chromasens

SphinxLib Tutorial

Index

Tables

1 Introduction

The SphinxLib enables the user to handle a GenICam device in an easy and convenient way. GenICamTL and GenCP are handled by this library so the user can use high-level calls. This documentation will give you examples from a C point of view.

2 Overview GenICam

GenICam defines for camera features and pixel formats names that have to be adapted by any company, which wants to be GenICam certified. The SphinxLib enables you to access these parameters through a generic interface, which is independent of transport layer and control protocol.

Every GenICam compatible device has a device XML stored on it. When you connect to it, the XML becomes transmitted to the host PC and can be interpreted by the SphinxLib. This XML contains all available Features, information about the data type (e.g. Boolean, Integer, Enumeration …) etc. Please refer to the GenICam Standard if you want know more about the XML content. You can find this information if you follow this link

https://www.emva.org/standards-technology/genicam/genicam-downloads/ [11th October 2018]

There you can Download “GenICam Package …”. Inside this ZIP file you can find the folder “GenAPI” which contains the document GenICam_Standard_vX_X_X.pdf. This document contains all information about the XML elements which you will have to face.

3 General Information

Two channels handle the communication between camera and PC. The Control Channel enables the read/write access of camera xml features and works with TCP. The Streaming Channel transmits the image packets via UDP. Separate timeouts can be defined for each channel.

4 Function Overview

4.1 Connection Functions

Name	Required for implementation
GEVDiscovery	yes
GEVInit	yes
GEVClose	yes
GEVGetConnectionStatus	optional
GEVInitFilterDriver	recommended for windows, but not available with user defined buffer
GEVGetFilterDriverVersion	optional
GEVCloseFilterDriver	yes (if GEVInitFilterDriver was called)
GEVOpenStreamChannel	yes
GEVCloseStreamChannel	yes
GEVForceIP	optional
GEVSetChannelParameter	recommended
GEVGetChannelParameter	recommended
GEVSetDetailedLog	optional
GEVGetDetailedLog	optional
GEVSetHeartbeatRate	recommended
GEVGetHeartbeatRate	optional
GEVGetMemorySize	
GEVSetNetConfig	
GEVGetNetConfig	
GEVSetActionCommand	
GEVSetMessageChannel	
GEVTestPacket	
GEVSetReadWriteParameter	
GEVGetReadWriteParameter	
GEVSetReadWriteMemoryCallback	
GEVWriteMemory	
GEVReadMemory	
GEVWriteRegister	
GEVReadRegister	

4.2 XML Functions

Name	Required for implementation
GEVInitXml	
GEVGetXmlSize	
GEVSetXmlFile	
GEVGetXmlFile	
GEVGetFeatureList	
GEVGetFeatureInvalidator	
GEVGetFeatureDisplayName	
GEVGetFeatureEnableStatus	
GEVGetFeatureParameter	
GEVGetFeatureTooltip	
GEVGetFeatureUnit	
GEVSetFeatureBoolean	
GEVGetFeatureBoolean	
GEVSetFeatureCommand	
GEVGetFeatureCommand	
GEVSetFeatureEnumeration	
GEVGetFeatureEnumeration	
GEVSetFeatureEnumerationName	
GEVGetFeatureEnumerationName	
GEVSetFeatureFloat	
GEVGetFeatureFloat	
GEVSetFeatureInteger	
GEVGetFeatureInteger	
GEVSetFeatureRegister	
GEVGetFeatureRegister	
GEVSetFeatureString	
GEVGetFeatureString	

4.3 Image Functions

Name	Required for implementation
GEVAcquisitionStart	
GEVAcquisitionStartEx	
GEVAcquisitionStop	
GEVSetBufferCount	
GEVGetBufferCount	
GEVGetImageBuffer	
GEVSetRingBuffer	
GEVGetImageRingBuffer	
GEVQueueRingBuffer	
GEVReleaseRingBuffer	
GEVGetImageFPS	
GEVSetPacketResend	
GEVGetPacketResend	
GEVSetPacketsOutOfOrder	
GEVGetPacketsOutOfOrder	
GEVSetSecureTransfer	
GEVGetSecureTransfer	

4.4 Test functions

Name	Required
GEVTestPacketResend	optional
GEVTestFindMaxPacketSize	yes

5 SphinxLib Tutorial

For a basic application, you need to proceed as follows:

1. search for available devices
2. connect to a specific device
3. configure the connection
4. write/read camera features
5. grab images
6. disconnect from the device

5.1 Search Available Devices

GEVDiscovery searches the connected network for compliant GigE Vision devices. See following tables for details:

