XIMEA CamTool

Generated by Doxygen 1.8.10

Tue Sep 27 2016 15:36:28

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 CxHsiCameraParams::CxBand Class Reference

Public Member Functions

- bool **isValid** () const
- double peakInRange (int iMinNm, int iMaxNm) const

  Returns peak value in a specified range.
- QString name () const

  Returns band name as "123 nm".

Public Attributes

- bool m_bSelected

  Flags if the signal of this band is within specifications (true) or not (false). Usage of non-selected bands will result in wrong spectra.
- QVector< double > m_vecPeakWaveLengths

  Central wavelength of peaks [nm]. At leask one peak should be defined.
- QVector< double > m_vecCalibData

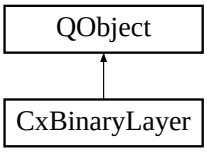
  Band response data, tag <response_composition> (not required for images)

The documentation for this class was generated from the following file:

- HsiCameraParams.h

4.2 CxBinaryLayer Class Reference

Inheritance diagram for CxBinaryLayer:



Signals

- void **changed** (TxBinaryLayerID idLayer)
- void **visibilityChanged** (TxBinaryLayerID idLayer)

Public Member Functions

- **CxBinaryLayer** (uint uiWidth, uint uiHeight)
- TxBinaryLayerID id () const

  ID of the layer.
- void setId (TxBinaryLayerID iId)

  Used by CxBinaryLayerContainer.
- QString name () const

  A custom name of the layer (defalt is "Layer #id").
- void setName (const QString &sName)

  Sets the custom layer name.
- const SxPicBuf & **picbuf** () const
- SxPicBuf & picbuf ()

  Use with caution!! Change only the buffer's data (m_pData) but nothing more!!!!
- void reset ()

  Resets the layer's data to zeros.
- void release ()

  Releases the layer's data.
- bool alloc (uint uiWidth, uint uiHeight)

  Allocs the layer's data. If the data exists then the allocation is not done and the method returns false.
- const QColor **color** () const
- void **setColor** (const QColor &color)
- int **alpha** () const
- void **setAlpha** (int iAlpha)
- bool **visible** () const
- void **setVisible** (bool bVisible)
- void notifyChanged ()

  When the layer's data is modified, call this method.

Static Public Member Functions

- static bool MakeColoredBinary (const CxBinaryLayer *pLayer, SxPicBuf &picRgb32)

  Converts the binary layer to colored RGB32 image with premultiplied rgb values (see QImage::Format_ARGB32_↩ Premultiplied)
- static bool BlendRGBWithBinary (const SxPicBuf &picRgb, const CxBinaryLayer *pLayer, SxPicBuf &pic↩ BlendedRgb)

  Blends the given RGB image (RGB32, RGB24, RGB565) with the binary layer. The resulting `picBlendedRgb` is the image of the same format as `picRgb`.

Protected Member Functions

- CxBinaryLayer & operator= (const CxBinaryLayer &rOther)

  Prevent to use the operator=.
- void **setAsShallowCopyOf** (const CxBinaryLayer &rOther)

Private Attributes

- TxBinaryLayerID **m_iId**
- QString **m_sName**
- SxPicBuf **m_binary**
- QColor **m_color**
- bool **m_bVisible**
- bool **m_bChanged**

The documentation for this class was generated from the following file:

- BinaryLayer.h

4.3 CxBinaryLayerContainer Class Reference

Public Member Functions

- TxBinaryLayerID addLayer (CxBinaryLayer *pLayer, bool bSetAutoColor=true)

  Add new layer and assign ID.
- void deleteLayer (TxBinaryLayerID idLayer)

  Deletes (and dallocates) the layer with given ID.
- void deleteAllLayers ()

  Deletes all layers from container.
- CxBinaryLayer * removeLayer (TxBinaryLayerID idLayer)

  Remove the layer from container and returns its pointer.
- QList< TxBinaryLayerID > listLayers () const

  List of all layers.
- CxBinaryLayer * layerWithId (TxBinaryLayerID idLayer) const

  Returns a pointer to the layer with given id. If it not exists then it returns NULL.
- bool hasVisibleLayer () const

  Returns true if at least one layer is visible.
- QList< TxBinaryLayerID > listVisibleLayers () const

  List of all visible layers.
- void saveLayersVisibilityState ()

  Saves the per-layer visibility state for later.
- void restoreLayersVisibilityState ()

  Restores the layers visibility state as saved in saveLayersVisibilityState();.
- void hideAllLayers ()

  Sets all layers as invisible.

Private Member Functions

- QColor **autoColor** ()

Private Attributes

- TxBinaryLayerMap **m_mapLayers**
- QColor m_colCurrent

  For auto color.
- int m_idMax

  ID generator.
- QMap< TxBinaryLayerID, bool > **m_mapVisibilityStates**

4.3.1 Member Function Documentation

4.3.1.1 void CxBinaryLayerContainer::deleteAllLayers ( )

Deletes all layers from container.

See also

deleteLayer()

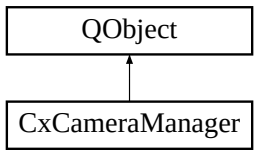
The documentation for this class was generated from the following file:

- BinaryLayerContainer.h

4.4 CxCameraManager Class Reference

The class that provides all necessary methods to access cameras and to set up camera parameters.

Inheritance diagram for CxCameraManager:



Public Slots

- void finishAll ()

  Stops all cameras and closes them. This method should be called on appliction exit.

Signals

- void acquisitionStarted (XICORE_HANDLE hCamera)

  The signal is emitted when the acquistion on the given camera was started.
- void acquisitionFinished (XICORE_HANDLE hCamera)

  The signal is emitted when the acquistion on the given camera was finished.
- void cameraOpened (XICORE_HANDLE hCamera)

  The signal is emitted when camera is opened.
- void cameraClosed (XICORE_HANDLE hCamera)

  The signal is emitted when the camera was closed.
- void cameraLost (XICORE_HANDLE hCamera)

  The signal is emitted when camera disconnected during app session.
- void cameraAcqError (XICORE_HANDLE hCamera, int iErrorCode, const QString &sInFunction)

  The signal is emitted when camera acquisition fails.
- void imageFormatChanging (XICORE_HANDLE hCamera)

  The signal is emitted when format of camera image (dimension, components..) is to be changed.
- void imageFormatChanged (XICORE_HANDLE hCamera)

  The signal is emitted when format of the camera image (dimension, komponents..) changed.

Public Member Functions

- ~CxCameraManager ()

  Destructor.
- uint cameraCount () const

  Gets the number of available cameras connected to PC.
- bool cameraInfo (uint uiCameraIdx, SxCameraInfo &aInfo) const

  Gets the info about camera with given index `uiCameraIdx`.
- bool **cameraSerialNumber** (uint uiCameraIdx, QString &sSerialNumber) const
- XICORE_HANDLE openCamera (uint uiCameraIdx, SxCameraInfo *pPreloadedInfo=NULL)

  Opens a camera with given index and returns its handle.
- uint openedCameraCount ()

  Gets the number of opened cameras.
- QList< XICORE_HANDLE > openedCameras ()

  Gets a list of handles to the opened cameras.
- bool **isCameraOpened** (XICORE_HANDLE hCamera)
- bool **closeCamera** (XICORE_HANDLE hCamera)
- bool **startAcquisition** (XICORE_HANDLE hCamera)
- bool **stopAcquisition** (XICORE_HANDLE hCamera)
- uint **runningCameraCount** ()
- bool **isCameraRunning** (XICORE_HANDLE hCamera)
- QList< XICORE_HANDLE > **runningCameras** ()
- CxCameraParameters * cameraParameters (XICORE_HANDLE hCamera)

  Gets the camera parameters.
- bool readCameraHsiParams (XICORE_HANDLE hCamera, const SxCameraInfo &info, CxHsiCameraParams *pHsiParams) const

  Reads HSI calibration XML from camera.
- void notifyCameraFormatChanging (XICORE_HANDLE hCamera)

  Call this method to notify the rest of the application that the camera format (image dimensions, components, ...) are to be changed.
- void notifyCameraFormatChanged (XICORE_HANDLE hCamera)

  Call this method to notify rest of the application that the camera format (image dimensions, components, ...) has been changed.
- void notifyCameraLost (XICORE_HANDLE hCamera)

  *Call this method to notify rest of the application that the camera just disconnected */.*
- void notifyCameraAcqError (XICORE_HANDLE hCamera, int iErrorCode, const QString &sInFunction)

  *Call this method to notify rest of the application that the camera acquisition is failing */.*
- bool cameraPicbufInfo (XICORE_HANDLE hCamera, SxPicBufInfo &picInfo)

  Gets the picbuf info currently produced by the given camera.
- XICORE_HANDLE cameraHandleBySerialId (const QString &sSerialId)

  Gets the camera handle by the serial id of the camera.
- bool newProcessChainEnabled () const

  State of XI_PRM_NEW_PROCESS_CHAIN_ENABLE parameter.
- void **setNewProcessChainEnabled** (bool bEnable)
- bool **newProcessChainSupported** () const
- void saveSettings (QSettings *pSettings)

  store some options to app settings
- void loadSettings (QSettings *pSettings)

  load some options from app settings
- void **createCameraSimulator** ()
- void **disconnectCameraSimulator** ()
- void **createHsiCameraSimulator** ()

Static Public Member Functions

- static CxCameraManager * instance ()

  Gets the instance if this object.

Private Member Functions

- CxCameraManager ()

  Private constructor (the class is singleton).

Private Attributes

- QMap< XICORE_HANDLE, CxCameraParameters * > m_mapCameras
- QSet< XICORE_HANDLE > m_setRunningCameras
- QMutex m_lock
- XICORE_HANDLE m_hLastCameraLost
- XICORE_HANDLE m_hLastCameraAcqError
- QString **m_sLastCameraAcqErrorFunc**
- bool **m_bEnableNewProcessChain**
- bool **m_bNewProcessingChainSupported**

4.4.1 Detailed Description

The class that provides all necessary methods to access cameras and to set up camera parameters.

4.4.2 Member Function Documentation

4.4.2.1 void CxCameraManager::cameraClosed (XICORE_HANDLE *hCamera*) [signal]

The signal is emitted when the camera was closed.

Remarks

The camera handle `hCamera` is invalid in the moment of emitting of this signal, so any attempt to communitation with the camera fails.

4.4.2.2 bool CxCameraManager::cameraInfo (uint *uiCameraIdx*, SxCameraInfo & *aInfo*) const

Gets the info about camera with given index `uiCameraIdx`.

Remarks

The index `uiCameraIdx` is related to cameras count returned by the cameraCount() method (NOT by the openedCameraCount() method!).

4.4.3 Member Data Documentation

4.4.3.1 XICORE_HANDLE CxCameraManager::m_hLastCameraAcqError [private]

Do not emit the cameraAcqError signal more then once

4.4.3.2 XICORE_HANDLE CxCameraManager::m_hLastCameraLost [private]

Do not emit the cameraLost signal more then once

4.4.3.3 QMutex CxCameraManager::m_lock [private]

The lock of the manager.

4.4.3.4 QMap<XICORE_HANDLE, CxCameraParameters*> CxCameraManager::m_mapCameras [private]

Camera parameters by the camera handle

4.4.3.5 QSet<XICORE_HANDLE> CxCameraManager::m_setRunningCameras [private]

Set of camera where the acquistion is running.

The documentation for this class was generated from the following file:

- CameraManager.h

4.5 CxCameraParam Class Reference

Classes

- struct SxEnumItem

  Struct holding all info for each item of enumerated parameter (ecpEnum)

Public Types

- enum **ExType** {
  ecpInvalid, **ecpInt**, **ecpFloat**, **ecpBool**,
  ecpString, **ecpEnum**, **ecpCommand** }
- enum **ExRepresentation** { **erepDefault**, **erepLogarithmic** }
- enum ExVisibility {
  evisInvisible = 0, **evisBeginner** = 1, **evisExpert** = 2, **evisGuru** = 3,
  evisMaxLevel = evisGuru }

  Camera parameter visibility level (selected in app GUI)

Public Member Functions

- **CxCameraParam** (ExType eType, const QString &sXiApiName, const QString &sCaption, const QString &s↩ UIGroup, ExVisibility eVisibility)
- CxCameraParam ()

  This default constructor should not be used.
- void readInfo (XICORE_HANDLE hCamera)

  Reads m_vMin, m_vMax and m_vIncrement from xiApi, should be called only once.
- bool readFromCamera (XICORE_HANDLE hCamera)

  Reads current value from the camera.
- bool writeToCamera (XICORE_HANDLE hCamera) const

  Sends the current value to camera.
- void filterEnumValues (XICORE_HANDLE hCamera)

Filter out enum values.

- void limitEnumsWorkaround (XICORE_HANDLE hCamera)

  Filter out enum values (old hack)
- bool updateIfInvalidatedWithParam (XICORE_HANDLE hCamera, const QString &sInvalidator)

  Returns true if it has sInvalidator among its invalidators.
- bool isVisibleInGUI () const

  Parameter should be visible in Camera settings.
- bool isVisibleForLevel (ExVisibility eLevel) const

  Same as isVisibleInGUI, but checks visibility level too.
- bool belongsToGroup (const QString &sUIGroup, ExVisibility eLevel=evisMaxLevel) const

  Same as isVisibleForLevel, but verifies the group membership too.
- bool isInvalidatedByParam (const QString &sParam) const

  Returns true if this parameter is invalidated by another parameter.
- CxValue paramValue () const

  returns current value. When enum, sends the value, not selection index.
- int indexOfEnumValue (const CxValue &vValue) const

  Returns index if value is among this param's enum values.
- bool setParamValue (const CxValue &vValue)

  sets parameter value. When enum, sets the correct selection index.

Public Attributes

- QString m_sXiApiName

  Parameter name as in xiApi (e.g. XI_PRM_EXPOSURE)
- QString m_sCaption

  User-friendly name of this parameter.
- QString m_sTooltip

  Some longer text about this parameter.
- QString m_sUIGroup

  Name of the section in camera settings this parameter should belong to.
- bool m_bReadOnly

  If we can write this parameter to camera.
- ExVisibility m_eVisible

  Tells that this camera parameter should be present in GUI.
- bool m_bNeedsRegularRefresh

  We should poll this parameter to get new updates automatically (even if there is no invalidator)
- bool m_bRequiresAcqStop

  Acquisition loop has to be stopped before settings this parameter.
- ExRepresentation m_eRepresentation

  How the parameter should look like in GUI.
- ExType m_eType

  Type of the parameter: int, float, bool, string, enum, command...
- CxValue m_vValue

  Current value of the parameter, type is selected accordnig to m_eType.
- CxValue m_vMin

  Minimum value as read from xiApi or defined in XML.
- CxValue m_vMax

  Maximum value as read from xiApi or defined in XML.
- CxValue m_vIncrement

  Value increment as read from xiApi or defined in XML.

- QVector< SxEnumItem > m_vecEnumValues

  Special case for enumerated parameter, selection stored in m_vValue.intVal. Filtered using XI_PRM_INFO_SETTABLE.
- CxValue m_vAppDefault

  Default value as defined in XML (may be undefined - type is etInvalid)
- bool m_bCameraError

  There was an error when getting the value of this parameter. Probably not supported by camera.

Private Attributes

- CxValueRegister **m_aValueRegister**
- CxValueRegister **m_aMinRegister**
- CxValueRegister **m_aMaxRegister**
- CxValueRegister **m_aIncRegister**
- QVector< SxEnumItem > m_vecEnumValuesUnfiltered

  All enum values as defined in XML. Filtered values according to camera caps are in m_vecEnumValues.

Friends

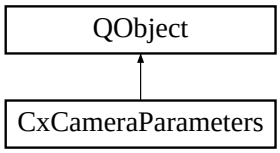
- class **CxCameraCapsLoader**
- class **CxCameraParameters**

The documentation for this class was generated from the following file:

- CameraParameters.h

4.6 CxCameraParameters Class Reference

Inheritance diagram for CxCameraParameters:



Signals

- void parametersNeedRefresh (const QStringList &lstParamNames)

  Signal for GUI that it should rehresh controls for these parameters.

Public Member Functions

- **CxCameraParameters** (XICORE_HANDLE hCamera, const SxCameraInfo &aCamInfo)
- XICORE_HANDLE cameraHandle () const

  Returns camera handle to which this object is connected.
- bool loadFromXMLDescription ()

  Clears everything and uses the definition stored in the external xml file for this camera.
- void saveDescriptionToXML (const QString &sFile)

  For testing purposes, inverse to loadFromXMLDescription.

- const SxCameraInfo & **cameraInfo** () const
- const CxHsiCameraParams & **hsiParams** () const
- const QSize & sensorResolution () const

  Physical resolution of the sensor.
- const QSize & downsampledResolution () const

  Maximum resolution at the current downsampling (when no ROI active)
- const QRectF & lastROI () const

  ROI relatively to resolution as float vals (i.e. 1.0=full width)
- void setLastROI (const QRectF &rc)

  Tells us to remember provided ROI geometry for next time.
- bool setCameraUserId (const QString &sUserId)

  Write custom User ID to camera flash memory.
- void saveParamValues (const QString &sStorageFolder)

  Save values of all parameters to file, so that we can load it next time (called on app exit)
- void saveParamValuesToFile (const QString &sFilename)

  Save values of all parameters to custom file.
- QString saveParamValuesToString ()

  Save values of all parameters to string.
- bool loadParamValues (const QString &sStorageFolder, QStringList *plstLoadedParams=NULL)

  Load saved values, called at the start of the application.
- bool loadParamValuesFromFile (const QString &sFilename, QStringList *plstLoadedParams=NULL, bool b↩ NotifyGUI=false)

  Load saved value from custom file.
- bool loadParamValuesFromString (const QString &sString, QStringList *plstLoadedParams=NULL, bool b↩ NotifyGUI=false)

  Load saved value from string.
- void **saveAdditionalSettingsToXml** (QDomDocument *pXmlDoc, QDomElement *pRootElement)
- void **loadAdditionalSettingsFromXml** (QDomElement *pRootElement)
- bool addParam (const CxCameraParam &aParam)

  Adds new parameter to our list.
- CxCameraParam * findParam (const QString &sXiApiParamName)

  Returns NULL when not found.
- CxValue currentParamValue (const QString &sXiApiParamName)

  Refresh current value from camera and return it.
- const QStringList & paramsOrderList () const

  Returns list with parameter names in the order they appeared in xml.
- bool commitSingleParamValue (const QString &sXiApiParamName, bool bRefreshCommitedParamsIn↩ Gui=false)

  Adds parameter to batch and commits it right away.
- void addParamCommitToBatch (const QString &sXiApiParamName)

  Marks the parameter to be sent to camera in this batch.
- bool endCommitBatch (bool bRefreshCommitedParamsInGui=false)

  Sends all parameters in batch to camera, and checks invalidators.
- bool setParamDirect (const QString &sXiApiParamName, const CxValue &vValue)

  Use commitSingleParamValue instead of this.
- bool getParamDirect (const QString &sXiApiParamName, CxValue &vValue) const

  Note: vValue.m_eType must be correctly filler.
- void setROIRectParams (int x, int y, int w, int h, bool bCheckMinMax=false)

  Sends new ROI to camera, while checking min/max/inc optionally.
- void setROIRectParams (const QRect &rc, bool bCheckMinMax=false)

  Sends new ROI to camera, while checking min/max/inc optionally.

- QRect currentROIRectParams ()

  Returns current ROI as rectangle.
- void setImageFormatKeepRoi (int eXiImgFormat)

  Switches XI_PRM_IMAGE_DATA_FORMAT and forces to keep current ROI.
- bool rereadHsiParamsForRange (int iStartNm, int iEndNm)

  switch to different filter range (when lens changed, e.g. for 5x5)
- bool **setWhiteImage** (const SxPicBuf &aPic, const CxImageMetadata &aMetadata)
- bool **setDarkImage** (const SxPicBuf &aPic, const CxImageMetadata &aMetadata)
- bool **setDarkBiasImage** (const SxPicBuf &aPic, const CxImageMetadata &aMetadata)
- CxHsiCorrectionImage & **whiteImage** ()
- CxHsiCorrectionImage & **darkImage** ()
- CxHsiCorrectionImage & **darkBiasImage** ()
- ExHsiCorrection **hsiCorrection** ()
- void **setHsiCorrection** (ExHsiCorrection eCorr)
- bool **isCorrectionReady** ()
- void **notifyShadingChanged** ()

Static Public Member Functions

- static bool queryCameraStoredInParamFile (const QString &sFilename, QString *psCamera, QString *ps↩ Serial)

  Returns camera model and serial stored in xml file with values.
- static bool queryCameraStoredInParamString (const QString &sString, QString *psCamera, QString *ps↩ Serial)

  Returns camera model and serial stored in xml string with values.

Public Attributes

- TxShadingPresetList **m_lstShadingPresets**

Private Slots

- void on_timerRefreshParams_update ()

  Timer for auto-refreshed parameters (temperature,...)

Private Member Functions

- void **addParamIfNotPresent** (const QString &sXiApiParamName, CxCameraParam::ExType eType)

Private Attributes

- QMap< QString, CxCameraParam > m_mapParams

  Map of all supported parameters by this camera, keys are their XiApiNames.
- QStringList m_lstParamsOrder

  Parameter names in the order they appeared in xml.
- XICORE_HANDLE m_hCamera

  Handle to camera.
- SxCameraInfo m_aCamInfo

  Basic camera informations.
- CxHsiCameraParams m_aHsiParams

Hyperspectral information.

- QSize m_sizeSensor

  Physical resolution of the sensor.
- QSize m_sizeDownsampledRes

  Remember the maximum resolution we can get at the current downsampling mode. It will enalbe us reset ROI in one go.
- QRectF m_rcLastROI

  Save ROI relatively to DownsampledRes as float vals (i.e. 1.0=full width)
- QStringList m_lstParamsToCommit

  Parameter xiApiNames in current commit batch.
- CxHsiCorrectionImage m_picWhite

  Exposure time T_ref.
- CxHsiCorrectionImage m_picDark

  Exposure time T_ref.
- CxHsiCorrectionImage m_picDarkBias

  Exposure time same as the current live image being corrected.
- ExHsiCorrection **m_eHsiCorrection**
- QMutex **m_lock**

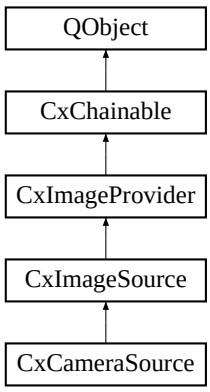
The documentation for this class was generated from the following file:

- CameraParameters.h

4.7 CxCameraSource Class Reference

The implementation of image source for Ximea cameras.

Inheritance diagram for CxCameraSource:



Public Member Functions

- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual CxChainable * clone ()

  Creates a deep copy of this object.

- virtual bool saveSettings (QDomDocument &aDoc, QDomElement &aElm) const

  The method saves the settings of this object to the DOM element.
- virtual bool loadSettings (const QDomElement &aDom)

  Loads settings from DOM.
- virtual void run ()

  This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).
- virtual bool currentOutputImageInfo (SxPicBufInfo &picInfo, CxImageMetadata *pMetadata=NULL)

  Gets the current output image info (format) provided by this object.
- CxCameraParameters * **parameters** ()
- XICORE_HANDLE **cameraHandle** () const
- void **invalidateCameraHandle** ()
- double cameraSourceFrameRate ()

  The current receiving frame rate of the camera source (it is not the same as camera frame rate!).

Protected Member Functions

- virtual int recommendedAdditionalBuffersInMemoryPool () const

  Recommended additional number of buffers that should be present in memory pool, but not reserved for this chainable only. Used in CameraSource to add more then minimum number of frames to pool.
- virtual void setupStart ()

  Sets up the object before the object is started.
- virtual void setupStop ()

  Sets up the object after the object is stopped.

Private Slots

- void onImageFormatChanged (XICORE_HANDLE hCamera)

  Connected to camera manager.

Private Member Functions

- CxCameraSource (XICORE_HANDLE hCamera)

  The constructor is private. Call the CxChainManager::createCameraSource() for new instance of the class.
- int bufferPolicy ()

  Gets the current buffer policy.
- bool **createFakeTransportPicture** (SxPicBuf *pPic, const SxPicBufInfo *pPicFormat, void *pSrcXiImg)
- void **checkCameraSkippingFrames** (QTime &timer, int &iApiCounter, int &iTransportCounter)
- bool **isUnpackingInMyChain** () const

Private Attributes

- XICORE_HANDLE m_hCamera
- qint32 m_iFrameNo
- QQueue< int > m_queLastFrameTimes
- QTime m_timerLastFrames
- SxPicBufInfo m_aCurrentOutputFormat
- bool m_bCurrentOutputFormatValid
- bool m_bFakeTransportFormat
- int m_iPayloadSize

  The payload size of the current image.

Friends

- class **CxChainManager**

Additional Inherited Members

4.7.1 Detailed Description

The implementation of image source for Ximea cameras.

4.7.2 Member Function Documentation

4.7.2.1 virtual bool CxCameraSource::currentOutputImageInfo (SxPicBufInfo & *picInfo,* CxImageMetadata * *pMetadata* = NULL) [virtual]

Gets the current output image info (format) provided by this object.

Remarks

The method should call getCurrentOutputImageInfo() on its precedessors and calculate the output image info provided by this object.

Reimplemented from CxImageProvider.

4.7.2.2 virtual void CxCameraSource::run ( ) [virtual]

This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).

See also

CxChainable

Reimplemented from CxChainable.

4.7.2.3 virtual bool CxCameraSource::saveSettings (QDomDocument & *aDoc,* QDomElement & *aElm*) const [virtual]

The method saves the settings of this object to the DOM element.

Remarks

The element aElm is already named by a caller, do not change the name (ie. do not call the aElm.setTag↩Name()). You can add attributes and child elements.

Reimplemented from CxChainable.

4.7.2.4 virtual void CxCameraSource::setupStart ( ) [protected], [virtual]

Sets up the object before the object is started.

