SynView

Reference Guide

Generated by Doxygen 1.8.7

Mon Nov 16 2015 14:18:50

Contents

Chapter 1

SynView Reference Guide

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Module Documentation

5.1 SynView Plain C API functions

Modules

5.1.1 Detailed Description

5.2 SynView General purpose functions

Functions

- LV_EXTC LV_DLLIMPORT uint32_t LvGetVersion ()
- LV_EXTC LV_DLLIMPORT LvStatus LvOpenLibrary ()
- LV_EXTC LV_DLLIMPORT LvStatus LvCloseLibrary ()
- LV_EXTC LV_DLLIMPORT void LvGetErrorMessage (LvStatus Error, char ∗pMessage, size_t Size)
- LV_EXTC LV_DLLIMPORT void LvGetLastErrorMessage (char ∗pMessage, size_t Size)
- LV_EXTC LV_DLLIMPORT void LvLog (const char ∗pLogMessage)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetLibInfo (LvEnum Info, int32_t ∗pInfo, int32_t Param)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetLibInfoStr (LvEnum Info, char ∗pInfoStr, size_t Size, int32_t Param)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetLibInfoStrSize (LvEnum Info, size_t ∗pSize, int32_t Param)

5.2.1 Detailed Description

5.2.2 Function Documentation

5.2.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvCloseLibrary ( )

Closes the SynView library. This must be done before you exit your application. Be sure to close first all dependent modules (System). If you are using SynView in a Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is still fully functional, which is not the case of PROCESS_DETACH in the DllMain(). If you have called LvOpenLibrary() multiple times, you must balance it by the same number of calls of this function. Only the last call actually does the uninitialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.2.2.2 LV_EXTC LV_DLLIMPORT void LvGetErrorMessage (LvStatus *Error*, char ∗ *pMessage*, size_t *Size*)

Returns a short description of the error. Note that only some of the errors are suitable for direct display to the user, many error values indicate states which are understandable to the programmer, but may not be understandable to the end user.

Parameters

Error	The error code (the return value of most SynView functions).
pMessage	Pointer to the text buffer.
Size	Size of the buffer.

See also

LvStatus definitions.

5.2.2.3 LV_EXTC LV_DLLIMPORT void LvGetLastErrorMessage (char ∗ *pMessage*, size_t *Size*)

Returns more detailed description of the last error, which happened in the thread from which this function was called. As the info is recorded inside SynView for each error, the desription provides more detailed info, including the name of the function, in which the error happened, and possibly more diagnostic info. The difference to LvGetErrorMessage() is that LvGetErrorMessage() returns a static string from a numbered table of errors while this

function returns additionally info recorded at the time the error happened. If a function returns LVSTATUS_OK, it does not reset this error message (for speed reasons) so the correct approach is to get the error number as the function return value and if this return value is not LVSTATUS_OK, then you can get more info about the error using this function. be sure to call it from the same thread.

Parameters

pMessage	Pointer to the text buffer.
Size	Size of the buffer.

See also

LvStatus definitions.

5.2.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvGetLibInfo (LvEnum *Info*, int32_t * *pInfo*, int32_t *Param*)

Gets a general info in form of a 32-bit integer value.

Parameters

Info	One of the LvLibInfo values.
pInfo	The value is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.2.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvGetLibInfoStr (LvEnum *Info*, char * *pInfoStr*, size_t *Size*, int32_t *Param*)

Gets a general info in form of a string value.

Parameters

Info	One of the LvLibInfo values.
pInfoStr	The string value is returned in this parameter.
Size	Size of the buffer (to which pInfoStr points).
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.2.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvGetLibInfoStrSize (LvEnum *Info*, size_t * *pSize*, int32_t *Param*)

Gets a buffer size needed for a general info in form of a string value.

Parameters

Info	One of the LvLibInfo values.
pSize	Size of the buffer is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.2.2.7 LV_EXTC LV_DLLIMPORT uint32_t LvGetVersion ( )

Returns SynView version.

Returns

The returned doubleword contains the build version in the low word and the high word is the major version in the upper byte and subversion in the lower byte. For example:

```
uint32_t Version = LvGetVersion();
printf("SynView %d.%02d.%03d",
        ((Version >> 24) & 0xFF),
        ((Version >> 16) & 0xFF),
        (Version & 0xFFFF));
```

5.2.2.8 LV_EXTC LV_DLLIMPORT void LvLog (const char * *pLogMessage*)

Adds a line to the sv.synview.log. The SynView log is a tool for New Electronic Technology technical support, in some cases may be useful to put to the log additional info from your code.

Parameters

pLogMessage	Pointer to the null teminated string with the message.

5.2.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvOpenLibrary ( )

Opens the SynView library. This must be done before you can use any other SynView function (with the exception of LvGetVersion() and LvGetErrorMessage()). If you are using SynView in Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is already fully initialized and there are no restrictions about synchronization (DllMain has such restrictions). If you call this function multiple times, you must balance it by the same number of the LvCloseLibrary() calls. Only the first call will actually do the initialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3 SynView System module functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvUpdateSystemList ()
- LV_EXTC LV_DLLIMPORT LvStatus LvGetNumberOfSystems (uint32_t *pNumberOfSystems)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetSystemId (uint32_t Index, char *pSystemId, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetSystemIdSize (uint32_t Index, size_t *pSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemOpen (const char *pSystemId, LvHSystem *phSystem)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemClose (LvHSystem *phSystem)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemUpdateInterfaceList (LvHSystem hSystem, uint32_t Timeout)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetNumberOfInterfaces (LvHSystem hSystem, uint32_t *p↩ NumberOfInterfaces)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetInterfaceId (LvHSystem hSystem, uint32_t Index, char *pInterfaceId, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetInterfaceIdSize (LvHSystem hSystem, uint32_t Index, size_t *pSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvSystemFindInterface (LvHSystem hSystem, LvEnum FindBy, const char *pFindStr, char *pInterfaceId, size_t Size)

5.3.1 Detailed Description

5.3.2 Function Documentation

5.3.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvGetNumberOfSystems (uint32_t * *pNumberOfSystems*)

Returns the number of systems found after the LvUpdateSystemList() call. Typical use of this function is in iterating systems using the LvGetSystemId() function.

Parameters

pNumberOf↩ Systems	The number of systems found.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvGetSystemId (uint32_t *Index*, char * *pSystemId*, size_t *Size*)

Returns the string ID of the system at given index. This ID is used in the LvSystemOpen() function for opening the system.

Parameters

Index	Zero-based index of the system, a value >= 0 and < number of systems, returned by the LvGetNumberOfSystems() function.
pSystemId	Pointer to a string buffer, where the system ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvGetSystemIdSize (uint32_t *Index*, size_t * *pSize*)

Returns the size of the string buffer needed to hold the system ID string, including the terminating zero character.

Parameters

Index	Zero-based index of the system, a value >= 0 and < number of systems, returned by the LvGetNumberOfSystems() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvSystemClose (LvHSystem * *phSystem*)

Closes the opened system. Actually it means freeing the coresponding GenTL library. Be sure you first close all dependent modules (Interface, Event etc.). If the System was opened multiple times, it only decreases the reference counter (see the note by the LvSystemOpen()).

Parameters

phSystem	Pointer to a handle to the System module, obtained from the LvSystemOpen() function. The handle is assigned 0 after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvSystemFindInterface (LvHSystem *hSystem,* LvEnum *FindBy,* const char * *pFindStr,* char * *pInterfaceId,* size_t *Size*)

Finds the interface according specified criteria and returns a string ID of the interface, which is used by the LvInterfaceOpen() function. This function does not update the interface list - if you need to do so, call the LvSystemUpdateInterfaceList() function before calling this function.

Parameters

hSystem	A handle to the System module, obtained from the LvSystemOpen() function.
FindBy	Specifies by which criteria to find the interface. Use one of the LvFindBy constants.
pFindStr	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the LvFindBy_IPAddress, this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
pInterfaceId	Pointer to a string buffer, where the interface ID will be placed.
Size	Size of the string buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions. If the Interface is found, the returned status is LVSTATUS_OK.

5.3.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetInterfaceId (LvHSystem *hSystem,* uint32_t *Index,* char * *pInterfaceId,* size_t *Size*)

Returns a string ID of the interface, which is used by the LvInterfaceOpen() function.

Parameters

hSystem	A handle to the System module, obtained from the LvSystemOpen() function.
Index	Zero-based index of the interface, a value >= 0 and < number of interfaces, returned by the LvSystemGetNumberOfInterfaces() function.
pInterfaceId	Pointer to a string buffer, where the interface ID will be placed.
Size	Size of the string buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetInterfaceIdSize (LvHSystem *hSystem*, uint32_t *Index*, size_t * *pSize*)

Returns the size of the string buffer needed to hold the Interface ID string, including the terminating zero character.

Parameters

hSystem	A handle to the System module, obtained from the LvSystemOpen() function.
Index	Zero-based index of the interface, a value >= 0 and < number of interfaces, returned by the LvSystemGetNumberOfInterfaces() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvSystemGetNumberOfInterfaces (LvHSystem *hSystem*, uint32_t * *pNumberOfInterfaces*)

Returns the number of found interfaces, after the LvSystemUpdateInterfaceList() call.

Parameters

hSystem	A handle to the System module, obtained from the LvSystemOpen() function.
pNumberOf↩ Interfaces	Number of interfaces found.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvSystemOpen (const char * *pSystemId*, LvHSystem * *phSystem*)

Opens the System module. Opening the system actually means loading the corresponding GenTL library. Note that before you can open the System, the LvOpenLibrary() must be called. The same system can be open multiple times (there is a reference counter inside); in such case there must be also the same number of LvSystemClose() calls used (every open increase the reference count and every close decreases it).

Parameters

pSystemId	A string ID of the system. This can be either an empty string - then the default system is opened, or it can be a string obtained from the LvGetSystemId() function.
phSystem	Pointer to a handle to the opened System module.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.10 LV_EXTC LV_DLLIMPORT LvStatus LvSystemUpdateInterfaceList (LvHSystem *hSystem,* uint32_t *Timeout*)

Updates the internal list of available interfaces. You can then iterate through them by LvSystemGetNumberOf↩Interfaces() and LvSystemGetInterfaceId().

Parameters

hSystem	A handle to the System module, obtained from the LvSystemOpen() function.
Timeout	Specifies a timeout in ms for searching the interfaces. This applies only to special cases of interfaces, where some delay can happen; common interfaces are detected without any significant delays.

Returns

If the timeout has expired while waiting for the completion, the function returns LVSTATUS_TIMEOUT, otherwise it returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.3.2.11 LV_EXTC LV_DLLIMPORT LvStatus LvUpdateSystemList ( )

Updates the list of systems available. This function must be called before iterating through the systems by the LvGetNumberOfSystems() and LvGetSystemId() functions. The systems are physically represented by GenTL libraries available in the operating systems, this call searches for them in standard locations. See also the description of the sv.synview.ini file in the SynView User's Guide. Note that this function is seldom needed, most applications will work with the default system (see LvSystemOpen() for details).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.4 SynView Interface module functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceOpen (LvHSystem hSystem, const char *pInterfaceId, LvHInterface *phInterface)
- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceClose (LvHInterface *phInterface)
- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceUpdateDeviceList (LvHInterface hInterface, uint32_t Timeout)
- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceGetNumberOfDevices (LvHInterface hInterface, uint32_t *pDevices)
- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceGetDeviceId (LvHInterface hInterface, uint32_t Index, char *pDeviceId, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceGetDeviceIdSize (LvHInterface hInterface, uint32_t Index, size_t *pSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceFindDevice (LvHInterface hInterface, LvEnum FindBy, const char *pFindStr, char *pDeviceId, size_t Size)

5.4.1 Detailed Description

5.4.2 Function Documentation

5.4.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceClose (LvHInterface * *phInterface*)

Closes the interface. If the Interface was opened multiple times, it only decreases the reference counter (see a note by the LvInterfaceOpen()). Be sure you first close all dependent modules (Device, Event etc.).

Parameters

phInterface	Pointer to a handle to the Interface module, obtained from the LvInterfaceOpen() function. The handle is assigned 0 after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.4.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceFindDevice (LvHInterface *hInterface,* LvEnum *FindBy,* const char * *pFindStr,* char * *pDeviceId,* size_t *Size*)

Finds the device according specified criteria and returns a string ID of the device, which can be used by the LvDeviceOpen() function. This function does not update the device list - if you need to do so, call the LvInterfaceUpdateDeviceList() function before calling this function.

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
FindBy	Specifies by which criteria to find the interface. Use one of the LvFindBy constants.
pFindStr	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the LvFindBy_IPAddress, this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).

pDeviceId	Pointer to a string buffer, where the device ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions. If the Device is found, the returned status is LVSTATUS_OK.

5.4.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceGetDeviceId (LvHInterface *hInterface,* uint32_t *Index,* char * *pDeviceId,* size_t *Size*)

Returns a string ID of the device at specified position in the list. Note that this device ID is stable (the same physical device has always the same ID) and it is unique (no other physical device can have the same ID). To hardcode directly the device ID in your application is not recommended, as the application would not be usable, when a deffective device needs to be replaced. The SynView User's Guide discuss the ways, how to solve such maintainability demands.

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
Index	Zero-based index of the device, a value >= 0 and < number of devices, returned by the LvInterfaceGetNumberOfDevices() function.
pDeviceId	Pointer to a string buffer, where the device ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.4.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceGetDeviceIdSize (LvHInterface *hInterface,* uint32_t *Index,* size_t * *pSize*)

Returns the size of the string buffer needed to hold the Device ID string, including the terminating zero character.

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
Index	Zero-based index of the device, a value >= 0 and < number of devices, returned by the LvInterfaceGetNumberOfDevices() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.4.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceGetNumberOfDevices (LvHInterface *hInterface,* uint32_t * *pDevices*)

Returns the number of devices found by the LvInterfaceUpdateDeviceList() function.

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
pDevices	Number of devices found.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.4.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceOpen (LvHSystem *hSystem*, const char * *pInterfaceId*, LvHInterface * *phInterface*)

Opens the Interface module. The same Interface can be open multiple times (there is a reference counter inside); in such case there must be also the same number of LvInterfaceClose() calls used (every open increase the reference count and every close decreases it) .

Parameters

hSystem	A handle to the System module, obtained from the LvSystemOpen() function.
pInterfaceId	A string interface ID, obtained by the LvSystemGetInterfaceId().
phInterface	In this parameter the handle to the interface is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.4.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvInterfaceUpdateDeviceList (LvHInterface *hInterface*, uint32_t *Timeout*)

Updates the Device list. The available devices are searched.

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
Timeout	Specifies a timeout in ms for searching the devices.

Returns

If the timeout has expired while waiting for the completion, the function returns LVSTATUS_TIMEOUT, otherwise it returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5 SynView Device module functions

Modules

- SynView Firmware update functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceOpen (LvHInterface hInterface, const char *pDeviceId, LvHDevice *phDevice, LvEnum Access)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceReOpen (LvHInterface hInterface, const char *pDeviceId, LvHDevice hDevice, LvEnum Access=LvDeviceAccess_Exclusive)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceClose (LvHDevice *phDevice)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceGetNumberOfStreams (LvHDevice hDevice, uint32_t *pNumberOfStreams)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceGetStreamId (LvHDevice hDevice, uint32_t Index, char *pStreamId, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceGetStreamIdSize (LvHDevice hDevice, uint32_t Index, size_t *pSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionStart (LvHDevice hDevice, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionStop (LvHDevice hDevice, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionAbort (LvHDevice hDevice, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionArm (LvHDevice hDevice, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceSaveSettings (LvHDevice hDevice, const char *pId, const char *pFileName, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceLoadSettings (LvHDevice hDevice, const char *pId, const char *pFileName, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceLoadBatch (LvHDevice hDevice, const char *pFileName)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceUniSetLut (LvHDevice hDevice, LvEnum Selector, void *pLUT, size_t Size, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvDeviceUniGetLut (LvHDevice hDevice, LvEnum Selector, void *pLUT, size_t Size, uint32_t Options)
- LvStatus LvDevice::LoadBatch (const char *pFileName)

5.5.1 Detailed Description

5.5.2 Function Documentation

5.5.2.1 LvStatus LvDevice::LoadBatch (const char * *pFileName*)

Loads device batch commands from a file.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
pFileName	The file specification, where the configuration is stored. It is a text file.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionAbort (LvHDevice *hDevice,* uint32_t *Options*)

Aborts athe acquisition immediately, without completing the current Frame or waiting on a trigger.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionArm (LvHDevice *hDevice*, uint32_t *Options*)

Prepares the device for acquisition, so that the acquisition using the LvDeviceAcquisitionStart() function then can start fast. If it is not called before LvDeviceAcquisitionStart(), it is called automatically inside the LvDevice↩ AcquisitionStart().

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionStart (LvHDevice *hDevice*, uint32_t *Options*)

Starts the acquisition. This function includes more than just calling the AcquisitionStart remote command on the device - it checks the size of the buffers, prepares the streams for the start, locks GenTL params and then starts the acquisition on the device itself. Always check the success of this function call.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceAcquisitionStop (LvHDevice *hDevice*, uint32_t *Options*)

Stops the acquisition.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceClose (LvHDevice * *phDevice*)

Closes the Device. Be sure you first close all dependent modules (Stream, Event etc.).

Parameters

phDevice	Pointer to a handle to the Device module, obtained from the LvDeviceOpen() function. This handle is assigned 0 after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceGetNumberOfStreams (LvHDevice *hDevice*, uint32_t * *pNumberOfStreams*)

Returns the number of available stream types for this device. You can then iterate the streams by the LvDeviceGetStreamId() function.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
pNumberOfStreams	The number of streams is returned here.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceGetStreamId (LvHDevice *hDevice*, uint32_t *Index*, char * *pStreamId*, size_t *Size*)

Returns a string Stream ID, needed for opening the stream.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Index	Zero-based index of the stream type, a value >= 0 and < number of streams, returned by the LvDeviceGetNumberOfStreams() function.
pStreamId	Pointer to a string buffer, where the stream ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceGetStreamIdSize (LvHDevice *hDevice*, uint32_t *Index*, size_t * *pSize*)

Returns the size of the string buffer needed to hold the stream ID at given index, including the space for the terminating zero character.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Index	Zero-based index of the stream type, a value >= 0 and < number of streams, returned by the LvDeviceGetNumberOfStreams() function.

pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.10 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceLoadBatch (LvHDevice *hDevice*, const char * *pFileName*)

Loads device batch commands from a file.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
pFileName	The file specification, where the configuration is stored. It is a text file.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.11 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceLoadSettings (LvHDevice *hDevice*, const char * *pId*, const char * *pFileName*, uint32_t *Options*)

Loads the device settings from a file. In the Options can be specified which parts of the device configuration are to be loaded - the Remote, Local and/or GenTL part. Note that there are several factors, which can break the compatibility of the settings file with the current device:

- When the current device is of different vendor/model, the settings file is most probably not compatible.
- When the current device is of the same vendor/model, but uses a different firmware version - this could mean that some remote device features are not present or behave differently.
- When the XML version of the Local and GenTL features changes - again this could mean that some features are not present or behave differently. For this reason this function checks the versions and if not the same, it returns either the LVSTATUS_SETTINGS_INCOMPATIBLE_MODEL or LVSTATUS_SETTINGS_INCOMPATIBLE_VERSION error states. As the difference in versions might not necessarily mean a real incompability, you can use the LvSaveFlag_IgnoreVersion and LvSaveFlag_IgnoreModel flags in the Options parameter in order to force this function to try to load the settings even if the possible incompatibility is found.

  Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
pId	A string ID enabling to protect the file. If you specify a non-empty ID in LvDeviceSaveSettings(), you must use the same ID in LvDeviceLoadSettings(), otherwise the settings are not loaded.
pFileName	The file specification, where the configuration is stored. It is a text file.
Options	One or or-ed combination of LvSaveFlag definitions.

  Returns

  Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.12 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceOpen (LvHInterface *hInterface*, const char * *pDeviceId*, LvHDevice * *phDevice*, LvEnum *Access*)

Opens the Device module. This physically means opening a connection with the device and retrieving a list of device remote features. Always check the success of this function call; the opening may fail for example when you request an exclusive access and the device is already open by some other application.

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
pDeviceId	A string ID of the device, obtained by LvInterfaceGetDeviceId() function.
phDevice	In this parameter the handle to the Device is returned.
Access	Desired device access, one of the LvDeviceAccess constants.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.13 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceReOpen (LvHInterface *hInterface*, const char * *pDeviceId*, LvHDevice *hDevice*, LvEnum *Access* = LvDeviceAccess_Exclusive)

Re-Opens the Device. does not create a new class, just tries to reopen the connection

Parameters

hInterface	A handle to the Interface module, obtained from the LvInterfaceOpen() function.
pDeviceId	A string ID of the device, obtained by LvInterfaceGetDeviceId() function.
hDevice	The handle to the Device
Access	Desired device access, one of the LvDeviceAccess constants.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.14 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceSaveSettings (LvHDevice *hDevice*, const char * *pId*, const char * *pFileName*, uint32_t *Options*)

Saves the device settings to a file. In the Options can be specified which parts of the device configuration are to be saved - the Remote, Local and/or GenTL part. See alo notes by LvDeviceLoadSettings().

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
pId	A string ID enabling to protect the file. If you specify a non-empty ID in LvDeviceSaveSettings(), you must use the same ID in LvDeviceLoadSettings(), otherwise the settings are not loaded.
pFileName	The file specification, to which the configuration is stored. It is a text file.
Options	One or or-ed combination of LvSaveFlag definitions.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.15 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceUniGetLut (LvHDevice *hDevice*, LvEnum *Selector*, void * *pLUT*, size_t *Size*, uint32_t *Options*)

Gets the lookup table. See LvDeviceUniSetLut() for details. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format. So you can for example read the 12-bit LUT to 8-bit LUT array.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Selector	Lookup table selector, see LvLUTSelector.
pLUT	Pointer to the lookup table.
Size	Size of the lookup table. The only valid values are • 256 for 8-bit LUT • 2048 for 10-bit LUT • 8192 for 12-bit LUT
Options	The LvUniLutFlags_HwLut option can be used to apply to function directly on HW LUT.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.5.2.16 LV_EXTC LV_DLLIMPORT LvStatus LvDeviceUniSetLut (LvHDevice *hDevice*, LvEnum *Selector*, void * *pLUT*, size_t *Size*, uint32_t *Options*)

Sets the lookup table. If the hardware lookup table is available, it is used, otherwise a software lookup table is set. This function belongs to a set of functions, which unify the functionality of devices with real-time processing embedded in hardware (RTF) and devices without real-time processing, for which the processing is made by software. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
Selector	Lookup table selector, see LvLUTSelector.
pLUT	Pointer to the lookup table.
Size	Size of the lookup table. The only valid values are • 256 for 8-bit LUT • 2048 for 10-bit LUT • 8192 for 12-bit LUT
Options	The LvUniLutFlags_HwLut option can be used to apply to function directly on HW LUT.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.6 SynView Stream module functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvStreamOpen (LvHDevice hDevice, const char *pStreamId, LvHStream *phStream)
- LV_EXTC LV_DLLIMPORT LvStatus LvStreamClose (LvHStream *phStream)
- LV_EXTC LV_DLLIMPORT LvStatus LvStreamGetBufferAt (LvHStream hStream, uint32_t BufferIndex, LvHBuffer *phBuffer)
- LV_EXTC LV_DLLIMPORT LvStatus LvStreamFlushQueue (LvHStream hStream, LvEnum Operation)
- LV_EXTC LV_DLLIMPORT LvStatus LvStreamStart (LvHStream hStream, uint32_t StartFlags, uint32_t ImagesToAcquire)
- LV_EXTC LV_DLLIMPORT LvStatus LvStreamStop (LvHStream hStream, uint32_t StopFlags)

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvStreamClose (LvHStream * *phStream*)

Closes the Stream. Be sure you first close all dependent modules (Buffers, Event, Renderer etc.).

Parameters

phStream	Pointer to a handle to the Stream module, obtained from the LvStreamOpen() function. This handle is assigned 0 after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.6.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvStreamFlushQueue (LvHStream *hStream,* LvEnum *Operation*)

Moves the buffers according to the LvQueueOperation specified.

Parameters

hStream	A handle to the Stream module, obtained from the LvStreamOpen() function.
Operation	One of the LvQueueOperation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.6.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvStreamGetBufferAt (LvHStream *hStream,* uint32_t *BufferIndex,* LvHBuffer * *phBuffer*)

Returns the buffer handle at given index.

Parameters

hStream	A handle to the Stream module, obtained from the LvStreamOpen() function.
BufferIndex	Zero-based index.
phBuffer	In this parameter the buffer handle is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.6.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvStreamOpen (LvHDevice *hDevice*, const char * *pStreamId*, LvHStream * *phStream*)

Opens the stream module, associated with the device.

Parameters

hDevice	A handle to the Device module, obtained from the LvDeviceOpen() function.
pStreamId	A string ID of the stream, obtained from LvDeviceGetStreamId(). If an empty string is used, the first found stream is opened. This is usually the image data stream.
phStream	In this parameter the handle to the Stream is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.6.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvStreamStart (LvHStream *hStream*, uint32_t *StartFlags*, uint32_t *ImagesToAcquire*)

Starts the stream. This function need not be used on the image stream, where it is called automatically in the LvDeviceAcquisitionStart() function.

Parameters

hStream	A handle to the Stream module, obtained from the LvStreamOpen() function.
StartFlags	One of the GroupSynView_StreamStartFlags.
ImagesToAcquire	Number of images to acquire.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.6.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvStreamStop (LvHStream *hStream*, uint32_t *StopFlags*)

Stops the stream. This function need not be used on the image stream, where it is called automatically in the LvDeviceAcquisitionStop() function.

Parameters

hStream	A handle to the Stream module, obtained from the LvStreamOpen() function.
StopFlags	One of the GroupSynView_StreamStopFlags.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7 SynView Buffer module functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvBufferOpen (LvHStream hStream, void *pDataPointer, size_t DataSize, void *pUserPointer, uint32_t Options, LvHBuffer *phBuffer)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferClose (LvHBuffer *phBuffer)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferAttachProcessBuffer (LvHBuffer hBuffer, void *pDataPointer, size_t DataSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferQueue (LvHBuffer hBuffer)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferParseChunkData (LvHBuffer hBuffer, uint32_t UpdateLayout)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferSaveImageToBmpFile (LvHBuffer hBuffer, const char *pFileName)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferSaveImageToJpgFile (LvHBuffer hBuffer, const char *pFileName, uint32_t Quality)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferSaveImageToTifFile (LvHBuffer hBuffer, const char *pFileName, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferGetImgInfo (LvHBuffer hBuffer, LvipImgInfo *pImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferGetLastPaintRect (LvHBuffer hBuffer, int32_t *pX, int32_t *pY, int32_t *pWidth, int32_t *pHeight)
- LV_EXTC LV_DLLIMPORT LvStatus LvBufferUniCalculateWhiteBalance (LvHBuffer hBuffer)

5.7.1 Detailed Description

5.7.2 Function Documentation

5.7.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvBufferAttachProcessBuffer (LvHBuffer *hBuffer*, void * *pDataPointer*, size_t *DataSize*)

Attaches a process buffer to a buffer. The process buffer may be needed for software processing, for example Bayer decoding, if the device hardware is not capable of it. The process buffer can be either supplied by the application by this function, or allocated automatically by SynView upon need.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
pDataPointer	Pointer to the supplied buffer.
DataSize	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvBufferClose (LvHBuffer * *phBuffer*)

Closes the buffer. On the GenTL level it corresponds to the DSRevokeBuffer() function.

Parameters

phBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function. This handle is assigned 0 after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvBufferGetImgInfo (LvHBuffer *hBuffer,* LvipImgInfo * *pImgInfo,* uint32_t *Options*)

Fills the LvipImgInfo structure for the image in the buffer. This simplifies a direct use of the SynView Image Processing Library. If the image is processed, the image info points to the processed image, othewise it points to the original image.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
pImgInfo	Pointer to the ImgInfo structure, to which are the image parameters stored.
Options	Currently unused, must be 0.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvBufferGetLastPaintRect (LvHBuffer *hBuffer,* int32_t * *pX,* int32_t * *pY,* int32_t * *pWidth,* int32_t * *pHeight*)

Returns the rectangle to which the buffer was last painted. This is useful namely in case you have a tile mode and want to identify the buffer according a mouse click location. If the buffer was not yet painted by the renderer, the returned values are 0.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
pX	Pointer to X offset in pixels.
pY	Pointer to Y offset in pixels.
pWidth	Pointer to Width in pixels.
pHeight	Pointer to Height in pixels.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvBufferOpen (LvHStream *hStream,* void * *pDataPointer,* size_t *DataSize,* void * *pUserPointer,* uint32_t *Options,* LvHBuffer * *phBuffer*)

Opens a buffer. On the GenTL level it corresponds to DSAnnounceBuffer() or DSAllocAndAnnounceBuffer().

Parameters

hStream	A handle to the Stream module, obtained from the LvStreamOpen() function.
pDataPointer	Pointer to image data buffer. This can be supplied by the application (in such case the Data↩ Size must be set to the actual size of the buffer), or can be left NULL - in such case the buffer is allocated by SynView.
DataSize	Size of the buffer supplied, or 0 if the pDataPointer is NULL.
pUserPointer	A user pointer, which is then passed back in the LvEventCallbackNewBufferFunct(). It enables the application to reference some own data structure associated with the buffer.
Options	Currently unused, must be 0.
phBuffer	To this parameter the handle to the buffer is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvBufferParseChunkData (LvHBuffer *hBuffer,* uint32_t *UpdateLayout*)

Parses the chunk data of the image. The chunk data are then accessible as device remote features (for example LvDevice_ChunkTimestamp).

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
UpdateLayout	If set to 1, the layout of chunk data is decoded. If set to 0, the data are only read from already decoded layout - this is faster. Usually, the layout of the chunk data is constant, so it needs to be decoded only at first call of this function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvBufferQueue (LvHBuffer *hBuffer*)

Puts the buffer to the input buffer pool. This is an important part of the image handling loop: after the buffer with the acquired image is passed to the application, the application must return it to the input buffer pool by this function after processing.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvBufferSaveImageToBmpFile (LvHBuffer *hBuffer*, const char * *pFileName*)

Saves the image to a file in Windows BMP format. If the image is in the pixel format not compatible with the BMP format, it is automatically converted.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
pFileName	The file name. Be sure to specify it with the full path.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvBufferSaveImageToJpgFile (LvHBuffer *hBuffer*, const char * *pFileName*, uint32_t *Quality*)

Saves the image to a file in JPEG format. If the image is in the pixel format not compatible with the JPEG format, it is automatically converted.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
pFileName	The file name. Be sure to specify it with the full path.
Quality	The quality factor in range from from 1 to 100. The higher is the factor, the higher is the quality and lower the compression.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.10 LV_EXTC LV_DLLIMPORT LvStatus LvBufferSaveImageToTifFile (LvHBuffer *hBuffer,* const char * *pFileName,* uint32_t *Options*)

Saves the image to a file in the TIFF format. If the image is in the pixel format not compatible with the TIF format, it is automatically converted.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.
pFileName	The file name. Be sure to specify it with the full path.
Options	Options for saved pixel format. The LvipOption_TiffConvertTo16Bit flag can be used there.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.7.2.11 LV_EXTC LV_DLLIMPORT LvStatus LvBufferUniCalculateWhiteBalance (LvHBuffer *hBuffer*)

Calculates white balance factors from the current image.

Parameters

hBuffer	A handle to the Buffer module, obtained from the LvBufferOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8 SynView Event module functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvEventOpen (LvHModule hModule, LvEnum EventType, LvHEvent ∗phEvent)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventClose (LvHEvent ∗phEvent)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventKill (LvHEvent hEvent)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventFlush (LvHEvent hEvent)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventWaitAndGetData (LvHEvent hEvent, void ∗pBuffer, size_t ∗pSize, uint32_t Timeout)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventWaitAndGetNewBuffer (LvHEvent hEvent, LvHBuffer ∗phBuffer, uint32_t Timeout)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventGetDataInfo (LvHEvent hEvent, void ∗pInBuffer, size_t InSize, LvEnum Info, void ∗pBuffer, size_t ∗pSize, LvEnum ∗pType, int32_t Param)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventPutData (LvHEvent hEvent, void ∗pBuffer, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventSetCallback (LvHEvent hEvent, LvEventCallbackFunct pFunction, void ∗pUserParam)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventSetCallbackNewBuffer (LvHEvent hEvent, LvEventCallbackNewBufferFunct pFunction, void ∗pUserParam)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventStartThread (LvHEvent hEvent)
- LV_EXTC LV_DLLIMPORT LvStatus LvEventStopThread (LvHEvent hEvent)

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvEventClose (LvHEvent ∗ *phEvent*)

Closes the Event module.

Parameters

phEvent	A handle to the Event module, obtained from the LvEventOpen() function. This handle is assigned 0 after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvEventFlush (LvHEvent *hEvent*)

Discards all buffers in the output buffer queue (waiting to be delivered to the application).

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvEventGetDataInfo (LvHEvent *hEvent*, void ∗ *pInBuffer*, size_t *InSize*, LvEnum *Info*, void ∗ *pBuffer*, size_t ∗ *pSize*, LvEnum ∗ *pType*, int32_t *Param*)

Enables to parse the buffer from LvEventWaitAndGetData.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.
pInBuffer	Pointer to a buffer containing event data. This value must not be NULL.
InSize	Size of the provided pInBuffer in bytes.
Info	One of the LvEventDataInfo.
pBuffer	Pointer to a user allocated buffer to receive the requested information. If this parameter is NULL, pSize will contain the minimal size of pBuffer in bytes. If the pType is a string, the size includes the terminating 0.
pSize	Size of the buffer is returned in this parameter.
pType	One of the LvInfoDataType.
Param	Additional parameter, if used, its role is explained by the LvEventDataInfo.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvEventKill (LvHEvent *hEvent*)

Terminates a single wait in the LvEventWaitAndGetData() function.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvEventOpen (LvHModule *hModule,* LvEnum *EventType,* LvHEvent * *phEvent*)

Opens the Event module for specified owner module.

Parameters

hModule	A handle to the System, Device or Stream module.
EventType	One of the LvEventType.
phEvent	To this parameter the Event handle is stored.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvEventPutData (LvHEvent *hEvent,* void * *pBuffer,* size_t *Size*)

Puts a new event to Event ouptut queue. This function can be used only for user-defined events.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.
pBuffer	Pointer to event data.

Size	Size of the event data.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvEventSetCallback (LvHEvent *hEvent*, LvEventCallbackFunct *pFunction*, void * *pUserParam*)

Specifies a callback function for the event thread. Note that the callback function cannot be a method of a class.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.
pFunction	The callback function in the forms of LvEventCallbackFunct.
pUserParam	User parameter, which will be passed to each callback call.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvEventSetCallbackNewBuffer (LvHEvent *hEvent*, LvEventCallbackNewBufferFunct *pFunction*, void * *pUserParam*)

Specifies a callback function for the thread of the Event of the LvEventType_NewBuffer. Once the application specifies this callback, it becomes responsible for returning the image buffers to the input buffer pool. Note that the callback function cannot be a method of a class.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.
pFunction	The callback function in the forms of LvEventCallbackNewBufferFunct.
pUserParam	User parameter, which will be passed to each callback call.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvEventStartThread (LvHEvent *hEvent*)

Starts an internal thread, which waits for events and passes them to specified callback function. When the thread is started, the application must no longer call the LvEventWaitAndGetData() or LvEventWaitAndGetNewBufer() functions - this is called internally in the thread and upon return from this function a callback function is called.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.10 LV_EXTC LV_DLLIMPORT LvStatus LvEventStopThread (LvHEvent *hEvent*)

Stops the event internal thread.

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.11 LV_EXTC LV_DLLIMPORT LvStatus LvEventWaitAndGetData (LvHEvent *hEvent*, void * *pBuffer*, size_t * *pSize*, uint32_t *Timeout*)

Waits for the event and gets its data in one atomic operation. Use this function only for events other than LvEvent↩ Type_NewBuffer, for the the LvEventType_NewBuffer event type use the LvEventWaitAndGetNewBuffer() function instead. Do not use this function if you use the callback - see LvEventSetCallback() or LvEventSetCallbackNew↩ Buffer().

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.
pBuffer	Pointer to a user allocated buffer to receive the event data. The buffer can be parsed by the LvEventGetDataInfo() function.
pSize	Size of the buffer must be specified in this parameter and after the function returns, the actual size is returned in this parameter.
Timeout	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the LvEventKill() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.8.2.12 LV_EXTC LV_DLLIMPORT LvStatus LvEventWaitAndGetNewBuffer (LvHEvent *hEvent*, LvHBuffer * *phBuffer*, uint32_t *Timeout*)

Waits for the event and gets its data in one atomic operation. Use this function only for events of the LvEvent↩ Type_NewBuffer type. Do not use this function if you use the callback - see LvEventSetCallback() or LvEventSet↩ CallbackNewBuffer().

Parameters

hEvent	A handle to the Event module, obtained from the LvEventOpen() function.
phBuffer	The handle to the received buffer is returned in this parameter.
Timeout	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the LvEventKill() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.9 SynView Renderer module functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvRendererOpen (LvHStream hStream, LvHRenderer *phRenderer)
- LV_EXTC LV_DLLIMPORT LvStatus LvRendererClose (LvHRenderer *phRenderer)
- LV_EXTC LV_DLLIMPORT LvStatus LvRendererSetWindow (LvHRenderer hRenderer, void *pDisplay, int64_t hWindow)
- LV_EXTC LV_DLLIMPORT LvStatus LvRendererCanDisplayImage (LvHRenderer hRenderer, LvHBuffer hBuffer, uint32_t RenderFlags)
- LV_EXTC LV_DLLIMPORT LvStatus LvRendererDisplayImage (LvHRenderer hRenderer, LvHBuffer hBuffer, uint32_t RenderFlags)
- LV_EXTC LV_DLLIMPORT LvStatus LvRendererRepaint (LvHRenderer hRenderer, uint32_t RenderFlags)

5.9.1 Detailed Description

5.9.2 Function Documentation

5.9.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvRendererCanDisplayImage (LvHRenderer *hRenderer,* LvHBuffer *hBuffer,* uint32_t *RenderFlags*)

Checks, if the image can be displayed. Namely the possibility to convert the image to desired display pixel format is checked.

Parameters

hRenderer	A handle to the Renderer module, obtained from the LvRendererOpen() function.
hBuffer	The buffer to be displayed.
RenderFlags	Zero or a combination of LvRenderFlags.

Returns

Returns the LvStatus value; the value LVSTATUS_OK indicates the display is possible, the value LVSTATUS_DISPLAY_CANNOT_DISPLAY indicates impossibility of pixel format conversion or a misconfiguration of the renderer. See LvStatus definitions.

5.9.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvRendererClose (LvHRenderer * *phRenderer*)

Closes the Renderer module.

Parameters

phRenderer	A handle to the Renderer module, obtained from the LvRendererOpen() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.9.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvRendererDisplayImage (LvHRenderer *hRenderer,* LvHBuffer *hBuffer,* uint32_t *RenderFlags*)

Displays the image. The image display mode is set by Renderer features, see LvRendererFtr.

Parameters

hRenderer	A handle to the Renderer module, obtained from the LvRendererOpen() function.
hBuffer	The buffer to be displayed.
RenderFlags	Zero or a combination of LvRenderFlags.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.9.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvRendererOpen (LvHStream *hStream*, LvHRenderer * *phRenderer*)

Opens the Renderer module for image display. The renderer attepts to load the sv.synview.display library. In case of SynView installation in an operating system without possibility to graphically display (for example Linux without XWindows), the load of this library fails and the calls to Renderer functions will return errors.

Parameters

hStream	A handle to the Stream module, obtained from the LvStreamOpen() function.
phRenderer	In this parameter the handle to the renderer is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.9.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvRendererRepaint (LvHRenderer *hRenderer*, uint32_t *RenderFlags*)

Repaints the contents of the display window. In order to be able to repaint, all images to be displayed must be still held by the application, i.e. must not be returned to the input buffer pool. See also LvStream_LvPostponeQueue↩ Buffers feature. A typical usage of this function is in the WM_PAINT handler in a Windows application.

Parameters

hRenderer	A handle to the Renderer module, obtained from the LvRendererOpen() function.
RenderFlags	Zero or a combination of LvRenderFlags.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.9.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvRendererSetWindow (LvHRenderer *hRenderer*, void * *pDisplay*, int64_t *hWindow*)

Sets the target window, in which the renderer has to display. Note that the application itself assure any repainting (when the window need to be repainted due to a movement of overlapping) - use LvRendererRepaint() in such case.

Parameters

hRenderer	A handle to the Renderer module, obtained from the LvRendererOpen() function.
pDisplay	Pointer to the display. It is defined as void* in order to make SynView header files independent on XWindows.

hWindow	Handle to the window.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10 SynView Feature control functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvGetNumFeatures (LvHModule hModule, LvEnum FeatureGroup, uint32_t *pNumItems)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetFeatureAt (LvHModule hModule, LvEnum FeatureGroup, uint32_t Index, LvFeature *pItem, uint32_t *pLevel)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetFeatureByName (LvHModule hModule, LvEnum FeatureGroup, const char *pName, LvFeature *pItem)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsImplemented (LvHModule hModule, LvFeature Feature)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsImplementedByName (LvHModule hModule, LvEnum FeatureGroup, const char *pName)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsImplementedEnumEntry (LvHModule hModule, LvFeature Feature, LvEnum EnumEntry)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsAvailable (LvHModule hModule, LvFeature Feature)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsAvailableByName (LvHModule hModule, LvEnum FeatureGroup, const char *pName)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsAvailableEnumEntry (LvHModule hModule, LvFeature Feature, LvEnum EnumEntry)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsReadable (LvHModule hModule, LvFeature Feature)
- LV_EXTC LV_DLLIMPORT uint32_t LvIsWritable (LvHModule hModule, LvFeature Feature)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetType (LvHModule hModule, LvFeature Feature, LvEnum *pFtrType, LvEnum *pFtrGui, LvEnum *pFtrGroup)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetBool (LvHModule hModule, LvFeature Feature, int32_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetBool (LvHModule hModule, LvFeature Feature, int32_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInt32 (LvHModule hModule, LvFeature Feature, int32_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetInt32 (LvHModule hModule, LvFeature Feature, int32_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInt32Range (LvHModule hModule, LvFeature Feature, int32_t *pMinValue, int32_t *pMaxValue, int32_t *pIncrement)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInt64 (LvHModule hModule, LvFeature Feature, int64_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetInt64 (LvHModule hModule, LvFeature Feature, int64_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInt64Range (LvHModule hModule, LvFeature Feature, int64_t *pMinValue, int64_t *pMaxValue, int64_t *pIncrement)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInt (LvHModule hModule, LvFeature Feature, int64_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetInt (LvHModule hModule, LvFeature Feature, int64_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetIntRange (LvHModule hModule, LvFeature Feature, int64_t *pMinValue, int64_t *pMaxValue, int64_t *pIncrement)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetFloat (LvHModule hModule, LvFeature Feature, double *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetFloat (LvHModule hModule, LvFeature Feature, double Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetFloatRange (LvHModule hModule, LvFeature Feature, double *pMinValue, double *pMaxValue, double *pIncrement)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetString (LvHModule hModule, LvFeature Feature, char *pValue, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetStringSize (LvHModule hModule, LvFeature Feature, size_t *pSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetString (LvHModule hModule, LvFeature Feature, const char *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetBuffer (LvHModule hModule, LvFeature Feature, void *pBuffer, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetBufferSize (LvHModule hModule, LvFeature Feature, size_t *pSize)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetBuffer (LvHModule hModule, LvFeature Feature, void *pBuffer, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetPtr (LvHModule hModule, LvFeature Feature, void **ppValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetPtr (LvHModule hModule, LvFeature Feature, void *pValue)

- LV_EXTC LV_DLLIMPORT LvStatus LvGetEnum (LvHModule hModule, LvFeature Feature, LvEnum ∗pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetEnum (LvHModule hModule, LvFeature Feature, LvEnum Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetEnumStr (LvHModule hModule, LvFeature Feature, char ∗pSymbolicName, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvSetEnumStr (LvHModule hModule, LvFeature Feature, const char ∗pSymbolicName)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetEnumValByStr (LvHModule hModule, LvFeature Feature, const char ∗pSymbolicName, LvEnum ∗pValue, LvEnum ∗pFtrAccess)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetEnumStrByVal (LvHModule hModule, LvFeature Feature, LvEnum Value, char ∗pSymbolicName, size_t SymbolicNameSize, LvEnum ∗pFtrAccess)
- LV_EXTC LV_DLLIMPORT LvStatus LvCmdExecute (LvHModule hModule, LvFeature Feature, uint32_t Timeout)
- LV_EXTC LV_DLLIMPORT LvStatus LvCmdIsDone (LvHModule hModule, LvFeature Feature, uint32_t ∗pIsDone)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetAccess (LvHModule hModule, LvFeature Feature, LvEnum ∗pFtrAccess)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetVisibility (LvHModule hModule, LvFeature Feature, LvEnum ∗pFtrVisibility)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInfo (LvHModule hModule, LvFeature Feature, LvEnum FtrInfo, int32_t ∗pInfo, int32_t Param)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInfoStr (LvHModule hModule, LvFeature Feature, LvEnum FtrInfo, char ∗pInfoStr, size_t Size, int32_t Param)
- LV_EXTC LV_DLLIMPORT LvStatus LvGetInfoStrSize (LvHModule hModule, LvFeature Feature, LvEnum FtrInfo, size_t ∗pSize, int32_t Param)
- LV_EXTC LV_DLLIMPORT LvStatus LvRegisterFeatureCallback (LvHModule hModule, LvFeature Feature, LvFeatureCallbackFunct pFunction, void ∗pUserParam, void ∗pFeatureParam)
- LV_EXTC LV_DLLIMPORT LvStatus LvStartPollingThread (LvHModule hModule, uint32_t PollingTime, int32_t PollChildren)
- LV_EXTC LV_DLLIMPORT LvStatus LvStopPollingThread (LvHModule hModule)
- LV_EXTC LV_DLLIMPORT LvStatus LvPoll (LvHModule hModule)

5.10.1 Detailed Description

5.10.2 Function Documentation

5.10.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvCmdExecute (LvHModule *hModule,* LvFeature *Feature,* uint32_t *Timeout*)

Executes a command.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Timeout	If greater than 0, the LvCmdIsDone() is called in a loop to wait for the command completion, until the LvCmdIsDone() returns true or the Timeout (in milliseconds) expires. If set to 0, no wait is done.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvCmdIsDone (LvHModule *hModule*, LvFeature *Feature*, uint32_t * *pIsDone*)

Checks if the command execution has completed.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pIsDone	In this parameter is returned 1, if the command is completed, otherwise 0.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvGetAccess (LvHModule *hModule*, LvFeature *Feature*, LvEnum * *pFtrAccess*)

Gets the access mode of the feature.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pFtrAccess	The access is returned in this parameter. One of the LvFtrAccess.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvGetBool (LvHModule *hModule*, LvFeature *Feature*, int32_t * *pValue*)

Gets a Boolean value (as 32-bit integer).

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The bool value (as 32-bit integer) is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvGetBuffer (LvHModule *hModule*, LvFeature *Feature*, void * *pBuffer*, size_t *Size*)

Gets a block of data.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

pBuffer	Pointer to a buffer, to which the data will be stored.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvGetBufferSize (LvHModule *hModule*, LvFeature *Feature*, size_t * *pSize*)

Gets the block data size.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pSize	The needed size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvGetEnum (LvHModule *hModule*, LvFeature *Feature*, LvEnum * *pValue*)

Gets the SynView constant for the enumeration entry, if exists. If does not exist, you must work with the string enumeration entry value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	SynView constant for the enum entry is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvGetEnumStr (LvHModule *hModule*, LvFeature *Feature*, char * *pSymbolicName*, size_t *Size*)

Gets the enumeration entry as string (symbolic name). It is not possible to get the needed size for this single feature, instead, it is possible to get the maximum size of the all enum values of this feature, by the LvGetInfo(LvFtrInfo_↩EnumEntryNameMaxSize) function.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

pSymbolicName	A pointer to a string buffer, where the symbolic name will be returned.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvGetEnumStrByVal (LvHModule *hModule*, LvFeature *Feature*, LvEnum *Value*, char * *pSymbolicName*, size_t *SymbolicNameSize*, LvEnum * *pFtrAccess*)

Returns a string symbolic name of the enum entry for the SynView constant.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Value	The SynView constant for the enum entry.
pSymbolicName	Pointer to string buffer, where the symbolic name is returned. Can be NULL.
SymbolicName↩ Size	Size of pSymbolicName buffer.
pFtrAccess	The access mode of the enum entry is returned in this parameter - one of LvFtrAccess. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.10 LV_EXTC LV_DLLIMPORT LvStatus LvGetEnumValByStr (LvHModule *hModule*, LvFeature *Feature*, const char * *pSymbolicName*, LvEnum * *pValue*, LvEnum * *pFtrAccess*)

Gets the SynView constant for the enumeration entry, if exists.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pSymbolicName	A string with symbolic name of the enum entry.
pValue	The SynView constant for the enum entry is returned in this parameter. If the SynView constant does not exist for this enumeration entry, 0 is returned (no error in indicated).
pFtrAccess	The feature access is returned in this parameter - one of GroupSynView_LvFtrAccess. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.11 LV_EXTC LV_DLLIMPORT LvStatus LvGetFeatureAt (LvHModule *hModule*, LvEnum *FeatureGroup*, uint32_t *Index*, LvFeature * *pItem*, uint32_t * *pLevel*)

Returns the feature ID at specified position. Can be used to iterate all the features in a list.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
FeatureGroup	One of the LvFtrGroup.
Index	Zero based index of the feature in the list.
pItem	Feature ID is returned in this parameter.
pLevel	Feature Level expressing its position in the tree is returned in this parameter. The base level has value 1.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.12 LV_EXTC LV_DLLIMPORT LvStatus LvGetFeatureByName (LvHModule *hModule*, LvEnum *FeatureGroup*, const char * *pName*, LvFeature * *pItem*)

Returns a feature ID based on the feature name. This function is a substantial function for the generic approach to the feature - by this function you can get the ID of any existing feature, that means also for those, for which a SynView constant is not defined. Be sure to check the success of this function - if the feature is not mandatory, it may not exist.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
FeatureGroup	One of the LvFtrGroup.
pName	Name of the feature.
pItem	Feature ID is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.13 LV_EXTC LV_DLLIMPORT LvStatus LvGetFloat (LvHModule *hModule*, LvFeature *Feature*, double * *pValue*)

Gets a float value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The float value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.14 LV_EXTC LV_DLLIMPORT LvStatus LvGetFloatRange (LvHModule *hModule*, LvFeature *Feature*, double * *pMinValue*, double * *pMaxValue*, double * *pIncrement*)

Returns a range of a float feature.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. If the increment is not defined, 0 is returned. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.15 LV_EXTC LV_DLLIMPORT LvStatus LvGetInfo (LvHModule *hModule*, LvFeature *Feature*, LvEnum *FtrInfo*, int32_t * *pInfo*, int32_t *Param*)

Gets an info in form of a 32-bit integer value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
FtrInfo	One of the LvFtrInfo.
pInfo	The value is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.16 LV_EXTC LV_DLLIMPORT LvStatus LvGetInfoStr (LvHModule *hModule*, LvFeature *Feature*, LvEnum *FtrInfo*, char * *pInfoStr*, size_t *Size*, int32_t *Param*)

Gets an info in form of a string value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
FtrInfo	One of the LvFtrInfo.
pInfoStr	The string value is returned in this parameter.
Size	Size of the buffer (to which pInfoStr points).
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.17 LV_EXTC LV_DLLIMPORT LvStatus LvGetInfoStrSize (LvHModule *hModule*, LvFeature *Feature*, LvEnum *FtrInfo*, size_t * *pSize*, int32_t *Param*)

Gets a buffer size needed for an info in form of a string value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
FtrInfo	One of the LvFtrInfo.
pSize	Size of the buffer is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.18 LV_EXTC LV_DLLIMPORT LvStatus LvGetInt (LvHModule *hModule*, LvFeature *Feature*, int64_t * *pValue*)

Gets a 64-bit integer value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The integer value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvGetInt64() function.

5.10.2.19 LV_EXTC LV_DLLIMPORT LvStatus LvGetInt32 (LvHModule *hModule*, LvFeature *Feature*, int32_t * *pValue*)

Gets a 32-bit integer value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The integer value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.10.2.20 LV_EXTC LV_DLLIMPORT LvStatus LvGetInt32Range (LvHModule *hModule*, LvFeature *Feature*, int32_t * *pMinValue*, int32_t * *pMaxValue*, int32_t * *pIncrement*)

Returns a range and increment of an 32-bit integer feature.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.10.2.21 LV_EXTC LV_DLLIMPORT LvStatus LvGetInt64 (LvHModule *hModule*, LvFeature *Feature*, int64_t * *pValue*)

Gets a 64-bit integer value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The integer value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvGetInt() function.

5.10.2.22 LV_EXTC LV_DLLIMPORT LvStatus LvGetInt64Range (LvHModule *hModule*, LvFeature *Feature*, int64_t * *pMinValue*, int64_t * *pMaxValue*, int64_t * *pIncrement*)

Returns a range and increment of an 64-bit integer feature.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvGetIntRange() function.

5.10.2.23 LV_EXTC LV_DLLIMPORT **LvStatus** LvGetIntRange (**LvHModule** *hModule,* **LvFeature** *Feature,* int64_t * *pMinValue,* int64_t * *pMaxValue,* int64_t * *pIncrement*)

Returns a range and increment of an 64-bit integer feature.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvGetInt64Range() function.

5.10.2.24 LV_EXTC LV_DLLIMPORT LvStatus LvGetNumFeatures (LvHModule *hModule*, LvEnum *FeatureGroup*, uint32_t * *pNumItems*)

Returns a number of features for specified group. This is useful for building a list of all available features (like the tree in lv.explorer).

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
FeatureGroup	One of the LvFtrGroup.
pNumItems	The number of features is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.25 LV_EXTC LV_DLLIMPORT LvStatus LvGetPtr (LvHModule *hModule*, LvFeature *Feature*, void ** *ppValue*)

Gets a pointer.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
ppValue	The pointer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.26 LV_EXTC LV_DLLIMPORT LvStatus LvGetString (LvHModule *hModule*, LvFeature *Feature*, char * *pValue*, size_t *Size*)

Gets a string value. If you need first to get the string size, use the LvGetStringSize() function.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	Pointer to a null-terminated string buffer.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.27 LV_EXTC LV_DLLIMPORT LvStatus LvGetStringSize (LvHModule *hModule*, LvFeature *Feature*, size_t * *pSize*)

Gets a buffer size needed for a string.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pSize	Size of the buffer (including space for terminating zero) is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.28 LV_EXTC LV_DLLIMPORT LvStatus LvGetType (LvHModule *hModule*, LvFeature *Feature*, LvEnum * *pFtrType*, LvEnum * *pFtrGui*, LvEnum * *pFtrGroup*)

Returns the feature type, GUI representation and group.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pFtrType	The feature type is returned in this parameter. The returned value is one of the LvFtrType. Can be NULL.
pFtrGui	The feature GUI representation is returned in this parameter. The returned value is one of the LvFtrGui. Can be NULL.
pFtrGroup	The feature group, to which the feature belongs. The returned value is one of the LvFtrGroup. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.29 LV_EXTC LV_DLLIMPORT LvStatus LvGetVisibility (LvHModule *hModule*, LvFeature *Feature*, LvEnum * *pFtrVisibility*)

Gets the feature visibility (beginner-expert-guru).

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pFtrVisibility	The visibility is returned in this parameter. One of the LvFtrVisibility.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.30 LV_EXTC LV_DLLIMPORT uint32_t LvIsAvailable (LvHModule *hModule*, LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the LvGetAccess() function.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

Returns

If the feature is available, returns 1, otherwise 0.

5.10.2.31 LV_EXTC LV_DLLIMPORT uint32_t LvIsAvailableByName (LvHModule *hModule*, LvEnum *FeatureGroup*, const char * *pName*)

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the LvGetAccess() and LvGetFeatureByName() functions.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
FeatureGroup	One of the LvFtrGroup.
pName	Name of the feature.

Returns

If the feature is available, returns 1, otherwise 0.

5.10.2.32 LV_EXTC LV_DLLIMPORT uint32_t LvIsAvailableEnumEntry (LvHModule *hModule*, LvFeature *Feature*, LvEnum *EnumEntry*)

A helper function, allowing simply to determine, if an enum entry of an enum feature is available.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

EnumEntry	The SynView constant for the enum entry.

Returns

If the enum entry is available, returns 1, otherwise 0.

5.10.2.33 LV_EXTC LV_DLLIMPORT uint32_t LvIsImplemented (LvHModule *hModule*, LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the LvGet↩Access() function.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

Returns

If the feature is implemented, returns 1, otherwise 0.

5.10.2.34 LV_EXTC LV_DLLIMPORT uint32_t LvIsImplementedByName (LvHModule *hModule*, LvEnum *FeatureGroup*, const char * *pName*)

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the LvGet↩Access() and LvGetFeatureByName() functions.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
FeatureGroup	One of the LvFtrGroup.
pName	Name of the feature.

Returns

If the feature is implemented, returns 1, otherwise 0.

5.10.2.35 LV_EXTC LV_DLLIMPORT uint32_t LvIsImplementedEnumEntry (LvHModule *hModule*, LvFeature *Feature*, LvEnum *EnumEntry*)

A helper function, allowing simply to determine, if an enum entry of an enum feature is implemented.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
EnumEntry	The SynView constant for the enum entry.

Returns

If the enum entry is implemented, returns 1, otherwise 0.

5.10.2.36 LV_EXTC LV_DLLIMPORT uint32_t LvIsReadable (LvHModule *hModule*, LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is readable. It is a wrapper around the LvGetAccess() function.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

Returns

If the feature is readable, returns 1, otherwise 0.

