Adaptive Vision

Intuitive

Powerful

Adaptable

Adaptive Vision Sp. z o. o.
ul. Bojkowska 35a, 44-100 Gliwice, Poland
Tel. +48 32 461 23 30
E-mail: info@adaptive-vision.com

Application Note

Changing parameters of GigEVision cameras

Created: 03 October 2018

Modified: 02 December 2019

Document version: 1.1.4

Contents

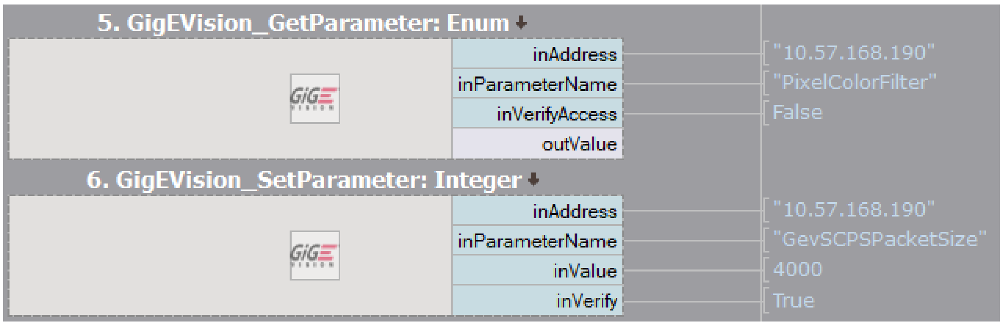
1. Introduction

The following guide provides information on how to programmatically choose and connect to a camera and modify its parameters. The modification of the acquisition parameters may involve restarting the acquisition. By implementing an interface to trigger the acquisition by software the program also shows how to execute camera commands. All the functionalities are available to the user through an HMI.

2. How to interact with camera

In Adaptive Vision Studio users can interact with a GigE camera through various filters. Those allow the user to change some of the camera's parameters or access acquisition data.

Filters *SetParameter* and *GetParameter* can be used to read and write parameter's values. Their common inputs include camera address and name of the parameter. The last common input – *inVerify* is used to determine if the program should check the validity of parameter before write or read.



5. GigEVision_GetParameter: Enum

inAddress	"10.57.168.190"
inParameterName	"PixelColorFilter"
inVerifyAccess	False
outValue	

6. GigEVision_SetParameter: Integer

inAddress	"10.57.168.190"
inParameterName	"GevSCPSPacketSize"
inValue	4000
inVerify	True

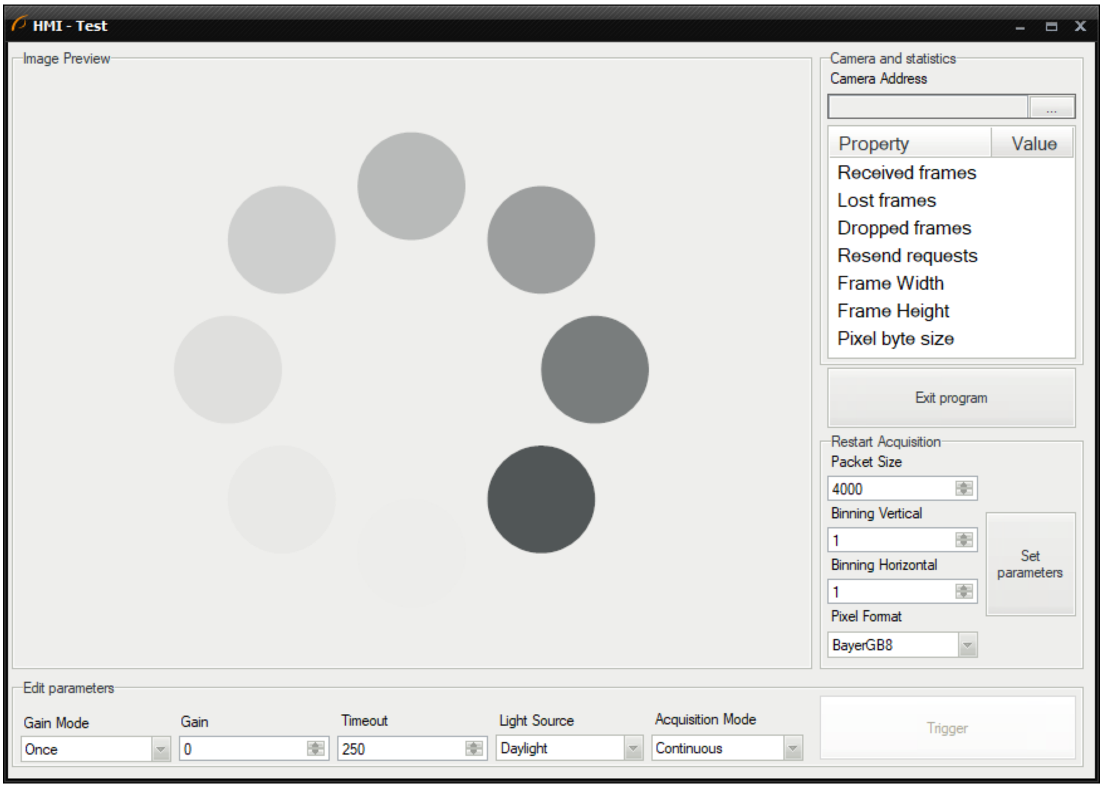
SetParameter has one more input, where the user specifies the value to be written. In *GetParameter* there is an output with the value of the parameter.

