**xiLIB** - LabVIEW Interface for XIMEA Cameras

Table of Contents

[Introduction 3](#_Toc354040861)

[Requirements 3](#_Toc354040862)

[Installation 3](#_Toc354040863)

[Getting Started with xiLIB 4](#_Toc354040864)

[Building Application with xiLIB 4](#_Toc354040865)

[xiCamLib 5](#_Toc354040866)

[xiCam – CloseDevice.vi 5](#_Toc354040867)

[xiCam – Create.vi 5](#_Toc354040868)

[xiCam – Get1DImageData.vi 5](#_Toc354040869)

[xiCam – Get1DImageData\_ExternalDimension.vi 6](#_Toc354040870)

[xiCam – GetNumberDevices.vi: 6](#_Toc354040871)

[xiCam – GetParam\_Float.vi 7](#_Toc354040872)

[xiCam – GetParam\_Int32.vi 7](#_Toc354040873)

[xiCam – GetParam\_String.vi 7](#_Toc354040874)

[xiCam – OpenDevice.vi 7](#_Toc354040875)

[xiCam – SetParam\_Float.vi 8](#_Toc354040876)

[xiCam – SetParam\_Int32.vi 8](#_Toc354040877)

[xiCam – StartAcquisition.vi 8](#_Toc354040878)

[xiCam – StopAcquisition.vi 8](#_Toc354040879)

[xiCam – GetLastImageParam.vi 9](#_Toc354040880)

[IMG\_PARAMS 9](#_Toc354040881)

[PRM - Basic 10](#_Toc354040882)

[PRM - Additional 10](#_Toc354040883)

[BUFF\_POLICY – Type.vi 11](#_Toc354040884)

[DOWNSAMPLING – Type.vi 11](#_Toc354040885)

[GPI\_MODE – Type.vi 12](#_Toc354040886)

[GPO\_MODE – Type.vi 12](#_Toc354040887)

[IMG\_COLOR\_FILTER\_ARRAY – Type.vi 12](#_Toc354040888)

[IMG\_FORMAT – Type.vi 13](#_Toc354040889)

[LED\_MODE – Type.vi 14](#_Toc354040890)

[PRM – GetParameter.vi 14](#_Toc354040891)

[TRG\_SOURCE – Type.vi 14](#_Toc354040892)

[xiLIB 15](#_Toc354040893)

[xiLIB – GetGPIStates.vi 15](#_Toc354040894)

[xiLIB – GetGPOStates.vi 15](#_Toc354040895)

[xiLIB – GetPRM\_INFO.vi 15](#_Toc354040896)

[xiLIB – GPIO\_Editor.vi 16](#_Toc354040897)

[xiLIB – LoadFromINI\_DIALOG.vi 16](#_Toc354040898)

[xiLIB – LoadFromINI.vi 16](#_Toc354040899)

[xiLIB – SaveToINI\_DIALOG.vi 17](#_Toc354040900)

[xiLIB – SaveToINI.vi 17](#_Toc354040901)

[xiControl.llb 18](#_Toc354040902)

# Introduction

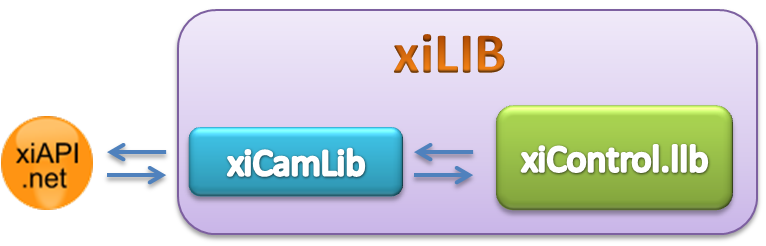
This toolkit is a LabVIEW wrapper of the original .NET based xiAPI.NET API and contains all necessary VIs for those who wish to program their XIMEA cameras in LabVIEW instead of C++ or .NET. It can fully handle all the functions provided by the original xiAPI.NET.

xiLIB consist of the following major parts:

• **xiCamLib**: Low level VIs that includes all necessary xiAPI.NET calls, DLLs; list of all usable parameters and methods

• **xiControl.llb**: High level function with user interface that is capable of: online editing camera parameters; serialize and deserialized the camera parameters to file; test software trigger mode; managing GPI/GPO states of the device; save image to HDD.

• **Utils**: Additional files referred in xiCamLib and xiControl.llb.



