffer.

[Returns]

An EventData object

[Return type]

pytelicam.EventData

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

A time-out error may occur or abnormal camera event data may be stored in the buffer.

Abnormal camera event data is as follows.

- A camera event data with an error information. An error information is set by the camera.
- A camera event data for which all data could not be captured normally due to packet loss, etc.

By checking the status value of the returned EventData object, you can judge whether normal camera event data has been acquired or not.

The returned EventData object is locked.   If you repeatedly get camera event data without releasing EventData objects, there will be no unused EventData object in the event FIFO buffer inside this API, and you will not be able to get camera event data. Release the EventData object immediately after using it. Use release() function or 'with' syntax of python to release it.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.5.6. clear_event_data

This function clears stored camera event data from the event FIFO buffer inside this API.

[Syntax]

pytelicam.CameraEvent.clear_event_data(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.5.7. get_queueing_count

This function gets the number of camera event data stored in the event FIFO buffer inside this API.

[Syntax]

pytelicam.CameraEvent.get_queueing_count(self)

[Returns]

The number of camera event data stored in the event FIFO buffer inside this API.

[Return type]

int

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.5.8. get_lost_count

This function gets the number of lost event data.

If there is no buffer available in the event FIFO buffer inside this API, the event data is discarded.

[Syntax]

pytelicam.CameraEvent.get_lost_count(self)

[Returns]

The number of lost event data.

[Return type]

int

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

Release the EventData object got by get_event_data() function as soon as possible.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6. CameraControl class

CameraControl is a class for controlling features of a camera. CameraControl object allows user application to control a camera without paying attention to interface type, model of the camera, and address of registers.

For cameras that comply with IIDC2 standard, GenICam GenApi libraries are not used. It is processed at high speed by register access.(Except for some functions)

For cameras that do not comply with IIDC 2, the GenICam GenApi library is used. All methods are not available when GenApi module was disabled on opening the camera.

There are methods that are available in all cameras. There also are methods that are not available in some model of camera because the corresponding features are not supported. Please check instruction manual of the camera to confirm the availability of the feature.

CameraControl object will be created as a member of CameraDevice object on creating CameraDevice class instance, and is under the management of CameraDevice class.

User application should not dispose (delete) CameraControl object.

[Syntax]

pytelicam.CameraControl

[Functions]

	Name	Description
ImageFormatControl		
◆	get_image_format_selector	Gets the image format of the camera.
◆	set_image_format_selector	Sets the image format of the camera.
Scable		
◆	get_sensor_width	Gets the effective width of the camera in pixels.
◆	get_sensor_height	Gets the effective height of the camera in pixels.
◆	get_roi	Gets an image Region Of Interest (ROI) of the camera.
◆	set_roi	Sets an image Region Of Interest (ROI) of the camera.
◆	get_width_min_max	Gets the minimum and maximum available image width and increment value of the image width.
◆	get_width	Gets the current image width in pixels.
◆	set_width	Sets the current image width in pixels.
◆	get_height_min_max	Gets the minimum and maximum available image height and increment value of the image height.
◆	get_height	Gets the current image height in pixels.
◆	set_height	Sets the current image height in pixels.
◆	get_offset_x_min_max	Gets the minimum and maximum available horizontal offset from the origin to the region of interest and the increment value.
◆	get_offset_x	Gets the horizontal offset from the origin to the region of interest in pixels.
◆	set_offset_x	Sets the horizontal offset from the origin to the region of interest in pixels.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
◆	get_offset_y_min_max	Gets the minimum and maximum available vertical offset from the origin to the region of interest and the increment value.
◆	get_offset_y	Gets the vertical offset from the origin to the region of interest in pixels.
◆	set_offset_y	Sets the vertical offset from the origin to the region of interest in pixels.
	Binning	
◆	get_binning_horizontal_min_max	Gets the minimum and maximum available horizontal binning mode value of the camera.
◆	get_binning_horizontal	Gets the horizontal binning mode value of the camera.
◆	set_binning_horizontal	Sets the horizontal binning mode value of the camera.
◆	get_binning_vertical_min_max	Gets the minimum and maximum available vertical binning mode value of the camera.
◆	get_binning_vertical	Gets the vertical binning mode value of the camera.
◆	set_binning_vertical	Sets the vertical binning mode value of the camera.
	Decimation	
◆	get_decimation_horizontal_min_max	Gets the minimum and maximum available horizontal decimation mode value of the camera.
◆	get_decimation_horizontal	Gets the horizontal decimation mode value of the camera.
◆	set_decimation_horizontal	Sets the horizontal decimation mode value of the camera.
◆	get_decimation_vertical_min_max	Gets the minimum and maximum available vertical decimation mode value of the camera.
◆	get_decimation_vertical	Gets the vertical decimation mode value of the camera.
◆	set_decimation_vertical	Sets the vertical decimation mode value of the camera.
	Reverse	
◆	get_reverse_x	Gets the horizontal image flip mode value of the camera.
◆	set_reverse_x	Sets the horizontal image flip mode value of the camera.
◆	get_reverse_y	Gets the vertical image flip mode value of the camera.
◆	set_reverse_y	Sets the vertical image flip mode value of the camera.
	PixelFormat	
◆	get_pixel_format	Gets the pixel format of the camera.
◆	set_pixel_format	Sets the pixel format of the camera.
	TestPattern	
◆	get_test_pattern	Gets the test pattern type of the camera.
◆	set_test_pattern	Sets the test pattern type of the camera.
	AquisitionControl	
◆	get_stream_payload_size	Gets the current payload size of image stream.
◆	Get_stream_enable	Gets the enable state of image streaming in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
◆	get_acquisition_frame_count_min_max	Gets the minimum and maximum available AcquisitionFrameCount value of the camera.
◆	get_acquisition_frame_count	Gets the AcquisitionFrameCount value of the camera.
◆	set_acquisition_frame_count	Sets the AcquisitionFrameCount value of the camera.
◆	get_acquisition_frame_control	Gets the frame rate setting of the camera.
◆	set_acquisition_frame_control	Sets the frame rate setting of the camera.
◆	get_acquisition_frame_rate_min_max	Gets the minimum and maximum available frame rate value of the camera.
◆	get_acquisition_frame_rate	Gets the frame rate value of the camera.
◆	set_acquisition_frame_rate	Sets the frame rate value of the camera.
◆	execute_acquisition_start	Sends a AcquisitionStart command to the camera.
◆	get_high_framerate_mode	Gets the HighFramerateMode value of the camera.
◆	set_high_framerate_mode	Sets the HighFramerateMode value of the camera.
	ImageBuffer	
◆	get_image_buffer_mode	Gets the current image buffer mode type of the camera.
◆	set_image_buffer_mode	Sets the current image buffer mode type of the camera.
◆	get_image_buffer_frame_count	Gets the number of images stored in the image buffer of the camera.
◆	execute_image_buffer_read	Reads images from the image buffer of the camera.
	TriggerControl	
◆	get_trigger_mode	Gets the trigger mode value of the camera.
◆	set_trigger_mode	Sets the trigger mode value of the camera.
◆	get_trigger_sequence	Gets the trigger sequence of random trigger shutter.
◆	set_trigger_sequence	Sets the trigger sequence of random trigger shutter.
◆	get_trigger_suorce	Gets the trigger source of random trigger shutter.
◆	set_trigger_suorce	Sets the trigger source of random trigger shutter.
◆	get_trigger_additional_parameter_min_max	Gets the minimum and maximum available TriggerAdditionalParameter (or AcquisitionBurstFrameCount) value of the camera.
◆	get_trigger_additional_parameter	Gets the TriggerAdditionalParameter(or AcquisitionBurstFrameCount) value of the camera.
◆	set_trigger_additional_parameter	Sets the TriggerAdditionalParameter(or AcquisitionBurstFrameCount) value of the camera.
◆	get_trigger_delay_min_max	Gets the minimum and maximum value available for “TriggerDelay” register.
◆	get_trigger_delay	Gets the TriggerDelay value of the camera.
◆	set_trigger_delay	Sets the TriggerDelay value of the camera.
◆	execute_software_trigger	Sends a software trigger command to the camera.
◆	get_trigger_activation	Gets the trigger activation of hardware trigger.
◆	set_trigger_activation	Sets the trigger activation of hardware trigger.
	ExposureTime	
◆	get_exposure_time_control	Gets the exposure time control mode of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
◆	set_exposure_time_control	Sets the exposure time control mode of the camera.
◆	get_exposure_time_min_max	Gets the minimum and maximum available exposure time of the camera when ExposureTimeControl is set to CAM_EXPOSURE_TIME_CONTROL_MANUAL.
◆	get_exposure_time	Gets the exposure time of the camera when ExposureTimeControl is set to CAM_EXPOSURE_TIME_CONTROL_MANUAL.
◆	set_exposure_time	Sets the exposure time of the camera when ExposureTimeControl is set to CAM_EXPOSURE_TIME_CONTROL_MANUAL.
◆	get_short_exposure_mode	Gets the ShortExposureModevalue of the camera.
◆	set_short_exposure_mode	Sets the ShortExposureModevalue of the camera.
DigitalIOControl		
◆	get_line_mode_all	Gets the input / output value for all digital I/O lines of the camera.
◆	set_line_mode_all	Sets the input / output value for all digital I/O lines of the camera.
◆	get_line_mode	Gets the input / output value for each digital I/O line of the camera.
◆	set_line_mode	Sets the input / output value for each digital I/O line of the camera.
◆	get_line_inverter_all	Gets the polarity value for all digital I/O lines of the camera.
◆	set_line_inverter_all	Sets the polarity value for all digital I/O lines of the camera.
◆	get_line_inverter	Gets the polarity value for each digital I/O line of the camera.
◆	set_line_inverter	Sets the polarity value for each digital I/O line of the camera.
◆	get_line_status_all	Gets the line status for all digital I/O lines of the camera.
◆	get_line_status	Gets the line status for each digital I/O line of the camera.
◆	get_user_output_value_all	Gets the user output value for all user output lines in the camera.
◆	set_user_output_value_all	Sets the user output value for all user output lines in the camera.
◆	get_user_output_value	Gets the user output value for each user output line of the camera.
◆	set_user_output_value	Sets the user output value for each user output line of the camera.
◆	get_line_source	Gets the source of the output signal.
◆	set_line_source	Sets the source of the output signal.
AntiGlitch / AntiChattering		
◆	get_anti_glitch_min_max	Gets the minimum and maximum integration time of digital input signal in thecamera.
◆	get_anti_glitch	Gets the integration time of digital input signal in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
◆	set_anti_glitch	Sets the integration time of digital input signal in the camera.
◆	get_anti_chattering_min_max	Gets the minimum and maximum insensible time of digital input signal in the camera.
◆	get_anti_chattering	Gets the insensible time of digital input signal in the camera.
◆	set_anti_chattering	Sets the insensible time of digital input signal in the camera.
TimerControl		
◆	get_timer_duration_min_max	Gets the minimum and maximum duration of Timer0Active signal.
◆	get_timer_duration	Gets the duration of Timer0Active signal.
◆	set_timer_duration	Sets the duration of Timer0Active signal.
◆	get_timer_delay_min_max	Gets the minimum and maximum delay of Timer0Active signal.
◆	get_timer_delay	Gets the delay of Timer0Active signal.
◆	set_timer_delay	Sets the delay of Timer0Active signal.
◆	get_timer_trigger_source	Gets the source of Timer0Active pulse.
◆	set_timer_trigger_source	Sets the source of Timer0Active pulse.
Gain		
◆	get_gain_min_max	Gets the minimum and maximum Gain value of the camera.
◆	get_gain	Gets the Gain value of the camera.
◆	set_gain	Sets the Gain value of the camera.
◆	get_gain_auto	Gets the AGC (automatic gain control) mode of the camera.
◆	set_gain_auto	Sets the AGC (automatic gain control) mode of the camera.
BlackLevel		
◆	get_black_level_min_max	Gets the minimum and maximum black level value of the camera.
◆	get_black_level	Gets the black level value of the camera.
◆	set_black_level	Sets the black level value of the camera.
Gamma		
◆	get_gamma_min_max	Gets the minimum and maximum gamma correction value of the camera.
◆	get_gamma	Gets the gamma correction value of the camera.
◆	set_gamma	Sets the gamma correction value of the camera.
WhiteBalance		
◆	get_balance_ratio_min_max	Gets the minimum and maximum white balance gain of the camera.
◆	get_balance_ratio	Gets the white balance gain of the camera.
◆	set_balance_ratio	Sets the white balance gain of the camera.
◆	get_balance_white_auto	Gets the auto white balance mode of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
◆	set_balance_white_auto	Sets the auto white balance mode of the camera.
	Hue	
◆	get_hue_min_max	Gets the minimum and maximum hue value of the camera.
◆	get_hue	Gets the hue value of the camera.
◆	set_hue	Sets the hue value of the camera.
	Saturation	
◆	get_saturation_min_max	Gets the minimum and maximum saturation value of the camera.
◆	get_saturation	Gets the saturation value of the camera.
◆	set_saturation	Sets the saturation value of the camera.
	Sharpness	
◆	get_sharpness_min_max	Gets the minimum and maximum sharpness value of the camera.
◆	get_sharpness	Gets the sharpness value of the camera.
◆	set_sharpness	Sets the sharpness value of the camera.
	ColorCorrectionMatrix	
◆	get_color_correction_matrix_ min_max	Gets the minimum and maximum coefficient value of color correction matrix in the camera.
◆	get_color_correction_matrix	Gets the coefficient value of color correction matrix in the camera.
◆	set_color_correction_matrix	Sets the coefficient value of color correction matrix in the camera.
	LUTControl	
◆	get_lut_enable	Gets the enable state of LUT function in the camera.
◆	set_lut_enable	Sets the enable state of LUT function in the camera.
◆	get_lut_value	Gets the output level of LUT in the camera.
◆	set_lut_value	Sets the output level of LUT in the camera.
	UserSetControl	
◆	execute_user_set_load	Loads parameters saved in non-volatile memory (user memory) of the camera and save them to registers in the camera.
◆	execute_user_set_save	Saves the current parameters to non-volatile memory (user memory) of the camera.
◆	execute_user_set_quick_save	Saves current parameters to volatile memory of the camera.
◆	get_user_set_default	Gets the user memory channel to be loaded when the camera is powered on.
◆	set_user_set_default	Sets the user memory channel to be loaded when the camera is powered on.
◆	execute_user_set_save_and_ set_default	Saves the current parameters to non-volatile memory (user memory) of the camera, and sets the user memory channel to be loaded when the camera is powered on.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
	SequentialShutterControl	
◆	get_sequential_shutter_enable	Gets the enable state of Sequential Shutter function in the camera.
◆	set_sequential_shutter_enable	Sets the enable state of Sequential Shutter function in the camera.
◆	get_sequential_shutter_ terminate_at_min_max	Gets the minimum and maximum number of index to repeat the sequence.
◆	get_sequential_shutter_ terminate_at	Gets the number of index to repeat the sequence.
◆	get_sequential_shutter_ terminate_at	Sets the number of index to repeat the sequence.
◆	get_sequential_shutter_ index_min_max	Gets the minimum and maximum value of sequence number of sequential shutter function.
◆	get_sequential_shutter_ entry_min_max	Gets the minimum and maximum value of UserSet number of sequential shutter function.
◆	get_sequential_shutter_entry	Gets the value of UserSet number of sequential shutter function.
◆	set_sequential_shutter_entry	Sets the value of UserSet number of sequential shutter function.
	UserDefinedName	
◆	get_user_defined_name	Gets the user-defined name of the camera.
◆	set_user_defined_name	Sets the user-defined name of the camera.
	Chunk	
◆	get_chunk_mode_active	Gets the activation state of Chunk function in the camera.
◆	set_chunk_mode_active	Sets the activation value of Chunk function in the camera.
◆	get_chunk_enable	Gets the enable state of Chunk data in the camera.
◆	set_chunk_enable	Sets the enable state of Chunk data in the camera.
◆	get_chunk_user_area_length	Gets the length of ChunkUserAreaTable in the camera.
◆	get_chunk_user_area_table	Gets the data of ChunkUserAreaTable in the camera.
◆	set_chunk_user_area_table	Sets the data of ChunkUserAreaTable in the camera.
	FrameSynchronization	
◆	get_frame_synchronization	Gets the camera frame synchronization method.
◆	set_frame_synchronization	Sets the camera frame synchronization method.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.1. ImageFormatControl

This feature group performs control of image format of the camera. (Format0 ～ Format2)

The feature of image format control depends on the camera or camera's firmware version.

For details about ImageFormatControl features, refer to instruction manual of the camera.

7.6.1.1. get_image_format_selector

Gets the image format of the camera.

[Syntax]

pytelicam.CameraControl.get_image_format_selector(self)

[Returns]

A tuple type data (status, format)

status	: A status code indicating the result of the operation.
format	: The type of image format.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraImageFormat)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if ImageFormatSelector register (or node) is not implemented in the camera.

The feature may be different depending on the camera.
For details about ImageFormatControl features, refer to instruction manual of the camera.

An example is as follows:

```
status, fmt = cam_device.genapi.get_image_format_selector()
print('status={0} , format={1}'.format(status, fmt))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.1.2. set_image_format_selector

Sets the image format of the camera.

[Syntax]

pytelicam.CameraControl.set_image_format_selector(self, format)

[Parameters]

Parameters	Description
format (pytelicam.CameraImageFormat)	The type of image format.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ImageFormatSelector register (or node) is not implemented in the camera.

Settings can not be changed during image stream data output.

The feature may be different depending on the camera.
For details about ImageFormatControl features, refer to instruction manual of the camera.

An example is as follows:

```
st = cam_device.genapi.set_image_format \
(pytelicam.pytelicam.CameraImageFormat.Format0)
print('status={0}'.format(st))
```

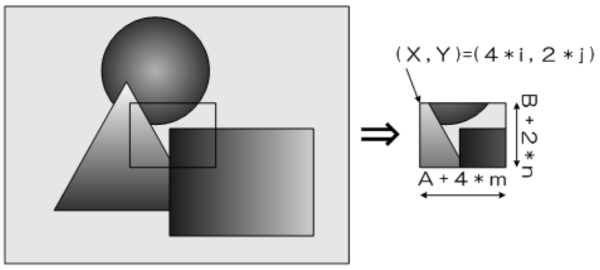
Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2. Scalable

This feature group performs control of scalable mode of the camera.

Scalable function reads out the region of interest (ROI) of the sensor.
If height size is set small, it is possible to increase the frame rate.
Only single rectangle is selectable. Concave or convex shape is not selectable.

- Window size: {A + 4 × m (H)} × {B + 2 × n (V)}A, B = minimum unit size
  m, n = integer
  The window size is equal or less than maximum image size.
- Start address: {4 x i (H)} x {2 x j (V)}
  i, j = integer
  The window size is equal or less than maximum image size.



For details about Scalable features, refer to instruction manual of the camera.

7.6.2.1. get_sensor_width

Gets the effective width of the camera in pixels.

[Syntax]

pytelicam.CameraControl.get_sensor_width(self)

[Returns]

A tuple type data (status, sensor_width)

status	: A status code indicating the result of the operation.
sensor_width	: The effective width of the sensor in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.2. get_sensor_height

Gets the effective height of the camera in pixels.

[Syntax]

pytelicam.CameraControl.get_sensor_height(self)

[Returns]

A tuple type data (status, sensor_height)

status	: A status code indicating the result of the operation.
sensor_height	: The effective height of the sensor in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.6.2.3. get_roi

Gets an image Region Of Interest (ROI) of the camera.

[Syntax]

pytelicam.CameraControl.get_roi(self)

[Returns]

A tuple type data (status, width, height, offset_x, offset_y)

status	: A status code indicating the result of the operation.
width	: The current image width, in pixels.
height	: The current image height, in pixels.
offset_x	: The current horizontal offset from the origin to the region of interest, in pixels.
offset_y	: The current vertical offset from the origin to the region of interest, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int, int, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.4. set_roi

Sets an image Region Of Interest (ROI) of the camera.

[Syntax]

pytelicam.CameraControl.set_roi(self, width, height, offset_x, offset_y)

[Parameters]

Parameters	Description
width (int)	The image width, in pixels.
height (int)	The image height, in pixels.
offset_x (int)	The horizontal offset from the origin to the region of interest, in pixels.
offset_y (int)	The vertical offset from the origin to the region of interest, in pixels.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

Settings cannot be changed during image stream data output.
However, some cameras can only change OffsetX and OffsetY even during image stream data output.
For details about Scalable features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.5. get_width_min_max

Gets the minimum and maximum available image width and increment value of the image width.

[Syntax]

pytelicam.CameraControl.get_width_min_max(self)

[Returns]

A tuple type data (status, width_min, width_max, width_inc)

status	: A status code indicating the result of the operation.
width_min	: The minimum image width, in pixels.
width_max	: The maximum image width, in pixels.
width_inc	: The increment value of the image width, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.6.2.6. get_width

Gets the current image width in pixels.

[Syntax]

pytelicam.CameraControl.get_width(self)

[Returns]

A tuple type data (status, width)

status	: A status code indicating the result of the operation.
width	: The current image width, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.7. set_width

Sets the current image width in pixels.

[Syntax]

pytelicam.CameraControl.set_width(self, width)

[Parameters]

Parameters	Description
width (int)	The image width, in pixels.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

Settings cannot be changed during image stream data output.

7.6.2.8. get_height_min_max

Gets the minimum and maximum available image height and increment value of the image height.

[Syntax]

pytelicam.CameraControl.get_height_min_max(self)

[Returns]

A tuple type data (status, height_min, height_max, height_inc)

status	: A status code indicating the result of the operation.
height_min	: The minimum image height, in pixels.
height_max	: The maximum image height, in pixels.
height_inc	: The increment value of the image height, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.9. get_height

Gets the current image height in pixels.

[Syntax]

pytelicam.CameraControl.get_height(self)

[Returns]

A tuple type data (status, height)

status : A status code indicating the result of the operation.

height : The current image height, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.6.2.10. set_height

Sets the current image height in pixels.

[Syntax]

pytelicam.CameraControl.set_height(self, height)

[Parameters]

Parameters	Description
height (int)	The image height, in pixels.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

Settings cannot be changed during image stream data output.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.11. get_offset_x_min_max

Gets the minimum and maximum available horizontal offset from the origin to the region of interest and the increment value.

[Syntax]

pytelicam.CameraControl.get_offset_x_min_max(self)

[Returns]

A tuple type data (status, offset_x_min, offset_x_max, offset_x_inc)

status	: A status code indicating the result of the operation.
offset_x_min	: The minimum horizontal image offset, in pixels.
offset_x_max	: The maximum horizontal image offset, in pixels.
offset_x_inc	: The increment value of horizontal image offset, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.6.2.12. get_offset_x

Gets the horizontal offset from the origin to the region of interest in pixels.

[Syntax]

pytelicam.CameraControl.get_offset_x(self)

[Returns]

A tuple type data (status, offset_x)

status	: A status code indicating the result of the operation.
offset_x	: The current horizontal image offset, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.13. set_offset_x

Sets the horizontal offset from the origin to the region of interest in pixels.

[Syntax]

pytelicam.CameraControl.set_offset_x(self, offset_x)

[Parameters]

Parameters	Description
offset_x (int)	The horizontal image offset, in pixels.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

Whether it can be set during image stream data output varies depending on the camera.
For details about Scalable features, refer to instruction manual of the camera.

7.6.2.14. get_offset_y_min_max

Gets the minimum and maximum available vertical offset from the origin to the region of interest and the increment value.

[Syntax]

pytelicam.CameraControl.get_offset_y_min_max(self)

[Returns]

A tuple type data (status, offset_y_min, offset_y_max, offset_y_inc)

status	: A status code indicating the result of the operation.
offset_y_min	: The minimum vertical image offset, in pixels.
offset_y_max	: The maximum vertical image offset, in pixels.
offset_y_inc	: The increment value of vertical image offset, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.2.15. get_offset_y

Gets the vertical offset from the origin to the region of interest in pixels.

[Syntax]

pytelicam.CameraControl.get_offset_y(self)

[Returns]

A tuple type data (status, offset_y)

status	: A status code indicating the result of the operation.
offset_y	: The current vertical image offset, in pixels.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.6.2.16. set_offset_y

Sets the vertical offset from the origin to the region of interest in pixels.

[Syntax]

pytelicam.CameraControl.set_offset_y(self, offset_y)

[Parameters]

Parameters	Description
offset_y (int)	The vertical image offset, in pixels.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

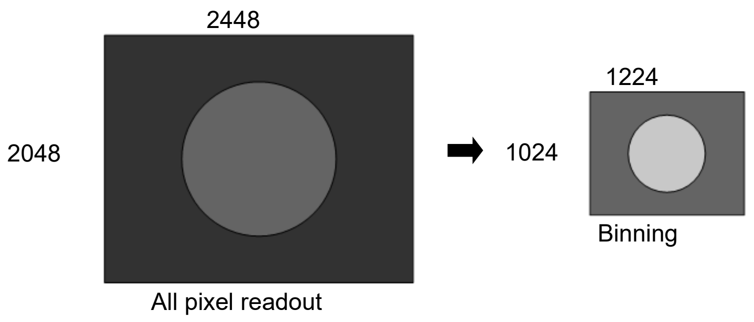
Whether it can be set during image stream data output varies depending on the camera.
For details about Scalable features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.3. Binning

This feature group performs control of binning features of the camera.

Binning features add the neighboring pixels together.
Binning features can increase the sensitivity of the image, make frame rate faster, and decrease interface bandwidth occupation.



Binning operation（e.g. 2x2 binning）

For details about Binning features, refer to instruction manual of the camera.

7.6.3.1. get_binning_horizontal_min_max

Gets the minimum and maximum available horizontal binning mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_binning_horizontal_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of horizontal binning mode.
max	: The maximum value of horizontal binning mode.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if BinningHorizontal register (or node) is not implemented in the camera.

Conditions that can get and set the binning parameters depend on the camera.
For details about Binning features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.3.2. get_binning_horizontal

Gets the horizontal binning mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_binning_horizontal(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of horizontal binning mode.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if BinningHorizontal register (or node) is not implemented in the camera.

Conditions that can get and set the binning parameters depend on the camera.
For details about Binning features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.3.3. set_binning_horizontal

Sets the horizontal binning mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_binning_horizontal(self, value)

[Parameters]

Parameters	Description
value (int)	The value of horizontal binning mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if BinningHorizontal register (or node) is not implemented in the c amera.

Settings cannot be changed during image stream data output.

Conditions that can get and set the binning parameters depend on the camera.
For details about Binning features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.3.4. get_binning_vertical_min_max

Gets the minimum and maximum available vertical binning mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_binning_vertical_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of vertical binning mode.
max	: The maximum value of vertical binning mode.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if BinningVertical register (or node) is not implemented in the camera.

Conditions that can get and set the binning parameters depend on the camera.
For details about Binning features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.3.5. get_binning_vertical

Gets the vertical binning mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_binning_vertical(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of vertical binning mode.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if BinningVertical register (or node) is not implemented in the camera.

Conditions that can get and set the binning parameters depend on the camera.
For details about Binning features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.3.6. set_binning_vertical

Sets the vertical binning mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_binning_vertical(self, value)

[Parameters]

Parameters	Description
value (int)	The value of vertical binning mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if BinningHorizontal register (or node) is not implemented in the camera.

Settings cannot be changed during image stream data output.

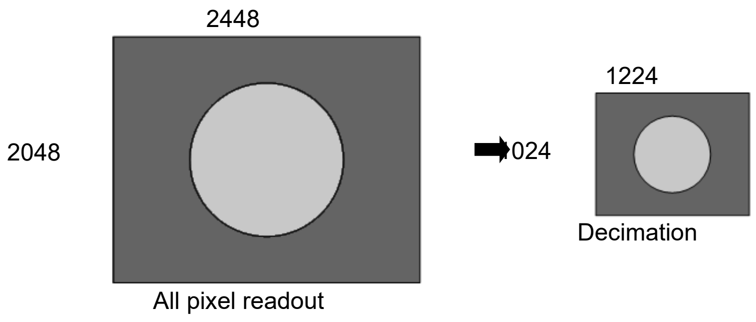
Conditions that can get and set the binning parameters depend on the camera.
For details about Binning features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.4. Decimation

This feature group performs control of decimation feature of the camera.

Decimation features reads out all effective areas at high speed by skipping pixels and lines.
Decimation features can make frame rate faster, and decrease interface bandwidth occupation.



Decimation operation（e.g. 2x2 decimation）

For details about Decimation features, refer to instruction manual of the camera.

7.6.4.1. get_decimation_horizontal_min_max

Gets the minimum and maximum available horizontal decimation mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_decimation_horizontal_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of horizontal decimation mode.
max	: The maximum value of horizontal decimation mode.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if DecimationHorizontal register (or node) is not implemented in the camera.

Conditions that can get and set the decimation parameters depend on the camera.
For details about Decimation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.4.2. get_decimation_horizontal

Gets the horizontal decimation mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_decimation_horizontal(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of horizontal decimation mode.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if DecimationHorizontal register (or node) is not implemented in the camera.

Conditions that can get and set the decimation parameters depend on the camera.
For details about Decimation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.4.3. set_decimation_horizontal

Sets the horizontal decimation mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_decimaion_horizontal(self, value)

[Parameters]

Parameters	Description
value (int)	The value of horizontal decimation mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if DecimationHorizontal register (or node) is not implemented in the camera.

Settings cannot be changed during image stream data output.

Conditions that can get and set the decimation parameters depend on the camera.
For details about Decimation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.4.4. get_decimation_vertical_min_max

Gets the minimum and maximum available vertical decimation mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_decimation_vertical_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of vertical decimation mode.
max	: The maximum value of vertical decimation mode.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if DecimationVertical register (or node) is not implemented in the camera.

Conditions that can get and set the decimation parameters depend on the camera.
For details about Decimation features, refer to instruction manual of the camera.

7.6.4.5. get_decimation_vertical

Gets the vertical decimation mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_decimation_vertical(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of vertical decimation mode.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if DecimationVertical register (or node) is not implemented in the camera.

Conditions that can get and set the decimation parameters depend on the camera.
For details about Decimation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.4.6. set_decimation_vertical

Sets the vertical decimation mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_decimation_vertical(self, value)

[Parameters]

Parameters	Description
value (int)	The value of vertical decimation mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if DecimationHorizontal register (or node) is not implemented in the camera.

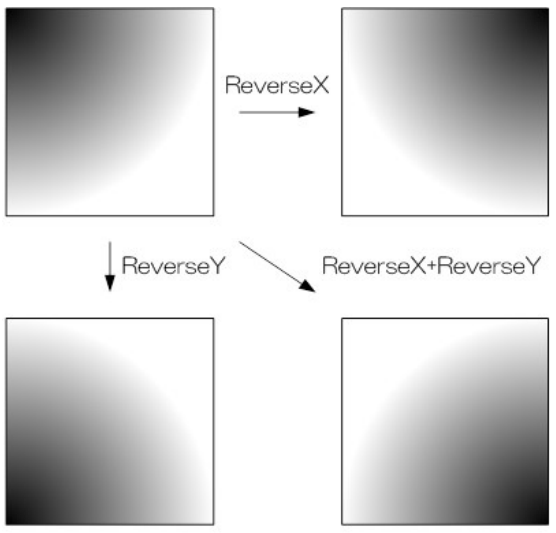
Settings cannot be changed during image stream data output.

Conditions that can get and set the decimation parameters depend on the camera.
For details about Decimation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.5. Reverse

This feature group performs control of image flipping feature of the camera.



For details about Reverse features, refer to instruction manual of the camera.

7.6.5.1. get_reverse_x

Gets the horizontal image flip mode value of the camera.

[Syntax]

Pytelicam.CameraControl.get_reverse_x(self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The value of horizontal image flip mode.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ReverseX register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.5.2. set_reverse_x

Sets the horizontal image flip mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_reverse_x(self, value)

[Parameters]

Parameters	Description
value (bool)	The value of horizontal image flip mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ReverseX register (or node) is not implemented in the camera.

Settings cannot be changed during image stream data output.

7.6.5.3. get_reverse_y

Gets the vertical image flip mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_reverse_y(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of vertical image flip mode.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ReverseY register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.5.4. set_reverse_y

Sets the vertical image flip mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_reverse_y(self, value)

[Parameters]

Parameters	Description
value (bool)	The value of vertical image flip mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ReverseY register (or node) is not implemented in the camera.

Settings cannot be changed during image stream data output.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.6. PixelFormat

This feature group performs control of PixelFormat of image to be acquired.
For details about PixelFormat features, refer to instruction manual of the camera.

7.6.6.1. get_pixel_format

Gets the pixel format of the camera.

[Syntax]

pytelicam.CameraControl.get_pixel_format(self)

[Returns]

A tuple type data (status, pixel_format)

status	: A status code indicating the result of the operation.
pixel_format	: The type of pixel format.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraPixelFormat)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if PixelFormat register (or node), or PixelCoding and PixelSize registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.6.2. set_pixel_format

Sets the pixel format of the camera.

[Syntax]

pytelicam.CameraControl.set_pixel_format_selector(self, pixel_format)

[Parameters]

Parameters	Description
pixel_format (pytelicam.CameraPixelFormat)	The type of pixel format.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if PixelFormat register (or node), or PixelCoding and PixelSize registers (or nodes) are not implemented in the camera.

Settings can not be changed during image stream data output.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.7. TestPattern

This feature group performs control of test pattern (test image) generation function of the camera.

For details about TestPattern features, refer to instruction manual of the camera.

7.6.7.1. get_test_pattern

Gets the test pattern type of the camera.

[Syntax]

pytelicam.CameraControl.get_test_pattern(self)

[Returns]

A tuple type data (status, test_pattern)

status : A status code indicating the result of the operation.

test_pattern: The type of test pattern.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraTestPattern)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TestPattern, or TestImageSelector register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.7.2. set_test_pattern

Sets the test pattern type of the camera.

[Syntax]

pytelicam.CameraControl.set_pixel_format_selector(self, testPattern)

[Parameters]

Parameters	Description
testPattern (pytelicam.CameraTestPattern)	The type of test pattern.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if “TestPattern” or “TestImageSelector” register or node is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8. AcquisitionControl

This feature group performs control of image acquisition of the camera.

For details about AcquisitionControl features, refer to instruction manual of the camera.

7.6.8.1. get_stream_payload_size

Gets the current payload size of image stream.
The payload size is the number of bytes transferred for each image or chunk on the stream channel.

[Syntax]

pytelicam.CameraControl.get_stream_payload_size(self)

[Returns]

A tuple type data (status, payload_size)

status	: A status code indicating the result of the operation.
payload_size	: The current payload size in bytes.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.6.8.2. get_stream_enable

Gets the enable state of image streaming in the camera.

[Syntax]

pytelicam.CameraControl.get_stream_enable(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The enable state of image streaming.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method is available only for USB3 camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.3. get_acquisition_frame_count_min_max

Gets the minimum and maximum available AcquisitionFrameCount value of the camera.

AcquisitionFrameCount is the number of frames to transfer in MultiFrame/ImageBuffer mode.

[Syntax]

pytelicam.CameraControl.get_acquisition_frame_count_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of AcquisitionFrameCount.
max	: The maximum value of AcquisitionFrameCount.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrameCount register (or node) is not implemented in the camera.

7.6.8.4. get_acquisition_frame_count

Gets the AcquisitionFrameCount value of the camera.

In MultiFrame acquisition mode, the camera will stop image acquisition when transferred image count reached to the AcquisitionFrameCount value.

In ImageBufferRead mode, the camera will send the number of frames set to the AcquisitionFrameCount from the image buffer when execute_image_buffer_read() is called.

[Syntax]

pytelicam.CameraControl.get_acquisition_frame_count (self)

[Returns]

A tuple type data (status, frame_count)

status	: A status code indicating the result of the operation.
frame_count	: The value of AcquisitionFrameCount.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrameCount register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.5. set_acquisition_frame_count

Sets the AcquisitionFrameCount value of the camera.

In MultiFrame acquisition mode, the camera will stop image acquisition when transferred image count reached to the AcquisitionFrameCount value.

In ImageBufferRead mode, the camera will send the number of frames set to the AcquisitionFrameCount from the image buffer when execute_image_buffer_read() is called.

[Syntax]

pytelicam.CameraControl.set_acquisition_frame_count(self, value)

[Parameters]

Parameters	Description
value (int)	The value of AcquisitionFrameCount.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrameCount register (or node) is not implemented in the camera.

Settings cannot be changed during image stream data output.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.6. get_acquisition_frame_rate_control

Gets the frame rate setting of the camera.

[Syntax]

pytelicam.CameraControl.get_acquisition_frame_rate_control (self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The type of frame rate setting.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraAcqFrameRateCtrl)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrame register (or AcquisitionFrameControl / AcquisitionFrameRateEnable node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.7. set_acquisition_frame_rate_control

Sets the frame rate settings type of the camera.

[Syntax]

pytelicam.CameraControl.set_acquisition_frame_rate_control(self, value)

[Parameters]

Parameters	Description
value (pytelicam.CameraAcqFrameRateCtrl)	The type of frame rate settings.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrame register (or AcquisitionFrameControl / AcquisitionFrameRateEnable node) is not implemented in the camera.

When CameraAcqFrameRateCtrl.NoSpecify is set, the AcquisitionFrameRate register (or node) value may change. Please check the operation of the camera to be used before using.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.8. get_acquisition_frame_rate_min_max

Gets the minimum and maximum available frame rate value of the camera.

[Syntax]

pytelicam.CameraControl.get_acquisition_frame_rate_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of frame rate.
max	: The maximum value of frame rate.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrame register (or AcquisitionFrameControl / AcquisitionFrameRateEnable node) is not implemented in the camera.

7.6.8.9. get_acquisition_frame_rate

Gets the frame rate value of the camera.

[Syntax]

pytelicam.CameraControl.get_acquisition_frame_rate (self)

[Returns]

A tuple type data (status, frame_rate)

status	: A status code indicating the result of the operation.
frame_rate	: The value of frame rate.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrame register (or AcquisitionFrameControl / AcquisitionFrameRateEnable node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.10. set_acquisition_frame_rate

Sets the frame rate value of the camera.

[Syntax]

pytelicam.CameraControl.set_acquisition_frame_rate(self, frame_rate)

[Parameters]

Parameters	Description
frame_rate (double)	The value of frame rate.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrame register (or AcquisitionFrameControl / AcquisitionFrameRateEnable node) is not implemented in the camera.

If AcquisitionFrameRateControl value is CameraAcqFrameRateCtrl.NoSpecify, this method may fail to write data, or the camera may overwrite AcquisitionFrameRateControl value with other value immediately.

Whether it can be set during image stream data output varies depending on the camera.
For details about AcquisitionControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.11. execute_acquisition_start

Sends a AcquisitionStart command to the camera.

It is valid only when streaming is running in single-frame mode or multi-frame mode.
Execute when acquiring new images.

[Syntax]

pytelicam.CameraControl.execute_acquisition_start(self)

[Returns]

A tuple type data (status)

status	: A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AcquisitionCommand register (or node) is not implemented in the camera.

Do not execute this method except when streaming is running in single-frame mode or multi-frame mode.

7.6.8.12. get_high_framerate_mode

Gets the HighFramerateMode value of the camera.

Some color cameras have HighFramerateMode. You can improve the frame rate by using HighFramerateMode.

[Syntax]

pytelicam.CameraControl.get_high_framerate_mode(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of HighFramerateMode.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if HighFramerateMode register (or node) is not implemented in the camera.

For details about HighFramerateMode features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.8.13. set_high_framerate_mode

Sets the vertical image flip mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_high_framerate_mode(self, value)

[Parameters]

Parameters	Description
