



FEBRUARY 11, 2019

FLI KELPER SCMOS CAMERA

LABVIEW 2014+ PROGRAMMING

PETER OLEYNIKOV



Conventions

The following conventions are used in this manual:



This icon denotes a note that contains an important information.

Bold

Bold text denotes the User Interface items in the software (clickable, for example, menu items, dialog box buttons etc.).

Italic

Italic text denotes emphasis, cross-references, or an introduction to a key concept. Italic text can also denote a text that that the user must enter.

Abbreviations

sCMOS	scientific CMOS camera
VI	LabVIEW Virtual Instrument
subVI	sub-Virtual Instrument

Contents

Chapter 1 Introduction	1-1
About the FLI Kepler sCMOS LabVIEW driver	1-1
Windows Environment	1-1
Chapter 2 FLI LabVIEW wrapper library Installation	2-1
Installation	2-1
Adding the LabVIEW wrapper library to your project	2-1
Chapter 3 Available functions	3-1
Functions	3-2
FLIPRO_UINT_PROPERTIES	3-2
FLIPRO_DBL_PROPERTIES	3-3
Properties	3-3
Using general get/set property functions	3-3
Using get/set property functions directly	3-5
Image buffer	3-6
Chapter 4 Starting with LabVIEW (Beginners)	4-1
VI Creation	4-1
Chapter 5 LabVIEW Application Builder	5-1
Chapter 6 Examples	6-1
GetDoublePropertyExample.vi, SetDoublePropertyExample.vi	6-1
GetUIntPropertyExample.vi, SetUIntPropertyExample.vi	6-1
GetGainTableExample.vi	6-1
TemperatureExample.vi	6-2
AcquireFrameExampleAdvanced.vi	6-3
AcquireVideoExample.vi	6-4

Chapter 1 Introduction

The current document describes a LabVIEW wrapper library of the .NET **FLISharp.DLL** assembly that contains the class **FliProCamera** with functions controlling the FLI Kepler sCMOS camera. The wrapper contains VIs that will assist the user in programming the access to the FLI Kepler sCMOS camera using LabVIEW.

The package contains the following parts:

- API – low level camera access functions including the parameters used to make .NET calls;
- Examples – a set of examples that show the capabilities of the wrapper library.

About the FLI Kepler sCMOS LabVIEW driver

The FLI LabVIEW driver gives you the ability to use the FLI Kepler cameras to control camera from LabVIEW environment.



Note The cameras may operate at frame rates, depending on camera settings. Please refer to the FLI web-site for the details.

Windows Environment

The LabVIEW wrapper library requires the following software to be installed on the target PC:

- National Instruments LabVIEW 2014 or higher;
- National Instruments Vision Development Module (NI IMAQ).
- Visual C++ 14.0 libraries (for FLI library)

The FLI Driver require Visual C++ 14.0 libraries. The installer will provide these if they are not already present on your PC.

Chapter 2 FLI LabVIEW wrapper library Installation

This chapter contains the installation notes of the FLI Kepler sCMOS camera LabVIEW driver.

Installation

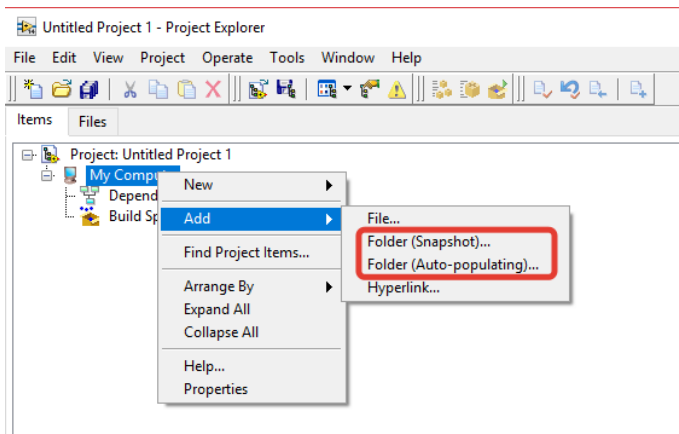
The setup package will install the following components:

- The Microsoft .NET Framework 3.5 (if not installed, web installer);
- The FLI DLL libflipro.x64.dll;
- The LabVIEW VIs.

The setup will also install the Visual C++ 14.0 libraries in automatic mode depending on the selected installation(s).

Adding the LabVIEW wrapper library to your project

Start a new or open an existing project that should have the FLI LabVIEW wrapper VIs. Using mouse, right-click on **My Computer** in the LabVIEW **Project Explorer**. Use either **Folder (Snapshot)...** or **Folder (Auto-populating)...** submenu item from the Add menu:



Chapter 3 Available functions

There are several functions and properties that can be used in LabVIEW.

All functions (except **CreateCamera** and **CreateAndOpenCamera**) in the FLI LabVIEW wrapper library accept as input at least 2 parameters:

- The C# .NET reference to the **FliProCamera** object (create by **CreateCamera** or **CreateAndOpenCamera**);
- **Error in** object.