Both filters have variants depending on the type of the parameter being accessed – *Real, Integer, Bool, Enum, and String.*

To see what the possible values for a given parameter are, the user may just click on *inParameterName* which shows a window with available parameters for the currently connected camera.

3. Program overview

The designed program allows the user to select a camera from a list of connected devices. After selecting one, the acquisition starts. The program displays a preview of the camera images. The user may modify selected camera parameters like gain, light source correction, and acquisition mode. If the camera is set to triggered acquisition mode, the user can release a software trigger. While running the program displays information about the current acquisition.



HMI - Test

Image Preview

Camera and statistics

Camera Address

Property	Value
Received frames	
Lost frames	
Dropped frames	
Resend requests	
Frame Width	
Frame Height	
Pixel byte size	

Restart Acquisition

Packet Size: 4000

Binning Vertical: 1

Binning Horizontal: 1

Pixel Format: BayerGB8

Edit parameters

Gain Mode: Once

Gain: 0

Timeout: 250

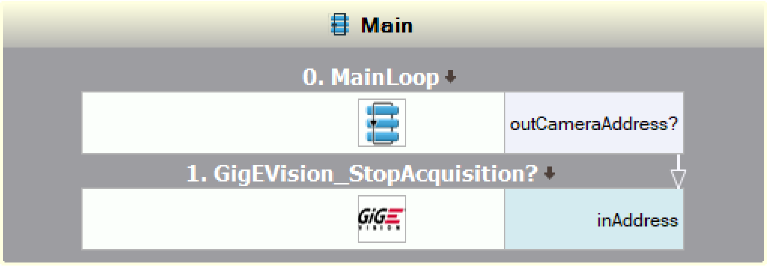
Light Source: Daylight

Acquisition Mode: Continuous

The user can choose which camera should be accessed with the control in the upper right corner of the HMI. Below that there is information about the current acquisition session. Under that is a button to exit the program.

Under the preview there are two groups of controls. The controls on the left allow changing the parameters that require a restart of acquisition. Because of that there is also a button which lets the user restart the acquisition with new parameters.

The other group has controls for the parameters that can be changed during the acquisition. The button labelled “Trigger” is used to start the acquisition using with a software trigger. It is disabled when in continuous mode.



Main

0. MainLoop

outCameraAddress?

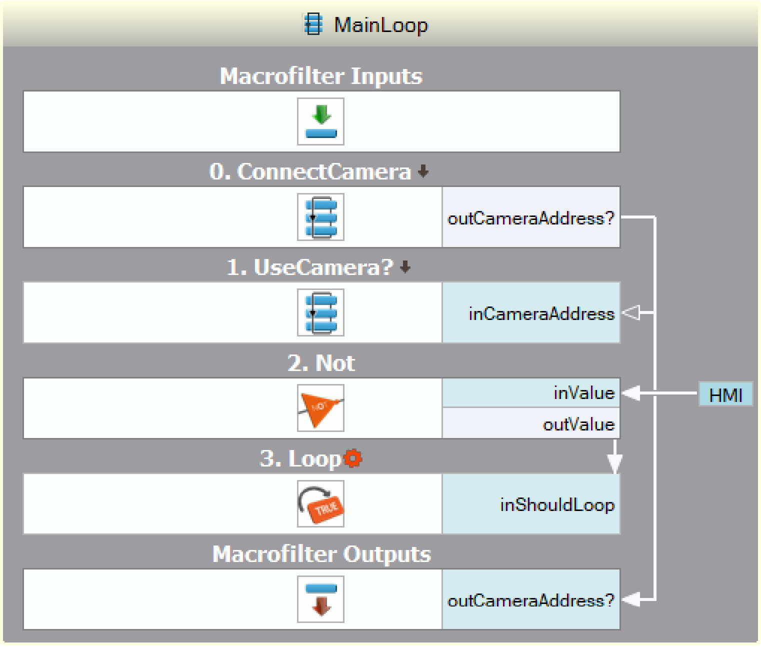
1. GigEVision_StopAcquisition?

inAddress

The *MainLoop* filter of the program can be divided into 3 parts. The first part is connecting the camera – here it is done by the *ConnectCamera* task macrofilter.

The second and the third part are done by the *UseCamera* task macrofilter. First the program starts the acquisition with set parameters. Then the program continuously acquires images while being able to change some of the camera parameters.

The *Not* and *Loop* filters control whether *MainLoop* will continue. The *Not* filter is connected to the "Exit program" HMI button. If the button has been pressed the loop will not continue – the program goes back top *Main* and stops the acquisition (if a camera had been connected) which finishes the program.



