



Chameleon API

User Guide 2014

International Distributors



Sky Blue Microsystems GmbH
Geisenhausenerstr. 18
81379 Munich, Germany
+49 89 780 2970, info@skyblue.de
www.skyblue.de



In Great Britain:
[Zerif Technologies Ltd.](http://ZerifTechnologiesLtd.com)
Winnington House, 2 Woodberry Grove
Finchley, London N12 0DR
+44 115 855 7883, info@zerif.co.uk
www.zerif.co.uk

1	Figures and Tables	4
2	Introduction.....	5
2.1	Safety Precautions	5
2.2	Disclaimer	6
3	Chameleon API Overview	7
3.1	Overview	7
3.2	Document structure	7
3.3	Directory and file hierarchy	8
3.4	Function call sequence	9
3.5	Stream defining camera parameters	10
3.6	Continues data stream	11
4	Connection and Info.....	12
4.1	KYCS_Scan().....	12
4.2	KYCS_Open().....	13
4.3	KYCS_Close()	13
4.4	KYCS_Pid2Name().....	13
4.5	KYCS_GetInfo()	14
5	Camera Configurations	15
5.1	KYCS_LoadCameraConfiguration().....	15
5.2	KYCS_SetCameraValue().....	16
5.2.1	KYCS_SetCameraValueInt().....	16
5.2.2	KYCS_SetCameraValueFloat()	17
5.2.3	KYCS_SetCameraValueBool().....	17
5.2.4	KYCS_SetCameraValueEnum().....	18
5.2.5	KYCS_ExecuteCommand().....	18
5.2.6	KYCS_SetCameraValueString().....	19
5.2.7	KYCS_SetCameraValueEnum_ByValueName()	19
5.3	KYCS_GetCameraValueType().....	20
5.4	KYCS_GetCameraValue().....	21
5.4.1	KYCS_GetCameraValueInt()	21
5.4.2	KYCS_GetCameraValueEnum()	22
5.4.3	KYCS_GetCameraValueFloat()	22
5.4.4	KYCS_GetCameraValueBool().....	23

	5.4.5 KYCS_GetCameraValueString()	23
6	Streaming Interface	25
	6.1 KYCS_AllocStream()	25
	6.2 KYCS_DeleteStream()	25
7	Buffer data interface	26
	7.1 KYCS_BufferGetPtr()	26
	7.2 KYCS_BufferGetSize()	26
	7.3 KYCS_BufferGetFrameIndex()	27
8	Data Loading	28
	8.1 KYCS_LoadPatternData()	28
	8.2 KYCS_LoadFileData()	29
9	Camera simulation control	31
	9.1 KYCS_CallbackRegister()	31
	9.2 KYCS_GetRuntimeInfo()	31
	9.3 KYCS_SimulationControl()	32
	9.4 KYCS_SimulationTrigger()	33
10	Low level bootstrap access	35
	10.1 KYCS_ReadBootstrapRegs()	35
	10.2 KYCS_WriteBootstrapRegs()	35
11	IO Control	37
	11.1 KYCS_SetTriggerSource()	37
	11.2 KYCS_ConfigureIO()	37
	11.3 KYCS_SetOutputEnable()	38
	11.4 IO Selector	39
12	Analog control	41
	12.1 Analog control options	41
13	API Defines and Macros	43
	13.1 CSCallback	43
14	Structures	44
	14.1 CSINFO	44
	14.2 CSRUNTIME	44
	14.3 VIDEO_PIXELIF	45
15	Enumerations	46
	15.1 CSSTATUS	46
	15.2 PATTERN_TYPE	47
	15.3 SIM_CONTROL	48

15.4	SIM_STATE	48
15.5	KY_CAM_PROPERTY_TYPE.....	49
15.6	VIDEO_DATA_WIDTH	49
15.7	VIDEO_DATA_TYPE.....	50
15.8	VIDEO_DATA_SUBTYPE.....	51
15.9	KY_IO.....	52
16	Configuration parameters.....	53
16.1	Simulation Control	53
16.2	Hardware Information	53

Index of tables

TABLE 1 : LINE SELECTOR IO FIELDS.....	39
TABLE 2 : LINE SELECTION OPTIONS	40
TABLE 3 : AVAILABLE CONFIGURATIONS FOR ANALOG CONTROL FIELDS	41
TABLE 4 : TYPICAL TX V_{OD} SETTING FOR TRANSCEIVER CHANNELS WITH TERMINATION OF $100\ \Omega$	41
TABLE 5 : TRANSMITTER PRE-EMPHASIS LEVELS	42
TABLE 6 : SIMULATION CONTROL FIELDS	53
TABLE 7 : HARDWARE INFORMATION FIELDS	53

Revision History

Version	Date	Notes
1.0	1.7.14	Initial release
1.1	1.11.14	Chameleon API release 1.0.5.132

2.1 Safety Precautions

With your KAYA Chameleon board in hand, please take a minute to read carefully the precautions listed below in order to prevent unnecessary injuries to you or other personnel or cause damage to property.

- **Before using the product, read these safety precautions carefully to assure correct use.**
- **These precautions contain serious safety instructions that must be observed.**
- **After reading through this manual, be sure to act upon it to prevent misuse of product.**



Caution

In the event of a failure, disconnect the power supply.
If the product is used as is, a fire or electric shock may occur. Disconnect the power supply immediately and contact our sales personnel for repair.
If an unpleasant smell or smoking occurs, disconnect the power supply.
If the product is used as is, a fire or electric shock may occur. Disconnect the power supply immediately. After verifying that no smoking is observed, contact our sales personnel for repair.
Do not disassemble, repair or modify the product.
Otherwise, a fire or electric shock may occur due to a short circuit or heat generation. For inspection, modification or repair, contact our sales personnel.
Do not touch a cooling fan.
As a cooling fan rotates in high speed, do not put your hand close to it. Otherwise, it may cause injury to persons. Never touch a rotating cooling fan.
Do not place the product on unstable locations.
Otherwise, it may drop or fall, resulting in injury to persons or failure.
If the product is dropped or damaged, do not use it as is.
Otherwise, a fire or electric shock may occur.
Do not touch the product with a metallic object.
Otherwise, a fire or electric shock may occur.
Do not place the product in dusty or humid locations or where water may splash.
Otherwise, a fire or electric shock may occur.
Do not get the product wet or touch it with a wet hand.
Otherwise, the product may break down or it may cause a fire, smoking or electric shock.
Do not touch a connector on the product (gold-plated portion).
Otherwise, the surface of a connector may be contaminated with sweat or skin oil, resulting in contact failure of a connector or it may cause a malfunction, fire or electric shock due to static electricity.

Do not use or place the product in the following locations.

- Humid and dusty locations
- Airless locations such as closet or bookshelf
- Locations which receive oily smoke or steam
- Locations close to heating equipment
- Closed inside of a car where the temperature becomes high
- Static electricity replete locations
- Locations close to water or chemicals

Otherwise, a fire, electric shock, accident or deformation may occur due to a short circuit or heat generation.

Do not place heavy things on the product.

Otherwise, the product may be damaged.

2.2 Disclaimer

This product should be used for CoaXPress video simulation. It also can be used for digital input/output (GPIO) simulation purposes. KAYA Instruments assumes no responsibility for any damages resulting from the use of this product for purposes other than those stated.

Even if the product is used properly, KAYA Instruments assumes no responsibility for any damages caused by the following:

- Earthquake, thunder, natural disaster or fire resulting from the use beyond our responsibility, acts caused by a third party or other accidents, the customer's willful or accidental misuse or use under other abnormal conditions.
- Secondary impact arising from use of this product or its unusable state (business interruption or others).
- Use of this product against the instructions given in this manual or malfunctions due to connection to other devices.

KAYA Instruments assumes no responsibility or liability for:

- Erasure or corruption of data arising from use of this product.
- Any consequences or other abnormalities arising from use of this product, or damage of this product not due to our responsibility or failure due to modification.

3.1 Overview

The purpose of this document is to list and demonstrate the provided functionality of the Chameleon Simulator API.

This API is to be used with Chameleon Simulator hardware provided by KAYA Instruments.

This is a high level API for connecting, configuring and simulating a CoaXPress stream of data over 1, 2 or 4 CXP channels.

The simulator is capable of producing data streams characterized by camera configuration fields in a loaded XML file.