Utils

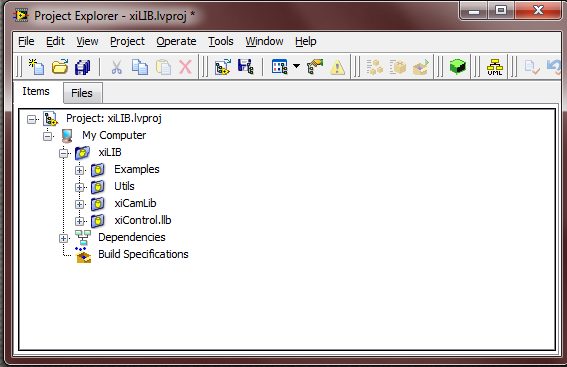
Requirements

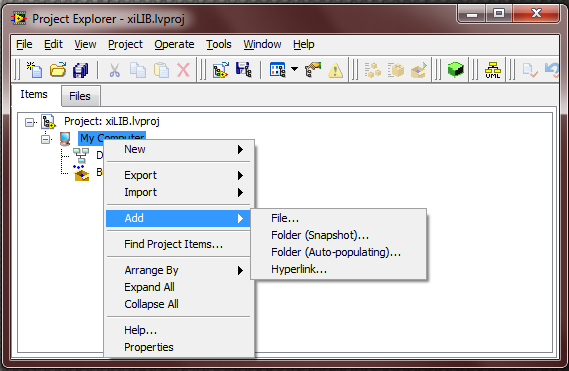
* **National Instruments** - [LabVIEW 2013](http://www.ni.com/labview) and later (Earlier version available on request)
* **National Instruments** - Vision Development Module
* Microsoft .NET Framework 3.5 - [Link](http://www.microsoft.com/en-us/download/details.aspx?id=21)

In order to achieve maximum frame rate and application performance it is suggested to use **NI IMAQ** functions that are located in National Instruments - Vision Development Module.

# Installation

In order to use xiLIB you have to add the xiLIB folder as **Snapshot** or **Auto-populating folder** to your LabVIEW project.

Right click on **My Computer** in LabVIEW Project Explorer and under the **Add** menu item select Snapshot or Auto-populating.



**<It is suggested to Mass Compile your project after xiLIB is added at first. You can easily do that by Right Clicking on the Project Icon in the LabVIEW Project Explorer and select Mass Compile from the menu.>**

# Getting Started with xiLIB

* As the xiLIB files added to LabVIEW project navigate to xiLIB\xiControl.llb\xiC - Main.vi to launch xiControl vi. Connect your XIMEA Device to your computer and Run the VI to adjust camera settings, display image and save camera parameters for later use.
* Navigate to xiLIB\xiCam\xiCam - AllVIs.vi to see all VIs that xiCamLib contains.
* Open the attached example(s) and read the comments on block diagram.

# Building Application with xiLIB

You can easily create executable from xiLIB with LabVIEW Application Builder but you have to take care of one restriction: LabVIEW Application builder creates a folder called "data" where only the xiApi.NET.dll file exists and if you start your exe file no device can be found. (Error message appears.)

Manually copy the following files next to the xiApi.NET.dll in order to run your application without any error. If case of **32Bit LabVIEW** copy the content of **xiCamLib\API\x86** or if your LabVIEW version is **64Bit** copy the **xiCamLib\API\x64**:

|  |
| --- |
| **xiapi32.dll/ xiapi64.dll** |
| **xiApi.NET.xml** |

# 

**Before you start your builded application (\*.exe) that uses xiLIB close all any applications that uses xiAPI.NET.dll such as LabVIEW.exe.**

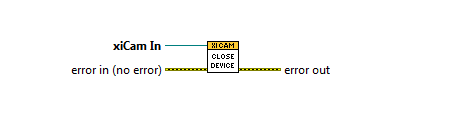
# xiCamLib

Under xiCamLib, **API** folder contains the .NET dll that LabVIEW VIs wrapping. In this version x86 version is used.

* Under **xiCam** you can find the low level VIs; name of the VIs are equal to the method that is wrapped and a prefix "xiCam".
* The **PRM** folder where .NET parameter enumeration can be found. In xiLIB the LabVIEW enumeration is implemented as Strict Type Def. control. The usage of PRM VIs presented later in this document.
* VIs with **Utils** prefix are responsible for any additional functionality such as converting an U8 image array to IMAQ Mono or Color image. **xiImageManager** is a LabVIEW programming tool called "Action Engine" that keeps the latest parameters that are necessary to convert the U8 byte array to the right image format and dimension.
* **xiCam - AllVIs.vi** lists all xiCam VIs on its Block Diagram to get a fast overview.