value (bool)	The value of HighFramerateMode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if HighFramerateMode register (or node) is not implemented in the camera.

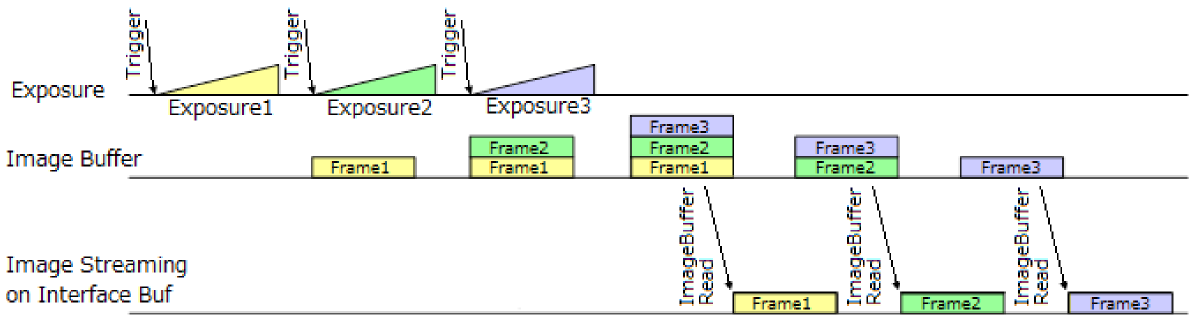
For details about HighFramerateMode features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.9. ImageBuffer

This feature group performs control of image buffer function of the camera.

In ImageBuffer mode, Camera stores images temporarily in image buffer, and read them out in arbitrary timing.
This function is typically used in Random Trigger Shutter mode.



For details about ImageBuffer features, refer to instruction manual of the camera.

7.6.9.1. get_image_buffer_mode

Gets the current image buffer mode type of the camera.

[Syntax]

pytelicam.CameraControl.get_image_buffer_mode(self)

[Returns]

A tuple type data (status, mode)

status : A status code indicating the result of the operation.

mode : The type of image buffer mode.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraImageBufferMode)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if ImageBufferMode register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.9.2. set_image_buffer_mode

Sets the current image buffer mode type of the camera.

[Syntax]

pytelicam.CameraControl.set_image_buffer_mode(self, mode)

[Parameters]

Parameters	Description
mode (pytelicam.CameraImageBufferMode)	The type of image buffer mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ImageBufferMode register (or node) is not implemented in the camera.

Settings cannot be changed during image stream data output.

7.6.9.3. get_image_buffer_frame_count

Gets the current image buffer mode type of the camera.

[Syntax]

pytelicam.CameraControl.get_image_buffer_frame_count(self)

[Returns]

A tuple type data (status, frame_count)

status	: A status code indicating the result of the operation.
frame_count	: The number of images stored in the image buffer of the camera.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ImageBufferMode register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.9.4. execute_image_buffer_read

Reads images from the image buffer of the camera.

[Syntax]

pytelicam.CameraControl.execute_image_buffer_read(self)

[Returns]

A tuple type data (status)

status	: A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ImageBufferMode register (or node) is not implemented in the camera.

"ImageBufferRead" is set in the AcquisitionMode register of the camera.

The camera will send specified frame count images for a single execute_image_buffer_read() call. User application can specify frame count by set_acquisition_frame_count().

User application can get images using CameraStream class.

For details about ImageBuffer features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10. TriggerControl

This feature group performs control of trigger function of the camera.

TriggerControl features are related to image acquisition using trigger.
This camera series provides two kinds of exposure synchronization.

1. Normal Shutter mode	: Free run operation (internal synchronization)
2. Random Trigger Shutter mode	: Synchronized with external trigger input

For details about TriggerControl features, refer to instruction manual of the camera.

7.6.10.1. get_trigger_mode

Gets the trigger mode value of the camera.

[Syntax]

pytelicam.CameraControl.get_trigger_mode(self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The value of trigger mode.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerMode register (or node) is not implemented in the camera.

If TriggerMode is OFF (Normal Shutter mode), Free-Run mode will be used as synchronization mode, and exposure time will be controlled by ExposureTime register. Value of TriggerSequence and TriggerSource registers will be ignored in this mode.

If TriggerMode is ON (Random Trigger Shutter mode), Hardware or Software Trigger mode is used as synchronization mode, and exposure time is controlled in various modes controlled by TriggerSequence and / or some other registers.

For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.2. set_trigger_mode

Sets the trigger mode value of the camera.

[Syntax]

pytelicam.CameraControl.set_trigger_mode(self, value)

[Parameters]

Parameters	Description
value (bool)	The value of trigger mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TriggerMode register (or node) is not implemented in the camera.

Whether it can be set during image stream data output varies depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.3. get_trigger_sequence

Gets the trigger sequence of random trigger shutter.

[Syntax]

pytelicam.CameraControl.get_trigger_sequence(self)

[Returns]

A tuple type data (status, sequence)

status	: A status code indicating the result of the operation.
sequence	: The type of trigger sequence.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraTriggerSequence)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if TriggerSequence register (or node), or ExposureMode and TriggerSelector registers (or nodes) are not implemented in the camera.

The registers to be accessed are different depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.4. set_trigger_sequence

Sets the trigger sequence of random trigger shutter.

[Syntax]

pytelicam.CameraControl.set_trigger_sequence(self, sequence)

[Parameters]

Parameters	Description
sequence (pytelicam.CameraTriggerSequence)	The type of trigger sequence.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TriggerSequence register (or node), or ExposureMode and TriggerSelector registers (or nodes) are not implemented in the camera.

The registers to be accessed are different depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.5. get_trigger_source

Gets the trigger source of random trigger shutter.

[Syntax]

pytelicam.CameraControl.get_trigger_source (self)

[Returns]

A tuple type data (status, sequence)

status : A status code indicating the result of the operation.

source : The type of trigger source.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraTriggerSource)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerSource register (or node) is not implemented in the camera.

7.6.10.6. set_trigger_source

Sets the trigger source of random trigger shutter.

[Syntax]

pytelicam.CameraControl.set_trigger_source(self, source)

[Parameters]

Parameters	Description
source (pytelicam.CameraTriggerSource)	The type of trigger source.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TriggerSource register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.7. get_trigger_additional_parameter_min_max

Gets the minimum and maximum available TriggerAdditionalParameter
(or AcquisitionBurstFrameCount) value of the camera.

TriggerAdditionalParameter (or AcquisitionBurstFrameCount) is the number of frames to exposure in Bulk (or FrameBurst) mode.

[Syntax]

pytelicam.CameraControl.get_trigger_additional_parameter_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of TriggerAdditionalParameter(or AcquisitionBurstFrameCount).
max	: The maximum value of TriggerAdditionalParameter(or AcquisitionBurstFrameCount).

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerAdditionalParameter or AcquisitionBurstFrameCount register (or node) is not implemented in the camera.

The register to be accessed is different depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.8. get_trigger_additional_parameter

Gets the TriggerAdditionalParameter(or AcquisitionBurstFrameCount) value of the camera.

TriggerAdditionalParameter (or AcquisitionBurstFrameCount) is the number of frames to exposure in Bulk (or FrameBurst) mode.

[Syntax]

pytelicam.CameraControl.get_trigger_additional_parameter (self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of TriggerAdditionalParameter(or AcquisitionBurstFrameCount).

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerAdditionalParameter or AcquisitionBurstFrameCount register (or node) is not implemented in the camera.

The register to be accessed is different depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.9. set_trigger_additional_parameter

Sets the TriggerAdditionalParameter(or AcquisitionBurstFrameCount) value of the camera.

TriggerAdditionalParameter (or AcquisitionBurstFrameCount) is the number of frames to exposure in Bulk (or FrameBurst) mode.

[Syntax]

pytelicam.CameraControl.set_trigger_additional_parameter(self, value)

[Parameters]

Parameters	Description
value (int)	The value of TriggerAdditionalParameter (or AcquisitionBurstFrameCount).

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if AcquisitionFrame register (or AcquisitionFrameControl / AcquisitionFrameRateEnable node) is not implemented in the camera.

If AcquisitionFrameRateControl value is CameraAcqFrameRateCtrl.NoSpecify, this method may fail to write data, or the camera may overwrite AcquisitionFrameRateControl value with other value immediately.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.10. get_trigger_delay_min_max

Gets the minimum and maximum available TriggerDelay value.

TriggerDelay is the delay to apply after the trigger signal reception before effectively activating it.

[Syntax]

pytelicam.CameraControl.get_trigger_delay_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of TriggerDelay in microseconds.
max	: The maximum value of TriggerDelay in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerDelay register (or node) is not implemented in the camera.

7.6.10.11. get_trigger_delay

Gets the TriggerDelay value of the camera.

TriggerDelay is the delay to apply after the trigger signal reception before effectively activating it.

[Syntax]

pytelicam.CameraControl.get_trigger_delay (self)

[Returns]

A tuple type data (status, microseconds)

status	: A status code indicating the result of the operation.
microseconds	: The value of TriggerDelay in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerDelay register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.12. set_trigger_delay

Sets the TriggerDelay value of the camera.

TriggerDelay is the delay to apply after the trigger signal reception before effectively activating it.

[Syntax]

pytelicam.CameraControl.set_trigger_delay(self, microseconds)

[Parameters]

Parameters	Description
microseconds (double)	The value of TriggerDelay in microseconds.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TriggerDelay register (or node) is not implemented in the camera.

7.6.10.13. execute_software_trigger

Sends a software trigger command to the camera.

[Syntax]

pytelicam.CameraControl.execute_software_trigger(self)

[Returns]

A tuple type data (status)

status	: A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if SoftwareTrigger register (or node) is not implemented in the camera.

If this method is called when the camera is not ready for accepting software trigger command, the camera may return a success status without doing anything.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.14. get_trigger_activation

Gets the trigger activation of hardware trigger.

TriggerActivation is the activation mode of the trigger.

[Syntax]

pytelicam.CameraControl.get_trigger_activation (self)

[Returns]

A tuple type data (status, sequence)

status	: A status code indicating the result of the operation.
activation	: The type of trigger activation.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraTriggerActivation)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerActivation, or LineInverterAll register (or node) is not implemented in the camera.

The value to be acquired is the setting value for the currently trigger source of random trigger shutter setting (TriggerSource). Before calling this method, set the trigger source of random trigger shutter by calling set_trigger_source() method.

If the camera has LineModeAll register, this function reads the value of the register and returns the value of the target line(trigger source).

For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.10.15. set_trigger_activation

Sets the trigger activation of hardware trigger.

TriggerActivation is the activation mode of the trigger.

[Syntax]

pytelicam.CameraControl.set_trigger_activation(self, source)

[Parameters]

Parameters	Description
activation (pytelicam.CameraTriggerActivation)	The type of trigger activation.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TriggerActivation or LineInverterAll register (or node) is not implemented in the camera.

The value to be set is the setting value for the currently trigger source of random trigger shutter setting (TriggerSource). Before calling this method, set the trigger source of random trigger shutter by calling set_trigger_source() method.

If the camera has LineModeAll register, this function reads the value of the register, changes only the value of the target line(trigger source), and sets the value in the register.
When set_line_inverter_all() or set_line_inverter() is called, the value set by this function may be changed.

For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.11. ExposureTime

This feature group performs control of exposure.

For details about ExposureTime features, refer to instruction manual of the camera.

7.6.11.1. get_exposure_time_control

Gets the exposure time control mode of the camera.

[Syntax]

pytelicam.CameraControl.get_exposure_time_control (self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The type of exposure time control mode.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraExposureTimeCtrl)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TriggerAdditionalParameter or AcquisitionBurstFrameCount register (or node) is not implemented in the camera.

The register to be accessed and the feature may be different depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.11.2. set_exposure_time_control

Sets the exposure time control mode of the camera.

[Syntax]

pytelicam.CameraControl.set_exposure_time_control(self, value)

[Parameters]

Parameters	Description
value (pytelicam.CameraExposureTimeCtrl)	The type of exposure time control mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TriggerAdditionalParameter or AcquisitionBurstFrameCount register (or node) is not implemented in the camera.

The register to be accessed and the feature may be different depending on the camera.
For details about TriggerControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.11.3. get_exposure_time_min_max

Gets the minimum and maximum available exposure time of the camera when ExposureTimeControl is set to CAM_EXPOSURE_TIME_CONTROL_MANUAL.

[Syntax]

pytelicam.CameraControl.get_exposure_time_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of exposure time in microseconds.
max	: The maximum value of exposure time in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ExposureTime register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.11.4. get_exposure_time

Gets the exposure time of the camera when ExposureTimeControl is set to CAM_EXPOSURE_TIME_CONTROL_MANUAL.

[Syntax]

pytelicam.CameraControl.get_exposure_time (self)

[Returns]

A tuple type data (status, microseconds)

status	: A status code indicating the result of the operation.
microseconds	: The value of exposure time in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ExposureTime register (or node) is not implemented in the camera.

7.6.11.5. set_exposure_time

Sets the exposure time of the camera when ExposureTimeControl is set to CAM_EXPOSURE_TIME_CONTROL_MANUAL.

[Syntax]

pytelicam.CameraControl.set_exposure_time(self, microseconds)

[Parameters]

Parameters	Description
microseconds (double)	The value of exposure time in microseconds.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ExposureTime register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.11.6. get_short_exposure_mode

Gets the ShortExposureMode value of the camera.

Some color cameras have ShortExposureMode which can set high-speed exposure time at the time of MANUAL setting.

[Syntax]

pytelicam.CameraControl.get_short_exposure_mode(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of ShortExposureMode.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ShortExposureMode register (or node) is not implemented in the camera.

For details about ShortExposureMode features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.11.7. set_short_exposure_mode

Sets the ShortExposureMode value of the camera.

Some color cameras have ShortExposureMode which can set high-speed exposure time at the time of MANUAL setting.

[Syntax]

pytelicam.CameraControl.set_short_exposure_mode(self, value)

[Parameters]

Parameters	Description
value (bool)	The value of ShortExposureMode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ShortExposureMode register (or node) is not implemented in the camera.

For details about ShortExposureMode features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12. DigitalIoControl

This feature group performs control of digital I/O ports.

For details about DigitalIoContol features, refer to instruction manual of the camera.

7.6.12.1. get_line_mode_all

Gets the input / output value for all digital I/O lines of the camera.

[Syntax]

pytelicam.CameraControl.get_line_mode_all(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of line mode.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if LineModeAll register (or node), or LineSelector and LineMode registers (or nodes) are not implemented in the camera.

If the camera does not have LineModeAll register, this method reads input / output setting values of each line, combines them, and outputs it.

For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.2. set_line_mode_all

Sets the input / output value for all digital I/O lines of the camera.

[Syntax]

pytelicam.CameraControl.set_line_mode_all(self, value)

[Parameters]

Parameters	Description
value (int)	The value of line mode for all digital I/O lines. Each bit corresponds to each I/O line.(Line0 : Bit0 , Line1 : Bit1 , Line2 : Bit2 , .) If bit value is 0, the line is input line, If bit value is 1, the line is output line. For example, If `value` is 0x06, Line0 : input, Line1 : output, Line2 : output , .)

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LineModeAll register (or node), or LineSelector and LineMode registers (or nodes) are not implemented in the camera.

If the camera does not have LineModeAll register, this method sets the values of line mode for each line.

The feature and writable lines may be different depending on the camera.
For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.3. get_line_mode

Gets the input / output value for each digital I/O line of the camera.

[Syntax]

pytelicam.CameraControl.get_line_mode(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.

[Returns]

A tuple type data (status, mode)

status : A status code indicating the result of the operation.

mode : The type of line mode.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraLineMode)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LineModeAll register (or node), or LineSelector and LineMode registers (or nodes) are not implemented in the camera.

If the camera has LineModeAll register, this method reads the value of the register, and gets the value of the target line.

For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.4. set_line_mode

Sets the input / output value for each digital I/O line of the camera.

[Syntax]

pytelicam.CameraControl.set_line_mode(self, selector, mode)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.
mode (pytelicam.CameraLineMode)	The type of line mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LineModeAll register (or node), or LineSelector and LineMode registers (or nodes) are not implemented in the camera.

If the camera has LineModeAll register, this method reads the value of the register, changes only the value of the target line, and sets the value in the register.

The feature and writable lines may be different depending on the camera.
For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.5. get_line_inverter_all

Gets the polarity value for all digital I/O lines of the camera.

[Syntax]

pytelicam.CameraControl.get_line_inverter_all(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of polarities.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LineInverterAll register (or node), or LineSelector and LineInverter registers (or nodes) are not implemented in the camera.

If the camera does not have LineInverterAll register, this method reads the polarity values of each line, combines them, and outputs it.

For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.6. set_line_inverter_all

Sets the polarity value for all digital I/O lines of the camera.

[Syntax]

pytelicam.CameraControl.set_line_inverter_all(self, value)

[Parameters]

Parameters	Description
value (int)	The value of polarities for all digital I/O lines. Each bit corresponds to each I/O line. (Line0 : Bit0 , Line1 : Bit1 , Line2 : Bit2 , .) If bit value is 0, the line is not inverted, If bit value is 1, the line is inverted. For example, If `value` is 0x06, Line0 : not inverted, Line1 : inverted, Line2 : inverted , .

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LineInverterAll register (or node), or LineSelector and LineInverter registers (or nodes) are not implemented in the camera.

If the camera does not have LineInverterAll register, this method sets the values of polarity for each line.

The feature and writable lines may be different depending on the camera.
If 1 (Inverted) is set in the bit data of the Line that can not be written, error status(`InvalidParameter`) may be returned.
For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.7. get_line_inverter

Gets the polarity value for each digital I/O line of the camera.

[Syntax]

pytelicam.CameraControl.get_line_inverter(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.

[Returns]

A tuple type data (status, invert)

status : A status code indicating the result of the operation.

invert : The value of polarity.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LineInverterAll register (or node), or LineSelector and LineInverter registers (or nodes) are not implemented in the camera.

If the camera has LineInverterAll register, this function reads the value of the register, and gets the value of the target line.

For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.8. set_line_inverter

Sets the polarity value for each digital I/O line of the camera.

[Syntax]

pytelicam.CameraControl.set_line_inverter(self, selector, invert)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.
invert (bool)	The value of polarity.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LineInverterAll register (or node), or LineSelector and LineInverter registers (or nodes) are not implemented in the camera.

If the camera has LineInverterAll register, this function reads the value of the register, changes only the value of the target line, and sets the value in the register.

The feature and writable lines may be different depending on the camera.
For details about DigitalIoContol features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.9. get_line_status_all

Gets the line status for all digital I/O lines of the camera.

[Syntax]

pytelicam.CameraControl.get_line_status_all(self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : Current status of all I/O lines.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LineStatusAll register (or node), or LineSelector and LineStatus registers (or nodes) are not implemented in the camera.

7.6.12.10. get_line_status

Gets the line status for each digital I/O line of the camera.

[Syntax]

pytelicam.CameraControl.get_line_status(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.

[Returns]

A tuple type data (status, invert)

status : A status code indicating the result of the operation.

invert : The value of line status.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LineStatusAll register (or node), or LineSelector and LineStatus registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.11. get_user_output_value_all

Gets the user output value for all user output lines in the camera.

[Syntax]

pytelicam.CameraControl.get_user_output_value_all(self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The user output value for all user output lines.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if UserOutputValueAll register (or node), or LineSelector and UserOutputValue registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.12. set_user_output_value_all

Sets the user output value for all user output lines in the camera.

[Syntax]

pytelicam.CameraControl.set_user_output_value_all(self, value)

[Parameters]

Parameters	Description
value (int)	The user output value for all user output lines. Each bit corresponds to each I/O line. (Line0 : Bit0 , Line1 : Bit1 , Line2 : Bit2 , .) If bit value is 0, the line is Low. If bit value is 1, the line is High.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if UserOutputValueAll register (or node), or LineSelector and UserOutputValue registers (or nodes) are not implemented in the camera.

The user output line is the line whose line source is set to UserOutput.
User application can use set_line_source() and set the line source to UserOutput.

The bits of the line whose line source is not set to UserOutput are ignored.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.13. get_user_output_value

Gets the user output value for each user output line of the camera.

[Syntax]

pytelicam.CameraControl.get_user_output_value(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The user output value.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if UserOutputValueAll register (or node), or LineSelector and UserOutputValue registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.14. set_user_output_value

Sets the user output value for each user output line of the camera.

[Syntax]

pytelicam.CameraControl.set_user_output_value(self, selector, value)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.
value (bool)	The user output value.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return_type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if UserOutputValueAll register (or node), or LineSelector and UserOutputValue registers (or nodes) are not implemented in the camera.

The user output line is the line whose line source is set to UserOutput.
User application can use set_line_source() and set the line source to UserOutput.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.15. get_line_source

Gets the source of the output signal.

[Syntax]

pytelicam.CameraControl.get_line_source(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.

[Returns]

A tuple type data (status, source)

status : A status code indicating the result of the operation.

source : The source type of the output signal.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraLineSource)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LineSelector and LineSource registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.12.16. set_line_source