All functions in the FLI LabVIEW wrapper library return as output at least 2 parameters:

- The C# .NET reference to the **FliProCamera** object (create by **CreateCamera** or **CreateAndOpenCamera**);
- **Error out** object.

As an example, the following block diagram shows the inputs and outputs of the **SetShutter** VI:

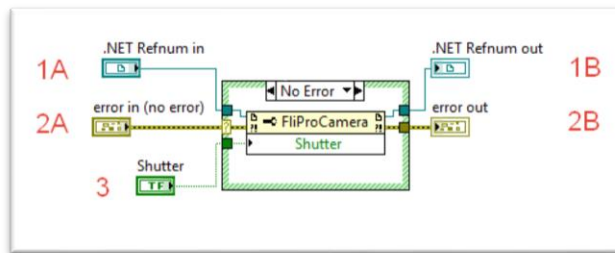


Figure 3-1. An example of the **SetShutter** VI: 1A and 1B: the input and output .NET references; 2A and 2B: the **error in** and **error out** objects. 3: extra input parameter.

The error objects can be linked together to keep track on the error (same is for the .NET input and output references):

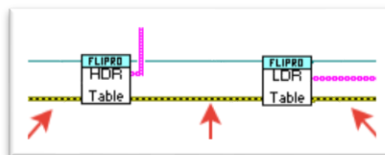















Figure 3-2. Chaining the **error in** and **error out** connections (marked by red arrows →).

The functions **CreateCamera** and **CreateAndOpenCamera** accept the camera ID (unsigned 32bit integer, 0 in most cases, unless there are more than one FLI Kepler camera connected) as the only input parameter.

Functions

The following table provides an overview of the functions available in the FLI LabVIEW wrapper library:

Function name	Input	Output	Icon	Description
CreateCamera	UInt32: Camera ID			Creates a new camera object
CreateAndOpenCamera	UInt32: Camera ID			Creates a new camera object and opens it
Open				Opens the created camera
Close				Closes the opened camera.
CaptureStart	UInt32: number of frames to capture. 0 is for infinite capture.			Starts the capture. The camera must be opened.
CaptureStartVideo				Start the video capture. The camera must be opened. Like CaptureStart but with number of frames set to 0.
CaptureAbort				Aborts the current frame acquisition.
CaptureStop				Finishes the current capture (started with either CaptureStart or CaptureStartVideo).
GetVideoFrame				Copies the video data into the specified buffer.
GetUIntProperty	FLIPRO_UINT_PROPERTIES: Property ID	UInt32		Gets a value of a UInt32 property. The property ID is defined by one of the FLIPRO_UINT_PROPERTIES enumeration values (see below).
GetDoubleProperty	FLIPRO_DBL_PROPERTIES: Property ID	Double precision		Gets a value of a Double property. The property ID is defined by one of the FLIPRO_DBL_PROPERTIES enumeration values (see below).
SetUIntProperty	FLIPRO_UINT_PROPERTIES: Property ID UInt32: Value			Sets a new value of a UInt32 property. The property ID is defined by one of the FLIPRO_UINT_PROPERTIES.
SetDoubleProperty	FLIPRO_DBL_PROPERTIES: Property ID Double: Value			Sets a new value of a Double property. The property ID is defined by one of the FLIPRO_DBL_PROPERTIES.

FLIPRO_UINT_PROPERTIES

The following enumeration values are available as UINT32 properties. **NOTE:** not all properties can be accessible for read/write. Read the comments of each enumeration for more details.

Enumeration name	Description	Read/write
Width	The camera width	No write.
Height	The camera height	No write.
SizeX	The ROI width	No write: the ROI width is fixed.
SizeY	The ROI height	
BinX	The horizontal binning	No write: the binning is fixed to 1.
BinY	The vertical binning	No write: the binning is fixed to 1.
OffsetX	The horizontal ROI offset	No write: the offset is fixed to 0.
OffsetY	The vertical ROI offset	
Exposure	The exposure time in milliseconds	
BlackLevelLo	The black level adjustment value for LDR channel	Limits: 0 – 16383
BlackLevelHi	The black level adjustment value for HDR channel	Limits: 0 – 16383
BlackSunLo	The black Sun adjustment value for LDR channel	Limits: 0 – 63
BlackSunHi	The black Sun adjustment value for HDR channel	Limits: 0 – 63
LDRGainIndex	The index of the LDR gain value in the gains table	Limited to the LDR gain table entries.
HDRGainIndex	The index of the HDR gain value in the gains table	Limited to the HDR gain table entries.
ModeIndex	The index of the mode value in the modes table	Limited to the modes table entries.
ControlShutter	Open/Closes the shutter	No read. Write: 1 = open, 0 = close.

FLIPRO_DBL_PROPERTIES

The following enumeration values are available as Double precision properties. **NOTE:** not all properties can be accessible for read/write. Read the comments of each enumeration for more details.