3.2 Document structure

This API guide is divided into few major topics each responsible for different functionalities:

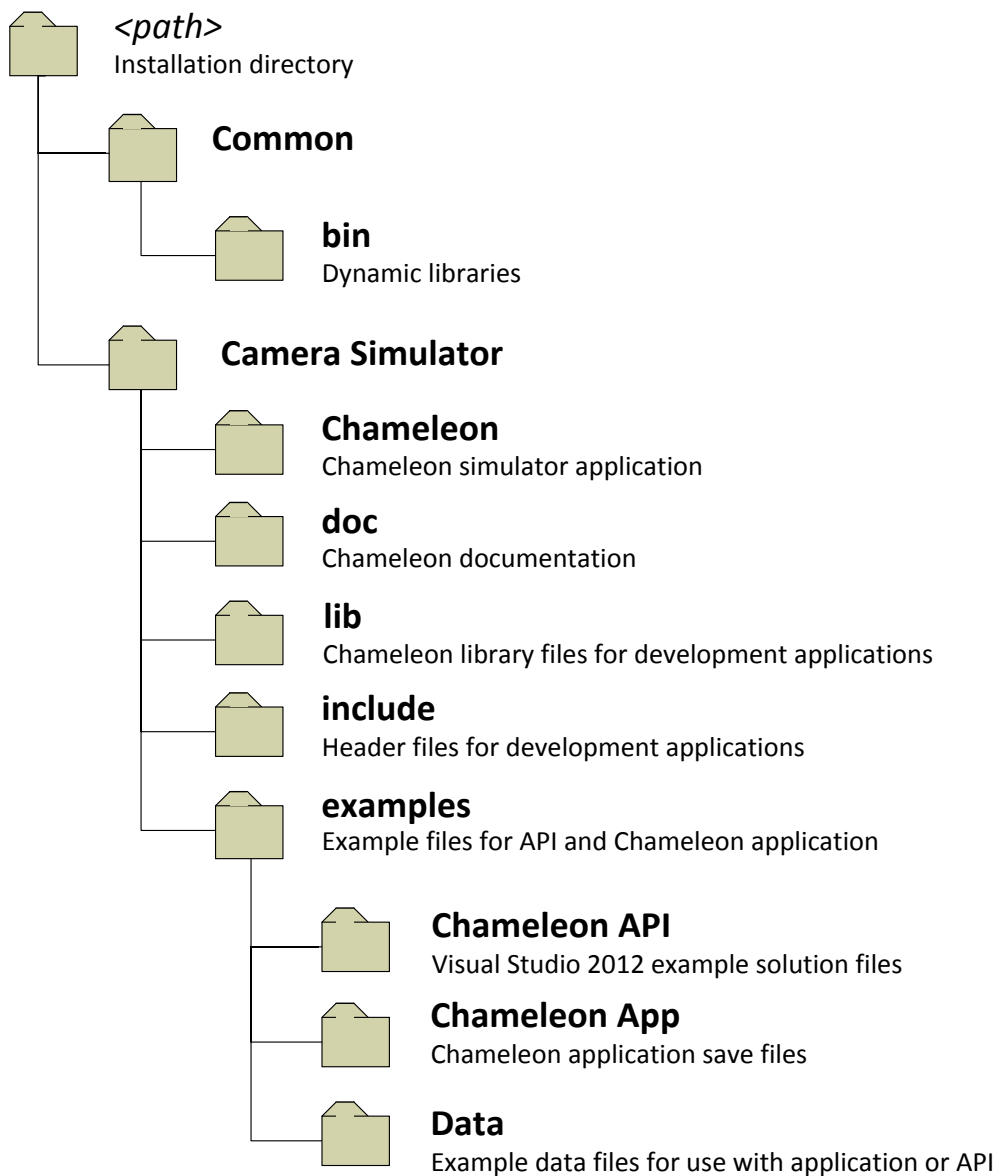
- Connection and Info
 - ✓ Connect/disconnect to a specific simulator
 - ✓ General hardware information
- Camera Configurations
 - ✓ Load a camera specific XML file
 - ✓ Use XML fields to configure camera parameters, according to GenCam standard naming and xml field definition and type
 - ✓ Exceptions handling and reporting in case of errors or wrong parameters
- Streaming Interface
 - ✓ Stream definitions and memory allocation/de-allocation
- Buffer data interface
 - ✓ Access and handle of each allocated buffer in memory
- Data Loading
 - ✓ Loading data to memory in a form of an image or raw data
- Camera simulation control
 - ✓ Command over simulation mode and state using an internal or external trigger
- Low level bootstrap access
 - ✓ Write/read to camera bootstrap space directly with no enforcement

- IO Control
 - ✓ Control of external IO pins
- Defines, Macros, Structures and Enumerations
 - ✓ All available parameters types and definitions that can be also used in the host application

These can be found under “<installation folder>/Camera Simulator/include”.

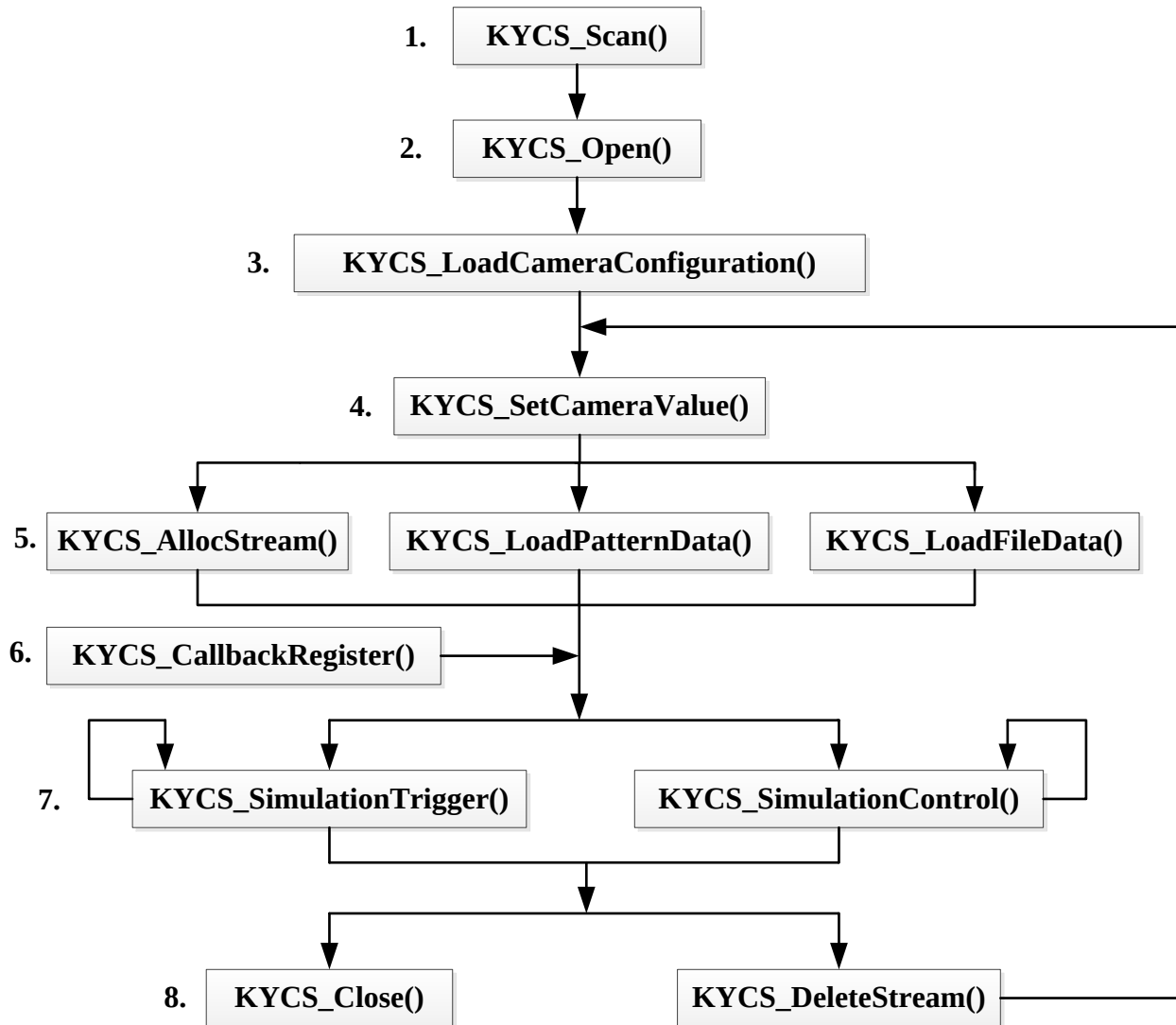
3.3 Directory and file hierarchy

The directory hierarchy of Chameleon application as can be seen after complete installation.