Sets the source of the output signal.

[Syntax]

pytelicam.CameraControl.set_line_source(self, selector, source)

[Parameters]

Parameters	Description
selector (pytelicam.CameraLineSelector)	Target line.
source (pytelicam.CameraLineSource)	The source type of the output signal.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LineSelector and LineSource registers (or nodes) are not implemented in the camera.

The user application can write to LineSource register and select which internal acquisition or I/O source signal to output on the selected output line.

The source types that can be set may be different depending on the camera.
For details about DigitalIoContol features, refer to instruction manual of the camera.

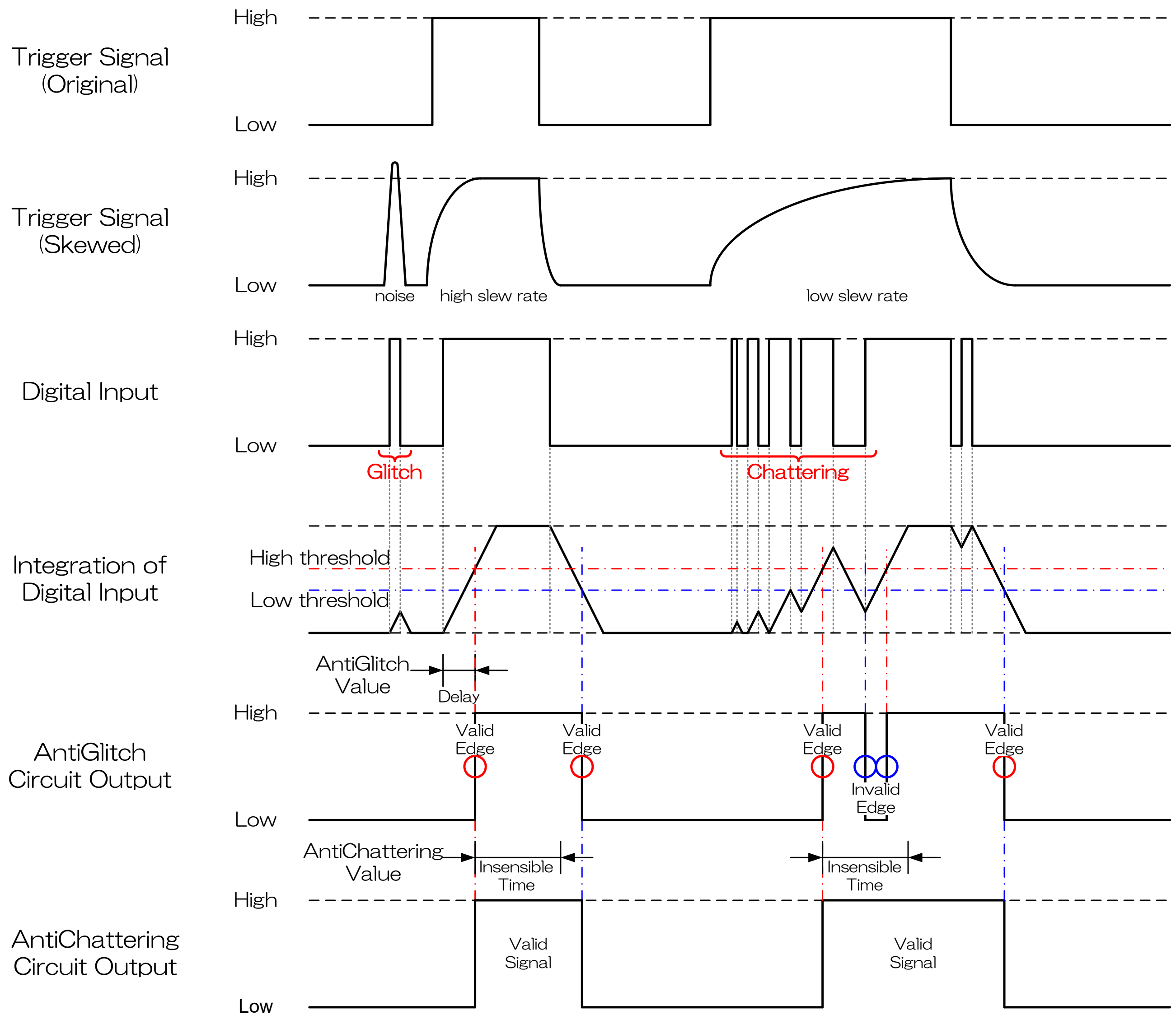
Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.13. AntiGlitch / AntiChattering

This feature group performs control of AntiGlitch and AntiChattering function of the camera.

AntiGlitch and AntiChattering functions filter noise and unstable state of the digital input (trigger signal).
AntiGlitch circuit performs the digital integration of the trigger signal.
It is effective to remove impulsive noise.

AntiChattering circuit sets the edge insensible time to avoid trigger malfunction.
It is effective to remove unstable logic state and switch-chattering.



For details about AntiGlitch and AntiChattering features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.13.1. get_anti_glitch_min_max

Gets the minimum and maximum integration time of digital input signal in the camera.

[Syntax]

pytelicam.CameraControl.get_anti_glitch_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum integration time of digital input signal, in microseconds.
max	: The maximum integration time of digital input signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AntiGlitch register (or node) is not implemented in the camera.

Please note that the unit is microsecond in this function.

7.6.13.2. get_anti_glitch

Gets the integration time of digital input signal in the camera.

[Syntax]

pytelicam.CameraControl.get_anti_glitch (self)

[Returns]

A tuple type data (status, microseconds)

status	: A status code indicating the result of the operation.
microseconds	: The integration time of digital input signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AntiGlitch register (or node) is not implemented in the camera.

Please note that the unit is microsecond in this function.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.13.3. set_anti_glitch

Sets the integration time of digital input signal in the camera.

[Syntax]

pytelicam.CameraControl.set_anti_glitch(self, microseconds)

[Parameters]

Parameters	Description
microseconds (double)	The integration time of digital input signal, in microseconds..

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if AntiGlitch register (or node) is not implemented in the camera.

Please note that the unit is microsecond in this function.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.13.4. get_anti_chattering_min_max

Gets the minimum and maximum insensible time of digital input signal in the camera.

[Syntax]

pytelicam.CameraControl.get_anti_chattering_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum insensible time of digital input signal, in microseconds.
max	: The maximum insensible time of digital input signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AntiChattering register (or node) is not implemented in the camera.

Please note that the unit is microsecond in this function.

7.6.13.5. get_anti_chattering

Gets the insensible time of digital input signal in the camera.

[Syntax]

pytelicam.CameraControl.get_anti_chattering (self)

[Returns]

A tuple type data (status, microseconds)

status	: A status code indicating the result of the operation.
microseconds	: The insensible time of digital input signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if AntiChattering register (or node) is not implemented in the camera.

Please note that the unit is microsecond in this function.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.13.6. set_anti_chattering

Sets the insensible time of digital input signal in the camera.

[Syntax]

pytelicam.CameraControl.set_anti_chattering(self, microseconds)

[Parameters]

Parameters	Description
microseconds (double)	The integration time of digital input signal, in microseconds..

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if AntiGlitch register (or node) is not implemented in the camera.

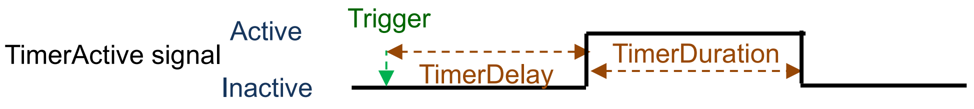
Please note that the unit is microsecond in this function.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.14. TimerControl

This feature group performs control of timer.

Timer in the camera is used to output TimerActive signal.



Target timer signal of functions in this section is fixed to Timer0Active because Current BU and BG series has only one timer.

For details about TimerControl features, refer to instruction manual of the camera.

7.6.14.1. get_timer_duration_min_max

Gets the minimum and maximum duration of Timer0Active signal.

[Syntax]

pytelicam.CameraControl.get_timer_duration_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum duration of Timer0Active signal, in microseconds.
max	: The maximum duration of Timer0Active signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TimerDuraiton register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.14.2. get_timer_duration

Gets the duration of Timer0Active signal.

[Syntax]

pytelicam.CameraControl.get_timer_duration (self)

[Returns]

A tuple type data (status, microseconds)

status	: A status code indicating the result of the operation.
microseconds	: The duration of Timer0Active signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TimerDuraiton register (or node) is not implemented in the camera.

7.6.14.3. set_timer_duration

Sets the duration of Timer0Active signal.

[Syntax]

pytelicam.CameraControl.set_timer_duration(self, microseconds)

[Parameters]

Parameters	Description
microseconds (double)	The duration of Timer0Active signal, in microseconds.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TimerDuraiton register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.14.4. get_timer_delay_min_max

Gets the minimum and maximum delay of Timer0Active signal.

[Syntax]

pytelicam.CameraControl.get_timer_delay_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum delay of Timer0Active signal, in microseconds.
max	: The maximum delay of Timer0Active signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if TimerDelay register (or node) is not implemented in the camera.

7.6.14.5. get_timer_delay

Gets the delay of Timer0Active signal.

[Syntax]

pytelicam.CameraControl.get_timer_delay (self)

[Returns]

A tuple type data (status, microseconds)

status	: A status code indicating the result of the operation.
microseconds	: The delay of Timer0Active signal, in microseconds.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if TimerDelay register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.14.6. set_timer_delay

Sets the delay of Timer0Active signal.

[Syntax]

pytelicam.CameraControl.set_timer_delay(self, microseconds)

[Parameters]

Parameters	Description
microseconds (double)	The delay of Timer0Active signal, in microseconds.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TimerDelay register (or node) is not implemented in the camera.

7.6.14.7. get_timer_trigger_source

Gets the source of Timer0Active pulse.

[Syntax]

pytelicam.CameraControl.get_timer_trigger_source(self)

[Returns]

A tuple type data (status, source)

status : A status code indicating the result of the operation.

source : The value of ShortExposureMode.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraTriggerSource)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if TimerTriggerSource register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.14.8. set_timer_trigger_source

Sets the source of Timer0Active pulse.

[Syntax]

pytelicam.CameraControl.set_timer_trigger_source(self, source)

[Parameters]

Parameters	Description
source (pytelicam.CameraTriggerSource)	The source type of Timer0Active pulse.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if TimerTriggerSource register (or node) is not implemented in the camera.

The trigger source types that can be set may be different depending on the camera.
For details about TimerControl features, refer to instruction manual of the camera

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

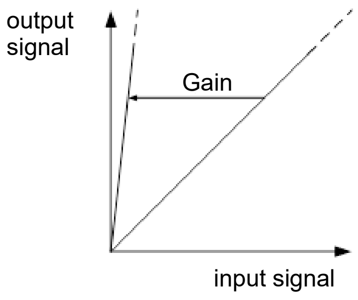
7.6.15. Gain

This feature group performs control of Gain.

Gain control adjusts an amplification factor applied to the output signal.

Gain feature adjusts manual gain.
GainAuto feature adjusts gain automatically.

For details about Gain features, refer to instruction manual of the camera.



7.6.15.1. get_gain_min_max

Gets the minimum and maximum Gain value of the camera.

[Syntax]

pytelicam.CameraControl.get_gain_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of gain, in dB or times.
max	: The maximum value of gain, in dB or times.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Gain register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.15.2. get_gain

Gets the Gain value of the camera.

[Syntax]

pytelicam.CameraControl.get_gain (self)

[Returns]

A tuple type data (status, gain)

status : A status code indicating the result of the operation.

gain : The value of gain, in dB or times.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Gain register (or node) is not implemented in the camera.

7.6.15.3. set_gain

Sets the Gain value of the camera.

[Syntax]

pytelicam.CameraControl.set_gain(self, gain)

[Parameters]

Parameters	Description
gain (double)	The value of gain, in dB or times.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if Gain register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.15.4. get_gain_auto

Gets the AGC (automatic gain control) mode of the camera.

[Syntax]

pytelicam.CameraControl.get_gain_auto(self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The type of AGC mode.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraGainAuto)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if Gain or GainAuto register (or node) is not implemented in the camera.

7.6.15.5. set_gain_auto

Sets the AGC (automatic gain control) mode of the camera.

[Syntax]

pytelicam.CameraControl.set_gain_auto(self, value)

[Parameters]

Parameters	Description
value (pytelicam.CameraGainAuto)	The type of AGC mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

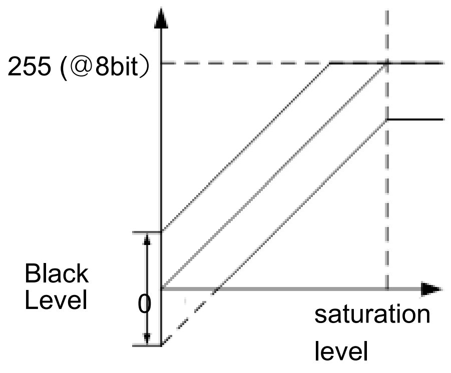
This method will return error status if Gain or GainAuto register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.16. BlackLevel

This feature group performs control of black level control.

For details about BlackLevel features, refer to instruction manual of the camera.



7.6.16.1. get_black_level_min_max

Gets the minimum and maximum black level value of the camera.

[Syntax]

pytelicam.CameraControl.get_gain_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of balck level, in %.
max	: The maximum value of balck level, in %.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if BlackLevel register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.16.2. get_black_level

Gets the black level value of the camera.

[Syntax]

pytelicam.CameraControl.get_black_level (self)

[Returns]

A tuple type data (status, black_level)

status : A status code indicating the result of the operation.

black_level : The value of balck level, in %.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if BlackLevel register (or node) is not implemented in the camera.

7.6.16.3. set_black_level

Sets the black level value of the camera.

[Syntax]

pytelicam.CameraControl.set_black_level(self, black_level)

[Parameters]

Parameters	Description
black_level (double)	The value of balck level, in %.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

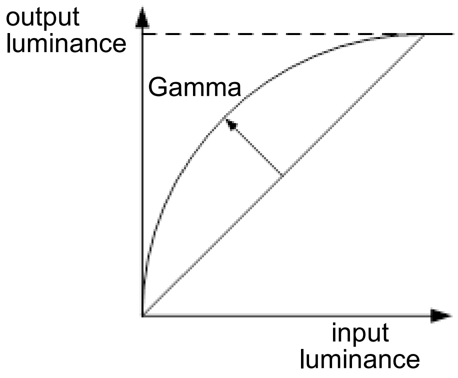
This method will return error status if BlackLevel register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.17. Gamma

This feature group performs control of gamma correction.

For details about Gamma features, refer to instruction manual of the camera.



7.6.17.1. get_gamma_min_max

Gets the minimum and maximum gamma correction value of the camera.

[Syntax]

pytelicam.CameraControl.get_gamma_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of gamma correction.
max	: The maximum value of gamma correction.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Gamma register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.17.2. get_gamma

Gets the gamma correction value of the camera.

[Syntax]

pytelicam.CameraControl.get_gamma (self)

[Returns]

A tuple type data (status, gamma)

status	: A status code indicating the result of the operation.
gamma	: The value of gamma correction.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Gamma register (or node) is not implemented in the camera.

7.6.17.3. set_gamma

Sets the gamma correction value of the camera.

[Syntax]

pytelicam.CameraControl.set_gamma(self, gamma)

[Parameters]

Parameters	Description
gamma (double)	The value of gamma correction.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if Gamma register (or node) is not implemented in the camera.

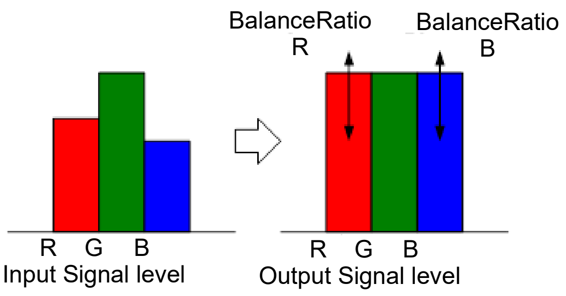
Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.18. WhiteBalance

This feature group performs control of White balance control.
It is available only in color model cameras.

Camera will control White balance, adjusting relative gain value of R/G and B/G, manually or automatically.

For details about BalanceRatio and BalanceWhiteAuto features, refer to instruction manual of the camera.



7.6.18.1. get_balance_ratio_min_max

Gets the minimum and maximum white balance gain of the camera.

[Syntax]

pytelicam.CameraControl.get_balance_ratio_min_max(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraBalanceRatioSelector)	Target color component of balance gain to control (R or B).

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of white balance gain.
max	: The maximum value of white balance gain.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if WhiteBalance* register (or BalanceRatioSelector / BalanceRatio node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.18.2. get_balance_ratio

Gets the white balance gain of the camera.

[Syntax]

pytelicam.CameraControl.get_balance_ratio (self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraBalanceRatioSelector)	Target color component of balance gain to control (R or B).

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The value of white balance gain.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if WhiteBalance* register (or BalanceRatioSelector / BalanceRatio node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.18.3. set_balance_ratio

Sets the gamma correction value of the camera.

[Syntax]

pytelicam.CameraControl.set_balance_ratio(self, selector, gamma)

[Parameters]

Parameters	Description
selector (pytelicam.CameraBalanceRatioSelector)	Target color component of balance gain to control (R or B).
value (double)	The value of white balance gain.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if WhiteBalance* register (or BalanceRatioSelector / BalanceRatio node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.18.4. get_balance_white_auto

Gets the auto white balance mode of the camera.

[Syntax]

pytelicam.CameraControl.get_balance_white_auto(self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The type of auto white balance mode.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraBalanceWhiteAuto)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if WhiteBalance* register (or BalanceWhiteAuto node) is not implemented in the camera.

7.6.18.5. set_balance_white_auto

Sets the auto white balance mode of the camera.

[Syntax]

pytelicam.CameraControl.set_balance_white_auto(self, value)

[Parameters]

Parameters	Description
value (pytelicam.CameraBalanceWhiteAuto)	The type of auto white balance mode.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success : It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

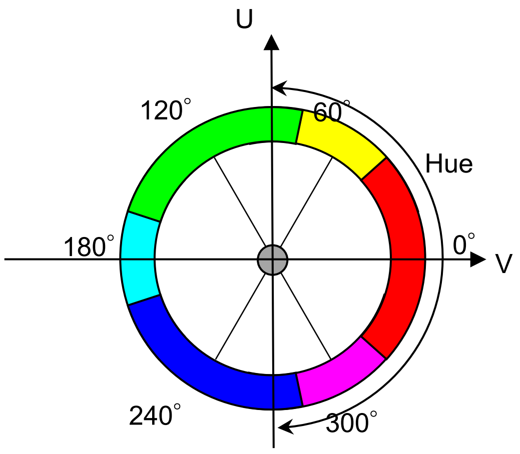
This method will return error status if Gain or GainAuto register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.19. Hue

This feature group performs control of Hue
It is available only in color model cameras.

For details about Hue features, refer to instruction manual of the camera.



7.6.19.1. get_hue_min_max

Gets the minimum and maximum hue value of the camera.

[Syntax]

pytelicam.CameraControl.get_hue_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of hue, in degrees.
max	: The maximum value of hue, in degrees.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if Hue register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.19.2. get_hue

Gets the hue value of the camera.

[Syntax]

pytelicam.CameraControl.get_hue (self)

[Returns]

A tuple type data (status, hue)

status	: A status code indicating the result of the operation.
hue	: The value of hue, in degrees.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Hue register (or node) is not implemented in the camera.

7.6.19.3. set_hue

Sets the hue value of the camera.

[Syntax]

pytelicam.CameraControl.set_hue(self, hue)

[Parameters]

Parameters	Description
hue (double)	The value of hue, in degrees.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if Hue register (or node) is not implemented in the camera.

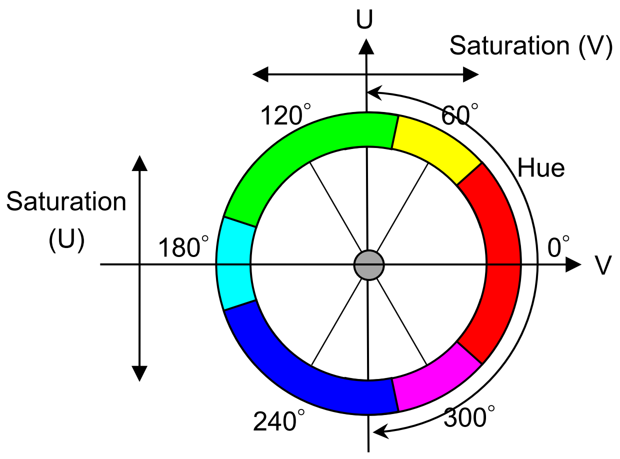
Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.20. Saturation

This feature group performs control of Saturation.
It is available only in color model cameras.

There are two types of cameras.
One is a camera that selects the target element (U / V) and sets the value in units of [times].
The other is a camera that sets the value in units of [%] common to U / V elements.

For details about Saturation features, refer to instruction manual of the camera.



7.6.20.1. get_saturation_min_max

Gets the minimum and maximum saturation value of the camera.

[Syntax]

pytelicam.CameraControl.get_saturation_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of saturation.
max	: The maximum value of saturation.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Saturation register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.20.2. get_saturation

Gets the saturation value of the camera.

[Syntax]

pytelicam.CameraControl.get_saturation(self, selector)
pytelicam.CameraControl.get_saturation(self)

[Parameters]

Parameters	Description
selector (pytelicam.CameraSaturationSelector)	Target element type of saturation.

[Returns]

A tuple type data (status, saturation)

status	: A status code indicating the result of the operation.
saturation	: The value of saturation.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Saturation register (or node) is not implemented in the camera.

There are two types of cameras.
One is a camera that selects the target element (U / V) and sets the value in units of [times].
The other is a camera that sets the value in units of [%] common to U / V elements.

If the camera is the type that sets the value in units of [%] common to U / V elements, use a method without the argument "`selector`".
If the camera is the type that selects the target element (U / V), use a method with the argument "`selector`".

For details about Saturation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.20.3. set_saturation

Sets the saturation value of the camera.

[Syntax]

pytelicam.CameraControl.set_saturation(self, selector, saturation)
pytelicam.CameraControl.set_saturation(self, saturation)

[Parameters]

Parameters	Description
selector (pytelicam.CameraSaturationSelector)	Target element type of saturation.
saturation (double)	The value of saturation.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if Saturation register (or node) is not implemented in the camera.

There are two types of cameras.
One is a camera that selects the target element (U / V) and sets the value in units of [times].
The other is a camera that sets the value in units of [%] common to U / V elements.

If the camera is the type that sets the value in units of [%] common to U / V elements, use a method without the argument "selector".
If the camera is the type that selects the target element (U / V), use a method with the argument "selector".

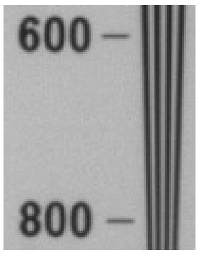
For details about Saturation features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

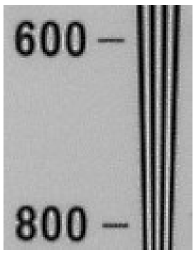
7.6.21. Sharpness

This feature group performs control of Sharpness.

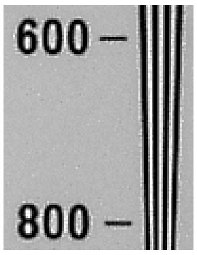
For details about Sharpness features, refer to instruction manual of the camera.



Sharpness=0



Sharpness=4



Sharpness=7

7.6.21.1. get_sharpness_min_max

Gets the minimum and maximum sharpness value of the camera.

[Syntax]

pytelicam.CameraControl.get_sharpness_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of sharpness.
max	: The maximum value of sharpness.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Sharpness register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.21.2. get_sharpness

Gets the sharpness value of the camera.

[Syntax]

pytelicam.CameraControl.get_sharpness (self)

[Returns]

A tuple type data (status, sharpness)

status	: A status code indicating the result of the operation.
sharpness	: The value of sharpness.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Sharpness register (or node) is not implemented in the camera.

7.6.21.3. set_sharpness

Sets the sharpness value of the camera.

[Syntax]

pytelicam.CameraControl.set_sharpness(self, sharpness)

[Parameters]

Parameters	Description
sharpness (int)	The value of sharpness.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if Sharpness register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.22. ColorCorrectionMatrix

This feature group performs control of Color Correction Matrix for correcting pixel color value.

It is available only in color model cameras.

The relationship between original pixel data (R, G, and B) and corrected pixel data (R', G', and B') are represented in the following formula.

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} 1 & -mask\_rg & -mask\_rb \\ -mask\_gr & 1 & -mask\_gb \\ -mask\_br & -mask\_bg & 1 \end{pmatrix} \begin{pmatrix} R & (G-R) & (B-R) \\ (R-G) & G & (B-G) \\ (R-B) & (G-B) & B \end{pmatrix}$$

$$R' = (1 - mask\_rg - mask\_rb) \times R + mask\_rg \times G + mask\_rb \times B$$

$$G' = mask\_gr \times R + (1 - mask\_gr - mask\_gb) \times G + mask\_gb \times B$$

$$B' = mask\_br \times R + mask\_bg \times G + (1 - mask\_br\,\hat{}\,mask\_bg) \times B$$

GenApi module uses selectors, “SelectorI” and “SelectorJ”, for selecting matrix element.
The correspondence of “SelectorI” and “SelectorJ” to color correction matrix element is as follows.

	SelectorJ = R	SelectorJ = G	SelectorJ = B
SelectorI = R	■	mask_rg	mask_rb
SelectorI = G	mask_gr	■	mask_gb
SelectorI = B	mask_br	mask_bg	■

The following methods use “CameraColorCorrectionMatrixSelector” type element selector, that specifies both “SelectorI” and “SelectorJ” in an enumeration member value.

For details about ColorCorrectionMatrix features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.22.1. get_color_correction_matrix_min_max

Gets the minimum and maximum coefficient value of color correction matrix in the camera.

[Syntax]

pytelicam.CameraControl.get_color_correction_matrix_min_max(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraColorCorrectionMatrixSelector)	Target column element of color correction matrix.

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum coefficient value of color correction matrix.
max	: The maximum coefficient value of color correction matrix.

[Return type]

tuple(pytelicam.CamApiStatus, double, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Masking* register, or ColorCorrectionMatrix and ColorCorrectionMatrixSelectorI and ColorCorrectionMatrixSelectorJ registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.22.2. get_color_correction_matrix

Sets the coefficient value of color correction matrix in the camera.

[Syntax]

pytelicam.CameraControl.get_color_correction_matrix (self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraColorCorrectionMatrixSelector)	Target column element of color correction matrix.

[Returns]

A tuple type data (status, matrix)

status : A status code indicating the result of the operation.

matrix : The coefficient value of color correction matrix.

[Return type]

tuple(pytelicam.CamApiStatus, double)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if Masking* register, or ColorCorrectionMatrix and ColorCorrectionMatrixSelectorI and ColorCorrectionMatrixSelectorJ registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.22.3. set_color_correction_matrix

Sets the coefficient value of color correction matrix in the camera.

[Syntax]

pytelicam.CameraControl.set_color_correction_matrix(self, selector, matrix)

[Parameters]

Parameters	Description
selector (pytelicam.CameraColorCorrectionMatrixSelector)	Target column element of color correction matrix.
matrix (double)	The coefficient value of color correction matrix.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

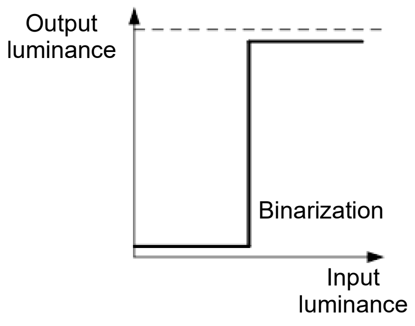
This method will return error status if Masking* register, or ColorCorrectionMatrix and ColorCorrectionMatrixSelectorI and ColorCorrectionMatrixSelectorJ registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.23. LUTControl

This feature group performs control of LUT (Look up table) feature for correcting pixel value.

For details about LUT features, refer to instruction manual of the camera.



example of LUT setting

7.6.23.1. get_lut_enable

Gets the enable state of LUT function in the camera.

[Syntax]

pytelicam.CameraControl.get_lut_enable (self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The enable state of LUT function.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if LUTEnable register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.23.2. set_lut_enable

Sets the enable state of LUT function in the camera.

[Syntax]

Pytelicam.CameraControl.set_lut_enable(self, value)

[Parameters]

Parameters	Description
value (bool)	The enable state of LUT function. If false, the LUT function is disable. If true, the LUT function is enable.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LUTEnable register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.23.3. get_lut_value

Gets the output level of LUT in the camera.

[Syntax]

pytelicam.CameraControl.get_lut_value (self, index)

[Parameters]

Parameters	Description
index (int)	The target input level of LUT. The range of the level is from 0 up to 1023, or from 0 up to 4095.