5.10.2.37 LV_EXTC LV_DLLIMPORT uint32_t LvIsWritable (LvHModule *hModule,* LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is writable. It is a wrapper around the LvGetAccess() function.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

Returns

If the feature is writable, returns 1, otherwise 0.

5.10.2.38 LV_EXTC LV_DLLIMPORT LvStatus LvPoll (LvHModule *hModule*)

Polls all the non-cached features of the module. If the feature polling interval expires, the value is read and the feature callback is called.

Parameters

hModule	A handle to the System, Interface, Device, Stream or Renderer module.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.39 LV_EXTC LV_DLLIMPORT LvStatus LvRegisterFeatureCallback (LvHModule *hModule,* LvFeature *Feature,* LvFeatureCallbackFunct *pFunction,* void * *pUserParam,* void * *pFeatureParam*)

Registers or unregisters a callback function for the feature. This callback is produced by GenApi when a feature changes its value or status. The application should process this callback fast. Note that the callback can be called also from another thread - see LvEventType_FeatureDevEvent. Important note: The feature callback function should never set any other feature. Doing so can lead to recursions, which would be probably hard to diagnose and could cause unexpected behavior.

Parameters

hModule	A handle to the System, Interface, Device, Stream or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

pFunction	The callback function in the form of LvFeatureCallbackFunct. If you want to unregister the function, use NULL at this parameter.
pUserParam	User parameter, which will be passed to each callback call.
pFeatureParam	Second user parameter, which will be passed to each callback call.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.40 LV_EXTC LV_DLLIMPORT LvStatus LvSetBool (LvHModule *hModule*, LvFeature *Feature*, int32_t *Value*)

Sets a Boolean value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Value	Value to be set (in form of 32_bit integer value).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.41 LV_EXTC LV_DLLIMPORT LvStatus LvSetBuffer (LvHModule *hModule*, LvFeature *Feature*, void * *pBuffer*, size_t *Size*)

Sets a block of data.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pBuffer	Pointer to the data.
Size	Size of the data.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.42 LV_EXTC LV_DLLIMPORT LvStatus LvSetEnum (LvHModule *hModule*, LvFeature *Feature*, LvEnum *Value*)

Sets the enumeration entry by the SynView constant. If the SynView constant is not defined for the feature, then use LvSetEnumStr() to set the enum entry by a string.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.

Value	SynView constant for the requested enumeration entry.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.43 LV_EXTC LV_DLLIMPORT LvStatus LvSetEnumStr (LvHModule *hModule*, LvFeature *Feature*, const char * *pSymbolicName*)

Sets enumeration entry by its string symbolic name.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pSymbolicName	A pointer to a string with the symbolic name of the enumeration entry.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.44 LV_EXTC LV_DLLIMPORT LvStatus LvSetFloat (LvHModule *hModule*, LvFeature *Feature*, double *Value*)

Sets a float value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Value	The value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.45 LV_EXTC LV_DLLIMPORT LvStatus LvSetInt (LvHModule *hModule*, LvFeature *Feature*, int64_t *Value*)

Sets a 64-bit integer value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvSetInt64() function.

5.10.2.46 LV_EXTC LV_DLLIMPORT LvStatus LvSetInt32 (LvHModule *hModule,* LvFeature *Feature,* int32_t *Value*)

Sets a 32-bit value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.10.2.47 LV_EXTC LV_DLLIMPORT LvStatus LvSetInt64 (LvHModule *hModule*, LvFeature *Feature*, int64_t *Value*)

Sets a 64-bit integer value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvSetInt() function.

5.10.2.48 LV_EXTC LV_DLLIMPORT LvStatus LvSetPtr (LvHModule *hModule*, LvFeature *Feature*, void * *pValue*)

Sets a pointer.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The pointer to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.49 LV_EXTC LV_DLLIMPORT LvStatus LvSetString (LvHModule *hModule*, LvFeature *Feature*, const char * *pValue*)

Sets a string value.

Parameters

hModule	A handle to the System, Interface, Device, Stream, Event, Buffer or Renderer module.
Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvGetFeatureByName() function.
pValue	The string value (null-terminated).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.50 LV_EXTC LV_DLLIMPORT LvStatus LvStartPollingThread (LvHModule *hModule,* uint32_t *PollingTime,* int32_t *PollChildren*)

Starts a thread, which in a loop polls the non-cached features. If the feature polling interval expires, the value is read and the feature callback is called.

Parameters

hModule	A handle to the System, Interface, Device, Stream or Renderer module.
PollingTime	A time in milliseconds between 2 calls to poll the features.
PollChildren	If set to true, also the features in all children modules are polled. For example, if your application uses only one System module, then it is a parent of all other modules, so the polling will be propagated to all modules from a single thread. If a module has started own polling thread, then it is excluded from the propagating.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.10.2.51 LV_EXTC LV_DLLIMPORT LvStatus LvStopPollingThread (LvHModule *hModule*)

Stops the polling thread. See LvStartPollingThread() for details.

Parameters

hModule	A handle to the System, Interface, Device, Stream or Renderer module.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.11 SynView Firmware update functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvFwGetFilePattern (LvHModule hModule, uint32_t Which, char *pFilePattern, size_t Size)
- LV_EXTC LV_DLLIMPORT LvStatus LvFwLoad (LvHModule hModule, uint32_t Which, const char *pFilePath)
- LV_EXTC LV_DLLIMPORT LvStatus LvFwGetLoadStatus (LvHModule hModule, uint32_t Which, uint32_t *pCurrentByteCount, uint32_t *pIsLoading)

5.11.1 Detailed Description

5.11.2 Function Documentation

5.11.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvFwGetFilePattern (LvHModule *hModule*, uint32_t *Which*, char * *pFilePattern*, size_t *Size*)

Returns the file name mask (with wildcard characters), for searching the file with the appropriate firmware update. The files with the FW update have in their names coded the hardware IDs, so using this mask (for example in a filter in a file open dialog box) assures the file appropriate for this device is used.

Parameters

hModule	A handle to the Device module.
Which	An ID specific for a hardware. Discussed in the SynView User's Guide.
pFilePattern	In this parameter the file pattern is returned.
Size	Size of the buffer (to which the pFilePattern points).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.11.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvFwGetLoadStatus (LvHModule *hModule*, uint32_t *Which*, uint32_t * *pCurrentByteCount*, uint32_t * *pIsLoading*)

Returns the byte count and whether the loading is still in progress.

Parameters

hModule	A handle to the Device module.
Which	An ID specific for a hardware. Discussed in the SynView User's Guide.
pCurrentByteCount	Returns number of bytes transferred so far.
pIsLoading	Returns 1 if the loading is still in progress.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.11.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvFwLoad (LvHModule *hModule*, uint32_t *Which*, const char * *pFilePath*)

Loads the firmware from a file to the hardware. It can be very long process (taking minutes) and this functions blocks the thread druting this process. It is recommended to check the load status from another thread using the LvFwGetLoadStatus() function.

Parameters

hModule	A handle to the Device module.
Which	An ID specific for a hardware. Discussed in the SynView User's Guide.
pFilePath	File specification, with full path.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.12 SynView C++ API functions

Modules

- SynView LvLibrary methods
- SynView LvSystem methods
- SynView LvInterface methods
- SynView LvDevice methods
- SynView LvStream methods
- SynView LvBuffer methods
- SynView LvEvent methods
- SynView LvRenderer methods
- SynView LvModule methods

5.12.1 Detailed Description

5.13 SynView LvLibrary methods

Functions

- static uint32_t LvLibrary::GetVersion ()
- static LvStatus LvLibrary::OpenLibrary ()
- static LvStatus LvLibrary::CloseLibrary ()
- static void LvLibrary::GetErrorMessage (LvStatus Error, char *pMessage, size_t Size)
- static std::string LvLibrary::GetErrorMessage (LvStatus Error)
- static void LvLibrary::GetLastErrorMessage (char *pMessage, size_t Size)
- static std::string LvLibrary::GetLastErrorMessage ()
- static void LvLibrary::Log (const char *pLogMessage)
- static LvStatus LvLibrary::GetLibInfo (LvEnum Info, int32_t *pInfo, int32_t Param=0)
- static LvStatus LvLibrary::GetLibInfoStr (LvEnum Info, char *pInfoStr, size_t Size, int32_t Param=0)
- static LvStatus LvLibrary::GetLibInfoStrSize (LvEnum Info, size_t *pSize, int32_t Param=0)
- static LvStatus LvLibrary::GetLibInfoStr (LvEnum Info, std::string &sInfo, int32_t Param=0)
- static LvStatus LvLibrary::UpdateSystemList ()
- static LvStatus LvLibrary::GetNumberOfSystems (uint32_t *pNumberOfSystems)
- static LvStatus LvLibrary::GetSystemId (uint32_t Index, char *pSystemId, size_t Size)
- static LvStatus LvLibrary::GetSystemIdSize (uint32_t Index, size_t *pSize)
- static LvStatus LvLibrary::GetSystemId (uint32_t Index, std::string &sSystemId)
- static void LvLibrary::SetThrowErrorEnable (bool bEnable)

5.13.1 Detailed Description

5.13.2 Function Documentation

5.13.2.1 static LvStatus LvLibrary::CloseLibrary ( ) [static]

Closes the SynView library. This must be performed before you exit your application. Be sure to close first all dependent modules (System). If you are using SynView in a Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is still fully functional, which is not the case of PROCESS_DETACH in the DllMain(). If you have called LvLibrary::OpenLibrary() multiple times, you must balance it by the same number of calls of this function. Only the last call actually does the uninitialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.2 static void LvLibrary::GetErrorMessage (LvStatus *Error*, char * *pMessage*, size_t *Size*) [static]

Returns a short description of the error. Note that only some of the errors are suitable for direct display to the user, many error values indicate states which are understandable to the programmer, but may not be understandable to the end user.

Parameters

Error	The error code (the return value of most SynView functions).
pMessage	Pointer to the text buffer.

Size	Size of the buffer.

See also

LvStatus definitions.

5.13.2.3 static std::string LvLibrary::GetErrorMessage (LvStatus *Error*) [static]

Returns a short description of the error. Note that only some of the errors are suitable for direct display to the user, many error values indicate states which are understandable to the programmer, but may not be understandable to the end user.

Parameters

Error	The error code (the return value of most SynView functions).

Returns

Error message in std::string.

See also

LvStatus definitions.

5.13.2.4 static void LvLibrary::GetLastErrorMessage (char * *pMessage*, size_t *Size*) [static]

Returns more detailed description of the last error, which happened in the thread from which this function was called. As the info is recorded inside SynView for each error, the desription provides more detailed info, including the name of the function, in which the error happened, and possibly more diagnostic info. The difference to LvLibrary::GetErrorMessage() is that LvLibrary::GetErrorMessage() returns a static string from a numbered table of errors while this function returns additionally info recorded at the time the error happened. If a function returns LVSTATUS_OK, it does not reset this error message (for speed reasons) so the correct approach is to get the error number as the function return value and if this return value is not LVSTATUS_OK, then you can get more info about the error using this function. be sure to call it from the same thread.

Parameters

pMessage	Pointer to the text buffer.
Size	Size of the buffer.

See also

LvStatus definitions.

5.13.2.5 static std::string LvLibrary::GetLastErrorMessage ( ) [static]

Returns more detailed description of the last error, which happened in the thread from which this function was called. As the info is recorded inside SynView for each error, the desription provides more detailed info, including the name of the function, in which the error happened, and possibly more diagnostic info. The difference to LvLibrary::GetErrorMessage() is that LvLibrary::GetErrorMessage() returns a static string from a numbered table of errors while this function returns additionally info recorded at the time the error happened. If a function returns LVSTATUS_OK, it does not reset this error message (for speed reasons) so the correct approach is to get the error number as the function return value and if this return value is not LVSTATUS_OK, then you can get more info about the error using this function. be sure to call it from the same thread.

Returns

Error message in std::string.

See also

LvStatus definitions.

5.13.2.6 static LvStatus LvLibrary::GetLibInfo (LvEnum *Info*, int32_t * *pInfo*, int32_t *Param* = 0) [static]

Gets a general info in form of a 32-bit integer value.

Parameters

Info	One of the LvLibInfo values.
pInfo	The value is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.7 static LvStatus LvLibrary::GetLibInfoStr (LvEnum *Info*, char * *pInfoStr*, size_t *Size*, int32_t *Param* = 0) [static]

Gets a general info in form of a string value.

Parameters

Info	One of the LvLibInfo values.
pInfoStr	The string value is returned in this parameter.
Size	Size of the buffer (to which pInfoStr points).
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.8 static LvStatus LvLibrary::GetLibInfoStr (LvEnum *Info*, std::string & *sInfo*, int32_t *Param* = 0) [static]

Gets a general info in form of a std::string value.

Parameters

Info	One of the LvLibInfo values.
sInfo	The string value is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.9 static LvStatus LvLibrary::GetLibInfoStrSize (LvEnum *Info*, size_t * *pSize*, int32_t *Param* = 0) [static]

Gets a buffer size needed for a general info in form of a string value.

Parameters

Info	One of the LvLibInfo values.
pSize	Size of the buffer is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.10 static LvStatus LvLibrary::GetNumberOfSystems (uint32_t * *pNumberOfSystems*) [static]

Returns the number of systems found after the LvLibrary::UpdateSystemList() call. Typical use of this function is in iterating systems using the LvLibrary::GetSystemId() function.

Parameters

pNumberOf↩ Systems	The number of systems found.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.11 static LvStatus LvLibrary::GetSystemId (uint32_t *Index*, char * *pSystemId*, size_t *Size*) [static]

Returns the string ID of the system at given index. This ID is used in the LvSystem::Open() function for opening the system.

Parameters

Index	Zero-based index of the system, a value >= 0 and < number of systems, returned by the LvLibrary::GetNumberOfSystems() function.
pSystemId	Pointer to a string buffer, where the system ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.12 static LvStatus LvLibrary::GetSystemId (uint32_t *Index*, std::string & *sSystemId*) [static]

Returns the string ID of the system at given index. This ID is used in the LvSystem::Open() function for opening the system.

Parameters

Index	Zero-based index of the system, a value >= 0 and < number of systems, returned by the LvLibrary::GetNumberOfSystems() function.
sSystemId	String, where the system ID will be placed.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.13 static LvStatus LvLibrary::GetSystemIdSize (uint32_t *Index*, size_t * *pSize*) [static]

Returns the size of the string buffer needed to hold the system ID string, including the terminating zero character.

Parameters

Index	Zero-based index of the system, a value >= 0 and < number of systems, returned by the LvLibrary::GetNumberOfSystems() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.14 static uint32_t LvLibrary::GetVersion ( ) [static]

Returns SynView version.

Returns

The returned doubleword contains the build version in the low word and the high word is the major version in the upper byte and subversion in the lower byte. For example:

```
uint32_t Version = LvLibrary::GetVersion();
printf("SynView %d.%02d.%03d",
        ((Version >> 24) & 0xFF),
        ((Version >> 16) & 0xFF),
        (Version & 0xFFFF));
```

5.13.2.15 static void LvLibrary::Log (const char * *pLogMessage*) [static]

Adds a line to the sv.synview.log. The SynView log is a tool for New Electronic Technology technical support, but in some cases may be useful to put to the log additional info from your code.

Parameters

pLogMessage	Pointer to the null teminated string with the message.

5.13.2.16 static LvStatus LvLibrary::OpenLibrary ( ) [static]

Opens the SynView library. This must be done before you can use any other SynView function (with the exception of LvLibrary::GetVersion() and LvLibrary::GetErrorMessage()). If you are using SynView in Windows DLL, avoid calling this in Windows DllMain() function - for proper functionality this function must be called when the application or DLL is already fully initialized and there are no restrictions about synchronization (DllMain has such restrictions). If you call this function multiple times, you must balance it by the same number of the LvLibrary::CloseLibrary() calls. Only the first call will actually do the initialization. IMPORTANT: The library must not be opened again once it was already uninitialized.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.13.2.17 static void LvLibrary::SetThrowErrorEnable (bool *bEnable*) [static]

Enables/disables the conversion of LvStatus return values (not equal to LVSTATUS_OK) to C++ exceptions of the LvException type.

Parameters

bEnable	Enable/disable the exception throwing.

5.13.2.18 static LvStatus LvLibrary::UpdateSystemList ( ) [static]

Updates the list of systems available. This function must be called before iterating through the systems by the LvLibrary::GetNumberOfSystems() and LvLibrary::GetSystemId() functions. The systems are physically represented by GenTL libraries available in the operating systems, this call searches for them in standard locations. See also the description of the sv.synview.ini file in the SynView User's Guide. Note that this function is seldom needed, most applications will work with the default system (see LvSystem::Open() for details).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14 SynView LvSystem methods

Functions

- static LvStatus LvSystem::Open (const char *pSystemId, LvSystem *&pSystem)
- static LvStatus LvSystem::Close (LvSystem *&pSystem)
- LvStatus LvSystem::UpdateInterfaceList (uint32_t Timeout=0xFFFFFFFF)
- LvStatus LvSystem::GetNumberOfInterfaces (uint32_t *pNumberOfInterfaces)
- LvStatus LvSystem::GetInterfaceId (uint32_t Index, char *pInterfaceId, size_t Size)
- LvStatus LvSystem::GetInterfaceIdSize (uint32_t Index, size_t *pSize)
- LvStatus LvSystem::GetInterfaceId (uint32_t Index, std::string &sInterfaceId)
- LvStatus LvSystem::FindInterface (LvFindBy FindBy, const char *pFindStr, char *pInterfaceId, size_t Size)
- LvStatus LvSystem::FindInterface (LvFindBy FindBy, const char *pFindStr, std::string &sInterfaceId)
- LvHSystem LvSystem::GetHandle ()
- LvStatus LvSystem::OpenInterface (const char *pInterfaceId, LvInterface *&pInterface)
- LvStatus LvSystem::CloseInterface (LvInterface *&pInterface)
- LvStatus LvSystem::OpenEvent (LvEventType EventType, LvEvent *&pEvent)
- LvStatus LvSystem::CloseEvent (LvEvent *&pEvent)

5.14.1 Detailed Description

5.14.2 Function Documentation

5.14.2.1 static LvStatus LvSystem::Close (LvSystem *& *pSystem*) [static]

Deletes the LvSystem class instance. Actually it means freeing the coresponding GenTL library. Be sure you first close all dependent modules (LvInterface, LvEvent etc.). If the System was opened multiple times, it only decreases the reference counter (see the note by the LvSystem::Open()).

Parameters

pSystem	Pointer to LvSystem instance, obtained from the LvSystem::Open() function. The pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14.2.2 LvStatus LvSystem::CloseEvent (LvEvent *& *pEvent*)

Deletes the LvEvent class instance. This method is provided just for convenience, it has the same functionality as the LvEvent::Close() static method.

Parameters

pEvent	Pointer the Event class instance, is assigned NULL after the closing is done.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvEvent::Close().

5.14.2.3 LvStatus LvSystem::CloseInterface (LvInterface *& *pInterface*)

Deletes the LvInterface class instance. This method is provided just for convenience, it has the same functionality as the LvInterface::Close() static method. If the Interface was opened multiple times, it only decreases the reference counter (see a note by the LvInterface::Open()). Be sure you first close all dependent modules (LvDevice, LvEvent etc.).

Parameters

pInterface	Pointer to the LvInterface instance, obtained from the LvInterface::Open() function. The pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvInterface::Close().

5.14.2.4 LvStatus LvSystem::FindInterface (LvFindBy *FindBy*, const char * *pFindStr*, char * *pInterfaceId*, size_t *Size*)

Finds the interface according specified criteria and returns a string ID of the interface, which is used by the LvInterface::Open() function. This function does not update the interface list - if you need to do so, call the LvSystem::UpdateInterfaceList() function before calling this function.

Parameters

FindBy	Specifies by which criteria to find the interface. Use one of the LvFindBy constants.
pFindStr	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the LvFindBy_IPAddress, this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
pInterfaceId	Pointer to a string buffer, where the interface ID will be placed.
Size	Size of the string buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions. If the Interface is found, the returned status is LVSTATUS_OK.

5.14.2.5 LvStatus LvSystem::FindInterface (LvFindBy *FindBy*, const char * *pFindStr*, std::string & *sInterfaceId*)

Finds the interface according specified criteria and returns a string ID of the interface, which is used by the LvInterface::Open() function. This function does not update the interface list - if you need to do so, call the LvSystem::UpdateInterfaceList() function before calling this function.

Parameters

FindBy	Specifies by which criteria to find the interface. Use one of the LvFindBy constants.
pFindStr	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the LvFindBy_IPAddress, this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).

sInterfaceId	String to which the interface ID will be placed.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions. If the Interface is found, the returned status is LVSTATUS_OK.

5.14.2.6 LvHSystem LvSystem::GetHandle ( )

Returns a handle of the System (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.14.2.7 LvStatus LvSystem::GetInterfaceId (uint32_t *Index*, char * *pInterfaceId*, size_t *Size*)

Returns a string ID of the interface, which is used by the LvInterface::Open() function.

Parameters

Index	Zero-based index of the interface, a value >= 0 and < number of interfaces, returned by the LvSystem::GetNumberOfInterfaces() function.
pInterfaceId	Pointer to a string buffer, where the interface ID will be placed.
Size	Size of the string buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14.2.8 LvStatus LvSystem::GetInterfaceId (uint32_t *Index*, std::string & *sInterfaceId*)

Returns a string ID of the interface, which is used by the LvInterface::Open() function.

Parameters

Index	Zero-based index of the interface, a value >= 0 and < number of interfaces, returned by the LvSystem::GetNumberOfInterfaces() function.
sInterfaceId	String, where the interface ID will be placed.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14.2.9 LvStatus LvSystem::GetInterfaceIdSize (uint32_t *Index*, size_t * *pSize*)

Returns the size of the string buffer needed to hold the Interface ID string, including the terminating zero character.

Parameters

Index	Zero-based index of the interface, a value >= 0 and < number of interfaces, returned by the LvSystem::GetNumberOfInterfaces() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14.2.10 LvStatus LvSystem::GetNumberOfInterfaces (uint32_t * *pNumberOfInterfaces*)

Returns the number of found interfaces, after the LvSystem::UpdateInterfaceList() call.

Parameters

pNumberOfInterfaces	Number of interfaces found.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14.2.11 static LvStatus LvSystem::Open (const char * *pSystemId,* LvSystem *& *pSystem*) [static]

Creates the LvSystem class instance. Opening the system actually means loading the corresponding GenTL library. Note that before you can open the System, the LvOpenLibrary() must be called. The same system can be open multiple times (there is a reference counter inside); in such case there must be also the same number of LvSystem::Close() calls used (every open increase the reference count and every close decreases it).

Parameters

pSystemId	A string ID of the system. This can be either an empty string - then the default system is opened, or it can be a string obtained from the LvGetSystemId() function.
pSystem	Pointer to the opened LvSystem instance is returned here in case the opening succeeds, NULL if fails

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.14.2.12 LvStatus LvSystem::OpenEvent (LvEventType *EventType,* LvEvent *& *pEvent*)

Creates the LvEvent class instance, owned by the System. This method is provided just for convenience, it has the same functionality as the LvEvent::Open() static method.

Parameters

EventType	One of the LvEventType.
pEvent	To this parameter the Event class instance is stored.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvEvent::Open().

5.14.2.13 LvStatus LvSystem::OpenInterface (const char * *pInterfaceId,* LvInterface *& *pInterface*)

Creates the LvInterface class instance. This method is provided just for convenience, it has the same functionality as the LvInterface::Open() static method. The same Interface can be open multiple times (there is a reference counter inside); in such case there must be also the same number of LvInterface::Close() or LvSystem::InterfaceClose() calls used (every open increase the reference count and every close decreases it).

Parameters

pInterfaceId	A string interface ID, obtained by the LvSystem::GetInterfaceId().
pInterface	In this parameter the pointer to the LvInterface instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvInterface::Open().

5.14.2.14 LvStatus LvSystem::UpdateInterfaceList (uint32_t *Timeout* = `0xFFFFFFFF`)

Updates the internal list of available interfaces. You can then iterate through them by LvSystem::GetNumberOf↩ Interfaces() and LvSystem::GetInterfaceId().

Parameters

Timeout	Specifies a timeout in ms for searching the interfaces. This applies only to special cases of interfaces, where some delay can happen; common interfaces are detected without any significant delays.

Returns

If the timeout has expired while waiting for the completion, the function returns LVSTATUS_TIMEOUT, otherwise it returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.15 SynView LvInterface methods

Functions

- static LvStatus LvInterface::Open (LvSystem *pSystem, const char *pInterfaceId, LvInterface *&pInterface)
- static LvStatus LvInterface::Close (LvInterface *&pInterface)
- LvStatus LvInterface::UpdateDeviceList (uint32_t Timeout=0xFFFFFFFF)
- LvStatus LvInterface::GetNumberOfDevices (uint32_t *pDevices)
- LvStatus LvInterface::GetDeviceId (uint32_t Index, char *pDeviceId, size_t Size)
- LvStatus LvInterface::GetDeviceIdSize (uint32_t Index, size_t *pSize)
- LvStatus LvInterface::GetDeviceId (uint32_t Index, std::string &sDeviceId)
- LvStatus LvInterface::FindDevice (LvFindBy FindBy, const char *pFindStr, char *pDeviceId, size_t Size)
- LvStatus LvInterface::FindDevice (LvFindBy FindBy, const char *pFindStr, std::string &sDeviceId)
- LvHInterface LvInterface::GetHandle ()
- LvStatus LvInterface::OpenDevice (const char *pDeviceId, LvDevice *&pDevice, LvDeviceAccess Access=LvDeviceAccess_Exclusive)
- LvStatus LvInterface::CloseDevice (LvDevice *&pDevice)

5.15.1 Detailed Description

5.15.2 Function Documentation

5.15.2.1 static LvStatus LvInterface::Close (LvInterface *& *pInterface*) [static]

Deletes the LvInterface class instance. If the Interface was opened multiple times, it only decreases the reference counter (see a note by the LvInterface::Open()). Be sure you first close all dependent modules (LvDevice, LvEvent etc.).

Parameters

pInterface	Pointer to the LvInterface instance, obtained from the LvInterface::Open() function. The pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvSystem::CloseInterface().

5.15.2.2 LvStatus LvInterface::CloseDevice (LvDevice *& *pDevice*)

Deletes the LvDevice class instance. This method is provided just for convenience, it has the same functionality as the LvDevice::Close() static method. Be sure you first close all dependent modules (LvStream, LvEvent etc.).

Parameters

pDevice	Pointer to the LvDevice instance, obtained from the LvDevice::Open() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvDevice::Close().

5.15.2.3 LvStatus LvInterface::FindDevice (LvFindBy *FindBy*, const char * *pFindStr*, char * *pDeviceId*, size_t *Size*)

Finds the device according specified criteria and returns a string ID of the device, which can be used by the LvDevice::Open() function. This function does not update the device list - if you need to do so, call the LvInterface::UpdateDeviceList() function before calling this function.

Parameters

FindBy	Specifies by which criteria to find the interface. Use one of the LvFindBy constants.
pFindStr	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the LvFindBy_IPAddress, this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
pDeviceId	Pointer to a string buffer, where the device ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions. If the Device is found, the returned status is LVSTATUS_OK.

5.15.2.4 LvStatus LvInterface::FindDevice (LvFindBy *FindBy*, const char * *pFindStr*, std::string & *sDeviceId*)

Finds the device according specified criteria and returns a string ID of the device, which can be used by the LvDevice::Open() function. This function does not update the device list - if you need to do so, call the LvInterface::UpdateDeviceList() function before calling this function.

Parameters

FindBy	Specifies by which criteria to find the interface. Use one of the LvFindBy constants.
pFindStr	Specifies the find string, the meaning of which is determined by the FindBy parameter, for example when using the LvFindBy_IPAddress, this string should contain the IP address searched for. The searched string is not case sensitive and need not be complete (is searched as a substring).
sDeviceId	To this string the found device ID is placed.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions. If the Device is found, the returned status is LVSTATUS_OK.

5.15.2.5 LvStatus LvInterface::GetDeviceId (uint32_t *Index*, char * *pDeviceId*, size_t *Size*)

Returns a string ID of the device at specified position in the list. Note that this device ID is stable (the same physical device has always the same ID) and it is unique (no other physical device can have the same ID). To hardcode directly the device ID in your application is not recommended, as the application would not be usable, when a deffective device needs to be replaced. The SynView User's Guide discuss the ways, how to solve such maintainability demands.

Parameters

Index	Zero-based index of the device, a value >= 0 and < number of devices, returned by the LvInterface::GetNumberOfDevices() function.

pDeviceId	Pointer to a string buffer, where the device ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.15.2.6 LvStatus LvInterface::GetDeviceId (uint32_t *Index*, std::string & *sDeviceId*)

Returns a string ID of the device at specified position in the list. Note that this device ID is stable (the same physical device has always the same ID) and it is unique (no other physical device can have the same ID). To hardcode directly the device ID in your application is not recommended, as the application would not be usable, when a deffective device needs to be replaced. The SynView User's Guide discuss the ways, how to solve such maintainability demands.

Parameters

Index	Zero-based index of the device, a value >= 0 and < number of devices, returned by the LvInterface::GetNumberOfDevices() function.
sDeviceId	String to which the device ID will be placed.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.15.2.7 LvStatus LvInterface::GetDeviceIdSize (uint32_t *Index*, size_t * *pSize*)

Returns the size of the string buffer needed to hold the Device ID string, including the terminating zero character.

Parameters

Index	Zero-based index of the device, a value >= 0 and < number of devices, returned by the LvInterface::GetNumberOfDevices() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.15.2.8 LvHInterface LvInterface::GetHandle ( )

Returns a handle of the Interface (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.15.2.9 LvStatus LvInterface::GetNumberOfDevices (uint32_t * *pDevices*)

Returns the number of devices found by the LvInterface::UpdateDeviceList() function.

Parameters

pDevices	Number of devices found.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.15.2.10 static LvStatus LvInterface::Open (LvSystem * *pSystem*, const char * *pInterfaceId*, LvInterface *& *pInterface*) [static]

Creates the LvInterface class instance. The same Interface can be open multiple times (there is a reference counter inside); in such case there must be also the same number of LvInterface::Close() or LvSystem::InterfaceClose() calls used (every open increase the reference count and every close decreases it) .

Parameters

pSystem	A pointer to the LvSystem instance, obtained from the LvSystem::Open() function.
pInterfaceId	A string interface ID, obtained by the LvSystem::GetInterfaceId().
pInterface	In this parameter the pointer to the LvInterface instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvSystem::OpenInterface().

5.15.2.11 LvStatus LvInterface::OpenDevice (const char * *pDeviceId*, LvDevice *& *pDevice*, LvDeviceAccess *Access* = LvDeviceAccess_Exclusive)

Creates the LvDevice class instance. This method is provided just for convenience, it has the same functionality as the LvDevice::Open() static method. This physically means opening a connection with the device and retrieving a list of device remote features. Always check the success of this function call; the opening may fail for example when you request an exclusive access and the device is already open by some other application.

Parameters

pDeviceId	A string ID of the device, obtained by LvInterface::GetDeviceId() function.
pDevice	In this parameter the pointer to the LvDevice instance is returned.
Access	Desired device access, one of the LvDeviceAccess constants.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvDevice::Open().

5.15.2.12 LvStatus LvInterface::UpdateDeviceList (uint32_t *Timeout* = 0xFFFFFFFF)

Updates the Device list. The available devices are searched.

Parameters

Timeout	Specifies a timeout in ms for searching the devices.

Returns

If the timeout has expired while waiting for the completion, the function returns LVSTATUS_TIMEOUT, otherwise it returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16 SynView LvDevice methods

Modules

- SynView LvDevice firmware update methods

Functions

- static LvStatus LvDevice::Open (LvInterface *pInterface, const char *pDeviceId, LvDevice *&pDevice, LvDeviceAccess Access=LvDeviceAccess_Exclusive)
- static LvStatus LvDevice::Close (LvDevice *&pDevice)
- LvStatus LvDevice::GetNumberOfStreams (uint32_t *pNumberOfStreams)
- LvStatus LvDevice::GetStreamId (uint32_t Index, char *pStreamId, size_t Size)
- LvStatus LvDevice::GetStreamIdSize (uint32_t Index, size_t *pSize)
- LvStatus LvDevice::GetStreamId (uint32_t Index, std::string &sStreamId)
- LvStatus LvDevice::AcquisitionStart (uint32_t Options=0)
- LvStatus LvDevice::AcquisitionStop (uint32_t Options=0)
- LvStatus LvDevice::AcquisitionAbort (uint32_t Options=0)
- LvStatus LvDevice::AcquisitionArm (uint32_t Options=0)
- LvStatus LvDevice::SaveSettings (const char *pId, const char *pFileName, uint32_t Options)
- LvStatus LvDevice::LoadSettings (const char *pId, const char *pFileName, uint32_t Options)
- LvStatus LvDevice::UniSetLut (LvLUTSelector Selector, void *pLUT, size_t Size, uint32_t Options=0)
- LvStatus LvDevice::UniGetLut (LvLUTSelector Selector, void *pLUT, size_t Size, uint32_t Options=0)
- LvStatus LvDevice::OpenStream (const char *pStreamId, LvStream *&pStream)
- LvStatus LvDevice::CloseStream (LvStream *&pStream)
- LvStatus LvDevice::OpenEvent (LvEventType EventType, LvEvent *&pEvent)
- LvStatus LvDevice::CloseEvent (LvEvent *&pEvent)
- LvHDevice LvDevice::GetHandle ()

5.16.1 Detailed Description

5.16.2 Function Documentation

5.16.2.1 LvStatus LvDevice::AcquisitionAbort (uint32_t *Options* = 0)

Aborts athe acquisition immediately, without completing the current Frame or waiting on a trigger.

Parameters

Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.2 LvStatus LvDevice::AcquisitionArm (uint32_t *Options* = 0)

Prepares the device for acquisition, so that the acquisition using the LvDevice::AcquisitionStart() function then can start fast. If it is not called before LvDevice::AcquisitionStart(), it is called automatically inside the LvDevice::AcquisitionStart().

Parameters

Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.3 LvStatus LvDevice::AcquisitionStart (uint32_t *Options* = 0)

Starts the acquisition. This function includes more than just calling the remote AcquisitionStart command on the device - it checks the size of the buffers, prepares the streams for the start, locks GenTL params and then starts the acquisition on the device itself. Always check the success of this function call.

Parameters

Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.4 LvStatus LvDevice::AcquisitionStop (uint32_t *Options* = 0)

Stops the acquisition.

Parameters

Options	Reserved for future use, must be 0 or ommitted.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.5 static LvStatus LvDevice::Close (LvDevice *& *pDevice*) [static]

Deletes the LvDevice class instance. Be sure you first close all dependent modules (LvStream, LvEvent etc.).

Parameters

pDevice	Pointer to the LvDevice instance, obtained from the LvDevice::Open() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvInterface::CloseDevice().

5.16.2.6 LvStatus LvDevice::CloseEvent (LvEvent *& *pEvent*)

Deletes the LvEvent class instance. This method is provided just for convenience, it has the same functionality as the LvEvent::Close() static method.

Parameters

pEvent	A pointer to the LvEvent class instance, obtained from the LvEvent::Open() or LvDevice::↩ OpenEvent() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvEvent::Close().

5.16.2.7 LvStatus LvDevice::CloseStream (LvStream *& *pStream*)

Deletes the LvStream class instance. This method is provided just for convenience, it has the same functionality as the LvStream::Close() static method. Be sure you first close all dependent modules (LvBuffers, LvEvent, Lv↩ Renderer etc.).

Parameters

pStream	Pointer to the LvStream instance, obtained from the LvStream::Open() or LvDevice::Open↩ Stream() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::Close().

5.16.2.8 LvHDevice LvDevice::GetHandle ( )

Returns a handle of the Device (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.16.2.9 LvStatus LvDevice::GetNumberOfStreams (uint32_t * *pNumberOfStreams*)

Returns the number of available stream types for this device. You can then iterate the streams by the LvDevice::↩ GetStreamId() function.

Parameters

pNumberOf↩ Streams	The number of streams is returned here.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.10 LvStatus LvDevice::GetStreamId (uint32_t *Index*, char * *pStreamId*, size_t *Size*)

Returns a string Stream ID, needed for opening the stream.

Parameters

Index	Zero-based index of the stream type, a value >= 0 and < number of streams, returned by the LvDevice::GetNumberOfStreams() function.
pStreamId	Pointer to a string buffer, where the stream ID will be placed.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.11 LvStatus LvDevice::GetStreamId (uint32_t *Index*, std::string & *sStreamId*)

Returns a string Stream ID, needed for opening the stream.

Parameters

Index	Zero-based index of the stream type, a value >= 0 and < number of streams, returned by the LvDevice::GetNumberOfStreams() function.
sStreamId	A string to which the stream ID is placed.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.12 LvStatus LvDevice::GetStreamIdSize (uint32_t *Index*, size_t * *pSize*)

Returns the size of the string buffer needed to hold the stream ID at given index, including the space for the terminating zero character.

Parameters

Index	Zero-based index of the stream type, a value >= 0 and < number of streams, returned by the LvDevice::GetNumberOfStreams() function.
pSize	Size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.13 LvStatus LvDevice::LoadSettings (const char * *pId*, const char * *pFileName*, uint32_t *Options*)

Loads the device settings from a file. In the Options can be specified which parts of the device configuration are to be loaded.

Parameters

pId	A string ID enabling to distinguish more configurations in one file. If empty, the "Default" will be used.
pFileName	The file specification, where the configuration is stored. It is a text file.
Options	One or or-ed combination of LvSaveFlag definitions.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.14 static LvStatus LvDevice::Open (LvInterface * *pInterface*, const char * *pDeviceId*, LvDevice *& *pDevice*, LvDeviceAccess *Access* = LvDeviceAccess_Exclusive) [static]

Creates the LvDevice class instance.

This physically means opening a connection with the device and retrieving a list of device remote features. Always check the success of this function call; the opening may fail for example when you request an exclusive access and the device is already open by some other application.

Parameters

pInterface	A pointer to the LvInterface instance, obtained from the LvInterface::Open() function.
pDeviceId	A string ID of the device, obtained by LvInterface::GetDeviceId() function.
pDevice	In this parameter the pointer to the LvDevice instance is returned.
Access	Desired device access, one of the LvDeviceAccess constants.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvInterface::OpenDevice().

5.16.2.15 LvStatus LvDevice::OpenEvent (LvEventType *EventType*, LvEvent *& *pEvent*)

Creates the LvEvent class instance for specified owner module. This method is provided just for convenience, it has the same functionality as the LvEvent::Open() static method.

Parameters

EventType	One of the LvEventType.
pEvent	To this parameter the pointer LvEvent instance is stored.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvEvent::Open().

5.16.2.16 LvStatus LvDevice::OpenStream (const char * *pStreamId*, LvStream *& *pStream*)

Creates the LvStream class instance associated with the device. This method is provided just for convenience, it has the same functionality as the LvStream::Open() static method.

Parameters

pStreamId	A string ID of the stream, obtained from LvDevice::GetStreamId(). If an empty string is used, the first found stream is opened. This is usually the image data stream.
pStream	In this parameter the pointer to the LvStream instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::Open().

5.16.2.17 LvStatus LvDevice::SaveSettings (const char * *pId*, const char * *pFileName*, uint32_t *Options*)

Saves the device settings to a file. In the Options can be specified which parts of the device configuration are to be saved.

Parameters

pId	A string ID enabling to distinguish more configurations in one file. If empty, the "Default" will be used.
pFileName	The file specification, to which the configuration is stored. It is a text file.
Options	One or or-ed combination of LvSaveFlag definitions.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.18 LvStatus LvDevice::UniGetLut (LvLUTSelector *Selector*, void * *pLUT*, size_t *Size*, uint32_t *Options* = 0)

Gets the lookup table. See LvDevice::UniSetLut() for details. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format. So you can for example read the 12-bit LUT to 8-bit LUT array.

Parameters

Selector	Lookup table selector, see LvLUTSelector.
pLUT	Pointer to the lookup table.
Size	Size of the lookup table. The only valid values are • 256 for 8-bit LUT • 2048 for 10-bit LUT • 8192 for 12-bit LUT
Options	The LvUniLutFlags_HwLut option can be used to apply to function directly on HW LUT.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.16.2.19 LvStatus LvDevice::UniSetLut (LvLUTSelector *Selector*, void * *pLUT*, size_t *Size*, uint32_t *Options* = 0)

Sets the lookup table. If the hardware lookup table is available, it is used, otherwise a software lookup table is set. This function belongs to a set of functions, which unify the functionality of devices with real-time processing embedded in hardware (RTF) and devices without real-time processing, for which the processing is made by software. The LUT is automatically recalculated to appropriate type, if you use different LUT bit depth than is the actually used for the current pixel format.

Parameters

Selector	Lookup table selector, see LvLUTSelector.
pLUT	Pointer to the lookup table.
Size	Size of the lookup table. The only valid values are • 256 for 8-bit LUT • 2048 for 10-bit LUT • 8192 for 12-bit LUT
Options	The LvUniLutFlags_HwLut option can be used to apply to function directly on HW LUT.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.17 SynView LvDevice firmware update methods

Functions

- LvStatus LvDevice::FwGetFilePattern (uint32_t Which, char *pFilePattern, size_t Size)
- LvStatus LvDevice::FwLoad (uint32_t Which, const char *pFilePath)
- LvStatus LvDevice::FwGetLoadStatus (uint32_t Which, uint32_t *pCurrentByteCount, bool *pIsLoading)

5.17.1 Detailed Description

5.17.2 Function Documentation

5.17.2.1 LvStatus LvDevice::FwGetFilePattern (uint32_t *Which*, char * *pFilePattern*, size_t *Size*)

Returns the file name mask (with wildcard characters), for searching the file with the appropriate firmware update. The files with the FW update have in their names coded the hardware IDs, so using this mask (for example in a filter in a file open dialog box) assures the file appropriate for this device is used.

Parameters

Which	An ID specific for a hardware. Discussed in the SynView User's Guide.
pFilePattern	In this parameter the file pattern is returned.
Size	Size of the buffer (to which the pFilePattern points).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.17.2.2 LvStatus LvDevice::FwGetLoadStatus (uint32_t *Which*, uint32_t * *pCurrentByteCount*, bool * *pIsLoading*)

Returns the byte count and whether the loading is still in progress.

Parameters

Which	An ID specific for a hardware. Discussed in the SynView User's Guide.
pCurrentByteCount	Returns number of bytes transferred so far.
pIsLoading	Returns true if the loading is still in progress.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.17.2.3 LvStatus LvDevice::FwLoad (uint32_t *Which*, const char * *pFilePath*)

Loads the firmware from a file to the hardware. It can be very long process (taking minutes) and this functions blocks the thread druting this process. It is recommended to check the load status from another thread using the LvFwGetLoadStatus() function.

Parameters

Which	An ID specific for a hardware. Discussed in the SynView User's Guide.

pFilePath	File specification, with full path.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.18 SynView LvStream methods

Functions

- static LvStatus LvStream::Open (LvDevice *pDevice, const char *pStreamId, LvStream *&pStream)
- static LvStatus LvStream::Close (LvStream *&pStream)
- LvStatus LvStream::GetBufferAt (uint32_t BufferIndex, LvBuffer *&pBuffer)
- LvStatus LvStream::FlushQueue (LvQueueOperation Operation)
- LvStatus LvStream::Start (uint32_t StartFlags=0x00000000, uint32_t ImagesToAcquire=0xFFFFFFFF)
- LvStatus LvStream::Stop (uint32_t StopFlags=0x00000000)
- LvHStream LvStream::GetHandle ()
- LvStatus LvStream::OpenBuffer (void *pDataPointer, size_t DataSize, void *pUserPointer, uint32_t Options, LvBuffer *&pBuffer)
- LvStatus LvStream::CloseBuffer (LvBuffer *&pBuffer)
- LvStatus LvStream::OpenEvent (LvEventType EventType, LvEvent *&pEvent)
- LvStatus LvStream::CloseEvent (LvEvent *&pEvent)
- LvStatus LvStream::OpenRenderer (LvRenderer *&pRenderer)
- LvStatus LvStream::CloseRenderer (LvRenderer *&pRenderer)

5.18.1 Detailed Description

5.18.2 Function Documentation

5.18.2.1 static LvStatus LvStream::Close (LvStream *& *pStream*) [static]

Deletes the LvStream class instance. Be sure you first close all dependent modules (LvBuffers, LvEvent, LvRenderer etc.).

Parameters

pStream	Pointer to the LvStream instance, obtained from the LvStream::Open() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvDevice::CloseStream().

5.18.2.2 LvStatus LvStream::CloseBuffer (LvBuffer *& *pBuffer*)

Deletes the LvBuffer class instance. On the GenTL level it corresponds to the DSRevokeBuffer() function. This method is provided just for convenience, it has the same functionality as the LvBuffer::Close() static method.

Parameters

pBuffer	A pointer to the LvBuffer instance, obtained from the LvBuffer::Open() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvBuffer::Close().

5.18.2.3 LvStatus LvStream::CloseEvent (LvEvent *& *pEvent*)

Deletes the LvEvent class instance. This method is provided just for convenience, it has the same functionality as the LvEvent::Close() static method.

Parameters

pEvent	Pointer the Event class instance, is assigned NULL after the closing is done.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvEvent::Close().

5.18.2.4 LvStatus LvStream::CloseRenderer (LvRenderer *& *pRenderer*)

Deletes the LvRenderer class instance. This method is provided just for convenience, it has the same functionality as the LvRenderer::Close() static method.

Parameters

pRenderer	A pointer to the LvRenderer instance, obtained from the LvRenderer::Open() function. This pointer is set to NULL after close.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvRenderer::Close().

5.18.2.5 LvStatus LvStream::FlushQueue (LvQueueOperation *Operation*)

Moves the buffers according to the LvQueueOperation specified.

Parameters

Operation	One of the LvQueueOperation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.18.2.6 LvStatus LvStream::GetBufferAt (uint32_t *BufferIndex,* LvBuffer *& *pBuffer*)

Returns the buffer instance at given index.

Parameters

BufferIndex	Zero-based index.
pBuffer	In this parameter the pointer to LvBuffer instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.18.2.7 LvHStream LvStream::GetHandle ( )

Returns a handle of the Stream (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.18.2.8 static LvStatus LvStream::Open (LvDevice * *pDevice*, const char * *pStreamId*, LvStream *& *pStream*) [static]

Creates the LvStream class instance, associated with the device.

Parameters

pDevice	A pointer to the LvDevice instance, obtained from the LvDevice::Open() function.
pStreamId	A string ID of the stream, obtained from LvDevice::GetStreamId(). If an empty string is used, the first found stream is opened. This is usually the image data stream.
pStream	In this parameter the pointer to the LvStream instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvDevice::OpenStream().

5.18.2.9 LvStatus LvStream::OpenBuffer (void * *pDataPointer*, size_t *DataSize*, void * *pUserPointer*, uint32_t *Options*, LvBuffer *& *pBuffer*)

Creates the LvBuffer class instance. On the GenTL level it corresponds to DSAnnounceBuffer() or DSAllocAnd↩ AnnounceBuffer(). This method is provided just for convenience, it has the same functionality as the LvBuffer::↩ Open() static method.

Parameters

pDataPointer	Pointer to image data buffer. This can be supplied by the application (in such case the Data↩ Size must be set to the actual size of the buffer), or can be left NULL - in such case the buffer is allocated by SynView.

DataSize	Size of the buffer supplied, or 0 if the pDataPointer is NULL.
pUserPointer	A user pointer, which is then passed back in the LvEventCallbackNewBufferFunct(). It enables the application to reference some own data structure associated with the buffer.
Options	Currently unused, must be 0.
pBuffer	To this parameter the pointer to the LvBuffer instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvBuffer::Open().

5.18.2.10 LvStatus LvStream::OpenEvent (LvEventType *EventType*, LvEvent *& *pEvent*)

Creates the LvEvent class instance, owned by the Stream. This method is provided just for convenience, it has the same functionality as the LvEvent::Open() static method.

Parameters

EventType	One of the LvEventType.
pEvent	To this parameter the Event class instance is stored.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvEvent::Open().

5.18.2.11 LvStatus LvStream::OpenRenderer (LvRenderer *& *pRenderer*)

Creates the LvRenderer class instance for image display. The renderer attepts to load the sv.synview.display library. In case of SynView installation in an operating system without possibility to graphically display (for example Linux without XWindows), the load of this library fails and the calls to Renderer functions will return errors. This method is provided just for convenience, it has the same functionality as the LvRenderer::Open() static method.

Parameters

pRenderer	In this parameter the pointer to the LvRenderer is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvRenderer::Open().

5.18.2.12 LvStatus LvStream::Start (uint32_t *StartFlags* = 0x00000000, uint32_t *ImagesToAcquire* = 0xFFFFFFFF)

Starts the stream. This function need not be used on the image stream, where it is called automatically in the LvDevice::AcquisitionStart() function.

Parameters

StartFlags	One of the GroupSynView_StreamStartFlags.
ImagesTo↩ Acquire	Number of images to acquire.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.18.2.13 LvStatus LvStream::Stop (uint32_t *StopFlags* = 0x00000000)

Stops the stream. This function need not be used on the image stream, where it is called automatically in the LvDevice::AcquisitionStop() function.

Parameters

StopFlags	One of the GroupSynView_StreamStopFlags.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19 SynView LvBuffer methods

Functions

- static LvStatus LvBuffer::Open (LvStream *pStream, void *pDataPointer, size_t DataSize, void *pUser↩ Pointer, uint32_t Options, LvBuffer *&pBuffer)
- static LvStatus LvBuffer::Close (LvBuffer *&pBuffer)
- LvStatus LvBuffer::AttachProcessBuffer (void *pDataPointer, size_t DataSize)
- LvStatus LvBuffer::Queue ()
- LvStatus LvBuffer::ParseChunkData (bool UpdateLayout=false)
- LvStatus LvBuffer::SaveImageToBmpFile (const char *pFileName)
- LvStatus LvBuffer::SaveImageToJpgFile (const char *pFileName, uint32_t Quality)
- LvStatus LvBuffer::SaveImageToTifFile (const char *pFileName, uint32_t Options=0)
- LvStatus LvBuffer::GetImgInfo (LvipImgInfo &ImgInfo, uint32_t Options=0)
- LvStatus LvBuffer::GetLastPaintRect (int32_t *pX, int32_t *pY, int32_t *pWidth, int32_t *pHeight)
- LvStatus LvBuffer::UniCalculateWhiteBalance ()
- LvHBuffer LvBuffer::GetHandle ()
- void * LvBuffer::GetUserPtr ()

5.19.1 Detailed Description

5.19.2 Function Documentation

5.19.2.1 LvStatus LvBuffer::AttachProcessBuffer (void * *pDataPointer*, size_t *DataSize*)

Attaches a process buffer to a buffer. The process buffer may be needed for software processing, for example Bayer decoding, if the device hardware is not capable of it. The process buffer can be either supplied by the application by this function, or allocated automatically by SynView, upon need.

Parameters

pDataPointer	Pointer to the supplied buffer.
DataSize	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.2 static LvStatus LvBuffer::Close (LvBuffer *& *pBuffer*) [static]

Deletes the LvBuffer class instance. On the GenTL level it corresponds to the DSRevokeBuffer() function.

Parameters

pBuffer	A pointer to the LvBuffer instance, obtained from the LvBuffer::Open() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::CloseBuffer().

5.19.2.3 LvHBuffer LvBuffer::GetHandle ( )

Returns a handle of the Buffer (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.19.2.4 LvStatus LvBuffer::GetImgInfo (LvipImgInfo & *ImgInfo,* uint32_t *Options* = 0)

Fills the LvipImgInfo structure for the image in the buffer. This simplifies a direct use of the SynView Image Processing Library. If the image is processed, the image info points to the processed image, othewise it points to the original image.

Parameters

ImgInfo	The ImgInfo structure, to which are the image parameters stored.
Options	Currently unused, must be 0.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.5 LvStatus LvBuffer::GetLastPaintRect (int32_t * *pX,* int32_t * *pY,* int32_t * *pWidth,* int32_t * *pHeight*)

Returns the rectangle to which the buffer was last painted. This is useful namely in case you have a tile mode and want to identify the buffer according a mouse click location. If the buffer was not yet painted by the renderer, the returned values are 0.

Parameters

pX	Pointer to X offset in pixels.
pY	Pointer to Y offset in pixels.
pWidth	Pointer to Width in pixels.
pHeight	Pointer to Height in pixels.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.6 void* LvBuffer::GetUserPtr ( )

Returns a pointer to the Buffer (used in the Plain C API), associated with this class.

Returns

void pointer.

5.19.2.7 static LvStatus LvBuffer::Open (LvStream * *pStream,* void * *pDataPointer,* size_t *DataSize,* void * *pUserPointer,* uint32_t *Options,* LvBuffer *&* *pBuffer*) [static]

Creates the LvBuffer class instance. On the GenTL level it corresponds to DSAnnounceBuffer() or DSAllocAnd↩ AnnounceBuffer().

Parameters

pStream	A pointer to the LvStream instance, obtained from the LvStream::Open() function.
pDataPointer	Pointer to image data buffer. This can be supplied by the application (in such case the Data↩ Size must be set to the actual size of the buffer), or can be left NULL - in such case the buffer is allocated by SynView.
DataSize	Size of the buffer supplied, or 0 if the pDataPointer is NULL.
pUserPointer	A user pointer, which is then passed back in the LvEventCallbackNewBufferFunct(). It enables the application to reference some own data structure associated with the buffer.
Options	Currently unused, must be 0.
pBuffer	To this parameter the pointer to the LvBuffer instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::OpenBuffer().

5.19.2.8 LvStatus LvBuffer::ParseChunkData (bool *UpdateLayout* = `false`)

Parses the chunk data of the image. The chunk data are then accessible as device remote features (for example LvDevice_ChunkTimestamp).

Parameters

UpdateLayout	If set to true, the layout of chunk data is decoded. If set to false, the data are only read from already decoded layout - this is faster. Usually, the layout of the chunk data is constant, so it needs to be decoded only at first call of this function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.9 LvStatus LvBuffer::Queue ( )

Puts the buffer to the input buffer pool. This is an important part of the image handling loop: after the buffer with the acquired image is passed to the application, the application must return it to the input buffer pool by this function after processing.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.10 LvStatus LvBuffer::SaveImageToBmpFile (const char * *pFileName*)

Saves the image to a file in Windows BMP format. If the image is in the pixel format not compatible with the BMP format, it is automatically converted.

Parameters

pFileName	The file name. Be sure to specify it with the full path.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.11 LvStatus LvBuffer::SaveImageToJpgFile (const char * *pFileName*, uint32_t *Quality*)

Saves the image to a file in JPEG format. If the image is in the pixel format not compatible with the JPEG format, it is automatically converted.

Parameters

pFileName	The file name. Be sure to specify it with the full path.
Quality	The quality factor in range from from 1 to 100. The higher is the factor, the higher is the quality and lower the compression.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.12 LvStatus LvBuffer::SaveImageToTifFile (const char * *pFileName*, uint32_t *Options* = 0)

Saves the image to a file in the TIFF format. If the image is in the pixel format not compatible with the TIF format, it is automatically converted.

Parameters

pFileName	The file name. Be sure to specify it with the full path.
Options	Options for saved pixel format. The LvipOption_TiffConvertTo16Bit flag can be used there.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.19.2.13 LvStatus LvBuffer::UniCalculateWhiteBalance ( )

Calculates white balance factors from the current image.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20 SynView LvEvent methods

Functions

- static LvStatus LvEvent::Open (LvSystem *pSystem, LvEventType EventType, LvEvent *&pEvent)
- static LvStatus LvEvent::Open (LvDevice *pDevice, LvEventType EventType, LvEvent *&pEvent)
- static LvStatus LvEvent::Open (LvStream *pStream, LvEventType EventType, LvEvent *&pEvent)
- static LvStatus LvEvent::Close (LvEvent *&pEvent)
- LvStatus LvEvent::Kill ()
- LvStatus LvEvent::Flush ()
- LvStatus LvEvent::WaitAndGetData (void *pBuffer, size_t *pSize, uint32_t Timeout=0xFFFFFFFF)
- LvStatus LvEvent::WaitAndGetNewBuffer (LvBuffer *&pBuffer, uint32_t Timeout=0xFFFFFFFF)
- LvStatus LvEvent::GetDataInfo (void *pInBuffer, size_t InSize, LvEventDataInfo Info, void *pBuffer, size_t *pSize, LvInfoDataType *pType=NULL, int32_t Param=0)
- LvStatus LvEvent::PutData (void *pBuffer, size_t Size)
- LvStatus LvEvent::SetCallback (LvEventCallbackFunct pFunction, void *pUserParam)
- LvStatus LvEvent::SetCallbackNewBuffer (LvEventCallbackNewBufferFunct pFunction, void *pUserParam)
- LvStatus LvEvent::StartThread ()
- LvStatus LvEvent::StopThread ()
- LvHEvent LvEvent::GetHandle ()

5.20.1 Detailed Description

5.20.2 Function Documentation

5.20.2.1 static LvStatus LvEvent::Close (LvEvent *& *pEvent*) [static]

Deletes the LvEvent class instance.

Parameters

pEvent	A pointer to the LvEvent instance, obtained from the LvEvent::Open() function. This pointer is assigned NULL after the operation.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvSystem::CloseEvent(), LvDevice::CloseEvent(), LvStream::CloseEvent().

5.20.2.2 LvStatus LvEvent::Flush ( )

Discards all buffers in the output buffer queue (waiting to be delivered to the application).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.3 LvStatus LvEvent::GetDataInfo (void * *pInBuffer*, size_t *InSize*, LvEventDataInfo *Info*, void * *pBuffer*, size_t * *pSize*, LvInfoDataType * *pType* = NULL, int32_t *Param* = 0)

Enables to parse the buffer from LvEvent::WaitAndGetData.