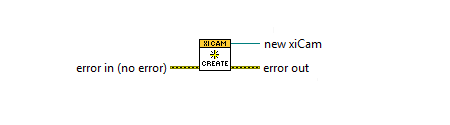
*List of xiCam VIs in alphabetic order.*

xiCam – CloseDevice.vi**:** Uninitializes the specified device and releases the allocated resources.



* **xiCam In** <.NET Refnum>: The xiCam object to be uninitialized

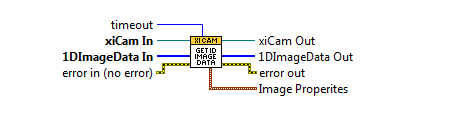
xiCam – Create.vi***:*** Creates a xiCam object.



* new xiCam <.NET Refnum>: The xiCam object created by the VI

xiCam – Get1DImageData.vi*:* Acquires the image and returns a byte array containing the image data.

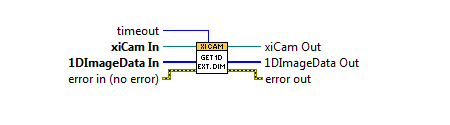
<Use this VI when PRM **values are changing** during acquisition and affects the image size and type.>



* timeout <I32>: Time interval required to wait for the image in milliseconds
* **1DImageData In <1D[U8]> :** Byte array where image data is copied.
* Image Properties: Returns current image properties. Use this value as input parameter for *Utils - Convert1DImageToImageImage.vi.*

xiCam – Get1DImageData\_ExternalDimension.vi : Acquires the image and returns a byte array containing the image data.

<Use this VI when PRM values **are kept constant** during acquisition and affects the image size and type.>

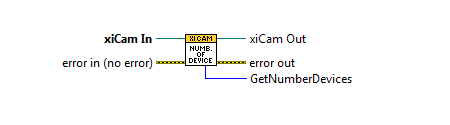
**

* timeout <I32>: Time interval required to wait for the image in milliseconds
* **1DImageData In <1D[U8]> :** Byte array where image data is copied.

< There are many workarounds how to convert byte array to image in LabVIEW for example **picture.llb\Draw Unflattened Pixmap.vi** but the experiments show **vision\Basic.llb\IMAQ ArrayToImage.vi** and **IMAQ ArrayToColorImage.vi** VIs take the least computation time.

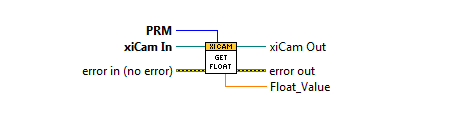
xiAPI.NET has more additional functions to grab image from the device such as **GetImage(Bitmap Image, I32 Timeout)** . If you want to do additional image processing on the bitmap, you should call .NET Lock and Unlock methods on BitmapData that requires additional time to complete. >

xiCam – GetNumberDevices.vi: Returns the number of all discovered devices. If no device presents the VI retruns an error with **Code: -1, Source: DeviceNotFound.**



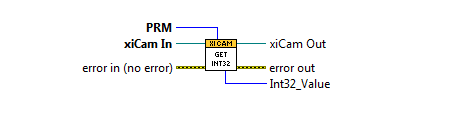
* GetNumberDevices <I32>: The number of cameras

xiCam – GetParam\_Float.vi *:* Returns the selected parameter as a Float value.



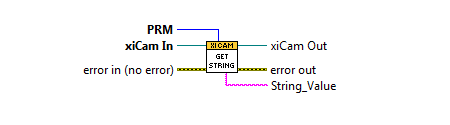
* **PRM** <typedef PRM.ctl>: Selects a parameter
* Float\_Value <SGL>: Float value of the selected parameter

xiCam – GetParam\_Int32.vi : Returns the selected parameter as an Int32 value.



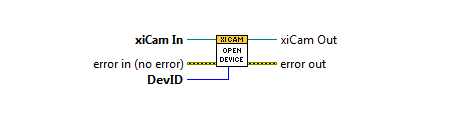
* **PRM** <typedef PRM.ctl>: Selects a parameter
* Int32\_Value <I32>: Int32 value of the selected parameter

xiCam – GetParam\_String.vi : Returns the selected parameter as a String value.



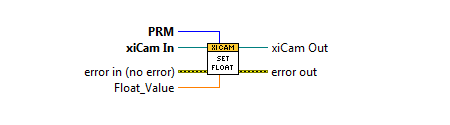
* **PRM** <typedef PRM.ctl>: Selects a parameter
* String\_Value <String>: String value of the selected parameter

xiCam – OpenDevice.vi : Initializes the device.