3.4 Function call sequence

In order for the API to carry out the desired results the following sequence of function calls should be followed:



1. Scan for simulators currently connected to the PC. This will return an array of found hardware and Virtual device PID's.
2. Open a connection to a selected device. Use the index corresponding to a simulator from the array acquired in previous step.
3. Load an XML file in order to characterize the desired camera field's structure. A camera configuration file, holding previously set values of camera fields, can also be presented.

4. Set the desired values to determine the streaming parameters using the differ methods available.
5. Allocate the memory required either by loading one or more files, or reserving memory space for loading user defined raw data.
6. A callback function can be registered before actually starting the stream. This callback will be called upon each data packet sent from the simulator. With the callback function, updated runtime information will arrive.
7. Control simulation by issuing the start/stop/pause/trigger to the simulator. This will start the arrival of callbacks to previously registered callback function.
8. If you no longer want to continue the simulation, close the connection to chosen device. If a different simulation is needed, previously streaming information is to be deleted and a new stream is to be allocated once again. Set the new camera parameters and follow steps 4-7 as described above.

3.5 Stream defining camera parameters

Several camera parameters directly effect the streaming output. These standard Gen<i>Cam parameters can be set using one or more of the [KYCS_SetCameraValue\(\)](#) different methods.

“Width” – the width of the image in pixels

“Height” – the height of the image in pixels

“PixelFormat” – the pixel type, sub-type and data color width.

Supported formats: “Mono8”, “Mono10”, “Mono12”, “Mono14”, “Mono16”, “RGB8”, “RGB10”, “RGB12”, “RGB14”, “RGB16”, “RGBA8”, “RGBA10”, “RGBA12”, “RGBA14”, “RGBA16”, “BayerGR8”, “BayerGR10”, “BayerGR12”, “BayerGR14”, “BayerGR16”, “BayerRG8”, “BayerRG10”, “BayerRG12”, “BayerRG14”, “BayerRG16”, “BayerGB8”, “BayerGB10”, “BayerGB12”, “BayerGB14”, “BayerGB16”, “BayerBG8”, “BayerBG10”, “BayerBG12”, “BayerBG14”, “BayerBG16”, “YUV444_8”, “YUV444_10”, “YUV444_12”, “YUV444_14”, “YUV444_16”, “YCbCr601_444_8”, “YCbCr601_444_10”, “YCbCr601_444_12”, “YCbCr601_444_14”, “YCbCr601_444_16”, “YCbCr709_444_8”, “YCbCr709_444_10”, “YCbCr709_444_12”, “YCbCr709_444_14”, “YCbCr709_444_16”

“AcquisitionMode” – the mode of the simulator operation

Supported acquisition modes: “Continuous”, “SingleFrame”, “MultiFrame”

“AcquisitionFrameCount” – number of frames to be sent in MultiFrame AcquisitionMode

“AcquisitionFrameRate” – frame rate for frame acquisition (number of frames per second)

“AcquisitionLineRate” – frame rate for line acquisition (number of lines per second)

“[SimulationFormat](#)” – this feature indicates the simulation scan format used during the acquisition.

Available configurations: “FrameScan”, “LineScan”

“[SimulationTriggerSource](#)” – This feature indicates the source that triggers frames transport in triggered mode. For available trigger sources refer to [KY_IO](#) bank.

“[TriggerMode](#)” – Controls the transmission trigger mode of the device, whether it’s free running or awaiting trigger to start a data packet transmission.

Available configurations: “FreeRun”, “Triggered”

3.6 Continues data stream

The simulator supports a continues stream of user defined raw data. In order to successfully configure any continues stream (e.g. a stream from a video source) the following steps should be taken:

1. A connection to a simulator should be opened using [KYCS_Open\(\)](#).
2. An XML file should be loaded using [KYCS_LoadCameraConfiguration\(\)](#). Also chosen camera parameters may be configured at this point, using the [KYCS_SetCameraValue\(\)](#) different functions.
3. A stream, consists of several frames, should be allocated using [KYCS_AllocStream\(\)](#).
4. A callback should be registered in order to monitor the outgoing frames, using [KYCS_CallbackRegister\(\)](#).
5. At least, the first 2 frames are to be filled with data; with the help of [KYCS_BufferGetPtr\(\)](#) to get the memory space of the chosen frames.
6. Start of stream is to be issued using [KYCS_SimulationControl\(\)](#).
7. Upon each callback call, the Next frame memory space should be loaded with data. Frame memory space can be acquired using [KYCS_BufferGetPtr\(\)](#) and chosen frame size can be acquired using [KYCS_BufferGetSize\(\)](#). User raw data should be copied to chosen memory; also it shouldn’t exceed the chosen frame size.

Next frame index can be calculated with the help of “currentindex” field (indicates the Nth frame that was currently send) in [CSRUNTIME](#) structure. The Next frame to load should be at least 1 frame apart from “currentindex”(e.g if currentindex = 3 then Next frame should be at least 5). The original number of frames should be taken into consideration so Next frame will not exceed it.

For example, this formula can be used to determine the Next frame to load:

$$\langle \begin{matrix} Next \\ Frame \end{matrix} \rangle = (\langle CSRUNTIME.currentindex \rangle + 2) \% \langle \begin{matrix} Total \\ number\ of \\ frames \end{matrix} \rangle$$

4.1 KYCS_Scan()

Scans for simulators currently connected to the PC PCIe slots and available Virtual simulators.

```
int KYCS_Scan(  
    unsigned int *pids_info,  
    int count) ;
```

Parameter name	Type	Description
pids_info	unsigned int*	Pointer to <pid> array of scanned devices. <i>Please see remarks!</i>
count	int	Number of devices to assign to pids_info array (assume pids_info array is valid)

Return value

pids_info array is filled with each simulator Product ID (pid).

Returns the number of connected hardware and virtual simulators.

Remarks

If pids_info parameter is called with nullptr, pids_info array will not be filled and the function will only return number of connected and virtual simulators.

Example code

```
unsigned int * info = nullptr;  
unsigned int infosize = 0;  
  
infosize = KYCS_Scan(nullptr, 0);  
info = (unsigned int *) malloc ( sizeof(unsigned int) * infosize );  
if(info != nullptr)  
{  
    infosize = KYCS_Scan(info,infosize); // Scans for simulators currently connected to PC.  
                                         // Returns Product ID array of found simulators  
}
```

4.2 KYCS_Open()

Connect to a specific simulator. Registers needed routines for stream operation, initializes all associated data and reserves space for connection.

```
CSHANDLE KYCS_Open(
    int index );
```

Parameter name	Type	Description
index	int	The index, from scan result array acquired with KYCS_Scan() function, of the camera simulator device to open.

Return value

Returns an API handle to camera simulator device.

4.3 KYCS_Close()

Close simulator specified by its handle. Unregisters all assigned routines and initializes all data associated with specified connection.

```
CSSTATUS KYCS_Close(
    CSHANDLE handle);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator

Return value

[CSSTATUS](#) - Simulator status and error report.

4.4 KYCS_Pid2Name()

Converts Product ID (pid) to its designated device name.

```
const char* KYCS_Pid2Name(
    unsigned int pid);
```

Parameter name	Type	Description
pid	unsigned int	Product id

Return value

The name of simulator issued by the specified pid.

4.5 KYCS_GetInfo()

Retrieves simulator hardware and firmware information.

```
CSSTATUS KYCS_GetInfo(
    CSHANDLE handle,
    CSINFO *info);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
info	CSINFO *	Pointer to device information structure

Return value

[CSSTATUS](#) - Simulator status and error report.