Remarks

The method is internally used by the start() method immediately before the run() method is invoked. For example, the CxCameraSource sets up internal frame counter to zero and starts camera acquisition. The default implementation does nothing.

Reimplemented from CxChainable.

4.7.2.5 virtual void CxCameraSource::setupStop ( ) [protected], [virtual]

Sets up the object after the object is stopped.

Remarks

The method is internally used by the stop() method immediatelly after the run() method is finished. For example, the CxCameraSource stops camera acquisition here. The default implementation does nothing.

Reimplemented from CxChainable.

4.7.3 Member Data Documentation

4.7.3.1 SxPicBufInfo CxCameraSource::m_aCurrentOutputFormat [private]

Cached current output format.

4.7.3.2 bool CxCameraSource::m_bCurrentOutputFormatValid [private]

Whether the cached output format is valid.

4.7.3.3 bool CxCameraSource::m_bFakeTransportFormat [private]

Camera is in Transport data format, generate and send fake images

4.7.3.4 XICORE_HANDLE CxCameraSource::m_hCamera [private]

Handle to camera.

4.7.3.5 qint32 CxCameraSource::m_iFrameNo [private]

Internal frame counter.

4.7.3.6 QQueue<int> CxCameraSource::m_queLastFrameTimes [private]

Acquisition times of the last frames.

4.7.3.7 QTime CxCameraSource::m_timerLastFrames [private]

Measuring of the frame's acquisition times.

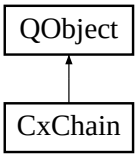
The documentation for this class was generated from the following file:

- CameraSource.h

4.8 CxChain Class Reference

This class implements a processing chain which is together with the CxChainable class (chainable objects) the keystones of the XIMEA CamTool.

Inheritance diagram for CxChain:



Public Types

- enum **ExChainState** { **eStateStopped**, **eStateStopping**, **eStateStarted**, **eStateStarting** }
- enum **ExChainErrorState** { **eErrStateNoErr**, **eErrStateWarning**, **eErrStateError** }

Signals

- void chainStarted (CxChain ∗pChain)

  The signal is emitted when the chain was started.
- void chainStopped (CxChain ∗pChain)

  The signal is emitted when the chain was stopped.
- void aboutToDelete (CxChain ∗pSender)

  The signal is emitted in destructor of the object.
- void aboutToRemove (CxChain ∗pSender, CxChainable ∗pRemoving)

  The signal is emitted from CxChain::remove() just before the chainable object is removed from the chain.
- void chainErrorState (CxChain::ExChainErrorState eState, QString sErrMsg)

  The signal is emitted from CxChain::start() if the chain start failed or some problem appeared while starting.

Public Member Functions

- ∼CxChain ()

  Destructor.
- bool start ()

  Starts processing on the chain.
- void stop ()

  Stops the chain. Each chainable stops its task immediately, all unprocessed data will be lost. The method leaves its execution when all chainables are stopped (it's a blocking operation).
- bool close ()

  Closes (but not deletes) the chain. It deletes all chainables and views in the chain.
- bool started ()

  Returns true if the chain is running.
- bool stopped ()

  Returns true if the chain is stopped.
- ExChainState state ()

  Returns a state of the chain.
- QString fullChainTitle ()

  Get the chain's full title.
- QString chainTitle (CxChainable ∗pChainable)

Gets the chain's title for the given chainable.

- QSet< CxChainable * > allChainables ()

  Gets a set of all chainables in the chain. It does not matter if the chain contains branches.

- QSet< CxImageSource * > imageSources ()

  Gets a set of all image sources in the chain.

- QSet< CxChainable * > consumers ()

  Gets a set of all data consumers in the chain.

- QSet< QObject * > nonChainableListeners ()

  Gets a set of all objects listening this chain.

- QSet< CxViewImageReceiver * > viewReceivers ()

  Gets all view receivers in the chain.

- QSet< XICORE_HANDLE > cameras ()

  Gets all cameras associated with this chain. Internally it searches all image sources which are the CxCameraSource.

Private Member Functions

- CxChain ()

  Constructor.

- bool insertBefore (CxChainable *pChainable, CxChainable *pBefore)

  Inserts new element before the specified element.

- bool branchAfter (CxChainable *pChainable, CxChainable *pAfter)

  Inserts new element after the specified element as a new branch.

- bool insertAfter (CxChainable *pChainable, CxChainable *pAfter)

  Insert new element after the specified element.

- bool remove (CxChainable *pChainable)

  The method removes given `pChainable` from this chain.

- bool connectNonChainable (CxChainable *pChainable, QObject *pListener, const char *onDataAvailable↩ Slot, const char *onFreePoolBuffersSlot)

  Connects an arbitrary object `pListener` to the `pChainable`.

- bool disconnectNonChainable (CxChainable *pChainable, QObject *pListener, const char *onData↩ AvailableSlot, const char *onFreePoolBuffersSlot)

  Disconnects an arbitrary object `pListener` from the `pChainable` which was connected using the connect↩ Chainable() method.

- CxChain * duplicateChainWithViews (CxChainable *pRoot, CxChainable *pNewRoot, bool bIncludeRoot)

  Duplicates the chain starting from `pRoot`. All views (CxMdiView) will be duplicated too.

- CxChain * duplicateBranch (CxChainable *pRoot, CxChainable *pLast, CxChainable *pNewRoot, bool b↩ IncludeRoot)

  Duplicates the branch of the starting from `pRoot` to `pLast`. If the `pLast` is an instance of CxViewImageReceiver then the eventual image view attached to the receiver will be duplitated as well.

- bool addClonesWithViews (CxChain *pChain, CxChainable *pChainable, CxChainable *pPrevClone, bool bAddChainableClone, bool bInsert)

  Internal method used by duplicateChainWithViews().

- CxMdiView * receiverView (CxViewImageReceiver *pChainable)

  Returns a view connected the the chainable.

- void deleteAll (CxChainable *pChainable)

  Deletes the `pChainable` and all its successors.

- QSet< CxChainable * > leaves (CxChainable *pRoot=NULL)

  Returns a set leaves (chainables a the end of each branche).

- void removeBranch (CxChainable *pLeaf)

  Removes a branch given by its leaf.

- bool saveSettings (QDomDocument &aXml, QDomElement &aRoot)

Exports the chain to DOM document.

- bool loadSettings (const QDomElement &aDom, QString *pErrorMsg=NULL)

  Loads the chain from DOM.
- bool saveChainableSettings (const CxChainable *pChainable, QDomDocument &aXml, QDomElement &aDom)

  Returns UniqueID from metadata. If it is not present then it returns class name.
- CxChainable * loadChainable (const QDomElement &aElm, CxChainable *pAfter)

  Recursive method that creates and loads a chainable and all successors from DOM.
- QSet< CxChainable * > disconnectSuccessors (CxChainable *pChainable)

  All chainables in the disconnected branches will be removed from chain (their chain will be set to NULL). For reconnecting use connectBranch()
- void connectBranch (CxChainable *pBranch, CxChainable *pRoot)

  Connect the `pBranch` to the `pRoot`. It is opposite method to disconnectSuccessors()
- void **deleteSuccessors** (CxChainable *pChainable)
- void **deleteViewReceiver** (CxViewImageReceiver *pReceiver)
- void viewReceivers (CxChainable *pChainable, QSet< CxViewImageReceiver * > &setReceivers)

  Recursive method that fills the `setReceivers` by all view receivers afther the `pChainable`.
- CxCameraSource * **getExistingNotOpenedCamera** (const QString &sCamSerial, const QString &sCamName, const QString &sCamType, QString *pErrorMsg)

Static Private Member Functions

- static bool **isImageSource** (const QString &sClass)

Private Attributes

- QMutex m_lock
- CxChainable * m_pRoot
- ExChainState m_eState

Friends

- class **CxChainManager**

4.8.1 Detailed Description

This class implements a processing chain which is together with the CxChainable class (chainable objects) the keystones of the XIMEA CamTool.

4.8.2 Chains and chainable objects

Processing chain (a pipeline) defines a data flow and processing of the data.

Note

Currently the chains are implemented as a tree with single root (data source).

In the XIMEA CamTool the chain consists of these chainables:

- Data source - currently implemented by CxImageSource.
- Processing elements - mostly implemented by CxImageProvider.

- Data consumers - a chainables where the chain ends i.e. consumers are leafs in the chain tree. The xiCore implements these consumers: CxViewImageReceiver, CxCircularBuffer, CxVideoToFile.

The data flow and interaction between chainables is shown on the image below. The chain on the image consists of:

- `CameraSource` (CxCameraSource) which is the implementation of the CxImageSource.
- RAW coloring (CxBayerPseudoColorsCnbl) - the implementation of the CxImageProvider.
- `VideoToFile` (CxVideoToFile) - the implementation of CxChainable as a data consumer. The `VideoToFile` has pointer to `ImageSaver` (IxImageDataSaver).

The data that flows through the chain must inherit the IxChainData interface. Currently the CxImageData and CxViewImageData classes based on that interface are implemented.

Any chain can consist of an arbitrary number of branches. Each branch gets its own copy of the data coming from data source. The copy of data is created automatically in the CxChainable::data() method.

The CxChain class implements several methods for adding/removing the CxChainable to/from the chain. These methods are private. Chains are managed by CxChainManager which is a friend class of the CxChain.

4.8.2.1 Connecting non-chainble object to the chain

The current implementation of chains allows to connect an arbitrary object to any chainable elemement. This object then receives its own copy of chain data. The arbitrary object can be connected to the chainable when it satisfies these conditions:

- It must be a descendant of the QObject.
- It must implement a slot to be able to listen the CxChainable::dataAvailable() signal.
- It must implement a slot to be able to listen the CxMemoryManager::freePoolBuffers() signal.

The object can be connected to the chainable in the chain by CxChainManager::connectNonChainable() method. When the object is connected to the chain then it behaves similar to CxChainable. It means that it recevies the CxChainable::dataAvailable() signal and can call CxChainable::data() or CxChainable::discardData() etc.

The mechanism of connected non-chainable objects is used by views (CxMdiView) in the GUI frontend. So that the chain architecture allows to connect any number of views to chainables in the chain and it is guaranteed that each of the connected objects (views) receives its own copy of the data.

4.8.3 Member Function Documentation

4.8.3.1 bool CxChain::branchAfter (CxChainable * *pChainable,* CxChainable * *pAfter*) [private]

Inserts new element after the specified element as a new branch.

Remarks

If the `pAfter` has successors then the `pChainable` will be added as a new branch. To be able to insert after the element, these conditions must be satisfied:

- both of the parameters must be at the state of CxChainable::eStopped,
- the current chain of the `pChainable` must be NULL (chain()),
- the current chain of the `pAfter` must be this chain,
- the `pAfter` must not be a data consumer (the CxChainable::isDataConsumer() must return true,
- if this chain is empty then the `pAfter` must be NULL.
- if the `pAfter` is not NULL then the `pChainable` must not be CxImageSource.

4.8.3.2 bool CxChain::connectNonChainable (CxChainable * *pChainable,* QObject * *pListener,* const char * *onDataAvailableSlot,* const char * *onFreePoolBuffersSlot*) [private]

Connects an arbitrary object `pListener` to the `pChainable`.

Remarks

the `onDataAvailableSlot` and `onFreePoolBuffersSlot` must be generated be the SLOT macro.

See also

disconnectNonChainable()

4.8.3.3 bool CxChain::disconnectNonChainable (CxChainable * *pChainable,* QObject * *pListener,* const char * *onDataAvailableSlot,* const char * *onFreePoolBuffersSlot*) [private]

Disconnects an arbitrary object `pListener` from the `pChainable` which was connected using the connect↩ Chainable() method.

Remarks

the `onDataAvailableSlot` and `onFreePoolBuffersSlot` must be generated be the SLOT macro.

4.8.3.4 CxChain* CxChain::duplicateBranch (CxChainable * *pRoot,* CxChainable * *pLast,* CxChainable * *pNewRoot,* bool *bIncludeRoot*) [private]

Duplicates the branch of the starting from `pRoot` to `pLast`. If the `pLast` is an instance of CxViewImageReceiver then the eventual image view attached to the receiver will be duplitated as well.

Remarks

All the duplicated members of the chain will be appended to the `pNewRoot`. If the `bIncludeRoot` is true, then the duplicated `pRoot` will be included in the duplicated chain as well.

4.8.3.5 CxChain* CxChain::duplicateChainWithViews (CxChainable * *pRoot,* CxChainable * *pNewRoot,* bool *bIncludeRoot*) [private]

Duplicates the chain starting from `pRoot`. All views (CxMdiView) will be duplicated too.

Remarks

All the duplicated members of the chain will be appended to the `pNewRoot`. If the `bIncludeRoot` is true, then the duplicated `pRoot` will be included in the duplicated chain as well.

4.8.3.6 bool CxChain::insertAfter (CxChainable * *pChainable,* CxChainable * *pAfter*) [private]

Insert new element after the specified element.

Remarks

The current sucessors of the `pAfter` will be reconnected to the `pChainable` and the `pChainable` becames the new successor of the `pAfter`.

4.8.3.7 bool CxChain::insertBefore (CxChainable * *pChainable,* CxChainable * *pBefore*) [private]

Inserts new element before the specified element.

Remarks

If the `pBefore` has a precedessor, then it will be reconnected so that it became the precedessor of the `pChainable`. To be able to insert before the element, these conditions must be satisfied:

- both of the parameters must be at the state of CxChainable::eStopped,
- the current chain of the `pChainable` must be NULL (chain()),
- the current chain of the `pBefore` must be this chain,
- the `pBefore` must not be a CxImageSource,
- the `pChainable` must not be a data consumer (the CxChainable::isDataConsumer() must return false.
- if this chain is empty then the `pBefore` must be NULL.

4.8.3.8 bool CxChain::loadSettings (const QDomElement & *aDom,* QString * *pErrorMsg* = `NULL`) [private]

Loads the chain from DOM.

Remarks

If the chain is empty all the chain from DOM will be imported. If the current chain is not empty (it contains a image source) then the image source will be kept and the rest of the chain will be replaced by the chain from DOM and the image source from DOM will be excluded from import.

4.8.3.9 bool CxChain::saveChainableSettings (const CxChainable * *pChainable,* QDomDocument & *aXml,* QDomElement & *aDom*) [private]

Returns UniqueID from metadata. If it is not present then it returns class name.

Recursive method that saves a single chainable settings and all its successors.

4.8.3.10 bool CxChain::start ( )

Starts processing on the chain.

Returns

The method return false if the chain is locked or already started.

4.8.4 Member Data Documentation

4.8.4.1 ExChainState CxChain::m_eState [private]

State of this chain.

4.8.4.2 QMutex CxChain::m_lock [private]

Lock of this object

4.8.4.3 CxChainable* CxChain::m_pRoot [private]

The root of the chain

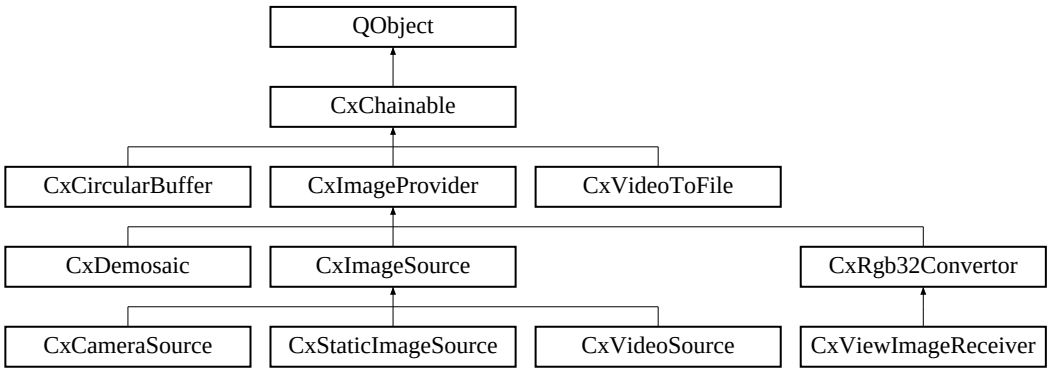
The documentation for this class was generated from the following file:

- Chain.h

4.9 CxChainable Class Reference

A base class of all chainable objects.

Inheritance diagram for CxChainable:



Public Types

- enum ExChainableState { **eUndefined** = 0, **eStopped**, **eRunning** }

  Value representing a state of the CxChainable object.
- enum ExDataSourceType { eSourceTypeNone = 0, eSourceTypeCamera, eSourceTypeStaticImage, eSourceTypeVideo }

  Values representing an data source type.
- enum ExChainableFlag {
  eHasOwnMemoryPool = 0x1, eRunIsInfiniteLoop = 0x2, eSupportsFpsRegulation = 0x4, eCanChangeEnableState = 0x8,
  eTakeLatestImage = 0x10 }

  Flags of the chainable object.

Signals

- void dataAvailable (CxChainable *pSender, int iFrameNo)

  The signal is emitted when this objects has data available for next chainable objects.
- void errorStateChanged (CxChainable *pSender, bool bError)

  The signal is emitted when some error appears or when an error is cleared. It is emitted by method setErrorMessage();.
- void aboutToDelete (CxChainable *pSender)

  The signal is emitted from the destructor of this object.
- void startRun ()

  The signal is internally connected to the run() slot.

Public Member Functions

- **CxChainable** (ExChainableFlags eFlags)
- virtual QString title () const =0

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual bool acceptsDataFrom (CxChainable *pPrecedessor)=0

  Query if this object can accept data from the given chainable object.
- virtual int buffersCountInMemoryPool () const =0

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual CxChainable * clone ()=0

  Creates a deep copy of this object.
- virtual bool isDataConsumer () const

  Returns true if the object is data consumer, i.e. the object has no successors and all the received data are "consumed" here e.g. they are displayed in image view, saved to disk etc. In other words this object is the a leaf in the chain.
- virtual ExChainableState **state** ()
- virtual IxChainData * data (QObject *pReceiver, qint32 iFrameNo)

  Gets the pointer to data processed by this object.
- virtual void discardData (QObject *pReceiver, qint32 iFrameNo)

  This method discards data for the given receiver.
- virtual IxChainData * latestData (QObject *pReceiver)

  Gets the latest data from the output queue. It is called automatically if ExChainableFlag::eTakeLatestImage is set.
- virtual void discardLatestData (QObject *pReceiver)

  Discards the latest data for the given receiver. It is called automatically if ExChainableFlag::eTakeLatestImage is set.
- virtual QWidget * configurationWidget ()

  Displays the object's configuration dialog.
- virtual bool hasConfigurationWidget () const

  Returns true when the the object has configuration widget (ie. the configurationWidget() returns not NULL).
- virtual bool saveSettings (QDomDocument &aDoc, QDomElement &aElm) const

  The method saves the settings of this object to the DOM element.
- virtual bool loadSettings (const QDomElement &aDom)

  Loads settings from DOM.
- virtual bool hasHelp () const

  Returns true if the object has help. The default is false.
- virtual QString help (QString &sImagesPath)

  Returns a string with help.
- virtual void setMaxFps (double dFps)

  Sets max. FPS (when zero or negative, then the FPS will not be regulated)
- virtual double maxFps () const

  Returns max. FPS (zero means that the max FPS is not set).
- const QSet< CxChainable * > & precedessors ()

  Gets the set of precedessors of this objects in the chain.
- const QSet< CxChainable * > & successors () const

  Gets the set of successors of this objects in the chain.
- CxChain * chain () const

  Gets a chain in which this object is contained.
- bool supportsFpsRegulation ()

  Whether this object supports FPS regulation.
- QSet< QObject * > nonChainableListeners ()

  Returns a set of all non-CxChainable objects connected to this object.
- TxMemPoolHandle memoryPool ()

Returns a handle to the object's memory pool.

- bool enabled () const

  Whether the object is enabled.
- void setEnabled (bool bEnabled)

  Can be applied when CxChainable::eCanChangeEnableState flag is set. The chain of this object must be stopped.
- ExChainableFlags **flags** () const
- QList< ExDataSourceType > chainDataSourceTypes () const

  Gets the type of data sources of this chainable object.
- QList< CxChainable * > dataSources ()

  Gets list of data sources of this chainable object.
- QString errorMessage ()

  Error message of this object.

Protected Slots

- virtual void onFreePoolBuffers (TxMemPoolHandle hMemoryPool)

  Releases all buffers allocated from the given memory pool.
- virtual void onDataAvailable (CxChainable *pSender, int iFrameNo)

  Reaction on the dataAvailable() emitted by precedessors of this object. The slot is connected automatically by the addSuccessor().
- virtual void run ()

  This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).
- virtual void onChainStarted (CxChain *pChain)

  Handle the CxChain::chainStarted(). The default implementation does nothing.
- virtual void onChainStopped (CxChain *pChain)

  Handle the CxChain::chainStarted(). The default implementation does nothing.

Protected Member Functions

- virtual bool init (QString *pErrMsg=NULL)=0

  Initializes the object. When calling this method, the object MUST be stopped.
- virtual bool **initialized** ()=0
- virtual void setupStart ()

  Sets up the object before the object is started.
- virtual void setupStop ()

  Sets up the object after the object is stopped.
- bool isMemoryPoolNeeded ()

  Returns true if this object requires memory pool. The default implementation just tests the m_eFlags, but for example the camera source should take into account the actual buffer policy of the camera.
- virtual int recommendedAdditionalBuffersInMemoryPool () const

  Recommended additional number of buffers that should be present in memory pool, but not reserved for this chainable only. Used in CameraSource to add more then minimum number of frames to pool.
- virtual IxChainData * processData (IxChainData *pReceivedData)

  The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.
- virtual bool addSuccessor (CxChainable *pSuccessor)

  Adds new successor to this object and connects the dataAvailable() signal to the successor's onDataAvailable() slot. It is internally used by CxChain class.
- virtual bool addPrecedessor (CxChainable *pPrecedessor)

Adds new precedessor to this object. It is internally used by CxChain class.

- virtual bool removeSuccessor (CxChainable *pSuccessor)

  Removes the successor from this object and discards the connection of the dataAvailable() signal from the successor's onDataAvailable() slot. It is internally used by CxChain class.
- virtual bool removePrecedessor (CxChainable *pPrecedessor)

  Removes the precedessor from this object. It is internally used by CxChain class.
- virtual bool clearSuccesors ()

  Removes and disconnect all the successors of this object. It internally calls the removeSuccessor().
- virtual bool clearPrecedessors ()

  Removes all the precedessors of this object. It internally calls the removePrecedessor().
- virtual bool addToOutputData (IxChainData *pData)

  Adds the given data to output data and emits the dataAvailable() signal if necessary.
- void deleteOldestOutputDataNoLock ()

  Deletes the output data. The method is not thread safe, it is neccassary to lock the m_lockOutputData!
- void setErrorMessage (const QString &sErrMsg)

  Set the error message and emits errorStateChanged() signal.
- bool start (CxChain::ExChainErrorState *pErrState=NULL, QString *pErrMsg=NULL)

  Starts this object and sets the correct state of the object.
- bool stop ()

  Stops running and resets this object (release appropriate buffers etc.). The method stops receiving input data and processing incoming data.
- void waitForStarted ()

  The method waits until the run() method is started.
- void waitForStopped ()

  The method waits until the run() method is finished.
- void freeOutputData ()

  The method frees all output data CxChainable::m_mapOutputData.
- void connect (QObject *pReceiver, const char *method)

  This method registers an arbitrary non-chainable receiver to receive a data provided by this object.
- void disconnect (QObject *pReceiver, const char *method)

  *Discards the connection between the `pReceiver` and this object created by connect(const QObject *pReceiver, const char *method) method.*
- int objectBuffersInMemoryPool ()

  *Returns the number of buffers required by **this** object to have in memory pool.*

Protected Attributes

- QSet< CxChainable * > m_setPrecedessors
- QSet< CxChainable * > m_setSuccessors
- int m_iMaxOutputDataCount
- ExChainableFlags m_eFlags
- double m_dMaxFps
- QQueue< int > m_queFpsCounter
- QTime m_timerFps
- ExChainableState m_eState
- ExChainableState m_eReqState
- QMutex m_lock
- QMutex m_lockOutputData
- QMap< int, IxChainData * > m_mapOutputData
- QMap< QObject *, QSet< int > > m_mapObjectOutputData
- XICORE_HANDLE m_hMemoryPool
- QSemaphore m_semStartStop
- bool m_bDataConsumer

Private Member Functions

- bool setChain (CxChain ∗pChain, bool bSetSuccessors=false)

  Sets the 'parent' chain to this object.
- double calcFps ()

  Calculates current FPS (if FPS regulation is not supported the it returns 0)
- bool **addToOutputDataNoLock** (IxChainData ∗pData)
- IxChainData ∗ **dataNoLock** (QObject ∗pReceiver, qint32 iFrameNo)
- void **discardDataNoLock** (QObject ∗pReceiver, qint32 iFrameNo)
- QThread ∗ myThread () const

  Gets the internal thread of this object.

Private Attributes

- CxChain ∗ m_pMyChain
- QThread ∗ m_pThread
- QString m_sErrorMessage
- bool m_bLoosingFrames
- int m_iFirstLostFrame
- bool m_bEnabled

Friends

- class **CxChain**
- class **CxChainManager**

4.9.1 Detailed Description

A base class of all chainable objects.

Remarks

The CxChainable class together with the CxChain is the key stone of the xiCore architecture. It is recommended to read the CxChain documentation before you start to study this class.

4.9.2 Implementation of chainable object

The CxChainable class is an abstract class that implements:

- starting and stopping the object (start(), stop());
- receiving a data from its precedessors in the chain (onDataAvailable(), data(), discardData());
- notifying its successors when it has a processed data ready (dataAvailable()).

This class generally can process any kind of data (IxChainData). When implementing a custom CxChainable class then in most cases the CxImageProvider is the class you should inherit from. Currently the CxImageProvider is the only class that can be registered by the application and thus that can be added to a processing chain by the chain editor (see plugins).