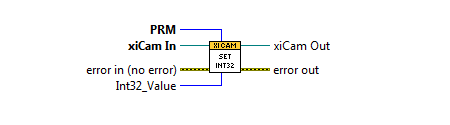
* **DevID** <Int32>: The ID of the device to be initialized

xiCam – SetParam\_Float.vi : Configures one of the device’s parameter with a Float value.



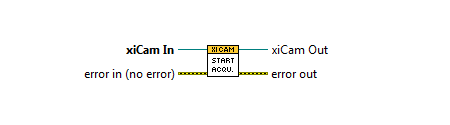
* **PRM** <typedef PRM.ctl>: Selects a parameter.
* Float\_Value <Float>: Float value of the selected parameter

xiCam – SetParam\_Int32.vi : Configures one of the device’s parameter with an Int32 value.



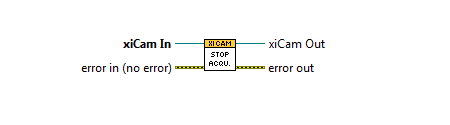
* **PRM** <typedef PRM.ctl>: Selects a parameter.
* Int32\_Value <Int>: Int32 value of the selected parameter

xiCam – StartAcquisition.vi : Starts the data acquisition on the device.

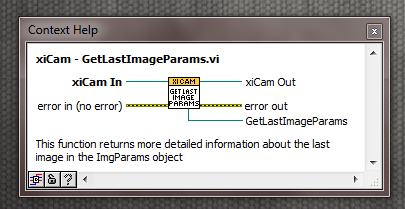


* **xiCam In** <.NET Refnum>: xiCam object
* xiCam out <.NET Refnum>: xiCam object

xiCam – StopAcquisition.vi : Stops the work cycle and data acquisition on the device.



xiCam – GetLastImageParam.vi : This function returns more detailed information about the last image in the ImgParams object.

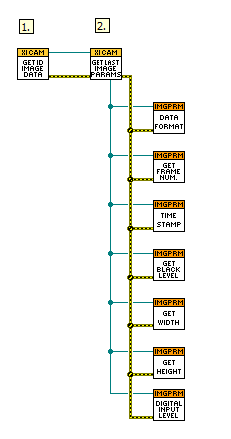


* **xiCam In** <.NET Refnum>: xiCam object
* xiCam out <.NET Refnum>: xiCam object
* GetLastImageParams<.NET Refnum>ImgParams object.

<In order to get last image attributes use IMG\_PARAMS methods under PRM\IMG\_PARAMS>

# IMG\_PARAMS

|  |  |
| --- | --- |
| Parameter Name | Description |
| IMG\_PARAMS - GetBlackLevel.vi | Returns image black level (Raw and MONO) |
| IMG\_PARAMS - GetDataFormat.vi | Returns data format of the last acquired image. |
| IMG\_PARAMS - GetDigitalnputLevel.vi | Returns GPI input level. |
| IMG\_PARAMS - GetFrameNumber.vi | Returns number of the last acquired image. |
| IMG\_PARAMS - GetHeight.vi | Returns height of the last acquired image. |
| IMG\_PARAMS - GetWidth.vi | Returns width of the last acquired image. |

**Usage:**

1. Make sure the first call is **xiCam - Get1DImageData.vi**. You can easily do this by keeping the error wire connected through xiAPI VIs.

2. Call **xiCam - GetLastImageParams.vi** that returns the ImgParams object that used to achieve more information about the last acquired image.

3. At last you can call any of IMG\_PARAMS - VIs.

# PRM - Basic

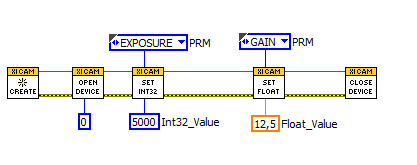
These VIs offer easy access to the camera’s parameters. The complete list of basic parameters are listed in a Strict Type Def. Enum control in **PRM.ctl**. Use this Enum as an Input value for vis:

* xiCam - GetParam\_Float.vi
* xiCam - GetParam\_Int32.vi
* xiCam - GetParam\_String.vi
* xiCam - SetParam\_Int32.vi
* xiCam - SetParam\_Float.vi

to select the basic parameter that is need to be changed or get.

In order to change the device parameter you have to use xiCam - SetParam\_Int32.vi or xiCam - SetParam\_Float.vi with PRM enumeration in the following way:

1. Select the proper camera parameter from the PRM enumeration
2. Visit the following link to check the parameter value: [LINK](http://www.ximea.com/support/wiki/apis/XiAPI_Manual) (for example the EXPOSURE parameter is I32 value but GAIN has floating type.)



Example code: How to set basic camera parameters with PRM.ctl.