Example code

```
CSINFO device_info;
KYCS_GetInfo(handle, &device_info);

cout << showbase << internal << setfill('0');
cout << "Device serial number: " << hex << setw(10) << device_info.serial << dec << "\n";
cout << "Device revision: " << device_info.revision << "\n";
cout << "Device firmware version: " << (uint16)(device_info.fw_version >>16) << "."
<< (uint16)device_info.fw_version << "\n";
cout << "Device memory size: " << device_info.memory_size << "\n";
cout << "Device max links: " << device_info.max_links << "\n";
```

5.1 KYCS_LoadCameraConfiguration()

Loads the chosen XML into the simulator. Also sets default values and saved configurations from camera, if available. Camera configuration file can be created from Chameleon App, with file extension of “.kyccf”.

```
CSSTATUS KYCS_LoadCameraConfiguration(
    CSHANDLE handle,
    char *defFile,
    char* valuesFile);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
defFile	char*	Path to camera XML file. <i>Please see remarks!</i>
valuesFile	char*	Path to a camera configuration file. Contains saved data for camera parameters. This file can be created from Chameleon App. In case of NULL pointer , default values are loaded. <i>Please see remarks!</i>

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

If the XML file doesn’t load correctly, an error probably exists in the chosen file. In this case the XML file structure must be repaired or another XML file is to be loaded.

This file can be taken from KAYA example camera definition or from the custom camera you want to simulate.

If any camera configuration isn’t updated, the saved values file might be absent, corrupt or the parameter in question might be missing from the mounted file.

Example code

Example XML file can be found under “<installation folder>/Camera Simulator/Examples/Data”

Example camera configuration file can be found under “<installation folder>/Camera Simulator/Examples/Chameleon App”

```
KYCS_LoadCameraConfiguration(handle, "../Data/KAYA_Chameleon.xml",
    "../Chameleon App/Chameleon_example_camera_configuration.kyccf");
```

5.2 KYCS_SetCameraValue()

Set camera configuration field value. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValue(
    CSHANDLE handle,
    const char *paramName,
    void *paramValue);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
paramValue	void*	A pointer to camera configuration value

Return value

[CSSTATUS](#) - Simulator status and error report.

5.2.1 KYCS_SetCameraValueInt()

Set camera configuration field value of Integer type. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValueInt(
    CSHANDLE handle,
    const char *paramName,
    long long value);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
value	long long	Value of chosen camera configuration

Return value

[CSSTATUS](#) - Simulator status and error report.

5.2.2 KYCS_SetCameraValueFloat()

Set camera configuration field value of Float type. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValueFloat(
    CSHANDLE handle,
    const char *paramName,
    float value);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
value	float	Floating point value of chosen camera configuration

Return value

[CSSTATUS](#) - Simulator status and error report.

5.2.3 KYCS_SetCameraValueBool()

Set camera configuration field value of Boolean type. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValueBool(
    CSHANDLE handle,
    const char *paramName,
    KYBOOL value);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
value	KYBOOL	Boolean value of chosen camera configuration

Return value

[CSSTATUS](#) - Simulator status and error report.

5.2.4 KYCS_SetCameraValueEnum()

Set camera configuration field value of Enumeration type by their numeric value. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValueEnum(
    CSHANDLE handle,
    const char *paramName,
    long long value);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
value	long long	Enumeration value of chosen camera configuration

Return value

[CSSTATUS](#) - Simulator status and error report.

Example code

The following example shows how to set the pixel format to mono 8bit (numeric value of 0x101 according to Gen<i>Cam standard).

```
long long pixel_format_value = 0x101;           // mono 8bit format
KYCS_SetCameraValueEnum(handle, "PixelFormat", pixel_format_value);
```

5.2.5 KYCS_ExecuteCommand()

Execute camera command; applicable for values of Command type. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_ExecuteCommand(
    CSHANDLE handle,
    const char *paramName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter

Return value

[CSSTATUS](#) - Simulator status and error report.

5.2.6 KYCS_SetCameraValueString()

Set camera configuration field value of String type. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValueString(
    CSHANDLE handle,
    const char *paramName,
    const char* value);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
value	const char*	String value of chosen camera configuration

Return value

[CSSTATUS](#) - Simulator status and error report.

5.2.7 KYCS_SetCameraValueEnum_ByValueName()

Set camera configuration enumeration field by field name and enumeration name. According to Gen<i>Cam standard naming and xml field definition and type.

```
CSSTATUS KYCS_SetCameraValueEnum_ByValueName(
    CSHANDLE handle,
    const char *paramName,
    const char *paramValueName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
paramValueName	const char*	Name of parameter enumeration choice

Return value

[CSSTATUS](#) - Simulator status and error report.

Example code

This example demonstrates how to set the Gen<i>Cam enumeration field named “AcquisitionMode” to 1 of its enumeration options “Continuous”.

```
KYCS_SetCameraValueEnum_ByValueName(handle, “AcquisitionMode”, “Continuous”);
```

5.3 KYCS_GetCameraValueType()

Get the camera configuration field type.

```
KY_CAM_PROPERTY_TYPE KYCS_GetCameraValueType(
    CSHANDLE handle,
    const char *paramName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter

Return value

Type of camera configuration as described in [KY_CAM_PROPERTY_TYPE](#) enumeration.

5.4 KYCS_GetCameraValue()

Get camera configuration field value.

```
CSSTATUS KYCS_GetCameraValue(
    CSHANDLE handle,
    const char *paramName,
    void *paramValue);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter
paramValue	void*	A pointer to camera configuration value

Return value

[CSSTATUS](#) - Simulator status and error report.

5.4.1 KYCS_GetCameraValueInt()

Get camera configuration value of Integer type field. According to Gen<i>Cam standard naming and xml field definition and type.

```
long long KYCS_GetCameraValueInt(
    CSHANDLE handle,
    const char *paramName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter

Return value

Integer value of camera configuration field of integer type.

5.4.2 KYCS_GetCameraValueEnum()

Get camera configuration value of Enumeration type field. According to Gen<i>Cam standard naming and xml field definition and type.

```
long long KYCS_GetCameraValueEnum(
    CSHANDLE handle,
    const char *paramName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter

Return value

Integer value of camera configuration field of enumeration type.

Example code

```
VIDEO\_PIXELIF pixelif;
long long pixelif64 = KYCS_GetCameraValueEnum(handle, "PixelFormat");
pixelif.data_width = (VIDEO\_DATA\_WIDTH)(0xF & pixelif64);
pixelif.data_subtype = (VIDEO\_DATA\_SUBTYPE)(0xF & (pixelif64>>4));
pixelif.data_type = (VIDEO\_DATA\_TYPE)(0xFF & (pixelif64>>8));
```

5.4.3 KYCS_GetCameraValueFloat()

Get camera configuration value of Float type field. According to Gen<i>Cam standard naming and xml field definition and type.

```
double KYCS_GetCameraValueFloat(
    CSHANDLE handle,
    const char *paramName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter

Return value

Floating point value of camera configuration field of Float type.

5.4.4 KYCS_GetCameraValueBool()

Get camera configuration value of Boolean type field. According to Gen<i>Cam standard naming and xml field definition and type.

```
KYBOOL KYCS_GetCameraValueBool(
    CSHANDLE handle,
    const char *paramName);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
paramName	const char*	Name of camera configuration parameter

Return value

Boolean value of camera configuration field of Boolean type.

5.4.5 KYCS_GetCameraValueString()

Get camera configuration value of String type field. ***Please see remarks!***

```
CSSTATUS KYCS_GetCameraValueString(
    CSHANDLE handle,
    const char *paramName,
    char** ptr);
```

Parameter name	Type	Description
Handle	CSHANDLE	API handle to chosen simulator
paramName	char*	Name of camera configuration parameter

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

This function allocates memory for string and caller is responsible for releasing this memory.

Example code

```
char* string_value;
const char* field_name = "DeviceVendorName";

if(CSSTATUS_OK == KYCS_GetCameraValueString(field_name , &string_value);
{
    cout << "Camera configuration value is" << string_value;
    free(string_value);
}
```

6.1 KYCS_AllocStream()

Allocates the needed space in memory, assigns it to simulator according to specified camera values and number of specified frames. *Please see remarks!*

```
CSSTATUS KYCS_AllocStream(  
    CSHANDLE handle,  
    unsigned int frames);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
frames	unsigned int	Total number of frames to allocate

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

The [KYCS_AllocStream\(\)](#) is to be called only after an XML file was loaded and all wanted camera parameters were set. Setting NEW camera parameters after calling [KYCS_AllocStream\(\)](#) may cause unexpected behavior.

6.2 KYCS_DeleteStream()

Deletes and initializes streaming data for the simulator. The device stays connected but all streaming information, buffers and camera values are initialized.

```
CSSTATUS KYCS_DeleteStream(  
    CSHANDLE handle);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator

Return value

[CSSTATUS](#) - Simulator status and error report.