Memory management

Any chainable object can use its own memory pool (CxMemoryManager). This is specified by the CxChainable::eHasOwnMemoryPool flag passed to the constructor. The memory pool should be initialized in the pure virtual method init(). This method is implemented by the CxImageProvider so that when you

inherit from it you do not need to implement this method as well as the initialized() method.
Each chainable object in the chain has its onFreePoolBuffers() slot connected to the CxMemoryManager::freePoolBuffers(). When the signal is emitted then the chainable object **allways must** immediatelly release all buffers allocated from the given memory pool otherwise the application crashes on assertation!

Data ownership

There are simple rules describing the data ownership:

1. If the object created (allocated) a data (IxChainData) then it is the owner of the data until some of its successors take the data by calling data() method.
2. If the object takes the data by calling data() method on its precedessor then it become the owner of the data until some of its successors take the data by calling data() method.

When the object is the owner of the data then there are two scenarios how the data can be managed:

1. The chainable object processes the data directly. It means that the format and the size of the data is unchanged and the data are processed in-place. After the processing the data are passed to the object's successors. In this case the chainable object cannot delete the data until the successors pick up the data by calling the data() method.
2. The chainable object changes the data size, format etc. so that it needs to create (alloc) new data which are passed to the object's successors. In this case the object must delete the original (received) data after the processing finished.

Sometimes it can happen that the chainable object cannot accept a data from its precedessor. It means that its precedessor notified the object by dataAvailable() signal but the object cannot process them i.e. it cannot call the data() method. In this case the object should call the discardData() on the precedessor instead of the data().

4.9.3 Member Enumeration Documentation

4.9.3.1 enum CxChainable::ExChainableFlag

Flags of the chainable object.

Enumerator

eHasOwnMemoryPool Object uses own memory pool.

eRunIsInfiniteLoop The run() method is implemented as an infinite loop.

eSupportsFpsRegulation The object allows to regulate a framerate by discarding input data.

eCanChangeEnableState The object can be disabled so that it will not process a data and the data flow is bypassed

eTakeLatestImage The object always gets the latest image from its precedessor's output queue.

4.9.3.2 enum CxChainable::ExDataSourceType

Values representing an data source type.

Enumerator

eSourceTypeNone The object is not a data source.

eSourceTypeCamera Data come from camera (CxCameraSource).

eSourceTypeStaticImage Data come from static image (CxStaticImageSource).

eSourceTypeVideo Data come from video (CxVideoSource).

4.9.4 Member Function Documentation

4.9.4.1 QList<ExDataSourceType> CxChainable::chainDataSourceTypes ( ) const

Gets the type of data sources of this chainable object.

Remarks

The current implementation of the CxChain supports only one data source so that the method return a list containing at most one element.

See also

dataSources()

4.9.4.2 virtual QWidget* CxChainable::configurationWidget ( ) [virtual]

Displays the object's configuration dialog.

Remarks

The configuration widget should not contain the Ok and Cancel button as they will be added automatically by the application.

Warning

When this chainable object changes the image format, don't forget to stop the current chain before applying new value from the dialog!
This method MUST be called from GUI (main) thread only!!

Returns

The default value is NULL. When overriden then it should return the pointer to the configuration widget.

Reimplemented in CxViewImageReceiver.

4.9.4.3 void CxChainable::connect (QObject * *pReceiver,* const char * *method*) [protected]

This method registers an arbitrary non-chainable receiver to receive a data provided by this object.

Remarks

The method creates connection between its dataAvailable() signal and the receiver's slot given by `method` parameter.

Warning

Allways use the SLOT macro to specify the method name given by `method` parameter.
Each object which wants to receive the data must be connected using this method otherwise the behaviour is undefined.

See also

disconnect(const QObject *pReceiver, const char *method)

4.9.4.4 virtual IxChainData* CxChainable::data (QObject * *pReceiver,* qint32 *iFrameNo*) [virtual]

Gets the pointer to data processed by this object.

See also

discardData()
CxChainable

Reimplemented in CxStaticImageSource.

4.9.4.5 void CxChainable::dataAvailable (CxChainable * *pSender,* int *iFrameNo*) [signal]

The signal is emitted when this objects has data available for next chainable objects.

See also

CxChainable

4.9.4.6 QList<CxChainable*> CxChainable::dataSources ( )

Gets list of data sources of this chainable object.

Remarks

The current implementation of the CxChain supports only one data source so that the method return a list containing at most one element.

See also

chainDataSourceTypes()

4.9.4.7 virtual void CxChainable::discardData (QObject * *pReceiver,* qint32 *iFrameNo*) [virtual]

This method discards data for the given receiver.

See also

data()
CxChainable

4.9.4.8 void CxChainable::disconnect (QObject * *pReceiver,* const char * *method*) [protected]

Discards the connection between the `pReceiver` and this object created by connect(const QObject *pReceiver, const char *method) method.

Warning

Allways use the SLOT macro to specify the method name given by the `method` parameter.

See also

connect(const QObject *pReceiver, const char *method)

4.9.4.9 bool CxChainable::enabled ( ) const

Whether the object is enabled.

See also

CxChainable::eCanChangeEnableState
setEnabled()

4.9.4.10 QString CxChainable::errorMessage ( )

Error message of this object.

See also

setErrorMessage()

4.9.4.11 virtual QString CxChainable::help (QString & *sImagesPath*) [virtual]

Returns a string with help.

Remarks

The text can be plain text or HTML. If the HTML help contains images then `sImagesPath` should contain an path to the images. The images path must be either absolute or relative to the application path (qApp->applicationDirPath()) and it also can be a prefix to embedded resource images (for example ":/images").

Reimplemented in CxDemosaic.

4.9.4.12 virtual bool CxChainable::init (QString * *pErrMsg* = NULL) [protected], [pure virtual]

Initializes the object. When calling this method, the object MUST be stopped.

Remarks

Methods init() and initialized() are internally used by the start() method. It should initialize the chainable object so that it is ready to start (start()) . If this object is the CxImageProvider then the method initializes internal memory pool.

Implemented in CxImageProvider, CxVideoToFile, and CxCircularBuffer.

4.9.4.13 TxMemPoolHandle CxChainable::memoryPool ( )

Returns a handle to the object's memory pool.

Remarks

If the object has no internal memory pool then the method returns NULL.

4.9.4.14 int CxChainable::objectBuffersInMemoryPool ( ) [protected]

Returns the number of buffers required by **this** object to have in memory pool.

Remarks

The method calls internally the buffersCountInMemoryPool() and add a number of listeners of this object.

4.9.4.15 virtual void CxChainable::onFreePoolBuffers (TxMemPoolHandle *hMemoryPool*) [protected], [virtual], [slot]

Releases all buffers allocated from the given memory pool.

Remarks

This slot is always connected to CxMemoryManager::freePoolBuffers() signal. The chainable object **MUST** release all buffers allocated from the given memory pool. If the buffers will not be released then the application's behaviour is undefined (the app. most probably crashes or freezes). The default implementation of the method checks the internal output data (CxChainable::m_mapOutputData) and releases all the data from the given memory pool.

4.9.4.16 virtual IxChainData* CxChainable::processData (IxChainData * *pReceivedData*) [protected], [virtual]

The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

Remarks

If the implementation creates (allocates) a new data here then the original (received) data should be deleted using the delete operator. The default implementation of this method returns a pointer to the received data without any processing.

Warning

If the method returns NULL, then the `pReceivedData` will be deleted automatically!

Reimplemented in CxViewImageReceiver, CxDemosaic, and CxRgb32Convertor.

4.9.4.17 virtual void CxChainable::run ( ) [inline], [protected], [virtual], [slot]

This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).

See also

CxChainable

Reimplemented in CxCameraSource, CxStaticImageSource, and CxVideoSource.

4.9.4.18 virtual bool CxChainable::saveSettings (QDomDocument & *aDoc,* QDomElement & *aElm*) const [virtual]

The method saves the settings of this object to the DOM element.

Remarks

The element `aElm` is already named by a caller, do not change the name (ie. do not call the aElm.setTagName()). You can add attributes and child elements.

Reimplemented in CxCameraSource, CxStaticImageSource, and CxVideoSource.

4.9.4.19 bool CxChainable::setChain (CxChain * *pChain*, bool *bSetSuccessors* = false) [private]

Sets the 'parent' chain to this object.

Remarks

The method succeedes when this object is not assigned to other chain. If the pChain is NULL, then the method succeedes when the current object's chain does not have this chainable object connected i.e. the CxChain::isConnectedInChain() method returns false.

4.9.4.20 virtual void CxChainable::setupStart ( ) [protected], [virtual]

Sets up the object before the object is started.

Remarks

The method is internally used by the start() method immediately before the run() method is invoked. For example, the CxCameraSource sets up internal frame counter to zero and starts camera acquisition. The default implementation does nothing.

Reimplemented in CxCameraSource, and CxDemosaic.

4.9.4.21 virtual void CxChainable::setupStop ( ) [protected], [virtual]

Sets up the object after the object is stopped.

Remarks

The method is internally used by the stop() method immediatelly after the run() method is finished. For example, the CxCameraSource stops camera acquisition here. The default implementation does nothing.

Reimplemented in CxCameraSource, CxDemosaic, and CxRgb32Convertor.

4.9.4.22 bool CxChainable::start (CxChain::ExChainErrorState * *pErrState* = NULL, QString * *pErrMsg* = NULL) [protected]

Starts this object and sets the correct state of the object.

Remarks

When the object is started then it can process and accept data.

4.9.4.23 void CxChainable::startRun ( ) [signal]

The signal is internally connected to the run() slot.

Remarks

It is internally emitted in the start() method.

Warning

Do not emit or receive the signal!

4.9.4.24 bool CxChainable::stop ( ) [protected]

Stops running and resets this object (release appropriate buffers etc.). The method stops receiving input data and processing incoming data.

Remarks

When the chainable object is stopped then it must be guaranted that:

- the object cannot emit dataAvailable(),
- when the data() is invoked, then the method must return NULL,
- when the onDataAvailable() slot is invoked then it cannot call the data() on the sender. I.e. when the object is stopped it cannot provide or receive any data. The default implementation of the CxChainable satisfies the conditions above.

4.9.4.25 void CxChainable::waitForStarted ( ) [protected]

The method waits until the run() method is started.

Remarks

The method is internally used by the start() method.

Warning

Do not use it in your code!

4.9.4.26 void CxChainable::waitForStopped ( ) [protected]

The method waits until the run() method is finished.

Remarks

The method is internally used by the stop() method.

Warning

Do not use it in your code!

4.9.5 Member Data Documentation

4.9.5.1 bool CxChainable::m_bDataConsumer [protected]

Whether the object is data consumer.

4.9.5.2 bool CxChainable::m_bEnabled [private]

Enable state is valid when CxChainable::eCanChangeEnableState flag is set

4.9.5.3 bool CxChainable::m_bLoosingFrames [private]

It is internally used to signalizing the error state.

Warning

Do not touch!

4.9.5.4 double CxChainable::m_dMaxFps [protected]

Max. FPS processed by this object. (-1 for unlimited)

4.9.5.5 ExChainableFlags CxChainable::m_eFlags [protected]

Flags of this object.

4.9.5.6 ExChainableState CxChainable::m_eReqState [protected]

Required state of this object. It is used when run() is an infinite loop. Do not touch!

4.9.5.7 ExChainableState CxChainable::m_eState [protected]

State of this object.

4.9.5.8 XICORE_HANDLE CxChainable::m_hMemoryPool [protected]

Handle to memory pool.

4.9.5.9 int CxChainable::m_iFirstLostFrame [private]

It is internally used to signalizing the error state.

Warning

Do not touch!

4.9.5.10 int CxChainable::m_iMaxOutputDataCount [protected]

Max number of output data per successor.

4.9.5.11 QMutex CxChainable::m_lock [protected]

Lock of this object.

4.9.5.12 QMutex CxChainable::m_lockOutputData [protected]

Lock of the output data.

4.9.5.13 QMap<QObject*, QSet<int> > CxChainable::m_mapObjectOutputData [protected]

The map of the output frame numbers. Key ~ the receiver, value ~ set of frame numbers

4.9.5.14 QMap<int, IxChainData*> CxChainable::m_mapOutputData [protected]

The output data, key ~ the frame number, value ~ the processed data.

4.9.5.15 CxChain* CxChainable::m_pMyChain [private]

Chain which this object belongs to.

4.9.5.16 QThread* CxChainable::m_pThread [private]

The thread in which this object lives (if NULL, then the object lives in the GUI thread)

4.9.5.17 QQueue<int> CxChainable::m_queFpsCounter [protected]

Queue of times when data was received - for FPS calculation.

4.9.5.18 QSemaphore CxChainable::m_semStartStop [protected]

Semaphore signalizing that the run() started or stopped.

Warning

Do not touch!

4.9.5.19 QString CxChainable::m_sErrorMessage [private]

The error message of this object.

4.9.5.20 QSet<CxChainable*> CxChainable::m_setPrecedessors [protected]

Set of precedessors of this object in the chain.

4.9.5.21 QSet<CxChainable*> CxChainable::m_setSuccessors [protected]

Set of successors of this object in the chain.

4.9.5.22 QTime CxChainable::m_timerFps [protected]

Timer for FPS measurement

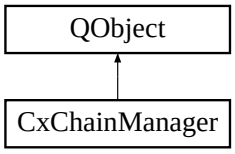
The documentation for this class was generated from the following file:

- Chainable.h

4.10 CxChainManager Class Reference

The class that creates, manages and manipulates with chains.

Inheritance diagram for CxChainManager:



Signals

- void **chainErrorState** (CxChain *pChain, CxChain::ExChainErrorState eState, QString sErrMsg)

Public Member Functions

- ~CxChainManager ()

  Destructor.
- CxChain * createChain ()

  Creates an instance of new chain.
- CxCameraSource * createCameraSource (XICORE_HANDLE hCamera)

  Creates an instance of camera source.
- CxCameraSource * cameraSource (XICORE_HANDLE hCamera)

  Gets the pointer to the existing camera source.
- void cameraDisconnected (XICORE_HANDLE hCamera)

  For internal use only. Keeps the existing cameraSource, but sets its handle to NULL.
- CxImageSource * createImageSource (IxImageDataLoader *pLoader)

  Creates an instance of an image source.
- CxStaticImageSource * createStaticImageSource (CxImageData *pImageData, const QString &sTitle)

  Creates an instance of static image source.
- bool closeChain (CxChain *pChain)

  Closes the chain.
- bool deleteChainable (CxChainable *pChainable)

  Removes and deletes the chainable from the manager. The chainable must not be connected to any chain.
- bool deleteBranches (CxChain *pChain, CxChainable *pRoot)

  Deletes all branches (including views) of the given chainable `pRoot`.
- CxChain * cameraChain (XICORE_HANDLE hCamera)

  Gets a camera's chain.
- bool insertBefore (CxChain *pChain, CxChainable *pChainable, CxChainable *pBefore)

  Inserts the `pChainable` before the `pBefore` object in the chain given by `pChain`.
- bool branchAfter (CxChain *pChain, CxChainable *pChainable, CxChainable *pAfter)

  Inserts the `pChainable` after the `pAfter` in the chain given by `pChain` as a new branch.
- bool insertAfter (CxChain *pChain, CxChainable *pChainable, CxChainable *pAfter)

  Inserts the `pChainable` after the `pAfter` in the chain given by `pChain`.
- bool remove (CxChain *&pChain, CxChainable *pChainable, bool bDeleteBranchWhenNoListener=false)

  Removes the `pChainable` from the `pChain`.
- bool connectNonChainable (CxChain *pChain, CxChainable *pChainable, QObject *pListener, const char *onDataAvailableSlot, const char *onFreePoolBuffersSlot)

  The method registers a custom object as a data receiver of the `pChainable`.
- bool disconnectNonChainable (CxChain *&pChain, CxChainable *pChainable, QObject *pListener, const char *onDataAvailableSlot, const char *onFreePoolBuffersSlot, bool bDeleteWhenNoListener=false)

  Discards the connection created by connectNonChainable()
- CxChain * duplicateChainWithViews (CxChain *pChain, CxChainable *pRoot, CxChainable *pNewRoot, bool bIncludeRoot)

  Duplicates the `pChain` starting from `pRoot`. All views (CxMdiView) will be duplicated too.
- CxChain * duplicateChainBranch (CxChain *pChain, CxChainable *pRoot, CxChainable *pLast, CxChainable *pNewRoot, bool bIncludeRoot)

  Duplicates the branch of the `pChain` starting from `pRoot` to `pLast`. If the `pLast` is an instance of CxViewImageReceiver then the eventual image view attached to the receiver will be duplitated as well.
- bool loadChainSettings (const QString &sFileName, QDomDocument &aDocChain, QString *pErrMsg=NULL)

  Loads the chain settings from the given file to a DOM document.

- bool importToExistingChain (CxChain *pChain, const QDomDocument &aDocChain)

  Replaces the existing chain by the chain imported from the given DOM document.
- CxChain * loadChain (const QDomDocument &aDocChain, QString *pErrorMsg=NULL)

  Loads and creates a chain from the DOM document.
- const QMap< XICORE_HANDLE, CxCameraSource * > cameraSources () const

  Gets all the camera sources.
- CxChainable * findChainableByUniqueID (CxChain *pChain, const QString &sUniqueID)

  Returns a chainable with given UniqueId. If it is not present in the `pChain` then it returns NULL.

Static Public Member Functions

- static CxChainManager * instance ()

  Gets the class instance.
- static bool exportChain (CxChain *pChain, QDomDocument &aXml, QDomElement &aRoot)

  Exports the chain to DOM document.
- static bool exportChain (CxChain *pChain, QString &sXml)

  Export the chain to XML string.
- static bool saveChain (CxChain *pChain, const QString &sFileName)

  Saves the chain to XML file.
- static void saveChainableSettings (CxChainable *pObject, QSettings *pSettings)

  Saved the object settings to QSettings.
- static void loadChainableSettings (CxChainable *pObject, QSettings *pSettings)

  Loads the object settings from QSettings.

Private Slots

- void **onChainErrorState** (CxChain::ExChainErrorState eState, QString sErrMsg)

Private Member Functions

- CxChainManager ()

  Private constructor (the class is singleton).
- void removeBlindBranches (CxChain *&pChain)

  Removes branches with no listeners. If the chain is empty, then it will be destroyed.
- void removeBlindBranches (CxChain *&pChain, CxChainable *pChainable)

  Removes the blind branches the `pChainable` is connected to.

Private Attributes

- QSet< CxChain * > m_setChains
- QMap< XICORE_HANDLE, CxCameraSource * > m_mapCameraSources
- QSet< CxChainable * > m_setChainables
- QMutex m_lock

4.10.1 Detailed Description

The class that creates, manages and manipulates with chains.

4.10.2 Member Function Documentation

4.10.2.1 bool CxChainManager::branchAfter (CxChain * *pChain,* CxChainable * *pChainable,* CxChainable * *pAfter*)

Inserts the `pChainable` after the `pAfter` in the chain given by `pChain` as a new branch.

Remarks

The `pAfter` must already be connected in the `pChain`.

4.10.2.2 CxCameraSource* CxChainManager::cameraSource (XICORE_HANDLE *hCamera*)

Gets the pointer to the existing camera source.

Returns

If the camera source exists then it returns pointer to it, otherwise it returns NULL.

4.10.2.3 bool CxChainManager::closeChain (CxChain * *pChain*)

Closes the chain.

Remarks

If the chain is started it will be stopped. The method removes all the chainable object from the chain and then deletes the chain.

4.10.2.4 bool CxChainManager::connectNonChainable (CxChain * *pChain,* CxChainable * *pChainable,* QObject * *pListener,* const char * *onDataAvailableSlot,* const char * *onFreePoolBuffersSlot*)

The method registers a custom object as a data receiver of the `pChainable`.

Remarks

Allways use the SLOT macro to define the `onDataAvailableSlot` and `onFreePoolBuffersSlot`. These slots will be connected to CxChainable::dataAvailable() signal and CxMemoryManager::freePoolBuffers().

See also

disconnectNonChainable()

4.10.2.5 CxCameraSource* CxChainManager::createCameraSource (XICORE_HANDLE *hCamera*)

Creates an instance of camera source.

Remarks

Each camera can have only one instance of the camera source. If the method is called more than once for the valid camera handle then it fails (it returns NULL).

4.10.2.6 CxImageSource* CxChainManager::createImageSource (IxImageDataLoader * *pLoader*)

Creates an instance of an image source.

Remarks

The function take the ownership of the `pLoader`, so that it takes care about its deletion.

4.10.2.7 CxStaticImageSource* CxChainManager::createStaticImageSource (CxImageData * *pImageData,* const QString & *sTitle*)

Creates an instance of static image source.

Remarks

If the function succedes then it takes the ownership of the `pImageData`, so that it takes care about its deletion.

4.10.2.8 CxChain* CxChainManager::duplicateChainBranch (CxChain * *pChain,* CxChainable * *pRoot,* CxChainable * *pLast,* CxChainable * *pNewRoot,* bool *bIncludeRoot*)

Duplicates the branch of the `pChain` starting from `pRoot` to `pLast`. If the `pLast` is an instance of CxViewImageReceiver then the eventual image view attached to the receiver will be duplitated as well.

Remarks

All the duplicated members of the chain will be appended to the `pNewRoot`. If the `bIncludeRoot` is true, then the duplicated `pRoot` will be included in the duplicated chain as well.

4.10.2.9 CxChain* CxChainManager::duplicateChainWithViews (CxChain * *pChain,* CxChainable * *pRoot,* CxChainable * *pNewRoot,* bool *bIncludeRoot*)

Duplicates the `pChain` starting from `pRoot`. All views (CxMdiView) will be duplicated too.

Remarks

All the duplicated members of the chain will be appended to the `pNewRoot`. If the `bIncludeRoot` is true, then the duplicated `pRoot` will be included in the duplicated chain as well.

4.10.2.10 bool CxChainManager::importToExistingChain (CxChain * *pChain,* const QDomDocument & *aDocChain*)

Replaces the existing chain by the chain imported from the given DOM document.

Remarks

. The chain being imported will be appended to the existing image source of the `pChain`. The image source from the from the DOM will be excluded.

4.10.2.11 bool CxChainManager::insertAfter (CxChain * *pChain,* CxChainable * *pChainable,* CxChainable * *pAfter*)

Inserts the `pChainable` after the `pAfter` in the chain given by `pChain`.

Remarks

The `pAfter` must already be connected in the `pChain`. All successors of the `pAfter` will be disconnected fro it and reconnected to the `pChainable`.

4.10.2.12 bool CxChainManager::insertBefore (CxChain * *pChain,* CxChainable * *pChainable,* CxChainable * *pBefore*)

Inserts the `pChainable` before the `pBefore` object in the chain given by `pChain`.

Remarks

The `pBefore` must already be connected in the `pChain`. All precedessors of the `pBefore` will be disconnected from it and reconnected to `pChainable`.

4.10.2.13 bool CxChainManager::remove (CxChain *& *pChain,* CxChainable * *pChainable,* bool *bDeleteBranchWhenNoListener* = `false`)

Removes the `pChainable` from the `pChain`.

Remarks

If the `bDeleteBranchWhenNoListener` and the branch from which the `pChainable` was removed has no listener then the branch will be removed from the chain. If the chain has only one branch then the chain will be removed and the `pChain` will be set to NULL.

4.10.3 Member Data Documentation

4.10.3.1 QMutex CxChainManager::m_lock [private]

The lock of this object.

4.10.3.2 QMap<XICORE_HANDLE, CxCameraSource*> CxChainManager::m_mapCameraSources [private]

Map of camera sources

4.10.3.3 QSet<CxChainable*> CxChainManager::m_setChainables [private]

Set of chainables.

4.10.3.4 QSet<CxChain*> CxChainManager::m_setChains [private]

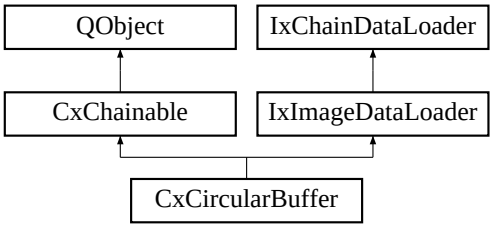
The set of chains.

The documentation for this class was generated from the following file:

- ChainManager.h

4.11 CxCircularBuffer Class Reference

Inheritance diagram for CxCircularBuffer:



Signals

- void **frameReceived** (qint32 iCirclePosition)

Public Member Functions

- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual bool acceptsDataFrom (CxChainable *pPrecedessor)

  Query if this object can accept data from the given chainable object.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual CxChainable * clone ()

  Creates a deep copy of this object.
- virtual bool **currentOutputImageInfo** (SxPicBufInfo &picInfo, CxImageMetadata *pMetadata=NULL)
- virtual bool **queryOutputImageInfo** (const SxPicBufInfo &picInfoInput, SxPicBufInfo &picInfoOutput, const CxImageMetadata *pMetadataInput, CxImageMetadata *pMetadataOutput=NULL)
- virtual qint32 sequenceSize ()

  In case the source has multiple data items, their count is returnes here.
- virtual IxChainData * loadedData (qint32 iSeqIdx=0)

  Gets the pointer to the current data (after the load() method was called).
- virtual bool load ()

  Loads data from some source.
- virtual QString inputFile ()

  If the loader load data from file then this method returns the the file's name.
- virtual void **setInputFile** (const QString &sFileName)
- virtual double loadedFps ()

  In case the source is a sequence, this should get its fps. Return 0 if uncertain.
- virtual TxImageFormatList supportedImageFormats ()

  Image formats capable of loading.
- void setCircleSize (qint32 iSize)

  Sets the number of buffers in the circular buffer.
- qint32 **circleSize** () const

Protected Slots

- virtual void **onDataAvailable** (CxChainable *pSender, int iFrameNo)

Protected Member Functions

- virtual bool init (QString *pErrMsg=NULL)

  Initializes the object. When calling this method, the object MUST be stopped.
- virtual bool **initialized** ()

Protected Attributes

- qint32 m_iCircleSize

  number of allocated pictures (cannot be all filled - when stopped in the first pass)
- qint32 m_iFramesCount

  number of frames catpured to circular buffer
- qint32 m_iLastFrame

  frame that we stored as last at this moment
- double m_dFps

  fps of the incoming images. Neeed to tell when providing the data further
- QVector< CxImageData * > m_vecCircBuffer

  this is the allocated circular buffer

Private Member Functions

- void **freeBuffers** ()

Additional Inherited Members

4.11.1 Detailed Description

Circular buffer has two modes:

1. as a CxChainable, serves as a storage for frames (primary task)
2. as a IxImageDataLoader, serves as a in-memory storage for CxVideoSource

4.11.2 Member Function Documentation

4.11.2.1 virtual bool CxCircularBuffer::init (QString * *pErrMsg* = `NULL`) `[protected], [virtual]`

Initializes the object. When calling this method, the object MUST be stopped.