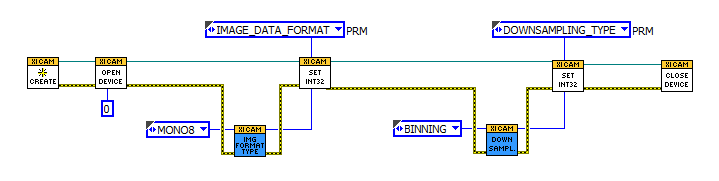
<PRM Enumeration: There are several entries in the PRM enumeration that are not listed on XIMEA official API such as: GPI\_x; GPO\_x; LED\_x, where x represents a numerical vaue. >

# PRM - Additional

|  |  |
| --- | --- |
| Parameter Name | Description |
| BUFF\_POLICY - Type.vi | Buffer Policy settings |
| DOWNSAMPLING - Type.vi | Downsampling settings. |
| GPI\_MODE - Type.vi | General Purpose Input settings. |
| GPO\_MODE - Type.vi | General Purpose Output settings. |
| IMG\_FORMAT - Type.vi | Image format. |
| LED\_MODE - Type.vi | LED settings. |
| TRG\_SOURCE - Type.vi | Camera Trigger settings. |

If you want to update one of the above listed camera parameter, the way is similar to PRM - Basic but you have to do an extra API call.

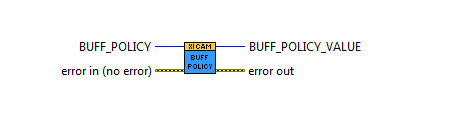
For example update the device IMAGE\_DATA \_FORMAT and DOWNSAMPLING\_TYPE, the procedure is the following:



BUFF\_POLICY – Type.vi **:** Returns the numeric value of the selected buffer policy.

**UNSAFE** : User gets pointer to internally allocated circle buffer and data may be overwritten by device.

**SAFE** : Data from device will be copied to user allocated buffer or xiApi allocated memory.

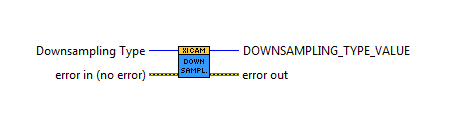


* BUFF\_POLICY <typedef BUFF\_POLICY.ctl>: The selected buffer policy
* BUFF\_POLICY\_VALUE <Int>: The value of the buffer policy

DOWNSAMPLING – Type.vi **:** Returns the numeric value of the selected downsampling type.

**BINNING** : Binned type for higher sensitivity.

**SKIPPING** : Skipped type for higher frame rates.



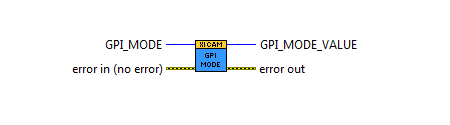
* Downsampling Type <typedef DOWNSAMPLING\_TYPE.ctl>: the selected downsampling type
* DOWNSAMPLING\_TYPE\_VALUE <Int>: The value of the downsampling type

GPI\_MODE – Type.vi **:** Returns the numeric value of the selected GPI mode.

**OFF** : Input off.

**TRIGGER** : Trigger input.

**EXT\_EVENT** : External signal input.



* GPI\_MODE <typedef GPI\_MODE.ctl>: The selected GPI mode
* GPI\_MODE\_VALUE <I32>: The value of the GPI mode

GPO\_MODE – Type.vi: Returns the numeric value of the selected GPO mode.

**XI\_GPO\_OFF**\*\*: Output is off (zero voltage or switched\_off)

**XI\_GPO\_ON**\*\*: Output is on (voltage or switched\_on)

**XI\_GPO\_FRAME\_ACTIVE**\*: Output is on while frame exposure,read,transfer

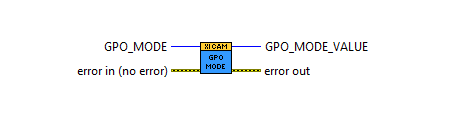
**XI\_GPO\_FRAME\_ACTIVE\_NEG**\*: Output is off while frame exposure,read,transfer

**XI\_GPO\_EXPOSURE\_ACTIVE**\*:Output is on while frame exposure

**XI\_GPO\_EXPOSURE\_ACTIVE\_NEG**\* Output is off while frame exposure

\*Note1: Modes FRAME\_ACTIVE or EXPOSURE\_ACTIVE are supported only if XI\_PRM\_TRG\_SOURCE is set to XI\_TRG\_SOFTWARE or XI\_TRG\_EDGE\_RISING or XI\_TRG\_EDGE\_FALLING.