7.1 KYCS_BufferGetPtr()

Retrieves pointer to specific frame of data. Works after a stream was already allocated, using [KYCS_AllocStream\(\)](#) function. This frame can be filled with data, available data size can be retrieved using [KYCS_BufferGetSize\(\)](#) function.

```
void* KYCS_BufferGetPtr(  
    CSHANDLE handle,  
    unsigned int frame);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
frame	unsigned int	The frame index, to which frame memory space access is needed.

Return value

Retrieves pointer to data buffer of specified frame.

7.2 KYCS_BufferGetSize()

Retrieves the size, in bytes, of a frame in chosen stream buffer. The function will return size only for already allocated stream, using [KYCS_AllocStream\(\)](#) function.

```
unsigned long long KYCS_BufferGetSize (  
    CSHANDLE handle);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator

Return value

The size of each frame in chosen stream buffer.

In case of wrong handle or unallocated stream, the function will return - 1 value.

7.3 KYCS_BufferGetFrameIndex()

Retrieves the index of the last acquired frame in chosen stream buffer.

```
unsigned int KYCS_BufferGetFrameIndex(  
    CSHANDLE handle);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator

Return value

The index of the last frame sent of the chosen stream buffer.

In case of wrong handle or unallocated stream, the function will return -1 value.

8.1 KYCS_LoadPatternData()

Allocates the needed space in memory and commits it to stream as a video source for simulation. Several patterns types are available for generation. Patterns image format is defined by the camera configuration parameters regardless whether it's colored or non-colored pattern.

```
CSSTATUS KYCS_LoadPatternData(  
    CSHANDLE handle,  
    PATTERN_TYPE type,  
    uint16 *FixedPatternColor);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
type	PATTERN_TYPE	Pattern type to be simulated.
FixedPatternColor	Unsigned short *	A pointer to an array of 16bit (Little Endian) aligned color channels.

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

In case a fixed pattern (PATTERN_FIXED) is to be generated, a color should be specified; whether an 8, 10, 12, 14 or 16bit color plane is chosen, the array values should be 16bit values, cropped to the right bit width.

Example code

This example demonstrates a generation of fixed pattern type with RGBA 12 bit color values.

```
// create the color array with following color values
// Red = 0x0F11, Green = 0x0F22, Blue = 0x0F33, Transparency = 0x0800(half transparent)
unsigned short FixedPatternColor[4] = {0x0F11, 0x0F22, 0x0F33, 0x0800};

KYCS_SetCameraValueEnum_ByValueName(handle, "PixelFormat" , "RGBA12");

// load the chosen pattern to simulator
if(CSSTATUS_OK == KYCS_LoadPatternData(handle,
                                     PATTERN_FIXED,
                                     FixedPatternColor)
{
    printf("Pattern was loaded successfully\n");
}
```

8.2 KYCS_LoadFileData()

Allocates the needed space in memory, and commits it to stream as a video source for simulation. Image types “.bmp” , “.tif” , “.pgn”, and “.raw” are supported. A RAW file may contain several frames; number of frames is calculated according to image format configurations and RAW file size.

```
CSSTATUS KYCS_LoadFileData(
    CSHANDLE handle,
    const char* file,
    const char* type,
    int frames);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
file	const char*	The path of chosen image file
type	const char*	Type of image file: “.bmp”, “.tif”, “.png” or “.raw”
frames	int	Number of frames to generate

Return value

[CSSTATUS](#) - Simulator status and error report.

Example code

Example image can be found under “<installation folder>/Camera Simulator/Examples/Data”

```
if(CSSTATUS_OK == KYCS_LoadFileData(handle,
                                     "../Data/example_rgb_8bit.bmp",
                                     "bmp",
                                     1)) // load an image file to the simulator
{
    printf("File was loaded successfully\n");
}
```

9.1 KYCS_CallbackRegister()

Registers a run-time callback function. Used to track simulation stream status in run-time. This callback will be called each time a frame has been sent.

```
CSSTATUS KYCS_CallbackRegister(  
    CSHANDLE handle,  
    CSCallback userFunc,  
    void* userContext);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
userFunc	CSCallback	Callback function prototype
userContext	void*	(optional) Pointer to user context. Afterwards this pointer is retrieved when the callback is issued. Helps to determine the origin of stream in host application.

Return value

[CSSTATUS](#) - Simulator status and error report.

9.2 KYCS_GetRuntimeInfo()

Get simulation runtime information. Shows the state of current simulation stream, the data buffer, last frame sent and other runtime status.

```
CSSTATUS KYCS_GetRuntimeInfo(  
    CSHANDLE handle,  
    CSRUNTIME* runtimeinfo);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
runtimeinfo	CSRUNTIME *	Pointer to runtime structure. This value is filled with simulation runtime data.

Return value

runtimeinfo is assigned the simulation runtime information.

[CSSTATUS](#) - Simulator status and error report.

Example code

```
CSRUNTIME runtime_info;
KYCS_GetRuntimeInfo(handle, &runtime_info);
cout << "Last frame sent was: " << runtime_info.currentindex + 1;
cout << "Sending data at: " << runtime_info.actual_fps << " frames per second";
```

9.3 KYCS_SimulationControl()

Issues a command to the simulation. This will change the state of simulation and start, stop or pause the current stream.

```
CSSTATUS KYCS_SimulationControl(
    CSHANDLE handle,
    SIM_CONTROL control);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
control	SIM_CONTROL	Defines the simulation command to be issued.

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

This command should be issued only after an XML file was loaded, camera parameters set and a video source selected.

Example code

This example shows how to issue the start of simulation. If start is successful, runtime callbacks will start to arrive at chosen rate.

```
if(CSSTATUS_OK == KYCS_SimulationControl (handle, SIM_START);
{
    cout << "Camera simulation has started";
}
```

9.4 KYCS_SimulationTrigger()

Issues a trigger to send a single packet. In case of frame scan cameras, each trigger event results in full frame output. In case of line scan cameras each trigger event results in full line output.

```
CSSTATUS KYCS_SimulationTrigger(
    CSHANDLE handle);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

In order to enable this functionality “Triggered” Enumeration choice must be chosen for “TriggerMode” field. To set, see reference [KAYA Simulation Control fields](#) and [KYCS_SetCameraValueEnum_ByValueName\(\)](#) function.

Example code

This example shows how to issue a send trigger to the simulator. If trigger was successfully issued, a single frame/line will be sent and a callback will be returned to host application.

For callback information see functions [KYCS_CallbackRegister\(\)](#) or [KYCS_GetRuntimeInfo\(\)](#) .

```
// Setting the camera parameters
KYCS_SetCameraValueEnum_ByValueName(handle, "TriggerMode" , "Triggered");

if(CSSTATUS_OK == KYCS_SimulationTrigger (handle))
{
    cout << "Camera simulation trigger was issued successfully";
}
```

10.1 KYCS_ReadBootstrapRegs()

Read simulator bootstrap registers as defined by CoaXPress protocol.

```
CSSTATUS KYCS_ReadBootstrapRegs(  
    CSHANDLE handle,  
    unsigned int address,  
    void* data,  
    long long size);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
address	unsigned int	The start address of the bootstrap register
data	void*	A pointer to array of bootstrap registers data
size	long long	Number of bytes to read

Return value

[CSSTATUS](#) - Simulator status and error report.

10.2 KYCS_WriteBootstrapRegs()

Write simulator bootstrap registers as defined by CoaXPress protocol.

```
CSSTATUS KYCS_WriteBootstrapRegs(  
    CSHANDLE handle,  
    unsigned int address,  
    const void* data,  
    long long size);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
address	unsigned int	The start address of the bootstrap register
data	const void*	A pointer to array of bootstrap registers data
size	long long	Number of bytes to write

Return value

[CSSTATUS](#) - Simulator status and error report.

11.1 KYCS_SetTriggerSource()

Set the trigger source for simulation from [KY_IO](#) bank. External trigger which is received from GPIO or from CoaXPress link. Used to determine the trigger when working in triggered mode. To select triggered mode see [KYCS_SimulationTrigger\(\)](#) Remarks.

```
CSSTATUS KYCS_SetTriggerSource(  
    CSHANDLE handle,  
    KY_IO source);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
source	KY_IO	Simulator I/O bank.

Return value

[CSSTATUS](#) - Simulator status and error report.