Parameter	Description
discovery	struct which contains information about the amount of cameras which are found and parameters of each device (IP, model name …)
discovery_callback_function	function which is called to get the search progress as an integer (0-100) in percent
discovery_timeout_ms	half of the time which is waited for the total discovery process (e.g. discovery_timeout_ms = 5000 means that the search process takes 10000 ms)
ignore_subnet	enables/disables device search in subnets

Table 1 GEVDiscovery: parameter description

Parameter	Description
progress	contains the current search progress in percent (0, 1, 2, … 100)

Table 2 discovery_callback_function:parameter description

```
BYTE error;
DISCOVERY discovery;
DWORD discovery_timeout_ms = 200;
BOOL ignore_subnet = FALSE;

BYTE WINAPI discovery_callback_function(int progress)
{
    //do something with progress information (range: 0-100)
    return 0;
}
```

Table 3 GEVDiscovery: example of use - related variables and methods

```
error = GEVDiscovery(&discovery, discovery_callback_function, discovery_timeout_ms,
ignore_subnet);
if(error != GEV_STATUS_SUCCESS)
{
    //handle the error
}

if(discovery.Count > 0)
{
    //a camera is available
    //use GEVInit to initialize the selected camera
}
```

Table 4 GEVDiscovery: example of use – main function

Back to SphinxLib Tutorial

5.2 Connect Specific Device

Once you have found a camera with GEVDiscovery you can connect to it by setting the connection parameters and call GEVInit. Found cameras are indexed starting at one. The discovery.param entries, which relate to the found cameras start at index 0. See following tables for details:

Parameter	Description
selected_camera	index which is used to identify the camera which shall be connected
connection	contains information about communication ports, camera IP …
error_callback_func	if an error occurs on the connected camera it will call this function so a output can be generated for the user
save_xml	the camera xml is downloaded from camera and stored to the SphinxLib directory if this flag is set
open_mode	connect to the camera in read only mode (OPEN_ACCESS), write only mode (CONTROL_ACCESS) or read/write mode (EXCLUSIVE_ACCESS)

Table 5 GEVInit: parameter description

Parameter	Description
calling_camera	indicates which camera has triggered error_callback_func
error_str	a string which describes the error

Table 6 error_callback_func: parameter description

```
CONNECTION connection;
BYTE selected_camera = 1; //the first camera starts at 1 and not at 0
BYTE save_xml = FALSE; //TRUE = store camera xml in SphinxLib directory; FALSE = don't store xml
BYTE open_mode = EXCLUSIVE_ACCESS;

BYTE WINAPI error_callback_func(BYTE calling_camera, char *error_str)
{
   printf("camera %i reported the following error: %s\n", calling_camera, error_str);
}
```

Table 7 GEVInit: example of use - related variables and methods

```
connection.AdapterIP = discovery.param[selected_camera - 1].AdapterIP;
connection.AdapterMask = discovery.param[selected_camera - 1].AdapterMask;
connection.IP_CANCam = discovery.param[selected_camera - 1].IP;
connection.PortCtrl = 49149; // control channel port
connection.PortData = 49150; // streaming channel port

error = GEVInit(selected_camera, &connection, error_callback_func, save_xml, open_mode);
if(error != GEV_STATUS_SUCCESS)
{
   //handle the connection error
}
```

Table 8 GEVInit: example of use - main function

Back to SphinxLib Tutorial

5.3 Configure Connection

5.3.1 GEVSetHeartbeatRate

This function sets the heartbeat timeout in ms. See following tables for details:

Parameter	Description
selected_camera	index which is used to identify the camera which shall be modified
heartbeat_interval_ms	defines the heartbeat interval in milliseconds

Table 9 GEVSetHeartbeatRate: parameter description

```
DWORD heartbeat_interval_ms = 3000;
```

Table 10 GEVSetHeartbeatRate: example of use - variables

```
error = GEVSetHeartbeatRate(selected_camera, heartbeat_interval_ms);
if(error)
{
   printf("couldn't set heartbeat rate");
}
```