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The output level of LUT.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if LUTValueAll, or LUTIndex and LUTValue registers (or nodes) are not implemented in the camera.

The range of input and output level may be different depending on the camera.
For details about LUT features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.23.4. set_lut_value

Sets the output level of LUT in the camera.

[Syntax]

Pytelicam.CameraControl.set_lut_value(self, index, value)

[Parameters]

Parameters	Description
index (int)	The target input level of LUT. The range of the level is from 0 up to 1023, or from 0 up to 4095.
value (bool)	The output level of LUT. The level should be from 0 up to 1023, or from 0 up to 4095.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if LUTValueAll, or LUTIndex and LUTValue registers (or nodes) are not implemented in the camera.

The range of input and output level may be different depending on the camera.
For details about LUT features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.24. UserSetControl

This feature group performs control of UserSet.

User application can save current settings of the camera to non-volatile user memory, load saved settings to camera registers using this control.

For details about UserSetControl features, refer to instruction manual of the camera.

7.6.24.1. execute_user_set_load

Loads parameters saved in non-volatile memory (user memory) of the camera and save them to registers in the camera.

[Syntax]

pytelicam.CameraControl.execute_user_set_load(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraUserSetSelector)	The type of user memory channel to load.

[Returns]

A tuple type data (status)

status : A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if UserSetSelector and UserSetCommand, or UserSetLoad registers (or nodes) are not implemented in the camera.

Refer to instruction manual of the camera, to check target registers for loading from user memory.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.24.2. execute_user_set_save

Saves the current parameters to non-volatile memory (user memory) of the camera.

[Syntax]

pytelicam.CameraControl.execute_user_set_save(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraUserSetSelector)	The type of user memory channel to save. CameraUserSetSelector.Default is not available for saving target.

[Returns]

A tuple type data (status)

status : A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if UserSetSelector and UserSetCommand, or UserSetSave registers (or nodes) are not implemented in the camera.

Refer to instruction manual of the camera, to check target registers for saving data to user memory.
The difference between UserSetSave and UserSetQuickSave is described in instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.24.3. execute_user_set_quick_save

Saves current parameters to volatile memory of the camera.

[Syntax]

pytelicam.CameraControl.execute_user_set_quick_save(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraUserSetSelector)	The type of user memory channel to save. CameraUserSetSelector.Default is not available for saving target.

[Returns]

A tuple type data (status)

status : A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if UserSetSelector and UserSetCommand, or UserSetQuickSave registers (or nodes) are not implemented in the camera.

UserSetQuickSave can reduce the overhead time of UserSetSave greatly because it stored to internal RAM.
You can also save UserSets to non-volatile memory(Serial Flash) if necessary by UserSetSave. (backward compatible)

Refer to instruction manual of the camera, to check target registers for saving parameters to user memory. The difference between UserSetSave and UserSetQuickSave is described in instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.24.4. get_user_set_default

Gets the user memory channel to be loaded when the camera is powered on.

[Syntax]

pytelicam.CameraControl.get_user_set_default (self)

[Returns]

A tuple type data (status, selector)

status	: A status code indicating the result of the operation.
selector	: The typelof user memory channel.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraUserSetSelector)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if UserSetDefault register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.24.5. set_user_set_default

Sets the user memory channel to be loaded when the camera is powered on.

[Syntax]

pytelicam.CameraControl.set_user_set_default(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraUserSetSelector)	The type of user memory channel.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if UserSetDefault register (or node) is not implemented in the camera.

When you write a value to UserSetDefault register, some cameras are saved in non-volatile memory, but some cameras are not saved. (Depending on the firmware version of the camera.) If you use the camera that is not saved in non-volatile memory, please use execute_user_set_save_and_set_default() function.

For details about UserSetControl features, refer to instruction manual of the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.24.6. execute_user_set_save_and_set_default

Saves the current parameters to non-volatile memory (user memory) of the camera, and sets the user memory channel to be loaded when the camera is powered on.

[Syntax]

pytelicam.CameraControl.execute_user_set_save(self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraUserSetSelector)	The type of user memory channel to save and set.

[Returns]

A tuple type data (status)

status : A status code indicating the result of the operation.

[Return type]

tuple(pytelicam.CamApiStatus)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if UserSetDefault and UserSetCommand, or UserSetDefault and UserSetSave registers (or nodes) are not implemented in the camera.

When other than CameraUserSetSelector.Default value is specified to selector, current parameters will be written to specified channel of non-volatile memory (user memory) in the camera. The camera will start up with parameters saved in the specified channed from the next power-on.

When CameraUserSetSelector.Default is specified to selector, current parameters will be discarded and initial factory settings will be loaded immediately. The camera will start up with initial factory settings from the next power-on.

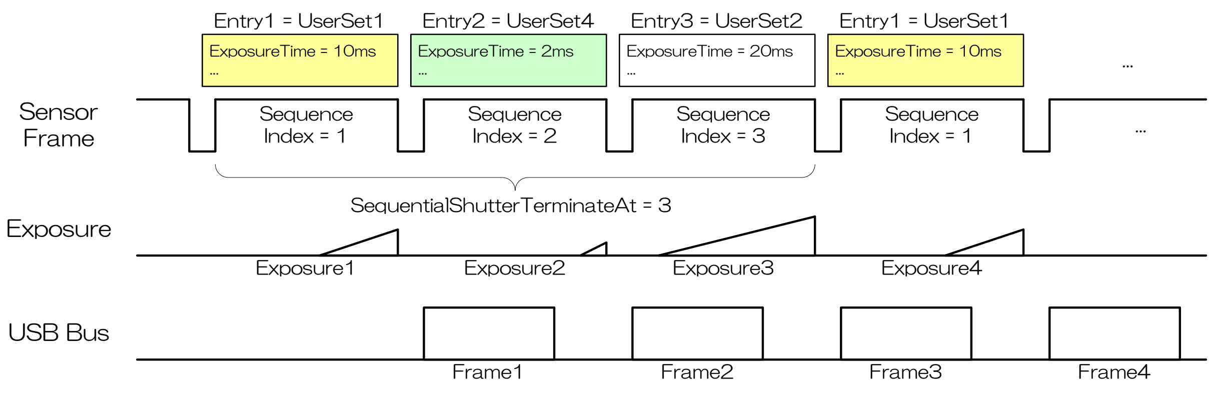
Refer to instruction manual of the camera, to check target registers for saving data to user memory.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25. SequentialShutterControl

This feature group performs control of Sequential shutter.

Sequential Shutter function performs sequential capturing with applying the settings of UserSet that have been made entry in advance.



For details about SequentialShutter features, refer to instruction manual of the camera.

7.6.25.1. get_sequential_shutter_enable

Gets the enable state of Sequential Shutter function in the camera.

[Syntax]

pytelicam.CameraControl.get_sequential_shutter_enable (self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The enable state of Sequential Shutter function.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterEnable register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25.2. set_sequential_shutter_enable

Sets the enable state of Sequential Shutter function in the camera.

[Syntax]

pytelicam.CameraControl.set_sequential_shutter_enable(self, value)

[Parameters]

Parameters	Description
value (bool)	The enable state of Sequential Shutter function If false, the Sequential Shutter function is disable. If true, the Sequential Shutter function is enable.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterEnable register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25.3. get_sequential_shutter_terminate_at_min_max

Gets the minimum and maximum number of index to repeat the sequence.

[Syntax]

pytelicam.CameraControl.get_sequential_shutter_terminate_at_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum number of index to repeat the sequence.
max	: The maximum number of index to repeat the sequence.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterTerminateAt register (or node) is not implemented in the camera.

7.6.25.4. get_sequential_shutter_terminate_at

Gets the number of index to repeat the sequence.

[Syntax]

pytelicam.CameraControl.get_sequential_shutter_terminate_at (self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The number of index to repeat the sequence.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterTerminateAt register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25.5. set_sequential_shutter_terminate_at

Sets the number of index to repeat the sequence.

[Syntax]

pytelicam.CameraControl.set_sequential_shutter_terminate_at(self, value)

[Parameters]

Parameters	Description
value (int)	The number of index.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterTerminateAt register (or node) is not implemented in the camera.

7.6.25.6. get_sequential_shutter_index_min_max

Gets the minimum and maximum value of sequence number of sequential shutter function.

[Syntax]

pytelicam.CameraControl.get_sequential_shutter_index_min_max(self)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of sequence number.
max	: The maximum value of sequence number.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterSequenceTable or SequentialShutterIndex register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25.7. get_sequential_shutter_entry_min_max

Gets the minimum and maximum value of UserSet number of sequential shutter function.

[Syntax]

pytelicam.CameraControl.get_sequential_shutter_entry_min_max(self, index)

[Parameters]

Parameters	Description
index (int)	Target sequence number.(SequentialShutterIndex)

[Returns]

A tuple type data (status, min, max)

status	: A status code indicating the result of the operation.
min	: The minimum value of UserSet number.
max	: The maximum value of UserSet number.

[Return type]

tuple(pytelicam.CamApiStatus, int, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterSequenceTable or SequentialShutterEntry register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25.8. get_sequential_shutter_entry

Gets the value of UserSet number of sequential shutter function.

[Syntax]

pytelicam.CameraControl.get_sequential_shutter_entry (self)

[Parameters]

Parameters	Description
index (int)	Target sequence number.(SequentialShutterIndex)

[Returns]

A tuple type data (status, entry)

status : A status code indicating the result of the operation.

entry : The value of UserSet number.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterSequenceTable or SequentialShutterEntry register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.25.9. set_sequential_shutter_entry

Sets the value of UserSet number of sequential shutter function.

[Syntax]

Pytelicam.CameraControl.set_sequential_shutter_entry(self, index, entry)

[Parameters]

Parameters	Description
index (int)	Target sequence number. (SequentialShutterIndex)
entry (int)	The value of UserSet number.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if SequentialShutterSequenceTable or SequentialShutterEntry register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.26. UserDefinedName

This feature group controls user-defined name ("UserDefinedName" or "DeviceUserID" register) of the camera.

User application can set any string to "UserDefinedName" or "DeviceUserID" register for identifying it. The register value is stored in non0volatile memory of the camera,

For details about UserDefinedName features, refer to instruction manual of the camera.

7.6.26.1. get_user_defined_name

Gets the user-defined name of the camera.

[Syntax]

pytelicam.CameraControl.get_user_defined_name (self)

[Returns]

A tuple type data (status, name)

status : A status code indicating the result of the operation.

name : The string of the user-defined name.

[Return type]

tuple(pytelicam.CamApiStatus, string)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if UserDefinedName or DeviceUserID register (or node) is not implemented in the camera.

If user-defined name registered in the camera is not "NULL terminated" , the last character may be ignored.

When user application sets new user-defined name, the old user-defined name may remain until "get_num_of_cameras()" of `CameraSystem` object is called.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.26.2. set_user_defined_name

Sets the user-defined name of the camera.

[Syntax]

Pytelicam.CameraControl.set_user_defined_name(self, name)

[Parameters]

Parameters	Description
name (string)	The string of the user-defined name.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if UserDefinedName or DeviceUserID register (or node) is not implemented in the camera.

The length of user-defined name register in GigE camera is 16 bytes, that in USB3 camera is 64 bytes.

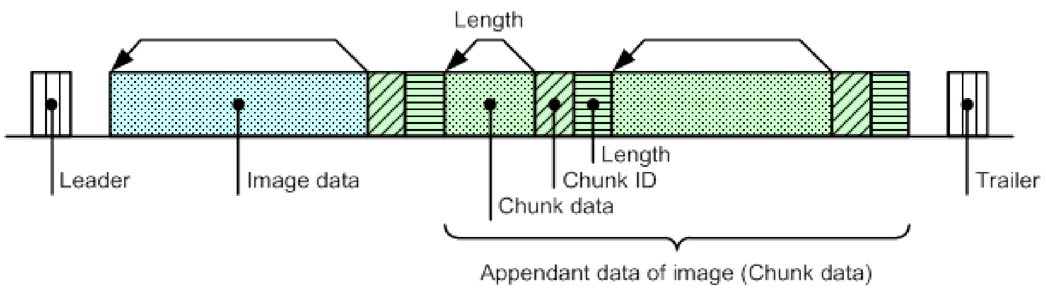
This method sets the user-defined name as NULL terminated string, which means that the length of the name, that user can specify as user-defined name, is register length minus 1.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.27. Chunk

This feature group performs control of Chunk feature.

Chunk data means tagged blocks of data.
The tags allow a chunk parser to dissect the data payload into its elements and to identify the content.
The length of a frame varies depending on the number of activated chunks.



For details about Chunk features, refer to instruction manual of the camera.

When using GenICam to acquire chunk data, it is necessary to attach the buffer storing the payload data to GenICam chunk adaptor.
For details, refer to the description of the chunk_attach_buffer() method.

7.6.27.1. get_chunk_mode_active

Gets the activation state of Chunk function in the camera.

[Syntax]

pytelicam.CameraControl.get_chunk_mode_active (self)

[Returns]

A tuple type data (status, value)

status : A status code indicating the result of the operation.

value : The activation state of Chunk function.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ChunkModeActive register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.27.2. set_chunk_mode_active

Sets the activation value of Chunk function in the camera.

[Syntax]

Pytelicam.CameraControl.set_chunk_mode_active(self, value)

[Parameters]

Parameters	Description
value (bool)	The activation value of Chunk function. If false, the Chunk function is inactive. If true, the Chunk function is active.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ChunkModeActive register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.27.3. get_chunk_enable

Gets the enable state of Chunk data in the camera.

[Syntax]

pytelicam.CameraControl.get_chunk_enable (self, selector)

[Parameters]

Parameters	Description
selector (pytelicam.CameraChunkSelector)	Target element of Chunk data.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The enable state of the selected Chunk data.

[Return_type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ChunkEnableOf*, or ChunkSelector and ChunkEnable registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.27.4. set_chunk_enable

Sets the enable state of Chunk data in the camera.

[Syntax]

Pytelicam.CameraControl.set_chunk_enable(self, selector, value)

[Parameters]

Parameters	Description
selector (pytelicam.CameraChunkSelector)	Target element of Chunk data.
value (bool)	The activation value of Chunk function. If false, the Chunk function is inactive. If true, the Chunk function is active.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ChunkEnableOf*, or ChunkSelector and ChunkEnable registers (or nodes) are not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.27.5. get_chunk_user_area_length

Gets the length of ChunkUserAreaTable in the camera.

[Syntax]

pytelicam.CameraControl.get_chunk_user_area_length (self)

[Returns]

A tuple type data (status, length)

status	: A status code indicating the result of the operation.
length	: A variable that receives the length of ChunkUserAreaTable, in byte(s).

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ChunkUserArea or ChunkUserAreaTable register (or node) is not implemented in the camera.

7.6.27.6. get_chunk_user_area_table

Gets the data of ChunkUserAreaTable in the camera.

[Syntax]

pytelicam.CameraControl.get_chunk_user_area_table (self)

[Returns]

A tuple type data (status, data)

status	: A status code indicating the result of the operation.
data	: A reference to a string that receives the data of ChunkUserAreaTable.

[Return type]

tuple(pytelicam.CamApiStatus, string)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This method will return error status if ChunkUserArea or ChunkUserAreaTable register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.27.7. set_chunk_user_area_table

Sets the data of ChunkUserAreaTable in the camera.

[Syntax]

Pytelicam.CameraControl.set_chunk_user_area_table(self, data)

[Parameters]

Parameters	Description
data (string)	The string to be set.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if ChunkUserArea or ChunkUserAreaTable register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.28. FrameSynchronization

This feature group performs control of frame synchronization method of the camera.

For details about FrameSynchronization features, refer to instruction manual of the camera.

7.6.28.1. get_frame_synchronization

Gets the camera frame synchronization method.

[Syntax]

pytelicam.CameraControl.get_frame_synchronization (self)

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The type of the camera frame synchronization method.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.CameraFrameSynchronization)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This method will return error status if FrameSynchronization register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.6.28.2. set_frame_synchronization

Sets the activation value of Chunk function in the camera.

[Syntax]

Pytelicam.CameraControl.set_frame_synchroniazation(self, value)

[Parameters]

Parameters	Description
value (pytelicam.CameraFrameSynchronization)	The type of the camera frame synchronization method.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This method will return error status if FrameSynchronization register (or node) is not implemented in the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7. GenApiWrapper class (Subset for Camera Control)

This class is a class for wrapping GenICam GenApi library, which allow user application to access various information of camera register easier.

Refer to https://www.emva.org/standards-technology/genicam/ about detail information of GenICam.

The object of this class is created and managed as a member of CameraDevice class. User application does not need to create or destroy this object.

[Syntax]

pytelicam.GenApiWrapper

[Functions]

	Name	Description
		INode node
●	get_node_type	Gets the node type of the node.
●	get_access_mode	Gets the access mode of the node.
●	get_visibility	Gets the recommended visibility of the node.
●	get_caching_mode	Gets the caching mode of the node.
●	get_description	Gets a long description of the node.
●	get_tool_tip	Gets a short description of the node.
●	get_representation	Gets the recommended representation of the node.
●	get_unit	Gets the physical unit name of the node.
		ICategory node
●	get_num_of_features	Gets the number of features contained in the ICategory node.
●	get_feature_name	Gets the name of the feature contained in the ICategory node.
●	get_available_feature_names	Gets a tuple of available feature names contained in the ICategory node.
		Common functions
●	get_feature_value	Gets the current value of the IInteger, IFloat, IBoolean, IEnumeration, and IString node as a string.
●	set_feature_value	Sets the value of the IInteger, IFloat, IBoolean, IEnumeration, and IString node as a string.
		IInteger node
●	get_int_min	Gets the current minimum value of an IInteger node.
●	get_int_max	Gets the current maximum value of an IInteger node.
●	get_int_inc	Gets the current increment value of an IInteger node.
●	get_int_value	Gets the current value of an IInteger node.
●	set_int_value	Sets the value of an IInteger node.
		IFloat node
●	get_float_min	Gets the current minimum value of an IFloat node.
●	get_float_max	Gets the current maximum value of an IFloat node.
●	get_float_has_inc	Gets whether the specified IFloat node has a constant increment value.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	Name	Description
◆	get_float_inc	Gets the current increment value of an IFloat node.
◆	get_float_display_notation	Gets the way that the float value of the specified IFloat node should be converted to a string.
◆	get_float_display_precision	Get the precision to be used when converting the float value of the specified IFloat node to a string.
◆	get_float_value	Gets the current value of an IFloat node.
◆	set_float_value	Sets the value of an IFloat node.
IBoolean node		
◆	get_bool_value	Gets the current value of an IBoolean node.
◆	set_bool_value	Sets the value of an IBoolean node.
IEnumeration、IEnumEntry node		
◆	get_enum_int_value	Gets the current value of an IEnumeration node as an integer value.
◆	set_enum_int_value	Sets the value of an IEnumeration node as an integer value.
◆	get_enum_str_value	Gets the current value of an IEnumeration node as a string.
◆	set_enum_str_value	Sets the value of an IEnumeration node as a string.
◆	get_available_enum_entry_names	Gets a tuple of available entry names contained in the IEnumeration node.
◆	get_num_of_enum_entries	Gets the number of entries contained in the IEnumeration node.
◆	get_enum_entry_access_mode	Gets the access mode of the IEnumEntry node contained in the IEnumeration node.
◆	get_enum_entry_int_value	Gets the value of the IEnumEntry node contained in the IEnumeration node as an integer value.
◆	get_enum_entry_str_value	Gets the value of the IEnumEntry node contained in the IEnumeration node as a string.
ICommand node		
◆	execute_command	Executes the command of ICommand node.
◆	get_command_is_done	Checks that the command execution of the ICommand node has finished or not.
IString node		
◆	get_str_value	Gets the current value of an IString node.
◆	set_str_value	Sets the value of an IString node.
Ohters		
◆	get_access_module	Gets the setting of the module (xml file) accessed by GenICam functions.
◆	set_access_module	Sets the setting of the module (xml file) accessed by GenICam functions.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1. INode node functions

7.7.1.1. get_node_type

This function gets the node type of the node.

[Syntax]

pytelicam.GenApiWrapper.get_node_type(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, node_type)

status	: A status code indicating the result of the operation.
node_type	: The node type of the node.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeType)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, node_type = cam_device.genapi.get_node_type('Width')
print('status={0} , node_type={1}'.format(status, node_type))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.2. get_access_mode

This function gets the access mode of the node.

[Syntax]

pytelicam.GenApiWrapper.get_access_mode(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, access_mode)

status	: A status code indicating the result of the operation.
access_mode	: The access mode of the node.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeAccessMode)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, access_mode = cam_device.genapi.get_access_mode('Width')
print('status={0} , access_mode={1}'.format(status, access_mode))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.3. get_visibility

This function gets the recommended visibility of the node.

[Syntax]

pytelicam.GenApiWrapper.get_visibility(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, visibility)

status	: A status code indicating the result of the operation.
visibility	: The recommended visibility of the node.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeVisibility)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, visibility = cam_device.genapi.get_visibility('Width')
print('status={0} , visibility={1}'.format(status, visibility))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.4. get_caching_mode

This function gets the caching mode of the node.

[Syntax]

pytelicam.GenApiWrapper.get_caching_mode(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, caching_mode)

status	: A status code indicating the result of the operation.
caching_mode	: The caching mode of the node.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeCachingMode)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, caching_mode = cam_device.genapi.get_caching_mode('Width')
print('status={0} , caching_mode={1}'.format(status, caching_mode))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.5. get_description

This function gets a long description of the node.

[Syntax]

pytelicam.GenApiWrapper.get_description(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, description)

status	: A status code indicating the result of the operation.
description	: A long description of the node.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, description = cam_device.genapi.get_description('Width')
print('status={0} , description={1}'.format(status, description))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.6. get_tool_tip

This function gets a short description of the node.

[Syntax]

pytelicam.GenApiWrapper.get_tool_tip(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, tool_tip)

status	: A status code indicating the result of the operation.
tool_tip	: A short description of the node.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, tool_tip = cam_device.genapi.get_tool_tip('Width')
print('status={0} , tool_tip={1}'.format(status, tool_tip))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.7. get_representation

This function gets the recommended representation of the node.

[Syntax]

pytelicam.GenApiWrapper.get_representation(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, representation)

status	: A status code indicating the result of the operation.
representation	: The recommended representation of the node.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeRepresentation)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, representation = cam_device.genapi.get_representation('Width')
print('status={0} , representation={1}'.format(status, representation))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.1.8. get_unit

This function gets the physical unit name of the node.

[Syntax]

pytelicam.GenApiWrapper.get_unit(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, unit)

status	: A status code indicating the result of the operation.
unit	: The physical unit name of the node.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

An example is as follows:

```
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('status={0} , unit={1}'.format(status, unit))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.2. ICategory node functions

7.7.2.1. get_num_of_features

This function gets the number of features contained in the ICategory node.

[Syntax]

pytelicam.GenApiWrapper.get_num_of_features(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name of the ICategory node.

[Returns]

A tuple type data (status, num)

status	: A status code indicating the result of the operation.
num	: The number of features.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This function is only for ICategory nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, feature_num = cam_device.genapi.get_num_of_features('Root')
print('status={0} , num={1}'.format(status, feature_num))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.2.2. get_feature_name

This function gets the name of the feature contained in the ICategory node.

[Syntax]

pytelicam.GenApiWrapper.get_feature_name(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name of the ICategory node.
index (int)	Index of features in the ICategory node. (Integer value greater than or equal to 0.)

[Returns]

A tuple type data (status, feature_name)

status	: A status code indicating the result of the operation.
feature_name	: The name of the feature.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This function is only for ICategory nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, node_name = cam_device.genapi.get_feature_name('Root', 0)
print('status={0} , node_name ={1}'.format(status, node_name))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.2.3. get_available_feature_names

This function gets a tuple of available feature names contained in the ICategory node.

[Syntax]

```
pytelicam.GenApiWrapper.get_available_feature_names(
                                        self,
                                        node_name,
                                        enable_beginner=True,
                                        enable_expert=True,
                                        enable_grur=True,
                                        enable_invisible=True)
```

[Parameters]

Parameters	Description
node_name (str)	Node name of the ICategory node.
enable_beginner (bool)	Sets the enabled state of nodes whose visibility is set to "Beginner". If set to “True”, these node names are added to the tuple.
enable_expert (bool)	Sets the enabled state of nodes whose visibility is set to "Expert". If set to “True”, these node names are added to the tuple.
enable_guru (bool)	Sets the enabled state of nodes whose visibility is set to "Guru". If set to “True”, these node names are added to the tuple.
enable_invisible (bool)	Sets the enabled state of nodes whose visibility is set to "Invisible". If set to “True”, these node names are added to the tuple.

[Returns]

A tuple type data of feature node names.

[Return type]

tuple(str, …)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

[Remarks]

This function is only for ICategory nodes.
If node type of the target node is different node type, this function will return an empty tuple.

The names of unavailable feature nodes are not added to the tuple.

An example is as follows:

```
feature_names = cam_device.genapi.get_available_feature_names('Root')
print(feature_names)
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.3. Common functions

7.7.3.1. get_feature_value

This function gets the current value of the IInteger, IFloat, IBoolean, IEnumeration, and IString node as a string.

[Syntax]

pytelicam.GenApiWrapper.get_feature_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value of the node.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IInteger, IFloat, IBoolean, IEnumeration and IString nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, trigger_mode = cam_device.genapi.get_feature_value('TriggerMode')
print('status={0} , trigger_mode={1}'.format(status, trigger_mode))

status, gain_value = cam_device.genapi.get_feature_value('Gain')
print('status={0} , gain_value={1}'.format(status, gain_value))

status, user_defined_name = cam_device.genapi.get_feature_value('UserDefinedName')
print('status={0} , user_defined_name={1}'.format(status, user_defined_name))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.3.2. set_feature_value

This function sets the value of the IInteger, IFloat, IBoolean, IEnumeration, and IString node as a string.

[Syntax]

pytelicam.GenApiWrapper.set_feature_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	Node name.
value (str)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for IInteger, IFloat, IBoolean, IEnumeration and IString nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_feature_value('TriggerMode', 'Off')
print('status={0}'.format(status))

status = cam_device.genapi.set_feature_value('Gain', '1.0')
print('status={0}'.format(status))

status = cam_device.genapi.set_feature_value('UserDefinedName', 'Test')
print('status={0}'.format(status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.4. IInteger node functions

7.7.4.1. get_int_min

This function gets the current minimum value of an IInteger node.

[Syntax]

pytelicam.GenApiWrapper.get_int_min(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IInteger node.

[Returns]

A tuple type data (status, min_value)

status	: A status code indicating the result of the operation.
min_value	: The current minimum value.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This function is only for IInteger nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, width_min = cam_device.genapi.get_int_min('Width')
print('gstatus={0} , width_min={1}'.format(status, width_min))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.4.2. get_int_max

This function gets the current maximum value of an IInteger node.

[Syntax]

pytelicam.GenApiWrapper.get_int_max(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IInteger node.

[Returns]

A tuple type data (status, max_value)

status	: A status code indicating the result of the operation.
max_value	: The current maximum value.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IInteger nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, width_max = cam_device.genapi.get_int_max('Width')
print('status={0} , width_max={1}'.format(status, width_max))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.4.3. get_int_inc

This function gets the current increment value of an IInteger node.

[Syntax]

pytelicam.GenApiWrapper.get_int_inc(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	Node name.

[Returns]

A tuple type data (status, inc_value)

status	: A status code indicating the result of the operation.
inc_value	: The current maximum value.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This function is only for IInteger nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, width_inc = cam_device.genapi.get_int_inc('Width')
print('status={0} , width_inc={1}'.format(status, width_inc))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.4.4. get_int_value

This function gets the current value of an IInteger node.

[Syntax]

pytelicam.GenApiWrapper.get_int_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IInteger node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IInteger nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, width = cam_device.genapi.get_int_value('Width')
print('status={0} , width={1}'.format(status, width))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.4.5. set_int_value

This function sets the value of an IInteger node.

[Syntax]

pytelicam.GenApiWrapper.set_int_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	The name of IInteger node.
value (int)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for IInteger nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_int_value('Width', 320)
print('status={0}'.format(status))
if status == pytelicam.CamApiStatus.GenICamError :
    error_message = cam_device.get_last_genicam_error()
    print('error_message ={0}'.format(error_message))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5. IFloat node functions

7.7.5.1. get_float_min

This function gets the current minimum value of an IFloat node.

[Syntax]

pytelicam.GenApiWrapper.get_float_min(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, min_value)

status	: A status code indicating the result of the operation.
min_value	: The current minimum value.

[Return type]

tuple(pytelicam.CamApiStatus, float)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, exposure_min = cam_device.genapi.get_float_min('ExposureTime')
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('exposure_min={0}[{1}]'.format(exposure_min, unit))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.2. get_float_max

This function gets the current maximum value of an IFloat node.

[Syntax]

pytelicam.GenApiWrapper.get_float_max(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, max_value)

status	: A status code indicating the result of the operation.
max_value	: The current maximum value.

[Return type]

tuple(pytelicam.CamApiStatus, float)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, exposure_max = cam_device.genapi.get_float_max('ExposureTime')
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('exposure_max={0}[{1}]'.format(exposure_max, unit))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.3. get_float_has_inc

This function gets whether the specified IFloat node has a constant increment value.

[Syntax]

pytelicam.GenApiWrapper.get_float_has_inc(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, has_inc)

status	: A status code indicating the result of the operation.
has_inc	: A flag that indicates whether the node has a constant increment value. If True, the node has a constant increment. If False, the node has not a constant increment.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, has_inc = cam_device.genapi.get_float_has_inc('ExposureTime')
print('has_inc={0}'.format(has_inc))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.4. get_float_inc

This function gets the current increment value of an IFloat node.

[Syntax]

pytelicam.GenApiWrapper.get_float_inc(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, inc_value)

status	: A status code indicating the result of the operation.
inc_value	: The current increment value.

[Return type]

tuple(pytelicam.CamApiStatus, float)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.5. get_float_display_notation

This function gets the way that the float value of the specified IFloat node should be converted to a string.

[Syntax]

pytelicam.GenApiWrapper.get_float_display_notation(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, display_notation)

status	: A status code indicating the result of the operation.
display_notation	: The way that the float value of the specified IFloat node should be converted to a string.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeDisplayNotation)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, display_notation = \
    cam_device.genapi.get_float_display_notation('ExposureTime')
print(' display_notation={0}'.format(display_notation))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.6. get_float_display_precision

This function gets the precision to be used when converting the float value of the specified IFloat node to a string.

[Syntax]

pytelicam.GenApiWrapper.get_float_display_precision(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, display_precision)

status	: A status code indicating the result of the operation.
display_precision	: The precision to be used when converting the float value of the specified IFloat node to a string.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, display_precision = \
    cam_device.genapi.get_float_display_precision('ExposureTime')
print(' display_precision={0}'.format(display_precision))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.7. get_float_value

This function gets the current value of an IFloat node.

[Syntax]

pytelicam.GenApiWrapper.get_float_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value.

[Return type]

tuple(pytelicam.CamApiStatus, float)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, exposure_time = cam_device.genapi.get_float_value('ExposureTime')
status, unit = cam_device.genapi.get_unit('ExposureTime')
print('exposure_time={0}[{1}]'.format(exposure_time, unit))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.5.8. set_float_value

This function sets the value of an IFloat node.

[Syntax]

pytelicam.GenApiWrapper.set_float_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	The name of IFloat node.
value (float)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for IFloat nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
# Set ExposureTime to 100[ms]
status = cam_device.genapi.set_float_value('ExposureTime', 100000.0)
print('status={0}'.format(status))
if status == pytelicam.CamApiStatus.GenICamError :
    error_message = cam_device.get_last_genicam_error()
    print('error_message ={0}'.format(error_message))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.6. IBoolean node functions

7.7.6.1. get_bool_value

This function gets the current value of an IBoolean node.

[Syntax]

pytelicam.GenApiWrapper.get_bool_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IBoolean node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value.

[Return type]

tuple(pytelicam.CamApiStatus, bool)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IBoolean nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, lut_enable = cam_device.genapi.get_bool_value('LUTEnable')
print('status={0} , lut_enable={1}'.format(status, lut_enable))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.6.2. set_bool_value

This function sets the value of an IBoolean node.

[Syntax]

pytelicam.GenApiWrapper.set_bool_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	The name of IBoolean node.
value (bool)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for IBoolean nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_bool_value('LUTEnable', True)
print('status={0}'.format(status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7. IEnumeration node functions

7.7.7.1. get_enum_int_value

This function gets the current value of an IEnumeration node as an integer value.

[Syntax]

pytelicam.GenApiWrapper.get_enum_int_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, trigger_mode = cam_device.genapi.get_enum_int_value('TriggerMode')
print('status={0} , trigger_mode={1}'.format(status, trigger_mode))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.2. set_enum_int_value

This function sets the value of an IEnumeration node as an integer value.

[Syntax]

pytelicam.GenApiWrapper.set_enum_int_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.
value (int)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_enum_int_value('TriggerMode', 1)
print('status={0}'.format(status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.3. get_enum_str_value

This function gets the current value of an IEnumeration node as a string.

[Syntax]

pytelicam.GenApiWrapper.get_enum_str_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value.

[Return_type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, trigger_mode = cam_device.genapi.get_enum_str_value('TriggerMode')
print('status={0} , trigger_mode={1}'.format(status, trigger_mode))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.4. set_enum_str_value

This function sets the value of an IEnumeration node as a string.

[Syntax]

pytelicam.GenApiWrapper.set_enum_str_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.
value (str)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_enum_str_value('TriggerMode', 'On')
print('status={0}'.format(status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.5. get_available_enum_entry_names

This function gets a tuple of available entry names contained in the IEnumeration node.

[Syntax]

pytelicam.GenApiWrapper.get_available_enum_entry_names(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.

[Returns]

A tuple type data of entry names.

[Return type]

tuple(str, …)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will return an empty tuple.

The names of unavailable entry are not added to the tuple.

An example is as follows:

```
entry_names = cam_device.genapi.get_available_enum_entry_names('TriggerMode')
print(entry_names)
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.6. get_num_of_enum_entries

This function gets the number of entries contained in the IEnumeration node.

[Syntax]

pytelicam.GenApiWrapper.get_num_of_enum_entries(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.

[Returns]

A tuple type data (status, num)

status	: A status code indicating the result of the operation.
num	: The number of entries.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.7. get_enum_entry_access_mode

This function gets the access mode of the IEnumEntry node contained in the IEnumeration node.

[Syntax]

pytelicam.GenApiWrapper.get_enum_entry_access_mode(self, node_name, index)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.
index (int)	Index of the list of entries contained in the specified IEnumeration node.