Parameters

pInBuffer	Pointer to a buffer containing event data. This value must not be NULL.
InSize	Size of the provided pInBuffer in bytes.
Info	One of the LvEventDataInfo.
pBuffer	Pointer to a user allocated buffer to receive the requested information. If this parameter is NULL, pSize will contain the minimal size of pBuffer in bytes. If the pType is a string, the size includes the terminating 0.
pSize	Size of the buffer is returned in this parameter.
pType	One of the LvInfoDataType.
Param	Additional parameter, if used, its role is explained by the LvEventDataInfo.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.4 LvHEvent LvEvent::GetHandle ( )

Returns a handle of the Event (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.20.2.5 LvStatus LvEvent::Kill ( )

Terminates a single wait in the LvEvent::WaitAndGetData() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.6 static LvStatus LvEvent::Open (LvSystem * *pSystem*, LvEventType *EventType*, LvEvent *& *pEvent*) [static]

Creates the LvEvent class instance for specified LvSystem module.

Parameters

pSystem	Owner LvSystem instance.
EventType	One of the LvEventType.
pEvent	In this parameter a pointer to LvEvent instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvSystem::OpenEvent().

5.20.2.7 static LvStatus LvEvent::Open (LvDevice * *pDevice*, LvEventType *EventType*, LvEvent *& *pEvent*) [static]

Creates the LvEvent class instance for specified LvDevice module.

Parameters

pDevice	Owner LvDevice instance.
EventType	One of the LvEventType.
pEvent	In this parameter a pointer to LvEvent instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvDevice::OpenEvent().

5.20.2.8 static LvStatus LvEvent::Open (LvStream * *pStream*, LvEventType *EventType*, LvEvent *& *pEvent*) [static]

Creates the LvEvent class instance for specified LvStream module.

Parameters

pStream	Owner LvStream instance.
EventType	One of the LvEventType.
pEvent	In this parameter a pointer to LvEvent instance is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::OpenEvent().

5.20.2.9 LvStatus LvEvent::PutData (void * *pBuffer*, size_t *Size*)

Puts a new event to Event ouptut queue. This function can be used only for user-defined events.

Parameters

pBuffer	Pointer to event data.
Size	Size of the event data.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.10 LvStatus LvEvent::SetCallback (LvEventCallbackFunct *pFunction*, void * *pUserParam*)

Specifies a callback function for the event thread. Note that the callback function cannot be a method of a class.

Parameters

pFunction	The callback function in the forms of LvEventCallbackFunct.
pUserParam	User parameter, which will be passed to each callback call.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.11 LvStatus LvEvent::SetCallbackNewBuffer (LvEventCallbackNewBufferFunct *pFunction*, void * *pUserParam*)

Specifies a callback function for the thread of the Event of the LvEventType_NewBuffer. Once the application specifies this callback, it becomes responsible for returning the image buffers to the input buffer pool. Note that the callback function cannot be a method of a class.

Parameters

pFunction	The callback function in the forms of LvEventCallbackNewBufferFunct.
pUserParam	User parameter, which will be passed to each callback call.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.12 LvStatus LvEvent::StartThread ( )

Starts an internal thread, which waits for events and passes them to specified callback function. When the thread is started, the application must no longer call the LvEvent::WaitAndGetData() or LvEvent::WaitAndGetNewBufer() functions - this is called internally in the thread and upon return from this function a callback function is called.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.13 LvStatus LvEvent::StopThread ( )

Stops the event internal thread.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.14 LvStatus LvEvent::WaitAndGetData (void * *pBuffer*, size_t * *pSize*, uint32_t *Timeout* = 0xFFFFFFFF)

Waits for the event and gets its data in one atomic operation. Use this function only for events other than LvEvent↩ Type_NewBuffer, for the the LvEventType_NewBuffer event type use the LvEvent::WaitAndGetNewBuffer() function instead. Do not use this function if you use the callback - see LvEvent::SetCallback() or LvEvent::SetCallbackNew↩ Buffer().

Parameters

pBuffer	Pointer to a user allocated buffer to receive the event data. The buffer can be parsed by the LvEvent::GetDataInfo() function.
pSize	Size of the buffer must be specified in this parameter and after the function returns, the actual size is returned in this parameter.
Timeout	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the LvEvent::Kill() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.20.2.15 LvStatus LvEvent::WaitAndGetNewBuffer (LvBuffer *& *pBuffer*, uint32_t *Timeout* = 0xFFFFFFFF)

Waits for the event and gets its data in one atomic operation. Use this function only for events of the LvEventType_NewBuffer type. Do not use this function if you use the callback - see LvEvent::SetCallback() or LvEvent::SetCallbackNewBuffer().

Parameters

pBuffer	The pointer to the received LvBuffer instance is returned in this parameter.
Timeout	The wait timeout in milliseconds. The value 0xFFFFFFFF is considered as infinite. Note that you can also kill waiting from another thread using the LvEvent::Kill() function.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.21 SynView LvRenderer methods

Functions

- static LvStatus LvRenderer::Open (LvStream *pStream, LvRenderer *&pRenderer)
- static LvStatus LvRenderer::Close (LvRenderer *&pRenderer)
- LvStatus LvRenderer::SetWindow (void *pDisplay, int64_t hWindow)
- LvStatus LvRenderer::DisplayImage (LvBuffer *pBuffer, uint32_t RenderFlags=0)
- LvStatus LvRenderer::Repaint (uint32_t RenderFlags=0)
- LvHRenderer LvRenderer::GetHandle ()

5.21.1 Detailed Description

5.21.2 Function Documentation

5.21.2.1 static LvStatus LvRenderer::Close (LvRenderer *& *pRenderer*) [static]

Deletes the LvRenderer class instance.

Parameters

pRenderer	A pointer to the LvRenderer instance, obtained from the LvRenderer::Open() function. This pointer is set to NULL after close.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::CloseRenderer().

5.21.2.2 LvStatus LvRenderer::DisplayImage (LvBuffer * *pBuffer*, uint32_t *RenderFlags* = 0)

Displays the image. The image display mode is set by Renderer features, see LvRendererFtr.

Parameters

pBuffer	Pointer to the LvBuffer to be displayed.
RenderFlags	Zero or a combination of LvRenderFlags.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.21.2.3 LvHRenderer LvRenderer::GetHandle ( )

Returns a handle of the Renderer (used in the Plain C API), associated with this class.

Returns

The Plain C API handle.

5.21.2.4 static LvStatus LvRenderer::Open (LvStream * *pStream,* LvRenderer *& *pRenderer*) [static]

Creates the LvRenderer class instance for image display. The renderer attepts to load the sv.synview.display library. In case of SynView installation in an operating system without possibility to graphically display (for example Linux without XWindows), the load of this library fails and the calls to Renderer functions will return errors.

Parameters

pStream	A pointer to the LvStream instance, obtained from the LvStream::Open() function.
pRenderer	In this parameter the pointer to the LvRenderer is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

See also

LvStream::OpenRenderer().

5.21.2.5 LvStatus LvRenderer::Repaint (uint32_t *RenderFlags* = 0)

Repaints the contents of the display window. In order to be able to repaint, all images to be displayed must be still held by the application, i.e. must not be returned to the input buffer pool. See also LvStream_LvPostponeQueue↩ Buffers feature. A typical usage of this function is in the WM_PAINT handler in a Windows application.

Parameters

RenderFlags	Zero or a combination of LvRenderFlags.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.21.2.6 LvStatus LvRenderer::SetWindow (void * *pDisplay*, int64_t *hWindow*)

Sets the target window, in which the renderer has to display. Note that the application itself assure any repainting (when the window need to be repainted due to a movement of overlapping) - use LvRenderer::Repaint() in such case.

Parameters

pDisplay	Pointer to the display. It is defined as void* in order to make SynView header files independent on XWindows.
hWindow	Handle to the window.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22 SynView LvModule methods

Functions

- LvStatus LvModule::GetNumFeatures (LvFtrGroup FtrGroup, uint32_t *pNumFeatures)
- LvStatus LvModule::GetFeatureAt (LvFtrGroup FtrGroup, uint32_t Index, LvFeature *pFeature, uint32_t *pLevel=NULL)
- LvStatus LvModule::GetFeatureByName (LvFtrGroup FtrGroup, const char *pName, LvFeature *pFeature)
- bool LvModule::IsImplemented (LvFeature Feature)
- bool LvModule::IsImplementedByName (LvEnum FeatureGroup, const char *pName)
- bool LvModule::IsAvailable (LvFeature Feature)
- bool LvModule::IsAvailableByName (LvEnum FeatureGroup, const char *pName)
- bool LvModule::IsReadable (LvFeature Feature)
- bool LvModule::IsWritable (LvFeature Feature)
- bool LvModule::IsAvailableEnumEntry (LvFeature Feature, LvEnum EnumEntry)
- bool LvModule::IsImplementedEnumEntry (LvFeature Feature, LvEnum EnumEntry)
- LvStatus LvModule::GetType (LvFeature Feature, LvFtrType *pFtrType, LvFtrGui *pFtrGui=NULL, LvFtrGroup *pFtrGroup=NULL)
- LvStatus LvModule::GetBool (LvFeature Feature, bool *pValue)
- LvStatus LvModule::SetBool (LvFeature Feature, bool Value)
- LvStatus LvModule::GetInt32 (LvFeature Feature, int32_t *pValue)
- LvStatus LvModule::SetInt32 (LvFeature Feature, int32_t Value)
- LvStatus LvModule::GetInt32Range (LvFeature Feature, int32_t *pMinValue, int32_t *pMaxValue, int32_t *pIncrement)
- LvStatus LvModule::GetInt64 (LvFeature Feature, int64_t *pValue)
- LvStatus LvModule::SetInt64 (LvFeature Feature, int64_t Value)
- LvStatus LvModule::GetInt64Range (LvFeature Feature, int64_t *pMinValue, int64_t *pMaxValue, int64_t *pIncrement)
- LvStatus LvModule::GetInt (LvFeature Feature, int64_t *pValue)
- LvStatus LvModule::SetInt (LvFeature Feature, int64_t Value)
- LvStatus LvModule::GetIntRange (LvFeature Feature, int64_t *pMinValue, int64_t *pMaxValue, int64_t *pIncrement)
- LvStatus LvModule::GetFloat (LvFeature Feature, double *pValue)
- LvStatus LvModule::SetFloat (LvFeature Feature, double Value)
- LvStatus LvModule::GetFloatRange (LvFeature Feature, double *pMinValue, double *pMaxValue, double *pIncrement=NULL)
- LvStatus LvModule::GetString (LvFeature Feature, char *pValue, size_t Size)
- LvStatus LvModule::GetStringSize (LvFeature Feature, size_t *pSize)
- LvStatus LvModule::GetString (LvFeature Feature, std::string &sValue)
- LvStatus LvModule::SetString (LvFeature Feature, const char *pValue)
- LvStatus LvModule::GetBuffer (LvFeature Feature, void *pBuffer, size_t Size)
- LvStatus LvModule::GetBufferSize (LvFeature Feature, size_t *pSize)
- LvStatus LvModule::SetBuffer (LvFeature Feature, void *pBuffer, size_t Size)
- LvStatus LvModule::GetPtr (LvFeature Feature, void **ppValue)
- LvStatus LvModule::SetPtr (LvFeature Feature, void *pValue)
- LvStatus LvModule::GetEnum (LvFeature Feature, LvEnum *pValue)
- LvStatus LvModule::SetEnum (LvFeature Feature, LvEnum Value)
- LvStatus LvModule::GetEnumStr (LvFeature Feature, char *pSymbolicName, size_t Size)
- LvStatus LvModule::GetEnumStr (LvFeature Feature, std::string &sSymbolicName)
- LvStatus LvModule::SetEnumStr (LvFeature Feature, const char *pSymbolicName)
- LvStatus LvModule::GetEnumValByStr (LvFeature Feature, const char *pSymbolicName, LvEnum *pValue, LvFtrAccess *pFtrAccess=NULL)
- LvStatus LvModule::GetEnumStrByVal (LvFeature Feature, LvEnum Value, char *pSymbolicName, size_t SymbolicNameSize, LvFtrAccess *pFtrAccess=NULL)

- LvStatus LvModule::GetEnumStrByVal (LvFeature Feature, LvEnum Value, std::string &sSymbolicName, LvFtrAccess *pFtrAccess=NULL)
- LvStatus LvModule::CmdExecute (LvFeature Feature, uint32_t Timeout=0)
- LvStatus LvModule::CmdIsDone (LvFeature Feature, bool *pIsDone)
- LvStatus LvModule::GetAccess (LvFeature Feature, LvFtrAccess *pFtrAccess)
- LvStatus LvModule::GetVisibility (LvFeature Feature, LvFtrVisibility *pFtrVisibility)
- LvStatus LvModule::GetInfo (LvFeature Feature, LvFtrInfo FtrInfo, int32_t *pInfo, int32_t Param=0)
- LvStatus LvModule::GetInfoStr (LvFeature Feature, LvFtrInfo FtrInfo, char *pInfoStr, size_t Size, int32_t Param=0)
- LvStatus LvModule::GetInfoStrSize (LvFeature Feature, LvFtrInfo FtrInfo, size_t *pSize, int32_t Param=0)
- LvStatus LvModule::GetInfoStr (LvFeature Feature, LvFtrInfo FtrInfo, std::string &sInfoStr, int32_t Param=0)
- LvStatus LvModule::RegisterFeatureCallback (LvFeature Feature, LvFeatureCallbackFunct pFunction, void *pUserParam=NULL, void *pFeatureParam=NULL)
- LvStatus LvModule::StartPollingThread (uint32_t PollingTime=1000, bool PollChildren=false)
- LvStatus LvModule::StopPollingThread ()
- LvStatus LvModule::Poll ()

Variables

- LvHModule LvModule::m_hModule

5.22.1 Detailed Description

5.22.2 Function Documentation

5.22.2.1 LvStatus LvModule::CmdExecute (LvFeature *Feature*, uint32_t *Timeout* = 0)

Executes a command.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Timeout	If greater than 0, the LvModule::CmdIsDone() is called in a loop to wait for the command completion, until the LvModule::CmdIsDone() returns true or the Timeout (in milliseconds) expires. If set to 0, no wait is done.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.2 LvStatus LvModule::CmdIsDone (LvFeature *Feature*, bool * *pIsDone*)

Checks if the command execution has completed.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pIsDone	In this parameter is returned true, if the command is completed, otherwise false.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.3 LvStatus LvModule::GetAccess (LvFeature *Feature,* LvFtrAccess * *pFtrAccess*)

Gets the access mode of the feature.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pFtrAccess	The access is returned in this parameter. One of the LvFtrAccess.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.4 LvStatus LvModule::GetBool (LvFeature *Feature*, bool * *pValue*)

Gets a Boolean value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	The bool value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.5 LvStatus LvModule::GetBuffer (LvFeature *Feature*, void * *pBuffer*, size_t *Size*)

Gets a block of data.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pBuffer	Pointer to a buffer, to which the data will be stored.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.6 LvStatus LvModule::GetBufferSize (LvFeature *Feature*, size_t * *pSize*)

Gets the block data size.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pSize	The needed size of the buffer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.7 LvStatus LvModule::GetEnum (LvFeature *Feature*, LvEnum * *pValue*)

Gets the SynView constant for the enumeration entry, if exists. If does not exist, you must work with the string enumeration entry value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	SynView constant for the enum entry is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.8 LvStatus LvModule::GetEnumStr (LvFeature *Feature*, char * *pSymbolicName*, size_t *Size*)

Gets the enumeration entry as a string (symbolic name). It is not possible to get the needed size for this single feature, instead, it is possible to get the maximum size of the all enum values of this feature, by the LvModule::↩ GetInfo(LvFtrInfo_EnumEntryNameMaxSize) function.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pSymbolicName	A pointer to a string buffer, where the symbolic name will be returned.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.9 LvStatus LvModule::GetEnumStr (LvFeature *Feature*, std::string & *sSymbolicName*)

Gets the enumeration entry as a standard string (symbolic name).

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
sSymbolicName	A string, where the symbolic name will be returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.10 LvStatus LvModule::GetEnumStrByVal (LvFeature *Feature*, LvEnum *Value*, char * *pSymbolicName*, size_t *SymbolicNameSize*, LvFtrAccess * *pFtrAccess* = NULL)

Returns a string symbolic name of the enum entry for the SynView constant.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	The SynView constant for the enum entry.
pSymbolicName	Pointer to string buffer, where the symbolic name is returned. Can be NULL.

SymbolicName↩ Size	Size of pSymbolicName buffer.
pFtrAccess	The access mode of the enum entry is returned in this parameter - one of LvFtrAccess. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.11 LvStatus LvModule::GetEnumStrByVal (LvFeature *Feature*, LvEnum *Value*, std::string & *sSymbolicName*, LvFtrAccess * *pFtrAccess* = NULL)

Returns a string symbolic name of the enum entry for the SynView constant.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	The SynView constant for the enum entry.
sSymbolicName	In this parameter the symbolic name is returned.
pFtrAccess	The access mode of the enum entry is returned in this parameter - one of LvFtrAccess. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.12 LvStatus LvModule::GetEnumValByStr (LvFeature *Feature*, const char * *pSymbolicName*, LvEnum * *pValue*, LvFtrAccess * *pFtrAccess* = NULL)

Gets the SynView constant for the enumeration entry, if exists.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pSymbolicName	A string with symbolic name of the enum entry.
pValue	The SynView constant for the enum entry is returned in this parameter. If the SynView constant does not exist for this enumeration entry, 0 is returned (no error in indicated).
pFtrAccess	The feature access is returned in this parameter - one of LvFtrAccess. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.13 LvStatus LvModule::GetFeatureAt (LvFtrGroup *FtrGroup*, uint32_t *Index*, LvFeature * *pFeature*, uint32_t * *pLevel* = NULL)

Returns the feature ID at specified position. Can be used to iterate all the features in a list.

Parameters

FtrGroup	One of the LvFtrGroup.
Index	Zero based index of the feature in the list.
pFeature	Feature ID is returned in this parameter.
pLevel	Feature Level expressing its position in the tree is returned in this parameter. The base level has value 1.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.14 LvStatus LvModule::GetFeatureByName (LvFtrGroup *FtrGroup*, const char * *pName*, LvFeature * *pFeature*)

Returns a feature ID based on the feature name. This function is a substantial function for the generic approach to the feature - by this function you can get the ID of any existing feature, that means also for those, for which a SynView constant is not defined. Be sure to check the success of this function - if the feature is not mandatory, it may not exist.

Parameters

FtrGroup	One of the LvFtrGroup.
pName	Name of the feature.
pFeature	Feature ID is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.15 LvStatus LvModule::GetFloat (LvFeature *Feature*, double * *pValue*)

Gets a float value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	The float value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.16 LvStatus LvModule::GetFloatRange (LvFeature *Feature*, double * *pMinValue*, double * *pMaxValue*, double * *pIncrement* = NULL)

Returns a range of a float feature.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.

pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. If the increment is not defined, 0 is returned. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.17 LvStatus LvModule::GetInfo (LvFeature *Feature*, LvFtrInfo *FtrInfo*, int32_t * *pInfo*, int32_t *Param* = 0)

Gets an info in form of a 32-bit integer value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
FtrInfo	One of the LvFtrInfo.
pInfo	The value is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.18 LvStatus LvModule::GetInfoStr (LvFeature *Feature*, LvFtrInfo *FtrInfo*, char * *pInfoStr*, size_t *Size*, int32_t *Param* = 0)

Gets an info in form of a string value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
FtrInfo	One of the LvFtrInfo.
pInfoStr	The string value is returned in this parameter.
Size	Size of the buffer (to which pInfoStr points).
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.19 LvStatus LvModule::GetInfoStr (LvFeature *Feature*, LvFtrInfo *FtrInfo*, std::string & *sInfoStr*, int32_t *Param* = 0)

Gets an info in form of a string value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.

FtrInfo	One of the LvFtrInfo.
sInfoStr	The string value is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.20 LvStatus LvModule::GetInfoStrSize (LvFeature *Feature*, LvFtrInfo *FtrInfo*, size_t * *pSize*, int32_t *Param* = 0)

Gets a buffer size needed for an info in form of a string value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
FtrInfo	One of the LvFtrInfo.
pSize	Size of the buffer is returned in this parameter.
Param	Additional parameter, required by some types of info.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.21 LvStatus LvModule::GetInt (LvFeature *Feature*, int64_t * *pValue*)

Gets a 64-bit integer value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	The integer value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvModule::GetInt64() function.

5.22.2.22 LvStatus LvModule::GetInt32 (LvFeature *Feature*, int32_t * *pValue*)

Gets a 32-bit integer value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.

pValue	The integer value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.22.2.23 LvStatus LvModule::GetInt32Range (LvFeature *Feature*, int32_t * *pMinValue*, int32_t * *pMaxValue*, int32_t * *pIncrement*)

Returns a range and increment of an 32-bit integer feature.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.22.2.24 LvStatus LvModule::GetInt64 (LvFeature *Feature*, int64_t * *pValue*)

Gets a 64-bit integer value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	The integer value is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvModule::GetInt() function.

5.22.2.25 LvStatus LvModule::GetInt64Range (LvFeature *Feature*, int64_t * *pMinValue*, int64_t * *pMaxValue*, int64_t * *pIncrement*)

Returns a range and increment of an 64-bit integer feature.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvModule::GetIntRange() function.

5.22.2.26 LvStatus LvModule::GetIntRange (LvFeature *Feature*, int64_t * *pMinValue*, int64_t * *pMaxValue*, int64_t * *pIncrement*)

Returns a range and increment of an 64-bit integer feature.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pMinValue	The minimum value is returned in this parameter. Can be NULL.
pMaxValue	The maximum value is returned in this parameter. Can be NULL.
pIncrement	The increment value is returned in this parameter. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvModule::GetInt64Range() function.

5.22.2.27 LvStatus LvModule::GetNumFeatures (LvFtrGroup *FtrGroup*, uint32_t * *pNumFeatures*)

Returns a number of features for specified group. This is useful for building a list of all available features (like the tree in lv.explorer).

Parameters

FtrGroup	One of the LvFtrGroup.
pNumFeatures	The number of features is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.28 LvStatus LvModule::GetPtr (LvFeature *Feature*, void ** *ppValue*)

Gets a pointer.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
ppValue	The pointer is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.29 LvStatus LvModule::GetString (LvFeature *Feature*, char * *pValue*, size_t *Size*)

Gets a string value. If you need first to get the string size, use the LvModule::GetStringSize() function.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	Pointer to a null-terminated string buffer.
Size	Size of the buffer.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.30 LvStatus LvModule::GetString (LvFeature *Feature*, std::string & *sValue*)

Gets a string value as std::string.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
sValue	In this parametr the string value is returned.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.31 LvStatus LvModule::GetStringSize (LvFeature *Feature*, size_t * *pSize*)

Gets a buffer size needed for a string.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pSize	Size of the buffer (including space for terminating zero) is returned in this parameter.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.32 LvStatus LvModule::GetType (LvFeature *Feature*, LvFtrType * *pFtrType*, LvFtrGui * *pFtrGui* = NULL, LvFtrGroup * *pFtrGroup* = NULL)

Returns the feature type, GUI representation and group.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pFtrType	The feature type is returned in this parameter. The returned value is one of the LvFtrType. Can be NULL.
pFtrGui	The feature GUI representation is returned in this parameter. The returned value is one of the LvFtrGui. Can be NULL.
pFtrGroup	The feature group, to which the feature belongs. The returned value is one of the LvFtrGroup. Can be NULL.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.33 LvStatus LvModule::GetVisibility (LvFeature *Feature*, LvFtrVisibility * *pFtrVisibility*)

Gets the feature visibility (beginner-expert-guru).

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pFtrVisibility	The visibility is returned in this parameter. One of the LvFtrVisibility.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.34 bool LvModule::IsAvailable (LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the LvModule::GetAccess() function.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.

Returns

If the feature is available, returns true, otherwise false.

5.22.2.35 bool LvModule::IsAvailableByName (LvEnum *FeatureGroup*, const char * *pName*)

A helper function, allowing simply to determine, if a feature is available. It is a wrapper around the LvModule::GetAccess() and LvModule::GetFeatureByName() functions.

Parameters

FeatureGroup	One of the LvFtrGroup.
pName	Name of the feature.

Returns

If the feature is available, returns true, otherwise false.

5.22.2.36 bool LvModule::IsAvailableEnumEntry (LvFeature *Feature,* LvEnum *EnumEntry*)

A helper function, allowing simply to determine, if an enum entry of an enum feature is available.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
EnumEntry	The SynView constant for the enum entry.

Returns

If the enum entry is available, returns true, otherwise false.

5.22.2.37 bool LvModule::IsImplemented (LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the LvModule::GetAccess() function.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.

Returns

If the feature is implemented, returns true, otherwise false.

5.22.2.38 bool LvModule::IsImplementedByName (LvEnum *FeatureGroup*, const char * *pName*)

A helper function, allowing simply to determine, if a feature is implemented. It is a wrapper around the LvModule::GetAccess() and LvModule::GetFeatureByName() functions.

Parameters

FeatureGroup	One of the LvFtrGroup.
pName	Name of the feature.

Returns

If the feature is implemented, returns true, otherwise false.

5.22.2.39 bool LvModule::IsImplementedEnumEntry (LvFeature *Feature*, LvEnum *EnumEntry*)

A helper function, allowing simply to determine, if an enum entry of an enum feature is implemented.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
EnumEntry	The SynView constant for the enum entry.

Returns

If the enum entry is implemented, returns true, otherwise false.

5.22.2.40 bool LvModule::IsReadable (LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is readable. It is a wrapper around the LvModule::GetAccess() function.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.

Returns

If the feature is readable, returns true, otherwise false.

5.22.2.41 bool LvModule::IsWritable (LvFeature *Feature*)

A helper function, allowing simply to determine, if a feature is writable. It is a wrapper areound the LvModule::Get↩Access() function.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.

Returns

If the feature is writable, returns true, otherwise false.

5.22.2.42 LvStatus LvModule::Poll ( )

Polls all the non-cached features of the module. If the feature polling interval expires, the value is read and the feature callback is called.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.43 LvStatus LvModule::RegisterFeatureCallback (LvFeature *Feature*, LvFeatureCallbackFunct *pFunction*, void * *pUserParam* = NULL, void * *pFeatureParam* = NULL)

Registers or unregisters a callback function for the feature. This callback is produced by GenApi when a feature changes its value or status. The application should process this callback fast. Note that the callback can be called also from another thread - see LvEventType_FeatureDevEvent. Important note: The feature callback function should never set any other feature. Doing so can lead to recursions, which would be probably hard to diagnose and could cause unexpected behavior.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pFunction	The callback function in the form of LvFeatureCallbackFunct. If you want to unregister the function, use NULL at this parameter.
pUserParam	User parameter, which will be passed to each callback call.
pFeatureParam	Second user parameter, which will be passed to each callback call.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.44 LvStatus LvModule::SetBool (LvFeature *Feature*, bool *Value*)

Sets a Boolean value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.45 LvStatus LvModule::SetBuffer (LvFeature *Feature*, void * *pBuffer*, size_t *Size*)

Sets a block of data.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pBuffer	Pointer to the data.
Size	Size of the data.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.46 LvStatus LvModule::SetEnum (LvFeature *Feature*, LvEnum *Value*)

Sets the enumeration entry by the SynView constant. If the SynView constant is not defined for the feature, then use LvModule::SetEnumStr() to set the enum entry by a string.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	SynView constant for the requested enumeration entry.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.47 LvStatus LvModule::SetEnumStr (LvFeature *Feature*, const char * *pSymbolicName*)

Sets enumeration entry by its string symbolic name.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pSymbolicName	A string with the symbolic name of the enumeration entry.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.48 LvStatus LvModule::SetFloat (LvFeature *Feature*, double *Value*)

Sets a float value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	The value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.49 LvStatus LvModule::SetInt (LvFeature *Feature*, int64_t *Value*)

Sets a 64-bit integer value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvModule::SetInt64() function.

5.22.2.50 LvStatus LvModule::SetInt32 (LvFeature *Feature*, int32_t *Value*)

Sets a 32-bit value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

The value is internally kept always as a 64-bit value; the functions for setting and getting a 32-bit value are provided just for convenience.

5.22.2.51 LvStatus LvModule::SetInt64 (LvFeature *Feature*, int64_t *Value*)

Sets a 64-bit integer value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
Value	Value to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

Note

This function is equal to the LvModule::SetInt() function.

5.22.2.52 LvStatus LvModule::SetPtr (LvFeature *Feature*, void * *pValue*)

Sets a pointer.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	The pointer to be set.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.53 LvStatus LvModule::SetString (LvFeature *Feature*, const char * *pValue*)

Sets a string value.

Parameters

Feature	The feature ID - use a symbolic constant (one of the Features) or an ID obtained by the LvModule::GetFeatureByName() function.
pValue	The string value (null-terminated).

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.54 LvStatus LvModule::StartPollingThread (uint32_t *PollingTime* = `1000`, bool *PollChildren* = `false`)

Starts a thread, which in a loop polls the non-cached features. If the feature polling interval expires, the value is read and the feature callback is called.

Parameters

PollingTime	A time in milliseconds between 2 calls to poll the features.
PollChildren	If set to true, also the features in all children modules are polled. For example, if your application uses only one System module, then it is a parent of all other modules, so the polling will be propagated to all modules from a single thread. If a module has started own polling thread, then it is excluded from the propagating.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.2.55 LvStatus LvModule::StopPollingThread ( )

Stops the polling thread. See LvModule::StartPollingThread() for details.

Returns

Returns the LvStatus value indicating the success of the requested operation. See LvStatus definitions.

5.22.3 Variable Documentation

5.22.3.1 LvHModule LvModule::m_hModule [protected]

The base class handle.

5.23 SynView

Modules

- SynView Plain C API functions
- SynView C++ API functions
- SynView defines and typedefs
- SynView enumerations
- SynView Image Processing Library
- SynView INI file API

5.23.1 Detailed Description

5.24 SynView defines and typedefs

Modules

- LvStreamStart() flags definitions
- LvStreamStop() flags definitions
- LvDeviceUniSetLut() and LvDeviceUniGetLut() flags definitions
- LvSaveFlag definitions
- LvPixelFormat definitions
- LvStatus definitions

Macros

- #define LV_DLLENTRY
- #define LVIP_DLLENTRY

Typedefs

- typedef uint32_t LvHModule
- typedef void(∗ LvEventCallbackFunct)(void ∗pBuffer, size_t Size, void ∗pUserParam)
- typedef void(∗ LvEventCallbackNewBufferFunct)(LvHBuffer hBuffer, void ∗pUserPointer, void ∗pUserParam)
- typedef void(∗ LvFeatureCallbackFunct)(void ∗pUserParam, void ∗pFeatureParam, const char ∗pName)

5.24.1 Detailed Description

5.24.2 Macro Definition Documentation

5.24.2.1 #define LV_DLLENTRY

Typedef for the API functions calling convention and export/import type.

5.24.2.2 #define LVIP_DLLENTRY

Typedef for the API functions calling convention and export/import type.

5.24.3 Typedef Documentation

5.24.3.1 typedef void(∗ LvEventCallbackFunct)(void ∗pBuffer, size_t Size, void ∗pUserParam)

Prototype for the general callback function, which can be registered at the Event, using the LvEventSetCallback() function or LvEvent::SetCallback() class. IMPORTANT: the function must have the LV_STDC calling convention.

Parameters

pBuffer	Pointer to buffer, extracted from the output queue.
Size	Buffer size.
pUserParam	User parameter, supplied in the LvEventSetCallback() function or LvEvent::SetCallback(). It enables the application to distingush from which object the callback was called in case the same callback function is shared by multiple Event modules.

5.24.3.2 typedef void(∗ LvEventCallbackNewBufferFunct)(LvHBuffer hBuffer, void ∗pUserPointer, void ∗pUserParam)

Prototype for the new image callback function, which can be registered at the Event, using the LvEventSetCallback↩NewBuffer() function or LvEvent::SetCallbackNewBuffer() class. IMPORTANT: the function must have the LV_ST↩DC calling convention.

Parameters

hBuffer	Handle to LvBuffer, extracted from the output queue.
pUserPointer	The pUserPointer of the LvBuffer is passed here. In the C++ wrapper class this is used to give direct pointer to the LvBuffer class instance.
pUserParam	User parameter, supplied in the LvEventSetCallbackNewBuffer() function or LvEvent::Set↩CallbackNewBuffer(). It enables the application to distinguish from which object the callback was called in case the same callback function is shared by multiple Event modules.

5.24.3.3 typedef void(∗ LvFeatureCallbackFunct)(void ∗pUserParam, void ∗pFeatureParam, const char ∗pName)

Prototype for the feature updated callback function, which can be registered using the LvRegisterFeatureCallback() function. IMPORTANT: the function must have the LV_STDC calling convention.

Parameters

pUserParam	User pUserParam, supplied in the LvRegisterFeatureCallback() function. It can be used to distinguish from which object the callback was called in case the same callback function is shared by multiple Event modules.
pFeatureParam	The pFeatureParam passed in the LvRegisterFeatureCallback(). It is usually used to identify the feature, which has changed.
pName	The string ID of the feature.

5.24.3.4 typedef uint32_t LvHModule

Base typedef for a handle to a module.

5.25 SynView enumerations

Modules

- Features
- Enumeration entries

Enumerations

- enum LvLibInfo {
  LvInfo_BinPath, LvInfo_AppDataPath, LvInfo_UserDataPath, LvInfo_CfgPath,
  LvInfo_InstPath, LvInfo_IniFile, LvInfo_BuildDate }
- enum LvFtrGroup {
  LvFtrGroup_DeviceRemote, LvFtrGroup_SystemGtl, LvFtrGroup_InterfaceGtl, LvFtrGroup_DeviceGtl,
  LvFtrGroup_StreamGtl, LvFtrGroup_BufferGtl, LvFtrGroup_SystemLocal, LvFtrGroup_InterfaceLocal,
  LvFtrGroup_DeviceLocal, LvFtrGroup_StreamLocal, LvFtrGroup_BufferLocal, LvFtrGroup_RendererLocal,
  LvFtrGroup_EventLocal, LvFtrGroup_BufferItemsGtl, LvFtrGroup_EventItemsGtl, LvFtrGroup_System↩
  Hidden,
  LvFtrGroup_InterfaceHidden, LvFtrGroup_DeviceHidden, LvFtrGroup_StreamHidden, LvFtrGroup_Buffer↩
  Hidden,
  LvFtrGroup_RendererHidden, LvFtrGroup_EventHidden }
- enum LvFtrType {
  LvFtrType_Integer, LvFtrType_Float, LvFtrType_String, LvFtrType_Enumeration,
  LvFtrType_Boolean, LvFtrType_Command, LvFtrType_Category, LvFtrType_StringList,
  LvFtrType_Pointer, LvFtrType_Buffer, LvFtrType_Other }
- enum LvFtrGui {
  LvFtrGui_IntEdit, LvFtrGui_IntEditHex, LvFtrGui_IntSlider, LvFtrGui_IntSliderLog,
  LvFtrGui_FloatEdit, LvFtrGui_FloatSlider, LvFtrGui_FloatSliderLog, LvFtrGui_Label,
  LvFtrGui_StringEdit, LvFtrGui_CheckBox, LvFtrGui_ComboBox, LvFtrGui_Button,
  LvFtrGui_IpV4Address, LvFtrGui_IpMacAddress, LvFtrGui_Undefined }
- enum LvFtrVisibility { LvFtrVisibility_Beginner, LvFtrVisibility_Expert, LvFtrVisibility_Guru, LvFtrVisibility_↩
  Invisible }
- enum LvFtrAccess {
  LvFtrAccess_NotImplemented, LvFtrAccess_NotAvailable, LvFtrAccess_WriteOnly, LvFtrAccess_ReadOnly,
  LvFtrAccess_ReadWrite }
- enum LvFtrInfo {
  LvFtrInfo_IsStreamable, LvFtrInfo_IsWrapped, LvFtrInfo_IsSelector, LvFtrInfo_IsCached,
  LvFtrInfo_PollingTime, LvFtrInfo_Name, LvFtrInfo_DisplayName, LvFtrInfo_Description,
  LvFtrInfo_PhysicalUnits, LvFtrInfo_ToolTip, LvFtrInfo_SymbolicConst, LvFtrInfo_SymbolicEnumConst,
  LvFtrInfo_SelectedFeatures, LvFtrInfo_SelectingFeatures, LvFtrInfo_SymbolicGroupConst, LvFtrInfo_↩
  ModuleName,
  LvFtrInfo_FitsTo32Bit, LvFtrInfo_TakeAsReadOnly, LvFtrInfo_EnumEntryName, LvFtrInfo_EnumEntry↩
  DisplayName,
  LvFtrInfo_EnumEntryDescription, LvFtrInfo_EnumEntryToolTip, LvFtrInfo_EnumEntryAccess, LvFtrInfo_↩
  EnumEntryValue,
  LvFtrInfo_EnumEntryCount, LvFtrInfo_EnumEntryNameMaxSize, LvFtrInfo_InterfaceID, LvFtrInfo_↩
  InterfaceDisplayName,
  LvFtrInfo_InterfaceTlType, LvFtrInfo_DeviceID, LvFtrInfo_DeviceVendor, LvFtrInfo_DeviceModel,
  LvFtrInfo_DeviceTlType, LvFtrInfo_DeviceDisplayName, LvFtrInfo_DeviceAccessStatus }
- enum LvInfoDataType {
  LvInfoDataType_Unknown, LvInfoDataType_String, LvInfoDataType_StringList, LvInfoDataType_Int16,
  LvInfoDataType_UInt16, LvInfoDataType_Int32, LvInfoDataType_UInt32, LvInfoDataType_Int64,
  LvInfoDataType_UInt64, LvInfoDataType_Float64, LvInfoDataType_Ptr, LvInfoDataType_Bool,
  LvInfoDataType_SizeT, LvInfoDataType_Buffer }
- enum LvQueueOperation {
  LvQueueOperation_InputToOutput, LvQueueOperation_OutputDiscard, LvQueueOperation_AllToInput, Lv↩

QueueOperation_UnqueuedToInput,
LvQueueOperation_AllDiscard }
- enum LvEventType { LvEventType_Error, LvEventType_NewBuffer, LvEventType_FeatureDevEvent }
- enum LvEventDataInfo { LvEventDataInfo_Id, LvEventDataInfo_Value }
- enum LvRenderFlags { LvRenderFlags_RepaintBackground, LvRenderFlags_DontPaintIncomplete, LvRenderFlags_IgnoreInvalidWinHandle }
- enum LvFindBy {
LvFindBy_UserID, LvFindBy_VendorName, LvFindBy_ModelName, LvFindBy_TLType,
LvFindBy_DisplayName, LvFindBy_GevIPAddress, LvFindBy_GevMACAddress, LvFindBy_SerialNumber,
LvFindBy_Any }

5.25.1 Detailed Description

5.25.2 Enumeration Type Documentation

5.25.2.1 enum LvEventDataInfo

LvEventDataInfo constants. Define values for the info specification in the LvEventGetDataInfo() function.

Enumerator

LvEventDataInfo_Id Represents the GenTL EVENT_DATA_ID - Event ID. See LvEventType for the explanation, what this ID means according to the event type.

LvEventDataInfo_Value Represents the GenTL EVENT_DATA_VALUE - Event Data. See LvEventType for the explanation, what this data means according to the event type.

5.25.2.2 enum LvEventType

LvEventType constants. Currently only the LvEventType_NewBuffer is supported by SynView.

Enumerator

LvEventType_Error Represents the GenTL EVENT_ERROR - Notification on module errors. For this type of event the LvEventDataInfo_Id is LvInfoDataType_Int32 and LvEventDataInfo_Value is LvInfoDataType_String.

LvEventType_NewBuffer Represents the GenTL EVENT_NEW_BUFFER - Notification on newly filled buffers placed to the output queue. For this type of event the LvEventDataInfo_Id is LvInfoDataType_Ptr (GenTL Buffer handle) and LvEventDataInfo_Value is LvInfoDataType_Ptr (Private pointer).

LvEventType_FeatureDevEvent Represents the GenTL EVENT_FEATURE_DEVEVENT. Notification if the GenTL Producer wants to inform the GenICam GenApi instance of the remote device that a GenApi compatible event was fired. This event is processed internally in SynView API - it is converted into the feature change callback - see the LvModule::RegisterFeatureCallback() function. However, the thread which checks the GenTL event and converts it into the callbacks must be started explicitly by the application - see the LvEventStartThread() function. This event type can be opened only on the Device module.

5.25.2.3 enum LvFindBy

Enum values for the LvSystemFindInterface() and LvInterfaceFindDevice() functions.

Enumerator

LvFindBy_UserID Can be used in the LvInterfaceFindDevice() for finding the device by its User ID (nickname).

LvFindBy_VendorName Can be used in the LvInterfaceFindDevice() for finding the device by its vendor name.

LvFindBy_ModelName Can be used in the LvInterfaceFindDevice() for finding the device by its model name.

LvFindBy_TLType Can be used in the LvSystemFindInterface() or LvInterfaceFindDevice() for finding the interface or device by its Transport Layer type. The search string can be then one of the following:

- "GEV" for GigE Vision,
- "CL" for Camera Link,
- "IIDC" for IIDC 1394,
- "UVC" for USB video class devices,
- "Custom" for not defined ones (for example the New Electronic Technology CorSight streaming device).

LvFindBy_DisplayName Can be used in the LvSystemFindInterface() or LvInterfaceFindDevice() for finding the interface or device by its display name.

LvFindBy_GevIPAddress Can be used in the LvSystemFindInterface() or LvInterfaceFindDevice() for finding the interface or device by its IP address (in case of the interface, it is the default IP address of the NIC).

LvFindBy_GevMACAddress Can be used in the LvInterfaceFindDevice() for finding the device by its model name.

LvFindBy_SerialNumber Can be used in the LvInterfaceFindDevice() for finding the device by its serial number.

LvFindBy_Any Tries to find the string in all available IDs (UserID, VendorName, ModelName...).

5.25.2.4 enum LvFtrAccess

LvFtrAccess constants. Define the current feature access mode. Used in the LvGetAccess(). Also used for enumeration features in functions LvGetEnumValByStr() and LvGetEnumStrByVal().

Enumerator

LvFtrAccess_NotImplemented The feature is not implemented at all.

LvFtrAccess_NotAvailable The feature is implemented, but under the current conditions is not available.

LvFtrAccess_WriteOnly The feature is available and is write only.

LvFtrAccess_ReadOnly The feature is available and is read only.

LvFtrAccess_ReadWrite The feature is available and is fully accessible.

5.25.2.5 enum LvFtrGroup

LvFtrGroup constants. Define the group of features. The group is composed of the module and the feature origin. The richest set is belonging to the Device module:

- Device remote features are those, which are provided by the device itself through GenICam GenApi.
- Device GenTL features are those, which are provided by the GenTL library through GenICam GenApi.
- Device local features are those, which are implemented directly in the SynView library. Used in LvGetNumFeatures(), LvGetFeatureAt(), LvGetFeatureByName().

Enumerator

LvFtrGroup_DeviceRemote Device remote features obtained from the device GenApi node tree.

LvFtrGroup_SystemGtl System features obtained from the GenTL GenApi node tree.

LvFtrGroup_InterfaceGtl Interface features obtained from the GenTL GenApi node tree.

LvFtrGroup_DeviceGtl Device features obtained from the GenTL GenApi node tree.

LvFtrGroup_StreamGtl Stream features obtained from the GenTL GenApi node tree.

LvFtrGroup_BufferGtl Buffer features obtained from the GenTL GenApi node tree.

LvFtrGroup_SystemLocal System local features, implemented in SynView.

LvFtrGroup_InterfaceLocal Interface local features, implemented in SynView.

LvFtrGroup_DeviceLocal Device local features, implemented in SynView.

LvFtrGroup_StreamLocal Stream local features, implemented in SynView.

LvFtrGroup_BufferLocal Buffer local features, implemented in SynView.

LvFtrGroup_RendererLocal Renderer local features, implemented in SynView.

LvFtrGroup_EventLocal Event local features, implemented in SynView.

LvFtrGroup_BufferItemsGtl Obsolete - will be removed. Buffer local GenTL features obtained from the GenTL plain C API.

LvFtrGroup_EventItemsGtl Obsolete - will be removed. Event local GenTL features obtained from the GenTL plain C API.

LvFtrGroup_SystemHidden System hidden features. Do not use, reserved for special purposes.

LvFtrGroup_InterfaceHidden Interface hidden features. Do not use, reserved for special purposes.

LvFtrGroup_DeviceHidden Device hidden features. Do not use, reserved for special purposes.

LvFtrGroup_StreamHidden Stream hidden features. Do not use, reserved for special purposes.

LvFtrGroup_BufferHidden Buffer hidden features. Do not use, reserved for special purposes.

LvFtrGroup_RendererHidden Renderer hidden features. Do not use, reserved for special purposes.

LvFtrGroup_EventHidden Event hidden features. Do not use, reserved for special purposes.

5.25.2.6 enum LvFtrGui

LvFtrGui constants. Define the recommended GUI representation of the feature. Used in the LvGetType() function.

Enumerator

LvFtrGui_IntEdit The recommended representation is an edit box with a decimal value. Used by LvFtrType_Integer.

LvFtrGui_IntEditHex The recommended representation is an edit box with a hexadecimal value. Used by LvFtrType_Integer.

LvFtrGui_IntSlider The recommended representation is a linear slider. Used by LvFtrType_Integer.

LvFtrGui_IntSliderLog The recommended representation is a logarithmic slider. Used by LvFtrType_Integer.

LvFtrGui_FloatEdit The recommended representation is an edit box. Used by LvFtrType_Float.

LvFtrGui_FloatSlider The recommended representation is a linear slider. Used by LvFtrType_Float.

LvFtrGui_FloatSliderLog The recommended representation is a logarithmic slider. Used by LvFtrType_Float.

LvFtrGui_Label The recommended representation is read-only label. Used by LvFtrType_Category.

LvFtrGui_StringEdit The recommended representation is an edit box for a string. Used by LvFtrType_String.

LvFtrGui_CheckBox The recommended representation is a check box. Used by LvFtrType_Boolean.

LvFtrGui_ComboBox The recommended representation is a combo box. Used by LvFtrType_Boolean.

LvFtrGui_Button The recommended representation is a button. Used by LvFtrType_Command.

LvFtrGui_IpV4Address The recommended representation is an edit box for a string with an IP address in the form N.N.N.N. Used by LvFtrType_Integer.

LvFtrGui_IpMacAddress The recommended representation is an edit box for a string with a MAC address in the form XX:XX:XX:XX:XX:XX. Used by LvFtrType_Integer.

LvFtrGui_Undefined The recommended representation is not defined.

5.25.2.7 enum LvFtrInfo

LvFtrInfo constants. Define the info type when querying for feature info by the LvGetInfo() and LvGetInfoStr() functions.

Enumerator

LvFtrInfo_IsStreamable Returns 1 if the feature has the Streamable attribute set. To be used in the LvGetInfo() function.

LvFtrInfo_IsWrapped Returns 1 if the feature should not be used directly, because SynView provides for this functionality a native API. For example the AcquisitionStart and AcquisitionStop device remote features are wrapped by additional functionality in SynView (for example locking TL params before the AcquisitionStart command is issued). To be used in the LvGetInfo() function.

LvFtrInfo_IsSelector Returns 1 if the feature is a selector, that means subsequent features are indexed by it . To be used in the LvGetInfo() function.

LvFtrInfo_IsCached Returns 1 if the feature is cached. To be used in the LvGetInfo() function.

LvFtrInfo_PollingTime Returns the polling time for a non-cached feature. If the feature is dependent on other non-cached features, the returned polling time is the minimum found. The polling time defines recommended time to update the non-cached feature. For example the LvDevice_DeviceTemperature is a typical non-cached feature - it changes independentely and a as it changes slowly, the recommended polling time might be 10000 = 10 seconds, i.e. the application, which displays the temperature, should update it on screen every 10 seconds. The returned value -1 means the polling time is not defined. To be used in the LvGetInfo() function.

LvFtrInfo_Name Returns the feature Name. Do not confuse it with the DisplayName - the Name is the string identifier, by which the feature can be identified and a numeric ID can be obtained for further actions (generic feature access). To be used in the LvGetInfoStr() function.

LvFtrInfo_DisplayName Returns the feature Display name for representation in GUI. To be used in the LvGetInfoStr() function.

LvFtrInfo_Description Returns the feature Description text. To be used in the LvGetInfoStr() function.

LvFtrInfo_PhysicalUnits Returns the feature Physical units, if defined. To be used in the LvGetInfoStr() function.

LvFtrInfo_ToolTip Returns the feature Tooltip (a short description to be used in the GUI). To be used in the LvGetInfoStr() function.

LvFtrInfo_SymbolicConst Returns the SynView symbolic constant of the feature, as a string (utilized in the Source code generator). To be used in the LvGetInfoStr() function.

LvFtrInfo_SymbolicEnumConst Returns the SynView symbolic constant of the enumeration feature, as a string (utilized in the Source code generator). To be used in the LvGetInfoStr() function.

LvFtrInfo_SelectedFeatures Returns the string ID of selected features belonging under this selector. Param = index (utilized in the Source code generator). To be used in the LvGetInfoStr() function.

LvFtrInfo_SelectingFeatures Returns the string ID of selecting features under which this feature belongs. Param = index (utilized in the Source code generator). To be used in the LvGetInfoStr() function.

LvFtrInfo_SymbolicGroupConst Returns the SynView symbolic constant for the feature group, to which the feature belongs, as a string (utilized in the Source code generator). To be used in the LvGetInfoStr() function.

LvFtrInfo_ModuleName Returns the string indicating the type of module, to which the fature belongs, for example "System", "Interface", "Device", ... (utilized in the Source code generator). To be used in the LvGetInfoStr() function.

LvFtrInfo_FitsTo32Bit Returns 1 if for this feature can be safely used 32-bit integer instead of 64-bit (if the feaure is of the LvFtrType_Integer type). This info is utilized in the source code generator. To be used in the LvGetInfo() function.

LvFtrInfo_TakeAsReadOnly Returns 1 if this feature is either permanently read-only (cannot become read-write depending on other features), or the feature is writable, but it is not usual to set its value from code. This info is utilized in the source code generator. To be used in the LvGetInfo() function.

LvFtrInfo_EnumEntryName Returns the symbolic name of the enum entry. To be used in the LvGetInfo↩ Str() function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the LvGetInfo() function with the LvFtrInfo_EnumEntryCount parameter. If the Param is set to LV_ENUMENTRY_CURRENT, the returned info is for the currently selected enum entry.

LvFtrInfo_EnumEntryDisplayName Returns the display name of the enum entry. To be used in the Lv↩ GetInfoStr() function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the LvGetInfo() function with the LvFtrInfo_Enum↩ EntryCount parameter. If the Param is set to LV_ENUMENTRY_CURRENT, the returned info is for the currently selected enum entry.

LvFtrInfo_EnumEntryDescription Returns the description of the enum entry. To be used in the LvGetInfo↩ Str() function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the LvGetInfo() function with the LvFtrInfo_EnumEntryCount parameter. If the Param is set to LV_ENUMENTRY_CURRENT, the returned info is for the currently selected enum entry.

LvFtrInfo_EnumEntryToolTip Returns the tooltip of the enum entry. To be used in the LvGetInfoStr() function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the LvGetInfo() function with the LvFtrInfo_EnumEntryCount parameter. If the Param is set to LV_ENUMENTRY_CURRENT, the returned info is for the currently selected enum entry.

LvFtrInfo_EnumEntryAccess Returns the access of the enum entry (one of the LvFtrAccess constants). To be used in the LvGetInfo() function. The Param specifies a zero based index of the entry or the SynView enum entry constant. You can obtain the number of entries by the LvGetInfo() function with the LvFtr↩ Info_EnumEntryCount parameter. If the Param is set to LV_ENUMENTRY_CURRENT, the returned info is for the currently selected enum entry.

LvFtrInfo_EnumEntryValue Returns the SynView constant for the enum entry (if exists). To be used in the LvGetInfo() function. The Param specifies a zero based index of the entry. You can obtain the number of entries by the LvGetInfo() function with the LvFtrInfo_EnumEntryCount parameter. If the Param is set to LV_ENUMENTRY_CURRENT, the returned info is for the currently selected enum entry.

LvFtrInfo_EnumEntryCount Returns the number of enum entries for the enum. To be used in the LvGetInfo() function.

LvFtrInfo_EnumEntryNameMaxSize Returns the maximum string size needed (including terminating zero) for any entry name of the enum To be used in the LvGetInfo() function.

LvFtrInfo_InterfaceID Returns the string ID of the interface. Param = interface index. This constant can be used only in the LvSystem module for enumerating unopened interfaces (LvGetInfoStr() function, as the Feature use LvSystem_Info).

LvFtrInfo_InterfaceDisplayName Returns the Display name of the interface. Param = interface index. This constant can be used only in the LvSystem module for enumerating unopened interfaces (LvGetInfoStr() function, as the Feature use LvSystem_Info).

LvFtrInfo_InterfaceTlType Returns the interface Transport layer type. Param = interface index. This constant can be used only in the LvSystem module for enumerating unopened interfaces (LvGetInfoStr() function, as the Feature use LvSystem_Info). For example a standard interface TL type is "GEV" for GigE-Vision devices.

LvFtrInfo_DeviceID Returns the string ID of the device. Param = device index. This constant can be used only in the LvInterface module for enumerating unopened devices (LvGetInfoStr() function, as the Feature use LvInterface_Info).

LvFtrInfo_DeviceVendor Returns the Vendor name of the device. Param = device index. This constant can be used only in the LvInterface module for enumerating unopened devices (LvGetInfoStr() function, as the Feature use LvInterface_Info).

LvFtrInfo_DeviceModel Returns the Model name of the device. Param = device index. This constant can be used only in the LvInterface module for enumerating unopened devices (LvGetInfoStr() function, as the Feature use LvInterface_Info).

LvFtrInfo_DeviceTlType Returns the Transport layer type of the device. Param = device index. This constant can be used only in the LvInterface module for enumerating unopened devices (LvGetInfoStr() function, as the Feature use LvInterface_Info).

LvFtrInfo_DeviceDisplayName Returns the Display name the device. Param = device index. This constant can be used only in the LvInterface module for enumerating unopened devices (LvGetInfoStr() function, as the Feature use LvInterface_Info).

LvFtrInfo_DeviceAccessStatus Returns the the device access. Param = device index. The returned value is one of the LvDeviceAccessStatus constants. Can be used only in the LvInterface module for enumerating unopened devices (LvGetInfo() function, as the Feature use LvInterface_Info).

5.25.2.8 enum LvFtrType

LvFtrType constants. Define the type of the feature. Used in the LvGetType() function.

Enumerator

LvFtrType_Integer Integer type, use LvGetInt32(), LvSetInt32(), LvGetInt64(), LvSetInt64() to get/set a value.

LvFtrType_Float Float type, use LvGetFloat() and LvSetFloat() to get/set a value.

LvFtrType_String String type, use LvGetString() and LvSetString() to get/set a value.

LvFtrType_Enumeration Enumeration type, use LvGetEnum(), LvSetEnum(), LvGetEnumStr() and LvSetEnumStr() to get/set a value.

LvFtrType_Boolean Boolean type, use LvGetInt32() and LvSetInt32() to get/set a value.

LvFtrType_Command Command type, use LvCmdExecute() and LvCmdIsDone() to execute and check.

LvFtrType_Category Category type, used in the tree of feaures build.

LvFtrType_StringList String list type (multiple strings in one, separated by terminating 0), use LvGetString() and LvSetString() to get/set a value.

LvFtrType_Pointer Pointer type, use LvGetPtr() and LvSetPtr() to get/set a value.

LvFtrType_Buffer Buffer type (in GenICam it corresponds with the Register type), use LvGetBuffer() and LvSetBuffer() to get/set a value. Do not confuse it with the LvBuffer module.

LvFtrType_Other Unknown type, cannot be accessed.

5.25.2.9 enum LvFtrVisibility

LvFtrVisibility constants. Define the visibility level of the feature. Used in LvGetVisibility(). Should be used for displaying the feature tree (or list).

Enumerator

LvFtrVisibility_Beginner Beginner level - the feature should be displayed always.

LvFtrVisibility_Expert Expert level - the feature should be displayed if at least the Expert level is selected.

LvFtrVisibility_Guru Guru level - the feature should be displayed if at least the Guru level is selected.

LvFtrVisibility_Invisible Invisible - the feature should not be displayed.

5.25.2.10 enum LvInfoDataType

LvInfoDataType constants. The enum is used only by the LvEventGetDataInfo() function - this function follows the GenTL EventGetDataInfo() function, which uses different data types, than the GenApi.

Enumerator

LvInfoDataType_Unknown Represents the GenTL INFO_DATATYPE_UNKNOWN info - Unknown data type.

LvInfoDataType_String Represents the GenTL INFO_DATATYPE_STRING info - 0-terminated C string (ASCII encoded).

LvInfoDataType_StringList Represents the GenTL INFO_DATATYPE_STRINGLIST info - Concatenated INFO_DATATYPE_STRING list. End of list is signaled with an additional 0.

LvInfoDataType_Int16 Represents the GenTL INFO_DATATYPE_INT16 info - Signed 16 bit integer.

LvInfoDataType_UInt16 Represents the GenTL INFO_DATATYPE_UINT16 info - unsigned 16 bit integer.

LvInfoDataType_Int32 Represents the GenTL INFO_DATATYPE_INT32 info - signed 32 bit integer.

LvInfoDataType_UInt32 Represents the GenTL INFO_DATATYPE_UINT32 info - unsigned 32 bit integer.

LvInfoDataType_Int64 Represents the GenTL INFO_DATATYPE_INT64 info - signed 64 bit integer.

LvInfoDataType_UInt64 Represents the GenTL INFO_DATATYPE_UINT64 info - unsingned64 bit integer.

LvInfoDataType_Float64 Represents the GenTL INFO_DATATYPE_FLOAT64 info - Signed 64 bit floating point number.

LvInfoDataType_Ptr Represents the GenTL INFO_DATATYPE_PTR info - Pointer type (void*). Size is platform dependent (32 bit on 32 bit platforms).