Table 11 GEVSetHeartbeatRate: example of use - main function

Back to SphinxLib Tutorial

5.3.2 GEVSetMessageChannelCallback

Usually the camera responds only to the desktop application if it is polled. For special reasons there is a message channel implemented which you can use to receive interrupts from the camera.

```
BYTE WINAPI msg_callback_func(BYTE selected_camera, MESSAGECHANNEL_PARAMETER
mc_param)
{
     switch (mcparam.EventID)
     {
          case EVENT_TRIGGER:
               puts(„trigger event");
               break;
          case EVENT_START_EXPOSUE:
               puts("start of exposure");
               break;
          case EVENT_STOP_EXPOSUE:
               puts("end of exposure");
               break;
          case EVENT_START_TRANSFER:
               puts(„stream channel start of transfer");
               break;
          case EVENT_STOP_TRANSFER:
               puts("stream channel end of transfer");
               break;
          case EVENT_PRIMARY_APP_SWITCH:
               puts("primary application switchover has been granted");
```

```
                break;
            case EVENT_LINK_SPEED_CHANGE:
                puts("indicates that the link speed has changed.");
                break;
            case EVENT_ACTION_LATE:
                puts("execution of a Scheduled Action Command was late");
                break;

            default:
                puts("device-specific event");
                break;
    }

    //do something with event information
}
```

```
error = GEVSetMessageChannelCallback(selected_camera, msg_callback_func);
if(error == GEV_STATUS_NOT_SUPPORTED)
{
  //do something, because message channel isn't supported
}
```

Table 12 GEVSetMessageChannelCallback: example of use - main function

Back to SphinxLib Tutorial

5.3.3 GEVSetPacketResend

The packet resend should be disabled to improve performance.

Parameter	Description
selected_camera	index of the selected camera
enable_resend	0 = disable 1 = enable

Table 13 GEVPacketResend: parameter description

```
BYTE enable_resend = 0; //disables packet resend
```

Table 14 GEVPacketResend: example of use - related variables and methods

```
error = GEVPacketResend(selected_camera, enable_resend);
```

Table 15 GEVPacketResend: example of use - main function

5.3.4 GEVTestFindMaxPacketSize

Tests the maximum available packet size and enables it for the connection.

Parameter	Description
selected_camera	index of the selected camera
packet_size	the biggest packet size which could be set for the connection
ps_min	set minimum packet size
ps_max	set maximum packet size
ps_inc	set increment steps for the method

Table 16 GEVTestFindMaxPacketSize: parameter description

```
WORD ps_min = 0;
WORD ps_max = 20000;
WORD ps_inc  = 4;
WORD packet_size = 0;
```

Table 17 GEVTestFindMaxPacketSize example of use - related variables and methods

```
error = GEVTestFindMaxPacketSize(selected_camera, &packet_size, ps_min, ps_max, ps_inc);
if(error)
{
    //handle error if packet size couldn't be re
}
```

Table 18 GEVTestFindMaxPacketSize: example of use – main function

5.4 Camera Feature Handling

The camera provides a XML that contains all information about available features. Depending on the data type of the feature you need to choose the correct method.

```
GEVInitXml(selected_camera); //SphinxLib interprets the camera xml so you can work with names
                             //instead of numbers

char featureEnumerationName[255];
error = GEVGetFeatureEnumerationName(selected_camera, "TestPattern",
featureEnumerationName, sizeof(featureEnumerationName);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
error = GEVSetFeatureEnumerationName(selected_camera, "TestPattern",
"HorizontalLineMoving", 21);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}

INT64 featureInteger;
error = GEVGetFeatureInteger(selected_camera, "Width", &featureInteger);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
error = GEVSetFeatureInteger(selected_camera, "Width", 12000);
{
    //handle error
}

DWORD featureBoolean
error = GEVGetFeatureBoolean(selected_camera, "LightEnable", &featureBoolean);
```

```
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
error = GEVSetFeatureBoolean(selected_camera, “LightEnable”, 1);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}




char featureString[32];
error = GEVGetFeatureString(selected_camera, “DeviceSerialNumber”, featureString);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
error = GEVSetFeatureString(selected_camera, “UserSetComment”, “TestUserSet”);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}

DWORD featureCommand;
error = GEVGetFeatureCommand(selected_camera, “AcquisitionStart”, &featureCommand);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
error = GEVSetFeatureCommand(selected_camera, “AcquisitionStart”, 1);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
```

5.5 Grab Images

For grabbing images, you need to

- open the streaming channel
- prepare User Managed Buffer or SphinxLib Managed Buffer
- start acquisition
- get the image from buffer
- stop acquisition
- close the streaming channel
- free the User Handled Buffer

Back to SphinxLib Tutorial

5.5.1 Open Streaming Channel

You can open the Streaming Channel (used for image transmission) with GEVOpenStreamChannel.