\*\*Note2: Some camera families (e.g. MR) does not support the software control of outputs. Only one of mode: FRAME\_ACTIVE and EXPOSURE\_ACTIVE can be set.



* GPO\_MODE <typedef GPO\_MODE.ctl>: The selected GPO mode
* GPO\_MODE\_VALUE <I32>: The value of the selected GPO mode

IMG\_COLOR\_FILTER\_ARRAY – Type.vi : Returns color filter array type of RAW data.

**NONE** : B/W sensors.

**BAYER\_RGGB** : Regular Bayer RGGB readout.

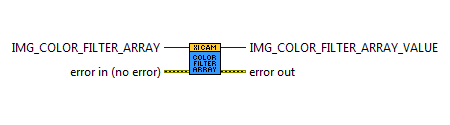
**CMYG** : AK Sony sensors.

**RGR** : 2R+G readout of RGGB sensor.

**BAYER\_BGGR** : Bayer BGGR readout.

**BAYER\_GRBG** : Bayer GRBG readout.

**BAYER\_GBRG** : Bayer GBRG readout.



* IMG\_COLOR\_FILTER\_ARRAY <typedef IMG\_COLOR\_FILTER\_ARRAY.ctl>: The selected color filter array type
* IMG\_COLOR\_FILTER\_ARRAY\_VALUE <I32>: The value of the selected type

IMG\_FORMAT – Type.vi : Returns the numeric value of the selected image format on the output. Image resolution can change after image format has been changed.

Default: MONO8

**MONO8**: [Intensity]

**MONO16** : [Intensity LSB] [Intensity MSB]

**RGB24** : [Blue][Green][Red] (Note1)

**RGB32** : [Blue][Green][Red][0]\* (Note1)

**RGB\_PLANAR** : [Red][Red]...[Green][Green]...[Blue][Blue]... (Note1)

**RAW8** : [pixel byte] raw data from sensor

**RAW16** : [pixel byte low] [pixel byte high] 16 bits raw data from sensor

\*Note1: Higher CPU processing is required when this mode is selected because color filter array processing is implemented on PC.

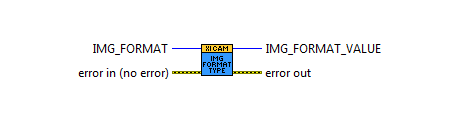
Bits alignment: Values are aligned to LSB.

Sensor bits per pixel values in modes XI\_RAW16:

10 0-1023

12 0-4095

14 0-16384



* IMG\_FORMAT <typedef IMG\_DATA\_FORMAT.ctl>: The selected data format
* IMG\_FORMAT\_VALUE <I32>: The value of the selected format

LED\_MODE – Type.vi : Returns the numeric value of the selected LED mode.

**HEARTBEAT** : Blink if link is ok (led 1), heartbeat mode (led 2).

**TRIGGER\_ACTIVE** : Blink led if trigger detected.

**EXT\_EVENT\_ACTIVE** : Blink led if external signal detected.

**ACQUISITION** : Blink led during data streaming.

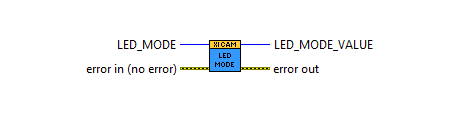
**EXPOSURE\_ACTIVE** : Blink led during sensor integration time.

**FRAME\_ACTIVE** : Blink if device busy/not busy.

**LINK** : Blink led if link is ok.

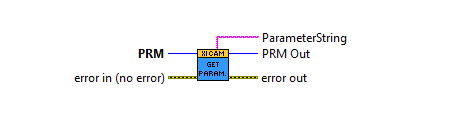
**OFF** : Set led to zero.

**ON** : Set led to one.



* LED\_MODE <LED\_MODE.ctl>: The selected LED mode
* LED\_MODE\_VALUE <Int>: The value of the selected LED mode

PRM – GetParameter.vi : Returns the name of the selected parameter. Converts LabVIEW enumeration to xiAPI parameter String.



* **PRM** <typedef PRM.ctl>: The selected parameter
* PRM out <typedef PRM.ctl>: The selected parameter
* ParameterString <String>: The name of the selected parameter

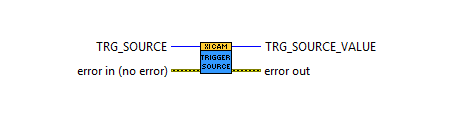
TRG\_SOURCE – Type.vi: Returns the numeric value of the selected trigger source.