[Returns]

A tuple type data (status, access_mode)

status	: A status code indicating the result of the operation.
access_mode	: The access mode of the node.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.NodeAccessMode)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

An example is as follows:

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
for index in range(entry_num):
    status, access_mode = cam_device.genapi.get_enum_entry_access_mode( \
        'TriggerMode', index)
    print('status={0} , access_mode={1}'.format(status, access_mode))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.8. get_enum_entry_int_value

This function gets the value of the IEnumEntry node contained in the IEnumeration node as an integer value.

[Syntax]

pytelicam.GenApiWrapper.get_enum_entry_int_value(self, node_name, index)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.
index (int)	Index of the list of entries contained in the specified IEnumeration node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of the IEnumEntry node.

[Return type]

tuple(pytelicam.CamApiStatus, int)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
for index in range(entry_num):
    status, value = cam_device.genapi.get_enum_entry_int_value( \
        'TriggerMode', index)
    print('status={0} , value ={1}'.format(status, value))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.7.9. get_enum_entry_str_value

This function gets the value of the IEnumEntry node contained in the IEnumeration node as a string.

[Syntax]

pytelicam.GenApiWrapper.get_enum_entry_int_value(self, node_name, index)

[Parameters]

Parameters	Description
node_name (str)	The name of IEnumeration node.
index (int)	Index of the list of entries contained in the specified IEnumeration node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The value of the IEnumEntry node.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value "status" and no exception is raised.

[Remarks]

This function is only for IEnumeration nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status, entry_num = cam_device.genapi.get_num_of_enum_entries('TriggerMode')
print('status={0} , entry_num={1}'.format(status, entry_num))
for index in range(entry_num):
    status, value = cam_device.genapi.get_enum_entry_str_value( \
        'TriggerMode', index)
    print('status={0} , value ={1}'.format(status, value))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.8. ICommand node functions

7.7.8.1. execute_command

This function executes the command of ICommand node.

[Syntax]

pytelicam.GenApiWrapper.execute_command(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of ICommand node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value and no exception is raised.

[Remarks]

This function is only for ICommand nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_enum_str_value('UserSetSelector', 'UserSet1')
print('status={0}'.format(status))

status = cam_device.genapi.execute_command('UserSetSave')
print('status={0}'.format(status))

while True:
    status, is_done = cam_device.genapi.get_command_is_done('UserSetSave')
    print('status={0}, is_done={1}'.format(status, is_done))
    if status == pytelicam.CamApiStatus.Success and is_done == True:
            break;
    time.sleep(0.2)
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.8.2. get_command_is_done

This function checks that the command execution of the ICommand node has finished or not.

[Syntax]

pytelicam.GenApiWrapper.get_command_is_done(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of ICommand node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: Completion Status. If True, the command has been executed. If False, the command is still executing.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for ICommand nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.9. IString node functions

7.7.9.1. get_str_value

This function gets the current value of an IString node.

[Syntax]

pytelicam.GenApiWrapper.get_str_value(self, node_name)

[Parameters]

Parameters	Description
node_name (str)	The name of IString node.

[Returns]

A tuple type data (status, value)

status	: A status code indicating the result of the operation.
value	: The current value.

[Return type]

tuple(pytelicam.CamApiStatus, str)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

[Remarks]

This function is only for IString nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

An example is as follows:

```
status,                              user_defined_name                             =
cam_device.genapi.get_str_value('UserDefinedName')
print('status={0} , user_defined_name={1}'.format(status, user_defined_name))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.9.2. set_str_value

This function sets the value of an IString node.

[Syntax]

pytelicam.GenApiWrapper.set_str_value(self, node_name, value)

[Parameters]

Parameters	Description
node_name (str)	The name of IString node.
value (str)	The value to set to the node.

[Returns]

A status code indicating the result of the operation.

The main return values are as follows.
See pytelicam.CamApiStatus for other return values.

CamApiStatus.Success :	It succeeded.
CamApiStatus.GenICamError :	Error occurred in GenICam GenApi.

[Return type]

pytelicam.CamApiStatus

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

This function is only for IString nodes.
If node type of the target node is different node type, this function will set an error code in status and return it.

When CamApiStatus.GenICamError occurs, you can get the error information with the get_last_genicam_error() function of the CameraDevice object.

An example is as follows:

```
status = cam_device.genapi.set_str_value('UserDefinedName', 'Test')
print('status={0}'.format(status))
```

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.7.10. Others

7.7.10.1. get_access_module

This function gets the setting of the module (xml file) accessed by GenICam functions.

This function is only valid for cameras opened by GenTL interface.

[Syntax]

pytelicam.GenApiWrapper.get_access_module(self)

[Returns]

A tuple type data (status, access_module)

status	: A status code indicating the result of the operation.
access_module	: The current value of the module.

[Return type]

tuple(pytelicam.CamApiStatus, pytelicam.AccessModuleType)

[Raises]

pytelicam.PytelicamError : Raised when an unexpected error occurs in this API.
When a normal error occurs, the status code is stored in the return value “status” and no exception is raised.

7.7.10.2. set_access_module

This function sets the setting of the module (xml file) accessed by GenICam functions.

This function is only valid for cameras opened by GenTL interface.

[Syntax]

pytelicam.GenApiWrapper.set_access_module(self, access_module)

[Parameters]

Parameters	Description
access_module (str)	Access module value to set.

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

If stream interface is not open, you cannot use GenApiWrapper class function to get stream module information even if pytelicam.AccessModuleType.Stream is set.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.8. SignalHandle class

This class is a class for managing a signal.
This class manages the handle to access the signal functions of TeliCamSDK.

This class is used to detect a signal as follows:
• Completion of image data acquisition from the camera.
• Completion of event data acquisition from the camera.
• Camera removal detection.

The object of this class is created by create_signa() function.

[Syntax]

pytelicam.GenApiWrapper

[Properties]

	Name	Description
	handle (PyCapsule)	A handle to the signal object inside this API.

7.9. CamSystemInfo class

This class is a class for providing system information for pytelicam.

[Syntax]

pytelicam.CamSystemInfo

[Properties]

	Name	Description
	dll_version (str)	The version of TeliCamAPI loaded by pytelicam.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.10. CameraInfo class

This class is a class for providing camera information.

[Syntax]

pytelicam.CameraInfo

[Properties]

	Name	Description
	cam_type (pytelicam.CameraType)	Interface type of the camera.
	cam_vendor (str)	Vendor name of the camera.
	cam_model (str)	Model name of the camera.
	cam_serial_number (str)	Serial number of the camera.
	cam_version (str)	Version of the camera.
	cam_user_defined_name (str)	User defined name of the camera.
	cam_display_name (str)	Display name of the camera.
	tl_vendor (str)	Vendor name of the Transport layer API.
	tl_model (str)	Model name of the Transport layer API.
	tl_version (str)	Version of the Transport layer API.
	tl_display_name (str)	Display name of the Transport layer API.
	tl_if_display_name (str)	Display name of the interface of the Transport layer API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.11. ImageData class

This class is a class for managing image data.

[Syntax]

pytelicam.ImageData

[Functions]

	Name	Description
	release	Unlocks own ImageData object.
	get_ndarray	Gets the image data converted to NumPy array.
	get_memoryview	Gets the image data with a python's memoryview object.

[Properties]

	Name	Description
	timestamp (int)	Timestamp of the camera when image data was sent. Unit of this value is nano second in USB3 Camera, In GigE cameras, the units of timestamps depend on the model. By reading the value of the "GevTimestampTickFrequency" node (unit: Hz) and taking the reciprocal, the unit of the timestamp can be determined.
	pixel_format (pytelicam.CameraPixelFormat)	Pixel format of the image data.
	size_x (int)	Image width, in pixels.
	size_y (int)	Image height, in pixels.
	offset_x (int)	Start position in horizontal direction of the image, in pixel.
	offset_y (int)	Start position in vertical direction of the image, in pixel.
	padding_x (int)	Number of padding data at the end of row, in bytes.
	block_id (int)	Image index of the image issued by the camera. The camera increments the image index each time it outputs an image. It will be reset when the image transfer is stopped.
	buffer_ptr (PyCapsule)	A pointer to the image data buffer. The image data buffer is managed by this API.
	size (int)	Size of the image data, in bytes.
	image_id (int)	Image index managed by this API. This API increments this value regardless of whether the image was acquired normally or not. Normally, use the block_id set by the camera.
	status (pytelicam.CamApiStatus)	Status code. It is used to know if the image data was successfully acquired or not.
	lock_buffer_index (int)	The buffer index in the stream ring buffer inside this API used by this ImageData object. When this ImageData object is got, its buffer has locked.
	camera_device (pytelicam.CameraDevice)	The object of the camera device which this ImageData was created.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.11.1. release

This function unlocks own ImageData object. This API recognizes that the user application has finished using this ImageData object.

The ImageData object is got by get_next_image() , get_current_buffered_image() or get_buffered_image() functions.

[Syntax]

pytelicam.ImageData.release(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

All ImageData objects are managed in the stream ring buffer inside this API. If you repeatedly get images without releasing ImageData objects, there will be no unused ImageData object in the stream ring buffer, and you will not be able to get images. Release the ImageData object immediately after using it.
The ImageData object is managed by a smart pointer. You can also release it with 'with' syntax or ‘del’ syntax of python.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.11.2. get_ndarray

This function gets the image data converted to NumPy array.

[Syntax]