Enumeration name	Description	Read/write
CoolTemperature	The camera cooling temperature	
BaseTemperature	The camera base temperature	Read-only
AmbientTemperature	The camera ambient temperature	Read-only

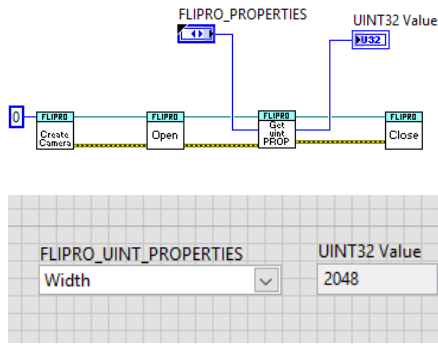
Properties

There are 2 ways to access the properties of the FliProCamera object:

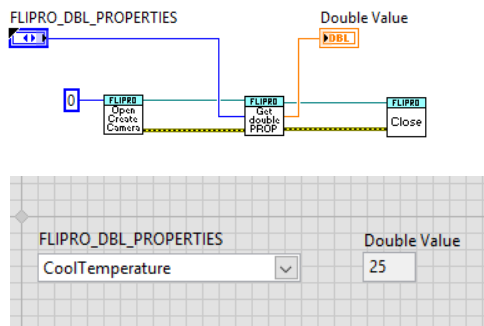
- Use **Get/Set<Type>Property** function (for example, **GetUIntProperty**, see above);
- Use the property **get/set** function directly.

Using general get/set property functions

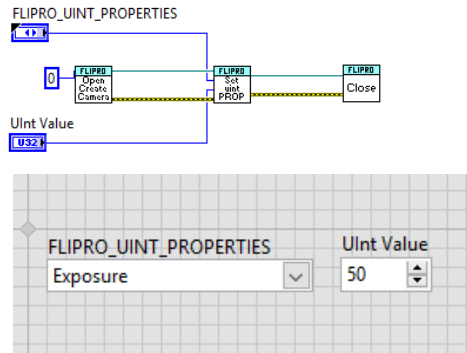
To get the value of an unsigned integer property use **GetUIntProperty**:



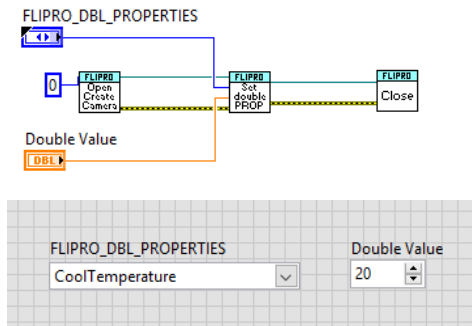
To get the value of a double precision property use **GetDoubleProperty**:



To set the value of an unsigned integer property use **SetUIntProperty**:































To set the value of a double precision property use **SetDoubleProperty**:



Using get/set property functions directly

The following table presents the properties available for the camera, the get and set values and a short description of each function:

Property	Get	Set	Icon get	Icon set	Description
BinningX	UInt	–			Horizontal binning. Get only (fixed to 1).
BinningY	UInt	–			Vertical binning. Get only (fixed to 1).
BlackLevel	UInt	UInt			Black Level adjustment for LDR
BlackLevelHi	UInt	UInt			Black Level adjustment for HDR
BlackSun	UInt	UInt			Black Sun adjustment for LDR
BlackSunHi	UInt	UInt			Black Sun adjustment for HDR
Exposure	UInt	UInt			Exposure time, milliseconds
Width	UInt	–			The ROI width value. Get only (fixed to 2048)
Height	UInt	UInt			The ROI height value (mod 2)
OffsetX	UInt	–			The ROI X-offset value. Get only (fixed to 0)
OffsetY	UInt	UInt			The ROI Y-offset value (mod 2)
NumDevices	UInt	–			Number of connected Kepler cameras
Shutter	–	UInt			Open (1)/Close (0) shutter
LDRGainIndex	UInt	UInt			Gain index in the LDRGainTable
LDRGainTable	Double[]	–			LDR gain table
HDRGainIndex	UInt	UInt			Gain index in the HDRGainTable
HDRGainTable	Double[]	–			HDR gain table
ModeIndex	UInt	UInt			Mode index in the Modes table





Modes	String[]	–			Modes table
Temperature	Double	Double			The cooling temperature of the camera
IsCapturing	Boolean	–			Returns True if the camera is in capturing mode