11.2 KYCS_ConfigureIO()

Determines the source and sink I/O from KY_IO bank. Connects 2 I/O pins on the simulator GPIO connector.

```
CSSTATUS KYCS_ConfigureIO(  
    CSHANDLE handle,  
    KY_IO source,  
    KY_IO sink,  
    KYBOOL Invert);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
source	KY_IO	Source I/O from simulator I/O bank.
sink	KY_IO	Sink I/O from simulator I/O bank.
invert	KYBOOL	If true the I/O would be inverted.

Return value

[CSSTATUS](#) - Simulator status and error report.

Example code

This example demonstrates how to assign Logic 0 to CoaXPress trigger 0 non - inverted

```
KYCS_ConfigureIO(handle, KY_LOGIC_0 , KY_CXPTRIG_0 , false);
```

11.3 KYCS_SetOutputEnable()

Set chosen I/O pin as output. By default all connected I/O pins are input.

```
CSSTATUS KYCS_SetOutputEnable(  
    CSHANDLE handle,  
    KY_IO sink);
```

Parameter name	Type	Description
handle	CSHANDLE	API handle to chosen simulator
sink	KY_IO	Sink I/O for operation, TTL or LVTTTL from KY_IO bank.

Return value

[CSSTATUS](#) - Simulator status and error report.

Remarks

Only TTL and LVTTTL I/O can be set as outputs.

11.4 IO Selector

Configuring IO using Gen<i>Cam interface. [KYCS_SetCameraValue\(\)](#) and all associated functions can be used to configure the simulator IO controller.

Line selector options and all configurable fields under the selected Line IO:

Output	Description	Gen<i>Cam name	Type	Possible values		Remarks
				Value	Gen<i>Cam Name	
Line Selector		LineSelector	Selector			See Table 2
Line Mode	Direction of IO pin	LineMode	Enumeration	0	Input	
				1	Output	
Line Invertor	Invert the output	LineInverter	Boolean	0 - false 1 - true		
Line Status	Report state of the line	LineStatus	Boolean	0 - false 1 - true		
Line Source	Source I/O	LineSource	Enumeration			See section 15.9

Table 1 : Line selector IO fields

Value	Output	Gen<i>Cam parameter name
1	Isolated Input 0	KY_OPTO_0
2	Isolated Input 1	KY_OPTO_1
3	Isolated Input 2	KY_OPTO_2
4	Isolated Input 3	KY_OPTO_3
5	LVDS Input 0	KY_LVDS_0
6	LVDS Input 1	KY_LVDS_1
7	LVDS Input 2	KY_LVDS_2
8	LVDS Input 3	KY_LVDS_3
9	TTL Input 0	KY_TTL_0
10	TTL Input 1	KY_TTL_1
11	TTL Input 2	KY_TTL_2
12	TTL Input 3	KY_TTL_3
13	TTL Input 4	KY_TTL_4
14	TTL Input 5	KY_TTL_5
15	TTL Input 6	KY_TTL_6
16	TTL Input 7	KY_TTL_7
17	LVTTL Input 0	KY_LVTTL_0
18	LVTTL Input 1	KY_LVTTL_1
19	LVTTL Input 2	KY_LVTTL_2
20	LVTTL Input 3	KY_LVTTL_3
21	CoaXPress trigger 0	KY_CXPTRIG_0
22	CoaXPress trigger 1	KY_CXPTRIG_1
23	CoaXPress trigger 2	KY_CXPTRIG_2

24	CoaXPress trigger 3	KY_CXPTRIG_3
25	Trigger to logic 0	KY_LOGICTRIG_0
26	Trigger to logic 1	KY_LOGICTRIG_1
27	Trigger to logic 2	KY_LOGICTRIG_2
28	Trigger to logic 3	KY_LOGICTRIG_3
32	CoaXPress GPIO 0	KY_CXP0_IO_0
33	CoaXPress GPIO 1	KY_CXP0_IO_1
34	CoaXPress GPIO 2	KY_CXP0_IO_2
35	CoaXPress GPIO 3	KY_CXP0_IO_3

Table 2 : Line selection options

Example code

This example demonstrates how set the simulator to be triggered via CoaXPress link 0

```
// setting the Line Selector to Logic trigger
KYCS_SetCameraValueEnum_ByValueName ( handle, "LineSelector", "KY_LOGICTRIG_0" );
// setting the source of Logic trigger to be link 0
KYCS_SetCameraValueEnum_ByValueName ( handle, "LineSource", "KY_CXP0_IO_0" );
// Setting the trigger to be input
KYCS_SetCameraValueEnum( handle, "LineMode", 0 );
```

12.1 Analog control options

The analog signal V_{OD} and Pre-Emphasis Levels can be modified using Gen<i>Cam interface.

Output	Description	Gen<i>Cam name	Type	Possible values		Remarks
				Value	Gen<i>Cam Name	
V_{OD}	V_{OD} differential peak to peak typical	Vod	Enumeration	9 to 60		See Table 4
Pre Emphasis Levels	Transmitter Pre-Emphasis Levels	PreEmphasis	Enumeration	0 to 31		See Table 5

Table 3 : Available configurations for analog control fields

Symbol	V_{OD} Settings	V_{OD} Value(mV)	Gen<i>Cam Name	V_{OD} Settings	V_{OD} Value(mV)	Gen<i>Cam Name
V_{OD} differential peak to peak typical				34	680	VOD_680
				35	700	VOD_700
				36	720	VOD_720
	9	180	VOD_180	37	740	VOD_740
	10	200	VOD_200	38	760	VOD_760
	11	220	VOD_220	39	780	VOD_780
	12	240	VOD_240	40	800	VOD_800
	13	260	VOD_260	41	820	VOD_820
	14	280	VOD_280	42	840	VOD_840
	15	300	VOD_300	43	860	VOD_860
	16	320	VOD_320	44	880	VOD_880
	17	340	VOD_340	45	900	VOD_900
	18	360	VOD_360	46	920	VOD_920
	19	380	VOD_380	47	940	VOD_940
	20	400	VOD_400	48	960	VOD_960
	21	420	VOD_420	49	980	VOD_980
	22	440	VOD_440	50	1000	VOD_1000
	23	460	VOD_460	51	1020	VOD_1020
	24	480	VOD_480	52	1040	VOD_1040
	25	500	VOD_500	53	1060	VOD_1060
	26	520	VOD_520	54	1080	VOD_1080
	27	540	VOD_540	55	1100	VOD_1100
	28	560	VOD_560	56	1120	VOD_1120
	29	580	VOD_580	57	1140	VOD_1140
	30	600	VOD_600	58	1160	VOD_1160
	31	620	VOD_620	59	1180	VOD_1180
	32	640	VOD_640	60	1200	VOD_1200
	33	660	VOD_660			

Table 4 : Typical TX V_{OD} Setting for Transceiver Channels with termination of 100 Ω

Value	Gen<i>Cam Name	V _{OD} Setting							Unit
		10 (200 mV)	20 (400 mV)	30 (600 mV)	35 (700 mV)	40 (800 mV)	45 (900 mV)	50 (1000 mV)	
0	PRE_EMPHASIS_0	0	0	0	0	0	0	0	dB
1	PRE_EMPHASIS_1	1.97	0.88	0.43	0.32	0.24	0.19	0.13	dB
2	PRE_EMPHASIS_2	3.58	1.67	0.95	0.76	0.61	0.5	0.41	dB
3	PRE_EMPHASIS_3	5.35	2.48	1.49	1.2	1	0.83	0.69	dB
4	PRE_EMPHASIS_4	7.27	3.31	2	1.63	1.36	1.14	0.96	dB
5	PRE_EMPHASIS_5	-	4.19	2.55	2.1	1.76	1.49	1.26	dB
6	PRE_EMPHASIS_6	-	5.08	3.11	2.56	2.17	1.83	1.56	dB
7	PRE_EMPHASIS_7	-	5.99	3.71	3.06	2.58	2.18	1.87	dB
8	PRE_EMPHASIS_8	-	6.92	4.22	3.47	2.93	2.48	2.11	dB
9	PRE_EMPHASIS_9	-	7.92	4.86	4	3.38	2.87	2.46	dB
10	PRE_EMPHASIS_10	-	9.04	5.46	4.51	3.79	3.23	2.77	dB
11	PRE_EMPHASIS_11	-	10.2	6.09	5.01	4.23	3.61	-	dB
12	PRE_EMPHASIS_12	-	11.56	6.74	5.51	4.68	3.97	-	dB
13	PRE_EMPHASIS_13	-	12.9	7.44	6.1	5.12	4.36	-	dB
14	PRE_EMPHASIS_14	-	14.44	8.12	6.64	5.57	4.76	-	dB
15	PRE_EMPHASIS_15	-	-	8.87	7.21	6.06	5.14	-	dB
16	PRE_EMPHASIS_16	-	-	9.56	7.73	6.49	-	-	dB
17	PRE_EMPHASIS_17	-	-	10.43	8.39	7.02	-	-	dB
18	PRE_EMPHASIS_18	-	-	11.23	9.03	7.52	-	-	dB
19	PRE_EMPHASIS_19	-	-	12.18	9.7	8.02	-	-	dB
20	PRE_EMPHASIS_20	-	-	13.17	10.34	8.59	-	-	dB
21	PRE_EMPHASIS_21	-	-	14.2	11.1	-	-	-	dB
22	PRE_EMPHASIS_22	-	-	15.38	11.87	-	-	-	dB
23	PRE_EMPHASIS_23	-	-	-	12.67	-	-	-	dB
24	PRE_EMPHASIS_24	-	-	-	13.48	-	-	-	dB
25	PRE_EMPHASIS_25	-	-	-	14.37	-	-	-	dB
26	PRE_EMPHASIS_26	-	-	-	-	-	-	-	dB
27	PRE_EMPHASIS_27	-	-	-	-	-	-	-	dB
28	PRE_EMPHASIS_28	-	-	-	-	-	-	-	dB
29	PRE_EMPHASIS_29	-	-	-	-	-	-	-	dB
30	PRE_EMPHASIS_30	-	-	-	-	-	-	-	dB
31	PRE_EMPHASIS_31	-	-	-	-	-	-	-	dB

Table 5 : Transmitter Pre-Emphasis Levels

13.1 API handles

CSHANDLE - API handle for simulator

13.2 KYBOOL

Definition for simple C code implementation for Boolean values.