Parameter	Description
selected_camera	index of the selected camera
connection.AdapterIP	defines the adapter ip which is used for communication with the camera
connection.PortData	defines the streaming channel port
multicast_disable	0: no multicast is used > 0: defines and uses multicast ip in big endian format

Table 19 GEVOpenStreamChannel: parameter description

```
CONNECTION connection;
BYTE selected_camera = 1;
DWORD multicast_disable = 0; //enter big endian ip address to use this function, otherwise enter 0
```

Table 20 GEVOpenStreamChannel: example of use - related variables and methods

```
connection.AdapterIP = discovery.param[selected_camera - 1].AdapterIP;
connection.AdapterMask = discovery.param[selected_camera - 1].AdapterMask;
connection.IP_CANCam = discovery.param[selected_camera - 1].IP;
connection.PortCtrl = 49149; // control channel port
connection.PortData = 49150; // streaming channel port

error = GEVOpenStreamChannel(selected_camera, connection.AdapterIP, connection.PortData,
multicast_disable);
```

Table 21 GEVOpenStreamChannel: example of use - main function

Back to Grab Images

5.5.2 Image Buffer Preparation

Received images are stored in a buffer. You can leave the control of it to the SphinxLib “SphinxLib Manged Buffer” or you can allocate the memory on your own and pass the buffers to the SphinxLib “User Managed Buffer”

5.5.2.1 User Managed Buffer

Note: User Managed Buffer is not compatible with the use of a filter driver.

Back to Grab Images

5.5.2.2 SphinxLib Managed Buffer

You can set the amount of ring buffers with “GEVSetBufferCount” or get the current amount of used buffers with “GEVGetBufferCount”. The SphinxLib handles the use of the buffers.

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected
buffer_count	the amount of buffers which shall be used for receiving images

Table 22 GEVSetBufferCount: parameter description

```
BYTE selected_camera = 1;
WORD buffer_count = 5;
```

Table 23 GEVSetBufferCount: example of use – related variables and methods

```
error = GEVSetBufferCount(selected_camera, buffer_count);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
```

Table 24 GEVSetBufferCount: example of use – main function

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected
buffer_count	the amount of buffers which shall be used for receiving images

Table 25 GEVGetBufferCount: parameter description

```
BYTE selected_camera = 1;
WORD buffer_count;
```

Table 26 GEVGetBufferCount: example of use – related variables and methods

```
error = GEVSetBufferCount(selected_camera, &buffer_count);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
```

Table 27 GEVGetBufferCount: example of use – main function

Back to Grab Images

5.5.3 Start Acquisition

GEVAcquisitionStart sends information to the camera to start the video stream.

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected
acquisition_count	the amount of images which shall be recorded 0 = unlimited > 0 = grab the specified amount of images

Table 28 GEVAcquisitionStart: parameter description

```
BYTE selected_camera = 1;
DWORD acquisition_count;
```

Table 29 GEVAcquisitionStart: example of use – related variables and methods

```
error = GEVAcquisitionStart(selected_camera, acquisition_count);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
```

Table 30 GEVAcquisitionStart: example of use – main function

Back to Grab Images

5.5.4 Get Image Buffer

You can call the following functions (depending on your buffer management) in a loop to get the image information you need. The GEVGetImage… Method blocks until whether a whole image has arrived or a timeout occurred.

5.5.4.1 User Managed Buffer

Back to Grab Images

5.5.4.2 SphinxLib Managed Buffer

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected
image_header	contains information about the image: size, missing packets, …
image_buffer	pointer to the raw image data for processing

Table 31 GEVGetImageBuffer: parameter description

```
BYTE selected_camera = 1;
IMAGE_HEADER image_header;
BYTE image_buffer[width*height*byte_per_pixel];
```

Table 32 GEVGetImageBuffer: example of use – related variables and methods

```
error = GEVGetImageBuffer (selected_camera, &image_header, image_buffer);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
else
{
    //do something with pixel data
}
```

Table 33 GEVGetImageBuffer: example of use – main function

Back to Grab Images

5.5.5 Stop Acquisition

Request the camera to stop image transmission.

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected

Table 34 GEVAcquisitionStop: parameter description

```
error = GEVAcquisitionStop(selected_camera);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
```

Table 35 GEVAcquisitionStop: example of use - main function

Back to Grab Images

5.5.6 Close Streaming Channel

Closes the streaming channel.

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected

Table 36 GEVCloseStreamChannel: parameter description

```
error = GEVCloseStreamChannel(selected_camera);
if(error != GEV_STATUS_SUCCESS)
{
    //handle error
}
```

Table 37 GEVCloseStreamChannel: example of use - main function

Back to Grab Images

5.5.7 Free User Managed Buffer

Back to Grab Images

5.6 Disconnect Device

This function disconnects the selected camera. Do not forget to close the Streaming Channel before you call this method. See following tables for details:

Parameter	Description
selected_camera	index which is used to identify the camera which shall be disconnected

Table 38 GEVClose: parameter description

```
error = GEVClose(selected_camera);
if(error != GEV_STATUS_SUCCESS)
{
    //handle close error
}
```

Table 39 GEVClose: example of use – main function

Back to SphinxLib Tutorial