```
pytelicam.ImageData.get_ndarray(
                                  self,
                                  output_type=pytelicam.OutputImageType.RAW,
                                  bayer_conv_mode=pytelicam.BayerConversionMode.BiLinear)
```

[Parameters]

Parameters	Description
output_type (pytelicam.OutputImageType)	Output image type
bayer_conv_mode (pytelicam.BayerConversionMode)	Bayer conversion processing method. This argument has no meaning in any of the following cases: • When output_type argument is set to RAW. • When pixel format of the image data is not in a Bayer format.

[Returns]

Image data.

[Return type]

numpy.ndarray

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

The ACPI method of bayer conversion processing takes longer processing time than the BiLinear method.

The Bayer conversion process can also be executed using image processing software such as OpenCV after getting the image data in RAW format.

7.11.3. get_memoryview

This function gets the image data with a python's memoryview object.

[Syntax]

pytelicam.ImageData.get_memoryview()

[Returns]

Image data.

[Return type]

memoryview

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.12. EventData class

This class is a class for managing camera event data.

[Syntax]

pytelicam.EventData

[Functions]

	Name	Description
	release	Unlocks own EventData object.
	get_ndarray	Gets the camera event data converted to NumPy array.
	get_memoryview	Gets the camera event data with a python's memoryview object.

[Properties]

	Name	Description
	request_id (int)	Request ID of the event data output by the camera.
	event_id (int)	Event ID of the event data output by the camera. Event ID can be found in the camera's instruction manual.
	timestamp (int)	Timestamp of the camera when image data was sent. Unit of this value is nano second in USB3 Camera, In GigE cameras, the units of timestamps depend on the model. By reading the value of the "GevTimestampTickFrequency" node (unit: Hz) and taking the reciprocal, the unit of the timestamp can be determined.
	status (pytelicam.CamApiStatus)	Status code. It is used to know if the camera event data was successfully acquired or not.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.12.1. release

This function unlocks own EventData object. This API recognizes that the user application has finished using this EventData object.

The EventData object is got by get_event_data() function.

[Syntax]

pytelicam.EventData.release(self)

[Returns]

None

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

[Remarks]

All EventData objects are managed the event FIFO buffer inside this API. If you repeatedly get camera event data without releasing EventData objects, there will be no unused EventData object in the event FIFO buffer, and you will not be able to get camera event data. Release the EventData object immediately after using it.
The EventData object is managed by a smart pointer. You can also release it with 'with' syntax or ‘del’ syntax of python.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.12.2. get_ndarray

This function gets the camera event data converted to NumPy array.

[Syntax]

pytelicam.EventData.get_ndarray()

[Returns]

Camera event data.

[Return type]

numpy.ndarray

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

7.12.3. get_memoryview

This function gets the camera event data with a python's memoryview object.

[Syntax]

pytelicam.EventData.get_memoryview()

[Returns]

Camera event data.

[Return type]

memoryview

[Raises]

pytelicam.PytelicamError : Raised when an error occurs in this API.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

7.13. PytelicamError class

This class is an exception class that is raised when an error occurs in this API.

[Syntax]

pytelicam.PytelicamError

[Properties]

	Name	Description
	message (str)	Exception message.
	status (pytelicam.CamApiStatus)	Error status code that caused the exception to be raised.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8. Enumeration types

pytelicam provides the following Enumeration types for user applications.

[Enumerations]

	Name	Description
	CamApiStatus	Specifies the status code.
	CameraType	Specifies the interface type of the camera.
	CameraAccessMode	Specifies the access mode of the camera.
	CameraPixelFormat	Specifies the pixel format of image data.
	CameraAcquisitionMode	Specifies the acquisition mode of the camera.
	CameraEventType	Specifies the type of camera event data.
	NodeType	Specifies the type of the GenICam feature node.
	NodeAccessMode	Specifies the access mode of the GenICam feature node.
	AccessModuleType	Specifies the module (xml file) accessed by GenApiWrapper class.
	OutputImageType	Specifies the type of the output image.
	BayerConversionMode	Specifies the mode of bayer conversion.
	CameraImageFormat	Specifies the image format of camera.
	CameraTestPattern	Specifies the test pattern choice.
	CameraAcqFrameRateCtrl	Specifies the frame rate control mode.
	CameraImageBufferMode	Specifies the status of image buffer in the camera
	CameraTriggerSequence	Specifies the exposure time controlling sequence.
	CameraTriggerSource	Specifies the trigger source of random trigger shutter.
	CameraTriggerActivation	Specifies the activation mode of the trigger.
	CameraExposureTimeCtrl	Specifies the control mode of exposure time.
	CameraLineSelector	Specifies the selector for I/O line in the camera.
	CameraLineSource	Specifies the source signal of I/O line in the camera.
	CameraLineMode	Specifies the input/ Output line for I/O in the camera.
	CameraTimerTriggerSource	Specifies the trigger source signal of Timer0Active sigal.
	CameraGainAuto	Specifies the status of AGC
	CameraBalanceRatioSelector	Specifies the color component gain selector for White Balance.
	CameraBalanceWhiteAuto	Specifies the mode of autonmatic white balance control.
	CameraSaturationSelector	Specifies the selector for Saturation.
	CameraColorCorrectionMatrixSelector	Specifies the element selector for color correction matrix.
	CameraUserSetSelector	Specifies the channel selector for UserSet feature.
	CameraChunkSelector	Specifies the selector for chunk data.
	CameraFrameSynchronization	Specifies the type of frame synchronization method.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

[Remarks]

Enumeration types that are not described in this manual are for use inside pytelicam only.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.1. CamApiStatus (Status Code)

Specifies the status code.

[Syntax]

pytelicam.CamApiStatus

[Members]

Name	Description
Success	It succeeded.
NotInitialize	The initialization for API has never been performed.
AlreadyInitialized	The initialization for API has already been performed.
NotFound	API detected no cameras.
AlreadyOpened	Camera or the other object has been already opened.
AlreadyActivated	Camera or the other object has been already activated.
InvalidCameraIndex	The specified camera index is invalid.
InvalidCameraHandle	The specified camera handle is invalid.
InvalidNodeHandle	The specified node handle is invalid.
InvalidStreamHandle	The specified stream handle is invalid.
InvalidRequestHandle	The specified request handle is invalid.
InvalidEventHandle	The specified event handle is invalid.
InvalidParameter	The specified parameter is invalid.
BufferTooSmall	The specified buffer is too small.
NoMemory	To allocate internal buffer of API is unsuccessful.
MemoryNoAccess	To access to the specified buffer is unsuccessful.
NotImplemented	The feature is not implemented in the camera or API. This status may be also returned when wrong register name or node name is used.
Timeout	The timeout occurred.
CameraNotResponding	The specified camera does not send response. The cable which has connected the camra may have separated. Please check the connection.
EmptyCompleteQueue	The request in the Complete Queue is empty.
NotReady	The target is not ready state.
AccessModeSetError	Api failed to set access mode of the camera.
InsufficientBufferNum	Insufficient number of buffers.
IoDeviceError	Controller caused an I/O error.
LogicalParameterError	Passed parameters have logical error(s).
NotConnectedToUSB3	The camera is connected by something other than USB3.
XML_LoadError	Api failed to load XML file (camera description file).
GenICamError	Error occurred in GenApi dll.
DLL_LoadError	Api failed to load DLL file.
NoSystemResources	Insufficient system resources exist to complete the requested service.
InvalidAddress	The specified addres is invalid.
WriteProtect	The register is protected from writing data.
BadAlignment	The address is not aligned to specified boundary.
AccessDenied	Accessing data was denied. User application may not have access privilege.
Busy	The camera is in busy state. Try again after a while.
NotReadable	The target is not readable.
NotWritable	The target is not writable.
NotAvailable	The function or parameter is not available.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

	The controlling camera feature functions that use GenICam GenApi function are not available when GenApi module was disabled on opening the camera.
VerifyError	A verify error occurred when data was written to the camera registers.
RequestTimeout	Timeout occurred in requesting stream or event.
ResendTimeout	The StreamRequests coud not be completed in time after the first packet was received. (only in GigE camera)
ResponseTimeout	The next packet could not be received in time after somepacket was received. (Only in GigE camera)
BufferFull	The received data size exceeded maximum size specified.
UnexpectedBufferSize	The data size actually received was different from the sizedescribed in the trailer.
UnexpectedNumber	The received packet count exceeded the maximum.
PacketStatusError	The packet returned error status.
ResendNotImplemented	Resend command is not implemented in the camera.
PacketUnavailable	The packet is unavailable.
MissingPackets	A leader of the next block has been received before the completion of a frame data. Packets may be lost.
FlushRequested	The requestwas flushed by the user.
TooManyPacketMissing	The loss of packet exceeded the specified value. (default:20) In GigE camera case, band width of the network may not be enough for sending images in current settings. Enabling Jumbo-Packet or restrictimg frame rate will be required.
FlushedByD0Exit	The request was flushed due to a change of the power state.
FlushedByCameraRemove	The request was flushed due to a disconnection with the camera.
Driver_LoadError	Api failed to load Driver.
MappingError	Mapping user buffer to system-space virtual address is failed. It may be caused by low system resources.
FileOpenError	Api failed to open file.
FileWriteError	Api failed to write data in file.
FileReadError	Api failed to read data from file.
FileNotFound	The specified file was not found.
InvalidParameterFromCam	Invalid parameter error was returned from the camera.
PayloadSizeNotAligned	The value written to the SI streaming size registers is not aligned to Payload Size Alignment value of the SI Info register.
DataDiscarded	Some data in the block has been discarded.
DataOverrun	The camera cannot send all data because the data does not fit within the programmed SIRM register settings.
OutOfMemory	The system and/or other hardware in the system(Framegrabber) ran out of memory.
U3vNotAvailable	Failed to load U3V DLL. (Warning) Interfaces other than USB3 are available.
GevNotAvailable	Failed to load GEV DLL. (Warning) Interfaces other than GigE are available.
GenTLNotAvailable	Failed to load GenTL DLL. (Warning) Interfaces other than GenTL are available.
Unsuccessful	The other error occurred.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.2. CameraType

Specifies the interface type of the camera.

[Syntax]

pytelicam.CameraType

[Members]

Name	Description
All	All interfaces using the device drivers provided by Toshiba Teli. GenTL interface is not included.
Gev	GigE interface.
U3v	USB3 interface.
GenTL	GenTL interface. GenTL Producer provided by the frame grabber manufacturer is used.
Unknown	Unknown interface.

8.3. CameraAccessMode

Specifies the access mode of the camera.

It has no meaning except for GigE cameras.

[Syntax]

pytelicam.CameraAccessMode

[Members]

Name	Description
Open	User application can read registers of the camera, cannot write data.
Control	User application can fully control the camera. The other application can read registers of the camera, cannot write data.
Exclusive	User application can fully control the camera. The other application cannot open the camera.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.4. CameraPixelFormat

Specifies the pixel format of image data.

[Syntax]

pytelicam.CameraPixelFormat

[Members]

Name	Description
Unknown	Unknown format.
Mono8	Mono8 format.
Mono10	Mono10 format.
Mono10p	Mono10 packed format.
Mono12	Mono12 format.
Mono12p	Mono12 packed format.
Mono16	Mono16 format.
BayerGR8	BayerGR8 format.
BayerGR10	BayerGR10 format.
BayerGR12	BayerGR12 format.
BayerRG8	BayerRG8 format.
BayerRG10	BayerRG10 format.
BayerRG12	BayerRG12 format.
BayerGB8	BayerGB8 format.
BayerGB10	BayerGB10 format.
BayerGB12	BayerGB12 format.
BayerBG8	BayerBG8 format.
BayerBG10	BayerBG10 format.
BayerBG12	BayerBG12 format.
RGB8	RGB8 format.
BGR8	BGR8 format.
BGR10	BGR10 format.
BGR12	BGR12 format.
YUV411_8	YUV411_8 format.
YUV422_8	YUV422_8 format.
YUV8	YUV8 format.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.5. CameraAcquisitionMode

Specifies the acquisition mode of the camera.

[Syntax]

pytelicam.CameraAcquisitionMode

[Members]

Name	Description
Continuous	The camera repeats image acquisition and transfer until stop() function is called. "Continuous" is set in the AcquisitionMode register of the camera.
SingleFrame	The camera acquires a single image and transfers it. When this function is called, the AcquisitionStart command is executed and an image is acquired. To acquire a new single image again, execute this function. When this function is called, the value of the AcquisitionFrameCount register of the camera is rewritten to 1.
MultiFrame	The camera repeats image acquisition and transfer until the number of images specified by the AcquisitionFrameCount register. Use GenApiWrapper object to change the value of the AcquisitionFrameCount register. Example: cam_device.genapi.set_int_value('AcquisitionFrameCount', 5) When this function is called, the AcquisitionStart command is executed and images are acquired. To acquire new images again, execute this function.
ImageBufferRead	The camera repeats acquiring images and storing them to image buffers in the camera until stop() function is called. When the ImageBufferRead command is executed, the images are transferred from the image buffer in the camera. Use GenApiWrapper object to execute the ImageBufferRead command. Example: cam_device.genapi.execute_command('ImageBufferRead')

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.6. CameraEventType

Specifies the type of camera event data.

[Syntax]

pytelicam.CameraEventType

[Members]

Name	Description
FrameTrigger	Accepted trigger signal for acquiring an image.
FrameTriggerError	Received trigger signal for acquiring image at invalid timing.
FrameTriggerWait	Started accepting trigger signal for image acquisition.
FrameTransferStart	Started transferring image stream of a frame.
FrameTransferEnd	Completed transferring image stream of a frame.
ExposureStart	Started exposure for a frame.
ExposureEnd	Finished exposure for a frame.
Timer0Start	Timer0 started counting.
Timer0End	Timer0 finished counting.
ALCLatestInformation	Updated Automatic Luminance Control data.
ALCConverged	Converged Automatic Luminance Control operation.
Unknown	Unknown type.

8.7. NodeType

Specifies the type of the GenICam feature node.

[Syntax]

pytelicam.NodeType

[Members]

Name	Description
Value	Value node. (IValue interface)
Base	Base node. (IBase interface)
Integer	Integer node. (IInteger interface)
Boolean	Boolean node. (IBoolean interface)
Command	Command node. (ICommand interface)
Float	Float node. (IFloat interface)
String	String node. (IString interface)
Register	Register node. (IRegister interface)
Category	Category node, (ICategory interface)
Enumeration	Enumeration node. (IEnumeration interface)
EnumEntry	EnumEntry node. (IEnumEntry interface)
Port	Port node. (IPort node)
Unknown	Unknown node type.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.8. NodeAccessMode

Specifies the access mode of the GenICam feature node.

[Syntax]

pytelicam.NodeAccessMode

[Members]

Name	Description
NotImplemented	Not implemented.
NotAvailable	Not available.
WriteOnly	Write only.
ReadOnly	Read only.
ReadAndWrite	Read and Write.
Undefined	Object is not yet initialized.
Unknown.	Unknown mode.

8.9. NodeVisibility

Specifies the recommended visibility (desirable user level for accessing a node) of the GenICam feature node.

[Syntax]

pytelicam.NodeVisibility

[Members]

Name	Description
Beginner	All users can access the node safely.
Expert	Experts and Guru can access the node safely.
Guru	Well trained specialist can access the node safely.
Invisible	No users can access the node.
Undefined	Node object is not initialized,
Unknown	Unknown.

8.10. NodeCachingMode

Specifies the caching mode of the GenICam feature node.

[Syntax]

pytelicam.NodeCachingMode

[Members]

Name	Description
NoCache	Not use cache.
WriteThrough	A value written to the camera is written to the cache as well.
WriteAround	Only read values are written to the cache.
Undefined	Node object is not initialized.
Unknown	Unknown.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.11. NodeRepresentation

Specifies the recommended representation of the GenICam feature node value.

[Syntax]

pytelicam.NodeRepresentation

[Members]

Name	Description
Linear	Slider with linear behavior.
Logarithmic	Slider with logarithmic behavior.
Boolean	Check box.
PureNumber	Decimal number in an edit control.
HexNumber	Hex number in an edit control.
IPV4Address	IP address,
MACAddress	MAC address.
Undefined	Node object is not initialized,
Unknown	Unknown.

8.12. NodeDisplayNotation

Specifies the float notation of the GenICam feature node value.

[Syntax]

pytelicam.NodeDisplayNotation

[Members]

Name	Description
Automatic	The notation if either scientific or fixed depending on what is shorter.
Fixed	The notation is fixed, e.g. 123.4
Scientific	The notation is scientific, e.g. 1.234e2
Undefined	Object is not yet initialized
Unknown	Unknown.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.13. AccessModuleType

Specifies the module (xml file) accessed by GenApiWrapper class.

[Syntax]

pytelicam.AccessModuleType

[Members]

Name	Description
RemoteDevice	Remote device (camera)
System	System of GenTL Producer
Interface	Interface of GenTL Producer
Device	Device of GenTL Producer
Stream	Stream of GenTL Producer

8.14. OutputImageType

Specifies the type of the output image.

[Syntax]

pytelicam.OutputImageType

[Members]

Name	Description
Raw	Raw format.
Bgr24	BGR 24bit format.
Bgra32	BGRA 32bit format.

8.15. BayerConversionMode

Specifies the mode of bayer conversion.

[Syntax]

pytelicam.BayerConversionMode

[Members]

Name	Description
BiLinear	BiLinear method.
Acpi	ACPI method.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.16. CameraImageFormat

Specifies the image format of the camera.

[Syntax]

pytelicam.CameraImageFormat

[Members]

Name	Description
Format0	Format0
Format1	Format1
Format2	Format2

8.17. CameraTestPattern

Specifies the test patter (test image) of the camera.

[Syntax]

pytelicam.CameraTestPattern

[Members]

Name	Description
Off	Test pattern disable(Normal data output)
Black	All pixel = 0 LSB
White	All pixel = 255 @Mono8
GreyA	All pixel = 170 @Mono8
GreyB	All pixel = 85 @Mono8
HotizontalRamp	Grey Horizontal Ramp
GreyScale	Grey scale
ColorBar	Color Bar
VertiaclRamp	Grey Vertical Ramp

8.18. CameraAcqFrameRateCtrl

Specifies the control mode of frame rate.

[Syntax]

pytelicam.CameraAcqFrameRateCtrl

[Members]

Name	Description
NoSpecify	Settings other than AcquisitionFrameRate (for example, Exposure Time, etc.,) has priority.
Manual	AcquisitionFrameRate parameter has priority over the other parameters.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.19. CameraImageBufferMode

Specifies the status of image buffer in the camera.

[Syntax]

pytelicam.CameraImageBufferMode

[Members]

Name	Description
Off	Image buffer mode is disabled.
On	Image buffer mode is enabled.

8.20. CameraTriggerSequence

Specifies the exposure time controlling sequence.

[Syntax]

pytelicam.CameraTriggerSequence

[Members]

Name	Description
Sequence0	Edge mode. The exposure time is determined by Exposure Time setting. • The camera that complie with IIDC2 : TriggerSequence register = TriggerSequence0 • The camera that does not comply with IIDC2 : ExposureMode register = Timed TriggerSelector register = FrameStart
Sequence1	Level mode. The exposure time is determined by the pulse width of the trigger signal. • The camera that complie with IIDC2: TriggerSequence register = TriggerSequence1 • The camera that does not comply with IIDC2 : ExposureMode register = TriggerWidth TriggerSelector register = FrameStart
Sequence6	Bulk (or FrameBurst) mode. Camera exposes and transfers multiple frames by a single trigger. • The camera that complie with IIDC2: TriggerSequence register = TriggerSequence6 • The camera that does not comply with IIDC2 : ExposureMode register = Timed TriggerSelector register = FrameBurstStart

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.21. CameraTriggerSource

Specifies the trigger source signal of random trigger shutter.

[Syntax]

pytelicam.CameraTriggerSource

[Members]

Name	Description
Line0	Hardware Trigger Line0
Line1	Hardware Trigger Line1
Line2	Hardware Trigger Line2
Software	Software trigger.

8.22. CameraTriggerActivation

Specifies the activation mode of the trigger.

[Syntax]

pytelicam.CameraTriggerActivation

[Members]

Name	Description
FallingEdge	Falling Edge mode
RisingEdge	Rising Edge mode

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.23. CameraExposureTimeCtrl

Specifies the control mode of exposure time.

[Syntax]

pytelicam.CameraExposureTimeCtrl

[Members]

Name	Description
NoSpecify	The camera will determine exposure time using registers other than ExposureTime register. • The camera that complie with IIDC2 : ExposureTimeControl register = NoSpecify • The camera that does not comply with IIDC2 : Not Available
Manual	The camera will determine exposure time using ExposureTime register value. • The camera that complie with IIDC2 : ExposureTimeControl register = Manula • The camera that does not comply with IIDC2 : ExposureAuto register = Off
Auto	The camera will control exposure time automatically. • The camera that complie with IIDC2 : ExposureTimeControl register = Auto • The camera that does not comply with IIDC2 : ExposureAuto register = Continuous

8.24. CameraLineSelector

Specifies the selection of I/O line in the camera.

[Syntax]

pytelicam.CameraLineSelector

[Members]

Name	Description
Line0	Line0
Line1	Line1
Line2	Line2

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.25. CameraLineSource

Specifies the source signal for I/O line in the camera.

[Syntax]

pytelicam.CameraLineSource

[Members]

Name	Description
Off	Off
VD	Internal VD sync signal.(Only for GigE Camera)
UserOutput	Outputs the value set in “UserOutputValue”.
Timer0Active	This signal can be used as strobe control signal. The delay time and pulse width of this signal are configurable.
AcquisitionActive	Indicates AcquisitionStart state of camera.
FrameTriggerWait	Indicates that camera is ready to accept trigger signal. (both hardware and software)
FrameActive	Period from exposure start to sensor read-out completion.
FrameTransferActive	Period of transferring image streaming data on interface bus.
ExposureActive	Period from exposure start to exposure end.

8.26. CameraLineMode

Specifies the input/ output line for I/O line in the camera.

[Syntax]

pytelicam.CameraLineMode

[Members]

Name	Description
Output	Output
Input	Input

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.27. CameraTimerTriggerSource

Specifies the trigger source signal for Timer0.

[Syntax]

pytelicam.CameraTimerTriggerSource

[Members]

Name	Description
Off	DisablesTimer0Active signal.
Line0Active	Starts when Line0 is active.
FrameTrigger	Starts with the reception of the Frame Start Trigger.
ExposureStart	Starts with the reception of the Exposure Start.
ExposureEnd	Starts with the reception of the Exposure End.

8.28. CameraGainAuto

Specifies AGC status of the camera.

[Syntax]

pytelicam.CameraGainAuto

[Members]

Name	Description
Off	Manual Gain Control (MANUAL)
Auto	Automatic Gain Control (AGC)

8.29. CameraBalanceRatioSelector

Specifies the target color component for setting White balance.

[Syntax]

pytelicam.CameraBalanceRatioSelector

[Members]

Name	Description
Blue	BalanceRatio = Blue Gain
Red	BalanceRatio = Red Gain

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.30. CameraBalanceWhiteAuto

Specifies the mode of automatic white balance.

[Syntax]

pytelicam.CameraBalanceWhiteAuto

[Members]

Name	Description
Off	No operation.
Continuous	Execute auto white balance continuously.
Once	Execute auto white balance once.

8.31. CameraSaturationSelector

Specifies the element of saturation.

[Syntax]

pytelicam.CameraSaturationSelector

[Members]

Name	Description
U	Saturation = U Gain
V	Saturation = V Gain

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.32. CameraColorCorrectionMatrixSelector

Specifies the element of color correction matrix.

[Syntax]

pytelicam.CameraColorCorrectionMatrixSelector

[Members]

Name	Description
RG	SelectorI = R, SelectorJ = G.
RB	SelectorI = R, SelectorJ = B.
GR	SelectorI = G, SelectorJ = R.
GB	SelectorI = G, SelectorJ = B.
BR	SelectorI = B, SelectorJ = R.
BG	SelectorI = B, SelectorJ = G.

8.33. CameraUserSetSelector

Specifies the channel of UserSet memory.

[Syntax]

pytelicam.CameraUserSetSelector

[Members]

Name	Description
Default	Initial factory setting. This channel is read-only.
UserSet1	Memory channel 1 for user setting.
UserSet2	Memory channel 2 for user setting.
UserSet3	Memory channel 3 for user setting.
UserSet4	Memory channel 4 for user setting.
UserSet5	Memory channel 5 for user setting.
UserSet6	Memory channel 6 for user setting.
UserSet7	Memory channel 7 for user setting.
UserSet8	Memory channel 8 for user setting.
UserSet9	Memory channel 9 for user setting.
UserSet10	Memory channel 10 for user setting.
UserSet11	Memory channel 11 for user setting.
UserSet12	Memory channel 12 for user setting.
UserSet13	Memory channel 13 for user setting.
UserSet14	Memory channel 14 for user setting.
UserSet15	Memory channel 15 for user setting.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

8.34. CameraChunkSelector

Specifies the element of Chunk.

[Syntax]

pytelicam.CameraChunkSelector

[Members]

Name	Description
BlockID	BlockID
FrameBurstTriggerCount	FrameBurstTriggerCount
SequentialShutterNumber	SequentialShutter number
SequentialShutterElement	SequentialShutter element
UserArea	User area
ExposureTime	Exposure time
Gain	Gain
WhiteBalanceR	WhiteBalanceR
WhiteBalanceB	WhiteBalanceB
LineStatusAll	LineStatusAll

8.35. CameraFrameSynchronization

Specifies the type of frame synchronization method.

[Syntax]

pytelicam.CameraFrameSynchronization

[Members]

Name	Description
Bus	Bus synchronization
Off	Internal synchronization

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

9. Sample source code

The knowledge about GenICam, GigE Vision, USB3 Vision, and IIDC2 register map will be useful to understand sample source code. The detail or latest information of these standards can be obtained from home page of these standards.

GenICam	https://www.emva.org/standards-technology/genicam/
GigE Vision	https://www.automate.org/a3-content/vision-standards-gige-vision
USB3 Vision	https://www.automate.org/a3-content/usb3-vision-standard
IIDC2	http://jiia.org/mv_dl/iidc2/

Refer to “pytelicam Samples Manual Eng.pdf” about for more information of sample source code.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

10. Others

10.1. Disclaimer

The disclaimer for the pytelicam follows the disclaimer for the TeliCamSDK.

The disclaimer of TeliCamSDK is described in another “License Agreement TeliCamSDK Eng.pdf”.
Make sure to read this Agreement carefully before using it.
Refer to TeliCamSDK installation folder/Licenses folder

10.2. License

The license for the pytelicam follows the license for the TeliCamSDK.

TeliCamSDK consists of multiple, independent software components. Each software component is copyrighted by a third party. TeliCamSDK uses software components that are distributed as freeware

under a third-party end-user license agreement or copyright notice (hereinafter referred to as a “EULA”).

Some EULAs require that the source code of the applicable component be disclosed as the condition for distributing the software component in executable format. You can check the software components subject to such EULA requirements. For more information, please contact our inquiries described in section 9.4.

Toshiba Teli corporation provides a warranty for TeliCamSDK under conditions set forth by Toshiba Teli corporation. (See “License Agreement TeliCamSDK Eng.pdf” and “License Agreement TeliCamSDK Sample Eng.pdf”) However, some of the software components distributed under an EULA are made available for use by the user on the assumption that they are not copyrighted or warranted by a third party. These software components are licensed to the user free of charge and therefore not covered by any warranty within the scope of the applicable laws. These software components are not subject to any copyrights or other third-party rights and are provided in “as is” condition without any warranty, whether express or implied. “Warranty” here includes, but not limited to, an implied warranty for marketability

or fitness for specific uses. All risks associated with the quality or performance of these software components are assumed by the user.

EULAs are included in the installation directory: [TeliCamSDK install folder]/licenses .

Toshiba Teli corporation shall not be liable whatsoever for any cost of repair or correction or other incidental expense incurred in connection with a defect found in any of these software components. Unless specified under the applicable laws or in a written agreement, a party who changes or redistributes the software with consent from the copyright holders or based on the aforementioned licenses shall not be held liable whatsoever for any loss arising from the use of or inability to use such software components. The same applies even when the copyright holders or relevant third parties have been informed of the possibility of such loss. “Loss” here includes normal, special, incidental and indirect loss (including, but not limited to, the loss of data or its accuracy; loss incurred by the user or any third party; and interface incompatibility with other software). Please read each EULA for details on the use conditions and items that must be observed regarding these software components.

The table below lists the software components using in TeliCamSDK, which are subject to EULAs.
The user should read the applicable EULAs carefully before using these software components.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

Project name	Project license
GenICam GenApi	GenICam License

GenICam GenApi uses the following third party software.

Project name	Project license
MathParser	LGPLv2.1
Log4Cpp	LGPLv2.1
CppUnit	LGPLv2.1
CLSerAll	NI license
xs3p	DSTC license
xxhash	xxhash license
XSLTProc	MIT license
XSDe	Proprietary

pytelicam uses the following third party software.

Project name	Project license
pybind11	BSD-style license
numpy	BSD license
wheel	MIT license

TeliCamSDK redistributes the binaries of LGPL-applied software, and for these source code only, you have the right to obtain, modify and redistribute it in accordance with the LGPL provisions.

To the customer who wants the source code, we write to the media (CD - ROM etc.) and send it by post.

Customers must pay for actual expenses such as shipping fee. If you want, please contact our inquiries described in section 9.4. We distribute source code only for open source software that you have right to obtain. (Source code of TeliCamSDK is not included.) Please understand beforehand that we can not answer questions about the content of the source code etc.

Microsoft, Windows, Windows XP, Windows Vista, Windows 7, Windows 8.1, Windows 10, Windows 11 and Visual C++ are the trademark or the registered trademark of Microsoft Corporation.
USB3 Vision and GigE Vision are trademark or registered trademark of AIA (Automated Imaging Association) of each company.
CoaXPress is registered trademark of JIIA (Japan Industrial Imaging Association).
GenICam is trademark of EMVA (European Machine Vision Association).
Furthermore, company name or product name might be trademark or registered trademark of each company.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/

10.3. Revision History

Date	Version	Description
2021/03/29	1.0.0	Created the initial version.
2021/11/09	1.0.1	• Added description of new functions. save_parameter() , load_parameter() • Added new members in CamApiStatus enumeration. • Added new members in CameraPixelFormat enumeration.
2022/10/27	1.0.2	• Added description of new functions. create_device_object_from_ip_address() , get_next_image_with_trigger () • Also, some other modifications such as minor changes, correction of descriptions, etc. were applied.
2023/07/24	1.1.0	• Added description of CameraControl class. • Added new members in CameraImageFormat enumeration. • Added new members in CameraTestPattern enumeration. • Added new members in CameraAcqFrameRateCtrl enumeration. • Added new members in CameraImageBufferMode enumeration. • Added new members in CameraTriggerSequence enumeration. • Added new members in CameraTriggerSource enumeration. • Added new members in CameraTriggerActivation enumeration. • Added new members in CameraExposureTimeCtrl enumeration. • Added new members in CameraLineSelector enumeration. • Added new members in CameraLineSource enumeration. • Added new members in CameraLineMode enumeration. • Added new members in CameraTimerTriggerSource enumeration. • Added new members in CameraGainAuto enumeration. • Added new members in CameraBalanceRatioSelector enumeration. • Added new members in CameraBalanceWhiteAuto enumeration. • Added new members in CameraSaturationSelector enumeration. • Added new members in CameraColorCorrectionMatrixSelector enumeration. • Added new members in CameraUserSetSelector enumeration. • Added new members in CameraChunkSelector enumeration. • Added new members in CameraFrameSynchronization enumeration • Also, some other modifications such as minor changes, correction of descriptions, etc. were applied.
2024/12/13	1.1.1	• Added description of the 9.Sample source code. • Also, Some other modifications such as minor changes, correction of descriptions, etc. were applied.

10.4. Inquiry

For frequently asked questions (FAQ) and answers about TeliCamSDK, GigE cameras, USB3 cameras, and CoaXPress cameras, please visit the "Support" - "Industrial Cameras FAQ" site on our website.

If you still cannot solve the problem, please contact us using the phone number or Inquiries form from "Contact Us" site on our website.

Copyright © 2021-2024 Toshiba Teli Corporation, All rights reserved. https://www.toshiba-teli.co.jp/en/