Remarks

Methods init() and initialized() are internally used by the start() method. It should initialize the chainable object so that it is ready to start (start()) . If this object is the CxImageProvider then the method initializes internal memory pool.

Implements CxChainable.

4.11.2.2 virtual IxChainData* CxCircularBuffer::loadedData (qint32 *iSeqIdx* = `0`) `[virtual]`

Gets the pointer to the current data (after the load() method was called).

Remarks

The caller of this method is the owner of the returned data so that it has to take care about its deletion using the `delete` operator.

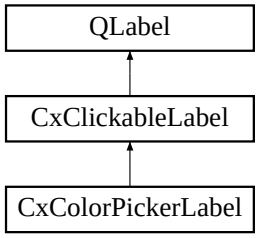
Implements IxChainDataLoader.

The documentation for this class was generated from the following file:

- CircularBuffer.h

4.12 CxClickableLabel Class Reference

Inheritance diagram for CxClickableLabel:



Signals

- void **clicked** ()

Public Member Functions

- **CxClickableLabel** (QWidget ∗parent=0)

Protected Member Functions

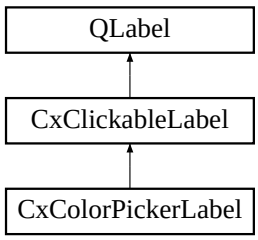
- virtual void **mouseReleaseEvent** (QMouseEvent ∗event)

The documentation for this class was generated from the following file:

- ClickableLabel.h

4.13 CxColorPickerLabel Class Reference

Inheritance diagram for CxColorPickerLabel:



Signals

- void **colorChanged** ()

Public Member Functions

- **CxColorPickerLabel** (QWidget ∗parent=0)
- unsigned **pickerColor** ()
- void **setPickerColor** (unsigned clColor)

Static Public Member Functions

- static QCursor **droppperCursor** ()

Protected Member Functions

- virtual void **paintEvent** (QPaintEvent ∗)

Protected Attributes

- unsigned **m_color**

Private Slots

- void **on_pickerClicked** ()

The documentation for this class was generated from the following file:

- ClickableLabel.h

4.14 CxLut::CxCompContrast Class Reference

Public Member Functions

- bool isReset (quint32 uiSrcBpc)

  does it applies contast?
- QString **saveToString** () const
- bool **loadFromString** (const QString &sString)

Public Attributes

- int **m_iSrcMinVal**
- int m_iSrcMaxVal

  increase contrast
- int **m_iDstMinVal**
- int m_iDstMaxVal

  decrease contrast
- double m_dGamma

  gamma during contrast

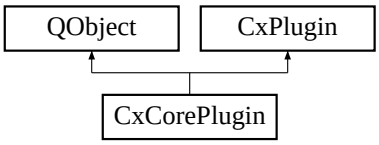
The documentation for this class was generated from the following file:

- Lut.h

4.15 CxCorePlugin Class Reference

The core plugin that registers all core classes (image loaders/savers etc.)

Inheritance diagram for CxCorePlugin:



Public Member Functions

- virtual bool init ()

  Called on application start.
- virtual QString name () const

  Plugin name displayed in the Plugin Manager view.
- virtual QString author () const

  Name of the author (company) of this plugin.
- virtual QString description () const

  Short description of the plugin, notes, warnings etc.
- virtual void version (int &major, int &minor, int &build) const

  Version of the plugin.
- virtual QString website () const

  The author's website adress.
- virtual QString copyrightNotice () const

  In case plugin needs to write some legal info to Help - About - Legal...

Static Public Member Functions

- static CxPlugin * **instance** ()

Private Attributes

- bool **m_bInitialized**

4.15.1 Detailed Description

The core plugin that registers all core classes (image loaders/savers etc.)

The documentation for this class was generated from the following file:

- CorePlugin.h

4.16 CxHsiCameraParams::CxCorrectionMatrix Class Reference

Public Attributes

- QString m_sName

  Name of the correction matrix.

- int **m_iStartNm**
- int m_iEndNm

  Filter range.
- ExCorrectionMatrixType m_eType

  Type of correction matrix.
- ExCorrectionMatrixAlgorithm m_eAlgorithm

  Algorithm used to create the matrix.
- QVector< CxVirtualBand > m_vecVirtualBands

  size is equal or lower than vecBands.count, may be present even in images (correction of spectrum). tag <spectral_correction_info>
- QVector< CxOpticalComponent > m_vecOpticalComponents

  List of optical components used in this matrix.

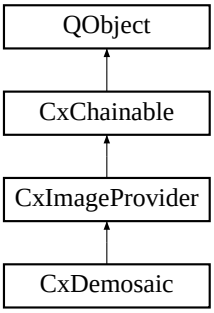
The documentation for this class was generated from the following file:

- HsiCameraParams.h

4.17 CxDemosaic Class Reference

The class performs a debayering on images in RAW format.

Inheritance diagram for CxDemosaic:



Public Member Functions

- **CxDemosaic** (const SxPicBufInfo &rOutputFormat)
- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual bool acceptsDataFrom (CxChainable *pPrecedessor)

  Query if this object can accept data from the given chainable object.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual CxChainable * clone ()

  Creates a deep copy of this object.
- virtual bool hasHelp () const

  Returns true if the object has help. The default is false.
- virtual QString help (QString &sImagesPath)

  Returns a string with help.
- virtual bool queryOutputImageInfo (const SxPicBufInfo &picInfoInput, SxPicBufInfo &picInfoOutput, const CxImageMetadata *pMetadataInput, CxImageMetadata *pMetadataOutput=NULL)

  Query the output image info (format) for given input image format.

Protected Member Functions

- virtual void setupStart ()

  Sets up the object before the object is started.
- virtual void setupStop ()

  Sets up the object after the object is stopped.
- virtual IxChainData * processData (IxChainData *pReceivedData)

  The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

Private Types

- enum **ExOfflineProcessing** { **eopNone**, **eopUnpacking**, **eopDebayering** }

Private Attributes

- bool **m_bUsePredefOutput**
- SxPicBufInfo **m_aPredefOutput**
- void * **m_pProc**
- ExOfflineProcessing **m_eProcType**

Additional Inherited Members

4.17.1 Detailed Description

The class performs a debayering on images in RAW format.

4.17.2 Member Function Documentation

4.17.2.1 virtual QString CxDemosaic::help (QString & *sImagesPath*) [virtual]

Returns a string with help.

Remarks

The text can be plain text or HTML. If the HTML help contains images then `sImagesPath` should contain an path to the images. The images path must be either absolute or relative to the application path (qApp->applicationDirPath()) and it also can be a prefix to embedded resource images (for example ":/images").

Reimplemented from CxChainable.

4.17.2.2 virtual IxChainData* CxDemosaic::processData (IxChainData * *pReceivedData*) [protected], [virtual]

The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

Remarks

If the implementation creates (allocates) a new data here then the original (received) data should be deleted using the delete operator. The default implementation of this method returns a pointer to the received data without any processing.

Warning

If the method returns NULL, then the `pReceivedData` will be deleted automatically!

Reimplemented from CxChainable.

4.17.2.3 virtual void CxDemosaic::setupStart ( ) [protected], [virtual]

Sets up the object before the object is started.

Remarks

The method is internally used by the start() method immediately before the run() method is invoked. For example, the CxCameraSource sets up internal frame counter to zero and starts camera acquisition. The default implementation does nothing.

Reimplemented from CxChainable.

4.17.2.4 virtual void CxDemosaic::setupStop ( ) [protected], [virtual]

Sets up the object after the object is stopped.

Remarks

The method is internally used by the stop() method immediatelly after the run() method is finished. For example, the CxCameraSource stops camera acquisition here. The default implementation does nothing.

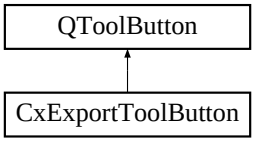
Reimplemented from CxChainable.

The documentation for this class was generated from the following file:

- Demosaic.h

4.18 CxExportToolButton Class Reference

Inheritance diagram for CxExportToolButton:



Public Types

- enum **ExExportType** { **extImageToClipboard** = 0x01, **extImageToFile** = 0x02, **extDataToClipboard** = 0x04, **extDataToTextFile** = 0x08 }

Signals

- void **exportClicked** (int eType)

Public Member Functions

- **CxExportToolButton** (ExExportTypes aSupportedExportTypes, QWidget *parent=0)
- void **addSupportedExportType** (ExExportType eType)
- void **setExportType** (ExExportType eType)
- ExExportType **exportType** ()

Static Public Member Functions

- static QString **textForType** (ExExportType eType)
- static QIcon **iconForType** (ExExportType eType)

Private Slots

- void **on_exportMenuClicked** ()
- void **on_selfClicked** ()

Private Attributes

- QMenu * **m_pPopupMenu**
- ExExportType **m_eSelExportType**

The documentation for this class was generated from the following file:

- ExportToolButton.h

4.19 CxFFT Class Reference

The class covers several 1D and 2D FFT functions.

Static Public Member Functions

- static bool Alloc2D (const SxPicBufInfo &picSrc, void *&pCfg, void *&pFFTin, void *&pFFTout)

  Preallocates a buffers needed for 2D FFT. The buffers must be freed using Free2D()
- static void Free2D (void *pCfg, void *pFFTin, void *pFFTout)

  Frees the temporary buffers allocated by Alloc2D()
- static bool Calc2D (const SxPicBuf &picSrcMono, SxPicBuf &picRe, SxPicBuf &picIm, void *pCfg, void *p↩ FFTin, void *pFFTout)

  Performs the 2D FFT on the `picSrcMono` image.
- static bool Calc2D (const SxPicBuf &picSrcMono, SxPicBuf &picRe, SxPicBuf &picIm)

  Performs the 2D FFT on the `picSrcMono` image.
- static bool Alloc1D (int iLength, void *&pCfg, void *&pFFTin, void *&pFFTout)

  Preallocates a buffers needed for 1D FFT. The buffers must be freed using Free1D()
- static void Free1D (void *pCfg, void *pFFTin, void *pFFTout)

  Frees the temporary buffers allocated by Alloc1D()
- static bool Calc1D (float *pInput, float *pReal, float *pImaginary, void *pCfg, void *pFFTin, void *pFFTout)

  Performs the 1D FFT on the `pInput` array.
- static bool Calc1D (float *pInput, float *pReal, float *pImaginary, int iLength)

  Performs the 1D FFT on the `pInput` array.

4.19.1 Detailed Description

The class covers several 1D and 2D FFT functions.

4.19.2 Member Function Documentation

4.19.2.1 static bool CxFFT::Calc1D (float * *pInput*, float * *pReal*, float * *pImaginary*, void * *pCfg*, void * *pFFTin*, void * *pFFTout*) [static]

Performs the 1D FFT on the `pInput` array.

Remarks

The `pReal` and `pImaginary` are array of the same lenght as `pInput` and the length must be the same as passed to the Alloc1D(). The `pCfg`, `pFFTin` and `pFFTout` are buffers preallocated by Alloc1D().

4.19.2.2 static bool CxFFT::Calc1D (float * *pInput*, float * *pReal*, float * *pImaginary*, int *iLength*) [static]

Performs the 1D FFT on the `pInput` array.

Remarks

The `pReal` and `pImaginary` are array of the same length as `pInput` given by `iLength`.

Warning

This function allocates the temporary buffers internally for each call so that it is less effective.

4.19.2.3 static bool CxFFT::Calc2D (const SxPicBuf & *picSrcMono*, SxPicBuf & *picRe*, SxPicBuf & *picIm*, void * *pCfg*, void * *pFFTin*, void * *pFFTout*) [static]

Performs the 2D FFT on the `picSrcMono` image.

Remarks

The `picRe` and `picIm` are images of the same size as `picSrc`. The image type of these images must be extypeInt32. The `pCfg`, `pFFTin` and `pFFTout` are buffers preallocated by Alloc2D().

4.19.2.4 static bool CxFFT::Calc2D (const SxPicBuf & *picSrcMono*, SxPicBuf & *picRe*, SxPicBuf & *picIm*) [static]

Performs the 2D FFT on the `picSrcMono` image.

Remarks

The `picRe` and `picIm` are images of the same size as `picSrc`. The image type of these images must be extypeInt32.

Warning

This function allocates the temporary buffers internally for each call so that it is less effective. It is suitable for static images.

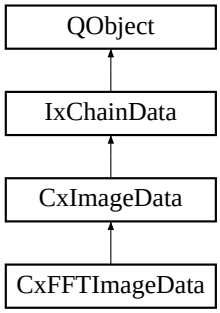
The documentation for this class was generated from the following file:

- FFT.h

4.20 CxFFTImageData Class Reference

Implementation of CxImageData that can keep the FFT on an image. It consists of real and imaginary parts both of type extypeInt32.

Inheritance diagram for CxFFTImageData:



Public Member Functions

- virtual IxChainData * clone (TxMemPoolHandle hMemPool, QObject *pMemPoolClient) const

  Creates the deep copy of this object. If the `hMemPool` is not NULL and it is a valid memory pool, then the data will be allocated from the memory pool. If the `hMemPool` is NULL then the data will be allocated from heap (operator new).
- virtual void setPicBuf (const SxPicBuf &picBuf)

  Set she real part the the FFT image. It must be one-channel image of type extypeInt32.
- bool setPicBufIm (const SxPicBuf &picIm)

  Sets the imaginary part of the FFT image. It must be one-channel image of type extypeInt32.
- SxPicBuf & picBufIm ()

  Reference to the imaginary parth of the FFT image. Can be modified.
- const SxPicBuf & picBufIm () const

  Const variant that should not change the returned buffer.

Protected Attributes

- SxPicBuf **m_aPicIm**

4.20.1 Detailed Description

Implementation of CxImageData that can keep the FFT on an image. It consists of real and imaginary parts both of type extypeInt32.

Remarks

The setPicBuf() and picBuf() methods sets/gets the real part of the FFT.

Warning

This class can keep only one-channel images!

The documentation for this class was generated from the following file:

- ImageData.h

4.21 CxHsiCameraParams::CxFilterZone Class Reference

Public Member Functions

- bool **isValid** () const
- CxBand ∗ bandAt (int xPos, int yPos)

  For xPos < m_iPatternWidth and yPos < m_iPatternHeight.
- const CxBand ∗ bandAt (int xPos, int yPos, int ∗piIdx=NULL) const

  For xPos < m_iPatternWidth and yPos < m_iPatternHeight.
- const CxBand ∗ bandAtPicPos (int xPos, int yPos, int ∗piIdx=NULL) const

  Use m_rcActiveArea to get the pattern pos.
- bool isPicPosInZone (int xPos, int yPos) const

  Use m_rcActiveArea to determine if picture position belongs into this zone.
- QVector< int > sortedBandIdxs (bool bOnlySelected=false) const

  Indexes to band vector sorted by band peak wavelength.
- int selectedBandsCount () const

  Number of selected bands in the spectrum.
- QVector< int > selectedBandsMapping () const

  Mapping of band index to selected band index. If a band is not selected then the mapping index is -1.

Public Attributes

- ExPattern m_eFilterPattern

  Mosaic?
- int **m_iPatternWidth**
- int m_iPatternHeight

  Pattern size (4x4, 5x5 for mosaic, or 1∗h for line scan)
- int **m_iFilterWidth**
- int m_iFilterHeight

  Filter size (1x1 for mosaic, w∗X for line scan) - repetition of same band inside pattern.
- QRect m_rcActiveArea

  Filter active area (optional, zero when N/A)
- int **m_iSpectralRangeStartNm**
- int m_iSpectralRangeEndNm

  Active sensor range in nm (optional, zero when N/A)
- QVector< CxBand > **m_vecBands**

The documentation for this class was generated from the following file:

- HsiCameraParams.h

4.22 CxImageFormatManager::CxFormatItem Class Reference

Public Attributes

- QString m_sRttiClass

  CxRTTI class name.
- TxImageFormatList m_lstFormats

  list of supported formats

The documentation for this class was generated from the following file:

- ImageFormatManager.h

4.23 CxGuiFunctions Class Reference

Static Public Member Functions

- static void makeToolbarButtonsSameHeight (QToolBar *pToolbar)

  Toolbar buttons without icon may be smaller then those that have one - looks horrible.
- static QMessageBox::StandardButton warningMsgBox (bool &bShowNextTime, QWidget *parent, const QString &title, const QString &text, QMessageBox::StandardButtons buttons=QMessageBox::Standard↩ Buttons(QMessageBox::Ok), QMessageBox::StandardButton defaultButton=QMessageBox::NoButton)

  Displays a warning message box with "Do not show next time this message" checkbox.
- static QMessageBox::StandardButton informationMsgBox (bool &bShowNextTime, QWidget *parent, const QString &title, const QString &text, QMessageBox::StandardButtons buttons=QMessageBox::Standard↩ Buttons(QMessageBox::Ok), QMessageBox::StandardButton defaultButton=QMessageBox::NoButton)

  Displays an information message box with "Do not show next time this message" checkbox.
- static QMessageBox::StandardButton criticalMsgBox (bool &bShowNextTime, QWidget *parent, const QString &title, const QString &text, QMessageBox::StandardButtons buttons=QMessageBox::Standard↩ Buttons(QMessageBox::Ok), QMessageBox::StandardButton defaultButton=QMessageBox::NoButton)

  Displays a critical message box with "Do not show next time this message" checkbox.
- static QPixmap dpiScaledPixmap (const QString &sPath, int iOrigWidth=0)

  Image scaled to current DPI.

The documentation for this class was generated from the following file:

- GuiFunctions.h

4.24 CxHistoData Class Reference

Class holding measured histogram and pixel statistics data.

Classes

- class CxPixelStats

  Class for measurement of data in the ROI/image per channel.

Public Types

- enum ExHistoDataFlag { eMeasureHisto = 0x01, eMeasureStats = 0x02, eSkipRows = 0x04, eTreatRawAsMono = 0x08 }

Public Member Functions

- CxHistoData & **operator=** (const CxHistoData &rhs)
- void **allocChannels** (int iTotalChannels, quint32 uiMaxVal, bool bIncludeStats)
- void **freeChannelData** ()
- bool **isChannelDataCompatible** (const CxHistoData &other) const
- void **calcPixelStats** (bool bEnableAlloc=false)
- void **sumWithData** (const CxHistoData &other)
- void **maxWithData** (const CxHistoData &other)
- void **divideBy** (quint32 uiVal)

Public Attributes

- quint32 m_uiMaxVal

  max value according to bpc (i.e. 1024 for 10 bit images)
- ExColorFilterArray m_eCfa

  RAW format of the image.
- QVector< quint32 * > m_vecChannelData

  measured histogram per channel
- QVector< QColor > m_vecChannelColor

  suggested channel colors
- QVector< QString > m_vecChannelName

  suggested channel names
- QVector< CxPixelStats > m_vecChannelStats

  measured pixel statistics
- QRect m_rc

  rectangle that was measured

4.24.1 Detailed Description

Class holding measured histogram and pixel statistics data.

4.24.2 Member Enumeration Documentation

4.24.2.1 enum CxHistoData::ExHistoDataFlag

Enumerator

eMeasureHisto measure histogram (fill m_vecChannelData)

eMeasureStats measure pixel statistics (fill m_vecChannelStats)

eSkipRows skip rows in large images

eTreatRawAsMono RAW image are usually split into channels, keep as mono.

The documentation for this class was generated from the following file:

- PicBufMeas.h

4.25 CxHsiCameraParams Class Reference

Classes

- class CxBand
- class CxCorrectionMatrix
- class CxFilterZone
- class CxOpticalComponent

  < Calibration data for used rejection filter (on camera, may not be present)
- class CxVirtualBand

  < Band record inside <virtual_bands>
- struct SxBandIdx

Public Types

- enum **ExPattern** { **epNone**, **epMosaic**, **epTiled**, **epWedge** }
- enum ExCorrectionMatrixType { ecmtHyperspectral = 0, ecmtRgb = 1, ecmtPanchromatic = 2 }
- enum ExCorrectionMatrixAlgorithm { **ecmaPitchfork** = 0, ecmaTacktile = 1, ecmaBcls = 2 }
- enum **ExOpticalComponentType** {
  eoctShortpassFilter = 0, **eoctLongpassFilter** = 1, **eoctBandpassFilter** = 2, **eoctNotchFilter** = 3,
  eoctLinearPolarizer = 4, **eoctCircularPolarizer** = 5, **eoctSource** = 6 }
- typedef QVector< SxBandIdx > **TxBandIdxVector**

Public Member Functions

- bool **isValid** () const
- bool isSimpleMosaicFilter () const

  Returns true if there is only one zone, and is mosaic.
- int totalBandCountInAllZones () const

  Returns the sum of number of bands in all zones.
- const CxFilterZone * filterZoneWithTotalBandIdx (int iTotalBandIdx, SxBandIdx *pIdx=NULL) const

  Find the zone the band belongs to when iterating from 0 to totalBandCountInAllZones()
- const CxFilterZone * filterZoneAtPicPos (int xPos, int yPos) const

  use m_rcActiveArea to get the pattern pos
- const CxBand * bandAtPicPos (int xPos, int yPos, SxBandIdx *pIdx=NULL) const

  use m_rcActiveArea to get the pattern pos
- TxBandIdxVector sortedBandIdxs (bool bOnlySelected=false) const

  Indexes to band vector sorted by band peak wavelength across all filter zones.
- void removeDataOutsideFilterRange ()

  Removes band peaks and correction matrixes outside current filter range.
- void removeBandCalibrationData ()

  Clears m_vecCalibData for all bands when no longer needed.
- bool isCorrectionMatrixPresent () const

  Returns true if the correction to virtual bands is available.
- const CxCorrectionMatrix * currentCorrectionMatrix () const

  Returns matrix if the correction to virtual bands for current filter range is available.
- bool loadFromCameraXmlFile (const QString &sFilename, QString *psErrorMsg=NULL)

  Load from camera XML file.
- bool loadFromXml (QDomElement *pRootElement)

  Load from XML in metadata or camera XML file.
- bool saveToXml (QDomDocument *pXmlDoc, QDomElement *pRootElement) const

  Save to XML compatible with camera XML.
- void **loadFromHsiApiStruct** (const XI_HSI_CAMERA_PARAMS *pParams)

Static Public Member Functions

- static bool **isHsiCamera** (XICORE_HANDLE hCamera)

Public Attributes

- int **m_iCalibrationRangeStartNm**
- int m_iCalibrationRangeEndNm

  Range of band calibration data (optional, zero when N/A)
- int m_iCalibrationResNm

  Calibration data resolution (is usually 1 nm) (optional, zero when N/A)
- int **m_iFilterRangeStartNm**
- int m_iFilterRangeEndNm

  Range of the active filter on camera (matches the filename, not read from XML)
- QVector< CxFilterZone > m_vecFilterZones

  List of zones on the sensor with active areas, bands definitions (>= 1)
- QVector< CxCorrectionMatrix > m_vecCorrectionMatrices

  Possible filter options for the camera sensor (as read from camera XMLs)

4.25.1 Member Enumeration Documentation

4.25.1.1 enum CxHsiCameraParams::ExCorrectionMatrixAlgorithm

Enumerator

ecmaTacktile correction matrix must be applied prior to reflectance calculation

ecmaBcls correction matrix must be applied after reflectance calculation

4.25.1.2 enum CxHsiCameraParams::ExCorrectionMatrixType

Enumerator

ecmtHyperspectral matrix generates corrected hyperspectral samples

ecmtRgb matrix generates true color RGB samples

ecmtPanchromatic matrix generates panchromatic samples

The documentation for this class was generated from the following file:

- HsiCameraParams.h

4.26 CxHsiCorrectionImage Class Reference

Public Member Functions

- void **reset** ()
- bool **isEmpty** ()
- bool setPicbuf (const SxPicBuf &aPic, const CxImageMetadata &aMetadata, bool isWhiteImage)

  The ownership of the `aPic` is passed to this object.
- bool **initDefault** (const SxPicBufInfo &aPicInfo, const CxImageMetadata &aMetadata)
- CxImageMetadata **metadata** ()
- bool **load** (const QString &sPath, bool bIsWhite)
- bool **save** (const QString &sPath)
- const double * **meansPerBand** ()
- void **lock** ()
- void **unlock** ()

Static Public Member Functions

- static bool **apply** (CxHsiCorrectionImage &aWhite, CxHsiCorrectionImage &aDark, CxHsiCorrectionImage &aDarkBias, SxPicBuf &aPic, const CxImageMetadata &aPicMetadata, ExHsiCorrection eCorrType, QString ∗pErrMsg=NULL)
- static bool **validate** (CxHsiCorrectionImage &aWhite, CxHsiCorrectionImage &aDark, CxHsiCorrectionImage &aDarkBias, bool &bDarkValid, bool &bBiasValid, QString ∗pErrMsg=NULL)
- static bool **validate** (CxHsiCorrectionImage &aWhite, CxHsiCorrectionImage &aDark, CxHsiCorrectionImage &aDarkBias, SxPicBufInfo &aPicInfo, const CxImageMetadata &aPicMetadata, bool &bDarkValid, bool &b↩ BiasValid, QString ∗pErrMsg=NULL)

Private Member Functions

- void **calculateMeansPerBand** ()

Static Private Member Functions

- static bool **validateNoLock** (CxHsiCorrectionImage &aWhite, CxHsiCorrectionImage &aDark, CxHsiCorrectionImage &aDarkBias, bool &bDarkValid, bool &bBiasValid, QString ∗pErrMsg=NULL)
- static bool **validateNoLock** (CxHsiCorrectionImage &aWhite, CxHsiCorrectionImage &aDark, CxHsiCorrectionImage &aDarkBias, SxPicBufInfo &aPicInfo, const CxImageMetadata &aPicMetadata, bool &bDarkValid, bool &b↩ BiasValid, QString ∗pErrMsg=NULL)

Private Attributes

- SxPicBuf **m_pic**
- CxImageMetadata **m_mdata**
- double ∗ m_pMeansPerBand

  Mean values per each band in the center of the WHITE image. For dark images it is NULL.
- QMutex **m_lock**

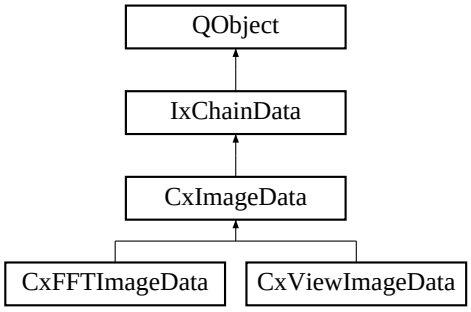
The documentation for this class was generated from the following file:

- CameraParameters.h

4.27 CxImageData Class Reference

The class implements an image data object exchanged by CxChainable objects.