LvInfoDataType_Bool Represents the GenTL INFO_DATATYPE_BOOL8 info - Boolean value occupying 8 bit. 0 for false and anything for true.

LvInfoDataType_SizeT Represents the GenTL INFO_DATATYPE_SIZET info - Platform dependent unsigned integer (32 bit on 32 bit platforms).

LvInfoDataType_Buffer Represents the GenTL INFO_DATATYPE_BUFFER info - Like the INFO_DATATYPE_STRING but with arbitrary data and no 0 termination.

5.25.2.11 enum LvLibInfo

Enum values for the Info parameter of the LvGetLibInfo(), LvGetLibInfoStr() and LvGetLibInfoStrSize() functions.

Enumerator

LvInfo_BinPath Returns the full path to the SynView binaries (applications and libraries - in Windows the Bin folder of SynView). LvFtrType_String.

LvInfo_AppDataPath Returns the full path to the SynView application data. This folder may be different from the BinPath, for example in Windows Vista the BinPath is write protected, while AppDataPath is at the read-write location and contains files like sv.synview.log etc. LvFtrType_String.

LvInfo_UserDataPath Returns the full path to the SynView user data. In Windows this is equal to AppDataPath. LvFtrType_String.

LvInfo_CfgPath Returns the full path to the SynView config data. In Windows this is equal to AppDataPath. LvFtrType_String.

LvInfo_InstPath Returns the full path to the SynView installation root folder. LvFtrType_String.

LvInfo_IniFile Returns the full path to the lv.SynView.ini file. LvFtrType_String.

LvInfo_BuildDate Returns the build date of the library.

5.25.2.12 enum LvQueueOperation

LvQueueOperation constants. Define enum values for the LvStreamFlushQueue() function.

Enumerator

LvQueueOperation_InputToOutput Represents the GenTL ACQ_QUEUE_INPUT_TO_OUTPUT. Flushes the input pool to the output queue and if necessary adds entries in the LvEventType_NewBuffer event data queue.

LvQueueOperation_OutputDiscard Represents the GenTL ACQ_QUEUE_OUTPUT_DISCARD. Discards all buffers in the output queue and if necessary removes the entries from the event data queue.

LvQueueOperation_AllToInput Represents the GenTL ACQ_QUEUE_ALL_TO_INPUT. Puts all buffers in the input pool. Even those in the output queue and discard entries in the event data queue.

LvQueueOperation_UnqueuedToInput Represents the GenTL ACQ_QUEUE_UNQUEUED_TO_INPUT. Puts all buffers that are not in the input pool or the output queue in the input pool.

LvQueueOperation_AllDiscard Represents the GenTL ACQ_QUEUE_ALL_DISCARD. Discards all buffers in the input pool and output queue.

5.25.2.13 enum LvRenderFlags

The flags passed as parameter to the functions LvRendererDisplayImage() and LvRendererRepaint().

Enumerator

LvRenderFlags_RepaintBackground Before painting the image, the window background is repainted. This is done automaticaly whenever the change of the window size is detected, or display mode is switched. You can also call LvRendererDisplayImage() with 0 as the buffer handle and this flag just to erase image painting area.

LvRenderFlags_DontPaintIncomplete If the buffer LvBuffer_IsIncomplete feature is true, it is not painted. The IsIncomplete feature indicates the contents of the buffer is a mixture of new and old image data, typically it happens when some packets with image data from a GigE camera are lost. If this flag is set simply the paint or repaint of such buffer is skipped, leaving whatever was before on the screen.

LvRenderFlags_IgnoreInvalidWinHandle This flag has a meaning ony for the LvRendererCanDisplayImage() function. If used, this function will not return an error if the window handle was not yet assigned by the LvRendererSetWindow() function. This can be utilized for checking if the image is displayable before the display window is actually used.

5.26 SynView Image Processing Library

Modules

- SynView Image Processing Library defines, typedefs and enums
- SynView Image Processing Library functions
- SynView Image Processing Library LvStatus definitions

5.26.1 Detailed Description

5.27 SynView Image Processing Library defines, typedefs and enums

Modules

- Definitions for Enumeration Entry Info

Classes

- struct LvipImgInfo

Macros

- #define LVIP_LUT_BAYER
- #define LVIP_LUT_BAYER_16

Enumerations

- enum LvipImgAttr {
  LvipImgAttr_BottomUp, LvipImgAttr_DWordAligned, LvipImgAttr_QWordAligned, LvipImgAttr_SSEAligned,
  LvipImgAttr_NotDataOwner }
- enum LvipOption {
  LvipOption_ReallocateDst, LvipOption_TiffConvertTo16Bit, LvipOption_BmpForceTopDown, LvipOption_↩
  BmpForceBottomUp,
  LvipOption_JpegConvertToBgr, LvipOption_JpegReadHeaderOnly, LvipOption_WbCorrectFactors }
- enum LvipLutType {
  LvipLutType_Uni, LvipLutType_8Bit, LvipLutType_10Bit, LvipLutType_12Bit,
  LvipLutType_UniBayer, LvipLutType_8BitBayer, LvipLutType_10BitBayer, LvipLutType_12BitBayer,
  LvipLutType_UniBayer16, LvipLutType_10BitBayer16, LvipLutType_12BitBayer16 }
- enum LvipColor { LvipColor_None }
- enum LvipTextAttr {
  LvipTextAttr_Bold, LvipTextAttr_Italic, LvipTextAttr_Underline, LvipTextAttr_Strikeout,
  LvipTextAttr_Nonantialiased, LvipTextAttr_Shadow, LvipTextAttr_Outline, LvipTextAttr_ShadowRB,
  LvipTextAttr_ShadowRT, LvipTextAttr_ShadowLB, LvipTextAttr_ShadowLT, LvipTextAttr_ShadowB,
  LvipTextAttr_ShadowT, LvipTextAttr_ShadowR, LvipTextAttr_ShadowL }

5.27.1 Detailed Description

5.27.2 Macro Definition Documentation

5.27.2.1 #define LVIP_LUT_BAYER

If the LUT is to be used in Bayer decoding/encoding functions, this attribute is to be OR-ed to the LvipLutType specification in the LvipAllocateLut() function Bayer LUT requires bigger size - is needed for the bilinear interpolation methods and for 10- and 12-bit source formats.

5.27.2.2 #define LVIP_LUT_BAYER_16

Bayer16 is a subset of LVIP_LUT_BAYER, suitable for all 10- and 12-bit decoding, with the exception of LvipBd↩
BilinearInterpolation() function.

5.27.3 Enumeration Type Documentation

5.27.3.1 enum LvipColor

Color definitions for the Overlay functions.

Enumerator

LvipColor_None Defines a non-color. This is useful for the transparent color - specifying the transparent color as LvipColor_None in LvipSetOverlayTransparentColor() switches off overlay transparency.

5.27.3.2 enum LvipImgAttr

Image attributes. Flags to be used in the Attributes of the LvipImgInfo structure.

Enumerator

LvipImgAttr_BottomUp Lines in the image buffer are ordered from the bottom line to the top line, so the image bufer begins with the bottom line.

LvipImgAttr_DWordAligned The line increment is aligned to double word (32 bits). This is required for example by the Windows Device Independent Bitmap format (DIB, BMP) This attribute is used only in the LvipInitImgInfo() function (which can be called as a result of the LvipOption_ReallocateDst attribute).

LvipImgAttr_QWordAligned The line increment is aligned to quad word (64 bits). This attribute is used in the LvipInitImgInfo() function (which can be called as a result of the LvipOption_ReallocateDst attribute).

LvipImgAttr_SSEAligned The line increment is aligned to SSE words (128 bits). This attribute is used in the LvipInitImgInfo() function (which can be called as a result of the LvipOption_ReallocateDst attribute).

LvipImgAttr_NotDataOwner The LvipImgInfo is not the owner of image data, so the LvipDeallocateImage↩ Data() function will not attempt to deallocate the image data. This attribute is used when the image data are owned by another LvipImgInfo or belonging to other code, for example when working directly with the image in the DMA buffer. Note that LvipDeallocateImageData() may be called from other functions, for example, when you use the LvipOption_ReallocateDst attribute.

5.27.3.3 enum LvipLutType

LUT type - to be used in the LvipAllocateLut() function.

Enumerator

LvipLutType_Uni LUT which internally contains 3 LUTs: 8-bit, 10-bit and 12-bit. All the LUTs are kept synchronized.

LvipLutType_8Bit 8-bit LUT type, used for images with LvPixelFormat_Mono8.

LvipLutType_10Bit 10-bit LUT type, used for images with LvPixelFormat_Mono10.

LvipLutType_12Bit 12-bit LUT type, used for images with LvPixelFormat_Mono12

LvipLutType_UniBayer LvipLutType_Uni type with the LVIP_LUT_BAYER.

LvipLutType_8BitBayer LvipLutType_8Bit type with the LVIP_LUT_BAYER.

LvipLutType_10BitBayer LvipLutType_10Bit type with the LVIP_LUT_BAYER.

LvipLutType_12BitBayer LvipLutType_12Bit type with the LVIP_LUT_BAYER.

LvipLutType_UniBayer16 LvipLutType_Uni type with the LVIP_LUT_BAYER_16.

LvipLutType_10BitBayer16 LvipLutType_10Bit type with the LVIP_LUT_BAYER_16.

LvipLutType_12BitBayer16 LvipLutType_12Bit type with the LVIP_LUT_BAYER_16.

5.27.3.4 enum LvipOption

Options for image processing functions in the Options parameter.

Enumerator

LvipOption_ReallocateDst The destination image data can be reallocated if it is needed. If the function stores a result of the operation to the destination image buffer, it first checks if the destination LvipImgInfo has appropriate parameters and the buffer(s) allocated. If not and this attribute is specified, it adapts the parameters of the LvipImgInfo and reallocates the buffer as needed. If this attribute is not specified, the function returns an error in case of mismatch.

LvipOption_TiffConvertTo16Bit The attribute will force conversion of the image to 16-bit mono format, if it is in 9- to 15-bit mono format. This can be used used when saving mono image to TIFF by the LvipSaveToTiff() function, as many software packages do not understand mono TIFF if it is in 9- to 15-bit mono format.

LvipOption_BmpForceTopDown The BMP file will be read to the top-down line layout. This attribute is used in the LvipLoadFromBmp() and LvipSaveToBmp() functions, as the BMP format can be either in the bottom-up line layout or in the top-down line layout.

LvipOption_BmpForceBottomUp The BMP file will be read to the bottom-up line layout. This attribute is used in the LvipLoadFromBmp() and LvipSaveToBmp() functions, as the BMP format can be either in the bottom-up line layout or in the top-down line layout.

LvipOption_JpegConvertToBgr The color JPEG images are stored in RGB format (24-bit). With this option the pixel format will be reversed to the BGR format in the LvipLoadFromJpg() function.

LvipOption_JpegReadHeaderOnly The JPEG image data will not be read, only the header will be read. This enables to allocate the image buffer and then read the full image.

LvipOption_WbCorrectFactors This attribute can be used in the LvipCalcWbFactors() function. If present, it is assumed that the white balance is calculated from the image to which were applied white balancing factors passed as input parameters. Thus only a correction is calculated and the existing factors are modified.

5.27.3.5 enum LvipTextAttr

Text attributes definitions for the Overlay functions.

Enumerator

LvipTextAttr_Bold Bold text. Text attribute for the LvipSetOverlayTextParams() function: Bold text

LvipTextAttr_Italic Italics text. Text attribute for the LvipSetOverlayTextParams() function: Italics text

LvipTextAttr_Underline Underlined text. Text attribute for the LvipSetOverlayTextParams() function: Underlined text

LvipTextAttr_Strikeout Strikeout text. Text attribute for the LvipSetOverlayTextParams() function: Strikeout text

LvipTextAttr_Nonantialiased Text antialiasing off. Text attribute for the LvipSetOverlayTextParams() function: Text antialiasing will be switched off - this is useful for text on transparent background, where antialiasing (like ClearType) can make undesirable effects.

LvipTextAttr_Shadow Text with a 1 pixel shadow. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at right-bottom direction.

LvipTextAttr_Outline Text with a 1 pixel outline. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel outline around the letters. This is useful namely for the text on transparent background - by adding the outline of different color, then the text is readable even if the background become of the same color, as the text.

LvipTextAttr_ShadowRB Text with a 1 pixel shadow at right-bottom direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at right-bottom direction (equal to LvipTextAttr_Shadow constant). This constant can be combined with other LvipTextAttr_ShadowXX constants.

LvipTextAttr_ShadowRT Text with a 1 pixel shadow at right-top direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at right-top direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

LvipTextAttr_ShadowLB Text with a 1 pixel shadow at left-bottom direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at left-bottom direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

LvipTextAttr_ShadowLT Text with a 1 pixel shadow at left-top direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at left-top direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

LvipTextAttr_ShadowB Text with a 1 pixel shadow at bottom direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at bottom direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

LvipTextAttr_ShadowT Text with a 1 pixel shadow at top direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at top direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

LvipTextAttr_ShadowR Text with a 1 pixel shadow at right direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at right direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

LvipTextAttr_ShadowL Text with a 1 pixel shadow at left direction. Text attribute for the LvipSetOverlayTextParams() function: Text with a 1 pixel shadow at left direction. This constant can be combined with other LVIP_TEXTATTR_SHADOW_x constants.

5.28 Definitions for Enumeration Entry Info

Macros

- #define LV_ENUMENTRY_CURRENT

Typedefs

- typedef LvHModule LvHSystem
- typedef LvHModule LvHInterface
- typedef LvHModule LvHDevice
- typedef LvHModule LvHStream
- typedef LvHModule LvHEvent
- typedef LvHModule LvHRenderer
- typedef LvHModule LvHBuffer
- typedef uint32_t LvHOverlay
- typedef uint32_t LvFeature
- typedef uint32_t LvEnum

5.28.1 Detailed Description

5.28.2 Macro Definition Documentation

5.28.2.1 #define LV_ENUMENTRY_CURRENT

If used as Param of the LvGetInfo(), LvGetInfoStr() and LvGetInfoStrSize() the returned value is for the current enum entry.

5.28.3 Typedef Documentation

5.28.3.1 typedef uint32_t LvEnum

Base typedef for the entry of the enumeration item.

5.28.3.2 typedef uint32_t LvFeature

Base typedef for the ID of the feature.

5.28.3.3 typedef LvHModule LvHBuffer

Typedef for a handle to the Buffer module.

5.28.3.4 typedef LvHModule LvHDevice

Typedef for a handle to the Device module.

5.28.3.5 typedef LvHModule LvHEvent

Typedef for a handle to the Event module.

5.28.3.6 typedef LvHModule LvHInterface

Typedef for a handle to the Interface module.

5.28.3.7 typedef uint32_t LvHOverlay

Typedef for a handle to the overlay.

5.28.3.8 typedef LvHModule LvHRenderer

Typedef for a handle to the Renderer module.

5.28.3.9 typedef LvHModule LvHStream

Typedef for a handle to the Stream module.

5.28.3.10 typedef LvHModule LvHSystem

Typedef for a handle to the System module.

5.29 Features

Enumerations

- enum LvSystemFtr {
  LvSystem_TLVendorName, LvSystem_TLModelName, LvSystem_TLID, LvSystem_TLVersion,
  LvSystem_TLPath, LvSystem_TLType, LvSystem_GenTLVersionMajor, LvSystem_GenTLVersionMinor,
  LvSystem_GevVersionMajor, LvSystem_GevVersionMinor, LvSystem_InterfaceUpdateList, LvSystem_InterfaceSelector,
  LvSystem_InterfaceID, LvSystem_GevInterfaceMACAddress, LvSystem_GevInterfaceDefaultIPAddress,
  LvSystem_GevInterfaceDefaultSubnetMask,
  LvSystem_GevInterfaceDefaultGateway, LvSystem_LvSystemDisplayName, LvSystem_Info }
- enum LvInterfaceFtr {
  LvInterface_InterfaceID, LvInterface_InterfaceType, LvInterface_GevInterfaceGatewaySelector, LvInterface_GevInterfaceGateway,
  LvInterface_GevMACAddress, LvInterface_GevInterfaceSubnetSelector, LvInterface_GevInterfaceSubnetIPAddress, LvInterface_GevInterfaceSubnetMask,
  LvInterface_DeviceUpdateList, LvInterface_DeviceSelector, LvInterface_DeviceID, LvInterface_DeviceVendorName,
  LvInterface_DeviceModelName, LvInterface_DeviceAccessStatus, LvInterface_GevDeviceIPAddress, LvInterface_GevDeviceSubnetMask,
  LvInterface_GevDeviceMACAddress, LvInterface_LvDeviceUserID, LvInterface_LvDeviceSerialNumber,
  LvInterface_LvInterfaceDisplayName,
  LvInterface_Info }
- enum LvDeviceFtr {
  LvDevice_DeviceVendorName, LvDevice_DeviceModelName, LvDevice_DeviceManufacturerInfo, LvDevice_DeviceVersion,
  LvDevice_DeviceFirmwareVersion, LvDevice_LvRecoveryFirmwareVersion, LvDevice_DeviceSerialNumber,
  LvDevice_DeviceUserID,
  LvDevice_LvSensorID, LvDevice_LvGrabberID, LvDevice_DeviceScanType, LvDevice_DeviceRegistersStreamingStart,
  LvDevice_DeviceRegistersStreamingEnd, LvDevice_DeviceRegistersCheck, LvDevice_DeviceRegistersValid, LvDevice_DeviceReset,
  LvDevice_DeviceClockSelector, LvDevice_DeviceClockFrequency, LvDevice_DeviceTemperatureSelector,
  LvDevice_DeviceTemperature,
  LvDevice_LvDeviceUpTime, LvDevice_LvDeviceType, LvDevice_SensorWidth, LvDevice_SensorHeight,
  LvDevice_WidthMax, LvDevice_HeightMax, LvDevice_Width, LvDevice_Height,
  LvDevice_OffsetX, LvDevice_OffsetY, LvDevice_PixelFormat, LvDevice_BinningHorizontal,
  LvDevice_BinningVertical, LvDevice_DecimationHorizontal, LvDevice_DecimationVertical, LvDevice_LvAOIMode,
  LvDevice_LvReadoutWidth, LvDevice_LvReadoutHeight, LvDevice_LvReadoutOffsetX, LvDevice_LvReadoutOffsetY,
  LvDevice_LvVariablePayloadSize, LvDevice_AcquisitionMode, LvDevice_TriggerSelector, LvDevice_TriggerMode,
  LvDevice_TriggerSoftware, LvDevice_TriggerSource, LvDevice_TriggerActivation, LvDevice_TriggerDelay,
  LvDevice_TriggerDivider, LvDevice_LvTriggerCaching, LvDevice_ExposureMode, LvDevice_LvLongRangeExposureMode,
  LvDevice_LvGlobalResetMode, LvDevice_ExposureTime, LvDevice_ExposureAuto, LvDevice_LvAcquisitionFrameRateControlMode,
  LvDevice_AcquisitionFrameRate, LvDevice_LineSelector, LvDevice_LineMode, LvDevice_LineFormat,
  LvDevice_LineSource, LvDevice_LineInverter, LvDevice_LineStatus, LvDevice_LineStatusAll,
  LvDevice_UserOutputSelector, LvDevice_UserOutputValue, LvDevice_UserOutputValueAll, LvDevice_UserOutputValueAllMask,
  LvDevice_CounterSelector, LvDevice_LvCounterMode, LvDevice_CounterEventSource, LvDevice_CounterReset,
  LvDevice_CounterValue, LvDevice_CounterDuration, LvDevice_TimerSelector, LvDevice_TimerDuration,
  LvDevice_TimerDelay, LvDevice_TimerTriggerSource, LvDevice_LvSpecialPurposeTriggerSelector, Lv

Device_LvSpecialPurposeTriggerSource,
LvDevice_LvSpecialPurposeTriggerActivation, LvDevice_LvSpecialPurposeTriggerSoftware, LvDevice_Lv↩
ImageStampsResetMask, LvDevice_LvImageStampSelector,
LvDevice_LvImageStampResetEnable, LvDevice_LvBootSwitch, LvDevice_LvBayerDecoderAlgorithm, Lv↩
Device_LvBayerDecoderThreshold,
LvDevice_LvWatchdogEnable, LvDevice_LvWatchdogTimerDuration, LvDevice_LvWatchdogTimerReset,
LvDevice_LvWatchdogFailed,
LvDevice_GainSelector, LvDevice_Gain, LvDevice_GainAuto, LvDevice_BlackLevelSelector,
LvDevice_BlackLevel, LvDevice_BlackLevelAuto, LvDevice_ColorTransformationSelector, LvDevice_Color↩
TransformationEnable,
LvDevice_ColorTransformationValueSelector, LvDevice_ColorTransformationValue, LvDevice_LvExternal↩
DeviceControlMode, LvDevice_LvExternalADCSelector,
LvDevice_LvExternalADCValue, LvDevice_LvPowerSwitchCurrentAction, LvDevice_LvPowerSwitch↩
Selector, LvDevice_LvPowerSwitchBoundADC,
LvDevice_LvPowerSwitchDrive, LvDevice_LvPowerSwitchPulsePlus, LvDevice_LvPowerSwitchPulseMinus,
LvDevice_LvLensControlCalibrate,
LvDevice_LvLensControlMinusEnd, LvDevice_LvLensControlPlusEnd, LvDevice_LvLensControlPulse↩
Period, LvDevice_LvLensControlDutyCycle,
LvDevice_LvLensControlTargetApproach, LvDevice_LvLensControlNrSlowSteps, LvDevice_LvLens↩
ControlTargetPosition, LvDevice_LvLensControlAdjustPosition,
LvDevice_LvPowerSwitchPulseDuration, LvDevice_LvLensControlMinCalibrationRange, LvDevice_Lv↩
LensControlCalibrateAll, LvDevice_LUTSelector,
LvDevice_LUTEnable, LvDevice_LUTIndex, LvDevice_LUTValue, LvDevice_LUTValueAll,
LvDevice_PayloadSize, LvDevice_GevVersionMajor, LvDevice_GevVersionMinor, LvDevice_GevDevice↩
ModeIsBigEndian,
LvDevice_GevDeviceModeCharacterSet, LvDevice_GevInterfaceSelector, LvDevice_GevMACAddress, Lv↩
Device_GevSupportedOptionSelector,
LvDevice_GevSupportedOption, LvDevice_GevCurrentIPConfigurationLLA, LvDevice_GevCurrentIP↩
ConfigurationDHCP, LvDevice_GevCurrentIPConfigurationPersistentIP,
LvDevice_GevCurrentIPAddress, LvDevice_GevCurrentSubnetMask, LvDevice_GevCurrentDefaultGateway,
LvDevice_GevPersistentIPAddress,
LvDevice_GevPersistentSubnetMask, LvDevice_GevPersistentDefaultGateway, LvDevice_GevNumberOf↩
Interfaces, LvDevice_GevMessageChannelCount,
LvDevice_GevStreamChannelCount, LvDevice_GevHeartbeatTimeout, LvDevice_GevTimestampTick↩
Frequency, LvDevice_GevTimestampControlLatch,
LvDevice_GevTimestampControlReset, LvDevice_GevTimestampControlLatchReset, LvDevice_Gev↩
TimestampValue, LvDevice_GevCCP,
LvDevice_GevStreamChannelSelector, LvDevice_GevSCPInterfaceIndex, LvDevice_GevSCPHostPort, Lv↩
Device_GevSCPSFireTestPacket,
LvDevice_GevSCPSDoNotFragment, LvDevice_GevSCPSBigEndian, LvDevice_GevSCPSPacketSize, Lv↩
Device_GevSCPD,
LvDevice_GevSCDA, LvDevice_GevLinkSpeed, LvDevice_UserSetSelector, LvDevice_UserSetLoad,
LvDevice_UserSetSave, LvDevice_UserSetDefaultSelector, LvDevice_ChunkModeActive, LvDevice_↩
ChunkSelector,
LvDevice_ChunkEnable, LvDevice_ChunkOffsetX, LvDevice_ChunkOffsetY, LvDevice_ChunkWidth,
LvDevice_ChunkHeight, LvDevice_ChunkPixelFormat, LvDevice_ChunkLinePitch, LvDevice_ChunkFrame↩
ID,
LvDevice_ChunkTimestamp, LvDevice_ChunkExposureTime, LvDevice_ChunkGainSelector, LvDevice_↩
ChunkGain,
LvDevice_ChunkBlackLevel, LvDevice_ChunkLineStatusAll, LvDevice_ChunkLvExternalADCSelector, Lv↩
Device_ChunkLvExternalADCValue,
LvDevice_EventSelector, LvDevice_EventNotification, LvDevice_LvSmartAppID, LvDevice_LvSmartApp↩
Int1,
LvDevice_LvSmartAppInt2, LvDevice_LvSmartAppInt3, LvDevice_LvSmartAppInt4, LvDevice_LvSmart↩
AppInt5,
LvDevice_LvSmartAppInt6, LvDevice_LvSmartAppInt7, LvDevice_LvSmartAppInt8, LvDevice_LvSmart↩
AppInt9,
LvDevice_LvSmartAppInt10, LvDevice_LvSmartAppInt11, LvDevice_LvSmartAppInt12, LvDevice_Lv↩

SmartAppInt13,
LvDevice_LvSmartAppInt14, LvDevice_LvSmartAppInt15, LvDevice_LvSmartAppInt16, LvDevice_Lv↩
SmartAppInt17,
LvDevice_LvSmartAppInt18, LvDevice_LvSmartAppInt19, LvDevice_LvSmartAppInt20, LvDevice_Lv↩
SmartAppInt21,
LvDevice_LvSmartAppInt22, LvDevice_LvSmartAppInt23, LvDevice_LvSmartAppInt24, LvDevice_Lv↩
SmartAppInt25,
LvDevice_LvSmartAppInt26, LvDevice_LvSmartAppInt27, LvDevice_LvSmartAppInt28, LvDevice_Lv↩
SmartAppInt29,
LvDevice_LvSmartAppInt30, LvDevice_LvSmartAppInt31, LvDevice_LvSmartAppInt32, LvDevice_Lv↩
SmartAppUint1,
LvDevice_LvSmartAppUint2, LvDevice_LvSmartAppUint3, LvDevice_LvSmartAppUint4, LvDevice_Lv↩
SmartAppUint5,
LvDevice_LvSmartAppUint6, LvDevice_LvSmartAppUint7, LvDevice_LvSmartAppUint8, LvDevice_Lv↩
SmartAppUint9,
LvDevice_LvSmartAppUint10, LvDevice_LvSmartAppUint11, LvDevice_LvSmartAppUint12, LvDevice_Lv↩
SmartAppUint13,
LvDevice_LvSmartAppUint14, LvDevice_LvSmartAppUint15, LvDevice_LvSmartAppUint16, LvDevice_Lv↩
SmartAppUint17,
LvDevice_LvSmartAppUint18, LvDevice_LvSmartAppUint19, LvDevice_LvSmartAppUint20, LvDevice_Lv↩
SmartAppUint21,
LvDevice_LvSmartAppUint22, LvDevice_LvSmartAppUint23, LvDevice_LvSmartAppUint24, LvDevice_Lv↩
SmartAppUint25,
LvDevice_LvSmartAppUint26, LvDevice_LvSmartAppUint27, LvDevice_LvSmartAppUint28, LvDevice_Lv↩
SmartAppUint29,
LvDevice_LvSmartAppUint30, LvDevice_LvSmartAppUint31, LvDevice_LvSmartAppUint32, LvDevice_Lv↩
SmartAppAsciiCmdString,
LvDevice_LvSmartAppAsciiCmdExecute, LvDevice_LvSmartAppAsciiCmdFeedback, LvDevice_LvSmart↩
AppAsciiCmdRetCode, LvDevice_LvSmartAppPath,
LvDevice_LvSmartAppStart, LvDevice_EventLvLog, LvDevice_EventLvLogTimestamp, LvDevice_EventLv↩
LogMessage,
LvDevice_EventLvSmartAppLog, LvDevice_EventLvSmartAppLogTimestamp, LvDevice_EventLvSmart↩
AppLogMessage, LvDevice_LvSerialPortBaudRate,
LvDevice_LvSerialPortParity, LvDevice_LvSerialPortDataBits, LvDevice_LvSerialPortStopBits, LvDevice_↩
LvSerialPortTimeout,
LvDevice_LvSerialPortEOTMarker, LvDevice_LvSerialPortMaxResponseLength, LvDevice_LvSerialPort↩
CommandString, LvDevice_LvSerialPortCommandSend,
LvDevice_LvSerialPortCommandResponse, LvDevice_LvSerialPortCommandStatus, LvDevice_LvSmart↩
AppExitEvent, LvDevice_LvWatchdogTimerValue,
LvDevice_LvLensControlInvertedPolarity, LvDevice_GevMCPHostPort, LvDevice_GevMCDA, LvDevice_↩
GevMCTT,
LvDevice_GevMCRC, LvDevice_ChunkLvSmartAppString, LvDevice_ChunkLvSmartAppIntSelector, Lv↩
Device_ChunkLvSmartAppInt,
LvDevice_ChunkLvSmartAppUintSelector, LvDevice_ChunkLvSmartAppUint, LvDevice_ChunkLvSmart↩
AppRegister, LvDevice_EventLvSmartAppString,
LvDevice_EventLvSmartAppStringTimestamp, LvDevice_EventLvSmartAppStringValue, LvDevice_Event↩
LvSmartAppInt, LvDevice_EventLvSmartAppIntTimestamp,
LvDevice_EventLvSmartAppIntSelector, LvDevice_EventLvSmartAppIntValue, LvDevice_EventLvSmart↩
AppUint, LvDevice_EventLvSmartAppUintTimestamp,
LvDevice_EventLvSmartAppUintSelector, LvDevice_EventLvSmartAppUintValue, LvDevice_EventLv↩
SmartAppRegister, LvDevice_EventLvSmartAppRegisterTimestamp,
LvDevice_EventLvSmartAppRegisterValue, LvDevice_DeviceSFNCVersionMajor, LvDevice_DeviceSFNC↩
VersionMinor, LvDevice_DeviceSFNCVersionSubMinor,
LvDevice_LvLineDebounceDuration, LvDevice_ActionDeviceKey, LvDevice_ActionSelector, LvDevice_↩
ActionGroupKey,
LvDevice_ActionGroupMask, LvDevice_LvLensControlCalibrationStatus, LvDevice_LvLUTMode, Lv↩
Device_BalanceRatioSelector,
LvDevice_BalanceRatio, LvDevice_BalanceWhiteAuto, LvDevice_GevDeviceClass, LvDevice_GevIP↩

ConfigurationStatus,
LvDevice_GevDiscoveryAckDelay, LvDevice_GevGVCPExtendedStatusCodes, LvDevice_GevGVCP↩
PendingAck, LvDevice_GevGVCPHeartbeatDisable,
LvDevice_GevGVCPPendingTimeout, LvDevice_GevPrimaryApplicationSwitchoverKey, LvDevice_Gev↩
PrimaryApplicationSocket, LvDevice_GevPrimaryApplicationIPAddress,
LvDevice_GevMCSP, LvDevice_GevSCCFGUnconditionalStreaming, LvDevice_GevSCCFGExtended↩
ChunkData, LvDevice_GevSCPDirection,
LvDevice_GevSCSP, LvDevice_ChunkLvTriggerDelayed, LvDevice_EventLvTriggerDropped, LvDevice_↩
EventLvTriggerDroppedTimestamp,
LvDevice_LvStrobeEnable, LvDevice_LvStrobeDurationMode, LvDevice_LvStrobeDuration, LvDevice_Lv↩
StrobeDelay,
LvDevice_LvStrobeBrightness, LvDevice_LvStrobeDropMode, LvDevice_LvLUTReset, LvDevice_Chunk↩
LvStrobeDropped,
LvDevice_ReverseX, LvDevice_ReverseY, LvDevice_RegionSelector, LvDevice_RegionMode,
LvDevice_RegionDestination, LvDevice_AcquisitionFrameCount, LvDevice_AcquisitionBurstFrameCount,
LvDevice_LvCustomID,
LvDevice_LvCustomInfo, LvDevice_LvCustomRegMode, LvDevice_LvCustomRegAddr, LvDevice_Lv↩
CustomRegData,
LvDevice_LvCustomRegMux, LvDevice_LinePitch, LvDevice_ChunkLvFrameAbort, LvDevice_ChunkLv↩
TriggerDropped,
LvDevice_ChunkLvTriggerError, LvDevice_ChunkLvEncoderPosition, LvDevice_ChunkLvEncoderRotation,
LvDevice_DeviceID,
LvDevice_DeviceType, LvDevice_GevDeviceIPAddress, LvDevice_GevDeviceSubnetMask, LvDevice_Gev↩
DeviceMACAddress,
LvDevice_GevDeviceGateway, LvDevice_LvGevDeviceStreamCaptureMode, LvDevice_StreamSelector,
LvDevice_StreamID,
LvDevice_DeviceEndianessMechanism, LvDevice_LvGevFindMaxPacketSize, LvDevice_LvGevPacket↩
SizeValue, LvDevice_LvGevTestPacketSize,
LvDevice_LvGevPacketSizeTestSuccess, LvDevice_LvGevCCTT, LvDevice_LvGevCCRC, LvDevice_LvC↩
CStatus,
LvDevice_LvDeviceDisplayName, LvDevice_LvDeviceIsAcquiring, LvDevice_LvUniProcessMode, Lv↩
Device_LvUniProcessEnableInPlace,
LvDevice_LvUniPixelFormat, LvDevice_LvUniProcessPayloadSize, LvDevice_LvUniLinePitch, LvDevice_↩
LvUniBayerDecoderAlgorithm,
LvDevice_LvUniBrightness, LvDevice_LvUniContrast, LvDevice_LvUniGamma, LvDevice_LvUniBalance↩
RatioSelector,
LvDevice_LvUniBalanceRatio, LvDevice_LvUniBalanceWhiteAuto, LvDevice_LvUniBalanceWhiteReset,
LvDevice_LvUniColorTransformationSelector,
LvDevice_LvUniColorTransformationEnable, LvDevice_LvUniColorTransformationValueSelector, LvDevice↩
_LvUniColorTransformationValue, LvDevice_LvUniSaturation,
LvDevice_LvUniProcessExecution, LvDevice_LvUniLUTMode, LvDevice_LvUniLUTSelector, LvDevice_Lv↩
UniLUTEnable,
LvDevice_LvUniLUTIndex, LvDevice_LvUniLUTValue, LvDevice_LvUniLUTValueAll, LvDevice_LvUniColor↩
TransformationMode,
LvDevice_LvDeviceExpiringDate, LvDevice_Info }

- enum LvStreamFtr {
LvStream_StreamID, LvStream_StreamAnnouncedBufferCount, LvStream_StreamAcquisitionMode↩
Selector, LvStream_StreamAnnounceBufferMinimum,
LvStream_StreamType, LvStream_LvStreamDisplayName, LvStream_LvCalcPayloadSize, LvStream_Lv↩
PostponeQueueBuffers,
LvStream_LvAwaitDeliveryLimit, LvStream_LvAutoAllocateProcessBuffers, LvStream_LvPreallocate↩
ProcessBuffers, LvStream_LvNumDelivered,
LvStream_LvNumUnderrun, LvStream_LvNumAnnounced, LvStream_LvNumQueued, LvStream_LvNum↩
AwaitDelivery,
LvStream_LvIsGrabbing, LvStream_LvNumAborted, LvStream_LvNumStarted, LvStream_Info }
- enum LvRendererFtr {
LvRenderer_LvAutoDisplay, LvRenderer_LvRenderType, LvRenderer_LvOffsetX, LvRenderer_LvOffsetY,
LvRenderer_LvWidth, LvRenderer_LvHeight, LvRenderer_LvIgnoreAspectRatio, LvRenderer_LvDisable↩

ScaleUp,
LvRenderer_LvDisableScaleDown, LvRenderer_LvCenterImage, LvRenderer_LvNumberOfTiles, LvRenderer_LvColumns,
LvRenderer_LvRows, LvRenderer_LvTileGap, LvRenderer_LvAutoTileCalculation, LvRenderer_Info }

- enum LvEventFtr { LvEvent_EventType, LvEvent_NumInQueue, LvEvent_NumFired }
- enum LvBufferFtr {
LvBuffer_Base, LvBuffer_Size, LvBuffer_UserPtr, LvBuffer_TimeStamp,
LvBuffer_NewData, LvBuffer_IsQueued, LvBuffer_IsAcquiring, LvBuffer_IsIncomplete,
LvBuffer_TlType, LvBuffer_SizeFilled, LvBuffer_Width, LvBuffer_Height,
LvBuffer_XOffset, LvBuffer_YOffset, LvBuffer_XPadding, LvBuffer_YPadding,
LvBuffer_FrameId, LvBuffer_ImagePresent, LvBuffer_ImageOffset, LvBuffer_PayloadType,
LvBuffer_PixelFormat, LvBuffer_PixelFormatNameSpace, LvBuffer_DeliveredImageHeight, LvBuffer_DeliveredChunkPayloadSize,
LvBuffer_ChunkLayoutId, LvBuffer_FileName, LvBuffer_UniBase, LvBuffer_UniSize,
LvBuffer_ProcessBase, LvBuffer_ProcessSize, LvBuffer_ExecProcess, LvBuffer_UniImageOffset }

5.29.1 Detailed Description

5.29.2 Enumeration Type Documentation

5.29.2.1 enum LvBufferFtr

LvBufferFtr constants.

Enumerator

LvBuffer_Base Represents the GenTL BUFFER_INFO_BASE info - Base address of the buffer memory. LvFtrType_Pointer.

LvBuffer_Size Represents the GenTL BUFFER_INFO_SIZE info - Size of the buffer in bytes. LvFtrType_Integer.

LvBuffer_UserPtr Represents the GenTL BUFFER_INFO_USER_PTR info - The user pointer (supplied by the application when the buffer was allocated). LvFtrType_Pointer. Note: This pointer should not be used in the C++ API and .Net Class Library, where this pointer is utilized internally for the LvBuffer class instance. The actual User pointer is available by the LvBuffer::GetUserPtr() function.

LvBuffer_TimeStamp Represents the GenTL BUFFER_INFO_TIMESTAMP info - Timestamp the buffer was acquired. The unit is device/implementation dependent. LvFtrType_Integer.

LvBuffer_NewData Represents the GenTL BUFFER_INFO_NEW_DATA info - Flag to indicate that the buffer contains new data since the last call. LvFtrType_Boolean.

LvBuffer_IsQueued Represents the GenTL BUFFER_INFO_IS_QUEUED info - Flag to indicate if the buffer is in the input pool or output queue. LvFtrType_Boolean.

LvBuffer_IsAcquiring Represents the GenTL BUFFER_INFO_ISACQUIRING info - Flag to indicate that the buffer is currently being filled with data. LvFtrType_Boolean.

LvBuffer_IsIncomplete Represents the GenTL BUFFER_INFO_ISINCOMPLETE info - Flag to indicate that a buffer was filled, but an error occurred during that process. LvFtrType_Boolean.

LvBuffer_TlType Represents the GenTL BUFFER_INFO_TLTYPE info - Transport layer technologies that are supported. LvFtrType_String.

LvBuffer_SizeFilled Represents the GenTL BUFFER_INFO_SIZE_FILLED info - Number of bytes written into the buffer last time it has been filled. This value is reset to 0 when the buffer is placed into the Input Buffer Pool. LvFtrType_Integer.

LvBuffer_Width Represents the GenTL 1.2 BUFFER_INFO_WIDTH info - Width of the data in the buffer in number of pixels. This information refers for example to the width entry in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_Height Represents the GenTL 1.2 BUFFER_INFO_HEIGHT info - Height of the data in the buffer in number of pixels as configured. For variable size images this is the max Height of the buffer. For example this information refers to the height entry in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_XOffset Represents the GenTL 1.2 BUFFER_INFO_XOFFSET info - XOffset of the data in the buffer in number of pixels from the image origin to handle areas of interest. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_YOffset Represents the GenTL 1.2 BUFFER_INFO_YOFFSET info - YOffset of the data in the buffer in number of lines from the image origin to handle areas of interest. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_XPadding Represents the GenTL 1.2 BUFFER_INFO_XPADDING info - XPadding of the data in the buffer in number of bytes. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_YPadding Represents the GenTL 1.2 BUFFER_INFO_YPADDING info - YPadding of the data in the buffer in number of bytes. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_FrameId Represents the GenTL 1.2 BUFFER_INFO_FRAMEID info - A sequentially incremented number of the frame. This information refers for example to the information provided in the GigE Vision image stream block id. The wrap around of this number is transportation technology dependent. For GigE Vision it is (so far) 16bit wrapping to 1. LvFtrType_Integer.

LvBuffer_ImagePresent Represents the GenTL 1.2 BUFFER_INFO_IMAGEPRESENT info - Flag to indicate if the current data in the buffer contains image data. This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType_Boolean.

LvBuffer_ImageOffset Represents the GenTL 1.2 BUFFER_INFO_IMAGEOFFSET info - Offset of the image data from the beginning of the delivered buffer in bytes. Applies for example when delivering the image as part of chunk data or on technologies requiring specific buffer alignment. LvFtrType_Integer.

LvBuffer_PayloadType Represents the GenTL 1.2 BUFFER_INFO_PAYLOADTYPE info - Payload type of the data. This information refers to the constants defined in GenTL PAYLOADTYPE_IDs (UNKNOWN=0, IMAGE=1, RAW_DATA=2, FILE=3, CHUNK_DATA=4, CUSTOM=1000) LvFtrType_Integer.

LvBuffer_PixelFormat Represents the GenTL 1.2 BUFFER_INFO_PIXELFORMAT info - This information refers for example to the information provided in the GigE Vision image stream data leader. The interpretation of the pixel format depends on the namespace the pixel format belongs to. This can be inquired using the LvBuffer_PixelFormatNameSpace feature. LvFtrType_Integer.

LvBuffer_PixelFormatNameSpace Represents the GenTL 1.2 BUFFER_INFO_PIXELFORMAT_NAMES↩PACE info - This information refers to the constants defined in GenTL 1.2 PIXELFORMAT_NAMESPAC↩E_IDs to allow interpretation of LvBuffer_PixelFormat (UNKNOWN=0, GEV=1, IIDC=2, CUSTOM=1000). LvFtrType_Integer.

LvBuffer_DeliveredImageHeight Represents the GenTL 1.2 BUFFER_INFO_DELIVERED_IMAGEHEIG↩HT info - The number of lines in the current buffer as delivered by the transport mechanism. For area scan type images this is usually the number of lines configured in the device. For variable size linescan images this number may be lower than the configured image height. This information refers for example to the information provided in the GigE Vision image stream data trailer. LvFtrType_Integer.

LvBuffer_DeliveredChunkPayloadSize Represents the GenTL 1.2 BUFFER_INFO_DELIVERED_CHUN↩KPAYLOADSIZE info - This information refers for example to the information provided in the GigE Vision image stream data leader. LvFtrType_Integer.

LvBuffer_ChunkLayoutId Represents the GenTL 1.2 BUFFER_INFO_CHUNKLAYOUTID info - This information refers for example to the information provided in the GigE Vision image stream data leader. The chunk layout id serves as an indicator that the chunk layout has changed and the application should reparse the chunk layout in the buffer. When a chunk layout (availability or position of individual chunks) changes since the last buffer delivered by the device through the same stream, the device MUST change the chunk layout id. As long as the chunk layout remains stable, the camera MUST keep the chunk layout id intact. When switching back to a layout, which was already used before, the camera can use the same id again or use a new id. A chunk layout id value of 0 is invalid. It is reserved for use by cameras not supporting the layout id functionality. LvFtrType_Integer.

LvBuffer_FileName Represents the GenTL 1.2 BUFFER_INFO_FILENAME info - This information refers for example to the information provided in the GigE Vision image stream data leader. For other technologies this is to be implemented accordingly. Since this is GigE Vision related information and the filename in GigE Vision is UTF8 coded, this filename is also UTF8 coded. LvFtrType_Integer.

LvBuffer_UniBase Unified base address of the buffer. If the image was processed to the output buffer, the pointer to the output buffer is returned, otherwise the pointer to the acquisition buffer is returned. This enables to write simple universal code for image handling. LvFtrType_Pointer. SynView feature.

LvBuffer_UniSize Size of the buffer returned on LvBuffer_UniBase call. LvFtrType_Integer. SynView feature.

LvBuffer_ProcessBase Pointer to the process buffer, attached to this acquisition buffer. LvFtrType_Pointer. SynView feature.

LvBuffer_ProcessSize Size of the process buffer, attached to this acquisition buffer. LvFtrType_Integer. SynView feature.

LvBuffer_ExecProcess Executes the SW image processing of the buffer. To be used when the LvDevice_LvUniProcessExecution is set to LvUniProcessExecution_OnExplicitRequest. LvFtrType_Command. SynView feature.

LvBuffer_UniImageOffset Unified image offset. If the image was processed to the output buffer, the image offset to the output buffer is returned, otherwise the image offset to the acquisition buffer is returned. This enables to write simple universal code for image handling. LvFtrType_Integer. SynView feature.

5.29.2.2 enum LvDeviceFtr

LvDeviceFtr constants.

Enumerator

LvDevice_DeviceVendorName Name of the manufacturer of the device. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceModelName Model name of the device. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceManufacturerInfo Manufacturer information about the device. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceVersion Version of the device. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceFirmwareVersion Version of the firmware loaded in the device. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvRecoveryFirmwareVersion String that indicates the version of the firmware and software to which the device would recover. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceSerialNumber Device identifier (serial number). LvFtrType_String. Note: This feature is called DeviceID in the SFNC, but we use rather the DeviceSerialNumber in order not to confuse it with the GenTL DeviceID, which is used for the device opening. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceUserID User-programmable device identifier. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSensorID Serial number of the sensor board. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvGrabberID Serial number of the grabber board. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceScanType Scan type of the sensor. LvFtrType_Enumeration. Values: LvDeviceScanType. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceRegistersStreamingStart Prepare the device for registers streaming without checking for consistency. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceRegistersStreamingEnd Announce the end of registers streaming. This will do a register set validation for consistency and activate it. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceRegistersCheck Perform the validation of the current register set for consistency. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceRegistersValid Reports if the current register set is valid and consistent. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceReset Resets the device and to put it in its power up state. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceClockSelector Selects a device clock frequency to be configured. LvFtrType_Enumeration. Values: LvDeviceClockSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceClockFrequency Frequency of the selected clock in Hz. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceTemperatureSelector Selects the location within the device, where the temperature will be measured. LvFtrType_Enumeration. Values: LvDeviceTemperatureSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceTemperature Current temperature at the selected location in degrees of Celcius LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvDeviceUpTime Current up-time of the device in milliseconds. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvDeviceType String that indicates the basic type of the device. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_SensorWidth Effective width of the sensor in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_SensorHeight Effective height of the sensor in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_WidthMax Maximum width of the image in pixels. The dimension is calculated after applying horizontal binning, decimation or readout width. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_HeightMax Maximum height of the image in pixels. The dimension is calculated after applying vertical binning, decimation or readout height. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_Width Image width provided by the device in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_Height Image height provided by the device in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_OffsetX Horizontal offset from the origin of the AOI (area of interest) in pixels. The AOI is applied to the result of binning and or decimation. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_OffsetY Vertical offset from the origin of the AOI (area of interest) in pixels. The AOI is applied to the result of binning and or decimation. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_PixelFormat Pixel format provided by the device. The feature combines pixel coding, size and color filter attributes. LvFtrType_Enumeration. Values: see LvPixelFormat. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BinningHorizontal Horizontal binning, number of horizontal pixels to combine together. This increases the intensity (and S/N ratio) of the pixels and reduces the horizontal resolution (width) of the image. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BinningVertical Vertical binning, number of vertical pixels to combine together. This increases the intensity (and S/N ratio) of the pixels and reduces the vertical resolution (height) of the image. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DecimationHorizontal Horizontal decimation (sub-sampling) of the image. This reduces the horizontal resolution (width) of the image by the specified factor. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DecimationVertical Vertical decimation (sub-sampling) of the image. This reduces the vertical resolution (height) of the image by the specified factor. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvAOIMode Selects the mode of controlling the area of interest LvFtrType_Enumeration. Values: LvAOIMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvReadoutWidth Width of the sensor-side area of interest in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvReadoutHeight Height of the sensor-side area of interest in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvReadoutOffsetX X offset (left offset) for the sensor-side area of interest in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvReadoutOffsetY Y offset (top offset) for the sensor-side area of interest in pixels. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvVariablePayloadSize This flag controls, whether the payload size can change during acquisition. When set, the image dimensions and other paramenters can vary during acquisition. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_AcquisitionMode Sets the acquisition mode of the device. It defines mainly the number of frames to capture during an acquisition and the way the acquisition stops. LvFtrType_Enumeration. Values: LvAcquisitionMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerSelector Selects the type of trigger to configure. LvFtrType_Enumeration. Values: LvTriggerSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerMode Controls if the selected trigger is active. LvFtrType_Enumeration. Values: LvTriggerMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerSoftware Generates a software trigger when trigger source is set to 'software' or any physical line. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerSource Specifies the internal signal or physical input line to use as the trigger source. LvFtrType_Enumeration. Values: LvTriggerSource. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerActivation Activation mode of the trigger - specifies which edge of the signal is active. LvFtrType_Enumeration. Values: LvTriggerActivation. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerDelay Trigger delay in microseconds, specifies a delay introduced between the trigger reception and its actual activation. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TriggerDivider Used to divide the number of incoming trigger pulses by an integer factor. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvTriggerCaching Sets the caching mode for the selected trigger. The feature controls how early triggers are treated by the device. LvFtrType_Enumeration. Values: LvTriggerCaching. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ExposureMode Controls the exposure (shutter) mode applied for each acquisition. LvFtrType_Enumeration. Values: LvExposureMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLongRangeExposureMode Switches to mode with wider range of exposure times, but slightly higher jitter. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvGlobalResetMode Switches to mode with wider range of exposure times, but slightly higher jitter. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ExposureTime Exposure time in microseconds. The feature controls how long are the pixels exposed to illumination. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ExposureAuto Selects the automatic exposure mode. LvFtrType_Enumeration. Values: LvExposureAuto. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvAcquisitionFrameRateControlMode Switches the acquisition frame rate control on or off. The camera might internally switch to different working mode, which can decrease the maximum frame rate. LvFtrType_Enumeration. Values: LvAcquisitionFrameRateControlMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_AcquisitionFrameRate Acquisition frame rate in frames per second (Hz) - the frequency with which the image frames are captured. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineSelector Selects the I/O line for querying and configuration. Note that to use given line to drive a device feature (trigger, counter, etc.), source of the given feature has to refer to the line. LvFtrType_Enumeration. Values: LvLineSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineMode Line mode - controls, whether given line is used as signal input or output. LvFtrType_Enumeration. Values: LvLineMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineFormat This feature controls the current electrical format of the selected physical input or output Line. LvFtrType_Enumeration. Values: LvLineFormat. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineSource Selects a device internal signal that should drive the output signal of the selected line. LineMode must be Output. Not applicable for input lines. LvFtrType_Enumeration. Values: LvLineSource. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineInverter Inverts the signal output on the selected line. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineStatus Reports the current status of the selected line. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LineStatusAll Bit field indicating status of all i/o lines. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserOutputSelector Selects the user output for querying and configuration. LvFtrType_Enumeration. Values: LvUserOutputSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserOutputValue Reports the current status of the selected user output. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserOutputValueAll Bit field indicating status of all user outputs. Only the bits defined in the User Output Value All Mask are used, the others are ignored. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserOutputValueAllMask Mask for the User Output Value All bitfield - defines which bits are used to change a user output value and which are ignored. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_CounterSelector Selects which counter to configure. LvFtrType_Enumeration. Values: LvCounterSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCounterMode Selects working mode of the selected counter. LvFtrType_Enumeration. Values: LvCounterMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_CounterEventSource Internal device signal incrementing the selected counter. LvFtrType_Enumeration. Values: LvCounterEventSource. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_CounterReset This command resets the selected counter LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_CounterValue Reads or sets the current value of the selected counter. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_CounterDuration Duration (or number of events) before the counter end event is generated and the counter expires. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TimerSelector Selects which timer to configure. LvFtrType_Enumeration. Values: LvTimerSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TimerDuration Sets the duration (in microseconds) of the timer active pulse. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TimerDelay Sets the delay (in microseconds) applied between activating the timer and issuing the timer active signal. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_TimerTriggerSource Internal device signal activating the selected timer. LvFtrType_Enumeration. Values: LvTimerTriggerSource. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSpecialPurposeTriggerSelector Selects the special purpose trigger type to configure. LvFtrType_Enumeration. Values: LvSpecialPurposeTriggerSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSpecialPurposeTriggerSource Specifies the internal signal or physical input line to use as the trigger source. LvFtrType_Enumeration. Values: LvSpecialPurposeTriggerSource. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSpecialPurposeTriggerActivation Activation mode of the trigger - specifies which edge of the signal is active. LvFtrType_Enumeration. Values: LvSpecialPurposeTriggerActivation. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSpecialPurposeTriggerSoftware Generates a software trigger for the selected trigger action when trigger source is set to 'software' or any physical line. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvImageStampsResetMask A single bitfield that selects which features will be reset by the timestamp reset trigger in one access. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvImageStampSelector Selects an image stamp type for configuration. LvFtrType_Enumeration. Values: LvImageStampSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvImageStampResetEnable Enables/disables the reset trigger functionality for the selected image stamp type. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvBootSwitch Selects the firmware type to load on next boot. LvFtrType_Enumeration. Values: LvBootSwitch. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvBayerDecoderAlgorithm Selects the algorithm used by the Bayer decoder. LvFtrType_↩Enumeration. Values: LvBayerDecoderAlgorithm. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvBayerDecoderThreshold Sets the threshold controlling the performance of the variable gradient Bayer decoder algorithm. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvWatchdogEnable Enables the watchdog reset function. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvWatchdogTimerDuration When watchdog is enabled, the device reboots when the timeout specified expires. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvWatchdogTimerReset Resets the watchdog timer, the watchdog starts counting the specified timeout again. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvWatchdogFailed Signals that the last device reboot was initiated by the watchdog function. After reading, reset this flag explicitly, it wouldn't be affected by a 'warm' system reboot. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GainSelector Selects which gain type to configure. LvFtrType_Enumeration. Values: LvGain↩Selector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_Gain Gain value for the selected gain type in dB. This is an amplification factor applied to the video signal. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GainAuto Controls the automatic gain control (AGC) mode. LvFtrType_Enumeration. Values: Lv↩GainAuto. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BlackLevelSelector Selects which black level type to configure. LvFtrType_Enumeration. Values: LvBlackLevelSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BlackLevel Controls the analog black level. This represents a DC offset applied to the video signal. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BlackLevelAuto Controls the automatic black level mode. LvFtrType_Enumeration. Values: Lv↩BlackLevelAuto. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ColorTransformationSelector Selects which color transformation module is controlled by the color transformation features. It also gives particular meaning to individual color transformation gains. LvFtrType_Enumeration. Values: LvColorTransformationSelector. Device remote feature (LvFtrGroup_↩DeviceRemote).