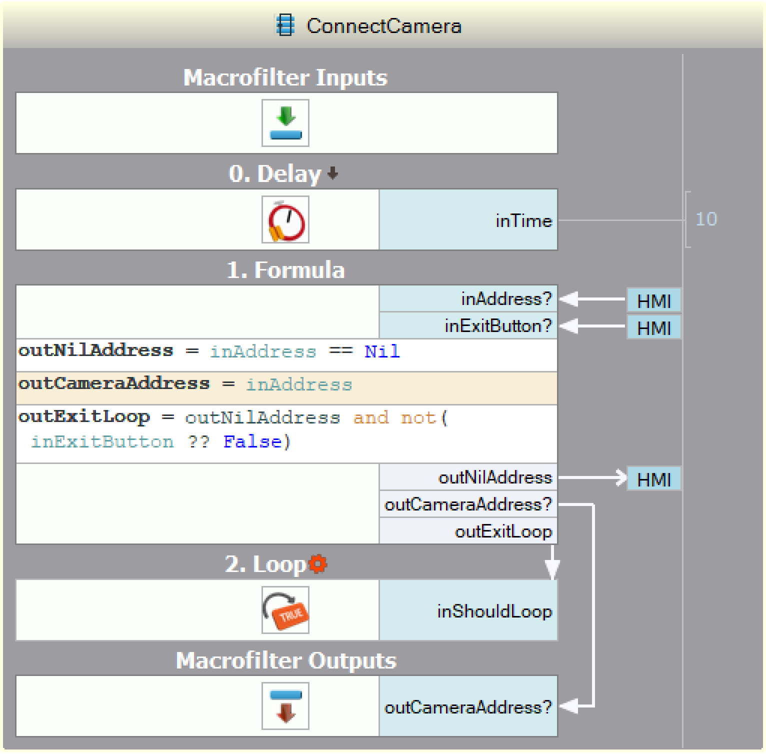
4. Connecting the camera

The first part of the program allows the user to connect a camera. The delay in the filter is used to limit the number of iterations per second. The *inAddress* input of the formula is connected to the HMI control labelled *Camera Address*.

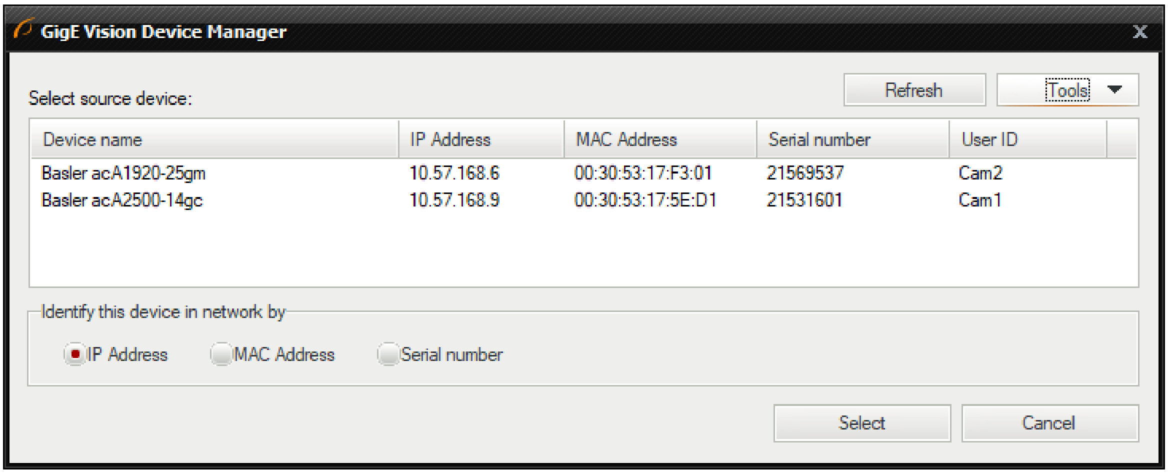
After clicking the 3-dot button a window opens letting the user select the camera. The selected camera's address is the sent to the formula block.

The animated waiting indicator is visible only if no camera has been chosen (so the address is Nil). The state of the indicator is controlled by the *outNilAddress* output of the formula.

If the user chooses a camera and its address is no longer Nil this exits the loop of this macrofilter and the camera address is outputted.



```
outNilAddress = inAddress == Nil
outCameraAddress = inAddress
outExitLoop = outNilAddress and not(
 inExitButton ?? False)
```



Select source device:

Device name	IP Address	MAC Address	Serial number	User ID
Basler acA1920-25gm	10.57.168.6	00:30:53:17:F3:01	21569537	Cam2
Basler acA2500-14gc	10.57.168.9	00:30:53:17:5E:D1	21531601	Cam1