Inheritance diagram for CxImageData:



Public Member Functions

- CxImageData ()

  Constructor.
- virtual ~CxImageData ()

  Destructor.
- virtual qint32 frameNo () const

  Gets the frame number of the data. For "static" data (e.g. static image) it is allways zero.
- virtual IxChainData * clone (TxMemPoolHandle hMemPool, QObject *pMemPoolClient) const

  Creates the deep copy of this object. If the `hMemPool` is not NULL and it is a valid memory pool, then the data will be allocated from the memory pool. If the `hMemPool` is NULL then the data will be allocated from heap (operator new).
- virtual IxChainData * clone (QObject *pMemPoolClient) const

  Creates the deep copy of this object from the same memory source as the original data.
- virtual bool isFromMemoryPool (TxMemPoolHandle hMemPool) const

  Checks if the data is from given memory pool.
- virtual SxPicBuf & picBuf ()

  Reference to the picture. Can be modified.
- virtual const SxPicBuf & picBuf () const

  Const variant should not change the returned buffer.
- virtual void **setPicBuf** (const SxPicBuf &picBuf)
- virtual const CxImageMetadata & **imageMetadata** () const
- virtual void **setImageMetadata** (const CxImageMetadata &aMetadata)

Protected Attributes

- SxPicBuf m_aPic
- CxImageMetadata **m_aMetadata**

4.27.1 Detailed Description

The class implements an image data object exchanged by CxChainable objects.

4.27.2 Member Data Documentation

4.27.2.1 SxPicBuf CxImageData::m_aPic [protected]

The image.

The documentation for this class was generated from the following file:

- ImageData.h

4.28 CxImageFormat Class Reference

Public Types

- enum EType { eSingle = 0x01, eSequence = 0x02 }

Public Member Functions

- **CxImageFormat** (const QString &sFormat, int iType)

Public Attributes

- QString m_sFormat

  Should be in format "Name (∗.ext)".
- int m_iType

  Combination of EType flags.

4.28.1 Member Enumeration Documentation

4.28.1.1 enum CxImageFormat::EType

Enumerator

eSingle Format can be used for saving single image.

eSequence Format can be used for saving sequence of images.

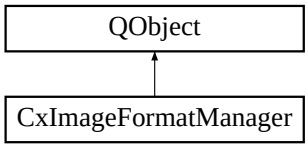
The documentation for this class was generated from the following file:

- ImageFormat.h

4.29 CxImageFormatManager Class Reference

Keeps list of supported file formats using CxRTTI, creates loader and saver objects.

Inheritance diagram for CxImageFormatManager:



Classes

- class CxFormatItem

Public Member Functions

- QStringList formatsForLoad (int iFormatTypes=CxImageFormat::eSingle)

  All formats we are capable of loading, items as "Name (∗.ext)".
- QStringList formatsForSave (int iFormatTypes=CxImageFormat::eSingle)

  All formats we are capable of saving, items as "Name (∗.ext)".
- QString stringWithAllFormatForLoad (int iFormatTypes=CxImageFormat::eSingle, bool bIncludeAllImages↩Filter=true)

  All formats we are capable of loading in format suitable for QFileDialog.
- QString stringWithAllFormatForSave (int iFormatTypes=CxImageFormat::eSingle)

  All formats we are capable of saving in format suitable for QFileDialog.
- IxImageDataLoader ∗ loaderForFile (const QString &sFileName)

  Creates new image loader instance according to the file extension.
- IxImageDataSaver ∗ saverForFile (const QString &sFileName)

  Creates new image saver instance according to the file extension.

- TxImageFormatOptions formatOptions (const QString &sExt)

  get defaults for file format
- void setFormatOptions (const QString &sExt, const TxImageFormatOptions &aOpt)

  set defaults for file format
- void saveSettings (QSettings *pSettings)

  store format options to app settings
- void loadSettings (QSettings *pSettings)

  load format options from app settings

Static Public Member Functions

- static CxImageFormatManager * instance ()

  Singleton class, this is the way to obtain the pointer.
- static QString fileExtFromFormatSignature (const QString &sFormatItem)

  returns "ext" from "name (.ext)"*
- static bool loadPicBufFromFile (const QString &sFileName, SxPicBuf &rPic, CxImageMetadata *pMetadata=NULL)

  Helper function to load one picture. The result is stored to rPic, caller is responsible freeing it.
- static bool savePicBufToFile (const QString &sFileName, const SxPicBuf *pPic, const CxImageMetadata *pMetadata=NULL)

  Helper function to store one picutre.

Private Slots

- void onRttiItemEnableStateChanged (TxRttiItem hItem, bool bEnabled)

  Handle the CxRTTI changes.

Private Member Functions

- void **resetFormatsForLoad** ()
- void **resetFormatsForSave** ()
- void loadAvailableLoaders ()

  Asks our CxRTTI to return all available loader classes.
- void loadAvailableSavers ()

  Asks our CxRTTI to return all available saver classes.

Static Private Member Functions

- static QString **findClassForFormat** (const QList< CxFormatItem > &lst, const QString &sExt)

Private Attributes

- bool m_bLoaderFormatsLoaded

  Did we call loadAvailableLoaders()?
- bool m_bSaverFormatsLoaded

  Did we call loadAvailableSavers()?
- QList< CxFormatItem > **m_lstFormatsForLoad**
- QList< CxFormatItem > **m_lstFormatsForSave**
- QMap< QString, TxImageFormatOptions > m_mapFormatOptions

  maps "ext" to its options (not class!)

4.29.1 Detailed Description

Keeps list of supported file formats using CxRTTI, creates loader and saver objects.

The documentation for this class was generated from the following file:

- ImageFormatManager.h

4.30 CxImageMetadata Class Reference

Metadata stored next to image pixel data.

Public Member Functions

- bool **isColorCorrMatrixEmpty** () const
- void fillFromCameraSettings (CxCameraParameters *pPars)

  Reads metadata values from current (cached) camera parameters.
- void saveToXml (QDomDocument *pXmlDoc, QDomElement *pRootElement) const

  Saves all values to XML document.
- bool loadFromXml (QDomElement *pRootElement)

  Loads all values from XML document.
- QString saveToXmlString () const

  Calls saveToXml, and saves the XML doc to string.
- bool loadFromXmlString (const QString &sString)

  Loads all values from XML doc stored in a string.

Static Public Member Functions

- static void **saveTag** (QDomDocument *pXmlDoc, QDomElement *pRootElement, const QString &sTag, const QString &sVal)

Public Attributes

- QString m_sCameraModel

  String got from XI_PRM_DEVICE_NAME.
- int m_iCameraModelId

  Integer got from XI_PRM_DEVICE_MODEL_ID.
- QString m_sCameraSerialNumber

  String got from XI_PRM_DEVICE_SN.
- QString m_sSensorSerialNumber

  String got from XI_PRM_DEVICE_SENS_SN.
- QString m_sCameraUserName

  String got from XI_PRM_DEVICE_USER_ID.
- QString m_sApiContextList

  String got from XI_PRM_API_CONTEXT_LIST.
- bool m_bAutoExposure

  Is auto exposuse switched on?
- int m_iExposure

  In microseconds - as in xiApi.
- float m_fGain

Gain in dB, as in xiApi.

- bool m_bAutoWB

  Is AWB switched on?

- float **m_fWhiteR**
- float **m_fWhiteG**
- float m_fWhiteB

  White balance coefficients.

- float m_fColorCorrMatrix [16]

  Current color correction matrix.

- float **m_fGammaY**
- float **m_fGammaC**
- float m_fSharpness

  Other color correction parameters.

- int m_iSensorTaps

  Number of sensor taps used when capturing the image.

- int m_iDownsamplingType

  Value of XI_PRM_DOWNSAMPLING_TYPE.

- float **m_fLensAperture**
- float m_fLensFocalLength

  Lens parameters (when present, 0 for none)

- float **m_fTempHousing**
- float m_fTempSensor

  Housing and sensor temperatures in deg Celsius.

- ExColorFilterArray m_eColorFilterArray

  Type of Bayer matrix, excfaNone when not RAW image.

- uint m_uTransportDataFormat

  The GenTL image format (see XI_GenTL_Image_Format_e in xiApi.h).

- CxHsiCameraParams m_aHsiParams

  Hyper Spectral Image information.

- QDateTime m_aAcqDateTime

  Date and time of exposure start.

4.30.1 Detailed Description

Metadata stored next to image pixel data.

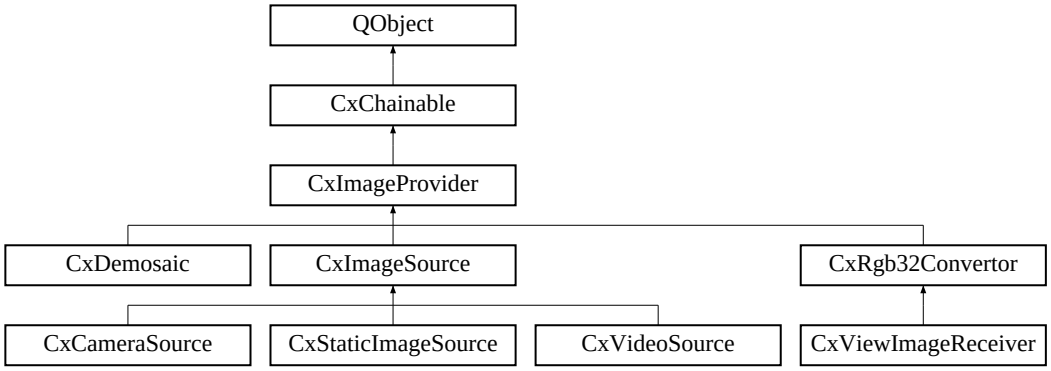
The documentation for this class was generated from the following file:

- ImageMetadata.h

4.31 CxImageProvider Class Reference

The base class for any chainable object which provides images.

Inheritance diagram for CxImageProvider:



Public Member Functions

- **CxImageProvider** (ExChainableFlags eFlags)
- virtual bool queryOutputImageInfo (const SxPicBufInfo &picInfoInput, SxPicBufInfo &picInfoOutput, const CxImageMetadata *pMetadataInput, CxImageMetadata *pMetadataOutput=NULL)=0

  Query the output image info (format) for given input image format.
- virtual bool currentOutputImageInfo (SxPicBufInfo &picInfo, CxImageMetadata *pMetadata=NULL)

  Gets the current output image info (format) provided by this object.
- virtual bool currentInputImageInfo (SxPicBufInfo &picInfo, CxImageMetadata *pMetadata=NULL)

  Gets the current input image info (format) which this object receives on its input.
- virtual int queryImagesCountInMemoryPool (const SxPicBufInfo &picInfo, TxMemPoolHandle hPool)

  Gets the required number of images reserved in the memory pool by its successors.

Protected Member Functions

- virtual bool init (QString *pErrMsg=NULL)

  Initializes the object. When calling this method, the object MUST be stopped.
- virtual bool **initialized** ()

Private Member Functions

- int totalBuffersCountInMemoryPool (TxMemPoolHandle hPool)

  Returns the total buffers count required by this object and its successors. If this object does not have its own memory pool then it returns zero.

Additional Inherited Members

4.31.1 Detailed Description

The base class for any chainable object which provides images.

4.31.2 Member Function Documentation

4.31.2.1 virtual bool CxImageProvider::currentOutputImageInfo (SxPicBufInfo & *picInfo*, CxImageMetadata * *pMetadata* = NULL) [virtual]

Gets the current output image info (format) provided by this object.

Remarks

The method should call getCurrentOutputImageInfo() on its precedessors and calculate the output image info provided by this object.

Reimplemented in CxCameraSource, CxStaticImageSource, and CxVideoSource.

4.31.2.2 virtual bool CxImageProvider::init (QString * *pErrMsg* = NULL) [protected], [virtual]

Initializes the object. When calling this method, the object MUST be stopped.

Remarks

Methods init() and initialized() are internally used by the start() method. It should initialize the chainable object so that it is ready to start (start()) . If this object is the CxImageProvider then the method initializes internal memory pool.

Implements CxChainable.

4.31.2.3 virtual int CxImageProvider::queryImagesCountInMemoryPool (const SxPicBufInfo & *picInfo,* TxMemPoolHandle *hPool*) [virtual]

Gets the required number of images reserved in the memory pool by its successors.

Remarks

If this object uses its own memory pool then the method allways returns zero!

Reimplemented in CxViewImageReceiver.

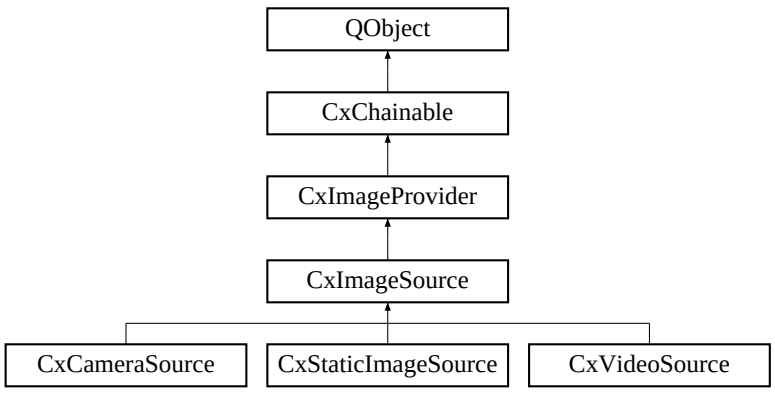
The documentation for this class was generated from the following file:

- Chainable.h

4.32 CxImageSource Class Reference

The base class of any image provider which is an image source, i.e. it has no precedessors and it provides images.

Inheritance diagram for CxImageSource:



Public Member Functions

- **CxImageSource** (ExChainableFlags eFlags)
- virtual bool acceptsDataFrom (CxChainable *pPrecedessor)

  Query if this object can accept data from the given chainable object.
- virtual bool addPrecedessor (CxChainable *pPrecedessor)

  Adds new precedessor to this object. It is internally used by CxChain class.
- virtual bool queryOutputImageInfo (const SxPicBufInfo &picInfoInput, SxPicBufInfo &picInfoOutput, const CxImageMetadata *pMetadataInput, CxImageMetadata *pMetadataOutput=NULL)

  Query the output image info (format) for given input image format.

Additional Inherited Members

4.32.1 Detailed Description

The base class of any image provider which is an image source, i.e. it has no precedessors and it provides images.

The documentation for this class was generated from the following file:

- Chainable.h

4.33 CxShadingPreset::CxItem Class Reference

Public Attributes

- QString m_sXiApiName

  Parameter name.
- CxValue m_vVal

  Parameter value.

The documentation for this class was generated from the following file:

- CameraParameters.h

4.34 CxLut Class Reference

Classes

- class CxCompContrast
- class CxPseudoColorTable

Public Types

- enum **ExDemosaicMode** { **exdemNone** = 0, **exdemMin** = 1, **exdemMax** = 2, **exdemAvg** = 3 }

Public Member Functions

- **CxLut** (quint32 uiInBpc, quint32 uiInComps)
- bool alloc ()

  allocate the mapping table

- bool allocRgb (quint32 uiRgbBits)

  allocate the mapping table to RGB24, RGB32 or RGB565 format
- CxLut * transformToFormat (quint32 uiInBpc, quint32 uiInComps) const

  create new Lut based on current setting
- bool **isCompatibleWith** (const SxPicBuf *pDstPic, const SxPicBuf *pSrcPic)
- bool **apply** (SxPicBuf *pDstPic, const SxPicBuf *pSrcPic, const CxImageMetadata *pSrcMetadata)

Static Public Member Functions

- static quint32 **fullMask** (quint32 uiComps)
- static quint32 **numCompsInMask** (quint32 uiMask)
- static int **firstCompInMask** (quint32 uiMask)
- static bool **addPseudoColorLut** (const QString &sFile)
- static int **pseudoColorLutCount** ()
- static QString **pseudoColorLutName** (int iIdx)
- static int **indexOfPseudoColorLutName** (const QString &sName)
- static void **setCustomPseudoColorLutFolder** (const QString &sFolderName)
- static QString **customPseudoColorLutFolder** ()
- static void **reloadCustomPseudoColorLuts** ()

Public Attributes

- quint32 m_uiSrcBpc

  bits ber component on input
- quint32 m_uiSrcComps

  number of input components
- quint32 m_uiCompMask

  binary mask of input comps idxs to map
- quint32 m_uiDstBpc

  bits ber component on output
- quint32 m_uiDstComps

  number of output components
- bool m_bSingleCompInColor

  show masked channel in its color or in gray
- int m_iPseudoColor

  in case one channel is mapped, use this pseodocolor map (-1 for none)
- quint32 m_clMonoCompColor

  when mono image with m_bSingleCompInColor, use this color
- QVector< CxCompContrast > m_vecContrast

  curve mapping per compoment. If size==1, same applied to all comps. Pre-created with size 1
- bool **m_bMarkOverexposed**
- bool m_bMarkUnderexposed

  mark over- or underexposed pixels?
- quint32 **m_clOverexposed**
- quint32 m_clUnderexposed

  color for over- and underexposed pixels (default: red, blue)
- enum CxLut::ExDemosaicMode **m_eDemosaicMode**

Protected Types

- enum ExMapMode {
  exmmUnset, exmmToRgb, exmmSeparateComps, exmmSeparate3Comps,
  exmmSeparateCompsSameTable, exmmSingleChannelToRgb }
- enum ExLutDataType { **exldtNull**, exldtUint8Ptr, exldtUint16Ptr, exldtUint32Ptr }

Protected Member Functions

- void init ()

  called from all constructors
- void **freeLutData** ()
- bool **applyOnOneCompToRgb** (SxPicBuf ∗pDstPic, const SxPicBuf ∗pSrcPic, int iComp, const CxImageMetadata ∗pSrcMetadata)
- void **markOverExposedRgb32** (SxPicBuf ∗pPic)
- void **markOverExposedRgb565** (SxPicBuf ∗pPic)

Protected Attributes

- bool **m_bWasAllocatedForRgb**
- enum CxLut::ExMapMode **m_eMapMode**
- void ∗ **m_pLutVoidData**
- enum CxLut::ExLutDataType **m_eLutDataType**
- SxPicBuf ∗ **m_pPicDemosaicCache**

4.34.1 Member Enumeration Documentation

4.34.1.1 enum CxLut::ExLutDataType [protected]

Enumerator

exldtUint8Ptr mapping components to 8-bit component

exldtUint16Ptr typically mapping single component to RGB16

exldtUint32Ptr typically mapping single component to RGB32

4.34.1.2 enum CxLut::ExMapMode [protected]

Enumerator

exmmToRgb try to convert the picture to RGB just using genering functions (no tables)

exmmSeparateComps map each component separately, each component has its own table

exmmSeparate3Comps like exmmSeparateComps, but use only first 3 components (for RGB32)

exmmSeparateCompsSameTable map each component separately, while using same table for all

exmmSingleChannelToRgb color is table for one comp, producing RGB image

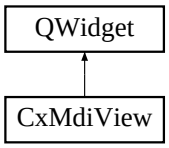
The documentation for this class was generated from the following file:

- Lut.h

4.35 CxMdiView Class Reference

The base of all MDI subwindows in the application.

Inheritance diagram for CxMdiView:



Public Types

- enum ExViewCap {
  evcCanZoomInOut = 0x01, evcCanZoomFit = 0x02, evcCanSaveToFile = 0x04, evcCanCapture = 0x10,
  evcCanRecordVideo = 0x20 }

Public Slots

- virtual void zoomIn ()

  Change zoom as user clicked on the toolbar button.
- virtual void zoomOut ()

  Change zoom as user clicked on the toolbar button.
- virtual void zoomFitToScreen ()

  Change zoom as user clicked on the toolbar button.
- virtual void **updateViewTitle** ()

Signals

- void receivedFrame (qint32 iFrame)

  New data arrived through chain to the view.
- void requestAdjustWindowSize (CxMdiView *pSender)

  Emit when parent MdiWindow should adjust its size to this view's "sizeHint() const".
- void requestCloseWindow (CxMdiView *pSender)

  For internal use only, called from closeMdiView()

Public Member Functions

- **CxMdiView** (CxChainable *pDataProvider, CxChainable *pDataReceiver)
- void **closeMdiView** ()
- int viewId ()

  Returns ID of this view.
- void setViewId (int iViewId)

  Do not use, main window sets this automatically.
- virtual ExViewCaps **viewCaps** ()
- virtual bool containsData ()=0

  Check whether view shows any data (maybe image view did not recieve any imageData)
- virtual bool **canZoomIn** ()
- virtual bool **canZoomOut** ()
- virtual bool save (QString &sLastUsedFolder)

Called from toolbar button, you should show the SaveAs dialog, or save the data directly.

- virtual bool saveToFile (const QString &sFilename)

  Save contents to file.
- virtual bool recordVideoToFile (const QString &sFilename)

  Start recording the video.
- virtual CxMdiView * **clone** (CxChainable *pNewDataProvider, CxChainable *pNewDataReceiver)=0
- CxChainable * dataProvider () const

  Returns chainable object view is connected to.
- CxChainable * dataReceiver () const

  Returns view's helper object (that is connected to dataProvider)
- CxImageSource * imageSource () const

  Image source in the chain view is connected to.
- virtual QString viewTitle () const

  Returns the view title excluding additional decorations (i.e. w/ zoom factor, fps)

Protected Attributes

- CxChainable * m_pDataReceiver

  The chainable object as a data receiver.
- int **m_iViewId**

4.35.1 Detailed Description

The base of all MDI subwindows in the application.

4.35.2 Member Enumeration Documentation

4.35.2.1 enum CxMdiView::ExViewCap

Enumerator

evcCanZoomInOut Enable zooming using + / - toolbar buttons.

evcCanZoomFit Enable zooming using bestFit toolbar button.

evcCanSaveToFile This view can save some data to file, enable toolbar button, and calling.

evcCanCapture Is reasonable to make snaphots of the data inside the view.

evcCanRecordVideo This view can record video.

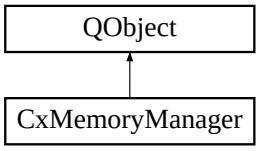
The documentation for this class was generated from the following file:

- MdiView.h

4.36 CxMemoryManager Class Reference

The singleton class for management of memory pools.

Inheritance diagram for CxMemoryManager:



Classes

- struct SxLimitDef

Public Types

- enum **ExAppOperation** { **eaopStartProgram**, **eaopStartAcquisition**, **eaopStartVideoRecord** }

Public Slots

- void finishAll ()

  Destroys all memory pools.

Signals

- void freePoolBuffers (TxMemPoolHandle hMemPool)

  The signal is emitted when the memory pool is freed and it still contains some allocated (non-released) buffers. The receiver of this signal MUST immediately releaese these buffers by calling freePicBuf(). It is very important to connect this signal with direct connection (see QObject::connect() and Qt::DirectConnection).

Public Member Functions

- ~CxMemoryManager ()

  Destructor.
- TxMemPoolHandle createPool ()

  Creates new empty memory pool.
- bool initPool (TxMemPoolHandle hPool, const SxPicBufInfo &picInfo, int iCount, int &rAllocatedCount)

  Initialization of the existing empty pool.
- void freePool (TxMemPoolHandle hPool)

  Releases the pool's internal buffers. It means that the memory pool itself will not be deleted. After calling this method the memory poll will be empty and it won't be possible to allocate any buffer from it. The method succeedes if the pool has all buffers free.
- bool deletePool (TxMemPoolHandle hPool)

  Tries to delete the memory pool.
- int buffersCount (TxMemPoolHandle hPool)

  Gets total number of buffers kept by this pool.
- int freeBuffersCount (TxMemPoolHandle hPool)

  Gets free buffers kept by this pool.
- bool poolSize (TxMemPoolHandle hPool, size_t &uiSize)

  Gets to total allocated size of the pool..
- bool bufferInfo (TxMemPoolHandle hPool, SxPicBufInfo &picInfo)

  Gets buffer info for which the pool was created.
- void reserveBuffersInPool (TxMemPoolHandle hPool, QObject *pClient, int iRequestCount)

  Objects (chainables) can reserve some buffer places for themselves, so that cloning imageData would not fail.
- void invalidateBufferReservations (TxMemPoolHandle hPool)

  Clear all reservations made in pool.
- bool allocPicBuf (TxMemPoolHandle hPool, QObject *pClient, SxPicBuf &picBuf, bool bReset)

  Allocates a buffer from given pool.
- bool freePicBuf (SxPicBuf &picBuf)

  Tries to releases the image buffer.
- void freeAllBuffers (TxMemPoolHandle hPool)

The method releases all buffers in the memory pool but will not dealocate the memory pool itself. If the pool is non empty then it emits the freePoolBuffers() signal.