Image buffer

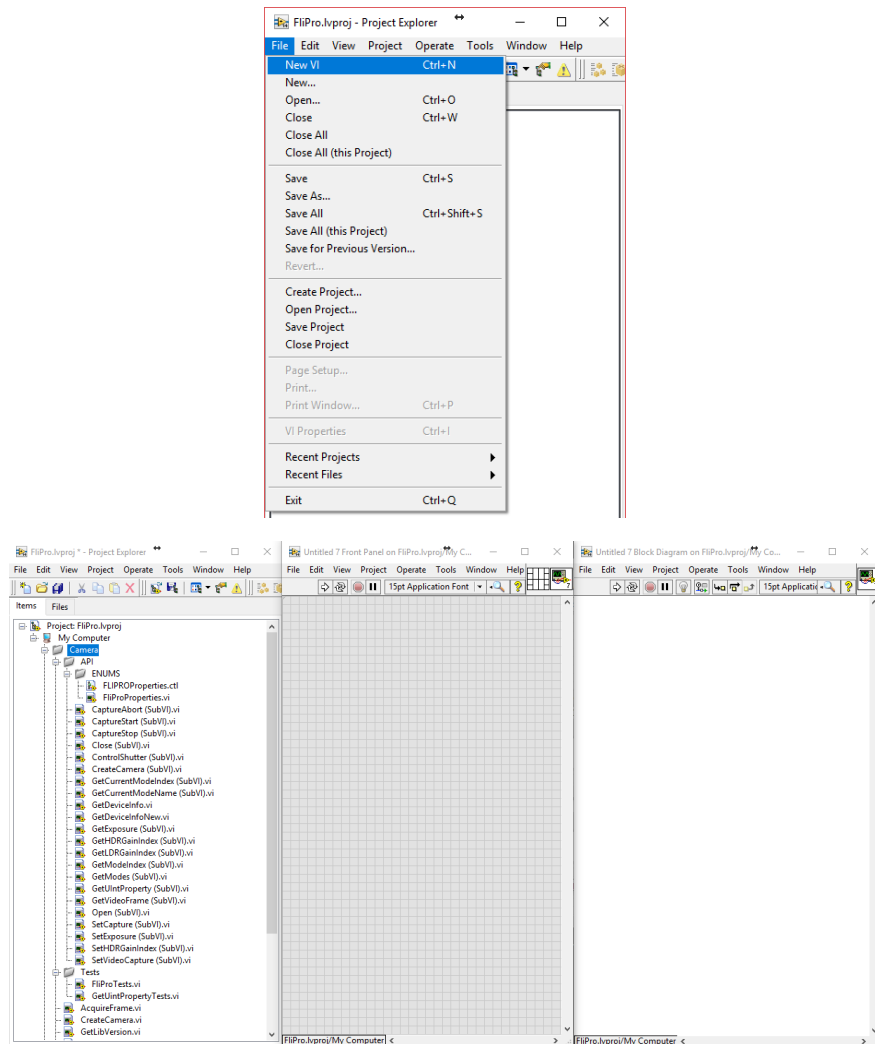
The access to the image data can be done by using the **CreateBuffer** VI. It returns a .NET object (class FrameBuffer) that has several properties, including a 2D array of U16 values. This array can be converted into an image using IMAQ **ArrayToImage** VI. **HINT:** press CTRL+Space inside the block diagram and type **ArrayToImage** in the Quick Drop window. See examples for image acquisition for the usage reference of the image buffer.

Chapter 4 Starting with LabVIEW (Beginners)

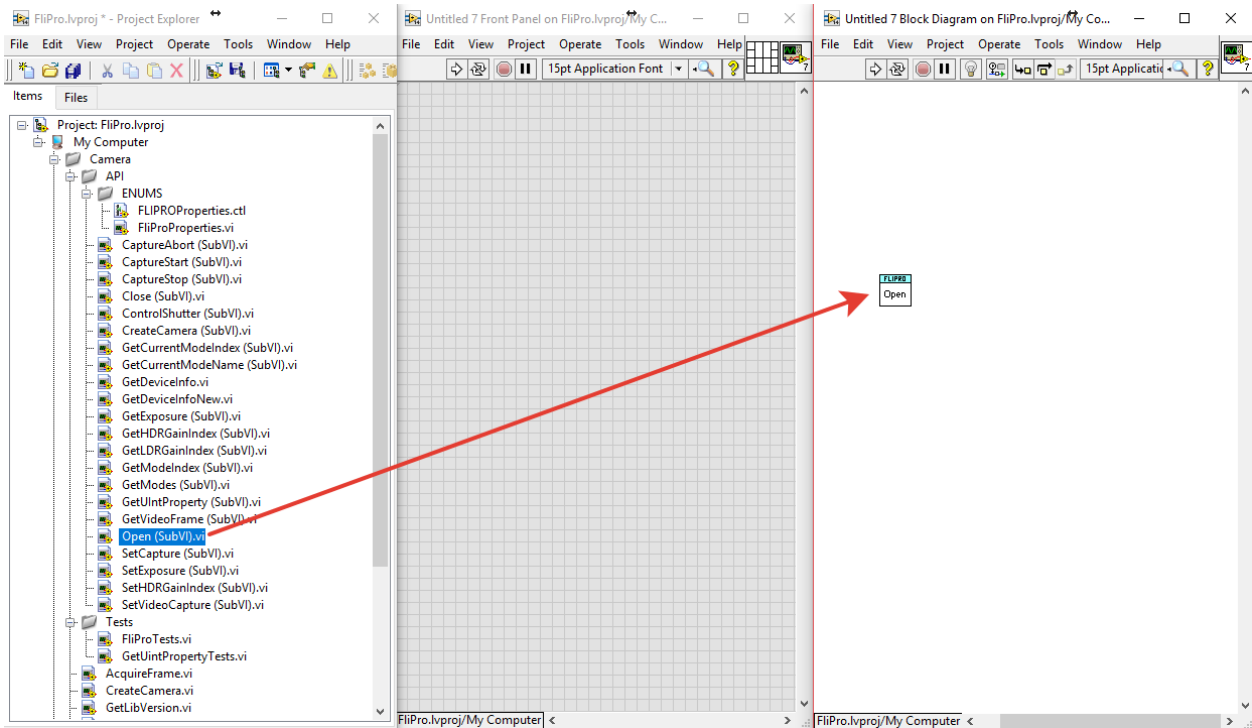
This chapter contains a step-by-step tutorial that shows how to create basic VIs in LabVIEW.