LvDevice_ColorTransformationEnable Activates the selected Color Transformation module. LvFtrType_↩Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ColorTransformationValueSelector Selects the gain factor or offset of the transformation matrix to configure LvFtrType_Enumeration. Values: LvColorTransformationValueSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ColorTransformationValue Value of the selected color transformation matrix entry. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvExternalDeviceControlMode Selects the operation mode of external device control. LvFtrType_Enumeration. Values: LvExternalDeviceControlMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvExternalADCSelector Selects the external ADC to configure. LvFtrType_Enumeration. Values: LvExternalADCSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvExternalADCValue Reads the value of the selected external ADC. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchCurrentAction Reports the automated action currently performed by a power switch. LvFtrType_Enumeration. Values: LvPowerSwitchCurrentAction. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchSelector Selects the power switch to configure. LvFtrType_Enumeration. Values: LvPowerSwitchSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchBoundADC Sets an external ADC to the selected power switch. The bound pair will work together during the automatic operation. LvFtrType_Enumeration. Values: LvPowerSwitchBoundADC. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchDrive Drives the selected power switch with desired polarity. LvFtrType_Enumeration. Values: LvPowerSwitchDrive. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchPulsePlus Pulses the selected power switch with plus polarity. Available in the automatic operation mode. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchPulseMinus Pulses the selected power switch with minus polarity. Available in the automatic operation mode. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlCalibrate Starts an automatic calibration on the selected power switch and bounded ADCs. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlMinusEnd Represents the calibrated minimal ADC achievable by driving the power switch's with minus polarity (plus if the polarity is inverted). LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlPlusEnd Represents the calibrated maximal ADC achievable by driving the power switch's with plus polarity (minus if the polarity is inverted). LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlPulsePeriod Represents the calibrated slow motion pulse period for the selected power switch, in microsesonds. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlDutyCycle Represents the calibrated slow motion duty cycle for the selected power switch (in %). Defines how much of the pulse period is the power switch actually active. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlTargetApproach Selects how the target lens position should be approached. LvFtrType_Enumeration. Values: LvLensControlTargetApproach. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlNrSlowSteps Sets the number of slow steps required before reaching the target position LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlTargetPosition Sets the target position (value) of the ADC bound to the selected power switch LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlAdjustPosition Adjusts the required target position (value) of the ADC bound to the selected power switch. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvPowerSwitchPulseDuration Duration (in microseconds) of the pulses issued at the power switch. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlMinCalibrationRange Minimum value range that has to be reached on the external ADC to count the calibration as valid. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlCalibrateAll Starts an automatic calibration on the active power switches and bounded ADCs. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LUTSelector Selects which LUT to configure. LvFtrType_Enumeration. Values: LvLUTSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LUTEnable Activates the selected LUT.. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LUTIndex Index of the element to access in the selected LUT LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LUTValue Value of the element for the current index in the selected LUT. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LUTValueAll This register accesses the entire content of the selected LUT in one chunk access. LvFtrType_Buffer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_PayloadSize Provides the number of bytes transferred for each image by the device, including image and chunk data. The value defines the required size of the target buffer used for acquisition. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevVersionMajor Major version of the GigE Vision specification implemented by the device. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevVersionMinor Minor version of the GigE Vision specification implemented by the device. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevDeviceModeIsBigEndian Endianess of the device registers. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevDeviceModeCharacterSet Character set used by all the strings of the device registers. LvFtrType_Enumeration. Values: LvGevDeviceModeCharacterSet. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevInterfaceSelector Selects which physical network interface to control. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMACAddress MAC address of the network interface. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSupportedOptionSelector Selects the GEV option to interrogate for existing support. LvFtrType_Enumeration. Values: LvGevSupportedOptionSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSupportedOption Returns if the selected GEV option is supported. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCurrentIPConfigurationLLA Indicates if Link Local Address IP configuration scheme is activated on the given network interface. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCurrentIPConfigurationDHCP Indicates if DHCP IP configuration scheme is activated on the given network interface. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCurrentIPConfigurationPersistentIP Indicates if persistent IP configuration scheme is activated on the given network interface. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCurrentIPAddress Reports the IP address for the given network interface. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCurrentSubnetMask Provides the subnet mask of the given interface. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCurrentDefaultGateway Indicates the default gateway IP address to be used on the given network interface. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevPersistentIPAddress Indicates the persistent IP address for this network interface. It is only used when the device boots with the persistent IP configuration scheme. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevPersistentSubnetMask Indicates the persistent subnet mask associated with the persistent IP address on this network interface. It is only used when the device boots with the Persistent IP configuration scheme. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevPersistentDefaultGateway Indicates the persistent default gateway for this network interface. It is only used when the device boots with the persistent IP configuration scheme. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevNumberOfInterfaces Indicates the number of physical network interfaces supported by this device. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMessageChannelCount Indicates the number of message channels supported by this device. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevStreamChannelCount Indicates the number of stream channels supported by this device. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevHeartbeatTimeout Indicates the current heartbeat timeout in milliseconds. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevTimestampTickFrequency Indicates the number of timestamp ticks during 1 second (frequency in Hz). LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevTimestampControlLatch Latches current timestamp counter into GevTimestampValue. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevTimestampControlReset Resets the Timestamp counter to 0. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevTimestampControlLatchReset Reset and latch in a single command. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevTimestampValue Returns the latched 64-bit value of the timestamp counter. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevCCP Controls the device access privilege of an application. LvFtrType_Enumeration. Values: LvGevCCP. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevStreamChannelSelector Selects the stream channel to control. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPInterfaceIndex Index of network interface to use. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPHostPort Indicates the port to which the device must send data stream. Setting this value to 0 closes the stream channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPSFireTestPacket Sends a test packet. When this feature is set, the device will fire one test packet. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPSDoNotFragment The state of this feature is copied into the "do not fragment" bit of IP header of each stream packet. It can be used by the application to prevent IP fragmentation of packets on the stream channel. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPSBigEndian Specifies the stream packet size in bytes to send on this channel. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPSPacketSize Specifies the stream packet size in bytes to send on this channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPD Indicates the delay (in timestamp counter unit, which is currently a microsecond) to insert between each packet for this stream channel. This can be used as a crude flow-control mechanism if the application or the network infrastructure cannot keep up with the packets coming from the device. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCDA Indicates the destination IP address for this stream channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevLinkSpeed Indicates the speed of transmission negotiated by the given network interface in Mb/s. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserSetSelector Selects the feature configuration user set to load, save or configure. LvFtrType_Enumeration. Values: LvUserSetSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserSetLoad Loads the selected user configuration set and makes it active LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserSetSave Saves the current device configuration into the selected user configuration set. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_UserSetDefaultSelector Selects the default feature configuration set to be loaded and activated upon camera boot or reset. LvFtrType_Enumeration. Values: LvUserSetDefaultSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkModeActive Activates the chunk mode, ie. inclusion of chunk data in the payload data. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkSelector Selects the chunk to configure. LvFtrType_Enumeration. Values: LvChunkSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkEnable Enables the inclusion of the selected chunk in the payload data. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkOffsetX X offset applied the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkOffsetY Y offset applied the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkWidth Width of the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkHeight Height of the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkPixelFormat Pixel format of the image included in the payload. LvFtrType_Enumeration. Values: see LvPixelFormat. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLinePitch Line pitch of the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkFrameID Frame id of the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkTimestamp Timestamp associated with the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkExposureTime Exposure time used to acquire the image included in the payload. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkGainSelector Selects the gain type to be reported in chunk data. LvFtrType_Enumeration. Values: LvChunkGainSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkGain Gain used to acquire the image included in the payload. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkBlackLevel Black level used to acquire the image included in the payload. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLineStatusAll Bit field indicating staut of all i/o lines at the time the image included in the payload was acquired. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvExternalADCSelector Selects the external ADC to be reported in chunk data. LvFtrType_Enumeration. Values: LvChunkLvExternalADCSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvExternalADCValue Reads the value of the selected external ADC at time of acquisition of the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventSelector Selects which event to signal to the host application. LvFtrType_Enumeration. Values: LvEventSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventNotification Activate or deactivate the notification to the host application of the selected event occurrence. LvFtrType_Enumeration. Values: LvEventNotification. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppID ID string the smart application LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt1 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt2 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt3 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt4 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt5 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt6 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt7 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt8 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt9 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt10 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt11 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt12 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt13 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt14 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt15 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt16 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt17 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt18 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt19 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt20 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt21 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt22 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt23 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt24 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt25 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt26 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt27 Generic signed integer register controlling a smart application. LvFtrType_↩ Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt28 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt29 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt30 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt31 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppInt32 Generic signed integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint1 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint2 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint3 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint4 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint5 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint6 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint7 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint8 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint9 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint10 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint11 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint12 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint13 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint14 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint15 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint16 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint17 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint18 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint19 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint20 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint21 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint22 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint23 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint24 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint25 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint26 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint27 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint28 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint29 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint30 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint31 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppUint32 Generic unsigned integer register controlling a smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppAsciiCmdString Characters of the ASCII command for the smart application. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppAsciiCmdExecute Executes the ASCII command for the smart application. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppAsciiCmdFeedback Response to the ASCII command for the smart application. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppAsciiCmdRetCode Numeric return code of the ASCII command for the smart application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppPath Path of the smart application to be started LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppStart Starts the smart application defined by the path. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvLog Returns the unique identifier of the log type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvLogTimestamp Returns the timestamp of the log event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvLogMessage The log message coming with the event LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppLog Returns the unique identifier of the smart application log type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppLogTimestamp Returns the timestamp of the smart application Smart Application Log Event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppLogMessage The smart application log message coming with the event. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortBaudRate Baud rate used for the serial port communication. LvFtrType_Enumeration. Values: LvSerialPortBaudRate. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortParity Parity used for the serial port communication. LvFtrType_Enumeration. Values: LvSerialPortParity. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortDataBits Data bits per character for the serial port communication. LvFtrType_Enumeration. Values: LvSerialPortDataBits. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortStopBits Stop bits per character for the serial port communication. LvFtrType_Enumeration. Values: LvSerialPortStopBits. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortTimeout Timeout value used to finish waiting for command response LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortEOTMarker Short string (or single character) marking end of transmission. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortMaxResponseLength Maximal expected length of the command response. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortCommandString String of the ASCII command to be sent over the serial port. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortCommandSend Sends the ASCII command over the serial port Command. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortCommandResponse Response to the ASCII command sent over the serial port StringReg. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSerialPortCommandStatus Status code indicating success of the last command. Values: LvSerialPortCommandStatus. LvFtrType_Enumeration. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvSmartAppExitEvent Sends an exit event to the running smart application. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvWatchdogTimerValue Current watchdog timer value - reports the current value, after which the timer expires and the device reboots. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlInvertedPolarity Indicates if the lens is wired with inverted polarity, meaning that driving the power switch to the plus side decreases the external ADC feedback. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMCPHostPort Controls the port to which the device must send messages. Setting this value to 0 closes the message channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMCDA Controls the destination IP address for the message channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMCTT Provides the transmission timeout value in milliseconds. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMCRC Controls the number of retransmissions allowed when a message channel message times out. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvSmartAppString The smart application string related to the delivered payload. LvFtrType_String Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvSmartAppIntSelector Selects one of the signed integer values related to the delivered payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvSmartAppInt The selected smart application signed integer related to the delivered payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvSmartAppUintSelector Selects one of the unsigned integer values related to the delivered payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvSmartAppUint The selected smart application unsigned integer related to the delivered payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvSmartAppRegister The smart application raw register related to the delivered payload. LvFtrType_Buffer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppString Returns the unique identifier of the smart application string type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppStringTimestamp Returns the timestamp of the smart application string event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppStringValue The smart application string value coming with the event. LvFtr↩Type_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppInt Returns the unique identifier of the smart application signed integer type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppIntTimestamp Returns the timestamp of the smart application signed integer event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppIntSelector Selects one of the signed integer values coming with the event. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppIntValue Value of the selected signed integer coming with the event. LvFtr↩Type_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppUint Returns the unique identifier of the smart application unsigned integer type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppUintTimestamp Returns the timestamp of the smart application unsigned integer event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppUintSelector Selects one of the unsigned integer values coming with the event. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppUintValue Value of the selected unsigned integer coming with the event. Lv↩FtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppRegister Returns the unique identifier of the smart application raw register type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppRegisterTimestamp Returns the timestamp of the smart application raw register event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvSmartAppRegisterValue The smart application raw register value coming with the event Register. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceSFNCVersionMajor Major version of the Standard Feature Naming Convention that was used to create the device's XML. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceSFNCVersionMinor Minor version of the Standard Feature Naming Convention that was used to create the device's XML. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceSFNCVersionSubMinor Sub-minor version of Standard Feature Naming Convention that was used to create the device's XML. LvFtrType_Integer. Device remote feature (LvFtrGroup_Device↩Remote).

LvDevice_LvLineDebounceDuration Sets the duration (in microseconds) of the line debounce period. Value of 0.0 switches the debouncer off. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ActionDeviceKey Provides the device key that allows the device to check the validity of action commands. The device internal assertion of an action signal is only authorized if the ActionDeviceKey and the action device key value in the protocol message are equal. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ActionSelector Selects to which action signal further action settings apply. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ActionGroupKey Provides the key that the device will use to validate the action on reception of the action protocol message. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ActionGroupMask Provides the mask that the device will use to validate the action on reception of the action protocol message. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLensControlCalibrationStatus Reports current calibration status of the selected power switch and its bound ADC. The status is computed from the current ADC range, no matter if it is a result of calibration operation or configured manually. LvFtrType_Enumeration. Values: LvLensControlCalibrationStatus. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLUTMode Selects the LUT control mode. LvFtrType_Enumeration. Values: LvLUTMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BalanceRatioSelector Selects which color channel to configure for white-balancing. LvFtrType_Enumeration. Values: LvBalanceRatioSelector. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BalanceRatio Controls white balance ratio coefficient to be applied on the selected color channel. Note that the white balance functionality is implemented using the LUT. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_BalanceWhiteAuto Controls the mode for automatic white balancing between the color channels. The white balancing ratios are automatically adjusted. Note that the white balance functionality is implemented using the LUT. LvFtrType_Enumeration. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevDeviceClass Returns the class of the device. LvFtrType_Enumeration. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevIPConfigurationStatus Reports the current IP configuration status, ie. the method through which the network interface was configured. LvFtrType_Enumeration. Values: LvGevIPConfigurationStatus. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevDiscoveryAckDelay Indicates the maximum randomized delay the device will wait to acknowledge a discovery command. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevGVCPExtendedStatusCodes Enables the generation of extended status codes. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevGVCPPendingAck Enables the generation of PENDING_ACK. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevGVCPHeartbeatDisable Disables the GVCP heartbeat. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevGVCPPendingTimeout Indicates the longest GVCP command execution time before a device returns a PENDING_ACK. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevPrimaryApplicationSwitchoverKey Controls the key to use to authenticate primary application switchover requests. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevPrimaryApplicationSocket Returns the UDP source port of the primary application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevPrimaryApplicationIPAddress Returns the address of the primary application. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevMCSP This feature indicates the source port for the message channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCCFGUnconditionalStreaming Enables the camera to continue to stream, for this stream channel, if its control channel is closed or regardless of the reception of any ICMP messages (such as destination unreachable messages). LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCCFGExtendedChunkData Enables cameras to use the extended chunk data payload type for this stream channel. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCPDirection Reports the direction of the stream channel. LvFtrType_Enumeration. Values: LvGevSCPDirection Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_GevSCSP Indicates the source port of the stream channel. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvTriggerDelayed Flag indicating if the trigger was delayed when acquiring the image included in the payload. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvTriggerDropped Returns the unique identifier of the dropped trigger type of event. This feature can be used to register a callback function to be notified of the event occurrence. Its value uniquely identifies the type of event that will be received. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_EventLvTriggerDroppedTimestamp Returns the timestamp of the dropped trigger event. It can be used to determine precisely when the event occured. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvStrobeEnable Selects the LED clusters of the strobe light that should be enabled. LvFtrType_Enumeration. Values: LvStrobeEnable. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvStrobeDurationMode Controls the way how the maximum time of strobe duration is calculated. LvFtrType_Enumeration. Values: LvStrobeDurationMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvStrobeDuration Duration of the strobe pulse in usec. The maximum time depends on the setting of Strobe Duration Mode feature. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvStrobeDelay A delay before the strobe pulse starts after frame trigger is applied in usec. LvFtrType_Float. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvStrobeBrightness Brighness (in %) of the strobe light. Allows to lower the full brighness of the strobe. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvStrobeDropMode Sets mode of handling of strobes not matching the device hardware constraints. If a strobe is required (activated by a frame trigger) before the strobe device is ready, the strobe must be dropped or delayed. LvFtrType_Enumeration. Values: LvStrobeDropMode. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvLUTReset Resets the LUT settings. LvFtrType_Command. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvStrobeDropped Flag indicating if the configured strobe was dropped when acquiring the image included in the payload. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ReverseX Flip horizontally the image sent by the device. The AOI is applied after the flipping. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ReverseY Flip vertically the image sent by the device. The AOI is applied after the flipping. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_RegionSelector Selects the region of interest to control. The RegionSelector feature allows devices that are able to extract multiple regions out of an image, to configure the features of those individual regions independently. LvFtrType_Enumeration. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_RegionMode Controls if the selected Region of interest is active and streaming. LvFtrType_Enumeration. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_RegionDestination Control the destination of the selected region. LvFtrType_Enumeration. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_AcquisitionFrameCount Number of frames to acquire in MultiFrame Acquisition mode. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_AcquisitionBurstFrameCount Number of frames to acquire for each FrameBurstStart trigger. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCustomID Revision number of the custom module. LvFtrType_String. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCustomInfo Info register in the custom module. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCustomRegMode Controls the way of addressing a register in the custom module. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCustomRegAddr Defines the address of a register in the custom module. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCustomRegData Transfers data to and from a register in the custom module. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LvCustomRegMux Defines the address and transfers data to and from a register in the custom module. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_LinePitch Total number of bytes between 2 successive lines. This feature is used to facilitate alignment of image data. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvFrameAbort Flag indicating if a frame was dropped when acquiring the image included in the payload. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvTriggerDropped Flag indicating if a trigger was dropped when acquiring the image included in the payload. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvTriggerError Flag indicating if a missmatch between trigger and sensor data was detected when acquiring the image included in the payload. LvFtrType_Boolean. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvEncoderPosition Encoder position generating the trigger for the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_ChunkLvEncoderRotation Encoder rotation generating the trigger for the image included in the payload. LvFtrType_Integer. Device remote feature (LvFtrGroup_DeviceRemote).

LvDevice_DeviceID Device ID. LvFtrType_String. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_DeviceType Device type. LvFtrType_Enumeration. Values: LvDeviceType. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_GevDeviceIPAddress Device IP address. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_GevDeviceSubnetMask Device subnet mask. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_GevDeviceMACAddress Device MAC address. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_GevDeviceGateway Device gateway. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevDeviceStreamCaptureMode Device stream capture mode. Controls, how the GVSP (image) stream is processed by the host machinge. LvFtrType_Enumeration. Values: LvGevDeviceStreamCaptureMode. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_StreamSelector Stream selector. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_StreamID Stream ID. LvFtrType_String. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_DeviceEndianessMechanism Identifies the endianess mode. This feature should be set to 'standard' for all GigE Vision remote devices based on GenICam schema version 1.1 (and newer). The value 'legacy' is intended for devices using GenICam schema version 1.0. Values: LvDeviceEndianessMechanism. LvFtrType_Enumeration. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevFindMaxPacketSize Determine the maximal usable packet size for streaming. The procedure is already applied automatically when opening the device. Do not use during active streaming. LvFtrType_Command. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevPacketSizeValue Streaming packet size to be verified using the Test Packet Size command. Do not use during active streaming. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevTestPacketSize Test if the selected packet size is suitable for streaming in current network configuration. LvFtrType_Command. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevPacketSizeTestSuccess Reports success of the last packet size test command. LvFtr↩ Type_Boolean. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevCCTT Controls the GigE Vision control channel transmission timeout value in milliseconds. When it expires, the acknowledge from the device will be considered as missing and the command eventually sent again. LvFtrType_Command. Device GenTL feature (LvFtrGroup_DeviceGtl).

LvDevice_LvGevCCRC Controls the number of GigE Vision control channel command retransmissions allowed when a control channel command times out. LvFtrType_Command. Device GenTL feature (LvFtr↩ Group_DeviceGtl).

LvDevice_LvCCStatus Control channel status. LvFtrType_Integer. Device GenTL feature (LvFtrGroup_↩ DeviceGtl).

LvDevice_LvDeviceDisplayName User readable name of the device. LvFtrType_String. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvDeviceIsAcquiring Returns true if the acquisition was started. Note that this feature does not tell whether the images are actually delivered to the output buffer queue; it simply informs that your application is between the AcquisitionStart and AcquisitionStop actions. LvFtrType_Boolean. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniProcessMode The UniProcessing provides unified API for image preprocessing, which is done either on the device itself, if it is supported by the hardware, or by software, if not. The preprocessing includes Bayer decoding or pixel format conversion, application of LUT and Color Correction. LvFtrType↩ _Enumeration. Values: LvUniProcessMode. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniProcessEnableInPlace If possible, the software image preprocessing will be preferably done in the same image (not to another buffer). This is possible only in case the preprocessing does not change the pixel format, that means the LvUniPixelFormat must be equal to PixelFormat (for example the Bayer decoding is not done by software). LvFtrType_Boolean. Device local feature (LvFtrGroup_↩ DeviceLocal).

LvDevice_LvUniPixelFormat If the image preprocessing is enabled, this is the desired pixel format, to which the image is to be converted. Only monochrome and RGB/BGR color pixel formats are supported. The processing chain is set so that:

- if the PixelFormat is undecoded Bayer, the Bayer decoding to desired LvUniPixelFormat is included
- otherwise if the PixelFormat is not equal to LvUniPixelFormat, a pixel format conversion is included. LvFtrType_Enumeration. Values: see LvPixelFormat. Device local feature (LvFtrGroup_Device↩ Local).

LvDevice_LvUniProcessPayloadSize Returns the size needed for the processing output buffer, which is to be used when the in-place processing is not enabled or not possible. Normally is this buffer allocated automatically for each acquisition buffer, but your application can also provide your own buffers and this feature gives the size of the buffers needed. LvFtrType_Integer. Device local feature (LvFtrGroup_↩ DeviceLocal).

LvDevice_LvUniLinePitch The line increment of the process buffer, if the processing is active, or of the source buffer, if processing is not active. To access the image regardless whether it was processed to the process buffer or not, you need 5 independent values:

- Pointer to the data - use LvUniBase feature of the Buffer, which points either to the acquired image (if no processing was done), or to the processed image (if it was processed).
- Width and height - these are the same for the acquired and processed image, so use the Width and Height from the remote device, or ChunkWidth and ChunkHeight if these can change during the acquisition.
- Pixel format - use LvUniPixelFormat - if this is different from the PixelFormat then processing is done, so LvUniPixelFormat is always correct.
- Line pitch - use LvUniLinePitch, which returns proper line pitch of the buffer, to which the LvUniBase pointer points. Note that the above is valid only in case the processing can be successfully done (for example the source image is not in unsupported PixelFormat) and is not disabled (for example by LvUniProcessExecution=OnExplicitRequest). LvFtrType_Integer. Device local feature (LvFtrGroup↩ _DeviceLocal).

LvDevice_LvUniBayerDecoderAlgorithm Selects the Bayer array decoding method for the software processing. This does not apply to the hardware Bayer decoding on the device, which is usually fixed to one method. LvFtrType_Enumeration. Values: LvBayerDecoderAlgorithm. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniBrightness Brightness of the image. It is realized by the LUT. Values under 1.0 means darker than original, above 1.0 lighter than the original. The LvUniLUTMode must be Generated, in order to enable this feature. LvFtrType_Float. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniContrast Contrast of the image. It is realized by the LUT. Values under 1.0 means lower contrast than original, above 1.0 higher contrast than the original. The LvUniLUTMode must be Generated, in order to enable this feature. LvFtrType_Float. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniGamma Gamma correction of the image. It is realized by the LUT. Values under 1.0 make the middle tones darker, above 1.0 lighter. The LvUniLUTMode must be Generated, in order to enable this feature. LvFtrType_Float. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniBalanceRatioSelector Selects which color channel will be accessed by the LvUniBalanceRatio feature. The LvUniLUTMode must be Generated, in order to enable this feature. LvFtrType_Enumeration. Values: LvUniBalanceRatioSelector. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniBalanceRatio The white balance factor to be applied on the selected color channel. The selected color channel of all pixels will be multiplied by this value (not directly, but through the precalucated LUT). If the value is < 1.0, the saturated pixels will become gray (white is no more white). Thus it is better if all 3 factors are greater than or equal to 1.0. LvFtrType_Float. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniBalanceWhiteAuto Selects the action for automatic white balance calculation. Currently only the option Once is available. Setting this option causes the following:

- if there is already acquired image available, the white balance factors are calculated from this image and LUT is updated to reflect the changes
- if there is no image acquired yet, an internal flag is set and the calculation is done when the image is acquired. Note that the enumeration is self-clearing, that means its value is automatically changed to Off, when the white balance calculation is finished. The newly calculated white balance is applied to to newly acquired images, not to the existing ones, unless you explicitly call the ExecProcess command for the already acquired buffers. At the time of calculation the camera should look at a neutral grey (not white) object, which should fill the whole image area. Making white balance from normal image can bring less satisfactory results. LvFtrType_Enumeration. Values: LvUniBalanceWhiteAuto. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniBalanceWhiteReset Sets all the white balance factors (LvUniBalanceRatio) to 1. The advantage of this feature in comparison with setting the 3 factors to 1 is that the LUT is updated only once, so it is faster. LvFtrType_Command. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniColorTransformationSelector Selects which color transformation module is controlled by the color transformation features. It also gives particular meaning to individual color transformation gains. LvFtrType_Enumeration. Values: LvUniColorTransformationSelector. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniColorTransformationEnable Enables the Color Transformation in the processing. When disabled, the Color Transformation matrix does not lose its values; when enabling it, the original values are retained. LvFtrType_Boolean. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniColorTransformationValueSelector Selects the cell of the Color Transformation matrix to be accessed by LvUniColorTransformationValue. LvFtrType_Enumeration. Values: LvColorTransformationValueSelector. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniColorTransformationValue The value of the selected cell of the Color Transformation matrix. LvFtrType_Float. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniSaturation Sets the Color Correction matrix according to specified saturation. The saturation set to 0 causes a conversion to greyscale, 1.0 leaves the image identical, 2.0 emphasizes the colors. LvFtrType_Float. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniProcessExecution Defines the point, when the software image processing of the buffer is done. You may need to define this point in case you do not need all the images to be processed. Note

that this applies only to the software processing; the hardware processing is done on the remote device always. LvFtrType_Enumeration. Values: LvUniProcessExecution. Device local feature (LvFtrGroup_↩ DeviceLocal).

LvDevice_LvUniLUTMode Selects the LUT control mode. The mode determines, if the LUT can be directly modified by the application, or if the LUT is to be reserved for implementation of white balance, gamma, brightnesss and contrast - in such case the LUT is filled with precalculated values by SynView library and cannot be directly modified. LvFtrType_Enumeration. Values: LvUniLUTMode. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniLUTSelector This selector selects for which LUT is applied LvUniLUTIndex/LvUniLUTValue. In case of monochrome image the LUT has only one array = Luminance. In case of color images, the LUT consists of 3 arrays, for Red, Green and Blue. LvFtrType_Enumeration. Values: LvUniLUTSelector. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniLUTEnable Enables the LUT in the processing. When disabled, the LUT does not lose its values, the disabled LUT is substitued by a linear LUT, and when enabling the LUT, the original values are retained. LvFtrType_Boolean. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniLUTIndex Index of the element to be accessed in the selected LUT via the LvUniLUTValue feature. Note that accessing the whole LUT by this approach can be very time consuming, namely on GigE cameras. If possible, it is better to use the LvUniLUTValueAll or SynView dedicated LUT functions. LvFtrType_Integer. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniLUTValue Value of the element for the current LvUniLUTIndex in the selected LUT. Note that accessing the whole LUT by this approach can be very time consuming, namely on GigE cameras. If possible, it is better to use the LvUniLUTValueAll or SynView dedicated LUT functions. LvFtrType_Integer. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniLUTValueAll This feature enables to get/set the entire content of the selected LUT in one block access. Beware that the LUT buffer structure is vendor and model dependent, so take care if your application is exptected to work with various types od devices or devices from various vendors. LvFtr↩ Type_Buffer. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvUniColorTransformationMode Selects the Color Transformation matrix control mode. The mode determines, if the matrix can be directly modified by the application, or if the matrix is to be reserved for implementation of the Saturation or other higher level features - in such case the matrix is filled with precalculated values by SynView library and cannot be directly modified. LvFtrType_Enumeration. Values: LvUniColorTransformationMode. Device local feature (LvFtrGroup_DeviceLocal).

LvDevice_LvDeviceExpiringDate expiring date of the device. LvFtrType_String. Device local feature (Lv↩ FtrGroup_DeviceLocal).

LvDevice_Info Constant to be used in LvGetInfo() and LvGetInfoStr() to obtain various info about the device.

5.29.2.3 enum LvEventFtr

LvEventFtr constants.

Enumerator

LvEvent_EventType Represents the GenTL EVENT_EVENT_TYPE info - The event type. LvFtrType_↩ Integer.

LvEvent_NumInQueue Represents the GenTL EVENT_NUM_IN_QUEUE info - Number of events in the event data queue. LvFtrType_Integer.

LvEvent_NumFired Represents the GenTL EVENT_NUM_FIRED info - Number of events, that were fired since the creation of the event. LvFtrType_Integer

5.29.2.4 enum LvInterfaceFtr

LvInterfaceFtr constants.

Enumerator

LvInterface_InterfaceID Interface ID. LvFtrType_String. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_InterfaceType Interface type. LvFtrType_Enumeration. Values: LvInterfaceType. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevInterfaceGatewaySelector Interface gateway selector. LvFtrType_Integer. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevInterfaceGateway Interface gateway. LvFtrType_Integer. Depends on LvInterface_GevInterfaceGatewaySelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevMACAddress Interface MAC address. LvFtrType_Integer. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevInterfaceSubnetSelector Interface subnet selector. LvFtrType_Integer. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevInterfaceSubnetIPAddress Interface subnet IP address. LvFtrType_Integer. Depends on LvInterface_GevInterfaceSubnetSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevInterfaceSubnetMask Interface subnet mask. LvFtrType_Integer. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_DeviceUpdateList Update internal list ofdevices. LvFtrType_Command. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_DeviceSelector Device selector. LvFtrType_Integer. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_DeviceID Device ID. LvFtrType_String. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_DeviceVendorName Device vendor name. LvFtrType_String. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_DeviceModelName Device Model name. LvFtrType_String. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_DeviceAccessStatus Device access status. LvFtrType_Enumeration. Values: LvDeviceAccessStatus. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevDeviceIPAddress Device IP address. LvFtrType_Integer. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevDeviceSubnetMask Device subnet mask. LvFtrType_Integer. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_GevDeviceMACAddress Device MAC address. LvFtrType_Integer. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_LvDeviceUserID Device User ID. LvFtrType_String. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_LvDeviceSerialNumber Device identifier (serial number). LvFtrType_String. Depends on LvInterface_DeviceSelector. Interface GenTL feature (LvFtrGroup_InterfaceGtl).

LvInterface_LvInterfaceDisplayName User readable name of the interface. LvFtrType_String. Interface local feature (LvFtrGroup_InterfaceLocal).

LvInterface_Info Constant to be used in LvGetInfo() and LvGetInfoStr() to obtain various info about the interface.

5.29.2.5 enum LvRendererFtr

LvRendererFtr constants.

Enumerator

LvRenderer_LvAutoDisplay If set, the image is automatically displayed before it is passed to the supplied callback. This is functional only in case the Event thread is started. LvFtrType_Boolean. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvRenderType Controls way how the acquired images are rendered on the screen. Note that all the Scale- options require scaling capability of the display and might not be supported in all operating systems. LvFtrType_Enumeration. Values: LvRenderType. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvOffsetX Sets the horizontal offset of the image to be rendered, i.e. the distance from the left edge of the display window. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvOffsetY Sets the vertical offset of the image to be rendered, i.e. the distance from the top edge of the display window. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvWidth Sets the width of the rectangle to which the image is to be rendered. Note that if the LvIgnoreAspectRatio feature is False, the real image width can be smaller, in order to keep the aspect ratio. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvHeight Sets the height of the rectangle to which the image is to be rendered. Note that if the LvIgnoreAspectRatio feature is False, the real image height can be smaller, in order to keep the aspect ratio. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvIgnoreAspectRatio Allows to ignore the original aspect ratio while rendering the image, so the image can be scaled up/down in one dimension with different factor than in the other dimension. LvFtrType_Boolean. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvDisableScaleUp Disables scaling the image up. LvFtrType_Boolean. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvDisableScaleDown Disables scaling the image down. LvFtrType_Boolean. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvCenterImage If the image is smaller than required window client size or the specified rectangle, the image is placed to the center. LvFtrType_Boolean. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvNumberOfTiles Sets the number of tiles used for image rendering. Note that for the tile repaint is needed that the corresponding buffers are still in the application ownership; once the buffer is placed to the input buffer pool, it should not be accessed for paint anymore (see also LvPostponeQueueBuffers). LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvColumns Sets the number of columns used for image rendering. When the value is 0, the number of columns is calculated automatically. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvRows Sets the number of rows used for image rendering. When the value is 0, the number of rows is calculated automatically. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvTileGap Sets the gap between the tiles in pixels. LvFtrType_Integer. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_LvAutoTileCalculation When set to True, the tile sizes and positions are calculated automatically. When the LvColumns and/or LvRows are 0, also the number of columns and/or rows is calculated automatically. LvFtrType_Boolean. Renderer local feature (LvFtrGroup_RendererLocal).

LvRenderer_Info Constant to be used in LvGetInfo() and LvGetInfoStr() to obtain various info about the device.

5.29.2.6 enum LvStreamFtr

LvStreamFtr constants.

Enumerator

LvStream_StreamID Stream ID. LvFtrType_String. Stream GenTL feature (LvFtrGroup_StreamGtl).

LvStream_StreamAnnouncedBufferCount Number of buffers announced for the data stream. LvFtrType_Integer. Stream GenTL feature (LvFtrGroup_StreamGtl).

LvStream_StreamAcquisitionModeSelector Selects desired acquisition mode. LvFtrType_Enumeration. Values: LvStreamAcquisitionModeSelector. Stream GenTL feature (LvFtrGroup_StreamGtl).

LvStream_StreamAnnounceBufferMinimum Minimum number of buffers to be announced for selected acquisition mode. LvFtrType_Integer. Stream GenTL feature (LvFtrGroup_StreamGtl).

LvStream_StreamType Stream type. LvFtrType_Enumeration. Values: LvStreamType. Stream GenTL feature (LvFtrGroup_StreamGtl).

LvStream_LvStreamDisplayName Returns the display name of the stream. LvFtrType_String. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvCalcPayloadSize Returns the payload size (size of buffer to hold the image data). If the payload size is not provided by the stream or device, it is calculated, so this feature returns always a valid value. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvPostponeQueueBuffers Number of buffers to be kept postponed before returning to the input buffer pool. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvAwaitDeliveryLimit Limit for images in the output buffer. Applicable only if the event thread is running - then if there is more than this number of buffers in the output queue, the oldest buffers are discarded and returned to input buffer pool. This is useful in case the application is not able to process all the images in time. LvFtrType_Integer. If the value is 0, this limit is inactive. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvAutoAllocateProcessBuffers Enable the auto allocation of process buffers. The process buffers are allocated only if they are needed for the image processing or conversion. You can disable the automatic buffer alocation and provide own buffers, using the LvBufferAttachProcessBuffer() function. LvFtrType_Boolean. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvPreallocateProcessBuffers Preallocates all the process buffers, even if it is not yet sure if they will be needed. With this command you can avoid time delays when allocating the buffers during the acquisition. LvFtrType_Command. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumDelivered Number of acquired frames since last acquisition start. It is equivalent to the GenTL STREAM_INFO_NUM_DELIVERED info. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumUnderrun Number of lost frames due to input buffer pool underrun since stream open. It is equivalent to the GenTL STREAM_INFO_NUM_UNDERRUN info. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumAnnounced Number of announced buffers. It is equivalent to the GenTL STREAM_INFO_NUM_ANNOUNCED info. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumQueued Number of buffers currently in the input pool. It is equivalent to the GenTL STREAM_INFO_NUM_QUEUED info. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumAwaitDelivery Number of buffers currently in the output queue. It is equivalent to the GenTL STREAM_INFO_NUM_AWAIT_DELIVERY info. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvIsGrabbing Flag indicating whether the acquisition engine is started or not. This is independent from the acquisition status of the remote device. It is equivalent to the GenTL STREAM_INFO_IS_GRABBING info. LvFtrType_Boolean. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumAborted Number of aborted frames since last acquisition start. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_LvNumStarted Number of started frames since stream open. It is equivalent to the GenTL STREAM_INFO_NUM_STARTED info. LvFtrType_Integer. Stream local feature (LvFtrGroup_StreamLocal).

LvStream_Info Constant to be used in LvGetInfo() and LvGetInfoStr() to obtain various info about the stream.

5.29.2.7 enum LvSystemFtr

LvSystemFtr constants.

Enumerator

LvSystem_TLVendorName GenTL producer vendor name. LvFtrType_String. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_TLModelName GenTL producer model name. LvFtrType_String. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_TLID GenTL producer ID. LvFtrType_String. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_TLVersion GenTL producer version. LvFtrType_String. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_TLPath GenTL producer path. LvFtrType_String. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_TLType GenTL producer type. LvFtrType_Enumeration. Values: LvTLType. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_GenTLVersionMajor GenTL version major. LvFtrType_Integer. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_GenTLVersionMinor GenTL version minor. LvFtrType_Integer. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_GevVersionMajor GigE Vision version major. LvFtrType_Integer. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_GevVersionMinor GigE Vision version minor. LvFtrType_Integer. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_InterfaceUpdateList Update internal list of interfaces. LvFtrType_Command. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_InterfaceSelector Interface selector. LvFtrType_Integer. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_InterfaceID Interface ID. LvFtrType_String. Depends on LvSystem_InterfaceSelector. System GenTL feature (LvFtrGroup_SystemGtl).

LvSystem_GevInterfaceMACAddress Interface MAC address. LvFtrType_Integer. Depends on LvSystem_InterfaceSelector. System GenTL feature (LvFtrGroup_InterfaceGtl).

LvSystem_GevInterfaceDefaultIPAddress Interface default IP address. LvFtrType_Integer. Depends on LvSystem_InterfaceSelector. System GenTL feature (LvFtrGroup_InterfaceGtl).

LvSystem_GevInterfaceDefaultSubnetMask Interface default subnet mask. LvFtrType_Integer. Depends on LvSystem_InterfaceSelector. System GenTL feature (LvFtrGroup_InterfaceGtl).

LvSystem_GevInterfaceDefaultGateway Interface default gateway. LvFtrType_Integer. Depends on LvSystem_InterfaceSelector. System GenTL feature (LvFtrGroup_InterfaceGtl).

LvSystem_LvSystemDisplayName User readable name of the system. LvFtrType_String. System local feature (LvFtrGroup_SystemLocal).

LvSystem_Info Constant to be used in LvGetInfo() and LvGetInfoStr() to obtain various info about the system.

5.30 Enumeration entries

Enumerations

- enum LvPixelFormat {
  LvPixelFormat_Mono8 = 0x01080001, LvPixelFormat_Mono8S = 0x01080002, LvPixelFormat_Mono10 = 0x01100003, LvPixelFormat_Mono10Packed = 0x010C0004,
  LvPixelFormat_Mono12 = 0x01100005, LvPixelFormat_Mono12Packed = 0x010C0006, LvPixelFormat_Mono14 = 0x01100025, LvPixelFormat_Mono16 = 0x01100007,
  LvPixelFormat_BayerGR8 = 0x01080008, LvPixelFormat_BayerRG8 = 0x01080009, LvPixelFormat_BayerGB8 = 0x0108000A, LvPixelFormat_BayerBG8 = 0x0108000B,
  LvPixelFormat_BayerGR10 = 0x0110000C, LvPixelFormat_BayerRG10 = 0x0110000D, LvPixelFormat_BayerGB10 = 0x0110000E, LvPixelFormat_BayerBG10 = 0x0110000F,
  LvPixelFormat_BayerGR12 = 0x01100010, LvPixelFormat_BayerRG12 = 0x01100011, LvPixelFormat_BayerGB12 = 0x01100012, LvPixelFormat_BayerBG12 = 0x01100013,
  LvPixelFormat_BayerGR10Packed = 0x010C0026, LvPixelFormat_BayerRG10Packed = 0x010C0027, LvPixelFormat_BayerGB10Packed = 0x010C0028, LvPixelFormat_BayerBG10Packed = 0x010C0029,
  LvPixelFormat_BayerGR12Packed = 0x010C002A, LvPixelFormat_BayerRG12Packed = 0x010C002B, LvPixelFormat_BayerGB12Packed = 0x010C002C, LvPixelFormat_BayerBG12Packed = 0x010C002D,
  LvPixelFormat_BayerGR16 = 0x0110002E, LvPixelFormat_BayerRG16 = 0x0110002F, LvPixelFormat_BayerGB16 = 0x01100030, LvPixelFormat_BayerBG16 = 0x01100031,
  LvPixelFormat_RGB8 = 0x02180014, LvPixelFormat_BGR8 = 0x02180015, LvPixelFormat_RGBA8 = 0x02200016, LvPixelFormat_BGRA8 = 0x02200017,
  LvPixelFormat_RGB10 = 0x02300018, LvPixelFormat_BGR10 = 0x02300019, LvPixelFormat_RGB12 = 0x0230001A, LvPixelFormat_BGR12 = 0x0230001B,
  LvPixelFormat_RGB16 = 0x02300033, LvPixelFormat_RGB10V1Packed = 0x0220001C, LvPixelFormat_RGB10P32 = 0x0220001D, LvPixelFormat_RGB12V1Packed = 0x02240034,
  LvPixelFormat_RGB565P = 0x02100035, LvPixelFormat_BGR565P = 0x02100036, LvPixelFormat_YUV411_8 = 0x020C001E, LvPixelFormat_YUV422_8_UYVY = 0x0210001F,
  LvPixelFormat_YUV8 = 0x02180020, LvPixelFormat_YUV422_8 = 0x02100032, LvPixelFormat_YCbCr422_8 = 0x0210003B, LvPixelFormat_YCbCr601_422_8 = 0x0210003E,
  LvPixelFormat_YCbCr601_422_8_CbYCrY = 0x02100044, LvPixelFormat_RGB8_Planar = 0x02180021, LvPixelFormat_RGB10_Planar = 0x02300022, LvPixelFormat_RGB12_Planar = 0x02300023,
  LvPixelFormat_RGB16_Planar = 0x02300024, LvPixelFormat_BGR555P = 0x021000E1 }
- enum LvDeviceAccess { LvDeviceAccess_None = 1, LvDeviceAccess_ReadOnly = 2, LvDeviceAccess_Control = 3, LvDeviceAccess_Exclusive = 4 }
- enum LvDeviceAccessStatus { LvDeviceAccessStatus_Unknown = 0, LvDeviceAccessStatus_ReadWrite = 1, LvDeviceAccessStatus_ReadOnly = 2, LvDeviceAccessStatus_NoAccess = 3 }
- enum LvDeviceScanType { LvDeviceScanType_Areascan, LvDeviceScanType_Linescan }
- enum LvDeviceClockSelector { LvDeviceClockSelector_SensorDigitization }
- enum LvDeviceTemperatureSelector { LvDeviceTemperatureSelector_Sensor, LvDeviceTemperatureSelector_Mainboard }
- enum LvAOIMode { LvAOIMode_Automatic, LvAOIMode_ClipOnTransfer, LvAOIMode_Manual }
- enum LvAcquisitionMode { LvAcquisitionMode_SingleFrame, LvAcquisitionMode_MultiFrame, LvAcquisitionMode_Continuous }
- enum LvExposureAuto { LvExposureAuto_Off, LvExposureAuto_Once, LvExposureAuto_Continuous }
- enum LvTriggerSelector { LvTriggerSelector_FrameStart, LvTriggerSelector_FrameBurstStart }
- enum LvTriggerMode { LvTriggerMode_Off, LvTriggerMode_On }
- enum LvTriggerSource {
  LvTriggerSource_Line1, LvTriggerSource_Line2, LvTriggerSource_Line3, LvTriggerSource_Line4,
  LvTriggerSource_Line5, LvTriggerSource_Line6, LvTriggerSource_Line7, LvTriggerSource_Line8,
  LvTriggerSource_Line17, LvTriggerSource_Line18, LvTriggerSource_Line19, LvTriggerSource_Line20,
  LvTriggerSource_Line21, LvTriggerSource_Line22, LvTriggerSource_Line23, LvTriggerSource_Line24,
  LvTriggerSource_Software, LvTriggerSource_Action1, LvTriggerSource_Action2, LvTriggerSource_Action3,
  LvTriggerSource_Action4, LvTriggerSource_Action5, LvTriggerSource_Action6, LvTriggerSource_Action7,
  LvTriggerSource_Action8, LvTriggerSource_Quad, LvTriggerSource_Counter1, LvTriggerSource_Counter2,
  LvTriggerSource_Counter3, LvTriggerSource_Counter4, LvTriggerSource_Timer1, LvTriggerSource_Timer2,
  LvTriggerSource_Timer3, LvTriggerSource_Timer4 }

- enum LvTriggerActivation { LvTriggerActivation_RisingEdge, LvTriggerActivation_FallingEdge }
- enum LvTriggerCaching { LvTriggerCaching_Cache, LvTriggerCaching_Drop }
- enum LvExposureMode { LvExposureMode_Timed }
- enum LvAcquisitionFrameRateControlMode { LvAcquisitionFrameRateControlMode_Off, LvAcquisition↩ FrameRateControlMode_On }
- enum LvLineSelector {
  LvLineSelector_Line1, LvLineSelector_Line2, LvLineSelector_Line3, LvLineSelector_Line4,
  LvLineSelector_Line5, LvLineSelector_Line6, LvLineSelector_Line7, LvLineSelector_Line8,
  LvLineSelector_Line9, LvLineSelector_Line10, LvLineSelector_Line11, LvLineSelector_Line12,
  LvLineSelector_Line13, LvLineSelector_Line14, LvLineSelector_Line15, LvLineSelector_Line16,
  LvLineSelector_Line17, LvLineSelector_Line18, LvLineSelector_Line19, LvLineSelector_Line20,
  LvLineSelector_Line21, LvLineSelector_Line22, LvLineSelector_Line23, LvLineSelector_Line24,
  LvLineSelector_Line25, LvLineSelector_Line26, LvLineSelector_Line27, LvLineSelector_Line28,
  LvLineSelector_Line29, LvLineSelector_Line30, LvLineSelector_Line31, LvLineSelector_Line32 }
- enum LvLineMode { LvLineMode_Input, LvLineMode_Output }
- enum LvLineFormat {
  LvLineFormat_NoConnect, LvLineFormat_TriState, LvLineFormat_TTL, LvLineFormat_LVDS,
  LvLineFormat_RS422, LvLineFormat_OptoCoupled }
- enum LvLineSource {
  LvLineSource_Off, LvLineSource_ExposureActive, LvLineSource_Timer1Active, LvLineSource_Timer2↩ Active,
  LvLineSource_Timer3Active, LvLineSource_Timer4Active, LvLineSource_UserOutput1, LvLineSource_↩ UserOutput2,
  LvLineSource_UserOutput3, LvLineSource_UserOutput4, LvLineSource_UserOutput5, LvLineSource_↩ UserOutput6,
  LvLineSource_UserOutput7, LvLineSource_UserOutput8, LvLineSource_Counter1Active, LvLineSource_↩ Counter2Active,
  LvLineSource_Counter3Active, LvLineSource_Counter4Active }
- enum LvCounterSelector { LvCounterSelector_Counter1, LvCounterSelector_Counter2, LvCounterSelector↩ _Counter3, LvCounterSelector_Counter4 }
- enum LvCounterMode { LvCounterMode_Autoreset }
- enum LvCounterEventSource {
  LvCounterEventSource_Off, LvCounterEventSource_FrameTrigger, LvCounterEventSource_TimerTick, Lv↩ CounterEventSource_Line1,
  LvCounterEventSource_Line2, LvCounterEventSource_Line3, LvCounterEventSource_Line4, LvCounter↩ EventSource_Line17,
  LvCounterEventSource_Line18 }
- enum LvTimerSelector { LvTimerSelector_Timer1, LvTimerSelector_Timer2, LvTimerSelector_Timer3, Lv↩ TimerSelector_Timer4 }
- enum LvTimerTriggerSource {
  LvTimerTriggerSource_Off, LvTimerTriggerSource_FrameTrigger, LvTimerTriggerSource_Counter1End, Lv↩ TimerTriggerSource_Counter2End,
  LvTimerTriggerSource_Counter3End, LvTimerTriggerSource_Counter4End, LvTimerTriggerSource_User↩ Output1, LvTimerTriggerSource_UserOutput2,
  LvTimerTriggerSource_UserOutput3, LvTimerTriggerSource_UserOutput4, LvTimerTriggerSource_User↩ Output5, LvTimerTriggerSource_UserOutput6,
  LvTimerTriggerSource_UserOutput7, LvTimerTriggerSource_UserOutput8 }
- enum LvSpecialPurposeTriggerSelector { LvSpecialPurposeTriggerSelector_ImageStampsReset }
- enum LvSpecialPurposeTriggerSource {
  LvSpecialPurposeTriggerSource_Off, LvSpecialPurposeTriggerSource_Line1, LvSpecialPurposeTrigger↩ Source_Line2, LvSpecialPurposeTriggerSource_Line3,
  LvSpecialPurposeTriggerSource_Line4, LvSpecialPurposeTriggerSource_Line5, LvSpecialPurposeTrigger↩ Source_Line6, LvSpecialPurposeTriggerSource_Line7,
  LvSpecialPurposeTriggerSource_Line8, LvSpecialPurposeTriggerSource_Line17, LvSpecialPurpose↩ TriggerSource_Line18, LvSpecialPurposeTriggerSource_Line19,
  LvSpecialPurposeTriggerSource_Line20, LvSpecialPurposeTriggerSource_Line21, LvSpecialPurpose↩ TriggerSource_Line22, LvSpecialPurposeTriggerSource_Line23,
  LvSpecialPurposeTriggerSource_Line24, LvSpecialPurposeTriggerSource_Action1, LvSpecialPurpose↩

TriggerSource_Action2, LvSpecialPurposeTriggerSource_Action3, LvSpecialPurposeTriggerSource_Action4, LvSpecialPurposeTriggerSource_Action5, LvSpecialPurposeTriggerSource_Action6, LvSpecialPurposeTriggerSource_Action7, LvSpecialPurposeTriggerSource_Action8 }

- enum LvSpecialPurposeTriggerActivation { LvSpecialPurposeTriggerActivation_RisingEdge, LvSpecialPurposeTriggerActivation_FallingEdge }
- enum LvImageStampSelector { LvImageStampSelector_Timestamp, LvImageStampSelector_FrameID }
- enum LvBootSwitch { LvBootSwitch_PureGEV, LvBootSwitch_Legacy }
- enum LvGainSelector { LvGainSelector_All, LvGainSelector_AnalogAll, LvGainSelector_DigitalAll }
- enum LvGainAuto { LvGainAuto_Off, LvGainAuto_Once, LvGainAuto_Continuous }
- enum LvBlackLevelSelector { LvBlackLevelSelector_All }
- enum LvBlackLevelAuto { LvBlackLevelAuto_Off, LvBlackLevelAuto_Once, LvBlackLevelAuto_Continuous }
- enum LvColorTransformationSelector { LvColorTransformationSelector_RGBtoRGB }
- enum LvColorTransformationValueSelector { LvColorTransformationValueSelector_Gain00, LvColorTransformationValueSelector_Gain01, LvColorTransformationValueSelector_Gain02, LvColorTransformationValueSelector_Gain10, LvColorTransformationValueSelector_Gain11, LvColorTransformationValueSelector_Gain12, LvColorTransformationValueSelector_Gain20, LvColorTransformationValueSelector_Gain21, LvColorTransformationValueSelector_Gain22 }
- enum LvExternalDeviceControlMode { LvExternalDeviceControlMode_Custom }
- enum LvExternalADCSelector { LvExternalADCSelector_ExternalADC1, LvExternalADCSelector_ExternalADC2, LvExternalADCSelector_ExternalADC3, LvExternalADCSelector_ExternalADC4 }
- enum LvPowerSwitchCurrentAction { LvPowerSwitchCurrentAction_Idle, LvPowerSwitchCurrentAction_Pulse, LvPowerSwitchCurrentAction_Calibrate, LvPowerSwitchCurrentAction_AdjustPosition, LvPowerSwitchCurrentAction_Drive }
- enum LvPowerSwitchSelector { LvPowerSwitchSelector_PowerSwitch1, LvPowerSwitchSelector_PowerSwitch2, LvPowerSwitchSelector_PowerSwitch3, LvPowerSwitchSelector_PowerSwitch4 }
- enum LvPowerSwitchDrive { LvPowerSwitchDrive_Off, LvPowerSwitchDrive_Plus, LvPowerSwitchDrive_Minus }
- enum LvPowerSwitchDriveAll { LvPowerSwitchDriveAll_Off, LvPowerSwitchDriveAll_Plus, LvPowerSwitchDriveAll_Minus }
- enum LvPowerSwitchBoundADC { LvPowerSwitchBoundADC_None, LvPowerSwitchBoundADC_ExternalADC1, LvPowerSwitchBoundADC_ExternalADC2, LvPowerSwitchBoundADC_ExternalADC3, LvPowerSwitchBoundADC_ExternalADC4 }
- enum LvLensControlTargetApproach { LvLensControlTargetApproach_Direct, LvLensControlTargetApproach_FromPlus, LvLensControlTargetApproach_FromMinus }
- enum LvLUTSelector { LvLUTSelector_Luminance, LvLUTSelector_Red, LvLUTSelector_Green, LvLUTSelector_Blue }
- enum LvGevDeviceModeCharacterSet { LvGevDeviceModeCharacterSet_UTF8 }
- enum LvGevSupportedOptionSelector { LvGevSupportedOptionSelector_IPConfigurationLLA, LvGevSupportedOptionSelector_IPConfigurationDHCP, LvGevSupportedOptionSelector_IPConfigurationPersistentIP, LvGevSupportedOptionSelector_CommandsConcatenation, LvGevSupportedOptionSelector_WriteMem, LvGevSupportedOptionSelector_PacketResend, LvGevSupportedOptionSelector_Event, LvGevSupportedOptionSelector_EventData, LvGevSupportedOptionSelector_PendingAck, LvGevSupportedOptionSelector_Action, LvGevSupportedOptionSelector_PrimaryApplicationSwitchover, LvGevSupportedOptionSelector_ExtendedStatusCodes, LvGevSupportedOptionSelector_DiscoveryAckDelayWritable, LvGevSupportedOptionSelector_DiscoveryAckDelay, LvGevSupportedOptionSelector_TestData, LvGevSupportedOptionSelector_ManifestTable, LvGevSupportedOptionSelector_CCPApplicationSocket, LvGevSupportedOptionSelector_LinkSpeed, LvGevSupportedOptionSelector_HeartbeatDisable, LvGevSupportedOptionSelector_SerialNumber, LvGevSupportedOptionSelector_UserDefinedName, LvGevSupportedOptionSelector_StreamChannelSourceSocket, LvGevSupportedOptionSelector_StreamChannel0ExtendedChunkData, LvGevSupportedOptionSelector_StreamChannel0UnconditionalStreaming, LvGevSupportedOptionSelector_StreamChannel0IPReassembly, LvGevSupportedOptionSelector_StreamChannel0BigAndLittleEndian, LvGevSupportedOptionSelector_MessageChannelSourceSocket }

- enum LvGevCCP { LvGevCCP_OpenAccess, LvGevCCP_ExclusiveAccess, LvGevCCP_ControlAccess, LvGevCCP_ControlAccessSwitchoverActive }
- enum LvUserSetSelector { LvUserSetSelector_Default, LvUserSetSelector_UserSet1, LvUserSetSelector_UserSet2, LvUserSetSelector_UserSet3, LvUserSetSelector_UserSet4 }
- enum LvUserSetDefaultSelector { LvUserSetDefaultSelector_Default, LvUserSetDefaultSelector_UserSet1, LvUserSetDefaultSelector_UserSet2, LvUserSetDefaultSelector_UserSet3, LvUserSetDefaultSelector_UserSet4, LvUserSetDefaultSelector_None }
- enum LvChunkSelector { LvChunkSelector_OffsetX, LvChunkSelector_OffsetY, LvChunkSelector_Width, LvChunkSelector_Height, LvChunkSelector_PixelFormat, LvChunkSelector_LinePitch, LvChunkSelector_FrameID, LvChunkSelector_Timestamp, LvChunkSelector_ExposureTime, LvChunkSelector_Gain, LvChunkSelector_LineStatusAll, LvChunkSelector_BlackLevel, LvChunkSelector_LvExternalADCValue, LvChunkSelector_LvSmartAppString, LvChunkSelector_LvSmartAppInt, LvChunkSelector_LvSmartAppUint, LvChunkSelector_LvSmartAppRegister, LvChunkSelector_LvTriggerDelayed, LvChunkSelector_LvStrobeDropped, LvChunkSelector_LvFrameAbort, LvChunkSelector_LvTriggerDropped, LvChunkSelector_LvTriggerError, LvChunkSelector_LvEncoderPosition, LvChunkSelector_LvEncoderRotation }
- enum LvChunkGainSelector { LvChunkGainSelector_AnalogAll, LvChunkGainSelector_DigitalAll }
- enum LvEventSelector { LvEventSelector_LvLog, LvEventSelector_LvSmartAppLog, LvEventSelector_LvSmartAppString, LvEventSelector_LvSmartAppInt, LvEventSelector_LvSmartAppUint, LvEventSelector_LvSmartAppRegister, LvEventSelector_LvTriggerDropped }
- enum LvEventNotification { LvEventNotification_Off, LvEventNotification_On }
- enum LvTLType { LvTLType_Mixed, LvTLType_Custom, LvTLType_GEV }
- enum LvInterfaceType { LvInterfaceType_Custom, LvInterfaceType_GEV, LvInterfaceType_U3V }
- enum LvDeviceType { LvDeviceType_Custom, LvDeviceType_GEV, LvDeviceType_U3V }
- enum LvGevDeviceStreamCaptureMode { LvGevDeviceStreamCaptureMode_SystemDefault, LvGevDeviceStreamCaptureMode_Socket, LvGevDeviceStreamCaptureMode_FilterDriver }
- enum LvStreamAcquisitionModeSelector { LvStreamAcquisitionModeSelector_Default }
- enum LvStreamType { LvStreamType_Custom, LvStreamType_GEV, LvStreamType_U3V }
- enum LvUniProcessMode { LvUniProcessMode_HwOnly, LvUniProcessMode_SwOnly, LvUniProcessMode_Auto, LvUniProcessMode_Off }
- enum LvBayerDecoderAlgorithm { LvBayerDecoderAlgorithm_NearestNeighbour, LvBayerDecoderAlgorithm_BilinearInterpolation, LvBayerDecoderAlgorithm_BilinearColorCorrection, LvBayerDecoderAlgorithm_PixelGrouping, LvBayerDecoderAlgorithm_VariableGradient }
- enum LvUniBalanceRatioSelector { LvUniBalanceRatioSelector_Red, LvUniBalanceRatioSelector_Green, LvUniBalanceRatioSelector_Blue }
- enum LvUniBalanceWhiteAuto { LvUniBalanceWhiteAuto_Off, LvUniBalanceWhiteAuto_Once }
- enum LvUniColorTransformationSelector { LvUniColorTransformationSelector_RGBtoRGB }
- enum LvUniColorTransformationValueSelector { LvUniColorTransformationValueSelector_Gain00, LvUniColorTransformationValueSelector_Gain01, LvUniColorTransformationValueSelector_Gain02, LvUniColorTransformationValueSelector_Gain10, LvUniColorTransformationValueSelector_Gain11, LvUniColorTransformationValueSelector_Gain12, LvUniColorTransformationValueSelector_Gain20, LvUniColorTransformationValueSelector_Gain21, LvUniColorTransformationValueSelector_Gain22 }
- enum LvRenderType { LvRenderType_FullSize, LvRenderType_ScaleToFit, LvRenderType_ScaleToSize, LvRenderType_ScaleToTiles }
- enum LvSerialPortBaudRate { LvSerialPortBaudRate_Baud2400, LvSerialPortBaudRate_Baud4800, LvSerialPortBaudRate_Baud9600,

LvSerialPortBaudRate_Baud14400, LvSerialPortBaudRate_Baud19200, LvSerialPortBaudRate_Baud38400, LvSerialPortBaudRate_Baud57600, LvSerialPortBaudRate_Baud115200 }

- enum LvSerialPortParity { LvSerialPortParity_None, LvSerialPortParity_Odd, LvSerialPortParity_Even }
- enum LvSerialPortDataBits { LvSerialPortDataBits_DataBits7, LvSerialPortDataBits_DataBits8 }
- enum LvSerialPortStopBits { LvSerialPortStopBits_StopBits1, LvSerialPortStopBits_StopBits1dot5, LvSerialPortStopBits_StopBits2 }
- enum LvSerialPortCommandStatus { LvSerialPortCommandStatus_Success, LvSerialPortCommandStatus_Timeout, LvSerialPortCommandStatus_PortBusy, LvSerialPortCommandStatus_CommunicationError, LvSerialPortCommandStatus_FrameError, LvSerialPortCommandStatus_ParityError, LvSerialPortCommandStatus_Overflow }
- enum LvChunkLvExternalADCSelector { LvChunkLvExternalADCSelector_ExternalADC1, LvChunkLvExternalADCSelector_ExternalADC2, LvChunkLvExternalADCSelector_ExternalADC3, LvChunkLvExternalADCSelector_ExternalADC4 }
- enum LvUserOutputSelector { LvUserOutputSelector_UserOutput1, LvUserOutputSelector_UserOutput2, LvUserOutputSelector_UserOutput3, LvUserOutputSelector_UserOutput4, LvUserOutputSelector_UserOutput5, LvUserOutputSelector_UserOutput6, LvUserOutputSelector_UserOutput7, LvUserOutputSelector_UserOutput8 }
- enum LvUniProcessExecution { LvUniProcessExecution_OnBufferPtrQuery, LvUniProcessExecution_OnPopFromQueue, LvUniProcessExecution_OnExplicitRequest }
- enum LvLensControlCalibrationStatus { LvLensControlCalibrationStatus_Invalid, LvLensControlCalibrationStatus_Valid }
- enum LvLUTMode { LvLUTMode_Direct, LvLUTMode_BalanceWhite }
- enum LvBalanceRatioSelector { LvBalanceRatioSelector_Red, LvBalanceRatioSelector_Green, LvBalanceRatioSelector_Blue }
- enum LvBalanceWhiteAuto { LvBalanceWhiteAuto_Off, LvBalanceWhiteAuto_Once, LvBalanceWhiteAuto_Continuous }
- enum LvGevDeviceClass { LvGevDeviceClass_Transmitter }
- enum LvGevIPConfigurationStatus { LvGevIPConfigurationStatus_None, LvGevIPConfigurationStatus_PersistentIP, LvGevIPConfigurationStatus_DHCP, LvGevIPConfigurationStatus_LLA, LvGevIPConfigurationStatus_ForceIP }
- enum LvGevSCPDirection { LvGevSCPDirection_Transmitter }
- enum LvDeviceEndianessMechanism { LvDeviceEndianessMechanism_Legacy, LvDeviceEndianessMechanism_Standard }
- enum LvUniLUTMode { LvUniLUTMode_Direct, LvUniLUTMode_Generated }
- enum LvUniLUTSelector { LvUniLUTSelector_Luminance, LvUniLUTSelector_Red, LvUniLUTSelector_Green, LvUniLUTSelector_Blue }
- enum LvUniColorTransformationMode { LvUniColorTransformationMode_Direct, LvUniColorTransformationMode_Generated }
- enum LvStrobeEnable { LvStrobeEnable_Off, LvStrobeEnable_AllClusters, LvStrobeEnable_LEDCluster1, LvStrobeEnable_LEDCluster2 }
- enum LvStrobeDurationMode { LvStrobeDurationMode_FrameRateRelated, LvStrobeDurationMode_Free }
- enum LvStrobeDropMode { LvStrobeDropMode_DropStrobe, LvStrobeDropMode_DelayFrame }
- enum LvRegionSelector { LvRegionSelector_Region0, LvRegionSelector_Region1, LvRegionSelector_Region2, LvRegionSelector_Region3 }

5.30.1 Detailed Description

5.30.2 Enumeration Type Documentation

5.30.2.1 enum LvAcquisitionFrameRateControlMode

Enum values for the LvDevice_LvAcquisitionFrameRateControlMode feature.