Identify this device in network by

● IP Address ○ MAC Address ○ Serial number

5. Setting the parameters

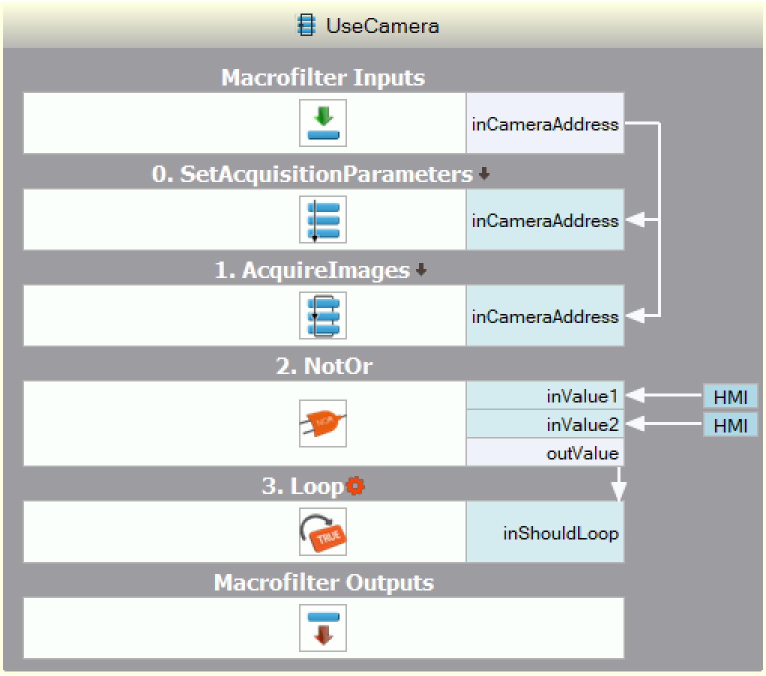
After the camera has been connected the next step is to start the acquisition. However, some acquisition parameters can only be set while not acquiring images. Such parameters include binning, packet size and pixel format. They are set in the *SetAcquisitionParameters* step macrofilter.

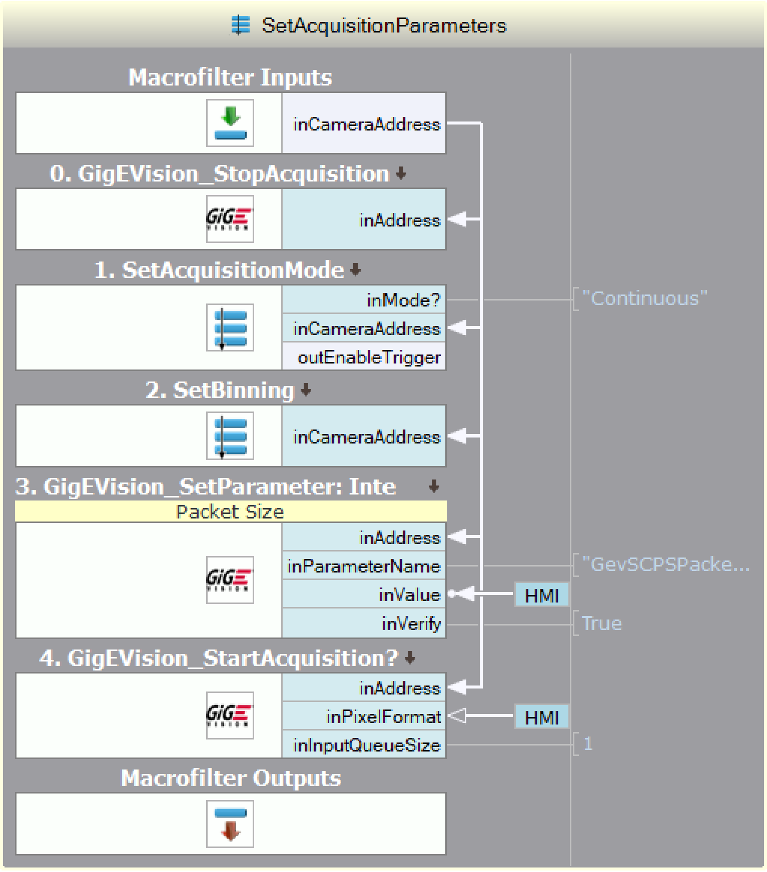
The *NotOr* filter negates the sum of two boolean values related to HMI controls. The first being *outAddressChanged* of the camera address picker and the second – whether the "Exit program" button has been pressed. If both are false the value of *NotOr* is true, which enables the loop to continue. If at least one of them is true – the program exits the *UseCamera* loop and goes back to *MainLoop.*

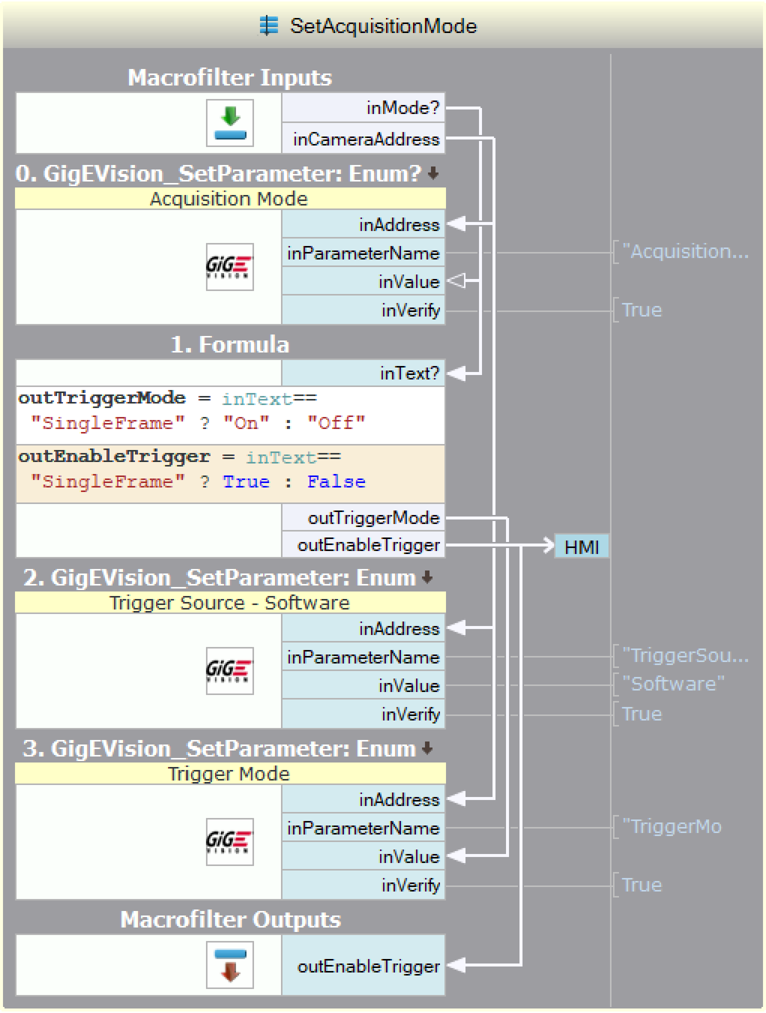
First any undergoing acquisition is stopped to ensure the parameters can be set. After that the mode of acquisition is set to "Continuous". This allows the camera to grab images to show in the preview of the program. The formula sets *outTriggerMode* to "Off" if the *inMode* parameter is anything other than "Single Frame".