```
typedef unsigned char KYBOOL;  
#define KYTRUE 1      // determine a true value  
#define KYFALSE 0    // determine a false value
```

13.3 CSCallback

Runtime callback function prototype, callback issued while simulation is transmitting data or held by trigger.

```
typedef void (*CSCallback)(  
    CSRUNTIME* runtime,  
    void* userContext);
```

14.1 CSINFO

Hardware information, structure for connected device information, firmware version and other general details.

Structure Field	Type	Description
fw_version	float	version of firmware on the device
serial	unsigned int	serial number of the device
revision	unsigned int	revision of the device
memory_size	unsigned int	Device available memory size in bytes
max_links	unsigned int	number of available links on the device
core_temperature	unsigned int	temperature of device core
PCIeGeneration	unsigned int	PCIe generation
PCIeLanes	unsigned int	number of available PCIe lanes

14.2 CSRUNTIME

Simulation runtime details, reflects the state of data transmission while the simulation is running.

Structure Field	Type	Description
State	SIM_STATE	Current simulator state
frames_send	unsigned int	Total number of frames sent
actual_fps	unsigned int	Frame rate (frames per second)
Size	unsigned int	Size of each frame in bytes
Currentindex	unsigned int	Indicates the Nth frame that was currently send.
Ptr	void*	Pointer to current frame the data
simulator_handle	CSHANDLE	Handle for simulation that generated this interrupt

14.3 VIDEO_PIXELIF

The pixel format code is formed as shown in Table 24 of JIJA CXP document. Note that the value 0x0000 is reserved for “RAW” data that does not match any defined format, such as user-specific formats.