- bool isImageCompatibleWithPool (TxMemPoolHandle hMemPool, const SxPicBufInfo &picInfo)

  Checks whether the given image info is compatible with the memory pool. It means whether it is possible to allocate the a buffer with given image info from the memory pool.
- bool loadLimits ()

  Loads the limits from XML file.
- bool checkLimits (ExAppOperation eOperation, XICORE_HANDLE hCam, QString &sMessage, const QString *pParamName=NULL, const CxValue *pParValue=NULL)

  Check a resources for given operation and camera.

Static Public Member Functions

- static CxMemoryManager * instance ()

  Gets the singleton instance of the class.
- static quint64 availableMemory ()

  Returns the amount of available RAM in bytes.
- static quint32 L2CacheSize ()

  Returns the size of L2 cache in bytes.

Private Slots

- void onFreePoolBuffers (TxMemPoolHandle hMemPool)

  Reaction on signal CxMemoryPool::freeBuffersRequest()

Private Member Functions

- CxMemoryManager ()

  Private constructor.

Private Attributes

- QMap< ExAppOperation, QList< SxLimitDef > > **m_mapLimitDefs**
- QSet< TxMemPoolHandle > m_setMemPools
- QMutex m_lock

4.36.1 Detailed Description

The singleton class for management of memory pools.

4.36.2 Member Function Documentation

4.36.2.1 bool CxMemoryManager::allocPicBuf (TxMemPoolHandle *hPool*, QObject * *pClient*, SxPicBuf & *picBuf*, bool *bReset*)

Allocates a buffer from given pool.

Remarks

The `picBuf` parameter must fit the `picInfo` parameter passed to createPool(). The method actualizes these members of the `picBuf`:

- SxPicBuf::m_pData
- SxPicBuf::m_uiAllocSize
- SxPicBuf::m_bDataOwner

4.36.2.2 bool CxMemoryManager::bufferInfo (TxMemPoolHandle *hPool*, SxPicBufInfo & *picInfo*)

Gets buffer info for which the pool was created.

See also

createPool()

4.36.2.3 int CxMemoryManager::buffersCount (TxMemPoolHandle *hPool*)

Gets total number of buffers kept by this pool.

Returns

If the method fails then it return -1.

4.36.2.4 bool CxMemoryManager::checkLimits (ExAppOperation *eOperation*, XICORE_HANDLE *hCam*, QString & *sMessage*, const QString * *pParamName* = NULL, const CxValue * *pParValue* = NULL)

Check a resources for given operation and camera.

Remarks

If it is neccessary to check an parameter and its value that is not set on the given camera yer, pass it via `pParamName` and `pParValue`. It is suitable for checking resources before parameter change.

4.36.2.5 TxMemPoolHandle CxMemoryManager::createPool ( )

Creates new empty memory pool.

See also

deletePool()
initPool()
freePool()

4.36.2.6 bool CxMemoryManager::deletePool (TxMemPoolHandle *hPool*)

Tries to delete the memory pool.

Remarks

The method internally calls freePool() and if it succeeds then the pool is deleted and the handle `hPool` became invalid.

4.36.2.7 void CxMemoryManager::finishAll ( ) [slot]

Destroys all memory pools.

Remarks

The method forces the memory pools destruction without checking if the pools contain only free (nonoccupied) buffers so that the method should be used very carefully. It should be called when the application is closing.

4.36.2.8 int CxMemoryManager::freeBuffersCount (TxMemPoolHandle *hPool*)

Gets free buffers kept by this pool.

Returns

If the method faise then it returns -1.

4.36.2.9 bool CxMemoryManager::freePicBuf (SxPicBuf & *picBuf*)

Tries to releases the image buffer.

Remarks

The function releases the buffer only when the SxPicBuf::m_hDataOwner contains a valid handle to memory pool.

4.36.2.10 void CxMemoryManager::freePool (TxMemPoolHandle *hPool*)

Releases the pool's internal buffers. It means that the memory pool itself will not be deleted. After calling this method the memory poll will be empty and it won't be possible to allocate any buffer from it. The method succeedes if the pool has all buffers free.

Remarks

The method does not delete the pool itself. The handle stays valid. It should be called before inicialization of the already initialized pool.

See also

deletePool()
initPool()

4.36.2.11 bool CxMemoryManager::initPool (TxMemPoolHandle *hPool*, const SxPicBufInfo & *picInfo*, int *iCount*, int & *rAllocatedCount*)

Initialization of the existing empty pool.

Remarks

The parameter `picInfo` must have set these members correctly:

- SxPicBufInfo::m_uiWidth
- SxPicBufInfo::m_uiHeight
- SxPicBufInfo::m_uiStride
- SxPicBufInfo::m_uiChannels.

4.36.3 Member Data Documentation

4.36.3.1 QMutex CxMemoryManager::m_lock [private]

Lock of this object.

4.36.3.2 QSet<TxMemPoolHandle> CxMemoryManager::m_setMemPools [private]

Map of memory pools and its threads.

The documentation for this class was generated from the following file:

- MemoryManager.h

4.37 CxHsiCameraParams::CxOpticalComponent Class Reference

< Calibration data for used rejection filter (on camera, may not be present)

Public Member Functions

- bool **isValid** () const
- void **clear** ()
- QString **typeString** () const

Public Attributes

- ExOpticalComponentType **m_eType**
- QString m_sManufacturer

  Manufacturer of the optical component.
- QString m_sPartId

  Textual description of the component.
- QString m_sDescription

  Unique part identifier as given by the manufacturer.
- QString m_sTag

  Custom tag uniquely identifying the component.
- int **m_iStartNm**
- int m_iEndNm

  Range of band calibration data.
- int m_iResNm

  Calibration data resolution (is usually 1 nm)
- QVector< double > m_vecCalibData

  Band response data, tag <response_composition> (not required for images)

4.37.1 Detailed Description

< Calibration data for used rejection filter (on camera, may not be present)

The documentation for this class was generated from the following file:

- HsiCameraParams.h

4.38 CxPicBufAPI Class Reference

Static Public Member Functions

- static size_t ImageTypeSize (ExImageDataType eDataType)

  Returns number of bytes occupied by given image data type.
- static bool AllocPicBufFromHeap (SxPicBuf &pic, quint32 uiWidth, quint32 uiHeight, quint32 uiChannels, ExImageDataType eDataType, quint32 uiStride=0, quint32 uiColorBitDepth=0, ExDataStorageFormat e↩ Format=exdataStoragePixel, ExColorFilterArray eCfa=excfaNone, bool bResetData=false, uint uiGenTL=0)

  Allocates and initializes the SxPicBuf structure.
- static bool AllocPicBufFromHeap (SxPicBuf &aPic, const SxPicBufInfo &aInfo, bool bResetData=false)

  Allocates and initializes the SxPicBuf using the initialized SxPicBufInfo structure. The allocation is preformed using the malloc() function from heap.
- static bool AllocPicBufFromPool (TxMemPoolHandle hPool, QObject ∗pMemPoolClient, SxPicBuf &aPic, const SxPicBufInfo &aInfo, bool bResetData=false)

  Allocates the `aPic` from the memory pool given by `hPool` handle using the initialized `aInfo` structure.
- static void FreePicBuf (SxPicBuf &pic, bool bReset=true)

  Deallocates the given image depending on SxPicBuf::m_bDataOwner.
- static bool CopyPicBuf (SxPicBuf &dstPic, const SxPicBuf &srcPic, bool bIncludeMetadata=true)

  Copies the `srcPic` to `dstPic`. The destination image must be correctly preallocated otherwise the function fails.
- static bool CopyRect (SxPicBuf &dstPic, quint32 dx, quint32 dy, const SxPicBuf &srcPic, quint32 sx, quint32 sy, quint32 w, quint32 h)

  Copies the rectangle from `srcPic` to `dstPic`.
- static bool CheckPicValidData (const SxPicBuf &pic)

  Checks if the data in N-bit image are really N-bit. E.g. returns false when value 1234 is stored in 10-bit image, where max value is 1023.
- static bool ConvertPicToRGB (const SxPicBuf &picSrc, SxPicBuf &picDst)

  Attepmt to convert source image to RGB, depending on the src and dst format.
- static bool ConvertGray8ToRGB32 (const SxPicBuf &picMono8, SxPicBuf &picRgb32)

  Converts the source MONO8 or RAW8 picbuf to picbuf in RGB32 format (0x00RRGGBB).
- static bool ConvertGray8ToRGB24 (const SxPicBuf &picMono8, SxPicBuf &picRgb24)

  Converts the source MONO8 or RAW8 picbuf to picbuf in RGB24 format (0xRRGGBB).
- static bool ConvertGray8ToRGB565 (const SxPicBuf &picMono8, SxPicBuf &picRgb565)

  Converts the source MONO8 or RAW8 picbuf to picbuf in RGB565 format.
- static bool ConvertGray16ToRGB32 (const SxPicBuf &picMono16, SxPicBuf &picRgb32)

  Converts the source MONO16 or RAW16 picbuf to picbuf in RGB32 format (0x00RRGGBB).
- static bool ConvertGray16ToRGB24 (const SxPicBuf &picMono16, SxPicBuf &picRgb24)

  Converts the source MONO16 or RAW16 picbuf to picbuf in RGB24 format (0xRRGGBB).
- static bool ConvertGray16ToGray8 (const SxPicBuf &picMono16, SxPicBuf &picMono8)

  Converts the source MONO16 or RAW16 picbuf to MONO8 or RAW8 picbuf.
- static bool ConvertGray16ToRGB565 (const SxPicBuf &picMono16, SxPicBuf &picRgb565)

  Converts the source MONO16 or RAW16 picbuf to picbuf in RGB565 format.
- static bool ConvertRGB24ToRGB32 (const SxPicBuf &picRgb24, SxPicBuf &picRgb32)

  Converts the source RGB24 (0xRRGGBB) picbuf to picbuf in RGB32 format (0x00RRGGBB).
- static bool ConvertRGB24ToRGB565 (const SxPicBuf &picRgb24, SxPicBuf &picRgb565)

  Converts the source RGB24 (0xRRGGBB) picbuf to picbuf in RGB565 format.
- static bool ConvertRGB32ToRGB24 (const SxPicBuf &picRgb32, SxPicBuf &picRgb24)

  Converts the source RGB32 (0x00RRGGBB) picbuf to picbuf in RGB24 format (0xRRGGBB).
- static bool ConvertRGB32ToRGB565 (const SxPicBuf &picRgb32, SxPicBuf &picRgb565)

  Converts the source RGB32 (0x00RRGGBB) picbuf to picbuf in RGB565 format.
- static bool ConvertRGBPlanarToRGB32 (const SxPicBuf &picRgbPlanar, SxPicBuf &picRgb32)

  Converts the source RGB Planar picbuf to picbuf in RGB32 format (0x00RRGGBB).

- static bool ConvertRGBPlanarToRGB24 (const SxPicBuf &picRgbPlanar, SxPicBuf &picRgb24)

  Converts the source RGB Planar picbuf to picbuf in RGB24 format (0xRRGGBB).
- static bool ConvertRGBPlanarToRGB565 (const SxPicBuf &picRgbPlanar, SxPicBuf &picRgb565)

  Converts the source RGB Planar picbuf to picbuf in RGB565 format.
- static bool ConvertFFTToRGB (const SxPicBuf &picRe, const SxPicBuf &picIm, SxPicBuf &picRGB, SxPicBuf *pTmp=NULL)

  Converts the source FFT images to a picbuf in the format of RGB.
- static bool ConvertFFTToRGB32 (const SxPicBuf &picRe, const SxPicBuf &picIm, SxPicBuf &picRGB32, SxPicBuf *pTmp=NULL)

  Converts the source FFT images to a picbuf in the format of RGB32.
- static bool ConvertFFTToRGB24 (const SxPicBuf &picRe, const SxPicBuf &picIm, SxPicBuf &picRGB24, SxPicBuf *pTmp=NULL)

  Converts the source FFT images to a picbuf in the format of RGB24.
- static bool ConvertFFTToRGB565 (const SxPicBuf &picRe, const SxPicBuf &picIm, SxPicBuf &picRGB565, SxPicBuf *pTmp=NULL)

  Converts the source FFT images to a picbuf in the format of RGB565.
- static bool ConvertPicToDifBpc (const SxPicBuf &picSrc, SxPicBuf &picDst, bool bStretchValues=false)

  Converts the source picbuf to picbuf with uint32 or uin16 type and same number of components.
- static void SwapRedAndBlue (SxPicBuf &picRgb)

  Swaps the red and green channels.
- static void SwapAllChannels (SxPicBuf &pic)

  Swaps all channels (i.e. ARGB -> BGRA).
- static void SwapBytes (SxPicBuf &pic)

  From little to big endian or back efficiently.
- static void FillChannelWithValue (SxPicBuf &pic, quint32 uiChannel, quint16 uiValue)

  Sets color to one component.
- static void ResetRGB32AlphaByte (SxPicBuf &pic)

  Sets alpha byte to 0xFF, as this is required by Qt in format QImage::Format_RGB32.
- static bool ReverseRowOrder (SxPicBuf &pic)

  The function reverses the row order of the image.
- static bool ReverseMemRowOrder (quint8 *pData, uint uiHeight, uint uiStride)

  The function reverses the row order of the memory block.
- static bool ReverseColOrder (SxPicBuf &pic)

  The function reverses the order of columns of the image.
- static bool DetectColorBitDepth (const SxPicBuf &picBuf, uint &uiBitDepth)

  The function detect the color bit depth of the image (per channel). Possible values that can be returned: 1, 8, 10, 12, 14, 16.
- static bool Resize (const SxPicBuf &srcPic, SxPicBuf &dstPic)

  Resizes the `srcPic` to the size given by `dstPic`. Both images must have the same image format. Resizing done using bilinear algorithm.
- static bool ResizeRect (const SxPicBuf &srcPic, quint32 sx, quint32 sy, quint32 sw, quint32 sh, SxPicBuf &dstPic, quint32 dx, quint32 dy, quint32 dw, quint32 dh)

  Resizes the rectangular section within `srcPic` to the rectangle within `dstPic`. Both images must have the same image format. Resizing done using bilinear algorithm.
- static bool ShiftBits (SxPicBuf &srcPic, int iShift)

  Shift pixel bits right (`iShift` > 0) or left (`iShift` < 0).
- static bool Subtract (SxPicBuf &pic1, const SxPicBuf &pic2)

  Subtracts values of pic2 from pic1, leaving the result in pic1.
- static bool Add (SxPicBuf &pic1, const SxPicBuf &picToAdd)

  Adds values of picToAdd to pic1, leaving the result in pic1. Pic1 can be Uint32.
- static bool DivideByValue (SxPicBuf &picSrc, int iDivideBy)

  Divides value in src picture by constant. picSrc can be Uint32.

- static bool DivideByValue (const SxPicBuf &picSrc, int iDivideBy, SxPicBuf &picDst)

  Divides value in src picture by constant, storing result to dst. picSrc can be Uint32.
- static bool ExtractChannel (const SxPicBuf &picSrc, SxPicBuf &picDst, const quint32 uiComp)

  Extracts one channel from source image. Dst image must be mono with same size as src.
- static bool FilterMinOnMosaicPicBuf (const SxPicBuf &picSrc, const QRect &rcSrc, int iMosaicWidth, int iMosaicHeight, SxPicBuf &picDst)

  Applies the "min" filter on the mosaic image (RAW or HSI).
- static bool FilterMaxOnMosaicPicBuf (const SxPicBuf &picSrc, const QRect &rcSrc, int iMosaicWidth, int iMosaicHeight, SxPicBuf &picDst)

  Applies the "max" filter on the mosaic image (RAW or HSI).
- static bool FilterAvgOnMosaicPicBuf (const SxPicBuf &picSrc, const QRect &rcSrc, int iMosaicWidth, int iMosaicHeight, SxPicBuf &picDst)

  Applies the "average" filter on the mosaic image (RAW or HSI).
- static bool FilterNormalizeOnMosaicPicBuf (const SxPicBuf &picSrc, const QRect &rcSrc, int iMosaicWidth, int iMosaicHeight, SxPicBuf &picDst)

  Applies the "normalization" filter on the mosaic image (RAW or HSI). The output image must have the same size as input image.
- static bool Median (const SxPicBuf &picSrc, SxPicBuf &picDst, int iKernelRadius, void *pBuffer=NULL)

  Performs a median filterring on the `picSrc` with given `kernelRadius`.
- static bool MedianAllocBuffer (const SxPicBufInfo &picSrc, int iKernelRadius, void **pBuffer)

  Allocates a temporary buffer used by Median() function.
- static void MedianFreeBuffer (void *pBuffer)

  Frees the buffer allocated by MedianAllocBuffer().

4.38.1 Member Function Documentation

4.38.1.1 static bool CxPicBufAPI::AllocPicBufFromHeap (SxPicBuf & *pic*, quint32 *uiWidth*, quint32 *uiHeight*, quint32 *uiChannels*, ExImageDataType *eDataType*, quint32 *uiStride* = `0`, quint32 *uiColorBitDepth* = `0`, ExDataStorageFormat *eFormat* = `exdataStoragePixel`, ExColorFilterArray *eCfa* = `excfaNone`, bool *bResetData* = `false`, uint *uiGenTL* = `0`) `[static]`

Allocates and initializes the SxPicBuf structure.

Remarks

If the `uiStride` is zero then it will be calculated. If the `bResetData` is true then the allocated data will be set to zero. The allocation is preformed using the malloc() function from heap.

4.38.1.2 static bool CxPicBufAPI::AllocPicBufFromPool (TxMemPoolHandle *hPool*, QObject * *pMemPoolClient*, SxPicBuf & *aPic*, const SxPicBufInfo & *aInfo*, bool *bResetData* = `false`) `[static]`

Allocates the `aPic` from the memory pool given by `hPool` handle using the initialized `aInfo` structure.

Remarks

If the `hPool` is NULL, then the image will be allocated using AllocPicBufFromHeap()

4.38.1.3 static bool CxPicBufAPI::ConvertFFTToRGB (const SxPicBuf & *picRe*, const SxPicBuf & *picIm*, SxPicBuf & *picRGB*, SxPicBuf * *pTmp* = `NULL`) `[static]`

Converts the source FFT images to a picbuf in the format of RGB.

Remarks

In fact the function calculates an amplitued of the FFT and converts it to RGB. Internally it calls ConvertFFTToRGB32() or ConvertFFTToRGB24() or ConvertFFTToRGB565().

4.38.1.4 static bool CxPicBufAPI::ConvertFFTToRGB24 (const SxPicBuf & *picRe*, const SxPicBuf & *picIm*, SxPicBuf & *picRGB24*, SxPicBuf * *pTmp* = NULL) [static]

Converts the source FFT images to a picbuf in the format of RGB24.

Remarks

In fact the function calculates an amplitued of the FFT and converts it to RGB24.

4.38.1.5 static bool CxPicBufAPI::ConvertFFTToRGB32 (const SxPicBuf & *picRe*, const SxPicBuf & *picIm*, SxPicBuf & *picRGB32*, SxPicBuf * *pTmp* = NULL) [static]

Converts the source FFT images to a picbuf in the format of RGB32.

Remarks

In fact the function calculates an amplitued of the FFT and converts it to RGB32.

4.38.1.6 static bool CxPicBufAPI::ConvertFFTToRGB565 (const SxPicBuf & *picRe*, const SxPicBuf & *picIm*, SxPicBuf & *picRGB565*, SxPicBuf * *pTmp* = NULL) [static]

Converts the source FFT images to a picbuf in the format of RGB565.

Remarks

In fact the function calculates an amplitued of the FFT and converts it to RGB565.

4.38.1.7 static bool CxPicBufAPI::ConvertPicToDifBpc (const SxPicBuf & *picSrc*, SxPicBuf & *picDst*, bool *bStretchValues* = false) [static]

Converts the source picbuf to picbuf with uint32 or uin16 type and same number of components.

Remarks

Suitable for averaging, see Add and DivideByValue.

4.38.1.8 static bool CxPicBufAPI::DetectColorBitDepth (const SxPicBuf & *picBuf*, uint & *uiBitDepth*) [static]

The function detect the color bit depth of the image (per channel). Possible values that can be returned: 1, 8, 10, 12, 14, 16.

Remarks

The function find maximum value in each channel and from the total maximum will quess the color bit depth. The color bit depth will be detected on images with 1 or 2 bytes per channel only.

4.38.1.9 static bool CxPicBufAPI::FilterAvgOnMosaicPicBuf (const SxPicBuf & *picSrc*, const QRect & *rcSrc*, int *iMosaicWidth*, int *iMosaicHeight*, SxPicBuf & *picDst*) [static]

Applies the "average" filter on the mosaic image (RAW or HSI).

Remarks

The output image can have either the size of:

- [picSrc.m_uiWidth, picSrc.m_uiHeight] for interpolated image,
- [rcSrc.width()/iMosaicWidth, rcSrc.height()/iMosaicHeight]. For interpolated image the inactive area of the input image is copied to the output image.

4.38.1.10 static bool CxPicBufAPI::FilterMaxOnMosaicPicBuf (const SxPicBuf & *picSrc,* const QRect & *rcSrc,* int *iMosaicWidth,* int *iMosaicHeight,* SxPicBuf & *picDst*) [static]

Applies the "max" filter on the mosaic image (RAW or HSI).

Remarks

The output image can have either the size of:

- [picSrc.m_uiWidth, picSrc.m_uiHeight] for interpolated image,
- [rcSrc.width()/iMosaicWidth, rcSrc.height()/iMosaicHeight]. For interpolated image the inactive area of the input image is copied to the output image.

4.38.1.11 static bool CxPicBufAPI::FilterMinOnMosaicPicBuf (const SxPicBuf & *picSrc,* const QRect & *rcSrc,* int *iMosaicWidth,* int *iMosaicHeight,* SxPicBuf & *picDst*) [static]

Applies the "min" filter on the mosaic image (RAW or HSI).

Remarks

The output image can have either the size of:

- [picSrc.m_uiWidth, picSrc.m_uiHeight] for interpolated image,
- [rcSrc.width()/iMosaicWidth, rcSrc.height()/iMosaicHeight]. For interpolated image the inactive area of the input image is copied to the output image.

4.38.1.12 static bool CxPicBufAPI::FilterNormalizeOnMosaicPicBuf (const SxPicBuf & *picSrc,* const QRect & *rcSrc,* int *iMosaicWidth,* int *iMosaicHeight,* SxPicBuf & *picDst*) [static]

Applies the "normalization" filter on the mosaic image (RAW or HSI). The output image must have the same size as input image.

Remarks

The normalization is performed per hyperpixel i.e. the hyperpixel's intensities are normalized to <0; max↩ Intensity>. An eventual inactive area of the input image is copied to the output image.

4.38.1.13 static void CxPicBufAPI::FreePicBuf (SxPicBuf & *pic,* bool *bReset* = true) [static]

Deallocates the given image depending on SxPicBuf::m_bDataOwner.

Remarks

If the SxPicBuf::m_eDataOwner member is not SxPicBuf::edoThisPicBuf then the function tries to release the buffer calling CxMemoryManager::freePicBuf(). The member SxPicBuf::m_pData is always set to NULL.

4.38.1.14 static bool CxPicBufAPI::Median (const SxPicBuf & *picSrc,* SxPicBuf & *picDst,* int *iKernelRadius,* void * *pBuffer* = NULL) [static]

Performs a median filterring on the `picSrc` with given `kernelRadius`.

Remarks

The kernel size used by the algorithm is (`kernelRadius`*2 + 1). The function operates with 8,10,12,14 and 16bit RGB or MONO images. The mosaic images (HSI, RAW) are not supported.

Temporary buffer `pBuffer`

If the `pBuffer` is NULL then the function allocates and frees it internally. For batch processing it is recommended to prealloc it before the batch starts. For allocation of the buffer use MedianAllocBuffer()

4.38.1.15 static bool CxPicBufAPI::ReverseColOrder (SxPicBuf & *pic*) [static]

The function reverses the order of columns of the image.

Warning

The function does not change the eventual SxPicBuf::m_eColorFilterArray for RAW images!!!

4.38.1.16 static bool CxPicBufAPI::ReverseRowOrder (SxPicBuf & *pic*) [static]

The function reverses the row order of the image.

Warning

The function does not change the eventual SxPicBuf::m_eColorFilterArray for RAW images!!!

The documentation for this class was generated from the following file:

- PicBufAPI.h

4.39 CxPicBufMeas Class Reference

Static Public Member Functions

- static bool **measureHistogram** (CxHistoData &rData, const CxImageData *pImageData, const QRect &rc, CxHistoData::ExHistoDataFlags aFlags)
- static bool **measureLineProfile** (CxProfileData &rData, const CxImageData *pImageData, const QLine &line, int iBandSize)

Static Private Member Functions

- static void **fillMeasChannels** (const SxPicBuf &rPic, QVector< QColor > &rvecChannelColor, QVector< QString > &rvecChannelName, bool bTreatRawAsMono)

The documentation for this class was generated from the following file:

- PicBufMeas.h

4.40 CxHistoData::CxPixelStats Class Reference

Class for measurement of data in the ROI/image per channel.

Public Attributes

- quint32 **m_uiMin**
- quint32 m_uiMax

  min, max
- double **m_dMean**
- double m_dStDev

  mean, standard deviation
- double m_dRms

  root mean square: sqrt(sum(x∗x)/n)

4.40.1 Detailed Description

Class for measurement of data in the ROI/image per channel.