The next steps *SetAcquistionParameters* set the binning of the camera (in both the horizontal and vertical axes) as well as the packet size. The possible values of the HMI controls used should be limited to be compatible with the camera value range. For example, the packet size here ranges from 220 to 16404 in increments of 4.

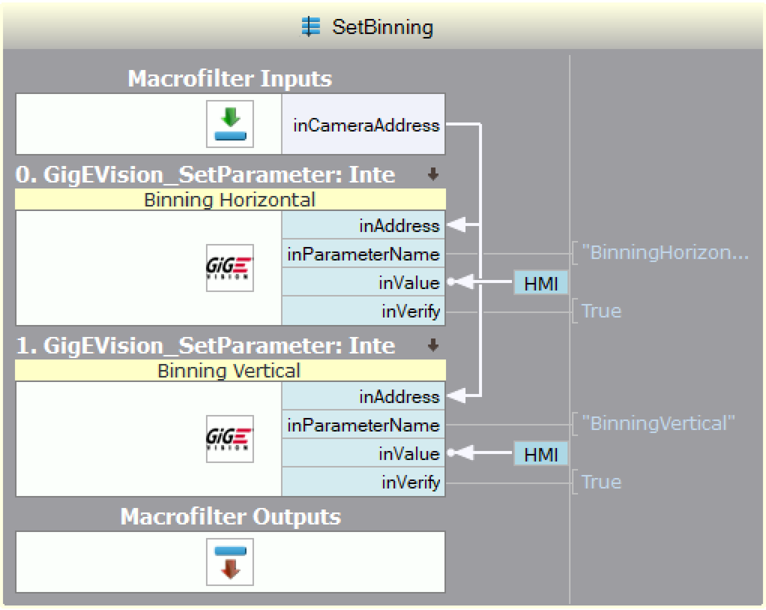
Finally, the program starts the acquisition with the pixel format chosen by the user.


UseCamera
Macrofilter Inputs
inCameraAddress
0. SetAcquisitionParameters
inCameraAddress
1. AcquireImages
inCameraAddress
2. NotOr
inValue1
inValue2
outValue
HMI
HMI
3. Loop
inShouldLoop
Macrofilter Outputs


SetAcquisitionParameters
Macrofilter Inputs
inCameraAddress
0. GigEVision_StopAcquisition
inAddress
1. SetAcquisitionMode
inMode?
inCameraAddress
outEnableTrigger
"Continuous"
2. SetBinning
inCameraAddress
3. GigEVision_SetParameter: Inte
Packet Size
inAddress
inParameterName
inValue
inVerify
"GevSCPSPacke...
HMI
True
4. GigEVision_StartAcquisition?
inAddress
inPixelFormat
inInputQueueSize
HMI
1
Macrofilter Outputs