Enumerator

LvAcquisitionFrameRateControlMode_Off Disables frame rate control - the camera operates at maximum frame rate

LvAcquisitionFrameRateControlMode_On Enables frame rate control - the rate can be explicitly adjusted

5.30.2.2 enum LvAcquisitionMode

Enum values for the LvDevice_AcquisitionMode feature.

Enumerator

LvAcquisitionMode_SingleFrame Single frame acquisition - after acquisition starts, single frame is acquired and acquisition stops.

LvAcquisitionMode_MultiFrame Multiple frame acquisition - after acquisition starts, specified numbr of frames is acquired and acquisition stops.

LvAcquisitionMode_Continuous Continuous acquisition - after starting, the acquisition is active until explicitly stopped.

5.30.2.3 enum LvAOIMode

Enum values for the LvDevice_LvAOIMode feature.

Enumerator

LvAOIMode_Automatic Camera automatically applies as much of the desired AOI setting on the sensor and the rest is cut on transfer

LvAOIMode_ClipOnTransfer The AOI is applied before the transfer, in camera memory

LvAOIMode_Manual Fine control of separate AOI setting on the sensor and before the transfer

5.30.2.4 enum LvBalanceRatioSelector

Enum values for the LvDevice_BalanceRatioSelector feature.

Enumerator

LvBalanceRatioSelector_Red Balance ratio will be applied to the red channel.

LvBalanceRatioSelector_Green Balance ratio will be applied to the green channel.

LvBalanceRatioSelector_Blue Balance ratio will be applied to the blue channel.

5.30.2.5 enum LvBalanceWhiteAuto

Enum values for the LvDevice_BalanceWhiteAuto feature.

Enumerator

LvBalanceWhiteAuto_Off Automatic white balance mode off - the automatic white balance is not applied.

LvBalanceWhiteAuto_Once Automatic white balance mode once - the white balance factors are once adjusted, then switches the enumeration back to the Off value.

LvBalanceWhiteAuto_Continuous Automatic white balance mode continuous - the white balance is continuously auto-adjusted.

5.30.2.6 enum LvBayerDecoderAlgorithm

Enum values for the LvDevice_LvBayerDecoderAlgorithm and LvDevice_LvUniBayerDecoderAlgorithm feature.

Enumerator

LvBayerDecoderAlgorithm_NearestNeighbour Nearest neighbour algorithm - Fastest decoding, giving the worst results, enables also decoding to a monochrome pixel format.

LvBayerDecoderAlgorithm_BilinearInterpolation Bilinear interpolation algorithm - Fast common decoding, enables also decoding to a monochrome pixel format.

LvBayerDecoderAlgorithm_BilinearColorCorrection Bilinear color correction algorithm - Decoding with quick enhacements on edges.

LvBayerDecoderAlgorithm_PixelGrouping Pixel grouping algorithm - Slower decoding, giving very good results.

LvBayerDecoderAlgorithm_VariableGradient Variable gradient algorithm - Slowest decoding, giving the best results.

5.30.2.7 enum LvBlackLevelAuto

Enum values for the LvDevice_BlackLevelAuto feature.

Enumerator

LvBlackLevelAuto_Off Automatic black level mode off - the black level value is controlled 'manually'.

LvBlackLevelAuto_Once Automatic black level mode 'once' - the black level value is calculated and applied once and the feature switches back to 'off' (manual mode).

LvBlackLevelAuto_Continuous Continuous automatic black level mode - the automatic black level is applied continuously.

5.30.2.8 enum LvBlackLevelSelector

Enum values for the LvDevice_BlackLevelSelector feature.

Enumerator

LvBlackLevelSelector_All Apply black level on all channels and taps.

5.30.2.9 enum LvBootSwitch

Enum values for the LvDevice_LvBootSwitch feature.

Enumerator

LvBootSwitch_PureGEV Selects the pure GigE Vision mode strictly following the GigE Vision specification

LvBootSwitch_Legacy Selects the legacy mode allowing dual operation through GigE Vision or custom protocol.

5.30.2.10 enum LvChunkGainSelector

Enum values for the LvDevice_ChunkGainSelector feature.

Enumerator

LvChunkGainSelector_AnalogAll Analog gain.

LvChunkGainSelector_DigitalAll Digital gain.

5.30.2.11 enum LvChunkLvExternalADCSelector

Enum values for the LvDevice_ChunkLvExternalADCSelector feature.

Enumerator

LvChunkLvExternalADCSelector_ExternalADC1 External ADC 1.
LvChunkLvExternalADCSelector_ExternalADC2 External ADC 2.
LvChunkLvExternalADCSelector_ExternalADC3 External ADC 3.
LvChunkLvExternalADCSelector_ExternalADC4 External ADC 4.

5.30.2.12 enum LvChunkSelector

Enum values for the LvDevice_ChunkSelector feature.

Enumerator

LvChunkSelector_OffsetX Selects the X offset chunk for configuration.
LvChunkSelector_OffsetY Selects the Y offset chunk for configuration.
LvChunkSelector_Width Selects the width chunk for configuration.
LvChunkSelector_Height Selects the height chunk for configuration.
LvChunkSelector_PixelFormat Selects the pixel format chunk for configuration.
LvChunkSelector_LinePitch Selects the line pitch chunk for configuration.
LvChunkSelector_FrameID Selects the frame id chunk for configuration.
LvChunkSelector_Timestamp Selects the time stamp chunk for configuration.
LvChunkSelector_ExposureTime Selects the exposure time chunk for configuration.
LvChunkSelector_Gain Selects the gain chunk for configuration.
LvChunkSelector_LineStatusAll Selects the line status all chunk for configuration.
LvChunkSelector_BlackLevel Selects the black level chunk for configuration.
LvChunkSelector_LvExternalADCValue Selects the external ADC chunk for configuration.
LvChunkSelector_LvSmartAppString Selects the smart application string chunk for configuration.
LvChunkSelector_LvSmartAppInt Selects the smart application signed integer chunk for configuration.
LvChunkSelector_LvSmartAppUint Selects the smart application unsigned integer chunk for configuration.

LvChunkSelector_LvSmartAppRegister Selects the smart application raw register chunk for configuration.

LvChunkSelector_LvTriggerDelayed Selects the trigger delayed chunk for configuration.
LvChunkSelector_LvStrobeDropped Selects the strobe dropped chunk for configuration.
LvChunkSelector_LvFrameAbort Selects the frame abort chunk for configuration.
LvChunkSelector_LvTriggerDropped Selects the trigger dropped chunk for configuration.
LvChunkSelector_LvTriggerError Selects the trigger error chunk for configuration.
LvChunkSelector_LvEncoderPosition Selects the encoder position chunk for configuration.
LvChunkSelector_LvEncoderRotation Selects the encoder rotation chunk for configuration.

5.30.2.13 enum LvColorTransformationSelector

Enum values for the LvDevice_ColorTransformationSelector feature.

Enumerator

LvColorTransformationSelector_RGBtoRGB RGB to RGB matrix transformation - currently the only Color Transformation matrix type.

5.30.2.14 enum LvColorTransformationValueSelector

Enum values for the LvDevice_ColorTransformationValueSelector feature.

Enumerator

LvColorTransformationValueSelector_Gain00 Selects the gain 00 (RR, red-red) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain01 Selects the gain 01 (RG, red-green) entry of the colortransformation matrix.

LvColorTransformationValueSelector_Gain02 Selects the gain 02 (RB, red-blue) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain10 Selects the gain 10 (GR, green-red) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain11 Selects the gain 11 (GG, green-green) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain12 Selects the gain 12 (GB, green-blue) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain20 Selects the gain 20 (BR, blue-red) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain21 Selects the gain 21 (BG, blue-green) entry of the color transformation matrix.

LvColorTransformationValueSelector_Gain22 Selects the gain 22 (BB, blue-blue) entry of the color transformation matrix.

5.30.2.15 enum LvCounterEventSource

Enum values for the LvDevice_CounterEventSource feature.

Enumerator

LvCounterEventSource_Off Switches counter event signal off - no signal will be incrementing the counter

LvCounterEventSource_FrameTrigger Switches counter event signal to frame trigger - activation of the frame trigger internal signal (before counting down eventual trigger delay) increments the counter.

LvCounterEventSource_TimerTick Switches counter event signal to timer tick - 1MHz clock increments the counter.

LvCounterEventSource_Line1 Switches counter event signal to line 1 (optocoupler input) - active edge of line 1 increments the counter.

LvCounterEventSource_Line2 Switches counter event signal to line 2 (optocoupler input) - active edge of line 2 increments the counter.

LvCounterEventSource_Line3 Switches counter event signal to line 3 (optocoupler input) - active edge of line 3 increments the counter.

LvCounterEventSource_Line4 Switches counter event signal to line 4 (optocoupler input) - active edge of line 4 increments the counter.

LvCounterEventSource_Line17 Switches counter event signal to line 17 (TTL input) - active edge of line 17 increments the counter.

LvCounterEventSource_Line18 Switches counter event signal to line 18 (TTL input) - active edge of line 18 increments the counter.

5.30.2.16 enum LvCounterMode

Enum values for the LvDevice_LvCounterMode feature.

Enumerator

LvCounterMode_Autoreset Automatic reset mode. Once completed, the counter automatically resets itself and starts counting again.

5.30.2.17 enum LvCounterSelector

Enum values for the LvDevice_CounterSelector feature.

Enumerator

LvCounterSelector_Counter1 Selects counter 1 for configuration.

LvCounterSelector_Counter2 Selects counter 2 for configuration.

LvCounterSelector_Counter3 Selects counter 3 for configuration.

LvCounterSelector_Counter4 Selects counter 4 for configuration.

5.30.2.18 enum LvDeviceAccess

This enum is used for opening the Device - see LvDeviceOpen().

Enumerator

LvDeviceAccess_None Represents the GenTL DEVICE_ACCESS_NONE. This either means that the Device is not open because it was not opened before or the access to it was denied.

LvDeviceAccess_ReadOnly Represents the GenTL DEVICE_ACCESS_READONLY. Open the Device read only. All Port functions can only read from the Device.

LvDeviceAccess_Control Represents the GenTL DEVICE_ACCESS_CONTROL. Open the Device in a way that other hosts/processes can have read only access to the Device. Device access level is read/write for this process.

LvDeviceAccess_Exclusive Represents the GenTL DEVICE_ACCESS_EXCLUSIVE. Open the Device in a way that only this host/process can have access to the Device. Device access level is read/write for this process.

5.30.2.19 enum LvDeviceAccessStatus

Values for the LvFtrInfo_DeviceAccessStatus and LvInterface_DeviceAccessStatus features.

Enumerator

LvDeviceAccessStatus_Unknown Represents the GenTL DEVICE_ACCESS_STATUS_UNKNOWN. The current availability of the Device is unknown.

LvDeviceAccessStatus_ReadWrite Represents the GenTL DEVICE_ACCESS_STATUS_READWRITE - The Device is available for Read/Write.

LvDeviceAccessStatus_ReadOnly Represents the GenTL DEVICE_ACCESS_STATUS_READONLY - The Device is available only for Read access (cannot be controlled).

LvDeviceAccessStatus_NoAccess Represents the GenTL DEVICE_ACCESS_STATUS_NOACCESS - The Device is not available either because it is already open or because it is not reachable.

5.30.2.20 enum LvDeviceClockSelector

Enum values for the LvDevice_DeviceClockSelector feature.

Enumerator

LvDeviceClockSelector_SensorDigitization Sensor digitization clock.

5.30.2.21 enum LvDeviceEndianessMechanism

Enum values for the LvDevice_DeviceEndianessMechanism feature.

Enumerator

LvDeviceEndianessMechanism_Legacy Legacy endianess handling mode, intended for GigE Vision remote devices using GenICam schema version 1.0.

LvDeviceEndianessMechanism_Standard Standard endianess handling mode, intended for GigE Vision remote devices using GenICam schema version 1.1 and newer.

5.30.2.22 enum LvDeviceScanType

Enum values for the LvDevice_DeviceScanType feature.

Enumerator

LvDeviceScanType_Areascan Indicates area scan sensor.

LvDeviceScanType_Linescan Indicates linr scan sensor.

5.30.2.23 enum LvDeviceTemperatureSelector

Enum values for the LvDevice_DeviceTemperatureSelector feature.

Enumerator

LvDeviceTemperatureSelector_Sensor Temperature on sensor

LvDeviceTemperatureSelector_Mainboard Temperature on main board

5.30.2.24 enum LvDeviceType

Enum values for the LvDevice_DeviceType feature.

Enumerator

LvDeviceType_Custom Device based on a custom technology.

LvDeviceType_GEV GigE Vision compatible device.

LvDeviceType_U3V USB3 Vision compatible device.

5.30.2.25 enum LvEventNotification

Enum values for the LvDevice_EventNotification feature.

Enumerator

LvEventNotification_Off The notifications for the selected event are deactivated.

LvEventNotification_On The notifications for the selected event are activated.

5.30.2.26 enum LvEventSelector

Enum values for the LvDevice_EventSelector feature.

Enumerator

LvEventSelector_LvLog This enumeration value selects the log event for configuration.

LvEventSelector_LvSmartAppLog This enumeration value selects the smart application log event for configuration.

LvEventSelector_LvSmartAppString This enumeration value selects the smart application string event for configuration.

LvEventSelector_LvSmartAppInt This enumeration value selects the smart application signed integer event for configuration.

LvEventSelector_LvSmartAppUint This enumeration value selects the smart application unsigned integer event for configuration.

LvEventSelector_LvSmartAppRegister This enumeration value selects the smart application raw register event for configuration.

LvEventSelector_LvTriggerDropped This enumeration value selects the dropped trigger event for configuration.

5.30.2.27 enum LvExposureAuto

Enum values for the LvDevice_ExposureAuto feature.

Enumerator

LvExposureAuto_Off Automatic exposure mode off - the automatic exposure is not applied.

LvExposureAuto_Once Automatic exposure mode once - the exposure time is once adjusted, then switches back to off.

LvExposureAuto_Continuous Automatic exposure mode continuous - the exposure time is continuously auto-adjusted.

5.30.2.28 enum LvExposureMode

Enum values for the LvDevice_ExposureMode feature.

Enumerator

LvExposureMode_Timed Timed exposure mode - the exposure time is controlled by corresponding feature.

5.30.2.29 enum LvExternalADCSelector

Enum values for the LvDevice_LvExternalADCSelector feature.

Enumerator

LvExternalADCSelector_ExternalADC1 Selects external ADC 1 for configuration.

LvExternalADCSelector_ExternalADC2 Selects external ADC 2 for configuration.

LvExternalADCSelector_ExternalADC3 Selects external ADC 3 for configuration.

LvExternalADCSelector_ExternalADC4 Selects external ADC 4 for configuration.

5.30.2.30 enum LvExternalDeviceControlMode

Enum values for the LvDevice_LvExternalDeviceControlMode feature.

Enumerator

LvExternalDeviceControlMode_Custom Selects the custom mode.

5.30.2.31 enum LvGainAuto

Enum values for the LvDevice_GainAuto feature.

Enumerator

LvGainAuto_Off Automatic gain mode off - the gain value is controlled 'manually'.

LvGainAuto_Once Automatic gain mode 'once' - the gain value is calculated and applied once and the feature switches back to 'off' (manual mode).

LvGainAuto_Continuous Continuous automatic gain mode - the AGC is applied continuously.

5.30.2.32 enum LvGainSelector

Enum values for the LvDevice_GainSelector feature.

Enumerator

LvGainSelector_All Apply gain on all channels and taps.

LvGainSelector_AnalogAll Apply analog gain.

LvGainSelector_DigitalAll Apply digital gain.

5.30.2.33 enum LvGevCCP

Enum values for the LvDevice_GevCCP feature.

Enumerator

LvGevCCP_OpenAccess Sets the control channel privilege feature to open.

LvGevCCP_ExclusiveAccess Sets the control channel privilege feature to exclusive.

LvGevCCP_ControlAccess Sets the control channel privilege feature to control.

LvGevCCP_ControlAccessSwitchoverActive Sets the control channel privilege feature to control with switchover active.

5.30.2.34 enum LvGevDeviceClass

Enum values for the LvDevice_GevDeviceClass feature.

Enumerator

LvGevDeviceClass_Transmitter Indicates that the device is a GigE Vision transmitter.

5.30.2.35 enum LvGevDeviceModeCharacterSet

Enum values for the LvDevice_GevDeviceModeCharacterSet feature.

Enumerator

LvGevDeviceModeCharacterSet_UTF8 Indicates that the camera uses the UTF8 character set.

5.30.2.36 enum LvGevDeviceStreamCaptureMode

Enum values for the LvDevice_LvGevDeviceStreamCaptureMode feature.

Enumerator

LvGevDeviceStreamCaptureMode_SystemDefault System default mode, configurable in the ini file.

LvGevDeviceStreamCaptureMode_Socket Socket mode, the GVSP stream is processed through the socket interface (regular operating system networks stack).

LvGevDeviceStreamCaptureMode_FilterDriver Filter driver mode, the GVSP stream is processed through the filter driver interface (bypassing operating system network stack).

5.30.2.37 enum LvGevIPConfigurationStatus

Enum values for the LvDevice_GevIPConfigurationStatus feature.

Enumerator

LvGevIPConfigurationStatus_None No IP configuration method was executed or it is not known.

LvGevIPConfigurationStatus_PersistentIP The current device IP configuration was obtained through persistent IP.

LvGevIPConfigurationStatus_DHCP The current device IP configuration was obtained through DHCP.

LvGevIPConfigurationStatus_LLA The current device IP configuration was obtained through LLA.

LvGevIPConfigurationStatus_ForceIP The current device IP configuration was obtained through ForceIP.

5.30.2.38 enum LvGevSCPDirection

Enum values for the LvDevice_GevSCPDirection feature.

Enumerator

LvGevSCPDirection_Transmitter Indicates that the stream channel is a transmitter.

5.30.2.39 enum LvGevSupportedOptionSelector

Enum values for the LvDevice_GevSupportedOptionSelector feature.

Enumerator

LvGevSupportedOptionSelector_IPConfigurationLLA Indicates whether the (first) network interface supports auto IP addressing (also known as LLA).

LvGevSupportedOptionSelector_IPConfigurationDHCP Indicates whether the (first) network interface supports DHCP IP addressing.

LvGevSupportedOptionSelector_IPConfigurationPersistentIP Indicates whether the (first) network interface supports fixed IP addressing (also known as persistent IP addressing).

LvGevSupportedOptionSelector_CommandsConcatenation Indicates whether command concatenation is supported by the device.

LvGevSupportedOptionSelector_WriteMem Indicates whether write memory scheme is supported by the device.

LvGevSupportedOptionSelector_PacketResend Indicates whether packet resendig is supported by the device.

LvGevSupportedOptionSelector_Event Indicates whether event (message channel) is supported by the device.

LvGevSupportedOptionSelector_EventData Indicates whether eventdata (message channel) is supported by the device.

LvGevSupportedOptionSelector_PendingAck Indicates whether pending acknowledge is supported by the device.

LvGevSupportedOptionSelector_Action Indicates whether action commands are supported by the device.

LvGevSupportedOptionSelector_PrimaryApplicationSwitchover Indicates whether primary application switchover is supported by the device.

LvGevSupportedOptionSelector_ExtendedStatusCodes Indicates whether extended GigE Vision status codes are supported by the device.

LvGevSupportedOptionSelector_DiscoveryAckDelayWritable Indicates whether writable discovery acknowledge delay is supported by the device.

LvGevSupportedOptionSelector_DiscoveryAckDelay Indicates whether discovery acknowledge delay is supported by the device.

LvGevSupportedOptionSelector_TestData Indicates whether test data is supported by the device.

LvGevSupportedOptionSelector_ManifestTable Indicates whether manifest table is supported by the device.

LvGevSupportedOptionSelector_CCPApplicationSocket Indicates whether the primary application port and IP address features are supported by the device.

LvGevSupportedOptionSelector_LinkSpeed Indicates whether link speed feature is supported by the device.

LvGevSupportedOptionSelector_HeartbeatDisable Indicates whether heartbeat disabling is supported by the device.

LvGevSupportedOptionSelector_SerialNumber Indicates whether serial number feature is supported by the device.

LvGevSupportedOptionSelector_UserDefinedName Indicates whether user defined name is supported by the device.

LvGevSupportedOptionSelector_StreamChannelSourceSocket Indicates whether the stream channel source port feature is supported by the device.

LvGevSupportedOptionSelector_StreamChannel0ExtendedChunkData Indicates whether the extended chunk data is supported by the device.

LvGevSupportedOptionSelector_StreamChannel0UnconditionalStreaming Indicates whether the unconditional streaming is supported by the device.

LvGevSupportedOptionSelector_StreamChannel0IPReassembly Indicates whether the reassembly of fragmented IP packets is supported by the device.

LvGevSupportedOptionSelector_StreamChannel0BigAndLittleEndian Indicates whether the big and little endian stream channel is supported by the device.

LvGevSupportedOptionSelector_MessageChannelSourceSocket Indicates whether the message channel source port feature is supported by the device.

5.30.2.40 enum LvImageStampSelector

Enum values for the LvDevice_LvImageStampSelector feature.

Enumerator

LvImageStampSelector_Timestamp Selects the flag controling reset of the image timestamp

LvImageStampSelector_FrameID Selects the flag controling reset of the image frame ID

5.30.2.41 enum LvInterfaceType

Enum values for the LvInterface_InterfaceType feature.

Enumerator

LvInterfaceType_Custom Interface supporting a custom technology devices.

LvInterfaceType_GEV Interface supporting GigE Vision devices.

LvInterfaceType_U3V Interface supporting USB3 Vision devices.

5.30.2.42 enum LvLensControlCalibrationStatus

Enum values for the LvDevice_LvLensControlCalibrationStatus feature.

Enumerator

LvLensControlCalibrationStatus_Invalid Current calibration parameters are invalid

LvLensControlCalibrationStatus_Valid Current calibration parameters are valid

5.30.2.43 enum LvLensControlTargetApproach

Enum values for the LvDevice_LvLensControlTargetApproach feature.

Enumerator

LvLensControlTargetApproach_Direct Approaches the target position directly, no matter from which side.

LvLensControlTargetApproach_FromPlus Approaches the target position always from the plus side to improve accuracy.

LvLensControlTargetApproach_FromMinus Approaches the target position always from the minus side to improve accuracy.

5.30.2.44 enum LvLineFormat

Enum values for the LvDevice_LineFormat feature.

Enumerator

LvLineFormat_NoConnect Not connected line.

LvLineFormat_TriState The Line is currently in Tri-state mode (Not driven).

LvLineFormat_TTL The Line is currently accepting or sending TTL level signals.

LvLineFormat_LVDS The Line is currently accepting or sending LVDS level signals.

LvLineFormat_RS422 The Line is currently accepting or sending RS-422 level signals.

LvLineFormat_OptoCoupled Optically isolated line (optocoupler).

5.30.2.45 enum LvLineMode

Enum values for the LvDevice_LineMode feature.

Enumerator

LvLineMode_Input The line is used as signal input.

LvLineMode_Output The line is used as signal output.

5.30.2.46 enum LvLineSelector

Enum values for the LvDevice_LineSelector feature.

Enumerator

LvLineSelector_Line1 Selects device's logical line 1 (optocoupler input).

LvLineSelector_Line2 Selects device's logical line 2 (optocoupler input).

LvLineSelector_Line3 Selects device's logical line 3 (optocoupler input).

LvLineSelector_Line4 Selects device's logical line 4 (optocoupler input).

LvLineSelector_Line5 Selects device's logical line 5.

LvLineSelector_Line6 Selects device's logical line 6.

LvLineSelector_Line7 Selects device's logical line 7.

LvLineSelector_Line8 Selects device's logical line 8.

LvLineSelector_Line9 Selects device's logical line 9 (optocoupler output).

LvLineSelector_Line10 Selects device's logical line 10 (optocoupler output).

LvLineSelector_Line11 Selects device's logical line 11 (optocoupler output).

LvLineSelector_Line12 Selects device's logical line 12 (optocoupler output).

LvLineSelector_Line13 Selects device's logical line 13.

LvLineSelector_Line14 Selects device's logical line 14.

LvLineSelector_Line15 Selects device's logical line 15.

LvLineSelector_Line16 Selects device's logical line 16.

LvLineSelector_Line17 Selects device's logical line 17 (TTL input).

LvLineSelector_Line18 Selects device's logical line 18 (TTL input).

LvLineSelector_Line19 Selects device's logical line 19.

LvLineSelector_Line20 Selects device's logical line 20.

LvLineSelector_Line21 Selects device's logical line 21.

LvLineSelector_Line22 Selects device's logical line 22.

LvLineSelector_Line23 Selects device's logical line 23.

LvLineSelector_Line24 Selects device's logical line 24.

LvLineSelector_Line25 Selects device's logical line 25 (TTL output).

LvLineSelector_Line26 Selects device's logical line 26 (TTL output).

LvLineSelector_Line27 Selects device's logical line 27.

LvLineSelector_Line28 Selects device's logical line 28.

LvLineSelector_Line29 Selects device's logical line 29.

LvLineSelector_Line30 Selects device's logical line 30.

LvLineSelector_Line31 Selects device's logical line 31.

LvLineSelector_Line32 Selects device's logical line 32.

5.30.2.47 enum LvLineSource

Enum values for the LvDevice_LineSource feature.

Enumerator

LvLineSource_Off Switches the line source off. This disables the line output (disconnects the line).

LvLineSource_ExposureActive Selects exposure active signal as line source.

LvLineSource_Timer1Active Selects timer 1 active signal as line source.

LvLineSource_Timer2Active Selects timer 2 active signal as line source.

LvLineSource_Timer3Active Selects timer 3 active signal as line source.

LvLineSource_Timer4Active Selects timer 4 active signal as line source.

LvLineSource_UserOutput1 Selects user output 1 value signal as line source.

LvLineSource_UserOutput2 Selects user output 2 value signal as line source.

LvLineSource_UserOutput3 Selects user output 3 value signal as line source.

LvLineSource_UserOutput4 Selects user output 4 value signal as line source.

LvLineSource_UserOutput5 Selects user output 5 value signal as line source.

LvLineSource_UserOutput6 Selects user output 6 value signal as line source.

LvLineSource_UserOutput7 Selects user output 7 value signal as line source.

LvLineSource_UserOutput8 Selects user output 8 value signal as line source.

LvLineSource_Counter1Active Selects counter 1 active signal as line source.

LvLineSource_Counter2Active Selects counter 2 active signal as line source.

LvLineSource_Counter3Active Selects counter 3 active signal as line source.

LvLineSource_Counter4Active Selects counter 4 active signal as line source.

5.30.2.48 enum LvLUTMode

Enum values for the LvDevice_LvLUTMode feature.

Enumerator

LvLUTMode_Direct In this mode the LUT is controlled directly.

LvLUTMode_BalanceWhite In this mode the LUT is controlled through the higher level features, such as brightness, contrast, gamma or white balance.

5.30.2.49 enum LvLUTSelector

Enum values for the LvDevice_LUTSelector feature.

Enumerator

LvLUTSelector_Luminance Selects the luminance LUT for configuration.

LvLUTSelector_Red Selects the red LUT for configuration.

LvLUTSelector_Green Selects the green LUT for configuration.

LvLUTSelector_Blue Selects the blue LUT for configuration.

5.30.2.50 enum LvPixelFormat

LvPixelFormat constants Enum for the LvDevice_PixelFormat, LvDevice_ChunkPixelFormat and LvDevice_LvUni↩ PixelFormat features. The Pixel format constants are defined by the GenICam standard. The value consists of 3 parts:

- byte 1 - color/mono/custom
- byte 2 - bits per pixel
- byte 3 and 4 - ID of the pixel format
  Exceptions are:
- LvPixelFormat_BGR555Packed - used only in the Image Processing Library (in conversions for display).

Enumerator

LvPixelFormat_Mono8 Monochrome 8-bit. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x0001).

LvPixelFormat_Mono8S Monochrome 8-bit signed. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY8BIT | 0x0002).

LvPixelFormat_Mono10 Monochrome 10-bit. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0003).

LvPixelFormat_Mono10Packed Monochrome 10-bit packed. Defined as (LV_PIX_MONO | LV_PIX_OCC↩ UPY12BIT | 0x0004).

LvPixelFormat_Mono12 Monochrome 12-bit. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0005).

LvPixelFormat_Mono12Packed Monochrome 12-bit packed. Defined as (LV_PIX_MONO | LV_PIX_OCC↩ UPY12BIT | 0x0006).

LvPixelFormat_Mono14 Monochrome 14-bit. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0025).

LvPixelFormat_Mono16 Monochrome 16-bit. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0007).

LvPixelFormat_BayerGR8 Undecoded 8-bit Bayer array with the GR array position. Defined as (LV_PIX_↩ MONO | LV_PIX_OCCUPY8BIT | 0x0008).

LvPixelFormat_BayerRG8 Undecoded 8-bit Bayer array with the RG array position. Defined as (LV_PIX_↩ MONO | LV_PIX_OCCUPY8BIT | 0x0009).

LvPixelFormat_BayerGB8 Undecoded 8-bit Bayer array with the GB array position. Defined as (LV_PIX_↩ MONO | LV_PIX_OCCUPY8BIT | 0x000A).

LvPixelFormat_BayerBG8 Undecoded 8-bit Bayer array with the BG array position. Defined as (LV_PIX_↩ MONO | LV_PIX_OCCUPY8BIT | 0x000B).

LvPixelFormat_BayerGR10 Undecoded 10-bit Bayer array with the GR array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x000C).

LvPixelFormat_BayerRG10 Undecoded 10-bit Bayer array with the RG array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x000D).

LvPixelFormat_BayerGB10 Undecoded 10-bit Bayer array with the GB array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x000E).

LvPixelFormat_BayerBG10 Undecoded 10-bit Bayer array with the BG array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x000F).

LvPixelFormat_BayerGR12 Undecoded 12-bit Bayer array with the GR array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x0010).

LvPixelFormat_BayerRG12 Undecoded 12-bit Bayer array with the RG array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x0011).

LvPixelFormat_BayerGB12 Undecoded 12-bit Bayer array with the GB array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x0012).

LvPixelFormat_BayerBG12 Undecoded 12-bit Bayer array with the BG array position. Defined as (LV_PI↩ X_MONO | LV_PIX_OCCUPY16BIT | 0x0013).

LvPixelFormat_BayerGR10Packed Undecoded 10-bit packed Bayer array with the GR array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x0026).

LvPixelFormat_BayerRG10Packed Undecoded 10-bit packed Bayer array with the RG array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x0027).

LvPixelFormat_BayerGB10Packed Undecoded 10-bit packed Bayer array with the GB array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x0028).

LvPixelFormat_BayerBG10Packed Undecoded 10-bit packed Bayer array with the BG array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x0029).

LvPixelFormat_BayerGR12Packed Undecoded 12-bit packed Bayer array with the GR array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x002A).

LvPixelFormat_BayerRG12Packed Undecoded 12-bit packed Bayer array with the RG array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x002B).

LvPixelFormat_BayerGB12Packed Undecoded 12-bit packed Bayer array with the GB array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x002C).

LvPixelFormat_BayerBG12Packed Undecoded 10-bit packed Bayer array with the BG array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY12BIT | 0x002D).

LvPixelFormat_BayerGR16 Undecoded 16-bit Bayer array with the GR array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x002E).

LvPixelFormat_BayerRG16 Undecoded 16-bit Bayer array with the RG array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x002F).

LvPixelFormat_BayerGB16 Undecoded 16-bit Bayer array with the GB array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0030).

LvPixelFormat_BayerBG16 Undecoded 16-bit Bayer array with the BG array position. Defined as (LV_PIX_MONO | LV_PIX_OCCUPY16BIT | 0x0031).

LvPixelFormat_RGB8 RGB 24-bit packed (3x8 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY24BIT | 0x0014).

LvPixelFormat_BGR8 BGR 24-bit packed (3x8 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY24BIT | 0x0015).

LvPixelFormat_RGBA8 RGB 32-bit packed (3x8 bits + 1x8 bits alpha). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY32BIT | 0x0016).

LvPixelFormat_BGRA8 BGR 32-bit packed (3x8 bits + 1x8 bits alpha). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY32BIT | 0x0017).

LvPixelFormat_RGB10 RGB 48-bit packed (3x10 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x0018).

LvPixelFormat_BGR10 BGR 48-bit packed (3x10 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x0019).

LvPixelFormat_RGB12 RGB 48-bit packed (3x12 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x001A).

LvPixelFormat_BGR12 BGR 48-bit packed (3x12 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x001B).

LvPixelFormat_RGB16 RGB 48-bit packed (3x16 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x0033).

LvPixelFormat_RGB10V1Packed RGB 32-bit packed (3x10 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY32BIT | 0x001C).

LvPixelFormat_RGB10P32 RGB 32-bit packed (3x10 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY32BIT | 0x001D).

LvPixelFormat_RGB12V1Packed RGB 36-bit packed (3x12 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY36BIT | 0x0034).

LvPixelFormat_RGB565P RGB 16-bit packed (5,6,5 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x0035).

LvPixelFormat_BGR565P BGR 16-bit packed (5,6,5 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x0036).

LvPixelFormat_YUV411_8 YUV 4-1-1 Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY12BIT | 0x001E).

LvPixelFormat_YUV422_8_UYVY YUV 4-2-2 UYVY Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x001F).

LvPixelFormat_YUV8 YUV 4-4-4 Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY24BIT | 0x0020).

LvPixelFormat_YUV422_8 YUV 4-2-2 YUYV Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x0032).

LvPixelFormat_YCbCr422_8 YCbCr 4-2-2 Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x003B).

LvPixelFormat_YCbCr601_422_8 YCbCr 4-2-2 Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x003E).

LvPixelFormat_YCbCr601_422_8_CbYCrY YCbCr 4-2-2 Packed. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x0044).

LvPixelFormat_RGB8_Planar RGB 8-bit planar. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY24BIT | 0x0021).

LvPixelFormat_RGB10_Planar RGB 10-bit planar. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x0022).

LvPixelFormat_RGB12_Planar RGB 12-bit planar. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x0023).

LvPixelFormat_RGB16_Planar RGB 16-bit planar. Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY48BIT | 0x0024).

LvPixelFormat_BGR555P RGB 15-bit packed (3x5 bits). Defined as (LV_PIX_COLOR | LV_PIX_OCCUPY16BIT | 0x00E1). Not a standard GenICam format, used only in the image processing library.

5.30.2.51 enum LvPowerSwitchBoundADC

Enum values for the LvDevice_LvPowerSwitchBoundADC feature.

Enumerator

LvPowerSwitchBoundADC_None Binds no external ADC to the power switch

LvPowerSwitchBoundADC_ExternalADC1 Binds external ADC 1 to the power switch

LvPowerSwitchBoundADC_ExternalADC2 Binds external ADC 2 to the power switch

LvPowerSwitchBoundADC_ExternalADC3 Binds external ADC 3 to the power switch

LvPowerSwitchBoundADC_ExternalADC4 Binds external ADC 4 to the power switch

5.30.2.52 enum LvPowerSwitchCurrentAction

Enum values for the LvDevice_LvPowerSwitchCurrentAction feature.

Enumerator

LvPowerSwitchCurrentAction_Idle Reports that all power switches are idle

LvPowerSwitchCurrentAction_Pulse Reports that a pulse command is pending

LvPowerSwitchCurrentAction_Calibrate Reports that a calibration is pending

LvPowerSwitchCurrentAction_AdjustPosition Reports that a position adjustment is pending

LvPowerSwitchCurrentAction_Drive Reports that a power switch drive operation is pending

5.30.2.53 enum LvPowerSwitchDrive

Enum values for the LvDevice_LvPowerSwitchDrive feature.

Enumerator

LvPowerSwitchDrive_Off Switches the selected power switch off.

LvPowerSwitchDrive_Plus Sswitches the selected power switch to plus polarity.

LvPowerSwitchDrive_Minus Switches the selected power switch to minus polarity.

5.30.2.54 enum LvPowerSwitchDriveAll

Enum values for the LvPowerSwitchDriveAll feature.

Enumerator

LvPowerSwitchDriveAll_Off Switches the active power switches off

LvPowerSwitchDriveAll_Plus Switches the active power switches to plus polarity

LvPowerSwitchDriveAll_Minus Switches the active power switches to minus polarity

5.30.2.55 enum LvPowerSwitchSelector

Enum values for the LvDevice_LvPowerSwitchSelector feature.

Enumerator

LvPowerSwitchSelector_PowerSwitch1 Selects power switch 1 for configuration.

LvPowerSwitchSelector_PowerSwitch2 Selects power switch 2 for configuration.

LvPowerSwitchSelector_PowerSwitch3 Selects power switch 3 for configuration.

LvPowerSwitchSelector_PowerSwitch4 Selects power switch 4 for configuration.

5.30.2.56 enum LvRegionSelector

Enum values for the LvDevice_RegionSelector feature.

Enumerator

LvRegionSelector_Region0 Selects region 0 for configuration.

LvRegionSelector_Region1 Selects region 1 for configuration.

LvRegionSelector_Region2 Selects region 2 for configuration.

LvRegionSelector_Region3 Selects region 3 for configuration.

5.30.2.57 enum LvRenderType

Enum values for the LvRenderer_LvRenderType feature.

Enumerator

LvRenderType_FullSize Renders the acquired image in full size.

LvRenderType_ScaleToFit Renders the acquired image to fit into the window.

LvRenderType_ScaleToSize Renders the acquired image scaled to required size.

LvRenderType_ScaleToTiles Renders the acquired images in tiles.

5.30.2.58 enum LvSerialPortBaudRate

Enum values for the LvDevice_LvSerialPortBaudRate feature.

Enumerator

LvSerialPortBaudRate_Baud2400 Baud rate of 2400 bauds.

LvSerialPortBaudRate_Baud4800 Baud rate of 4800 bauds.

LvSerialPortBaudRate_Baud9600 Baud rate of 9600 bauds.

LvSerialPortBaudRate_Baud14400 Baud rate of 14400 bauds.

LvSerialPortBaudRate_Baud19200 Baud rate of 19200 bauds.

LvSerialPortBaudRate_Baud38400 Baud rate of 38400 bauds.

LvSerialPortBaudRate_Baud57600 Baud rate of 57600 bauds.

LvSerialPortBaudRate_Baud115200 Baud rate of 115200 bauds.

5.30.2.59 enum LvSerialPortCommandStatus

Enum values for the LvDevice_LvSerialPortCommandStatus feature.

Enumerator

LvSerialPortCommandStatus_Success Last command was successfully transferred.

LvSerialPortCommandStatus_Timeout Last command ended with timeout (depending on configuration this might be problem or not).

LvSerialPortCommandStatus_PortBusy Last command failed: port busy.

LvSerialPortCommandStatus_CommunicationError Last command failed: generic communication error.

LvSerialPortCommandStatus_FrameError Last command failed: frame error.

LvSerialPortCommandStatus_ParityError Last command failed: parity error.

LvSerialPortCommandStatus_Overflow Last command failed: overflow.

5.30.2.60 enum LvSerialPortDataBits

Enum values for the LvDevice_LvSerialPortDataBits feature.

Enumerator

LvSerialPortDataBits_DataBits7 7 data bits per character.

LvSerialPortDataBits_DataBits8 8 data bits per character.

5.30.2.61 enum LvSerialPortParity

Enum values for the LvDevice_LvSerialPortParity feature.

Enumerator

LvSerialPortParity_None Parity method 'none', parity bit not used.

LvSerialPortParity_Odd Parity method 'odd', odd number of set bits in each character.

LvSerialPortParity_Even Parity method 'even', even number of set bits in each character.

5.30.2.62 enum LvSerialPortStopBits

Enum values for the LvDevice_LvSerialPortStopBits feature.

Enumerator

LvSerialPortStopBits_StopBits1 1 stop bit per character.

LvSerialPortStopBits_StopBits1dot5 1.5 stop bit per character.

LvSerialPortStopBits_StopBits2 2 stop bits per character.

5.30.2.63 enum LvSpecialPurposeTriggerActivation

Enum values for the LvDevice_LvSpecialPurposeTriggerActivation feature.

Enumerator

LvSpecialPurposeTriggerActivation_RisingEdge Selects the trigger signal's rising edge as active.

LvSpecialPurposeTriggerActivation_FallingEdge Selects the trigger signal's falling edge as active

5.30.2.64 enum LvSpecialPurposeTriggerSelector

Enum values for the LvDevice_LvSpecialPurposeTriggerSelector feature.

Enumerator

LvSpecialPurposeTriggerSelector_ImageStampsReset Timestamps reset trigger - controls reset of timestamp, frame ID and other image stamps.

5.30.2.65 enum LvSpecialPurposeTriggerSource

Enum values for the LvDevice_LvSpecialPurposeTriggerSource feature.

Enumerator

LvSpecialPurposeTriggerSource_Off Sets trigger source off - it can be still be issued by an explicit software trigger

LvSpecialPurposeTriggerSource_Line1 Sets the signal source for the selected trigger to line 1 (optocoupler input).

LvSpecialPurposeTriggerSource_Line2 Sets the signal source for the selected trigger to line 2 (optocoupler input).

LvSpecialPurposeTriggerSource_Line3 Sets the signal source for the selected trigger to line 3 (optocoupler input).

LvSpecialPurposeTriggerSource_Line4 Sets the signal source for the selected trigger to line 4 (optocoupler input).

LvSpecialPurposeTriggerSource_Line5 Sets the signal source for the selected trigger to line 5.

LvSpecialPurposeTriggerSource_Line6 Sets the signal source for the selected trigger to line 6.

LvSpecialPurposeTriggerSource_Line7 Sets the signal source for the selected trigger to line 7.

LvSpecialPurposeTriggerSource_Line8 Sets the signal source for the selected trigger to line 8.

LvSpecialPurposeTriggerSource_Line17 Sets the signal source for the selected trigger to line 17 (TTL input).

LvSpecialPurposeTriggerSource_Line18 Sets the signal source for the selected trigger to line 18 (TTL input).

LvSpecialPurposeTriggerSource_Line19 Sets the signal source for the selected trigger to line 19.

LvSpecialPurposeTriggerSource_Line20 Sets the signal source for the selected trigger to line 20.

LvSpecialPurposeTriggerSource_Line21 Sets the signal source for the selected trigger to line 21.

LvSpecialPurposeTriggerSource_Line22 Sets the signal source for the selected trigger to line 22.

LvSpecialPurposeTriggerSource_Line23 Sets the signal source for the selected trigger to line 23.

LvSpecialPurposeTriggerSource_Line24 Sets the signal source for the selected trigger to line 24.

LvSpecialPurposeTriggerSource_Action1 Sets the signal source for the selected trigger to action signal 1.

LvSpecialPurposeTriggerSource_Action2 Sets the signal source for the selected trigger to action signal 2.

LvSpecialPurposeTriggerSource_Action3 Sets the signal source for the selected trigger to action signal 3.

LvSpecialPurposeTriggerSource_Action4 Sets the signal source for the selected trigger to action signal 4.

LvSpecialPurposeTriggerSource_Action5 Sets the signal source for the selected trigger to action signal 5.

LvSpecialPurposeTriggerSource_Action6 Sets the signal source for the selected trigger to action signal 6.

LvSpecialPurposeTriggerSource_Action7 Sets the signal source for the selected trigger to action signal 7.

LvSpecialPurposeTriggerSource_Action8 Sets the signal source for the selected trigger to action signal 8.

5.30.2.66 enum LvStreamAcquisitionModeSelector

Enum values for the LvStream_StreamAcquisitionModeSelector feature.

Enumerator

LvStreamAcquisitionModeSelector_Default Default acquisition mode.

5.30.2.67 enum LvStreamType

Enum values for the LvStream_StreamType feature.

Enumerator

LvStreamType_Custom Stream belonging to a custom technology device.

LvStreamType_GEV Stream belonging to a GigE Vision compatible device.

LvStreamType_U3V Stream belonging to a USB3 Vision compatible device.

5.30.2.68 enum LvStrobeDropMode

Enum values for the LvDevice_LvStrobeDropMode feature.

Enumerator

LvStrobeDropMode_DropStrobe Strobe drop mode 'drop' - the strobe is dropped, image is acquired without the strobe.

LvStrobeDropMode_DelayFrame Strobe drop mode 'delay' - the frame acquisition is delayed, until the strobe can be issued.

5.30.2.69 enum LvStrobeDurationMode

Enum values for the LvDevice_LvStrobeDurationMode feature.

Enumerator

LvStrobeDurationMode_FrameRateRelated The maximum strobe duration depends on the maximum frame rate of the camera. For very fast sensors the max. strobe duration time, dependent on the specification of the LEDs used, cannot be applied in full length, as the recovery time may become too short. The calculation is done automatically depending on the LEDs used and the max. frame rate of the camera in its actual mode of operation. Such calculation also includes boosted frame rates e.g. when the camera is in partial scanning and/or binning mode.

LvStrobeDurationMode_Free The maximum strobe duration depends on the maximum allowed ON-time and the minimum required recovery time of the LEDs used. The user can program the strobe duration free, according to his request, but must be aware himself about the relation of strobe ON-time and recovery time. An automatic protection circuit in HW drops a strobe in case the proper relation of ON-time to recovery time is not guaranteed. In such case a related error code is returned by the SW (frame message or otherwise)

5.30.2.70 enum LvStrobeEnable

Enum values for the LvDevice_LvStrobeEnable feature.

Enumerator

LvStrobeEnable_Off Switches the strobe off.

LvStrobeEnable_AllClusters Switches on all LED clusters of the strobe light. On strobe lights possessing just a single LED cluster this cluster is switche on.

LvStrobeEnable_LEDCluster1 Switches on LED cluster 1 only. The strobe will use just the LEDs in this cluster.

LvStrobeEnable_LEDCluster2 Switches on LED cluster 2 only. The strobe will use just the LEDs in this cluster.

5.30.2.71 enum LvTimerSelector

Enum values for the LvDevice_TimerSelector feature.

Enumerator

LvTimerSelector_Timer1 Selects timer 1 for configuration.

LvTimerSelector_Timer2 Selects timer 2 for configuration.

LvTimerSelector_Timer3 Selects timer 3 for configuration.

LvTimerSelector_Timer4 Selects timer 4 for configuration.

5.30.2.72 enum LvTimerTriggerSource

Enum values for the LvDevice_TimerTriggerSource feature.

Enumerator

LvTimerTriggerSource_Off Switches timer trigger signal off - no signal will be firing the timer

LvTimerTriggerSource_FrameTrigger Switches timer trigger signal to frame trigger - activation of the frame trigger internal signal (before counting down eventual trigger delay) activates the timer.

LvTimerTriggerSource_Counter1End Switches timer trigger signal to counter 1 end - expiration of counter 1 activates the timer.

LvTimerTriggerSource_Counter2End Switches timer trigger signal to counter 2 end - expiration of counter 2 activates the timer.

LvTimerTriggerSource_Counter3End Switches timer trigger signal to counter 3 end - expiration of counter 3 activates the timer.

LvTimerTriggerSource_Counter4End Switches timer trigger signal to counter 4 end - expiration of counter 4 activates the timer.

LvTimerTriggerSource_UserOutput1 Switches timer trigger signal to user output 1 - activation of user output 1 activates the timer.

LvTimerTriggerSource_UserOutput2 Switches timer trigger signal to user output 2 - activation of user output 2 activates the timer.

LvTimerTriggerSource_UserOutput3 Switches timer trigger signal to user output 3 - activation of user output 3 activates the timer.

LvTimerTriggerSource_UserOutput4 Switches timer trigger signal to user output 4 - activation of user output 4 activates the timer.

LvTimerTriggerSource_UserOutput5 Switches timer trigger signal to user output 5 - activation of user output 5 activates the timer.

LvTimerTriggerSource_UserOutput6 Switches timer trigger signal to user output 6 - activation of user output 6 activates the timer.

LvTimerTriggerSource_UserOutput7 Switches timer trigger signal to user output 7 - activation of user output 7 activates the timer.

LvTimerTriggerSource_UserOutput8 Switches timer trigger signal to user output 8 - activation of user output 8 activates the timer.

5.30.2.73 enum LvTLType

Enum values for the LvSystem_TLType feature.

Enumerator

LvTLType_Mixed GenTL producer supporting mixed technologies.

LvTLType_Custom GenTL producer supporting a custom technology devices.

LvTLType_GEV GenTL producer supporting GigE Vision devices.

5.30.2.74 enum LvTriggerActivation

Enum values for the LvDevice_TriggerActivation feature.

Enumerator

LvTriggerActivation_RisingEdge Selects the trigger signal's rising edge as active.

LvTriggerActivation_FallingEdge Selects the trigger signal's falling edge as active

5.30.2.75 enum LvTriggerCaching

Enum values for the LvDevice_LvTriggerCaching feature.

Enumerator

LvTriggerCaching_Cache Trigger caching mode 'cache' - early triggers are cached and applied as soon as possible.

LvTriggerCaching_Drop Trigger caching mode 'cache' - early triggers are dropped

5.30.2.76 enum LvTriggerMode

Enum values for the LvDevice_TriggerMode feature.

Enumerator

LvTriggerMode_Off Trigger mode off - disables selected trigger

LvTriggerMode_On Trigger mode on - enables selected trigger.

5.30.2.77 enum LvTriggerSelector

Enum values for the LvDevice_TriggerSelector feature.

Enumerator

LvTriggerSelector_FrameStart Frame start trigger - controls a new frame acquisition.

LvTriggerSelector_FrameBurstStart Frame burst start trigger - Selects a trigger starting the capture of the bursts of frames.

5.30.2.78 enum LvTriggerSource

Enum values for the LvDevice_TriggerSource feature.

Enumerator

LvTriggerSource_Line1 Sets the signal source for the selected trigger to line 1 (optocoupler input).

LvTriggerSource_Line2 Sets the signal source for the selected trigger to line 2 (optocoupler input).

LvTriggerSource_Line3 Sets the signal source for the selected trigger to line 3 (optocoupler input).

LvTriggerSource_Line4 Sets the signal source for the selected trigger to line 4 (optocoupler input).

LvTriggerSource_Line5 Sets the signal source for the selected trigger to line 5.

LvTriggerSource_Line6 Sets the signal source for the selected trigger to line 6.

LvTriggerSource_Line7 Sets the signal source for the selected trigger to line 7.

LvTriggerSource_Line8 Sets the signal source for the selected trigger to line 8.

LvTriggerSource_Line17 Sets the signal source for the selected trigger to line 17 (TTL input).

LvTriggerSource_Line18 Sets the signal source for the selected trigger to line 18 (TTL input).

LvTriggerSource_Line19 Sets the signal source for the selected trigger to line 19.

LvTriggerSource_Line20 Sets the signal source for the selected trigger to line 20.

LvTriggerSource_Line21 Sets the signal source for the selected trigger to line 21.

LvTriggerSource_Line22 Sets the signal source for the selected trigger to line 22.

LvTriggerSource_Line23 Sets the signal source for the selected trigger to line 23.

LvTriggerSource_Line24 Sets the signal source for the selected trigger to line 24.

LvTriggerSource_Software Sets the signal source for the selected trigger to software.

LvTriggerSource_Action1 Sets the signal source for the selected trigger to action signal 1.

LvTriggerSource_Action2 Sets the signal source for the selected trigger to action signal 2.

LvTriggerSource_Action3 Sets the signal source for the selected trigger to action signal 3.

LvTriggerSource_Action4 Sets the signal source for the selected trigger to action signal 4.

LvTriggerSource_Action5 Sets the signal source for the selected trigger to action signal 5.

LvTriggerSource_Action6 Sets the signal source for the selected trigger to action signal 6.

LvTriggerSource_Action7 Sets the signal source for the selected trigger to action signal 7.

LvTriggerSource_Action8 Sets the signal source for the selected trigger to action signal 8.

LvTriggerSource_Quad Sets the signal source for the selected trigger to quadrature decoder.

LvTriggerSource_Counter1 Sets the signal source for the selected trigger to counter 1.

LvTriggerSource_Counter2 Sets the signal source for the selected trigger to counter 2.

LvTriggerSource_Counter3 Sets the signal source for the selected trigger to counter 3.

LvTriggerSource_Counter4 Sets the signal source for the selected trigger to counter 4.

LvTriggerSource_Timer1 Sets the signal source for the selected trigger to timer 1.

LvTriggerSource_Timer2 Sets the signal source for the selected trigger to timer 2.

LvTriggerSource_Timer3 Sets the signal source for the selected trigger to timer 3.

LvTriggerSource_Timer4 Sets the signal source for the selected trigger to timer 4.

5.30.2.79 enum LvUniBalanceRatioSelector

Enum values for the LvDevice_LvUniBalanceRatioSelector feature.

Enumerator

LvUniBalanceRatioSelector_Red Selects the red channel for configuration.

LvUniBalanceRatioSelector_Green Selects the green channel for configuration.

LvUniBalanceRatioSelector_Blue Selects the blue channel for configuration.

5.30.2.80 enum LvUniBalanceWhiteAuto

Enum values for the LvDevice_LvUniBalanceWhiteAuto feature.

Enumerator

LvUniBalanceWhiteAuto_Off Automatic white balance mode off - the automatic white balance is not applied.

LvUniBalanceWhiteAuto_Once Automatic white balance mode once - the white balance factors are once adjusted, then switches back to off.

5.30.2.81 enum LvUniColorTransformationMode

Enum values for the LvDevice_LvUniColorTransformationMode feature.

Enumerator

LvUniColorTransformationMode_Direct In this mode the Color Transformation matrix can be controlled directly.

LvUniColorTransformationMode_Generated In this mode the Color Transformation matrix is set through the higher level features, such as the Saturation.

5.30.2.82 enum LvUniColorTransformationSelector

Enum values for the LvDevice_LvUniColorTransformationSelector feature.

Enumerator

LvUniColorTransformationSelector_RGBtoRGB RGB to RGB matrix transformation - currently the only Color Transformation matrix type.

5.30.2.83 enum LvUniColorTransformationValueSelector

Enum values for the LvDevice_LvUniColorTransformationValueSelector feature.