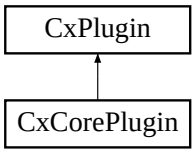
The documentation for this class was generated from the following file:

- PicBufMeas.h

4.41 CxPlugin Class Reference

Base class of all plugins. For more informations about plugin development see the plugins.

Inheritance diagram for CxPlugin:



Public Member Functions

- virtual bool init ()

  Called on application start.
- virtual void finish ()

  Called on application exit.
- virtual void loadSettings (QSettings ∗pSettings)

  Called right after init(), and then while loading program state.
- virtual void saveSettings (QSettings ∗pSettings)

  Called before finish(), and when program state is saved.
- virtual QString copyrightNotice () const

  In case plugin needs to write some legal info to Help - About - Legal...
- virtual QString name () const

  Plugin name displayed in the Plugin Manager view.
- virtual QString author () const

  Name of the author (company) of this plugin.

- virtual QString description () const

  Short description of the plugin, notes, warnings etc.
- virtual void version (int &major, int &minor, int &build) const

  Version of the plugin.
- virtual QString version () const

  Returns a version string. The default implementation returns a string formatted as M.m.b (i.e. Major.minor.build).
- virtual QString website () const

  The author's website adress.
- virtual void activeViewChanged (int iViewId)

  When new MDI view activated (or created)
- virtual void viewAboutToClose (int iViewId)

  The view will closed and deleted immediatelly.
- virtual void newCameraConnected (XICORE_HANDLE hCamera)

  After new camera connected after app start.
- virtual void cameraDisconnected (XICORE_HANDLE hCamera)

  After new camera disconnected during app session (not when exiting)
- virtual void viewGotNewImage (int iViewId, const CxImageData *pImageData)

  New image received when acqusition running.
- virtual bool viewMouseBtnPressEvent (int iViewId, QGraphicsSceneMouseEvent *event)

  Return true to prevent further processing.
- virtual bool viewMouseMoveEvent (int iViewId, QGraphicsSceneMouseEvent *event)

  Return true to prevent further processing.
- virtual bool viewMouseBtnReleaseEvent (int iViewId, QGraphicsSceneMouseEvent *event)

  Return true to prevent further processing.
- virtual bool viewMouseDoubleClickEvent (int iViewId, QGraphicsSceneMouseEvent *event)

  Return true to prevent further processing.
- virtual bool customCommand (const QString &sCmdName, int iParam1, int iParam2, void *pParam1, void *pParam2)

  for internal use.

4.41.1 Detailed Description

Base class of all plugins. For more informations about plugin development see the plugins.

The documentation for this class was generated from the following file:

- Plugin.h

4.42 CxProfileData Class Reference

Class holding measured line profile data.

Public Member Functions

- void **freeChannelData** ()

Public Attributes

- quint32 m_uiMaxVal

  max value according to bpc (i.e. 1024 for 10 bit images)

- int m_iLength

  size of arrays in m_vecChannelData

- QVector< float * > m_vecChannelData

  measured profile per channel (float because of averaging)

- QVector< QColor > m_vecChannelColor

  suggested channel colors

- QVector< QString > m_vecChannelName

  suggested channel names

4.42.1 Detailed Description

Class holding measured line profile data.

The documentation for this class was generated from the following file:

- PicBufMeas.h

4.43 CxLut::CxPseudoColorTable Class Reference

Public Member Functions

- bool **loadFromFile** (const QString &sFile)
- bool **saveToFile** (const QString &sFile)

Public Attributes

- QString **m_sName**
- quint32 **m_arrValues** [256]

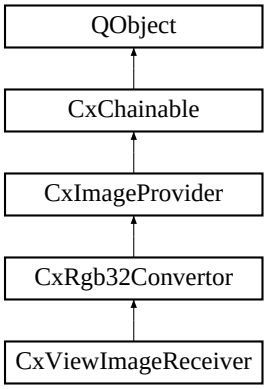
The documentation for this class was generated from the following file:

- Lut.h

4.44 CxRgb32Convertor Class Reference

The class converts all incoming image data to RGB32 format.".

Inheritance diagram for CxRgb32Convertor:



Public Types

- enum **ExConversionOutputFormat** { **ecofRGB32**, **ecofRGB24**, **ecofRGB565** }

Public Member Functions

- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.

- virtual bool acceptsDataFrom (CxChainable ∗pPrecedessor)

  Query if this object can accept data from the given chainable object.

- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*

- virtual CxChainable ∗ clone ()

  Creates a deep copy of this object.

- virtual bool queryOutputImageInfo (const SxPicBufInfo &picInfoInput, SxPicBufInfo &picInfoOutput, const CxImageMetadata ∗pMetadataInput, CxImageMetadata ∗pMetadataOutput=NULL)

  Query the output image info (format) for given input image format.

- void **setConversionOutputFormat** (ExConversionOutputFormat eFmt)
- ExConversionOutputFormat **conversionOutputFormat** ()
- void setLut (CxLut ∗pLut)

  set conversion LUTs. Objects takes ownership of the pointer.

- CxLut ∗ **lut** ()

Protected Member Functions

- virtual IxChainData ∗ processData (IxChainData ∗pReceivedData)

  The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

- virtual void setupStop ()

  Sets up the object after the object is stopped.

- bool **createConvertedPicBuf** (const CxImageData ∗pSrcData, SxPicBuf ∗pDstPic, CxImageMetadata ∗p↩ MetadataOutput)

Protected Attributes

- ExConversionOutputFormat **m_eConvOutputFormat**
- CxLut * **m_pLut**
- QMutex m_lockLuts
- SxPicBuf m_picFFTtmp

  Preallocated temporary buffer for FFT conversion to RGB.

Additional Inherited Members

4.44.1 Detailed Description

The class converts all incoming image data to RGB32 format.".

4.44.2 Member Function Documentation

4.44.2.1 virtual IxChainData* CxRgb32Convertor::processData (IxChainData * *pReceivedData*) [protected], [virtual]

The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

Remarks

If the implementation creates (allocates) a new data here then the original (received) data should be deleted using the delete operator. The default implementation of this method returns a pointer to the received data without any processing.

Warning

If the method returns NULL, then the `pReceivedData` will be deleted automatically!

Reimplemented from CxChainable.

Reimplemented in CxViewImageReceiver.

4.44.2.2 virtual void CxRgb32Convertor::setupStop ( ) [protected], [virtual]

Sets up the object after the object is stopped.

Remarks

The method is internally used by the stop() method immediatelly after the run() method is finished. For example, the CxCameraSource stops camera acquisition here. The default implementation does nothing.

Reimplemented from CxChainable.

4.44.3 Member Data Documentation

4.44.3.1 QMutex CxRgb32Convertor::m_lockLuts [protected]

Lock when working with LUTs.

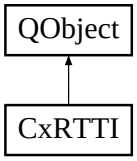
The documentation for this class was generated from the following file:

- Rgb32Convertor.h

4.45 CxRTTI Class Reference

Our RTTI builds upon Qt meta object system, and just adds class discovery.

Inheritance diagram for CxRTTI:



Signals

- void itemEnableStateChanged (TxRttiItem hItem, bool bEnabled)

  The signal is emitted from setItemEnabled() method every time the item's enable state changes.

Public Member Functions

- void registerClass (CxPlugin *pPlugin, const QMetaObject &aClassMetaObj, const QString &sInterface)

  This method is called from xiRTTI_REGISTER macro, do not use directly.
- QStringList classesForInterface (const QString &sInterface, bool bEnabledOnly=true)

  Queries available classes - get all classes that implement given interface.
- QString interfaceForClass (const QString &sClass)

  Returns an interface for the given registered class name. It returns empty string when the class is not registered.
- QString customNameForClass (const QString &sClass)

  Title that should be presented to user when enumerating the class.
- QString uniqueIdForClass (const QString &sClass)

  UniqueID of the class (it may not exist and the method may return empty string in that case).
- QString classForUniqueId (const QString &sUniqueId)

  Returns a classname for given UniqueID of chainable (if it exists).
- template<class T >
  T createInstanceAndCast (const QString &sClass)
- bool itemEnabled (TxRttiItem hItem)

  Checks whether the corresponding Rtti item is enabled (used by plugin manager).
- void setItemEnabled (TxRttiItem hItem, bool bEnable)

  Sets the corresponding Rtti item enabled/disabled.
- QString itemName (TxRttiItem hItem)

  Gets the "CustomName" of the corresponding Rtti item from metadata of the object.
- QMap< QString, QList< TxRttiItem > > itemsByInterface ()

  Returns Rtti items grouped by interface.
- const CxPlugin * pluginForItem (TxRttiItem hItem) const

  Returns a plugin from which this item was registered.
- QList< TxRttiItem > itemsForPlugin (const CxPlugin *pPlugin) const

  Returns a list of rtti items registered by the plugin.
- QString interfaceForItem (TxRttiItem hItem)

  Returns the interface which this item implements.
- void saveSettings (QSettings *pSettings)

  Saves the current state of the Rtti items (their enable state etc.)
- void loadSettings (const QSettings *pSettings)

  Loads the state of Rtti items.

Static Public Member Functions

- static CxRTTI * **instance** ()

Protected Member Functions

- QObject * createInstanceOfClass (const QString &sClass)

  Helper function. Use createInstanceAndCast instead.
- QString defaultNameForInterfaceClass (const QString &interfaceClass)

  If the Rtti item class has not the CustomName then this function generates a default name.
- QString nameFromMetadataForClass (const QString &sClass, const QString &sMetadataTag)

  Returns the name from metadata for given class and metadata tag.

4.45.1 Detailed Description

Our RTTI builds upon Qt meta object system, and just adds class discovery.

Registered interfaces are: CxChainable, IxImageDataLoader, IxImageDataSaver, IxVideoEncoder, IxVideoDecoder

4.45.2 Member Function Documentation

4.45.2.1 template<class T > T CxRTTI::createInstanceAndCast (const QString & *sClass*) [inline]

Creates instance of the class and casts for requected interface. Important: the default constructor of that class must be declared as Q_INVOKABLE!

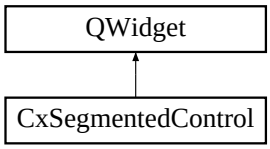
The documentation for this class was generated from the following file:

- xiRtti.h

4.46 CxSegmentedControl Class Reference

Set of exclusively checkable buttons next to each other, inspired by OS X.

Inheritance diagram for CxSegmentedControl:



Signals

- void **selectedSegmentChanged** ()

Public Member Functions

- **CxSegmentedControl** (QWidget *parent=0)
- void **setSegmentCount** (int nCount)
- void **setSelecedSegment** (int iIdx)
- int **selectedIndex** ()

- void **setSegmentText** (int iIdx, const QString &sText)
- void **setSegmentIcon** (int iIdx, const QIcon &rIcon, int iWidth=-1, int iHeight=-1)

Protected Member Functions

- QToolButton * **segmentBtn** (int iIdx)
- virtual bool **eventFilter** (QObject *pObject, QEvent *pEvent)

Protected Attributes

- int **m_nSegmentCount**

Private Slots

- void **buttonPressed** ()

4.46.1 Detailed Description

Set of exclusively checkable buttons next to each other, inspired by OS X.

The documentation for this class was generated from the following file:

- SegmentedControl.h

4.47 CxShadingPreset Class Reference

Classes

- class CxItem

Public Member Functions

- void saveCurrentValuesFrom (CxCameraParameters *pPars)

  Save values of defining camera parameters.
- void saveShadingImages (CxCameraParameters *pPars, SxPicBuf *pPicDark, SxPicBuf *pPicMid, const QString &sStorageFolder)

  Save shading images.
- void removeShadingImages ()

  Delete stored images.
- bool hasSameValuesAs (CxCameraParameters *pPars) const

  Are camera parameters same?
- void activateValuesTo (CxCameraParameters *pPars) const

  Activate shading correction.

Static Public Member Functions

- static QStringList definingParamList ()

  List of defining camera parameters (exposure, downsampling, etc...)
- static bool isShadingActive (CxCameraParameters *pPars)

  Returns true if shading is switched on on the camera (TODO: remove after parameter exists in xiApi)
- static void disableShading (CxCameraParameters *pPars)

  Switches off shading (TODO: remove after parameter exists in xiApi)

Public Attributes

- QString m_sName

  Name of the preset.
- QList< CxItem > m_lstParamValues

  Important camera parameters (defined in definingParamList())
- QRect m_rcRoi

  ROI is always stored.
- QString m_sDarkFilename

  File with dark frame for shaindg correction.
- QString m_sMidFilename

  File with light frame for shaindg correction.

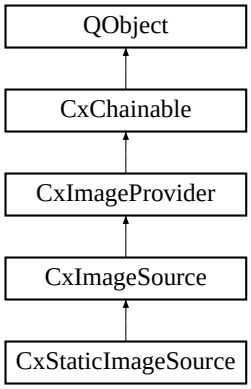
The documentation for this class was generated from the following file:

- CameraParameters.h

4.48 CxStaticImageSource Class Reference

The image source of static image.

Inheritance diagram for CxStaticImageSource:



Public Member Functions

- CxStaticImageSource (const QString &sPath)

  Creates the instance from the existing file name.
- CxStaticImageSource (const SxPicBuf &picBuf, const QString &sName)

  Creates the instance from picbuf.
- CxStaticImageSource (CxImageData *pImageData)

  Creates the instance from image data.
- ~CxStaticImageSource ()

  Destructor.
- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*

- virtual CxChainable * clone ()

  Creates a deep copy of this object.
- virtual IxChainData * data (QObject *pReceiver, qint32 iFrameNo)

  Gets the pointer to data processed by this object.
- virtual IxChainData * latestData (QObject *pReceiver)

  Gets the latest data from the output queue. It is called automatically if ExChainableFlag::eTakeLatestImage is set.
- virtual bool saveSettings (QDomDocument &aDoc, QDomElement &aElm) const

  The method saves the settings of this object to the DOM element.
- virtual bool loadSettings (const QDomElement &aDom)

  Loads settings from DOM.
- virtual bool currentOutputImageInfo (SxPicBufInfo &picInfo, CxImageMetadata *pMetadata=NULL)

  Gets the current output image info (format) provided by this object.
- void run ()

  This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).
- bool isLoaded () const

  Whether the static image is loaded correctly.
- bool reload ()

  Reloads the image.
- QString path ()

  Get the image full path.
- void setPath (const QString &path)

  Sets the image's path. The method will not reload the image when the path differs from the original path.

Protected Member Functions

- bool **loadFromFile** (const QString &sPath)

Private Attributes

- CxImageData * m_pData

  Pointer to image data.
- QString m_sPath

  Path of file.

Additional Inherited Members

4.48.1 Detailed Description

The image source of static image.

4.48.2 Member Function Documentation

4.48.2.1 virtual bool CxStaticImageSource::currentOutputImageInfo (SxPicBufInfo & *picInfo*, CxImageMetadata * *pMetadata* = NULL) [virtual]

Gets the current output image info (format) provided by this object.

Remarks

The method should call getCurrentOutputImageInfo() on its precedessors and calculate the output image info provided by this object.

Reimplemented from CxImageProvider.

4.48.2.2 virtual IxChainData* CxStaticImageSource::data (QObject * *pReceiver,* qint32 *iFrameNo*) [virtual]

Gets the pointer to data processed by this object.

See also

discardData()
CxChainable

Reimplemented from CxChainable.

4.48.2.3 void CxStaticImageSource::run ( ) [virtual]

This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).

See also

CxChainable

Reimplemented from CxChainable.

4.48.2.4 virtual bool CxStaticImageSource::saveSettings (QDomDocument & *aDoc,* QDomElement & *aElm*) const [virtual]

The method saves the settings of this object to the DOM element.

Remarks

The element `aElm` is already named by a caller, do not change the name (ie. do not call the aElm.setTag↩Name()). You can add attributes and child elements.

Reimplemented from CxChainable.

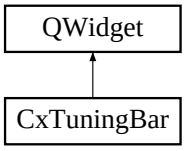
The documentation for this class was generated from the following file:

- StaticImageSource.h

4.49 CxTuningBar Class Reference

Our own implementation of slides, working with CxValue types etInt and etFloat.

Inheritance diagram for CxTuningBar:



Public Types

- enum **ExBarColorScale** {
  ebcsNone, **ebcsGray**, **ebcsRed**, **ebcsGreen**,
  ebcsBlue, **ebcsColorful** }
- enum **ExBarTransform** { **ebtLinear**, **ebtLogarithmic** }

Public Slots

- void **setValue** (const CxValue &vValue)
- void **setReadOnly** (bool bReadOnly)
- void **processExternalWheelEvent** (QWheelEvent ∗event)

Signals

- void **valueChanged** (const CxValue &vValue)

Public Member Functions

- **CxTuningBar** (QWidget ∗parent=0)
- void **setRange** (const CxValue &vMin, const CxValue &vMax, const CxValue &vIncrement)
- const CxValue & **rangeMin** () const
- const CxValue & **rangeMax** () const
- const CxValue & **increment** () const
- const CxValue & **value** () const

Public Attributes

- bool **m_bShowTickMarks**
- bool **m_bSnapToTickMarks**
- ExBarColorScale **m_eColorScale**
- ExBarTransform **m_eTransform**

Protected Member Functions

- int **calcPaintXPos** (const CxValue &vValue)
- CxValue **calcValueFromPos** (int nXPos)
- bool **setValueAndEmitSignal** (CxValue vNewVal)
- CxValue **smallStep** (int iRangeFraction, bool bUp)
- virtual QSize **sizeHint** () const
- virtual void **paintEvent** (QPaintEvent ∗)
- virtual void **mousePressEvent** (QMouseEvent ∗event)
- virtual void **mouseMoveEvent** (QMouseEvent ∗event)
- virtual void **mouseReleaseEvent** (QMouseEvent ∗event)
- virtual void **wheelEvent** (QWheelEvent ∗event)
- virtual void **keyPressEvent** (QKeyEvent ∗event)

Protected Attributes

- CxValue **m_vRangeMin**
- CxValue **m_vRangeMax**
- CxValue **m_vIncrement**
- CxValue **m_vValue**
- bool **m_bReadOnly**
- bool **m_bMovingArrow**
- int **m_iBorder_X**
- int **m_iSnapDist**

Static Protected Attributes

- static QPixmap * **m_pPicArrow**
- static QPixmap * **m_pPicArrowDis**
- static QPixmap * **m_pPicBarColorful**

4.49.1 Detailed Description

Our own implementation of slides, working with CxValue types etInt and etFloat.

The documentation for this class was generated from the following file:

- TuningBar.h

4.50 CxUnits Class Reference

Class the defines all known units and measures, and includes conversions between them.

Public Types

- enum ExUnitGroup {
  eugInvalid, **eugTime**, **eugLength**, **eugTemperature**,
  eugRatio }

  All measures defined here.
- enum **ExTime** {
  etMicrosecond = 0, **etMillisecond** = 1, **etSecond** = 2, **etMinute** = 3,
  etHour = 4 }
- enum **ExLength** { **etMicrometre** = 0, **etMillimetre** = 1, **etCentimetre** = 2, **etMetre** = 3 }
- enum **ExTemperature** { **etCelsius** = 0, **etFahrenheit** = 1, **etKelvin** = 2 }
- enum ExRatio { erFraction = 0, erPercent = 1 }

Static Public Member Functions

- static int countInGroup (ExUnitGroup eUnitGroup)

  Number of units for given group.
- static QString text (ExUnitGroup eUnitGroup, int iUnit)

  Text for the unit.
- static int unitFromText (ExUnitGroup eUnitGroup, const QString &sText)

  Unit from text, returns -1 when not found.
- static double **convertValue** (double dValue, ExUnitGroup eUnitGroup, int iUnitFrom, int iUnitTo)

4.50.1 Detailed Description

Class the defines all known units and measures, and includes conversions between them.

4.50.2 Member Enumeration Documentation

4.50.2.1 enum CxUnits::ExRatio

Enumerator

erFraction Ratio as decimal value between 0 and 1.

erPercent Ratio as percent value between 0 and 100.

The documentation for this class was generated from the following file:

- Units.h

4.51 CxValue Class Reference

Class for storing values similar to variant. Used in camera parameters.

Public Types

- enum Type {
  etInvalid, etInt, etFloat, etBool,
  etString }

Public Member Functions

- **CxValue** (int nVal)
- **CxValue** (float fVal)
- **CxValue** (bool bVal)
- **CxValue** (const QString &sVal)
- bool **operator==** (const CxValue &rOther) const
- bool **operator!=** (const CxValue &rOther) const
- bool **operator**< (const CxValue &rOther) const
- bool **operator**> (const CxValue &rOther) const
- CxValue **operator+** (const CxValue &rOther) const
- CxValue **operator-** (const CxValue &rOther) const
- CxValue **operator**∗ (const CxValue &rOther) const
- CxValue **operator/** (const CxValue &rOther) const
- int toInt () const

  Returns value as int (not implemented for etString)
- float toFloat () const

  Returns value as float (for etInt and etFloat)
- bool toBool () const

  Returns value as bool (value is 1)
- bool fromString (const QString &sString)

  Loading from string using the type already in m_eType.
- QString toString () const

  Save value to string.

Public Attributes

- enum CxValue::Type m_eType

  Value type.
- union {
  int **m_intVal**
  float **m_floatVal**
  bool **m_boolVal**
  };
- QString **m_stringVal**

4.51.1 Detailed Description

Class for storing values similar to variant. Used in camera parameters.

4.51.2 Member Enumeration Documentation

4.51.2.1 enum CxValue::Type

Enumerator

etInvalid Marks an error, or undefined value.

etInt Use m_intVal.

etFloat Use m_floarVal.

etBool Use m_boolVal.

etString Use m_stringVal.

The documentation for this class was generated from the following file:

- ValueTypes.h

4.52 CxValueRegInvalidator Class Reference

Class describing invalidator connection.

Public Member Functions

- **CxValueRegInvalidator** (const QString &sXiApiName, bool bReSet=false)
- **CxValueRegInvalidator** (const char ∗szXiApiName)

Public Attributes

- QString m_sXiApiName

  name of the parameter that invalidates this register
- bool m_bReSet

  whe set to true, we should set the read value back to API again

4.52.1 Detailed Description

Class describing invalidator connection.

The documentation for this class was generated from the following file:

- CameraParameters.h

4.53 CxValueRegister Class Reference

This class is responsible for the communitaion with the camera.

Public Member Functions

- bool readValue (XICORE_HANDLE hCamera, CxValue &vValue) const

  Note: vValue.m_eType must be correctly filler.
- bool **storeValue** (XICORE_HANDLE hCamera, const CxValue &vValue) const
- bool **isInvalidatedByParam** (const QString &sInvalidator) const
- bool **readIfInvalidated** (XICORE_HANDLE hCamera, CxValue &vValue, const QString &sInvalidator) const

Static Public Member Functions

- static bool **readValue** (XICORE_HANDLE hCamera, const QString &sXiApiParamName, CxValue &vValue)
- static bool **storeValue** (XICORE_HANDLE hCamera, const QString &sXiApiParamName, const CxValue &vValue, int *piRetVal=NULL)

Public Attributes

- QString m_sXiApiName

  which parameter should we call (read from XML, e.g. "exposure" or "exposure:min". If empty, no update possible
- TxValueRegInvalidatorList m_lstInvalidators

  paramaters that invalidate this register when they change

4.53.1 Detailed Description

This class is responsible for the communitaion with the camera.

The documentation for this class was generated from the following file:

- CameraParameters.h

4.54 CxVecObjContainer Class Reference

Our special class above QGraphicsScene maintains list of our vector objects and assigns IDs to them.

Public Member Functions

- **CxVecObjContainer** (QGraphicsScene *pScene)
- QRectF **sceneRect** () const
- TxVecObjID addObject (CxVectorObject *pObj)

Add object to scene, and assign it a new ID.

- void deleteObject (TxVecObjID idObj)

  Removes the object from scene and deletes the object.
- void deleteAllObjects ()

  Removes all our vector object from scene.
- QVector< TxVecObjID > listAllObjects ()

  Returns the list of all objects in the scene.
- CxVectorObject * objectWithId (TxVecObjID idObj)

  Returns pointer from ID.
- void **scaleObjects** (double sx, double sy, bool bReflectConstraints=true)
- void **moveObjects** (double dx, double dy, bool bReflectConstraints=true)
- void saveObjVisibilityState ()

  Save the per-object visility for later.
- void restoreObjVisibilityState ()

  Restores the state as saved in saveObjVisibilityState.
- void hideAllObjects ()

  Sets all vector objects as invisible.

Protected Attributes

- QGraphicsScene * m_pScene

  Each container has to be connected to a scene.
- TxVecObjID m_idMax

  Object ID generator.
- TxVectorObjectMap m_mapObjs

  Mapping from object id to the pointer.
- QMap< TxVecObjID, bool > m_mapObjVisibilityState

  Saved visibility state using saveObjVisibilityState.

4.54.1 Detailed Description

Our special class above QGraphicsScene maintains list of our vector objects and assigns IDs to them.

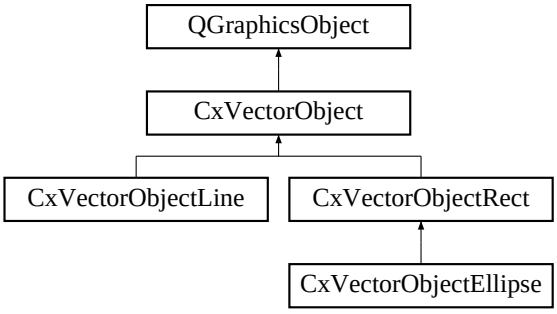
The documentation for this class was generated from the following file:

- VectorContainer.h

4.55 CxVectorObject Class Reference

Base class for all our vector objects in QGraphicsScene. Needed to send signals away.