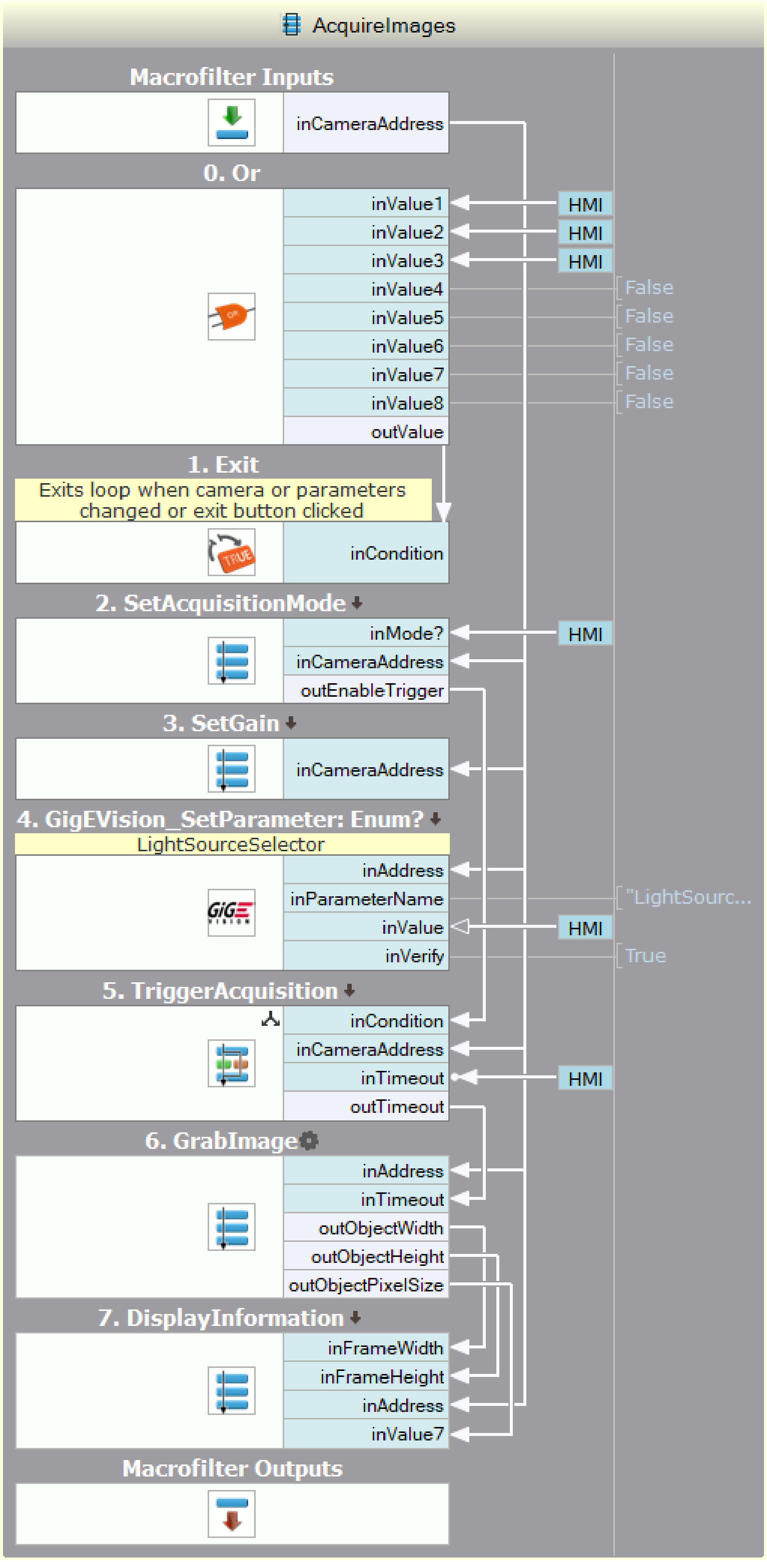


```
outTriggerMode = inText==
 "SingleFrame" ? "On" : "Off"
outEnableTrigger = inText==
 "SingleFrame" ? True : False
```



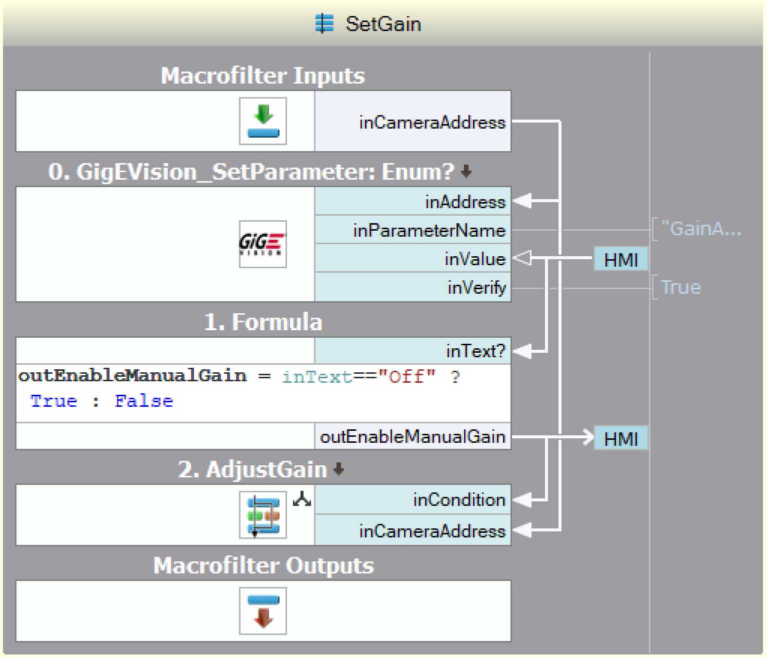
6. Acquiring images

Acquiring images takes place in a loop in the filter *AcquireImages*. It begins with an *Or* filter that's sums the states of three HMI controls: "Exit program" button, *outAddressChanged* of the camera address picker and "Set parameters" button. If any of them is true, the program exits the loop and goes back to *UseCamera*.