VI Creation

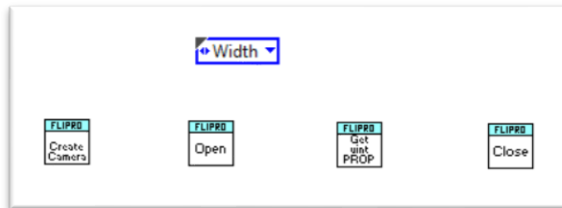
Create a new VI, press CTRL+E (Show Block Diagram) to enable the Block diagram:



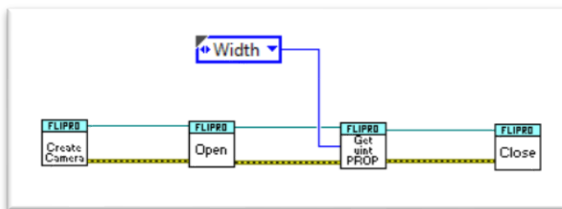
Drag n' drop the **Open** SubVI from the project **Items** (Project → My Computer → Camera → API) list:



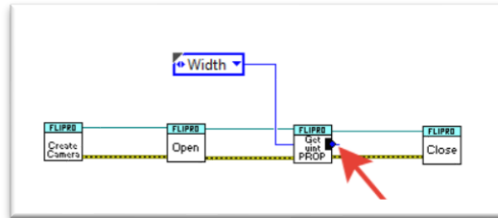
Do the same drag'n'drop for **CreateCamera**, **GetUIntProperty** and **Close** subVI. Drag the ENUMS control **FLIPRO_UINT_PROPERTIES.ctd** (Project → My Computer → Camera → API → ENUMS) to the Front panel:



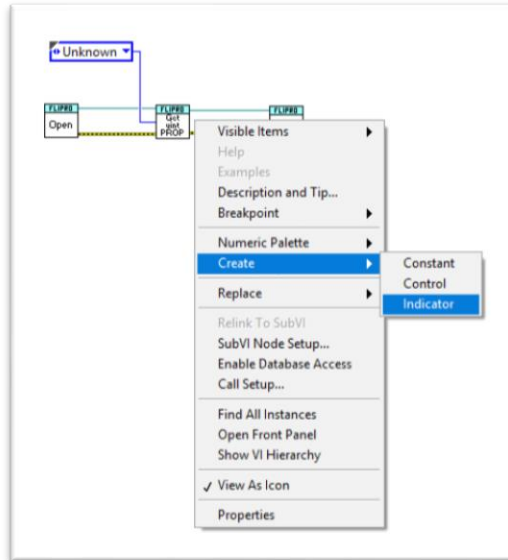
Connect them as show in the screenshot below:



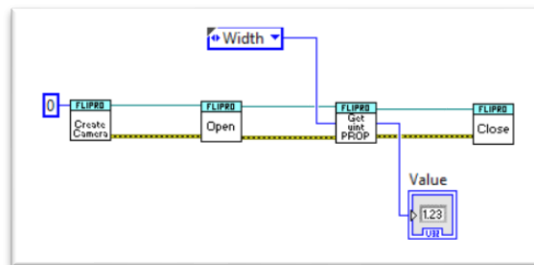
Right-click on the right part of the **Get uint PROP** subVI (marked by the red arrow →):



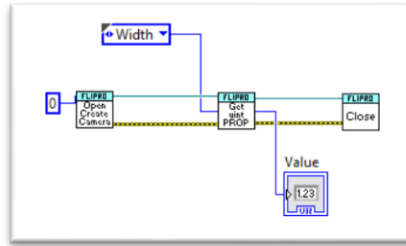
Select **Create** → **Indicator** from the opened context menu:



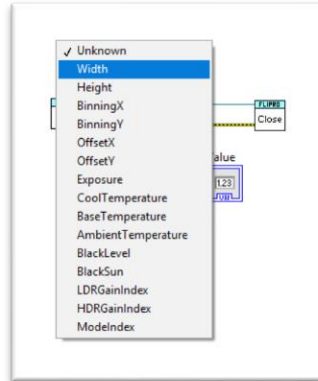
This will bind a numerical value indicator to the output of the subVI. In a similar manner create a constant on the left side (input) of the **Create Camera** subVI (keep 0 in the created constant). The final diagram should look as follows:



There is a more general subVI called “**CreateAndOpenCamera**” that merges two subVIs **CreateCamera** and **Open** camera into a single subVI, so the above block diagram can be simplified in the following way:



Select a property, for example, **Width**, by clicking on the “unknown” enumeration:

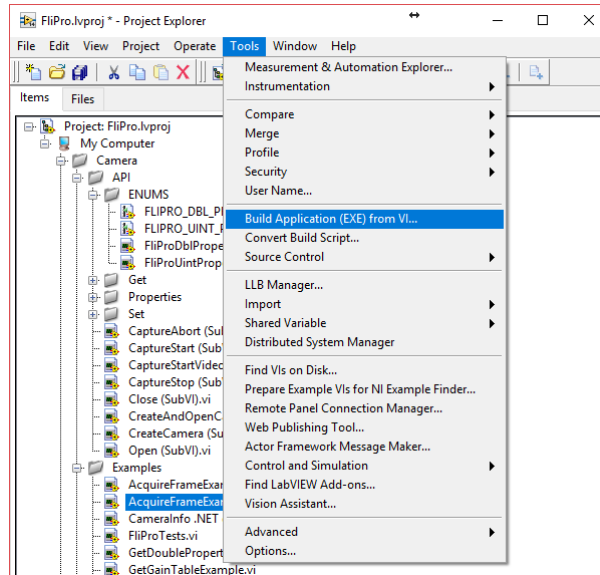


Now, switch back to the Front panel (press CTRL+E in the block diagram) and run the code. If the camera is connected, then the output should give the Value = 2048.

Chapter 5 LabVIEW Application Builder

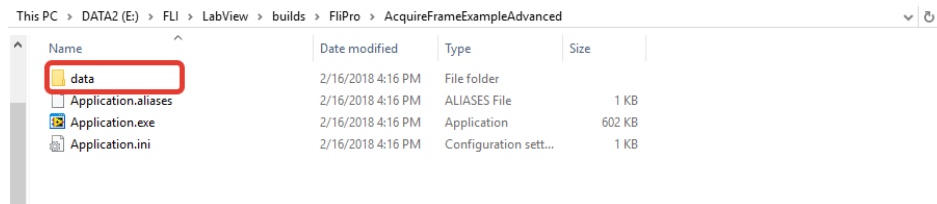
LabVIEW allows users building Windows Applications from the existing VI.

Select the VI in the LabVIEW Project Explorer and select **Tools → Build Application (EXE) from VI...** from the main menu:



Follow the instructions and generate the EXE file from the selected VI.

NOTE: The LabVIEW application builder cannot detect dynamically loaded DLL files. This means that the application builder cannot determine the dependence of the **FLISharp.DLL** on the **libflipro.x64.dll** and it will not be copied to the newly created EXE application folders. The user should perform the copy manually: the **libflipro.x64.dll** file is available in the FLI LabVIEW wrapper library installation and should be copied to the “**data**” folder of the newly created EXE application:

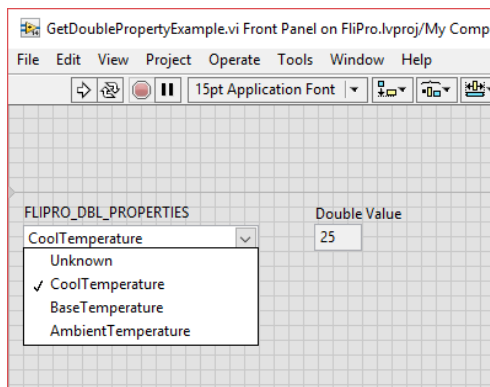


Chapter 6 Examples

This section provides a brief description of sample VIs that are provided with the FLI LabVIEW support library.

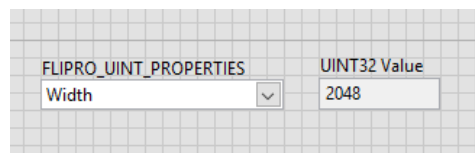
GetDoublePropertyExample.vi, SetDoublePropertyExample.vi

Shows how to get and set the double-precision property.



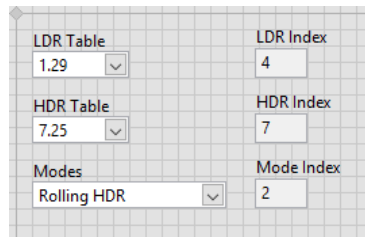
GetUIntPropertyExample.vi, SetUIntPropertyExample.vi

Shows how to get and set the UINT32 property.