**OFF** : Camera works in free run mode.

**EDGE\_RISING** : External trigger (rising edge).

**EDGE\_FALLING** : External trigger (falling edge).

**SOFTWARE** : Software(manual) trigger.

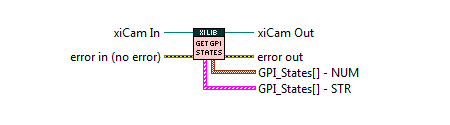


* TRG\_SOURCE <typedef TRG\_SOURCE.ctl>: The selected trigger source
* TRG\_SOURCE\_VALUE <Int>: The value of the trigger source

# xiLIB

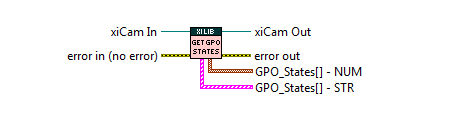
This folder contains ready-to-use VIs built from the above ones. Their purpose is to further simplify both the block diagram and both the programming process.

xiLIB – GetGPIStates.vi *:* Returns the Current GPI States of Camera. This VI checks all GPI values regarding the current camera version.



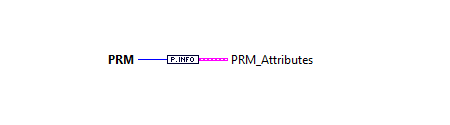
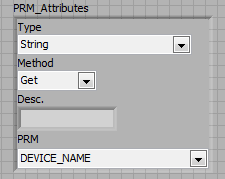
* GPI\_States[] - NUM: Returns GPI\_INDEX - GPI\_MODE value pairs as array of cluster.
* GPI\_States[] - STR: Returns GPI\_INDEX - GPI\_MODE value pairs as array of cluster.

xiLIB – GetGPOStates.vi *:* Returns the Current GPO States of Camera. This VI check all GPO values regarding the current camera version.

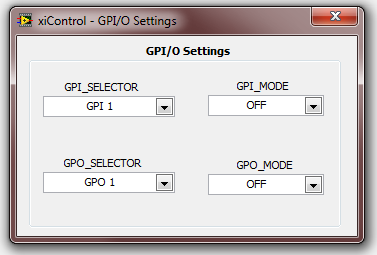


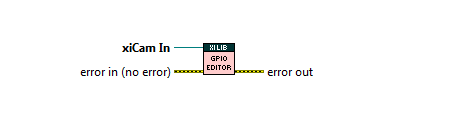
* GPO\_States[] - NUM: Returns GPO\_INDEX-GPO\_MODE value pairs as array of cluster.
* GPO\_States[] - STR: Returns GPO\_INDEX - GPO\_MODE value pairs as array of cluster.

xiLIB – GetPRM\_INFO.vi **:** Returns the description of a PRM parameter. See xiLIB - PRM\_Info.vi for more info.



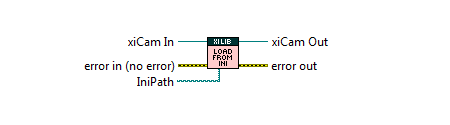
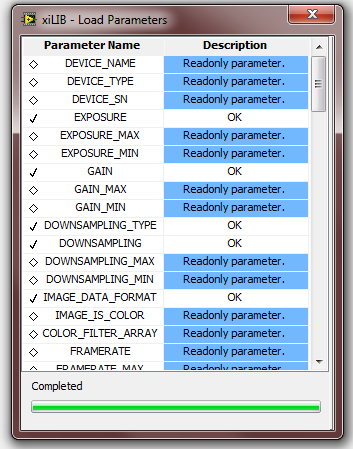
* Type: Type of the PRM.
* Method: Get / Set , Get , Set
* Desc.: Description of the parameter.
* PRM.: The parameter itself as a member of cluster.

xiLIB – GPIO\_Editor.vi *-* Dialog based GPIO editor.



This VI is used for Get or Set the Camera GPI / GPO states.

xiLIB – LoadFromINI\_DIALOG.vi : Dialog based parameter load function.



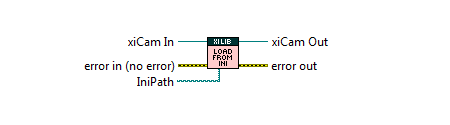
* IniPath <Path>: Path to xiParam.ini file.

Use this VI to restore the camera configuration from file. After the loading process a table shows the result of the load: SUCCESS, ERROR, WARNING - Not Valid Parameter, Warning - Get Parameter.

*Not Valid Parameter indicates that Parameter is not allowed for that device.*

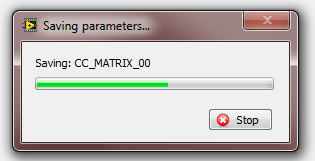
xiLIB – LoadFromINI.vi : Parameter load function.