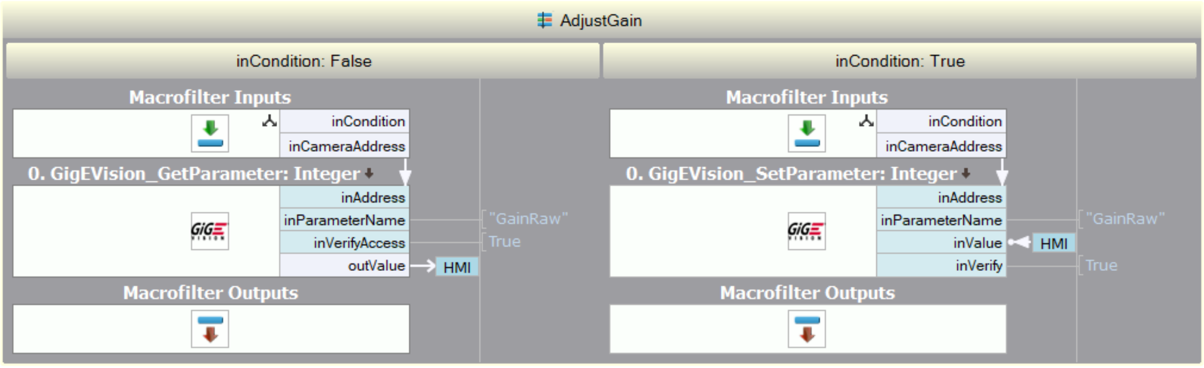


After that there is another instance of *SetAcquisitionMode*. This time the *inMode* parameter is connected to the appropriate control allowing the user to switch between continuous and triggered acquisition.

SetGain allows the user to switch between different modes of gain adjustment as well as set its value manually. First the program sets whether the gain will be adjusted manually or not based on the value from the control. If it is, the formula below sets the state for the next filter as well as enables (or disables) the gain value control in the HMI.



```
outEnableManualGain = inText=="Off" ?
 True : False
```

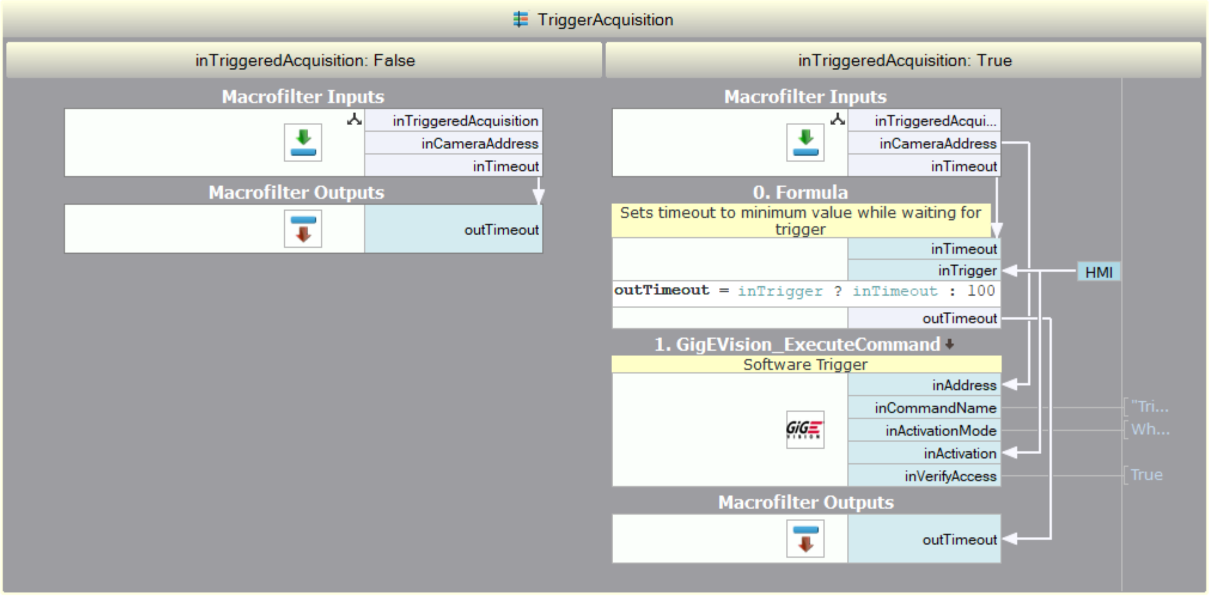


AdjustGain is a variant macrofilter. If gain is to be adjusted manually it sets the parameter to the given value. If gain is controlled automatically, it instead reads the value. This enables the user to see the automatically adjusted gain value in real time.

Next the light source is set. It allows the user to select the preset best suited to the current lighting conditions and by extension – to make colors look more natural.

TriggerAcquisition is another variant macrofilter. The *inTimeout* parameter sets how long should the program wait for an image from the camera. The variant for continuous acquisition is almost empty – it only passes *inTimeout* to *outTimeout*.