GetGainTableExample.vi

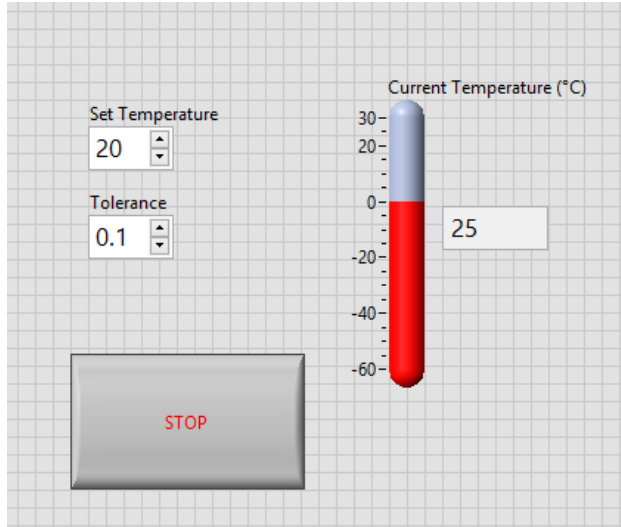
Shows how to get values from the LDR and HDR gain tables.



TemperatureExample.vi

Shows how to control the temperature using a loop. The loop executes while the cooling temperature read from the camera is not satisfying the equation

$$|T_{\text{Set}} - T_{\text{Cooling}}| \leq \text{Tolerance}$$



AcquireFrameExampleAdvanced.vi

This example shows how to use the FLI Kepler camera to acquire an image with the following settings that the user can change using the provided user interface:

- the exposure time;
- the mode (LDR or HDR);
- the LDR/HDR gains
- the black level adjustment
- the black Sun adjustment.

Run the block diagram in a standard way. Use **RECORD** button to acquire images; use the **EXIT** button to stop the execution.



Note The execution is configured so that changing values during exposition is not possible – the RECORD button and all input controls are disabled.

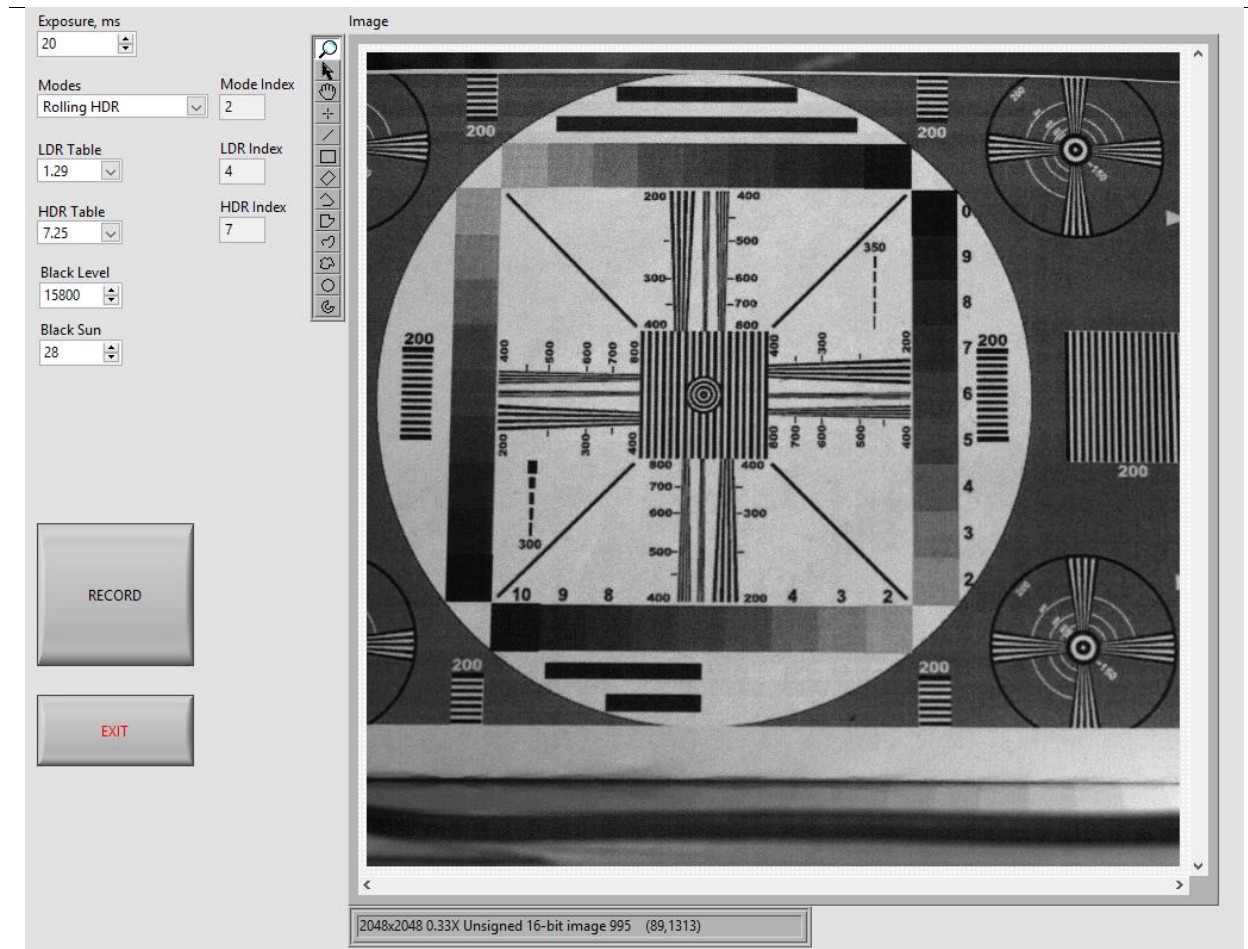


Figure. The single frame acquisition during the block diagram execution.