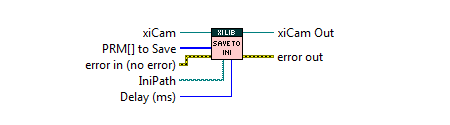
<This VI equals to the above one, but no dialog window appears.>

**

* IniPath <Path>: Path to xiParam.ini file.

xiLIB – SaveToINI\_DIALOG.vi : Save the specified Parameters to IniPath. The result file format (\*.ini) is readable and editable with Notepad or any similar program.

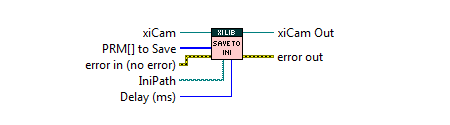




* PRM[] to Save <PRM.ctl>: Parameters that needs to be saved.
* IniPath <Path>: Path where parameters will be saved.
* Delay (ms) <U32> : Slow down saving process.

xiLIB – SaveToINI.vi : Save the specified Parameters to IniPath. The result file format (\*.ini) is readable and editable with Notepad or any similar program.

<This VI equals to the above one, but no dialog window appears.>



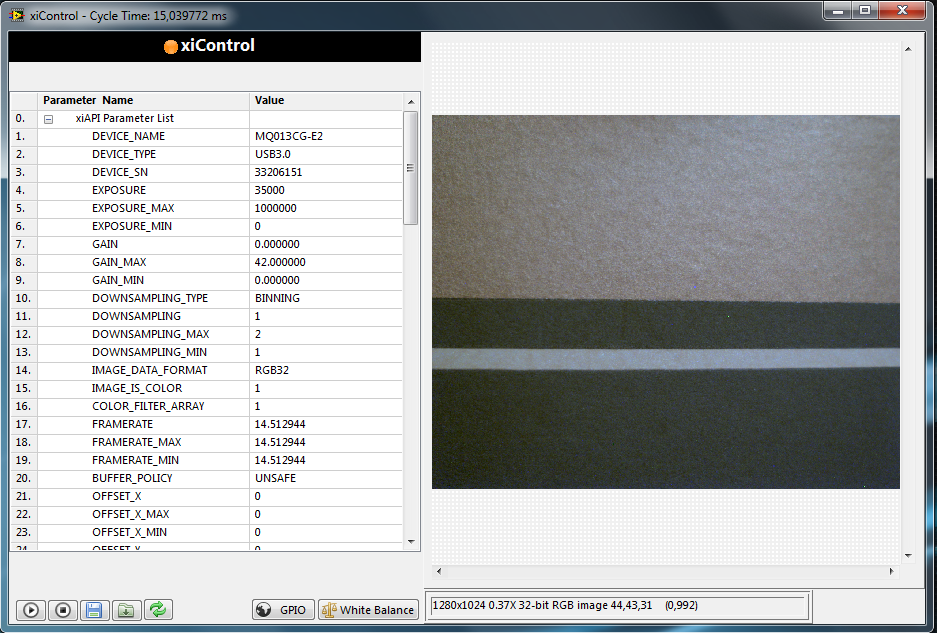
* PRM[] to Save <PRM.ctl>: Parameters that needs to be saved.
* IniPath <Path>: Path where parameters will be saved.

# xiControl.llb

Apart from the numerous VIs a premade xiControl is also included. It is an utility to set and get parameters of your camera. It is fully functional on its own as a top-level VI, but you can easily call this as a dialog VI.

High level camera configuration interface based on xiCam VIs. The following functionality has been implemented:

* Update camera parameters online.
* Save camera parameters to file.
* Load camera parameters from file.
* Test Software trigger mode.
* Manage GPI / GPO states.
* White Balance: Calculates the white balance.
* ( You can save image to HDD by right click on the Image Display control and choose *Save Image* )



Above the user interface of the xiControl is visible. On the right side of the screen the display got place, which shows the image(s) captured by the camera. On its left side the list of the camera's parameters are listed in a tree control.

By clicking any of tree row the selected parameter appears at the top of the screen, where (if the parameter is writeable it can be changed by the user. At the bottom of the front panel, the Start, Stop button responsible for image acquisition while the buttons with save and load icons can be used to save our current set of parameters or load a previous one. The rightmost control indicates the settings for GPIO.

xiControl has an easy-to-use and simple user interface designed for fast development and parameter editing.

This VI can be really helpful when you want to adjust the camera parameters; when it is done you can save this configuration to file, then your image processing application should start with a VI call:  **xiLIB - LoadFromINI\_DIALOG.vi**, hence all the previously adjusted parameters are applied.