```
typedef struct _video_pixelif
{
    VIDEO_DATA_WIDTH    data_width    : 4;    // Data Width
    VIDEO_DATA_SUBTYPE  data_subtype  : 4;    // Sub-type
    VIDEO_DATA_TYPE      data_type     : 8;    // Data Type
}VIDEO_PIXELIF;
```

Structure Field	Type	Description
data_width	VIDEO_DATA_WIDTH	Data Width (4 bit value)
data_subtype	VIDEO_DATA_SUBTYPE	Sub-type (4 bit value)
data_type	VIDEO_DATA_TYPE	Data Type (8 bit value)

15.1 CSSTATUS

Execution of system error and status. Defines the status returned after each function execution. While some error statuses are general some point to a specific error.

Enumeration Field	Value	Description
CSSTATUS_OK	0x2000	The operation has successfully executed
CSSTATUS_UNKNOWN_SIM_HANDLE	0x2001	Unknown API handle
CSSTATUS_HW_NOT_FOUND	0x2002	Error with hardware. Hardware function failed to execute.
CSSTATUS_BUSY	0x2003	Can't execute function at the current moment, the simulator is busy
CSSTATUS_FILE_NOT_FOUND	0x2004	Wasn't able to open file in given file path
CSSTATUS_FILE_READ_ERROR	0x2005	Wasn't able to read file, error in file or the file is too long
CSSTATUS_CONFIG_NOT_LOADED	0x2006	Can't load current camera configuration
CSSTATUS_INVALID_VALUE	0x2007	The value given as parameter is out of acceptable range
CSSTATUS_MAX_CONNECTIONS	0x2008	No more devices can be connected to system
CSSTATUS_COULD_NOT_STOP	0x2009	The simulation couldn't stop
CSSTATUS_CANNOT_LOAD_IMAGE_FILE	0x200A	Wasn't able to load image file
CSSTATUS_MEMORY_ERROR	0x200B	General memory error or an allocation has failed
CSSTATUS_UNKNOWN_SIM_CONTROL	0x200C	Wrong simulation command
CSSTATUS_WRONG_PARAMETER_NAME	0x200D	Camera configuration wasn't found
CSSTATUS_WRONG_PARAMETER_TYPE	0x200E	Wrong parameter type for camera configuration
CSSTATUS_GENICAM_EXCEPTION	0x200F	General GenCam exception
CSSTATUS_OUT_OF_RANGE_ADDRESS	0x2010	The specified address is not suitable for writing
CSSTATUS_PATH_INVALID	0x2011	Specified file couldn't be opened, wrong file path or file not found

CSSTATUS_FILE_TYPE_INVALID	0x2012	Invalid file type was entered
CSSTATUS_UNSUPPORTED_IMAGE	0x2013	Mounted image is not supported
CSSTATUS_UNSUPPORTED_IMAGE_CONVERSION	0x2014	Couldn't convert image type or couldn't rescale current image
CSSTATUS_UNSUPPORTED_DEPTH_CONVERSION	0x2015	Couldn't convert image due to unsupported color depth
CSSTATUS_INVALID_VALUES_FILE	0x2016	Invalid camera configuration values file was loaded
CSSTATUS_FILE_WRITE_ERROR	0x2017	Wasn't able to save camera configuration values file
CSSTATUS_BUFFER_NOT_LOADED	0x2018	Incompatible buffer was selected, or no buffer was allocated
CSSTATUS_UNKNOWN_ERROR	0x2FFF	Unknown error

15.2 PATTERN_TYPE

Patterns generation types. Actual pattern scaling and format is defined by values in the appropriate camera configuration fields.

Enumeration Field	Value	Description
PATTERN_XRAMP	0	Ramp pattern over X coordinate
PATTERN_XRAMP_COLOR	1	Ramp pattern over X coordinate with color (3 color format)
PATTERN_YRAMP	2	Ramp pattern over Y coordinate
PATTERN_YRAMP_COLOR	3	Ramp pattern over Y coordinate with color (3 color format)
PATTERN_XYRAMP	4	Ramp pattern over XY coordinates
PATTERN_XYRAMP_COLOR	5	Ramp pattern over XY coordinates with color (3 color format)
PATTERN_FIXED	6	Fixed color pattern

15.3 SIM_CONTROL

Simulation Control commands. Defines which command to issue to the simulation stream. Start, stop or pause the simulator.

Enumeration Field	Value	Description
SIM_STOP	0	Command to stop the simulation
SIM_START	1	Command to start the simulation
SIM_PAUSE	2	Command to pause the simulation

15.4 SIM_STATE

Simulation runtime state, defines the state of simulation at any given moment.

Enumeration Field	Value	Description
SIM_STOPPED	0	Stopped simulation state
SIM_PAUSED	1	Paused simulation state
SIM_SENDING_DATA	2	Data sending simulation state
SIM_WAITING_TRIGGER	3	Waiting for a trigger simulation state
SIM_LOADING	4	Loading data simulation state
SIM_ERROR	5	Error simulation state
SIM_UNKNOWN	-1	Unknown simulation state

15.5 KY_CAM_PROPERTY_TYPE

Gen<i>Cam field type. Camera configuration field type as stated in loaded XML file.

Enumeration Field	Value	Description
PROPERTY_TYPE_INT	0x00	Camera configuration of Integer type
PROPERTY_TYPE_BOOL	0x01	Camera configuration of Boolean type
PROPERTY_TYPE_STRING	0x02	Camera configuration of String type
PROPERTY_TYPE_FLOAT	0x03	Camera configuration of Floating Point type
PROPERTY_TYPE_ENUM	0x04	Camera configuration of Enumeration type
PROPERTY_TYPE_COMMAND	0x05	Camera configuration of Command type
PROPERTY_TYPE_UNKNOWN	-1	Camera configuration of an unknown type

15.6 VIDEO_DATA_WIDTH

Data width of the pixel, defined in section 9.4.1.2 of the JIJA CXP standard document

Enumeration Field	Value	Description
DATA_WIDTH_UNKNOWN	0x00	Unknown number of bits per pixel data
DATA_WIDTH_8BIT	0x01	8 bit per pixel data
DATA_WIDTH_10BIT	0x02	10 bit per pixel data
DATA_WIDTH_12BIT	0x03	12 bit per pixel data
DATA_WIDTH_14BIT	0x04	14 bit per pixel data
DATA_WIDTH_16BIT	0x05	16 bit per pixel data

15.7 VIDEO_DATA_TYPE

Data types of the pixel, defined in section 9.4.1 of the JIIA CXP standard document

Enumeration Field	Value	Description
DATA_TYPE_MONO	0x01	This is used for luminance data. This has no sub-types. This is defined in Table 27 of JIIA CXP standard document
DATA_TYPE_PLANAR	0x02	This is used for planar data, such as individual red, green or blue planes, additional alpha (overlay) planes, or the separate planes in YUV420. This is defined in Table 28 JIIA CXP standard document. Subtypes include all the DATA_SUBTYPE_PLANAR_xx
DATA_TYPE_BAYER	0x03	This is used for Bayer data. This is defined in Table 29 JIIA CXP standard document. Subtypes include all DATA_SUBTYPE_BAYER_xx
DATA_TYPE_RGB	0x04	This is used for RGB data, transmitted in the order red, green, blue. This has no sub-types. This is defined in Table 30 JIIA CXP standard document.
DATA_TYPE_RGBA	0x05	This is used for RGBA data, where “A” is the alpha (or overlay) plane, transmitted in the order red, green, blue, alpha. This has no sub-types. This is defined in Table 31 JIIA CXP standard document.
DATA_TYPE_YUV	0x06	This is used for YUV data. This is defined in Table 32 JIIA CXP standard document. Subtypes include all DATA_SUBTYPE_YUV_xxx
DATA_TYPE_YCBCR601	0x07	This is used for YCbCr data, as specified by ITU-R BT.601. This is defined in Table 33 JIIA CXP standard document. Subtypes include all DATA_SUBTYPE_UCBCR_xxx
DATA_TYPE_YCBCR709	0x08	This is used for YCbCr data, as specified by ITU-R BT.709. This is defined in Table 34 JIIA CXP standard document. Subtypes include all DATA_SUBTYPE_UCBCR_xxx

15.8 VIDEO_DATA_SUBTYPE

Data sub-types of the pixel, defined in section 9.4.1 of the J11A CXP standard document

Enumeration Field	Value	Description
DATA_SUBTYPE_NONE	0x00	None
DATA_SUBTYPE_PLANAR_RY	0x01	Standard usage: R, Y
DATA_SUBTYPE_PLANAR_GUCB	0x02	Standard usage: G, U, Cb
DATA_SUBTYPE_PLANAR_BVCR	0x03	Standard usage: B, V, Cr
DATA_SUBTYPE_BAYER_GR	0x01	1st line transmission order G, R. 2nd line transmission order B, G
DATA_SUBTYPE_BAYER_RG	0x02	1st line transmission order R, G. 2nd line transmission order G, B
DATA_SUBTYPE_BAYER_GB	0x03	1st line transmission order G, B. 2nd line transmission order R, G
DATA_SUBTYPE_BAYER_BG	0x04	1st line transmission order B, G. 2nd line transmission order G, R
DATA_SUBTYPE_YUV_411	0x01	Transmission order Y, Y, U, Y, Y, V
DATA_SUBTYPE_YUV_422	0x02	Transmission order Y, U, Y, V
DATA_SUBTYPE_YUV_444	0x03	Transmission order Y, U, V
DATA_SUBTYPE_YCBCR_411	0x01	Transmission order Y, Y, Cb, Y, Y, Cr
DATA_SUBTYPE_YCBCR_422	0x02	Transmission order Y, Cb, Y, Cr
DATA_SUBTYPE_YCBCR_444	0x03	Transmission order Y, Cb, Cr

15.9 KY_IO

User accessible I/Os. Can be defined as trigger source or paired for high speed transmission channels.

Enumeration Field	Value	Description
KY_LOGIC_0	0x00	Fixed Logic 0
KY_OPTO_0	0x01	Isolated Input 0 from external connector
KY_OPTO_1	0x02	Isolated Input 1 from external connector
KY_OPTO_2	0x03	Isolated Input 2 from external connector
KY_OPTO_3	0x04	Isolated Input 3 from external connector
KY_LVDS_0	0x05	LVDS Input 0 from external connector
KY_LVDS_1	0x06	LVDS Input 1 from external connector
KY_LVDS_2	0x07	LVDS Input 2 from external connector
KY_LVDS_3	0x08	LVDS Input 3 from external connector
KY_TTL_0	0x09	TTL Input 0 from external connector
KY_TTL_1	0x0A	TTL Input 1 from external connector
KY_TTL_2	0x0B	TTL Input 2 from external connector
KY_TTL_3	0x0C	TTL Input 3 from external connector
KY_TTL_4	0x0D	TTL Input 4 from external connector
KY_TTL_5	0x0E	TTL Input 5 from external connector
KY_TTL_6	0x0F	TTL Input 6 from external connector
KY_TTL_7	0x10	TTL Input 7 from external connector
KY_LVTTL_0	0x11	LVTTL Input 0 from external connector
KY_LVTTL_1	0x12	LVTTL Input 1 from external connector
KY_LVTTL_2	0x13	LVTTL Input 2 from external connector
KY_LVTTL_3	0x14	LVTTL Input 3 from external connector
KY_CXPTRIG_0	0x15	Trigger Input from CoaXPress Link
KY_CXP0_IO_0	0x20	GPIO Input 0 from CoaXPress Link
KY_CXP0_IO_1	0x21	GPIO Input 1 from CoaXPress Link
KY_CXP0_IO_2	0x22	GPIO Input 2 from CoaXPress Link
KY_CXP0_IO_3	0x23	GPIO Input 3 from CoaXPress Link
KY_LOGIC_1	0x3F	Fixed Logic 1

Additional configurations of the simulator are available using Gen<i>Cam interface.

16.1 Simulation Control

Output	Description	Gen<i>Cam name	Type	Possible values		Remarks
				Value	Gen<i>Cam name	
Link Discovery	Initial discovery speed	LinkDiscovery	Enumeration	0x00010028	DISCOVERY_CXP_1	
				0x00010038	DISCOVERY_CXP_3	
Simulation Format	Simulate Frame or Line scan cameras	SimulationFormat	Enumeration	1	FrameScan	
				2	LineScan	
Simulation Trigger Source	Trigger source of simulator to initiate acquisition	SimulationTriggerSource	Enumeration			See section 15.9 for trigger sources
Trigger Mode	Defines the mode to start the acquisition	TriggerMode	Enumeration	0	FreeRun	
				1	Triggered	

Table 6 : Simulation Control fields

16.2 Hardware Information

Parameter	Description	Gen<i>Cam name	Type	Possible values		Remarks
				Value	Gen<i>Cam name	
Firmware Version	The firmware version of the device	FirmwareVersion	Float			
Serial Number	Serial Number of the device	SerialNumber	Integer			
Device Revision	Revision of the device	DeviceRevision	Integer			
Device Memory Size	Device memory size in bytes	DeviceMemorySize	Integer			
Maximum Links	Maximum available links on the device	MaxLinks	Integer			
Device PCIe Generation	Supported generation of connected PCIe	DevicePciGeneration	Integer			
PCIe lanes	Number of connected PCIe lanes	DevicePciLanes	Integer			
Device Temperature	Device CPU Temperature	DeviceTemperature	Integer			Temperature is in °C

Table 7 : Hardware Information fields

International Distributors