AcquireVideoExample.vi

This example shows how to use the FLI Kepler camera in video mode with the following settings that the user can change using the provided user interface:

- the exposure time;
- the mode (LDR or HDR);
- the LDR/HDR gains
- the black level adjustment
- the black Sun adjustment.

Run the block diagram in a standard way. The video acquisition will start immediately. the **STOP** button to stop the execution.



Note Set all necessary values (the exposure time etc.) before running the program. Changing the values during the program is not possible.

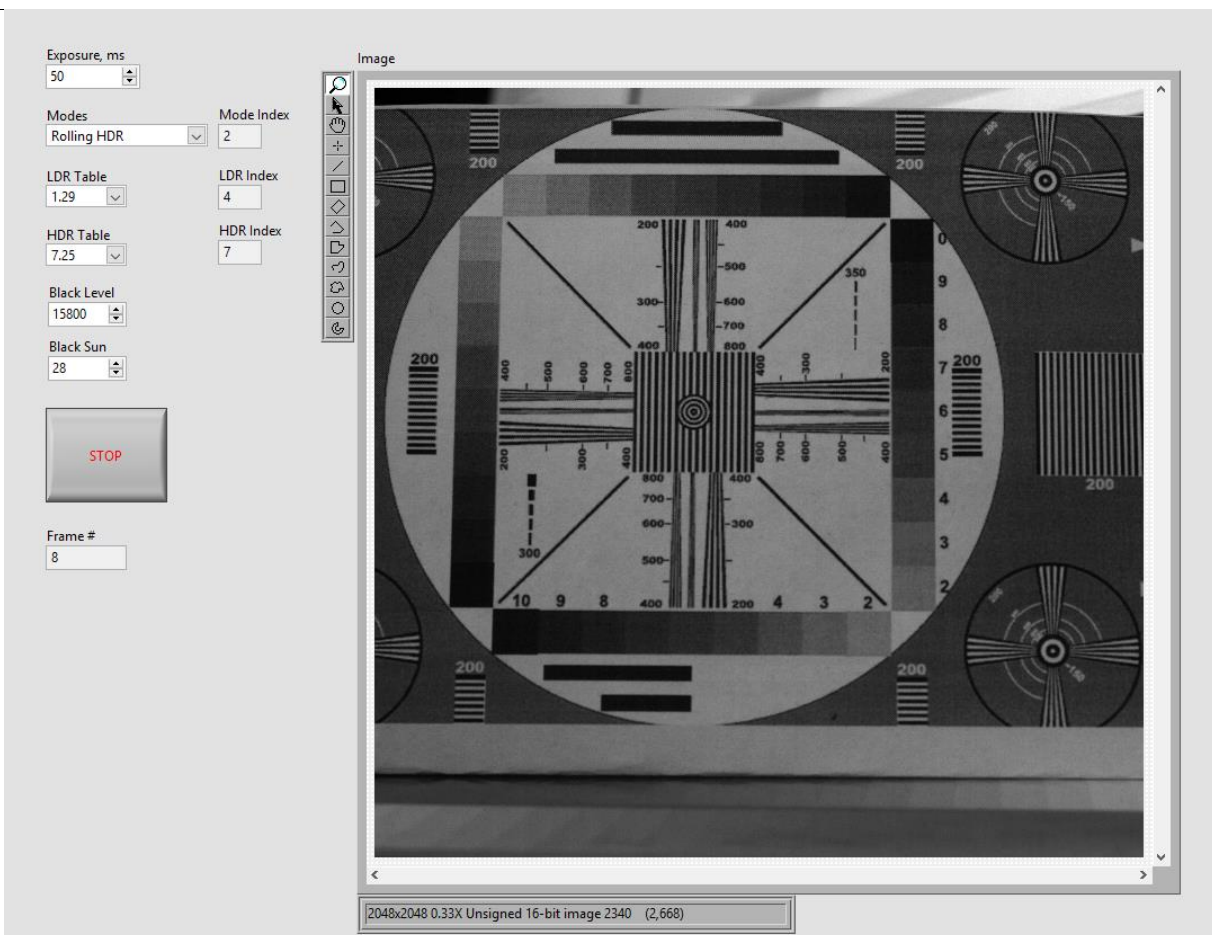


Figure. The video acquisition during the block diagram execution.