Enumerator

LvUniColorTransformationValueSelector_Gain00 Selects the gain 00 (RR, red-red) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain01 Selects the gain 01 (RG, red-green) entry of the color-transformation matrix.

LvUniColorTransformationValueSelector_Gain02 Selects the gain 02 (RB, red-blue) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain10 Selects the gain 10 (GR, green-red) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain11 Selects the gain 11 (GG, green-green) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain12 Selects the gain 12 (GB, green-blue) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain20 Selects the gain 20 (BR, blue-red) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain21 Selects the gain 21 (BG, blue-green) entry of the color transformation matrix.

LvUniColorTransformationValueSelector_Gain22 Selects the gain 22 (BB, blue-blue) entry of the color transformation matrix.

5.30.2.84 enum LvUniLUTMode

Enum values for the LvDevice_LvUniLUTMode feature.

Enumerator

LvUniLUTMode_Direct In this mode the LUT is controlled directly.

LvUniLUTMode_Generated In this mode the LUT is controlled through the higher level features, such as brightness, contrast, gamma or white balance.

5.30.2.85 enum LvUniLUTSelector

Enum values for the LvDevice_LvUniLUTSelector feature.

Enumerator

LvUniLUTSelector_Luminance Selects the luminance LUT for configuration.

LvUniLUTSelector_Red Selects the red LUT for configuration.

LvUniLUTSelector_Green Selects the green LUT for configuration.

LvUniLUTSelector_Blue Selects the blue LUT for configuration.

5.30.2.86 enum LvUniProcessExecution

Enum values for the LvDevice_LvUniProcessExecution feature.

Enumerator

LvUniProcessExecution_OnBufferPtrQuery The SW image processing is delayed to the time the application asks for the LvBuffer_UniBase or LvBuffer_ProcessBase pointer or for the LvipImgInfo data. This enables to the application to skip the processing in case it is not needed. If this is queried several times for the same image, the processing is done only once.

LvUniProcessExecution_OnPopFromQueue The SW image processing is done always - at the moment the buffer is popped from the output buffer queue, before delivering it to the application.

LvUniProcessExecution_OnExplicitRequest The SW processing is not done automatically, but must be explicitly done by the ExecProcess command of the Buffer.

5.30.2.87 enum LvUniProcessMode

Enum values for the LvDevice_LvUniProcessMode feature.

Enumerator

LvUniProcessMode_HwOnly HwOnly - The processing is done only in case it is available directly on the hardware (device). The images will be delivered to the output buffer queue already processed.

LvUniProcessMode_SwOnly SwOnly - The processing will be done by software even if the hardware could support the operation. The software processing is done when the buffer is passed to the output buffer queue (or later - see LvUniProcessExecution).

LvUniProcessMode_Auto Auto - The processing will be done by hardware and by software will be processed only the part, which is not possible to do on hardware. Note that if the Bayer decoding is done by software (this happens when you select an undecoded Bayer pixel format as the device PixelFormat), the LUT must be then also done by software, even if it is available in hardware; that's because it must be applied after the Bayer decoding.

LvUniProcessMode_Off Off - The automatic processing is not available. You can use the HW features (LUT etc.) directly.

5.30.2.88 enum LvUserOutputSelector

Enum values for the LvDevice_UserOutputSelector feature.

Enumerator

LvUserOutputSelector_UserOutput1 Selects user output 1.

LvUserOutputSelector_UserOutput2 Selects user output 2.

LvUserOutputSelector_UserOutput3 Selects user output 3.

LvUserOutputSelector_UserOutput4 Selects user output 4.

LvUserOutputSelector_UserOutput5 Selects user output 5.

LvUserOutputSelector_UserOutput6 Selects user output 6.

LvUserOutputSelector_UserOutput7 Selects user output 7.

LvUserOutputSelector_UserOutput8 Selects user output 8.

5.30.2.89 enum LvUserSetDefaultSelector

Enum values for the LvDevice_UserSetDefaultSelector feature.

Enumerator

LvUserSetDefaultSelector_Default Selects the default user set as the default startup set.

LvUserSetDefaultSelector_UserSet1 Selects user set 1 as the default startup set.

LvUserSetDefaultSelector_UserSet2 Selects user set 2 as the default startup set.

LvUserSetDefaultSelector_UserSet3 Selects user set 3 as the default startup set.

LvUserSetDefaultSelector_UserSet4 Selects user set 4 as the default startup set.

LvUserSetDefaultSelector_None When resetting/connecting the camera, no user set is applied, the last camera configuration remains. During camera boot, the default user set is applied.

5.30.2.90 enum LvUserSetSelector

Enum values for the LvDevice_UserSetSelector feature.

Enumerator

LvUserSetSelector_Default Selects the default configuration set.

LvUserSetSelector_UserSet1 Selects user set 1.

LvUserSetSelector_UserSet2 Selects user set 2.

LvUserSetSelector_UserSet3 Selects user set 3.

LvUserSetSelector_UserSet4 Selects user set 4.

5.31 LvStreamStart() flags definitions

Macros

- #define LvStreamStartFlags_Default

5.31.1 Detailed Description

5.31.2 Macro Definition Documentation

5.31.2.1 #define LvStreamStartFlags_Default

Default stream start flag

5.32 LvStreamStop() flags definitions

Macros

- #define LvStreamStopFlags_Default
- #define LvStreamStopFlags_Kill

5.32.1 Detailed Description

5.32.2 Macro Definition Documentation

5.32.2.1 #define LvStreamStopFlags_Default

Stop the acquisition engine when the currently running tasks like filling a buffer are completed. This is the default.

5.32.2.2 #define LvStreamStopFlags_Kill

Stop the acquisition engine immediately and leave buffers currently being filled in the Input Buffer Pool.

5.33 LvDeviceUniSetLut() and LvDeviceUniGetLut() flags definitions

Macros

- #define LvUniLutFlags_HwLut

5.33.1 Detailed Description

5.33.2 Macro Definition Documentation

5.33.2.1 #define LvUniLutFlags_HwLut

If present, the operation is done directly on HW LUT, passing the UniProcess mechanism.

5.34 LvSaveFlag definitions

Macros

- #define LvSaveFlag_RemoteFtr
- #define LvSaveFlag_LocalFtr
- #define LvSaveFlag_GenTlFtr
- #define LvSaveFlag_All
- #define LvSaveFlag_IgnoreVersion
- #define LvSaveFlag_IgnoreModel

5.34.1 Detailed Description

5.34.2 Macro Definition Documentation

5.34.2.1 #define LvSaveFlag_All

Save/load device all features (combines all flags above).

5.34.2.2 #define LvSaveFlag_GenTlFtr

Save/load device GenTL XML features.

5.34.2.3 #define LvSaveFlag_IgnoreModel

If specified, the remote device model check is not done when reading the file - the file is read even if it was created by different device model (this may lead to errors by some features).

5.34.2.4 #define LvSaveFlag_IgnoreVersion

If specified, the remote device FW version check is not done when reading the file - the file is read even if it was created by device with a different FW version (this may lead to errors by some features).

5.34.2.5 #define LvSaveFlag_LocalFtr

Save/load device local XML features.

5.34.2.6 #define LvSaveFlag_RemoteFtr

Save/load device remote XML features.

5.35 LvPixelFormat definitions

Macros

- #define LV_PIX_MONO
- #define LV_PIX_COLOR
- #define LV_PIX_CUSTOM
- #define LV_PIX_COLOR_MASK
- #define LV_PIX_OCCUPY8BIT
- #define LV_PIX_OCCUPY12BIT
- #define LV_PIX_OCCUPY16BIT
- #define LV_PIX_OCCUPY24BIT
- #define LV_PIX_OCCUPY32BIT
- #define LV_PIX_OCCUPY36BIT
- #define LV_PIX_OCCUPY48BIT
- #define LV_PIX_EFFECTIVE_PIXEL_SIZE_MASK
- #define LV_PIX_EFFECTIVE_PIXEL_SIZE_SHIFT
- #define LvPixelFormat_Mono8Signed
- #define LvPixelFormat_RGB8Packed
- #define LvPixelFormat_BGR8Packed
- #define LvPixelFormat_RGBA8Packed
- #define LvPixelFormat_BGRA8Packed
- #define LvPixelFormat_RGB10Packed
- #define LvPixelFormat_BGR10Packed
- #define LvPixelFormat_RGB12Packed
- #define LvPixelFormat_BGR12Packed
- #define LvPixelFormat_RGB16Packed
- #define LvPixelFormat_RGB10V2Packed
- #define LvPixelFormat_RGB565Packed
- #define LvPixelFormat_BGR565Packed
- #define LvPixelFormat_YUV411Packed
- #define LvPixelFormat_YUV422Packed
- #define LvPixelFormat_YUV422YUYVPacked
- #define LvPixelFormat_YUV444Packed
- #define LvPixelFormat_RGB8Planar
- #define LvPixelFormat_RGB10Planar
- #define LvPixelFormat_RGB12Planar
- #define LvPixelFormat_RGB16Planar
- #define LvPixelFormat_Mono8s
- #define LvPixelFormat_RGBa8
- #define LvPixelFormat_BGRa8
- #define LvPixelFormat_RGB565p
- #define LvPixelFormat_BGR565p
- #define LvPixelFormat_RGB10p32
- #define LvPixelFormat_BGR555p
- #define LvPixelFormat_YUV411_8_UYYVYY
- #define LvPixelFormat_YUV8_UYV

5.35.1 Detailed Description

5.35.2 Macro Definition Documentation

5.35.2.1 #define LV_PIX_COLOR

PixelFormat component: The pixel format is color.

5.35.2.2 #define LV_PIX_COLOR_MASK

Mask for the color flag

5.35.2.3 #define LV_PIX_CUSTOM

PixelFormat component: The pixel format is custom.

5.35.2.4 #define LV_PIX_EFFECTIVE_PIXEL_SIZE_MASK

Mask for the pixel size part.

5.35.2.5 #define LV_PIX_EFFECTIVE_PIXEL_SIZE_SHIFT

Shift for the pixel size part.

5.35.2.6 #define LV_PIX_MONO

PixelFormat component: The pixel format is monochrome.

5.35.2.7 #define LV_PIX_OCCUPY12BIT

PixelFormat component: One pixel occupies 12 bits.

5.35.2.8 #define LV_PIX_OCCUPY16BIT

PixelFormat component: One pixel occupies 16 bits.

5.35.2.9 #define LV_PIX_OCCUPY24BIT

PixelFormat component: One pixel occupies 24 bits.

5.35.2.10 #define LV_PIX_OCCUPY32BIT

PixelFormat component: One pixel occupies 32 bits.

5.35.2.11 #define LV_PIX_OCCUPY36BIT

PixelFormat component: One pixel occupies 36 bits.

5.35.2.12 #define LV_PIX_OCCUPY48BIT

PixelFormat component: One pixel occupies 48 bits.

5.35.2.13 #define LV_PIX_OCCUPY8BIT

PixelFormat component: One pixel occupies 8 bits.

5.35.2.14 #define LvPixelFormat_BGR10Packed

Alias for LvPixelFormat_BGR10.

5.35.2.15 #define LvPixelFormat_BGR12Packed

Alias for LvPixelFormat_BGR12.

5.35.2.16 #define LvPixelFormat_BGR555p

Alias for LvPixelFormat_BGR555P.

5.35.2.17 #define LvPixelFormat_BGR565p

Alias for LvPixelFormat_BGR565P.

5.35.2.18 #define LvPixelFormat_BGR565Packed

Alias for LvPixelFormat_BGR565P.

5.35.2.19 #define LvPixelFormat_BGR8Packed

Alias for LvPixelFormat_BGR8.

5.35.2.20 #define LvPixelFormat_BGRa8

Alias for LvPixelFormat_BGRA8.

5.35.2.21 #define LvPixelFormat_BGRA8Packed

Alias for LvPixelFormat_BGRA8.

5.35.2.22 #define LvPixelFormat_Mono8s

Alias for LvPixelFormat_Mono8S.

5.35.2.23 #define LvPixelFormat_Mono8Signed

Alias for LvPixelFormat_Mono8S.

5.35.2.24 #define LvPixelFormat_RGB10p32

Alias for LvPixelFormat_RGB10P32.

5.35.2.25 #define LvPixelFormat_RGB10Packed

Alias for LvPixelFormat_RGB10.

5.35.2.26 #define LvPixelFormat_RGB10Planar

Alias for LvPixelFormat_RGB10_Planar.

5.35.2.27 #define LvPixelFormat_RGB10V2Packed

Alias for LvPixelFormat_RGB10P32.

5.35.2.28 #define LvPixelFormat_RGB12Packed

Alias for LvPixelFormat_RGB12.

5.35.2.29 #define LvPixelFormat_RGB12Planar

Alias for LvPixelFormat_RGB12_Planar.

5.35.2.30 #define LvPixelFormat_RGB16Packed

Alias for LvPixelFormat_RGB16.

5.35.2.31 #define LvPixelFormat_RGB16Planar

Alias for LvPixelFormat_RGB16_Planar.

5.35.2.32 #define LvPixelFormat_RGB565p

Alias for LvPixelFormat_RGB565P.

5.35.2.33 #define LvPixelFormat_RGB565Packed

Alias for LvPixelFormat_RGB565P.

5.35.2.34 #define LvPixelFormat_RGB8Packed

Alias for LvPixelFormat_RGB8.

5.35.2.35 #define LvPixelFormat_RGB8Planar

Alias for LvPixelFormat_RGB8_Planar.

5.35.2.36 #define LvPixelFormat_RGBa8

Alias for LvPixelFormat_RGBA8.

5.35.2.37 #define LvPixelFormat_RGBA8Packed

Alias for LvPixelFormat_RGBA8.

5.35.2.38 #define LvPixelFormat_YUV411_8_UYYVYY

Alias for LvPixelFormat_YUV411_8.

5.35.2.39 #define LvPixelFormat_YUV411Packed

Alias for LvPixelFormat_YUV411_8.

5.35.2.40 #define LvPixelFormat_YUV422Packed

Alias for LvPixelFormat_YUV422_8_UYVY.

5.35.2.41 #define LvPixelFormat_YUV422YUYVPacked

Alias for LvPixelFormat_YUV422_8.

5.35.2.42 #define LvPixelFormat_YUV444Packed

Alias for LvPixelFormat_YUV8.

5.35.2.43 #define LvPixelFormat_YUV8_UYV

Alias for LvPixelFormat_YUV8.

5.36 SynView Image Processing Library functions

Modules

- Common functions
- Image initialization functions
- Region of Interest (ROI) functions
- Lookup Table (LUT) functions
- Bayer decoding/encoding functions
- Rotation and line manipulation functions
- Pixel format conversion functions
- Saving/loading functions
- Overlay functions
- RGB color correction and convolution functions
- Shading correction functions

5.36.1 Detailed Description

5.37 Common functions

Functions

- LV_EXTC LV_DLLIMPORT void LvipGetStatusMsg (LvStatus TlStatus, char ∗pMsg, size_t MsgBufSize)

5.37.1 Detailed Description

5.37.2 Function Documentation

5.37.2.1 LV_EXTC LV_DLLIMPORT void LvipGetStatusMsg (LvStatus *TlStatus,* char ∗ *pMsg,* size_t *MsgBufSize*)

Retrieves a text desctibing the status.

Parameters

TlStatus	Error status code.
pMsg	Pointer to buffer for the error message.
MsgBufSize	Size of the buffer.

5.38 Image initialization functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipInitImgInfo (LvipImgInfo *pImgInfo, uint32_t Width, uint32_t Height, uint32_t PixelFormat, uint32_t Attributes)
- LV_EXTC LV_DLLIMPORT uint32_t LvipGetImageDataSize (LvipImgInfo *pImgInfo)
- LV_EXTC LV_DLLIMPORT LvStatus LvipAllocateImageData (LvipImgInfo *pImgInfo)
- LV_EXTC LV_DLLIMPORT LvStatus LvipDeallocateImageData (LvipImgInfo *pImgInfo)
- LV_EXTC LV_DLLIMPORT LvStatus LvipFillWithColor (LvipImgInfo *pImgInfo, uint8_t Red, uint8_t Green, uint8_t Blue, uint32_t Options)

5.38.1 Detailed Description

5.38.2 Function Documentation

5.38.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipAllocateImageData (LvipImgInfo * *pImgInfo*)

Allocates appropriate space to pData or color planes, according to the Height and LineIncrement.

Parameters

pImgInfo	pointer to the LvipImgInfo of the image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.38.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipDeallocateImageData (LvipImgInfo * *pImgInfo*)

Deallocates the image data buffer(s) If the flags is not containing LvipImgAttr_NotDataOwner, deallocates pData or color planes and sets them to NULL.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.38.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvipFillWithColor (LvipImgInfo * *pImgInfo*, uint8_t *Red*, uint8_t *Green*, uint8_t *Blue*, uint32_t *Options*)

Fills image data with specified color.

Parameters

pImgInfo	pointer to LvipImgInfo structure, the data of which has to be filled with the selected color
Red	8bit Red value
Green	8bit Green value
Blue	8bit Blue value
Options	Options reserved - should be set to 0.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.38.2.4 LV_EXTC LV_DLLIMPORT uint32_t LvipGetImageDataSize (LvipImgInfo * *pImgInfo*)

Returns the data size required for the image. Expects the Height and LineIncrement are already calculated. In case of color planes returns the size of one plane

Parameters

pImgInfo	pointer to the LvipImgInfo of the image.

Returns

The data size required for the image in bytes.

5.38.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvipInitImgInfo (LvipImgInfo * *pImgInfo*, uint32_t *Width*, uint32_t *Height*, uint32_t *PixelFormat*, uint32_t *Attributes*)

Initializes the LvipImgInfo to specified values, calculates the line increment and sets pData to NULL (be sure to deallocate the image buffers if were allocated, before this function call). If pData of other owner is used, set the LvipImgAttr_NotDataOwner flag so that the data are not deallocated when LvipDeallocateImageData() is applied to this ImgInfo.

Parameters

pImgInfo	Pointer to LvipImgInfo structure which is to be initialized
Width	Width of image in pixels
Height	Height of image in pixels
PixelFormat	Pixel format; one of the LvPixelFormat.
Attributes	Image attributes; OR-ed combination of the LvipImgAttr.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.39 Region of Interest (ROI) functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipCopyArea (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, int32_t DstXOffset, int32_t DstYOffset, uint32_t DstWidth, uint32_t DstHeight, uint32_t Options)

5.39.1 Detailed Description

5.39.2 Function Documentation

5.39.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipCopyArea (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* int32_t *DstXOffset,* int32_t *DstYOffset,* uint32_t *DstWidth,* uint32_t *DstHeight,* uint32_t *Options*)

Extracts from the source bitmap a rectangle as destination bitmap. If the rectangle goes outside of the source image, the intersection rectangle is taken, that means the result width and/or height can be smaller than required. If the intersection is zero, the function returns returns LVSTATUS_LVIP_DST_RECT_OUTSIDE_SRC.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
DstXOffset	Left offset of the rectangle
DstYOffset	Upper offset of the rectangle
DstWidth	Width of area which has to be copied
DstHeight	Height of area which has to be copied
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40 Lookup Table (LUT) functions

Functions

- LV_EXTC LV_DLLIMPORT LvipHLut LvipAllocateLut (uint32_t LutType)
- LV_EXTC LV_DLLIMPORT LvStatus LvipFreeLut (LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipResetLut (LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet8BitLut (LvipHLut hLut, uint8_t *pRed, uint8_t *pGreen, uint8_t *pBlue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipGet8BitLut (LvipHLut hLut, uint8_t *pRed, uint8_t *pGreen, uint8_t *pBlue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet10BitLut (LvipHLut hLut, uint16_t *pRed, uint16_t *pGreen, uint16_t *pBlue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipGet10BitLut (LvipHLut hLut, uint16_t *pRed, uint16_t *pGreen, uint16_t *pBlue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet12BitLut (LvipHLut hLut, uint16_t *pRed, uint16_t *pGreen, uint16_t *pBlue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipGet12BitLut (LvipHLut hLut, uint16_t *pRed, uint16_t *pGreen, uint16_t *pBlue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet8BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32_t Index, uint8_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvipGet8BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32_t Index, uint8_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet10BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32_t Index, uint16_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvipGet10BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32_t Index, uint16_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet12BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32_t Index, uint16_t Value)
- LV_EXTC LV_DLLIMPORT LvStatus LvipGet12BitLutValue (LvipHLut hLut, LvEnum LutSelector, uint32_t Index, uint16_t *pValue)
- LV_EXTC LV_DLLIMPORT LvStatus LvipAddGammaToLut (uint32_t Gamma, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipAddWbToLut (uint32_t FactorRed, uint32_t FactorGreen, uint32_t FactorBlue, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipAddOffsetAndGainToLut (int32_t Offset, int32_t Gain, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipAddBrightnessAndContrastToLut (int32_t Brightness, int32_t Contrast, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipApplyLut (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, LvipHLut hLut, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipCalcWbFactors (LvipImgInfo *pSrcImgInfo, uint32_t *pFactorRed, uint32_t *pFactorGreen, uint32_t *pFactorBlue, uint32_t Options)

5.40.1 Detailed Description

5.40.2 Function Documentation

5.40.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipAddBrightnessAndContrastToLut (int32_t *Brightness*, int32_t *Contrast*, LvipHLut *hLut*)

Adds brightness and contrast to LUT. Recalculates each value in the LUT table by adding the brightness and multiplying by contrast. This function is similar to the LvipAddOffsetAndGainToLut() function, with the following 2 differences:

- The Brightness middle value is 1000, meaning no change. The Brightness 0 means black image and 2000 means fully white image, because subtracting or adding the 1000 means subtracting or adding the maximum pixel value.
- The Brightness factor is internally corrected in dependence on contrast. The Contrast is equivalent to Gain in the LvipAddOffsetAndGainToLut() function. It is a factor multiplied by 1000, i.e. 1000 means 1.0 = no change. Can be also negative - 1000 makes inversion.

Parameters

Brightness	The Brightness to be added expressed in 1/1000 of the maximum pixel value. See the explanation above.
Contrast	The Contrast factor multiplied by 1000.
hLut	Handle to the LUT.

Returns

LVSTATUS_OK in case of success, or error status in case of failure.

5.40.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipAddGammaToLut (uint32_t *Gamma,* LvipHLut *hLut*)

Adds gamma to LUT. Recalculates each value in the LUT table by applying the Gamma curve. Gamma is supplied multiplied by 1000, i.e. for gamma = 1.0 the passed value will be 1000. There is a possibility to do image lighter/darker using different gamma value. This gamma will be added to LUT and when the image is being transformed using any of function, add this LUT to this function as the last parameter.

Note

There is a need to have LUT - see LvipAllocateLut() and its company.

Parameters

Gamma	Minimal gamma value is 10 - it means that there is a need to enter gamma multiplies by 1000
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvipAddOffsetAndGainToLut (int32_t *Offset,* int32_t *Gain,* LvipHLut *hLut*)

Adds offset and gain to LUT. Recalculates each value in the LUT table by adding the offset and multiplying by gain. The offset is in range -1000 to +1000, where 0 means no change and 1000 the maximum pixel value - adding 1000 will make the image fully white, adding -1000 will make it fully black. The offset is corresponding to Brightness - 1000, see LvipAddBrightnessAndContrastToLut().
The gain is the gain factor multiplied by 1000, i.e. 1000 means 1.0 = no change. Can be also negative - 1000 makes inversion. It is equivalent to contrast.

Parameters

Offset	The Offset to be added expressed in 1/1000 of the maximum pixel value. See the explanation above.

Gain	The Gain factor multiplied by 1000.
hLut	Handle to the LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure.

5.40.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvipAddWbToLut (uint32_t *FactorRed,* uint32_t *FactorGreen,* uint32_t *FactorBlue,* LvipHLut *hLut*)

Adds white balance to LUT. Recalculates each value in the LUT table by applying the white balance factors. The factors are supplied multiplied by 1000, i.e. for the factor = 1.0 the passed value will be 1000. See LvipCalcWb↩Factors() for obtaining the WB factors from an image.

Parameters

FactorRed	Red factor of white balance, multiplied by 1000
FactorGreen	Green factor of white balance, multiplied by 1000
FactorBlue	Blue factor of white balance, multiplied by 1000
hLut	Handle to the LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.5 LV_EXTC LV_DLLIMPORT LvipHLut LvipAllocateLut (uint32_t *LutType*)

Allocates the LUT.

Parameters

LutType	type of LUT which has to be allocated. One of LvipLutType, this value could be optionally OR-ed with the LVIP_LUT_BAYER flag

Returns

handle to the allocated LUT

Note

LUT has to be freed up before end using LvipFreeLut() function

5.40.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvipApplyLut (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* LvipHLut *hLut,* uint32_t *Options*)

Apply LUT to source image and save it in the destination image. Applies the LUT to the image. Note that the LUT can be applied in other functions as well, which is faster than this separate processing.

Supported input pixel formats: 8-bit mono, 10-bit mono, 12-bit mono, 24-bit BGR, 32-bit BGR, 24-bit RGB, 32-bit RGB.
Supported output pixel formats: equal to the input pixel format.
Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
hLut	Handle to the LUT, which has to be applied
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvipCalcWbFactors (LvipImgInfo * *pSrcImgInfo*, uint32_t * *pFactorRed*, uint32_t * *pFactorGreen*, uint32_t * *pFactorBlue*, uint32_t *Options*)

Calculates white balance factors. The image is expected to be obtained from camera pointed at a neutral grey area. The factor is a gain applied to each pixel component. The gain = 1.0 means no change. In order to avoid using float numbers, the factors are multiplied by 1000 and stored in uint32_t. If the image pixel format is MONO, the image is expected to be Bayer Array encoded. The factors are normalized, so that all are >= 1.0. This assures the areas with saturated colors remain white.
The obtained factors could be used in the LvipAddWbToLut() function.

Note

If the LvipOption_WbCorrectFactors flag is used, it is assumed that the white balance is calculated from the image to which were applied white balancing factors passed as parameters. Thus only a correction is calculated and the existing factors are modified. This flag cannot be used on undecoded Bayer array image.

Parameters

pSrcImgInfo	Source image info from which the white balance has to be calculated
pFactorRed	Pointer to uint32_t variable to which will be saved the Red factor, multiplied by 1000. If the LvipOption_WbCorrectFactors flag is used, the variable should contain the factor already used for WB of the current image.
pFactorGreen	Pointer to uint32_t variable to which will be saved the Green factor, multiplied by 1000. If the LvipOption_WbCorrectFactors flag is used, the variable should contain the factor already used for WB of the current image.
pFactorBlue	Pointer to uint32_t variable to which will be saved the Blue factor, multiplied by 1000. If the LvipOption_WbCorrectFactors flag is used, the variable should contain the factor already used for WB of the current image.
Options	Options, see LvipOption_WbCorrectFactors

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvipFreeLut (LvipHLut *hLut*)

Deallocates the LUT.

Parameters

hLut	Handle to LUT (which had been allocated by the LvipAllocateLut() function)

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.9 LV_EXTC LV_DLLIMPORT LvStatus LvipGet10BitLut (LvipHLut *hLut,* uint16_t * *pRed,* uint16_t * *pGreen,* uint16_t * *pBlue*)

Gets 10-bit LUT data. This function fills up supplied arrays with the current LUT data. It is useful for example after calling LvipAddGammaToLut() or LvipAddWbToLut() to get the values of current LUT.

Parameters

hLut	Handle to LUT
pRed	pointer to an array of 1024 uint16_t values, will be filled with red
pGreen	pointer to an array of 1024 uint16_t values, will be filled with green
pBlue	pointer to an array of 1024 uint16_t values, will be filled with blue

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.10 LV_EXTC LV_DLLIMPORT LvStatus LvipGet10BitLutValue (LvipHLut *hLut*, LvEnum *LutSelector*, uint32_t *Index*, uint16_t * *pValue*)

Gets 10-bit LUT value. This function reads one value from the LUT. Note that for reading the whole LUT a more effective function LvipGet10BitLut() is available.

Parameters

hLut	Handle to LUT
LutSelector	LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one.
Index	Value index in the LUT
pValue	Pointer to the variable which receives the value

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.11 LV_EXTC LV_DLLIMPORT LvStatus LvipGet12BitLut (LvipHLut *hLut*, uint16_t * *pRed*, uint16_t * *pGreen*, uint16_t * *pBlue*)

Gets 12-bit LUT data. This function fills up supplied arrays with the current LUT data. It is useful for example after calling LvipAddGammaToLut() or LvipAddWbToLut() to get the values of current LUT.

Parameters

hLut	Handle to LUT
pRed	pointer to an array of 4096 uint16_t values, will be filled with red
pGreen	pointer to an array of 4096 uint16_t values, will be filled with green
pBlue	pointer to an array of 4096 uint16_t values, will be filled with blue

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.12 LV_EXTC LV_DLLIMPORT LvStatus LvipGet12BitLutValue (LvipHLut *hLut*, LvEnum *LutSelector*, uint32_t *Index*, uint16_t * *pValue*)

Gets 12-bit LUT value. This function reads one value from the LUT. Note that for reading the whole LUT a more effective function LvipGet12BitLut() is available.

Parameters

hLut	Handle to LUT
LutSelector	LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one.
Index	Value index in the LUT
pValue	Pointer to the variable which receives the value

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.13 LV_EXTC LV_DLLIMPORT LvStatus LvipGet8BitLut (LvipHLut *hLut*, uint8_t * *pRed*, uint8_t * *pGreen*, uint8_t * *pBlue*)

Gets 8-bit LUT data. This function fills up supplied arrays with the current LUT data. It is useful for example after calling LvipAddGammaToLut() or LvipAddWbToLut() to get the values of current LUT.

Parameters

hLut	Handle to LUT
pRed	pointer to an array of 256 uint8_t values, which will be filled with the Red LUT values
pGreen	pointer to an array of 256 uint8_t values, which will be filled with the Green LUT values
pBlue	pointer to an array of 256 uint8_t values, will be filled with the LUT values

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.14 LV_EXTC LV_DLLIMPORT LvStatus LvipGet8BitLutValue (LvipHLut *hLut*, LvEnum *LutSelector*, uint32_t *Index*, uint8_t * *pValue*)

Gets one 8-bit LUT value. This function reads one value from the LUT. Note that for reading the whole LUT a more effective function LvipGet8BitLut() is available.

Parameters

hLut	Handle to LUT
LutSelector	LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one.
Index	Value index in the LUT
pValue	Pointer to the variable which receives the value

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.15 LV_EXTC LV_DLLIMPORT LvStatus LvipResetLut (LvipHLut *hLut*)

Resets the LUT data to the linear order.

Parameters

hLut	Handle to LUT allocated using the LvipAllocateLut() function

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.16 LV_EXTC LV_DLLIMPORT LvStatus LvipSet10BitLut (LvipHLut *hLut*, uint16_t * *pRed*, uint16_t * *pGreen*, uint16_t * *pBlue*)

Sets up 10-bit LUT data. Sets the LUT from 3 arrays of 1024 uint16_t values with 10-bit values. For processing the monochrome images only the green is used.

Parameters

hLut	Handle to LUT
pRed	pointer to an array of 1024 uint16_t red LUT values
pGreen	pointer to an array of 1024 uint16_t green LUT values
pBlue	pointer to an array of 1024 uint16_t blue LUT values

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.17 LV_EXTC LV_DLLIMPORT LvStatus LvipSet10BitLutValue (LvipHLut *hLut*, LvEnum *LutSelector*, uint32_t *Index*, uint16_t *Value*)

Sets one 10-bit LUT value. This function writes one value to the LUT. Note that for writing the whole LUT a more effective function LvipSet10BitLut() is available.

Parameters

hLut	Handle to LUT
LutSelector	LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one.
Index	Value index in the LUT.
Value	The value.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.18 LV_EXTC LV_DLLIMPORT LvStatus LvipSet12BitLut (LvipHLut *hLut*, uint16_t * *pRed*, uint16_t * *pGreen*, uint16_t * *pBlue*)

Sets up 12-bit LUT data. Sets the LUT from 3 arrays of 4096 uint16_t values with 12-bit values. For processing the monochrome images only the green is used.

Parameters

hLut	Handle to LUT
pRed	pointer to an array of 4096 uint16_t red LUT values
pGreen	pointer to an array of 4096 uint16_t green LUT values
pBlue	pointer to an array of 4096 uint16_t blue LUT values

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.19 LV_EXTC LV_DLLIMPORT LvStatus LvipSet12BitLutValue (LvipHLut *hLut*, LvEnum *LutSelector*, uint32_t *Index*, uint16_t *Value*)

Sets one 12-bit LUT value. This function writes one value to the LUT. Note that for writing the whole LUT a more effective function LvipSet12BitLut() is available.

Parameters

hLut	Handle to LUT
LutSelector	LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one.
Index	Value index in the LUT.
Value	The value.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.20 LV_EXTC LV_DLLIMPORT LvStatus LvipSet8BitLut (LvipHLut *hLut,* uint8_t * *pRed,* uint8_t * *pGreen,* uint8_t * *pBlue*)

Sets up the 8-bit LUT data. Sets the LUT from 3 arrays of 256 uint8_t values. For processing the monochrome images only the green is used.

Parameters

hLut	Handle to LUT
pRed	pointer to an array of 256 uint8_t values with red LUT values
pGreen	pointer to an array of 256 uint8_t values with green LUT values
pBlue	pointer to an array of 256 uint8_t values with blue LUT values

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.40.2.21 LV_EXTC LV_DLLIMPORT LvStatus LvipSet8BitLutValue (LvipHLut *hLut,* LvEnum *LutSelector,* uint32_t *Index,* uint8_t *Value*)

Sets one 8-bit LUT value. This function writes one value to the LUT. Note that for writing the whole LUT a more effective function LvipSet8BitLut() is available.

Parameters

hLut	Handle to LUT
LutSelector	LUT selector (see LvLUTSelector). The Luminance LUT is actually stored in the Green one.
Index	Value index in the LUT.
Value	The value.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41 Bayer decoding/encoding functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipBdShowMosaic (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDst↩ ImgInfo, uint32_t DstPixelFormat, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdGreenToGreyscale (LvipImgInfo *pSrcImgInfo, LvipImgInfo *p↩ DstImgInfo, uint32_t DstPixelFormat, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdNearestNeighbour (LvipImgInfo *pSrcImgInfo, LvipImgInfo *p↩ DstImgInfo, uint32_t DstPixelFormat, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdBilinearInterpolation (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, uint32_t DstPixelFormat, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdBilinearColorCorrection (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, uint32_t DstPixelFormat, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdVariableGradients (LvipImgInfo *pSrcImgInfo, LvipImgInfo *p↩ DstImgInfo, uint32_t DstPixelFormat, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdPixelGrouping (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDst↩ ImgInfo, uint32_t DstPixelFormat, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipBdEncodeToBayer (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDst↩ ImgInfo, uint32_t DstPixelFormat, uint32_t Options)

5.41.1 Detailed Description

5.41.2 Function Documentation

5.41.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipBdBilinearColorCorrection (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options*)

Bayer Decoding: The Bilinear interpolation with Linear Color Correction method The interpolation with Linear Color Correction (LCC) is another adaptive algorithm and optimized for images with edges in horizontal and vertical direction.

Note

This function does not support LUT due to the 2-pass algorithm

Parameters

pSrcImgInfo	Pointer to source image info
pDstImgInfo	Pointer to destination image info
DstPixelFormat	to which LvPixelFormat has to be image converted
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipBdBilinearInterpolation (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options,* LvipHLut *hLut*)

Bayer Decoding: The Bilinear Interpolation method The most commonly used method for fast Bayer decoding. For the color not directly available for the given pixel makes the linear interpolation between the 2 or 4 neigbouring pixels to get it. Gives good results with a high speed.

- Supported input pixel formats: 8-bit mono.

- Supported output pixel formats: 8-bit mono, 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Bayer encoded source image info
pDstImgInfo	Destination image info
DstPixelFormat	To which LvPixelFormat
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvipBdEncodeToBayer (LvipImgInfo * *pSrcImgInfo*, LvipImgInfo * *pDstImgInfo*, uint32_t *DstPixelFormat*, uint32_t *Options*)

This function encode an RGB image back to a Bayer encoded image. This function is generally for testing purposes.

Parameters

pSrcImgInfo	RGB source image info
pDstImgInfo	It will contain bayer encoded image
DstPixelFormat	To which LvPixelFormat convert
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvipBdGreenToGreyscale (LvipImgInfo * *pSrcImgInfo*, LvipImgInfo * *pDstImgInfo*, uint32_t *DstPixelFormat*, uint32_t *Options*)

Bayer Decoding: Convert green to greyscale Converts fast but roughly the Bayer encoded image to a greyscale by using only the green pixels, which cover the half of all pixels. The other half is calculated by bilinear interpolation.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 8-bit mono.
- Can be done in-place: No.

Parameters

pSrcImgInfo	source image info
pDstImgInfo	destination image info
DstPixelFormat	destination LvPixelFormat
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvipBdNearestNeighbour (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options,* LvipHLut *hLut*)

Bayer Decoding: The Nearest Neigbour method The fastest method for Bayer array decoding. It uses the nearest pixel with the required lense color to get the pixel value. Gives rough results.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 8-bit mono, 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Bayer encoded source image info
pDstImgInfo	Destination image info
DstPixelFormat	To which LvPixelFormat
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvipBdPixelGrouping (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options*)

Bayer Decoding: The Pixel Grouping method. A method similar to the LvipBdVariableGradients(), but simplified and thus faster, still giving very good results.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

Note

This function does not support LUT operations because of too high CPU load

Parameters

pSrcImgInfo	Bayer encoded source image info
pDstImgInfo	Destination image info
DstPixelFormat	To which LvPixelFormat convert
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.7 LV_EXTC LV_DLLIMPORT LvStatus LvipBdShowMosaic (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options*)

Bayer Decoding: Show mosaic. This function converts the Bayer encoded image to RGB format, without decoding the color information, only showing in the color how the image is seen by the chip after the light goes through the color lenses. The purpose of this function is only to help in Bayer decoding algorithms investigations.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Bayer encoded image
pDstImgInfo	Where to save displayable image
DstPixelFormat	To which LvPixelFormat convert bayer encoded image
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.41.2.8 LV_EXTC LV_DLLIMPORT LvStatus LvipBdVariableGradients (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options,* LvipHLut *hLut*)

Bayer Decoding: Variable gradients method One of the best known methods for Bayer decoding, but significantly slower than the bilinear interpolation. It is based on evaluation the color gradients in 8 directions around the pixel and selecting the set of best set for the color interpolation.

- Supported input pixel formats: 8-bit mono.
- Supported output pixel formats: 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Bayer encoded source image info
pDstImgInfo	Destination image info
DstPixelFormat	To which LvPixelFormat convert
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.42 Rotation and line manipulation functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipDeinterlace (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipRotate90 (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, int32_t ClockWise, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipMirror (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, int32_t TopBottomMirror, int32_t LeftRightMirror, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipRotate90AndMirror (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, int32_t ClockWise, int32_t TopBottomMirror, int32_t LeftRightMirror, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipReverseLines (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipReverseLinesFast (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, void *pLineBuffer, uint32_t Options)

5.42.1 Detailed Description

5.42.2 Function Documentation

5.42.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipDeinterlace (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *Options*)

Deinterlacing. Deinterlaces by averaging the neighbour lines. Deinterlace function reduces the artefacts resulting from capturing a moving object by an interlaced camera.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.42.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipMirror (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* int32_t *TopBottomMirror,* int32_t *LeftRightMirror,* uint32_t *Options,* LvipHLut *hLut*)

Mirrors the image along the horizontal axis (TopBottomMirror) or vertical axis (LeftRightMirror).

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info.
TopBottomMirror	**1** for top-bottom mirror, **0** if not
LeftRightMirror	**1** for left-right mirror, **0** if not
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

Note

the LUT in this function is not yet implemented.

5.42.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvipReverseLines (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *Options*)

Reversed lines for switching between the top-down and bottom-up formats. Performs the same action as TopBottom mirror, but updates also the LvipImgInfo with a flag indicating the orientation (this has a meaning when switching between top-down and bottom-up formats).

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.42.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvipReverseLinesFast (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* void * *pLineBuffer,* uint32_t *Options*)

Fastly reverses lines (copying whole lines). The pDstImgInfo can be NULL (in-place reversion). In such case a temporary buffer for a line is needed.
The buffer can be supplied in pLineBuffer (must have sufficient size to hold the whole line in its pixel format, that means >= ImgInfo.dwLineIncrement).
If the pLineBuffer is NULL, the buffer is temporarily allocated and deallocated, which might require additional CPU time, so for the repeated call of this function it is better to allocate the buffer outside the function and pass it as pLineBuffer parameter.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info; can be NULL - then in-place reversion will be done (but in this case a temporary buffer for line is needed)
pLineBuffer	Pointer to temporary line buffer. Can be NULL.
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.42.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvipRotate90 (LvipImgInfo * *pSrcImgInfo*, LvipImgInfo * *pDstImgInfo*, int32_t *ClockWise*, uint32_t *Options*, LvipHLut *hLut*)

Rotates the image by 90 degrees clockwise or counterclockwise.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

Note

the LUT in this function is not yet implemented.
For 180 degrees rotation use the LvipMirror() function and set mirroring along both axes.

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
ClockWise	**1** if the image has to be rotated clockwise, **0** if the image has to be rotated by counterclockwise
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.42.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvipRotate90AndMirror (LvipImgInfo * *pSrcImgInfo*, LvipImgInfo * *pDstImgInfo*, int32_t *ClockWise*, int32_t *TopBottomMirror*, int32_t *LeftRightMirror*, uint32_t *Options*, LvipHLut *hLut*)

It does the rotation and mirroring in the same step. If the Options contain LvipOption_ReallocateDst and the p↩ DstImgInfo contains different image width or height or the pData is NULL, the pData is reallocated and the image parameters are adjusted. The Options in such case can contain also LvipImgAttr flags for new image descriptor creation.

- Supported input pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
ClockWise	**1** if image has to be rotated by 90 degrees clockwise, otherwise (counterclockwise) **0**
TopBottomMirror	**1** if top-bottom mirror has to be used, otherwise **0**
LeftRightMirror	**1** if left-right mirror has to be used, otherwise **0**
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

Note

the LUT in this function is not yet implemented.

5.43 Pixel format conversion functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipConvertToPixelFormat (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, uint32_t DstPixelFormat, uint32_t Options)
- LV_EXTC LV_DLLIMPORT uint32_t LvipCanConvertToPixelFormat (uint32_t dwSrcPixelFormat, uint32_t dwDstPixelFormat)

5.43.1 Detailed Description

5.43.2 Function Documentation

5.43.2.1 LV_EXTC LV_DLLIMPORT uint32_t LvipCanConvertToPixelFormat (uint32_t *dwSrcPixelFormat,* uint32_t *dwDstPixelFormat*)

Returns 1 if the source pixel format can be converted to destination pixel format by the LvipConvertToPixelFormat() function.

Parameters

dwSrcPixelFormat	Source LvPixelFormat
dwDstPixelFormat	Destination LvPixelFormat

Returns

1 in case of the conversion is available, otherwise 0.

5.43.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipConvertToPixelFormat (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* uint32_t *DstPixelFormat,* uint32_t *Options*)

Converts the image from one pixel format to another one.

- Supported input pixel formats: 8-bit to 16-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: 8-bit mono, 24-bit BGR, 32-bit BGR.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Source Image Info
pDstImgInfo	Destination Image Info
DstPixelFormat	Destination LvPixelFormat
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.44 Saving/loading functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipSaveToTiff (const char *pFileName, LvipImgInfo *pImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipLoadFromTiff (const char *pFileName, LvipImgInfo *pImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSaveToBmp (const char *pFileName, LvipImgInfo *pImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipLoadFromBmp (const char *pFileName, LvipImgInfo *pImgInfo, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSaveToJpeg (const char *pFileName, LvipImgInfo *pImgInfo, uint32_t QualityFactor)
- LV_EXTC LV_DLLIMPORT LvStatus LvipLoadFromJpeg (const char *pFileName, LvipImgInfo *pImgInfo, uint32_t Options)

5.44.1 Detailed Description

5.44.2 Function Documentation

5.44.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipLoadFromBmp (const char * *pFileName*, LvipImgInfo * *pImgInfo*, uint32_t *Options*)

Loads image from BMP file. Formats with less 8 bits per pixel are not supported. The color palette by 8-bit pixel format is expected to be greyscale.

Parameters

pFileName	File name
pImgInfo	Image info for the loaded image
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.44.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipLoadFromJpeg (const char * *pFileName*, LvipImgInfo * *pImgInfo*, uint32_t *Options*)

Loads the image from JPEG file.

Parameters

pFileName	File name.
pImgInfo	Image info for the loaded image.
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image. See LvipOption_JpegConvertToBgr, LvipOption_JpegReadHeaderOnly and LvipOption_ReallocateDst.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

Note

You can either supply the pImgInfo with already allocated buffer or use empty ImgInfo and specify the LvipOption_ReallocateDst flag. In the first case you can utilize the LvipOption_JpegReadHeaderOnly flag to obtain the image attributes.

5.44.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvipLoadFromTiff (const char * *pFileName*, LvipImgInfo * *pImgInfo*, uint32_t *Options*)

Loads the image from TIFF file. Is preferred to load the image from TIFF file which had been previously saved by LvipSaveToTiff() function - this library supports only a base TIFF format and there it is not assured that the TIFF image created by another application could be loaded without error.

Parameters

pFileName	File name
pImgInfo	Image info for the loaded image.
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

Note

The pData are always reallocated.

5.44.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvipSaveToBmp (const char * *pFileName*, LvipImgInfo * *pImgInfo*, uint32_t *Options*)

Saves the image to a BMP file if the pixel format is compatible with BMP. *Compatible with BMP* means that LvPixelFormat is one of 8-bit mono, 24- or 32-bit BGR.

- Supported pixel formats: 8-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.

Parameters

pFileName	File name
pImgInfo	Image info of an image to be saved
Options	Options - OR-ed combination of LvipOption.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.44.2.5 LV_EXTC LV_DLLIMPORT LvStatus LvipSaveToJpeg (const char * *pFileName*, LvipImgInfo * *pImgInfo*, uint32_t *QualityFactor*)

Saves the image to the JPEG file. In contrast to the BMP format, it enables to store also 9- to 16-bit mono formats.

- Supported pixel formats: 8-bit to 16-bit mono, all RGB and BGR formats. For JPEG the native pixel format is either 8-bit mono or 24-bit BGR. If the image is in different pixel format, it is automatically converted to one of these 2 formats.

Parameters

pFileName	File name.
pImgInfo	Image info of an image to be saved.
QualityFactor	The quality factor in range from 0 to 100. The higher the quality, the lower is the compression. The default quality is 75.

Returns

LVSTATUS_OK in case of success, or error status in case of failure.

5.44.2.6 LV_EXTC LV_DLLIMPORT LvStatus LvipSaveToTiff (const char * *pFileName*, LvipImgInfo * *pImgInfo*, uint32_t *Options*)

Saves the image to the TIFF file. In contrast to the BMP format, it enables to store also 9- to 16-bit mono formats. The flag LvipOption_TiffConvertTo16Bit can be used to force the conversion to 16bit format, which is supported by wider range of applications.

- Supported pixel formats: 8-bit to 16-bit mono, 15-bit BGR, 16-bit BGR, 24-bit BGR, 32-bit BGR.

Parameters

pFileName	File name
pImgInfo	Image info of an image to be saved
Options	Options - OR-ed combination of LvipOption.

Returns

LVSTATUS_OK in case of success, or error status in case of failure.

5.45 Overlay functions

5.46 RGB color correction and convolution functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipApplyRgbColorCorrection (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, int32_t *piMatrix, uint32_t Options, LvipHLut hLut)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSetSaturationMatrix (uint32_t SaturationFactor, int32_t *piMatrix, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipSet3x3MatrixSharpening (int32_t Factor, int32_t *piMatrix, uint32_t Options)
- LV_EXTC LV_DLLIMPORT LvStatus LvipApply3x3Convolution (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, int32_t *piMatrix, uint32_t Options, LvipHLut hLut)

5.46.1 Detailed Description

5.46.2 Function Documentation

5.46.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipApply3x3Convolution (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* int32_t * *piMatrix,* uint32_t *Options,* LvipHLut *hLut*)

Does 3x3 convolution. Applies the 3x3 matrix convolution operation. Typically, if the matrix is set for sharpening, sharpens the image.

See also

LvipSet3x3MatrixSharpening() for creation of the sharpening matrix.

- Supported input pixel formats: 8-bit to 16-bit mono, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: No.

Parameters

pSrcImgInfo	Source image
pDstImgInfo	Destination image info
piMatrix	Matrix for the convolution operation.
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

Note

the LUT in this function is not yet implemented.

5.46.2.2 LV_EXTC LV_DLLIMPORT LvStatus LvipApplyRgbColorCorrection (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* int32_t * *piMatrix,* uint32_t *Options,* LvipHLut *hLut*)

RGB color correction. A color correction 3x3 matrix is applied to RGB components of each pixel.

- Supported input pixel formats: 24-bit BGR, 32-bit BGR, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Image which needs to correct colors
pDstImgInfo	Where to save image with correct colors
piMatrix	3x3 matrix used to correct colors. It could be filled up using LvipSetSaturationMatrix(). The factors in the matrix are expressed as multiplied by 1000, that means 1000 = factor 1.0.
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT (could be NULL)

Returns

LVSTATUS_OK in case of success, or error status in case of failure

Note

the LUT in this function is not yet implemented.

5.46.2.3 LV_EXTC LV_DLLIMPORT LvStatus LvipSet3x3MatrixSharpening (int32_t *Factor*, int32_t * *piMatrix*, uint32_t *Options*)

Sets up sharpening matrix. Fills the matrix with weighted values for 3x3 sharpening.
The factor is 0 for no-change matrix, +100 for maximum sharpening, -100 for blurring

Parameters

Factor	Factor of sharpening
piMatrix	3x3 matrix of int32_t values which will obtain the calculated values
Options	Options: 0 for faster sharpening from 4 neighboring pixels, 1 for full sharpening from 8 neighboring pixels.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.46.2.4 LV_EXTC LV_DLLIMPORT LvStatus LvipSetSaturationMatrix (uint32_t *SaturationFactor*, int32_t * *piMatrix*, uint32_t *Options*)

Sets up the color saturation 3x3 matrix. The saturation factor is in percents, eg. 100 = 1.0 = unchanged image. The matrix can be used as parameter in the LvipApplyRgbColorCorrection() function.

Parameters

SaturationFactor	the saturation factor in percents
piMatrix	3x3 matrix of int32_t values which will obtain the calculated factors
Options	Options, reserved for future use, must be 0.

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.47 Shading correction functions

Functions

- LV_EXTC LV_DLLIMPORT LvStatus LvipApplyShadingCorrection (LvipImgInfo *pSrcImgInfo, LvipImgInfo *pDstImgInfo, LvipImgInfo *pBlackRefImgInfo, LvipImgInfo *pWhiteRefImgInfo, uint32_t Options, LvipHLut hLut)

5.47.1 Detailed Description

5.47.2 Function Documentation

5.47.2.1 LV_EXTC LV_DLLIMPORT LvStatus LvipApplyShadingCorrection (LvipImgInfo * *pSrcImgInfo,* LvipImgInfo * *pDstImgInfo,* LvipImgInfo * *pBlackRefImgInfo,* LvipImgInfo * *pWhiteRefImgInfo,* uint32_t *Options,* LvipHLut *hLut*)

Applies the shading correction. The pBlackRefImgInfo and pWhiteRefImgInfo must be either NULL or must point to a valid image of the same pixel format as the pSrcImgInfo.

- Supported input pixel formats: 8-bit to 16-bit mono, 24-bit BGR, 32-bit BGR.
- Supported output pixel formats: equal to the input pixel format.
- Can be done in-place: Yes (DstImgInfo can be NULL).

Parameters

pSrcImgInfo	Source image info
pDstImgInfo	Destination image info
pBlackRefImg↩ Info	Black reference image
pWhiteRefImg↩ Info	White reference image
Options	Options - OR-ed combination of LvipOption. If the LvipOption_ReallocateDst is used, then also can contain attributes from LvipImgAttr for (re)allocated image.
hLut	Handle to LUT

Returns

LVSTATUS_OK in case of success, or error status in case of failure

5.48 SynView INI file API

Functions

- LV_EXTC LVINI_PUBLIC LvHIniFile LvIniOpen (const char *pCommentSeparator)
- LV_EXTC LVINI_PUBLIC void LvIniClose (LvHIniFile hIniFile)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniLoad (LvHIniFile hIniFile, const char *pFileName)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniSave (LvHIniFile hIniFile, const char *pFileName, uint32_t Create↩ Backup)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniModified (LvHIniFile hIniFile)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniItemExists (LvHIniFile hIniFile, const char *pSection, const char *p↩ Name, uint32_t Order)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniSectionExists (LvHIniFile hIniFile, const char *pSection)
- LV_EXTC LVINI_PUBLIC void LvIniDeleteItem (LvHIniFile hIniFile, const char *pSection, const char *p↩ Name, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniDeleteSection (LvHIniFile hIniFile, const char *pSection)
- LV_EXTC LVINI_PUBLIC int32_t LvIniGetInteger (LvHIniFile hIniFile, const char *pSection, const char *p↩ Name, int32_t Default, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniSetInteger (LvHIniFile hIniFile, const char *pSection, const char *pName, int32_t Value, uint32_t Hexadecimal, uint32_t Order)
- LV_EXTC LVINI_PUBLIC double LvIniGetFloat (LvHIniFile hIniFile, const char *pSection, const char *p↩ Name, double Default, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniSetFloat (LvHIniFile hIniFile, const char *pSection, const char *pName, double Value, uint32_t Order)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniGetBool (LvHIniFile hIniFile, const char *pSection, const char *p↩ Name, uint32_t Default, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniSetBool (LvHIniFile hIniFile, const char *pSection, const char *pName, uint32_t Value, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniGetString (LvHIniFile hIniFile, const char *pSection, const char *pName, const char *pDefault, char *pString, uint32_t Size, uint32_t Order)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniGetStringSize (LvHIniFile hIniFile, const char *pSection, const char *pName, const char *pDefault, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniSetString (LvHIniFile hIniFile, const char *pSection, const char *pName, const char *pValue, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniGetSectionRawLine (LvHIniFile hIniFile, const char *pSection, char *p↩ Line, uint32_t Size, uint32_t Order)
- LV_EXTC LVINI_PUBLIC uint32_t LvIniGetSectionRawLineSize (LvHIniFile hIniFile, const char *pSection, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniSetSectionRawLine (LvHIniFile hIniFile, const char *pSection, const char *pLine, uint32_t Order)
- LV_EXTC LVINI_PUBLIC void LvIniSetParent (LvHIniFile hIniFile, const char *pSection, const char *pName)

5.48.1 Detailed Description

The sv.synview.ini is a helper library, enabling to read and write INI files in all supported operating systems.

5.48.2 Function Documentation

5.48.2.1 LV_EXTC LVINI_PUBLIC void LvIniClose (LvHIniFile *hIniFile*)

Closes the INI file underlying structures and all its parents. It does nto write the contents to disk; if you want to save the INI content, use first LvIniSave().

Parameters

hIniFile	Handle to the INI file.

5.48.2.2 LV_EXTC LVINI_PUBLIC void LvIniDeleteItem (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* uint32_t *Order*)

Deletes the item.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Order	Can be used to distiguish between multiple items of the same name in one section.

5.48.2.3 LV_EXTC LVINI_PUBLIC void LvIniDeleteSection (LvHIniFile *hIniFile,* const char * *pSection*)

Deletes the section.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).

5.48.2.4 LV_EXTC LVINI_PUBLIC uint32_t LvIniGetBool (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* uint32_t *Default,* uint32_t *Order*)

Reads a boolean value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Default	Default value to be used when the item is not found or is empty.
Order	Can be used to distiguish between multiple items of the same name in one section.

Returns

Read value (0 or 1).

5.48.2.5 LV_EXTC LVINI_PUBLIC double LvIniGetFloat (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* double *Default,* uint32_t *Order*)

Reads a float value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.

Default	Default value to be used when the item is not found or is empty.
Order	Can be used to distiguish between multiple items of the same name in one section.

Returns

Read value.

5.48.2.6 LV_EXTC LVINI_PUBLIC int32_t LvIniGetInteger (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* int32_t *Default,* uint32_t *Order*)

Reads an integer value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Default	Default value to be used when the item is not found or is empty.
Order	Can be used to distiguish between multiple items of the same name in one section.

Returns

Read value.

5.48.2.7 LV_EXTC LVINI_PUBLIC void LvIniGetSectionRawLine (LvHIniFile *hIniFile,* const char * *pSection,* char * *pLine,* uint32_t *Size,* uint32_t *Order*)

Gets the raw string in specified section at position, specified by Order. Commented out and empty lines are not counted. If the item is not found, empty string is returned.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pLine	Buffer for the line contents.
Size	Size of the pLine buffer.
Order	Order of the raw line. The first valid line has Order=1

5.48.2.8 LV_EXTC LVINI_PUBLIC uint32_t LvIniGetSectionRawLineSize (LvHIniFile *hIniFile,* const char * *pSection,* uint32_t *Order*)

Gets the raw size of buffer needed for the raw line in specified section at position, specified by Order. Commented out and empty lines are not counted. If the item is not found, empty string is returned.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
Order	Order of the raw line. The first valid line has Order=1.

Returns

Size of the buffer (string length+1).

5.48.2.9 LV_EXTC LVINI_PUBLIC void LvIniGetString (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* const char * *pDefault,* char * *pString,* uint32_t *Size,* uint32_t *Order*)

Reads a string value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
pDefault	Default value to be used when the item is not found or is empty.
pString	The string value is returned in this parameter.
Size	Size of the pString buffer.
Order	Can be used to distiguish between multiple items of the same name in one section.