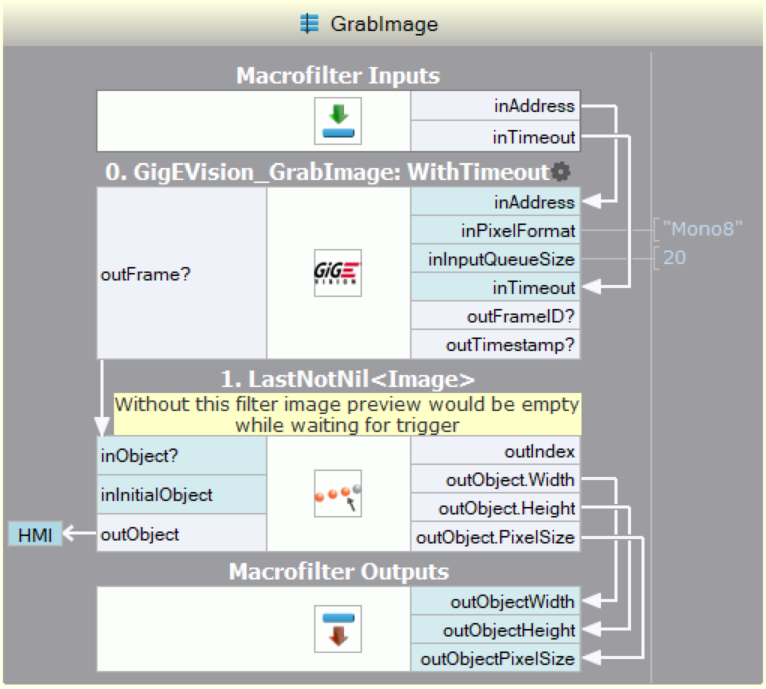
If the acquisition is triggered the program first check the state of the "Trigger" button. If it had been pressed the program executes the trigger command and passes *inTimeout* to *outTimeout*. If the button had not been pressed no command is executed and timeout is set to its lowest possible value – 100. Low timeout makes program more responsive when the user does not trigger the acquisition.



Sets timeout to minimum value while waiting for trigger

```
outTimeout = inTrigger ? inTimeout : 100
```

Software Trigger

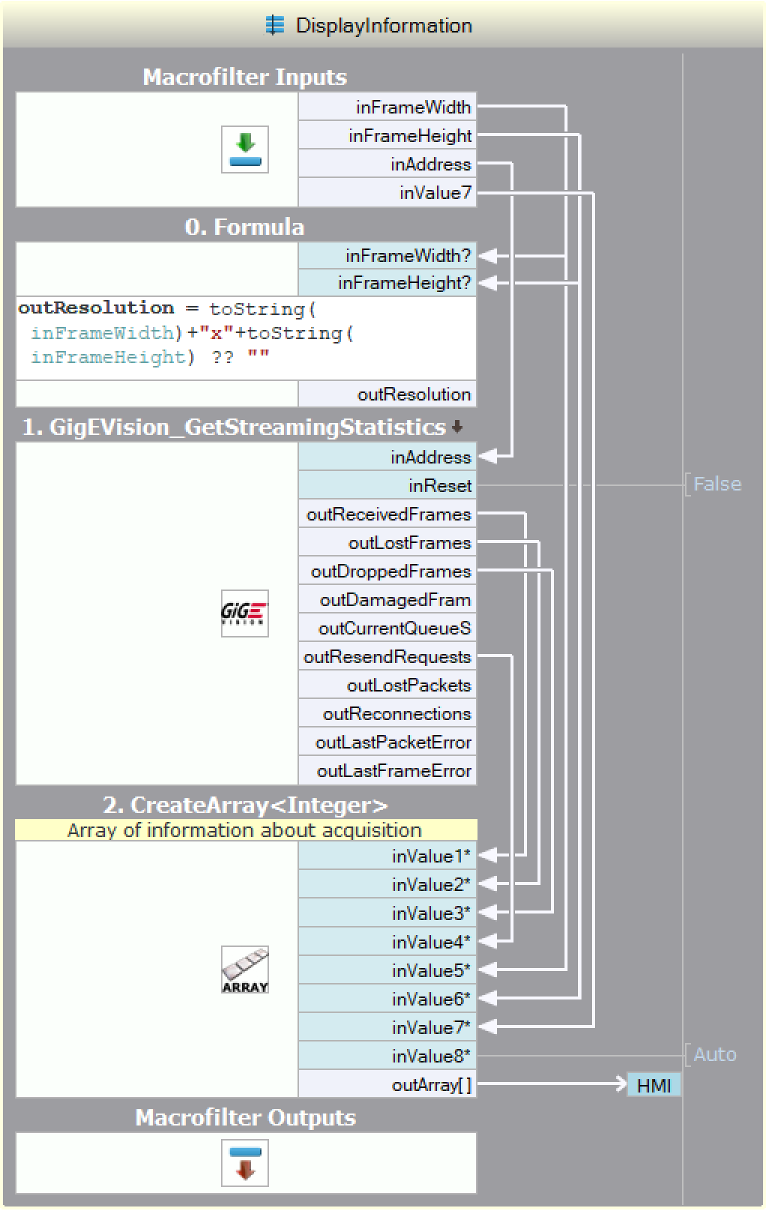


Without this filter image preview would be empty while waiting for trigger

After that, in *GrabImage* macrofilter, the program grabs and displays the image from the camera. Since the *inPixelFormat* has been set when starting the acquisition in the previous part of the program it is not necessary to change the value of this parameter here.

LastNotNil filter keeps the image in the preview in case the user is in triggered acquisition mode. Without it all images from the camera would only be displayed for one iteration of *AcquireImages*.

The last macrofilter in *AcquireImages* gathers data about the current acquisition and displays it in the HMI.



```
outResolution = toString(
 inFrameWidth)+"x"+toString(
 inFrameHeight) ?? ""
```

7. Notes

- The program may not work on every camera. This is dependent on the available parameters as well as their values.
- It is possible to modify more parameters in the program.
- To better understand how some of the parameters work reading the camera's documentation is recommended.
- If functionalities of some of the filters used in this example are unclear, check their documentation.