5.48.2.10 LV_EXTC LVINI_PUBLIC uint32_t LvIniGetStringSize (LvHIniFile *hIniFile*, const char * *pSection*, const char * *pName*, const char * *pDefault*, uint32_t *Order*)

Returns a size of buffer needed to read the string.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
pDefault	Default value to be used when the item is not found or is empty.
Order	Can be used to distiguish between multiple items of the same name in one section.

Returns

Size of the buffer (string length+1).

5.48.2.11 LV_EXTC LVINI_PUBLIC uint32_t LvIniItemExists (LvHIniFile *hIniFile*, const char * *pSection*, const char * *pName*, uint32_t *Order*)

Checks if the item exists.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Order	Can be used to distiguish between multiple items of the same name in one section.

Returns

The return value 1 indicates the existence of the item.

5.48.2.12 LV_EXTC LVINI_PUBLIC uint32_t LvIniLoad (LvHIniFile *hIniFile*, const char * *pFileName*)

Loads the ini file contents. All subsequent changes are done in memory, until the ini file is saved by the LvIniSave() function. If there is a parent already specified (see LvIniSetParent()), it is opened (recursively) as well.

Parameters

hIniFile	Handle to the INI file.
pFileName	The INI file name.

Returns

If the file does not exist, or file I/O fails, it returns 0, otherwise 1.

5.48.2.13 LV_EXTC LVINI_PUBLIC uint32_t LvIniModified (LvHIniFile *hIniFile*)

Returns 1 if the file content was modified.

Returns

1 if at least one item was modified or deleted or section deleted.

5.48.2.14 LV_EXTC LVINI_PUBLIC LvHIniFile LvIniOpen (const char * *pCommentSeparator*)

Creates the INI file underlying structures and returns a handle to it. It does not read the INI file contents, this is done by LvIniLoad().

Parameters

pCommentSeparator	If specified, all lines where it appears at the beginning of the line are considered to be a comment. It need not be placed at the very first position, but to the left must be only whitespace characters, otherwise the line is not considered to be a comment. If a value is commented out and the new value of the same name is written, it is placed before the commented value (not to the end of the section).

5.48.2.15 LV_EXTC LVINI_PUBLIC uint32_t LvIniSave (LvHIniFile *hIniFile*, const char * *pFileName*, uint32_t *CreateBackup*)

Saves the INI file contents to a file.

Parameters

hIniFile	Handle to the INI file.
pFileName	The INI file name.
CreateBackup	If is set, the original file is preserved with added ".bak" extension.

Returns

The return value 1 indicates a success of the file I/O.

5.48.2.16 LV_EXTC LVINI_PUBLIC uint32_t LvIniSectionExists (LvHIniFile *hIniFile*, const char * *pSection*)

Checks is the section exists.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).

Returns

1 if the section exists.

5.48.2.17 LV_EXTC LVINI_PUBLIC void LvIniSetBool (LvHIniFile *hIniFile*, const char * *pSection*, const char * *pName*, uint32_t *Value*, uint32_t *Order*)

Writes a boolean value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Value	Value of the item to be set.
Order	Can be used to distiguish between multiple items of the same name in one section.

5.48.2.18 LV_EXTC LVINI_PUBLIC void LvIniSetFloat (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* double *Value,* uint32_t *Order*)

Writes a float value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Value	Value of the item to be set.
Order	Can be used to distiguish between multiple items of the same name in one section.

5.48.2.19 LV_EXTC LVINI_PUBLIC void LvIniSetInteger (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* int32_t *Value,* uint32_t *Hexadecimal,* uint32_t *Order*)

Writes an integer value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
Value	Value of the item to be set.
Hexadecimal	If true, the value is written as hexa with the "0x" prefix.
Order	Can be used to distiguish between multiple items of the same name in one section.

5.48.2.20 LV_EXTC LVINI_PUBLIC void LvIniSetParent (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName*)

Sets the section and name, where the specification of the parent INI file should be read. For example for the following INI the call would be SetParent("Linked", "Parent");

```
[Linked]
Parent=main.ini
```

Use empty strings to disable the parent.

When you specify a parent, you can automatically work with a hierarchy of INI files - when the file is open, it searches for a section and name, specified in the SetParent() function. If found and a valid file name is specified there, it creates a parent class instance and reads to it the contents of the file. Then every item, which is not explicitly specified in own INI file is searched in this parent INI file. The opening is recusive - the parent INI can have its own parent etc., so be sure you do not make a circular reference. The maximum level of recursion is intentionally limited to 10.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.

5.48.2.21 LV_EXTC LVINI_PUBLIC void LvIniSetSectionRawLine (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pLine,* uint32_t *Order*)

Sets the raw string in specified section at position, specified by Order. Commented out and empty lines are not counted. The first valid line has Order=1 If the item with the Order is not found, a new line is created and added at the end of section.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pLine	The raw line to be set.
Order	Order of the raw line.

5.48.2.22 LV_EXTC LVINI_PUBLIC void LvIniSetString (LvHIniFile *hIniFile,* const char * *pSection,* const char * *pName,* const char * *pValue,* uint32_t *Order*)

Writes a string value.

Parameters

hIniFile	Handle to the INI file.
pSection	Section in the INI file (without brackets).
pName	Name of the item in the section.
pValue	Value of the item to be set.
Order	Can be used to distiguish between multiple items of the same name in one section.

5.49 LvStatus definitions

Macros

- #define LVSTATUS_OK
- #define LVSTATUS_LIBRARY_NOT_LOADED
- #define LVSTATUS_LIBRARY_NOT_OPEN
- #define LVSTATUS_AVISAVER_TOO_MANY_INSTANCES
- #define LVSTATUS_DEVICE_TOO_MANY_INSTANCES
- #define LVSTATUS_CANNOT_LOAD_GENTL
- #define LVSTATUS_DISABLED_BY_CALLBACK
- #define LVSTATUS_DISPLAY_LIBRARY_NOT_LOADED
- #define LVSTATUS_DISPLAY_CANNOT_DISPLAY
- #define LVSTATUS_DISPLAY_NOT_OPEN
- #define LVSTATUS_ENUM_ENTRY_INVALID
- #define LVSTATUS_EVENT_NOT_POSSIBLE
- #define LVSTATUS_EVENT_TOO_MANY_INSTANCES
- #define LVSTATUS_FILE_CANNOT_CREATE
- #define LVSTATUS_FILE_CANNOT_OPEN
- #define LVSTATUS_GENICAM_EXCEPTION
- #define LVSTATUS_HANDLE_INVALID
- #define LVSTATUS_CHUNK_ADAPTER_NOT_AVAILABLE
- #define LVSTATUS_INDEX_OUT_OF_RANGE
- #define LVSTATUS_INTERFACE_TOO_MANY_INSTANCES
- #define LVSTATUS_INVALID_IN_THIS_MODULE
- #define LVSTATUS_ITEM_GROUP_INVALID
- #define LVSTATUS_ITEM_INVALID
- #define LVSTATUS_NO_CONSTANT_FOR_THIS_ENUMENTRY
- #define LVSTATUS_INVALID_ENUMENTRY_ID
- #define LVSTATUS_ITEM_NOT_APPLICABLE
- #define LVSTATUS_ITEM_NOT_AVAILABLE
- #define LVSTATUS_ITEM_NOT_READABLE
- #define LVSTATUS_ITEM_NOT_WRITABLE
- #define LVSTATUS_NODE_MAP_CANNOT_GET
- #define LVSTATUS_NOT_IMPLEMENTED
- #define LVSTATUS_PARAMETER_INVALID
- #define LVSTATUS_RENDERER_TOO_MANY_INSTANCES
- #define LVSTATUS_STREAM_ALREADY_STARTED
- #define LVSTATUS_STREAM_ALREADY_STOPPED
- #define LVSTATUS_STREAM_TOO_MANY_INSTANCES
- #define LVSTATUS_SYSTEM_TOO_MANY_INSTANCES
- #define LVSTATUS_DEVICE_NOT_ACCESSIBLE
- #define LVSTATUS_DEVICE_NOT_READWRITE
- #define LVSTATUS_NOT_SUPPORTED_FOR_THIS_EVENT
- #define LVSTATUS_NOT_ENOUGH_BUFFERS
- #define LVSTATUS_INSUFFICIENT_BUFFER_SIZE
- #define LVSTATUS_INVALID_IP_OR_MAC_ADDRESS_FORMAT
- #define LVSTATUS_CANNOT_LOAD_XML
- #define LVSTATUS_INSUFFICIENT_STRING_BUFFER_SIZE
- #define LVSTATUS_NOT_FOUND
- #define LVSTATUS_PARAM_NOT_APPLICABLE
- #define LVSTATUS_ENUM_ENTRY_NOT_AVAILABLE
- #define LVSTATUS_TIMEOUT
- #define LVSTATUS_LUT_NOT_AVAILABLE

- #define LVSTATUS_LUT_UNSUPPORTED_SIZE
- #define LVSTATUS_XML_UNZIP_ERROR
- #define LVSTATUS_ACQUISITION_CANNOT_BE_STARTED
- #define LVSTATUS_ACQUISITION_CANNOT_BE_STOPPED
- #define LVSTATUS_SETTINGS_INCOMPATIBLE_MODEL
- #define LVSTATUS_SETTINGS_INCOMPATIBLE_VERSION
- #define LVSTATUS_SETTINGS_INCOMPATIBLE_ID
- #define LVSTATUS_BUFFER_IS_QUEUED
- #define LVSTATUS_BUFFER_NOT_FILLED
- #define LVSTATUS_CANNOT_REOPEN_LIBRARY
- #define LVSTATUS_GC_UNKNOWN
- #define LVSTATUS_GC_ERROR
- #define LVSTATUS_GC_NOT_INITIALIZED
- #define LVSTATUS_GC_NOT_IMPLEMENTED
- #define LVSTATUS_GC_RESOURCE_IN_USE
- #define LVSTATUS_GC_ACCESS_DENIED
- #define LVSTATUS_GC_INVALID_HANDLE
- #define LVSTATUS_GC_INVALID_ID
- #define LVSTATUS_GC_NO_DATA
- #define LVSTATUS_GC_INVALID_PARAMETER
- #define LVSTATUS_GC_IO
- #define LVSTATUS_GC_TIMEOUT
- #define LVSTATUS_GC_ABORT
- #define LVSTATUS_GC_INVALID_BUFFER
- #define LVSTATUS_GC_NOT_AVAILABLE
- #define LVSTATUS_SRCGEN_TEMPLATE_NOT_AVAILABLE
- #define LVSTATUS_SRCGEN_SYMBOLIC_NOT_AVAILABLE
- #define LVSTATUS_LICENSE_NOT_AVAILABLE
- #define LVSTATUS_LAST_ERROR_NOT_AVAILABLE
- #define LVSTATUS_ERROR

Typedefs

- typedef uint32_t LvStatus

5.49.1 Detailed Description

5.49.2 Macro Definition Documentation

5.49.2.1 #define LVSTATUS_ACQUISITION_CANNOT_BE_STARTED

The AcquisitionStart command cannot be executed. This is probably because the acquisition is already running, or the conditions for starting the acquisition are not satisfied.

5.49.2.2 #define LVSTATUS_ACQUISITION_CANNOT_BE_STOPPED

The AcquisitionStop command cannot be executed. This is probably because the acquisition already had been stopped, or the conditions for stopping the acquisition are not satisfied.

5.49.2.3 #define LVSTATUS_AVISAVER_TOO_MANY_INSTANCES

Too many concurrent instances of the AviSaver class.

5.49.2.4 #define LVSTATUS_BUFFER_IS_QUEUED

The operation on the buffer is not possible, the buffer is queued for acquisition.

5.49.2.5 #define LVSTATUS_BUFFER_NOT_FILLED

The buffer was not yet filled with image data.

5.49.2.6 #define LVSTATUS_CANNOT_LOAD_GENTL

The GenTL library cannot be loaded. Check its name specification and compatibility.

5.49.2.7 #define LVSTATUS_CANNOT_LOAD_XML

The XML file with feature definitions could not be loaded. This may indicate a misconfiguration of the SynView system - reinstallation recommended.

5.49.2.8 #define LVSTATUS_CANNOT_REOPEN_LIBRARY

The SynView library cannot be reopened after it was once closed.

5.49.2.9 #define LVSTATUS_CHUNK_ADAPTER_NOT_AVAILABLE

The cunk data cannot be parsed, because the ChunkAdapter could not be obtained.

5.49.2.10 #define LVSTATUS_DEVICE_NOT_ACCESSIBLE

The device is not accessible. It is either used by other application or, if it is a GigE device, it can be on a different network or there can be an IP address confict.

5.49.2.11 #define LVSTATUS_DEVICE_NOT_READWRITE

The device (camera) does not permit a read-write access. The application attempted to open the device for controllable or exclusive access, but it is probably currently used by another application.

5.49.2.12 #define LVSTATUS_DEVICE_TOO_MANY_INSTANCES

Too many concurrent instances of the Device class.

5.49.2.13 #define LVSTATUS_DISABLED_BY_CALLBACK

This function is disabled, because the callback function was registered.

5.49.2.14 #define LVSTATUS_DISPLAY_CANNOT_DISPLAY

The buffer contents cannot be displayed, either because its pixel format cannot be converted to a displayable one, or the display options are not supported by the operating system.

5.49.2.15 #define LVSTATUS_DISPLAY_LIBRARY_NOT_LOADED

The lv.display library is not loaded. Failure to load this library may be caused by the operating system environment.

5.49.2.16 #define LVSTATUS_DISPLAY_NOT_OPEN

The display is not open - an invalid window handle (and/or in Linux display pointer) was set.

5.49.2.17 #define LVSTATUS_ENUM_ENTRY_INVALID

The requested enum entry name or value does not exist.

5.49.2.18 #define LVSTATUS_ENUM_ENTRY_NOT_AVAILABLE

The requested enumeration entry is not available.

5.49.2.19 #define LVSTATUS_ERROR

Undefined error. More info in the sv.synview.log file.

5.49.2.20 #define LVSTATUS_EVENT_NOT_POSSIBLE

The requested functionality cannot be used for this Event type.

5.49.2.21 #define LVSTATUS_EVENT_TOO_MANY_INSTANCES

Too many concurrent instances of the Event class.

5.49.2.22 #define LVSTATUS_FILE_CANNOT_CREATE

Cannot create file. Might be caused by a wrong path specification, insufficient rights or protected file existence.

5.49.2.23 #define LVSTATUS_FILE_CANNOT_OPEN

Cannot open file. Might be caused by a wrong path specification, insufficient rights or file nonexistence.

5.49.2.24 #define LVSTATUS_GC_ABORT

The GenTL library returned the GC_ERR_ABORT error. More info in the sv.synview.log file.

5.49.2.25 #define LVSTATUS_GC_ACCESS_DENIED

The GenTL library returned the GC_ERR_ACCESS_DENIED error. More info in the sv.synview.log file.

5.49.2.26 #define LVSTATUS_GC_ERROR

The GenTL library returned the GC_ERR_ERROR error. More info in the sv.synview.log file.

5.49.2.27 #define LVSTATUS_GC_INVALID_BUFFER

The GenTL library returned the GC_ERR_INVALID_BUFFER error. More info in the sv.synview.log file.

5.49.2.28 #define LVSTATUS_GC_INVALID_HANDLE

The GenTL library returned the GC_ERR_INVALID_HANDLE error. More info in the sv.synview.log file.

5.49.2.29 #define LVSTATUS_GC_INVALID_ID

The GenTL library returned the GC_ERR_INVALID_ID error. More info in the sv.synview.log file.

5.49.2.30 #define LVSTATUS_GC_INVALID_PARAMETER

The GenTL library returned the GC_ERR_INVALID_PARAMETER error. More info in the sv.synview.log file.

5.49.2.31 #define LVSTATUS_GC_IO

The GenTL library returned the GC_ERR_IO error. More info in the sv.synview.log file.

5.49.2.32 #define LVSTATUS_GC_NO_DATA

The GenTL library returned the GC_ERR_NO_DATA error. More info in the sv.synview.log file.

5.49.2.33 #define LVSTATUS_GC_NOT_AVAILABLE

The GenTL library returned the GC_ERR_NOT_AVAILABLE error. More info in the sv.synview.log file.

5.49.2.34 #define LVSTATUS_GC_NOT_IMPLEMENTED

The GenTL library returned the GC_ERR_NOT_IMPLEMENTED error. More info in the sv.synview.log file.

5.49.2.35 #define LVSTATUS_GC_NOT_INITIALIZED

The GenTL library returned the GC_ERR_NOT_INITIALIZED error. More info in the sv.synview.log file.

5.49.2.36 #define LVSTATUS_GC_RESOURCE_IN_USE

The GenTL library returned the GC_ERR_RESOURCE_IN_USE error. More info in the sv.synview.log file.

5.49.2.37 #define LVSTATUS_GC_TIMEOUT

The GenTL library returned the GC_ERR_TIMEOUT error. More info in the sv.synview.log file.

5.49.2.38 #define LVSTATUS_GC_UNKNOWN

The GenTL library returned the GC_ERR_UNKNOWN error. More info in the sv.synview.log file.

5.49.2.39 #define LVSTATUS_GENICAM_EXCEPTION

An exception occured when using the GenICam GenApi library. More info about the exception nature is recorded to the sv.synview.log file.

5.49.2.40 #define LVSTATUS_HANDLE_INVALID

An invalid handle was passed to a SynView API function.

5.49.2.41 #define LVSTATUS_INDEX_OUT_OF_RANGE

The specified index is out of range.

5.49.2.42 #define LVSTATUS_INSUFFICIENT_BUFFER_SIZE

Some of the allocated buffers are of smaller size, than is the payload size needed for the acquisition. See also LvStream_LvCalcPayloadSize.

5.49.2.43 #define LVSTATUS_INSUFFICIENT_STRING_BUFFER_SIZE

The string buffer passed to the function does not have the size big enough to hold the returned string.

5.49.2.44 #define LVSTATUS_INTERFACE_TOO_MANY_INSTANCES

Too many concurrent instances of the Interface class.

5.49.2.45 #define LVSTATUS_INVALID_ENUMENTRY_ID

The specified enum entry ID is not a valid SynView constant. Use the enum entry string identifiers to handle enum entries which do not have a SynView constant defined.

5.49.2.46 #define LVSTATUS_INVALID_IN_THIS_MODULE

This function cannot be used in this module.

5.49.2.47 #define LVSTATUS_INVALID_IP_OR_MAC_ADDRESS_FORMAT

The IP or MAC Address used in the LvSetString() has wrong format. The proper format is N.N.N.N for IP address and XX:XX:XX:XX:XX:XX for MAC address, where N is decadic number between 0 and 255, XX is 2 digit hexdecimal number.

5.49.2.48 #define LVSTATUS_ITEM_GROUP_INVALID

Invalid FtrGroup specified.

5.49.2.49 #define LVSTATUS_ITEM_INVALID

Invalid Item ID specified.

5.49.2.50 #define LVSTATUS_ITEM_NOT_APPLICABLE

This function is not applicable to this item.

5.49.2.51 #define LVSTATUS_ITEM_NOT_AVAILABLE

This function requires availability of specific item, but it is not available. More info in the sv.synview.log file.

5.49.2.52 #define LVSTATUS_ITEM_NOT_READABLE

The item is not readable.

5.49.2.53 #define LVSTATUS_ITEM_NOT_WRITABLE

The item is not writable.

5.49.2.54 #define LVSTATUS_LAST_ERROR_NOT_AVAILABLE

The last error status could not be recorded. This is most probably caused by too many threads used by the application.

5.49.2.55 #define LVSTATUS_LIBRARY_NOT_LOADED

The base library is not loaded. Failure to load the library may be caused by missing DLLs, check the SynView installation.

5.49.2.56 #define LVSTATUS_LIBRARY_NOT_OPEN

The SynView library was not open by the LvOpenLibrary() function.

5.49.2.57 #define LVSTATUS_LICENSE_NOT_AVAILABLE

License error. License not available

5.49.2.58 #define LVSTATUS_LUT_NOT_AVAILABLE

The lookup table is not available for the current pixel format.

5.49.2.59 #define LVSTATUS_LUT_UNSUPPORTED_SIZE

The lookup table has unsupported size.

5.49.2.60 #define LVSTATUS_NO_CONSTANT_FOR_THIS_ENUMENTRY

SynView constant does not exists for this enum entry. Use the enum entry string identifier to handle it.

5.49.2.61 #define LVSTATUS_NODE_MAP_CANNOT_GET

Cannot obtain a feature node map for the device. More info in the sv.synview.log file.

5.49.2.62 #define LVSTATUS_NOT_ENOUGH_BUFFERS

The number of allocated buffers is smaller than required minimum number of buffers. See also the LvStream_↩ StreamAnnounceBufferMinimum.

5.49.2.63 #define LVSTATUS_NOT_FOUND

The Interface or Device was not found according to the search criteria.

5.49.2.64 #define LVSTATUS_NOT_IMPLEMENTED

The functionality is not implemented for the requested parameters.

5.49.2.65 #define LVSTATUS_NOT_SUPPORTED_FOR_THIS_EVENT

The requested function is not supported by this event type.

5.49.2.66 #define LVSTATUS_OK

No error.

5.49.2.67 #define LVSTATUS_PARAM_NOT_APPLICABLE

A parameter passed to the function is not applicable at this function. For example in the LvSystemFindInterface() the constant LvFindBy_UserID is not applicable, because this constant can be used only for devices.

5.49.2.68 #define LVSTATUS_PARAMETER_INVALID

Invalid parameter passed to a SynView API function (for example an invalid pointer).

5.49.2.69 #define LVSTATUS_RENDERER_TOO_MANY_INSTANCES

Too many concurrent instances of the Renderer class.

5.49.2.70 #define LVSTATUS_SETTINGS_INCOMPATIBLE_ID

The file with settings were saved with an ID. The ID specified when loading is different.

5.49.2.71 #define LVSTATUS_SETTINGS_INCOMPATIBLE_MODEL

The file with settings was created by different remote device vendor/model. This may cause its incompatibility with the current remote device.

5.49.2.72 #define LVSTATUS_SETTINGS_INCOMPATIBLE_VERSION

The file with settings was created by different remote device firmware version. This may cause its incompatibility with the current remote device.

5.49.2.73 #define LVSTATUS_SRCGEN_SYMBOLIC_NOT_AVAILABLE

The symbolic for requested item is not available.

5.49.2.74 #define LVSTATUS_SRCGEN_TEMPLATE_NOT_AVAILABLE

The template for requested item is not available.

5.49.2.75 #define LVSTATUS_STREAM_ALREADY_STARTED

The stream was already started.

5.49.2.76 #define LVSTATUS_STREAM_ALREADY_STOPPED

The stream was already stopped.

5.49.2.77 #define LVSTATUS_STREAM_TOO_MANY_INSTANCES

Too many concurrent instances of the Stream class.

5.49.2.78 #define LVSTATUS_SYSTEM_TOO_MANY_INSTANCES

Too many concurrent instances of the System class.

5.49.2.79 #define LVSTATUS_TIMEOUT

The function has returned because a timeout has expired.

5.49.2.80 #define LVSTATUS_XML_UNZIP_ERROR

The XML file with camera remote features could not be extracted from the ZIP file.

5.49.3 Typedef Documentation

5.49.3.1 typedef uint32_t LvStatus

General typedef for the error status. Status values are available as defines prefixed with LVSTATUS_xxx. Value of 0 (LVSTATUS_OK) indicates no error. Most SynView functions are returning the status value to indicate the success of the function call. See also the LvGetErrorMessage() function.

5.50 SynView Image Processing Library LvStatus definitions

Macros

- #define LVSTATUS_LVIP_INVALID_POINTER
- #define LVSTATUS_LVIP_INVALID_SRC_POINTER
- #define LVSTATUS_LVIP_INVALID_DST_POINTER
- #define LVSTATUS_LVIP_INVALID_PIXEL_FORMAT
- #define LVSTATUS_LVIP_IMAGEINFO_NOT_INITIALIZED
- #define LVSTATUS_LVIP_MEMORY_ALLOC_FAILED
- #define LVSTATUS_LVIP_UNSUPPORTED_BMP_HEADER
- #define LVSTATUS_LVIP_BMP_INCOMPATIBLE_PIXEL_FORMAT
- #define LVSTATUS_LVIP_BMP_INCOMPATIBLE_LINE_INCREMENT
- #define LVSTATUS_LVIP_IMAGEINFO_NOT_EQUAL
- #define LVSTATUS_LVIP_UNSUPPORTED
- #define LVSTATUS_LVIP_UNSUPPORTED_SRC_PIXEL_FORMAT
- #define LVSTATUS_LVIP_UNSUPPORTED_DST_PIXEL_FORMAT
- #define LVSTATUS_LVIP_UNSUPPORTED_COLOR_PLANES
- #define LVSTATUS_LVIP_UNSUPPORTED_REVERSION
- #define LVSTATUS_LVIP_LINEINCREMENT_TOO_BIG
- #define LVSTATUS_LVIP_DST_IMG_INFO_INCOMPATIBLE
- #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE
- #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE_ROTATED
- #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_PIXEL_FORMAT
- #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_FLAGS
- #define LVSTATUS_LVIP_DST_RECT_OUTSIDE_SRC
- #define LVSTATUS_LVIP_SRC_IMAGEINFO_NO_DATA
- #define LVSTATUS_LVIP_DST_IMAGEINFO_NO_DATA
- #define LVSTATUS_LVIP_NOT_DISPLAYABLE_FORMAT
- #define LVSTATUS_LVIP_INVALID_LUT_HANDLE
- #define LVSTATUS_LVIP_INVALID_LUT_TYPE
- #define LVSTATUS_LVIP_INCOMPATIBLE_REF_PIXEL_FORMAT
- #define LVSTATUS_LVIP_INCOMPATIBLE_REF_FLAGS
- #define LVSTATUS_LVIP_CANNOT_OPEN_READ_FILE
- #define LVSTATUS_LVIP_CANNOT_CREATE_WRITE_FILE
- #define LVSTATUS_LVIP_TIFF_CONTENTS_INVALID
- #define LVSTATUS_LVIP_BMP_CONTENTS_INVALID
- #define LVSTATUS_LVIP_NOT_BAYER_PIXEL_FORMAT
- #define LVSTATUS_LVIP_JPEG_SAVE_FAILED
- #define LVSTATUS_LVIP_JPEG_LOAD_FAILED

5.50.1 Detailed Description

5.50.2 Macro Definition Documentation

5.50.2.1 #define LVSTATUS_LVIP_BMP_CONTENTS_INVALID

The contents of the BMP file is in the invalid or in the unsupported form. You are trying to read a BMP file which has different (or invalid) format than which is supported by this library.

5.50.2.2 #define LVSTATUS_LVIP_BMP_INCOMPATIBLE_LINE_INCREMENT

Color format has incompatible line increment to BMP possibilities. The BMP has to have line increment aligned to 4 bytes.

5.50.2.3 #define LVSTATUS_LVIP_BMP_INCOMPATIBLE_PIXEL_FORMAT

Source pixel format is incompatible to BITMAP pixel format possibilities. BITMAP could be created only using one of these pixel formats:

```
LVIP_PIXEL_FORMAT_MONO8
LVIP_PIXEL_FORMAT_RGB555_PACKED
LVIP_PIXEL_FORMAT_RGB565_PACKED
LVIP_PIXEL_FORMAT_RGB8_PACKED
LVIP_PIXEL_FORMAT_RGBA8_PACKED
```

If you have the image in other pixel format, you should convert the image by the LVIP_ConvertToPixelFormat() function.

5.50.2.4 #define LVSTATUS_LVIP_CANNOT_CREATE_WRITE_FILE

A system error occured when trying to create or write to the file. Possible reasons could be an invalid path and/or file name, without rights to be created/written.

5.50.2.5 #define LVSTATUS_LVIP_CANNOT_OPEN_READ_FILE

A system error occured when trying to open or read the file. Possible reasons could be a wrong path and/or file name, the file currently locked or without rights to be opened/read or the file does not exist yet.

5.50.2.6 #define LVSTATUS_LVIP_DST_IMAGEINFO_NO_DATA

The destination image info structure has no data. The destination image info has no image data - it means that the destination image info does not point to any valid image data and it is not permitted to allocate the buffer automatically (the LvipOption_ReallocateDst flag was not specified).

5.50.2.7 #define LVSTATUS_LVIP_DST_IMG_INFO_INCOMPATIBLE

Destination image info is incompatible to expected output format. You are trying to call a function and the supplied destination image info has other than expected contents.
Possible reason is that you don'did not specify the LvipOption_ReallocateDst flag, so if the function has different output format than specified in the destination image info, the image info could not be rellocated and this error happens.
Note that some functions may require different destination parameters than expected, the size of the destination image in the LVIP_CopyArea() function depends on whether the desired rectangle lies fully in the image or not.

5.50.2.8 #define LVSTATUS_LVIP_DST_RECT_OUTSIDE_SRC

The specified rectangle is outside of source image data. It means that you are trying to copy an area which not exist in the source image. To create a destination image, the rectangle must at least partially overlap the source image.

5.50.2.9 #define LVSTATUS_LVIP_IMAGEINFO_NOT_EQUAL

Image info not equal.

5.50.2.10 #define LVSTATUS_LVIP_IMAGEINFO_NOT_INITIALIZED

The LvipImgInfo parameter is not initialized. The supplied LvipImgInfo parameter has invalid contents. Either initialize it with the LVIP_InitImgInfo() function or setup in your code all the members to appropriate values.

5.50.2.11 #define LVSTATUS_LVIP_INCOMPATIBLE_REF_FLAGS

The reference image has incompatible flags. When using the reference image info (in the LVIP_ApplyShading↩ Correction() function), it must have compatible flags as the source image. For example: LvipImgAttr_BottomUp flags must be same.

5.50.2.12 #define LVSTATUS_LVIP_INCOMPATIBLE_REF_PIXEL_FORMAT

The reference image is in incompatible pixel format. When using the reference image info (in the LVIP_Apply↩ ShadingCorrection() function), it must be in the same pixel format as the source image.

5.50.2.13 #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_FLAGS

Source and destination image info have incompatible flags. Some functions need to have both of source and destination image info flags compatible. This applies for example for the bitmap orientation (top-down versus bottom-up).

5.50.2.14 #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_PIXEL_FORMAT

The source and destination image info have different pixel format. You are trying to call some function which expects that both of source and destination image info has the same pixel format. But the destination image info has different pixel format set and it is not permitted to change this (by the LvipOption_ReallocateDst flag).

5.50.2.15 #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE

Destination image info size is different from the source image info size. It means that the function expects that the source and destination image info size is the same. Because it is not, and the destination image info cannot be changed using the LvipOption_ReallocateDst flag, this error happens.

5.50.2.16 #define LVSTATUS_LVIP_INCOMPATIBLE_SRC_AND_DST_SIZE_ROTATED

Rotated destination image info size is different to the expected size. It means that the function expects that the source width of image info has to be the same to the height of the destination image info (and vice versa) - and it is not - and additionaly destination image info cannot be changed using the LvipOption_ReallocateDst flag.

5.50.2.17 #define LVSTATUS_LVIP_INVALID_DST_POINTER

Pointer to the destination data image info or its data is invalid.

5.50.2.18 #define LVSTATUS_LVIP_INVALID_LUT_HANDLE

Invalid LUT handle passed as a parameter of the function. The LUT is in incompatible format which couldn't be used in this function.
General this happens when trying to use some Bayer-decoding function and a non-Bayer LUT is used or the LUT has been created for the different pixel format.

5.50.2.19 #define LVSTATUS_LVIP_INVALID_LUT_TYPE

Invalid LUT type passed as a parameter of the function. The LUT is in incompatible format which couldn't be used in this function.
General this happens when trying to use some Bayer-decoding function and a non-Bayer LUT is used or the LUT has been created for the different pixel format.

5.50.2.20 #define LVSTATUS_LVIP_INVALID_PIXEL_FORMAT

The source or destination image info has invalid or unexpected pixel format or the dwPixelIncrement value in the of LvipImgInfo structure. Another reason when this error code could be retrieved is when using the LVIP_Bmp↩ InfoToImgInfo() function with BITMAPINFOHEADER containing in its *biBitCount* member a different value than is supported. Supported values in BITMAPINFOHEADER are:

```
8           LVIP_PIXEL_FORMAT_MONO8
16          LVIP_PIXEL_FORMAT_RGB555_PACKED or LVIP_PIXEL_FORMAT_RGB565_PACKED
            (depends on biCompression member of the BITMAPINFOHEADER)
24          LVIP_PIXEL_FORMAT_RGB8_PACKED
32          LVIP_PIXEL_FORMAT_RGBA8_PACKED
```

5.50.2.21 #define LVSTATUS_LVIP_INVALID_POINTER

Invalid pointer. One of the pointers used by the function is NULL or invalid.

5.50.2.22 #define LVSTATUS_LVIP_INVALID_SRC_POINTER

Pointer to the source data image info or its data is invalid.

5.50.2.23 #define LVSTATUS_LVIP_JPEG_LOAD_FAILED

Loading from JPEG failed. More info in the LOG file.

5.50.2.24 #define LVSTATUS_LVIP_JPEG_SAVE_FAILED

Saving to JPEG failed. More info in the LOG file.

5.50.2.25 #define LVSTATUS_LVIP_LINEINCREMENT_TOO_BIG

Image line increment is too big. This some functions support only limited line increment. This currently applies only to LVIP_Deinterlace() function, which has a limit of 2048 $*$ 32bit RGB image - it means that line increment could be more than 8192 bytes.

5.50.2.26 #define LVSTATUS_LVIP_MEMORY_ALLOC_FAILED

Memory allocation failed. This error code happens when the operating system does not allow to allocate any new memory to this library.
When this error happens, there is a critical insufficience of memory; it might indicate a huge memory leak, typically caused by not deallocating used images when processing in a loop. Another cause could be big image dimensions resulting in an attempt to allocate a huge memory amount.
Typically this error code could be retrieved from LVIP_AllocateImageData() or LVIP_SaveToTiff() functions.

5.50.2.27 #define LVSTATUS_LVIP_NOT_BAYER_PIXEL_FORMAT

The PixelFormat is not BayerArray

5.50.2.28 #define LVSTATUS_LVIP_NOT_DISPLAYABLE_FORMAT

The image isn't in the displayable format. If there is a need to display an image, there is a need to have this image in a displayable format (in Windows one of the BMP pixel formats and the line increment alighed to 4 bytes). The image does not have such format and the automatic conversion to a displayable format was not enabled (using the third parameter of the LVIP_DisplayImage() function and optionally the LvipOption_ReallocateDst flag).

5.50.2.29 #define LVSTATUS_LVIP_SRC_IMAGEINFO_NO_DATA

The source image info structure has no data. The source image info has no image data - it means that the source image info does not point to any valid image data.
Use LVIP_AllocateImageData() function to allocate the buffer for the image or point the pData member(s) to a valid image.

5.50.2.30 #define LVSTATUS_LVIP_TIFF_CONTENTS_INVALID

The contents of the TIFF file is in the invalid or in the unsupported form. You are probably trying to read a TIFF file, which was not created by this library (see the LVIP_SaveToTiff() function)

5.50.2.31 #define LVSTATUS_LVIP_UNSUPPORTED

The requested function or format is not supported. If you are not sure which functionality is not supported, see New Electronic Technology Log Messages Receiver application for details.

5.50.2.32 #define LVSTATUS_LVIP_UNSUPPORTED_BMP_HEADER

BMP header is unsupported. Some members of BITMAPINFOHEADER (part of BITMAPINFO) have unexpected or unsupported values. Check if the BITMAPINFO has correct data and that is correctly filled up. Members *biSize* and *biCompression* have to be correctly filled up. The biCompression member has to be filled up with BI_RGB or BI_BITFIELDS values.

5.50.2.33 #define LVSTATUS_LVIP_UNSUPPORTED_COLOR_PLANES

The image uses color planes and the called function doesn't support it.

5.50.2.34 #define LVSTATUS_LVIP_UNSUPPORTED_DST_PIXEL_FORMAT

Unsupported pixel format of the destination. See documentation of function which returns this error code for supported destination pixel formats.

5.50.2.35 #define LVSTATUS_LVIP_UNSUPPORTED_REVERSION

The image uses reversion and the called function doesn't support it.

5.50.2.36 #define LVSTATUS_LVIP_UNSUPPORTED_SRC_PIXEL_FORMAT

Unsupported pixel format of the source. See documentation of the function which returns this error code for supported source pixel formats.

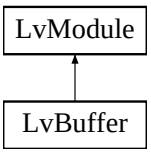
Chapter 6

Class Documentation

6.1 LvBuffer Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvBuffer:



Public Member Functions

- LvStatus AttachProcessBuffer (void *pDataPointer, size_t DataSize)
- LvStatus Queue ()
- LvStatus ParseChunkData (bool UpdateLayout=false)
- LvStatus SaveImageToBmpFile (const char *pFileName)
- LvStatus SaveImageToJpgFile (const char *pFileName, uint32_t Quality)
- LvStatus SaveImageToTifFile (const char *pFileName, uint32_t Options=0)
- LvStatus GetImgInfo (LvipImgInfo &ImgInfo, uint32_t Options=0)
- LvStatus GetLastPaintRect (int32_t *pX, int32_t *pY, int32_t *pWidth, int32_t *pHeight)
- LvStatus UniCalculateWhiteBalance ()
- LvHBuffer GetHandle ()
- void * GetUserPtr ()

Static Public Member Functions

- static LvStatus Open (LvStream *pStream, void *pDataPointer, size_t DataSize, void *pUserPointer, uint32↩ _t Options, LvBuffer *&pBuffer)
- static LvStatus Close (LvBuffer *&pBuffer)

Additional Inherited Members

6.1.1 Detailed Description

The LvBuffer class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class

instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

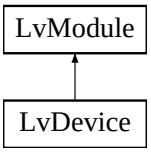
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.2 LvDevice Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvDevice:



Public Member Functions

- LvStatus GetNumberOfStreams (uint32_t *pNumberOfStreams)
- LvStatus GetStreamId (uint32_t Index, char *pStreamId, size_t Size)
- LvStatus GetStreamIdSize (uint32_t Index, size_t *pSize)
- LvStatus GetStreamId (uint32_t Index, std::string &sStreamId)
- LvStatus AcquisitionStart (uint32_t Options=0)
- LvStatus AcquisitionStop (uint32_t Options=0)
- LvStatus AcquisitionAbort (uint32_t Options=0)
- LvStatus AcquisitionArm (uint32_t Options=0)
- LvStatus SaveSettings (const char *pId, const char *pFileName, uint32_t Options)
- LvStatus LoadSettings (const char *pId, const char *pFileName, uint32_t Options)
- LvStatus LoadBatch (const char *pFileName)
- LvStatus UniSetLut (LvLUTSelector Selector, void *pLUT, size_t Size, uint32_t Options=0)
- LvStatus UniGetLut (LvLUTSelector Selector, void *pLUT, size_t Size, uint32_t Options=0)
- LvStatus FwGetFilePattern (uint32_t Which, char *pFilePattern, size_t Size)
- LvStatus FwLoad (uint32_t Which, const char *pFilePath)
- LvStatus FwGetLoadStatus (uint32_t Which, uint32_t *pCurrentByteCount, bool *pIsLoading)
- LvStatus OpenStream (const char *pStreamId, LvStream *&pStream)
- LvStatus CloseStream (LvStream *&pStream)
- LvStatus OpenEvent (LvEventType EventType, LvEvent *&pEvent)
- LvStatus CloseEvent (LvEvent *&pEvent)
- LvHDevice GetHandle ()

Static Public Member Functions

- static LvStatus Open (LvInterface *pInterface, const char *pDeviceId, LvDevice *&pDevice, LvDeviceAccess Access=LvDeviceAccess_Exclusive)
- static LvStatus Close (LvDevice *&pDevice)

Additional Inherited Members

6.2.1 Detailed Description

The LvDevice class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

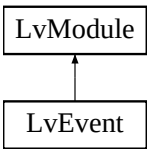
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.3 LvEvent Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvEvent:



Public Member Functions

- LvStatus Kill ()
- LvStatus Flush ()
- LvStatus WaitAndGetData (void *pBuffer, size_t *pSize, uint32_t Timeout=0xFFFFFFFF)
- LvStatus WaitAndGetNewBuffer (LvBuffer *&pBuffer, uint32_t Timeout=0xFFFFFFFF)
- LvStatus GetDataInfo (void *pInBuffer, size_t InSize, LvEventDataInfo Info, void *pBuffer, size_t *pSize, LvInfoDataType *pType=NULL, int32_t Param=0)
- LvStatus PutData (void *pBuffer, size_t Size)
- LvStatus SetCallback (LvEventCallbackFunct pFunction, void *pUserParam)
- LvStatus SetCallbackNewBuffer (LvEventCallbackNewBufferFunct pFunction, void *pUserParam)
- LvStatus StartThread ()
- LvStatus StopThread ()
- LvHEvent GetHandle ()

Static Public Member Functions

- static LvStatus Open (LvSystem *pSystem, LvEventType EventType, LvEvent *&pEvent)
- static LvStatus Open (LvDevice *pDevice, LvEventType EventType, LvEvent *&pEvent)
- static LvStatus Open (LvStream *pStream, LvEventType EventType, LvEvent *&pEvent)
- static LvStatus Close (LvEvent *&pEvent)

Additional Inherited Members

6.3.1 Detailed Description

The LvEvent class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class

instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.4 LvException Class Reference

`#include <sv.synview.class.h>`

Public Member Functions

- **LvException** (const char *pMessage, LvStatus Number) throw ()
- **LvException** (const LvException &e) throw ()
- const char * **Message** () throw ()
- LvStatus **Number** () throw ()

6.4.1 Detailed Description

Undefine LV_USE_STDLIB in case you do not want to use the standard template library. If LV_USE_STDLIB is defined, the functions returning strings are available also overloaded having a std::string& parameter for returning the string.
Define LV_USE_STDEXCEPTION in case you want to use the exception class from the standard library instead of LvException.
Call LvLibrary::SetThrowErrorEnable(true) in case you want to use the C++ exceptions of the LvException type to be thrown when the function returns a status not equal to LVSTATUS_OK. Then you can use the error handling in the form shown in the example below:

```
try
{
    m_pDevice->AcquisitionStart();
    // ... and more SynView API calls, without checking the return value
}
catch (LvException e)
{
    DisplayErrorMsg(e.Message(), e.Number());
    return;
}
```

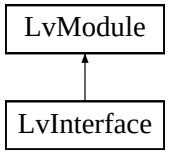
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.5 LvInterface Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvInterface:



Public Member Functions

- LvStatus UpdateDeviceList (uint32_t Timeout=0xFFFFFFFF)
- LvStatus GetNumberOfDevices (uint32_t *pDevices)
- LvStatus GetDeviceId (uint32_t Index, char *pDeviceId, size_t Size)
- LvStatus GetDeviceIdSize (uint32_t Index, size_t *pSize)
- LvStatus GetDeviceId (uint32_t Index, std::string &sDeviceId)
- LvStatus FindDevice (LvFindBy FindBy, const char *pFindStr, char *pDeviceId, size_t Size)
- LvStatus FindDevice (LvFindBy FindBy, const char *pFindStr, std::string &sDeviceId)
- LvHInterface GetHandle ()
- LvStatus OpenDevice (const char *pDeviceId, LvDevice *&pDevice, LvDeviceAccess Access=LvDevice↩Access_Exclusive)
- LvStatus CloseDevice (LvDevice *&pDevice)

Static Public Member Functions

- static LvStatus Open (LvSystem *pSystem, const char *pInterfaceId, LvInterface *&pInterface)
- static LvStatus Close (LvInterface *&pInterface)

Additional Inherited Members

6.5.1 Detailed Description

The LvInterface class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.6 LvipImgInfo Struct Reference

```
#include <sv.synview.defs.h>
```

Public Attributes

- uint32_t StructSize
- uint32_t Width
- uint32_t Height
- uint32_t PixelFormat
- uint32_t Attributes
- uint32_t BytesPerPixel
- uint32_t LinePitch
- uint8_t * pData
- uint8_t * pDataR
- uint8_t * pDataG
- uint8_t * pDataB

6.6.1 Detailed Description

Image Info structure. Each image handled by the library must be described by the LvipImgInfo structure.
Although you can set the Image Info members directly, it is highly recommended to use the LvipInitImgInfo() function for the structure initialization.

6.6.2 Member Data Documentation

6.6.2.1 uint32_t LvipImgInfo::Attributes

Image attributes. OR-ed definitions from LvipImgAttr definitions.

6.6.2.2 uint32_t LvipImgInfo::BytesPerPixel

Size of one pixel in bytes.

6.6.2.3 uint32_t LvipImgInfo::Height

Height of the image in pixels.

6.6.2.4 uint32_t LvipImgInfo::LinePitch

Size of one line in bytes.
Example:

```
8-bit mono image: LineIncrement = Width;
24-bit RGB image: LineIncrement = Width * 3;
```

However, when the LvipImgAttr_DWordAligned attribute is used, the line increment must be rounded up to whole double-words, so the calculation would then look like this:

```
8-bit mono image: LineIncrement = (Width+3)/4 * 4;
24-bit RGB image: LineIncrement = ((Width*3)+3)/4 * 4;
```

6.6.2.5 uint8_t* LvipImgInfo::pData

Pointer to image data. If color planes are not used, this member points to the data of the image. Use LvipAllocate↩ ImageData() to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this LvipImgInfo, use the LvipImgAttr_NotDataOwner attribute.

6.6.2.6 uint8_t* LvipImgInfo::pDataB

If color planes are used, this member points to the Blue plane data of the image. Use LvipAllocateImageData() to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this LvipImgInfo, use the LvipImgAttr_NotDataOwner attribute.

6.6.2.7 uint8_t* LvipImgInfo::pDataG

If color planes are used, this member points to the Green plane data of the image. Use LvipAllocateImageData() to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this LvipImgInfo, use the LvipImgAttr_NotDataOwner attribute.

6.6.2.8 uint8_t* LvipImgInfo::pDataR

If color planes are used, this member points to the Red plane data of the image. Use LvipAllocateImageData() to allocate the buffer for the image. If you set the pointer to an existing image, which is not owned by this LvipImgInfo, use the LvipImgAttr_NotDataOwner attribute.

6.6.2.9 uint32_t LvipImgInfo::PixelFormat

Pixel format of the image which is saved in this structure. One of the LvPixelFormat
In case of color planes, the pixel format applies to one plane, so use only the MONO formats for the planes. For example for 3x8-bit RGB use the LvPixelFormat_Mono8 format.

6.6.2.10 uint32_t LvipImgInfo::StructSize

Size of image info structure. Should be set to the sizeof(LvipImgInfo). This member may be used in the future versions for the compatibility check.

6.6.2.11 uint32_t LvipImgInfo::Width

Width of the image in pixels.

The documentation for this struct was generated from the following file:

- include/sv.synview.defs.h

6.7 LvLibrary Class Reference

```
#include <sv.synview.class.h>
```

Static Public Member Functions

- static uint32_t GetVersion ()
- static LvStatus OpenLibrary ()
- static LvStatus CloseLibrary ()
- static void GetErrorMessage (LvStatus Error, char *pMessage, size_t Size)
- static std::string GetErrorMessage (LvStatus Error)
- static void GetLastErrorMessage (char *pMessage, size_t Size)
- static std::string GetLastErrorMessage ()
- static void Log (const char *pLogMessage)
- static LvStatus GetLibInfo (LvEnum Info, int32_t *pInfo, int32_t Param=0)
- static LvStatus GetLibInfoStr (LvEnum Info, char *pInfoStr, size_t Size, int32_t Param=0)
- static LvStatus GetLibInfoStrSize (LvEnum Info, size_t *pSize, int32_t Param=0)
- static LvStatus GetLibInfoStr (LvEnum Info, std::string &sInfo, int32_t Param=0)
- static LvStatus UpdateSystemList ()
- static LvStatus GetNumberOfSystems (uint32_t *pNumberOfSystems)
- static LvStatus GetSystemId (uint32_t Index, char *pSystemId, size_t Size)
- static LvStatus GetSystemIdSize (uint32_t Index, size_t *pSize)
- static LvStatus GetSystemId (uint32_t Index, std::string &sSystemId)
- static void SetThrowErrorEnable (bool bEnable)

6.7.1 Detailed Description

The LvLibrary class has all its members static and it is not possible to create the instance of this class. You can consider its methods as global functions.

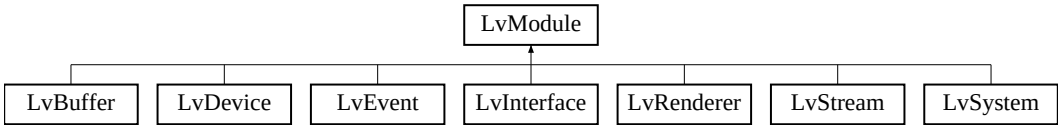
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.8 LvModule Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvModule:



Public Member Functions

- LvStatus GetNumFeatures (LvFtrGroup FtrGroup, uint32_t *pNumFeatures)
- LvStatus GetFeatureAt (LvFtrGroup FtrGroup, uint32_t Index, LvFeature *pFeature, uint32_t *pLevel=NULL)
- LvStatus GetFeatureByName (LvFtrGroup FtrGroup, const char *pName, LvFeature *pFeature)
- bool IsImplemented (LvFeature Feature)
- bool IsImplementedByName (LvEnum FeatureGroup, const char *pName)
- bool IsAvailable (LvFeature Feature)
- bool IsAvailableByName (LvEnum FeatureGroup, const char *pName)
- bool IsReadable (LvFeature Feature)
- bool IsWritable (LvFeature Feature)
- bool IsAvailableEnumEntry (LvFeature Feature, LvEnum EnumEntry)
- bool IsImplementedEnumEntry (LvFeature Feature, LvEnum EnumEntry)
- LvStatus GetType (LvFeature Feature, LvFtrType *pFtrType, LvFtrGui *pFtrGui=NULL, LvFtrGroup *pFtr↩ Group=NULL)
- LvStatus GetBool (LvFeature Feature, bool *pValue)
- LvStatus SetBool (LvFeature Feature, bool Value)
- LvStatus GetInt32 (LvFeature Feature, int32_t *pValue)
- LvStatus SetInt32 (LvFeature Feature, int32_t Value)
- LvStatus GetInt32Range (LvFeature Feature, int32_t *pMinValue, int32_t *pMaxValue, int32_t *pIncrement)
- LvStatus GetInt64 (LvFeature Feature, int64_t *pValue)
- LvStatus SetInt64 (LvFeature Feature, int64_t Value)
- LvStatus GetInt64Range (LvFeature Feature, int64_t *pMinValue, int64_t *pMaxValue, int64_t *pIncrement)
- LvStatus GetInt (LvFeature Feature, int64_t *pValue)
- LvStatus SetInt (LvFeature Feature, int64_t Value)
- LvStatus GetIntRange (LvFeature Feature, int64_t *pMinValue, int64_t *pMaxValue, int64_t *pIncrement)
- LvStatus GetFloat (LvFeature Feature, double *pValue)
- LvStatus SetFloat (LvFeature Feature, double Value)
- LvStatus GetFloatRange (LvFeature Feature, double *pMinValue, double *pMaxValue, double *p↩ Increment=NULL)
- LvStatus GetString (LvFeature Feature, char *pValue, size_t Size)
- LvStatus GetStringSize (LvFeature Feature, size_t *pSize)
- LvStatus GetString (LvFeature Feature, std::string &sValue)

- LvStatus SetString (LvFeature Feature, const char *pValue)
- LvStatus GetBuffer (LvFeature Feature, void *pBuffer, size_t Size)
- LvStatus GetBufferSize (LvFeature Feature, size_t *pSize)
- LvStatus SetBuffer (LvFeature Feature, void *pBuffer, size_t Size)
- LvStatus GetPtr (LvFeature Feature, void **ppValue)
- LvStatus SetPtr (LvFeature Feature, void *pValue)
- LvStatus GetEnum (LvFeature Feature, LvEnum *pValue)
- LvStatus SetEnum (LvFeature Feature, LvEnum Value)
- LvStatus GetEnumStr (LvFeature Feature, char *pSymbolicName, size_t Size)
- LvStatus GetEnumStr (LvFeature Feature, std::string &sSymbolicName)
- LvStatus SetEnumStr (LvFeature Feature, const char *pSymbolicName)
- LvStatus GetEnumValByStr (LvFeature Feature, const char *pSymbolicName, LvEnum *pValue, LvFtr↩Access *pFtrAccess=NULL)
- LvStatus GetEnumStrByVal (LvFeature Feature, LvEnum Value, char *pSymbolicName, size_t Symbolic↩NameSize, LvFtrAccess *pFtrAccess=NULL)
- LvStatus GetEnumStrByVal (LvFeature Feature, LvEnum Value, std::string &sSymbolicName, LvFtrAccess *pFtrAccess=NULL)
- LvStatus CmdExecute (LvFeature Feature, uint32_t Timeout=0)
- LvStatus CmdIsDone (LvFeature Feature, bool *pIsDone)
- LvStatus GetAccess (LvFeature Feature, LvFtrAccess *pFtrAccess)
- LvStatus GetVisibility (LvFeature Feature, LvFtrVisibility *pFtrVisibility)
- LvStatus GetInfo (LvFeature Feature, LvFtrInfo FtrInfo, int32_t *pInfo, int32_t Param=0)
- LvStatus GetInfoStr (LvFeature Feature, LvFtrInfo FtrInfo, char *pInfoStr, size_t Size, int32_t Param=0)
- LvStatus GetInfoStrSize (LvFeature Feature, LvFtrInfo FtrInfo, size_t *pSize, int32_t Param=0)
- LvStatus GetInfoStr (LvFeature Feature, LvFtrInfo FtrInfo, std::string &sInfoStr, int32_t Param=0)
- LvStatus RegisterFeatureCallback (LvFeature Feature, LvFeatureCallbackFunct pFunction, void *pUser↩Param=NULL, void *pFeatureParam=NULL)
- LvStatus StartPollingThread (uint32_t PollingTime=1000, bool PollChildren=false)
- LvStatus StopPollingThread ()
- LvStatus Poll ()

Protected Attributes

- LvHModule m_hModule

6.8.1 Detailed Description

The base class for all modules. It provides methods for manipulating the features, if the module provides any. This class cannot be instantiated, it only serves as a base class.

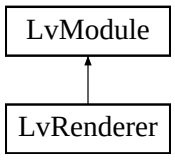
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.9 LvRenderer Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvRenderer:



Public Member Functions

- LvStatus SetWindow (void *pDisplay, int64_t hWindow)
- LvStatus DisplayImage (LvBuffer *pBuffer, uint32_t RenderFlags=0)
- LvStatus Repaint (uint32_t RenderFlags=0)
- LvHRenderer GetHandle ()

Static Public Member Functions

- static LvStatus Open (LvStream *pStream, LvRenderer *&pRenderer)
- static LvStatus Close (LvRenderer *&pRenderer)

Additional Inherited Members

6.9.1 Detailed Description

The LvRenderer class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

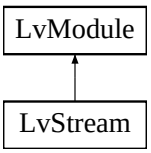
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.10 LvStream Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvStream:



Public Member Functions

- LvStatus GetBufferAt (uint32_t BufferIndex, LvBuffer *&pBuffer)
- LvStatus FlushQueue (LvQueueOperation Operation)
- LvStatus Start (uint32_t StartFlags=0x00000000, uint32_t ImagesToAcquire=0xFFFFFFFF)
- LvStatus Stop (uint32_t StopFlags=0x00000000)
- LvHStream GetHandle ()
- LvStatus OpenBuffer (void *pDataPointer, size_t DataSize, void *pUserPointer, uint32_t Options, LvBuffer *&pBuffer)
- LvStatus CloseBuffer (LvBuffer *&pBuffer)
- LvStatus OpenEvent (LvEventType EventType, LvEvent *&pEvent)
- LvStatus CloseEvent (LvEvent *&pEvent)
- LvStatus OpenRenderer (LvRenderer *&pRenderer)
- LvStatus CloseRenderer (LvRenderer *&pRenderer)

Static Public Member Functions

- static LvStatus Open (LvDevice *pDevice, const char *pStreamId, LvStream *&pStream)
- static LvStatus Close (LvStream *&pStream)

Additional Inherited Members

6.10.1 Detailed Description

The LvStream class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

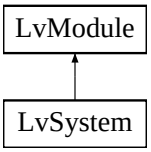
The documentation for this class was generated from the following file:

- include/sv.synview.class.h

6.11 LvSystem Class Reference

`#include <sv.synview.class.h>`

Inheritance diagram for LvSystem:



Public Member Functions

- LvStatus UpdateInterfaceList (uint32_t Timeout=0xFFFFFFFF)
- LvStatus GetNumberOfInterfaces (uint32_t *pNumberOfInterfaces)
- LvStatus GetInterfaceId (uint32_t Index, char *pInterfaceId, size_t Size)
- LvStatus GetInterfaceIdSize (uint32_t Index, size_t *pSize)
- LvStatus GetInterfaceId (uint32_t Index, std::string &sInterfaceId)
- LvStatus FindInterface (LvFindBy FindBy, const char *pFindStr, char *pInterfaceId, size_t Size)
- LvStatus FindInterface (LvFindBy FindBy, const char *pFindStr, std::string &sInterfaceId)
- LvHSystem GetHandle ()
- LvStatus OpenInterface (const char *pInterfaceId, LvInterface *&pInterface)
- LvStatus CloseInterface (LvInterface *&pInterface)
- LvStatus OpenEvent (LvEventType EventType, LvEvent *&pEvent)
- LvStatus CloseEvent (LvEvent *&pEvent)

Static Public Member Functions

- static LvStatus Open (const char *pSystemId, LvSystem *&pSystem)
- static LvStatus Close (LvSystem *&pSystem)

Additional Inherited Members

6.11.1 Detailed Description

The LvSystem class. @ note For all the SynView module classes you cannot use the new and delete operators directly (the constructor and destructor are private). Instead, the static methods for opening and closing the class instance assure that if the opening is successful, you get a valid pointer, otherwise you get a NULL pointer. Also, the closing functions set the pointer back to NULL. Another advantage is that these functions return a status value, which can clarify the error nature, if the opening or closing fails.

The documentation for this class was generated from the following file:

- include/sv.synview.class.h