Inheritance diagram for CxVectorObject:



Public Types

- enum ExCameraFormatChangeBehavior { excfcbAttachSensor, excfcbAttachScene }

Signals

- void customContextMenuRequested (const QPoint &pos)

  Sent after right mouse click on the object.
- void **doubleClicked** ()
- void **clicked** (int idObject)

Public Member Functions

- **CxVectorObject** (QGraphicsItem *parent=0)
- TxVecObjID **id** ()
- void **setId** (TxVecObjID idObj)
- virtual void **setPenColor** (const QColor &color)
- virtual void **setPenWidth** (int iWidth)
- virtual void setSelectedPenWidth (int iWidth)

  Set pen width of selected object.
- virtual void **setBrushColor** (const QColor &color)
- virtual void setCoordIncrement (int nXInc, int nYInc)

  0 for no limit, 1 for rounding to nearest int, 4 for coords in multiples of 4
- virtual void setResizable (bool bEnable)

  Enable object resizing with mouse.
- virtual void setKeepInScene (bool bKeep)

  Forces the object to stay within image bounds.
- virtual void scaleObject (double sx, double sy, bool bReflectConstraints=true)

  Object scaling.
- virtual void moveObject (double dx, double dy, bool bReflectConstraints=true)

  Object translation.
- QColor **penColor** ()
- QColor **brushColor** ()
- int **penWidth** ()
- int selectedPenWidth ()

  Width of a pen of selected object.
- ExCameraFormatChangeBehavior cameraFormatChangeBehavior ()

  See ExCameraFormatChangeBehavior.
- virtual void setCameraFormatChangeBehavior (ExCameraFormatChangeBehavior eMode)

  See ExCameraFormatChangeBehavior.

Protected Member Functions

- QPen **defaultPen** () const
- QBrush **defaultBrush** () const
- bool **reflectCoordIncrement** (QPointF &pt) const
- qreal **viewScaleAtEvent** (QGraphicsSceneEvent *pEvent)
- virtual void **mouseDoubleClickEvent** (QGraphicsSceneMouseEvent *event)
- virtual void **mousePressEvent** (QGraphicsSceneMouseEvent *event)

Protected Attributes

- TxVecObjID m_id

  ID as set by CxVecObjContainer.
- QColor m_clPen

  Color of the pen used when painting the object.
- QColor m_clBrush

  Fill color.
- int **m_iPenWidth**
- int m_iSelectedPenWidth

  Pen width of selected object.
- int m_nCoordXIncrement

  0 for no limit, 1 for rounding to nearest int, 4 for coords in multiples of 4
- int **m_nCoordYIncrement**
- bool m_bIsResizable

  It is possible to resize the object (rectangle, line end points, ...)
- bool m_bKeepInScene

  Do not allow to move the object outside the image boundaries.
- ExCameraFormatChangeBehavior m_eCameraFormatChangeBehavior

  Defaults to excfcbAttachSensor.

4.55.1 Detailed Description

Base class for all our vector objects in QGraphicsScene. Needed to send signals away.

4.55.2 Member Enumeration Documentation

4.55.2.1 enum CxVectorObject::ExCameraFormatChangeBehavior

Enumerator

excfcbAttachSensor Keep same position on chip relative to ROI and binning (default)

excfcbAttachScene Stay visually on same place in the image view.

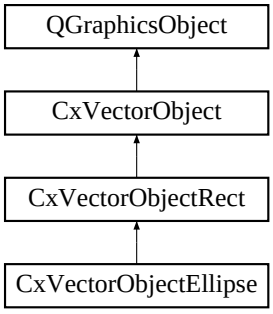
The documentation for this class was generated from the following file:

- VectorObject.h

4.56 CxVectorObjectEllipse Class Reference

Object representing the ellipse, using the functionality implemented in rectangle.

Inheritance diagram for CxVectorObjectEllipse:



Public Member Functions

- **CxVectorObjectEllipse** (CxVectorObject ∗parent=0)
- virtual void **paint** (QPainter ∗painter, const QStyleOptionGraphicsItem ∗, QWidget ∗)

Protected Member Functions

- virtual ExRectHitTest **hitTest** (qreal x, qreal y, qreal dViewScale)

Additional Inherited Members

4.56.1 Detailed Description

Object representing the ellipse, using the functionality implemented in rectangle.

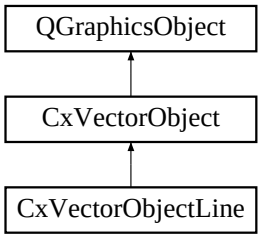
The documentation for this class was generated from the following file:

- VectorObject.h

4.57 CxVectorObjectLine Class Reference

Object representing the single line, from start point to end point.

Inheritance diagram for CxVectorObjectLine:



Public Types

- enum ExLineStyle { **elsFree**, elsHorizontal, elsVertical }

Signals

- void lineChanging (const QLineF &line)

  Object is being changed (sent during mouse moves)
- void lineChanged (const QLineF &line)

  Object has new geometry (sent after mouse up)

Public Member Functions

- **CxVectorObjectLine** (CxVectorObject ∗parent=0)
- QLineF line ()

  Returns line geometry.
- void setLine (const QLineF &line)

  Sets line geometry.

- virtual void setCoordIncrement (int nXInc, int nYInc)

  Reimplemented to check both end points.
- virtual void scaleObject (double sx, double sy, bool bReflectConstraints=true)

  Object scaling.
- virtual void moveObject (double dx, double dy, bool bReflectConstraints=true)

  Object translation.
- ExLineStyle **lineStyle** ()
- void **setLineStyle** (ExLineStyle eStyle)
- virtual void **paint** (QPainter ∗painter, const QStyleOptionGraphicsItem ∗, QWidget ∗)

Protected Types

- enum **ExLineHitTest** { **ehtNone**, **ehtEdge**, **ehtStartPoint**, **ehtEndPoint** }

Protected Member Functions

- double **distanceFromLine** (const QPointF &pt) const
- virtual ExLineHitTest **hitTest** (qreal x, qreal y, qreal dViewScale)
- virtual QRectF **boundingRect** () const
- virtual void **hoverMoveEvent** (QGraphicsSceneHoverEvent ∗event)
- virtual void **mousePressEvent** (QGraphicsSceneMouseEvent ∗event)
- virtual void **mouseMoveEvent** (QGraphicsSceneMouseEvent ∗event)
- virtual void **mouseReleaseEvent** (QGraphicsSceneMouseEvent ∗event)

Protected Attributes

- QPointF **m_ptEnd**
- ExLineStyle **m_eLineStyle**
- enum CxVectorObjectLine::ExLineHitTest **m_eLastHitTest**
- bool **m_bResizing**
- QPointF **m_ptStartDragMousePos**
- QPointF **m_ptStartPos**

4.57.1 Detailed Description

Object representing the single line, from start point to end point.

4.57.2 Member Enumeration Documentation

4.57.2.1 enum CxVectorObjectLine::ExLineStyle

Enumerator

elsHorizontal Force line to be horizontal.

elsVertical Force line to be vertical.

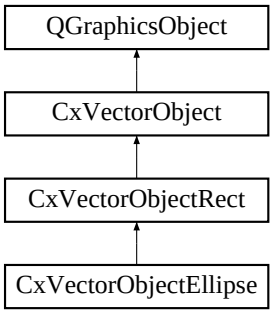
The documentation for this class was generated from the following file:

- VectorObject.h

4.58 CxVectorObjectRect Class Reference

Object representing the rectangular object. Implements mouse resizing and caption.

Inheritance diagram for CxVectorObjectRect:



Public Types

- enum **ExRectShape** { **ersSharpCorners**, **ersRoundCorners** }

Signals

- void rectChanging (const QRectF &rc)

  Object is being changed (sent during mouse moves)
- void rectChanged (const QRectF &rc)

  Object has new geometry (sent after mouse up)

Public Member Functions

- **CxVectorObjectRect** (CxVectorObject ∗parent=0)
- QRectF **rect** ()
- void **setRect** (const QRectF &rect)
- void **setRect** (qreal x, qreal y, qreal w, qreal h)
- qreal **width** () const
- qreal **height** () const
- void **setWidth** (qreal dWidth)
- void **setHeight** (qreal dHeight)
- void setSizeIncrement (int nWidthInc, int nHeightInc)

  0 for no limit, 1 for rounding to nearest int, 4 for coords in multiples of 4
- void setMinSize (qreal dMinWidth, qreal dMinHeight)

  Minimal size of the rectangle.
- QString caption ()

  Caption is text drawn in the center of rectangle.
- void **setCaption** (const QString &sCaption)
- QColor **captionColor** ()
- void **setCaptionColor** (const QColor &color)
- virtual void scaleObject (double sx, double sy, bool bReflectConstraints=true)

  Object scaling.
- virtual void moveObject (double dx, double dy, bool bReflectConstraints=true)

  Object translation.
- virtual void **paint** (QPainter ∗painter, const QStyleOptionGraphicsItem ∗, QWidget ∗)
- virtual QRectF **boundingRect** () const

- ExRectShape **rectShape** () const
- void **setRectShape** (ExRectShape eShape)
- double **cornerRadiusX** () const
- double **cornerRadiusY** () const
- Qt::SizeMode **cornerSizeMode** () const
- void **setCornersRadius** (double dXradius, double dYradius, Qt::SizeMode mode=Qt::AbsoluteSize)
- void **setCornersRadius** (double dRadius, Qt::SizeMode mode=Qt::AbsoluteSize)

Protected Types

- enum **ExRectHitTest** {
  ehtNone, **ehtMove**, **ehtTopEdge**, **ehtLeftEdge**,
  ehtRightEdge, **ehtBottomEdge**, **ehtTLCorner**, **ehtTRCorner**,
  ehtBLCorner, **ehtBRCorner** }

Protected Member Functions

- virtual ExRectHitTest **hitTest** (qreal x, qreal y, qreal dViewScale)
- bool **reflectSizeIncrement** (qreal &w, qreal &h) const
- virtual void **hoverMoveEvent** (QGraphicsSceneHoverEvent ∗event)
- virtual void **mousePressEvent** (QGraphicsSceneMouseEvent ∗event)
- virtual void **mouseMoveEvent** (QGraphicsSceneMouseEvent ∗event)
- virtual void **mouseReleaseEvent** (QGraphicsSceneMouseEvent ∗event)

Protected Attributes

- ExRectShape **m_eShape**
- double **m_dXradius**
- double **m_dYradius**
- Qt::SizeMode **m_eCornerSizeMode**
- qreal **m_dWidth**
- qreal **m_dHeight**
- enum CxVectorObjectRect::ExRectHitTest **m_eLastHitTest**
- int **m_nWidthIncrement**
- int **m_nHeightIncrement**
- QSizeF **m_sizeMin**
- QString **m_sCaption**
- QColor **m_clCaption**
- bool **m_bResizing**
- QPointF **m_ptStartDragMousePos**
- QPointF **m_ptStartRectPos**

4.58.1 Detailed Description

Object representing the rectangular object. Implements mouse resizing and caption.

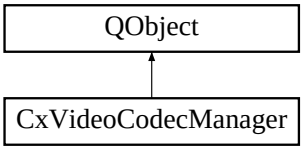
The documentation for this class was generated from the following file:

- VectorObject.h

4.59 CxVideoCodecManager Class Reference

The class that manages video codecs used by video loaders and savers.

Inheritance diagram for CxVideoCodecManager:



Classes

- struct SxCodecItem

Public Member Functions

- IxVideoEncoder * **encoderForFourCC** (TxFourCC fcc)
- IxVideoDecoder * **decoderForFourCC** (TxFourCC fcc)
- QList< QPair< TxFourCC, QString > > encoders ()

  Returns a list of encoders with their FourCCs and names.
- QList< QPair< TxFourCC, QString > > decoders ()

  Returns a list of decoders with their FourCCs and names.
- void setCodecOptions (TxFourCC fccCodec, const TxImageFormatOptions &aOpt)

  set defaults codec
- void saveSettings (QSettings *pSettings)

  store format options to app settings
- void loadSettings (QSettings *pSettings)

  load format options from app settings

Static Public Member Functions

- static CxVideoCodecManager * instance ()

  Singleton class, this is the way to obtain the pointer.

Private Slots

- void onRttiItemEnableStateChanged (TxRttiItem hItem, bool bEnabled)

  Handle the CxRTTI changes.

Private Member Functions

- void **resetEncoders** ()
- void **resetDecoders** ()
- void **loadAvailableEncoders** ()
- void **loadAvailableDecoders** ()

Static Private Member Functions

- static QString **findClassForFourCC** (const QList< SxCodecItem > &lstCodecs, TxFourCC fourCC)

Private Attributes

- bool **m_bEncodersLoaded**
- bool **m_bDecodersLoaded**
- QList< SxCodecItem > **m_lstEncoders**
- QList< SxCodecItem > **m_lstDecoders**
- QMap< TxFourCC, TxImageFormatOptions > m_mapCodecOptions

  maps FourCC of a codec to its options

4.59.1 Detailed Description

The class that manages video codecs used by video loaders and savers.

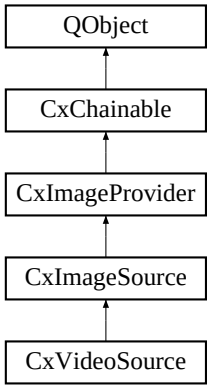
The documentation for this class was generated from the following file:

- VideoCodecManager.h

4.60 CxVideoSource Class Reference

Provides image sequences.

Inheritance diagram for CxVideoSource:



Public Member Functions

- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual CxChainable * clone ()

  Creates a deep copy of this object.
- virtual void run ()

  This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).
- virtual bool saveSettings (QDomDocument &aDoc, QDomElement &aElm) const

  The method saves the settings of this object to the DOM element.

- virtual bool loadSettings (const QDomElement &aDom)

  Loads settings from DOM.
- virtual bool currentOutputImageInfo (SxPicBufInfo &picInfo, CxImageMetadata *pMetadata=NULL)

  Gets the current output image info (format) provided by this object.
- virtual int **framesCount** ()
- virtual void **requestFrame** (int iFrameNo)
- virtual double frameRate ()

  Fps, it defines the duration of the video.
- virtual void **setDataLoader** (IxImageDataLoader *pLoader)

Protected Attributes

- IxImageDataLoader * m_pLoader

  Our source of the images.
- SxPicBufInfo **m_aImgFormat**
- qint32 **m_iFrameCount**
- double **m_dFps**

Additional Inherited Members

4.60.1 Detailed Description

Provides image sequences.

4.60.2 Member Function Documentation

4.60.2.1 virtual bool CxVideoSource::currentOutputImageInfo (SxPicBufInfo & *picInfo,* CxImageMetadata * *pMetadata =* NULL) [virtual]

Gets the current output image info (format) provided by this object.

Remarks

The method should call getCurrentOutputImageInfo() on its precedessors and calculate the output image info provided by this object.

Reimplemented from CxImageProvider.

4.60.2.2 virtual void CxVideoSource::run ( ) [virtual]

This method is invoked automatically by the start() method. If the method is implemented as an infinite loop then it must use the CHAINABLE_RUN_WHILE_TRUE and CHAINABLE_RUN_END_WHILE macros to start and finish the infinite loop. For most of the chainables object it may not be implemented. It should be implemented for image sources (CxImageSource).

See also

CxChainable

Reimplemented from CxChainable.

4.60.2.3 virtual bool CxVideoSource::saveSettings (QDomDocument & *aDoc,* QDomElement & *aElm*) const [virtual]

The method saves the settings of this object to the DOM element.

Remarks

The element `aElm` is already named by a caller, do not change the name (ie. do not call the aElm.setTag↩ Name()). You can add attributes and child elements.

Reimplemented from CxChainable.

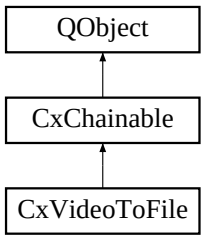
The documentation for this class was generated from the following file:

- VideoSource.h

4.61 CxVideoToFile Class Reference

The chainable class that allows to save images coming from camera to a video file.

Inheritance diagram for CxVideoToFile:



Signals

- void **frameSaved** (int iSeqIndex, quint64 uiTimeStamp)

Public Member Functions

- **CxVideoToFile** (IxImageDataSaver *pVideoSaver)
- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual bool acceptsDataFrom (CxChainable *pPrecedessor)

  Query if this object can accept data from the given chainable object.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual CxChainable * clone ()

  Creates a deep copy of this object.

Protected Slots

- virtual void **onDataAvailable** (CxChainable *pSender, int iFrameNo)

Protected Member Functions

- virtual bool init (QString *pErrMsg=NULL)

  Initializes the object. When calling this method, the object MUST be stopped.
- virtual bool **initialized** ()

Private Attributes

- IxImageDataSaver * **m_pVideoSaver**
- int **m_iSeqIndex**

Additional Inherited Members

4.61.1 Detailed Description

The chainable class that allows to save images coming from camera to a video file.

4.61.2 Member Function Documentation

4.61.2.1 virtual bool CxVideoToFile::init (QString * *pErrMsg* = NULL) [protected], [virtual]

Initializes the object. When calling this method, the object MUST be stopped.

Remarks

Methods init() and initialized() are internally used by the start() method. It should initialize the chainable object so that it is ready to start (start()) . If this object is the CxImageProvider then the method initializes internal memory pool.

Implements CxChainable.

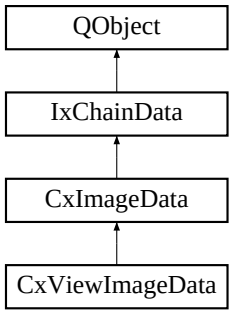
The documentation for this class was generated from the following file:

- VideoToFile.h

4.62 CxViewImageData Class Reference

Implementation of CxImageData intended for displaying in views.

Inheritance diagram for CxViewImageData:



Public Member Functions

- CxViewImageData (CxImageData *pOrigData, SxPicBuf *pViewPicBufRgb32)

  Constructor.

- virtual IxChainData * clone (TxMemPoolHandle hMemPool, QObject *pMemPoolClient) const

  Creates the deep copy of this object. If the `hMemPool` is not NULL and it is a valid memory pool, then the data will be allocated from the memory pool. If the `hMemPool` is NULL then the data will be allocated from heap (operator new).

- virtual bool isFromMemoryPool (TxMemPoolHandle hMemPool) const

  Returns true if the original picbuf (origImageData()->picBuf()) or the current picbuf is allocated from the given memory pool.

- const CxImageData * origImageData () const

  Returns the original image data.

Private Attributes

- CxImageData * m_pOrigData

Additional Inherited Members

4.62.1 Detailed Description

Implementation of CxImageData intended for displaying in views.

Remarks

The class keeps pointer to the original data and to the data converted to RGB32 which can be displayed by QGraphicsView.

4.62.2 Constructor & Destructor Documentation

4.62.2.1 CxViewImageData::CxViewImageData (CxImageData * *pOrigData*, SxPicBuf * *pViewPicBufRgb32*)

Constructor.

Remarks

The object takes ownership of the `pOrigData` and `pViewPicBuf`. If the `pViewPicBuf` is NULL then the object internally uses pOrigData->picBuf() as a view picbuf.

4.62.3 Member Data Documentation

4.62.3.1 CxImageData* CxViewImageData::m_pOrigData [private]

Original image data.

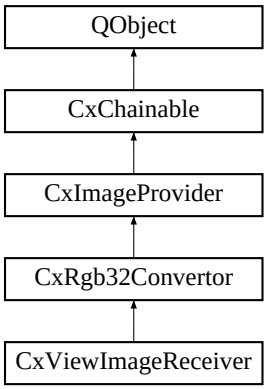
The documentation for this class was generated from the following file:

- ImageData.h

4.63 CxViewImageReceiver Class Reference

This class is primarily used by images views to receive a image data from a chain.

Inheritance diagram for CxViewImageReceiver:



Public Member Functions

- void setRenderingEnabled (bool bEnable)

  Enable rendering in view. Can be disabled when wanting to get higest possible FPS on camera.
- bool isRenderingEnabled ()

  Returns true when rendering is enabled.
- void setForceFpsLimit (bool bForce, int iFpsLimit=-1)

  User-defined fps limit. Limit should be above zero.
- bool **isFpsLimitForced** (int *piFpsLimit=NULL)
- virtual QString title () const

  Gets a title of this chainable object. The title is usually used to display a window title in the GUI etc. It should be a localized string.
- virtual CxChainable * clone ()

  Creates a deep copy of this object.
- virtual int buffersCountInMemoryPool () const

  *Gets a number of buffers which **this** object needs to have reserved in memory pool. If this object allocates no buffers then the method should return 0. See the CxChainable.*
- virtual int queryImagesCountInMemoryPool (const SxPicBufInfo &picInfo, TxMemPoolHandle hPool)

  Gets the required number of images reserved in the memory pool by its successors.
- virtual QWidget * configurationWidget ()

  Displays the object's configuration dialog.
- virtual bool hasConfigurationWidget () const

  Returns true when the the object has configuration widget (ie. the configurationWidget() returns not NULL).
- virtual void setMaxFps (double dFps)

  Sets max. FPS (when zero or negative, then the FPS will not be regulated)

Protected Member Functions

- virtual IxChainData * processData (IxChainData *pReceivedData)

  The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

Protected Attributes

- bool m_bRenderingEnabled

  Enable rendering in view.
- bool m_bForceFpsLimit

  User-set fps limit (when false, our automatics code is used)
- int m_iFpsLimitForced

  User-set fps limit.
- double m_dFpsLimitAutomatic

  Fps limit as defined by CamTool (from monitor frequency)

Additional Inherited Members

4.63.1 Detailed Description

This class is primarily used by images views to receive a image data from a chain.

Remarks

It converts the incoming data to RGB32 and creates an instance of CxViewImageData for the view.

Warning

It is not recommended to use this class in plugins or chainables because the usage of this classe somewhere in the middle of the chain can have a lot of unwanted side effects! Just forget this class exists and always work with CxImageData.

4.63.2 Member Function Documentation

4.63.2.1 virtual QWidget* CxViewImageReceiver::configurationWidget ( ) [virtual]

Displays the object's configuration dialog.

Remarks

The configuration widget should not contain the Ok and Cancel button as they will be added automatically by the application.

Warning

When this chainable object changes the image format, don't forget to stop the current chain before applying new value from the dialog!
This method MUST be called from GUI (main) thread only!!

Returns

The default value is NULL. When overriden then it should return the pointer to the configuration widget.

Reimplemented from CxChainable.

4.63.2.2 virtual IxChainData* CxViewImageReceiver::processData (IxChainData * *pReceivedData*) [protected], [virtual]

The method processes the received data. It is called automatically in the default implementation of the onDataAvailable() slot.

Remarks

If the implementation creates (allocates) a new data here then the original (received) data should be deleted using the delete operator. The default implementation of this method returns a pointer to the received data without any processing.

Warning

If the method returns NULL, then the `pReceivedData` will be deleted automatically!

Reimplemented from CxRgb32Convertor.

4.63.2.3 virtual int CxViewImageReceiver::queryImagesCountInMemoryPool (const SxPicBufInfo & *picInfo,* TxMemPoolHandle *hPool*) [virtual]

Gets the required number of images reserved in the memory pool by its successors.

Remarks

If this object uses its own memory pool then the method allways returns zero!

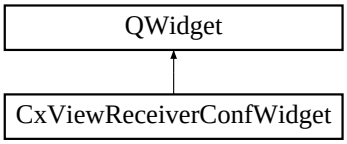
Reimplemented from CxImageProvider.

The documentation for this class was generated from the following file:

- ViewImageReceiver.h

4.64 CxViewReceiverConfWidget Class Reference

Inheritance diagram for CxViewReceiverConfWidget:



Public Member Functions

- **CxViewReceiverConfWidget** (CxViewImageReceiver *pReceiver, QWidget *parent=0)

Private Slots

- void **on_chkRendering_clicked** (bool bChecked)
- void **on_rbFpsAuto_clicked** (bool bChecked)
- void **on_rbFpsForced_clicked** (bool bChecked)
- void **on_spinFps_edited** (int iValue)

Private Attributes

- CxViewImageReceiver * **m_pReceiver**

The documentation for this class was generated from the following file:

- ViewImageReceiver.h

4.65 CxHsiCameraParams::CxVirtualBand Class Reference

< Band record inside <virtual_bands>

Public Member Functions

- bool **isValid** () const
- QString name () const

  Returns virtual band name as "123 nm".

Public Attributes

- double m_dWaveLength

  Central wavelength of the virtual band in nm.
- double m_dFwhm

  Full width of the virtual band at half the band's maximum, in nm.
- QVector< double > m_vecCorrectionData

  Correction data, size is equal to vecBands.count.

4.65.1 Detailed Description

< Band record inside <virtual_bands>

The documentation for this class was generated from the following file:

- HsiCameraParams.h

4.66 IxAppDelegate Class Reference

Class for plugins to communicate with the application. Use IxAppDelegate::instance() to get the pointer.

Public Types

- enum ExSound { eSoundCapture, eSoundRecord }

Public Member Functions

- virtual QMainWindow * mainWindow ()=0

  For adding something to main menu, or anything.
- virtual QToolBar * mainToolbar ()=0

  For adding toolbar buttons.

- virtual QMenu * mainMenu (ExMainMenu eMenu)=0

  For adding actions to main menu.
- virtual QSettings * settings ()=0

  Use for storing plugin settings.
- virtual QString appDataDirectory ()=0

  Store there your internal data, preserved for furute sessions.
- virtual QString lastOpenedDir ()=0

  Directory use last used.
- virtual void setLastOpenedDir (const QString &sDir)=0

  Set directory use last used.
- virtual void addDockedWidget (QWidget *pWidget, Qt::DockWidgetArea eDockArea, Qt::DockWidgetAreas aAllowedAreas, bool bShow)=0

  Add new widget to main window. QDockWidget is created around it and docked in correct place.
- virtual void showNotification (const QString &sMessage)=0

  Popup notification in top right corner.
- virtual CxChain * activeChain ()=0

  Chain of the active view.
- virtual int activeViewId ()=0

  iViewId of the current view
- virtual void activateView (int iViewId)=0

  Let the view become selected.
- virtual void enableGUI (bool bEnable)=0

  Disables or enables application GUI.
- virtual CxChain * loadChainFromXML (const QString &sXml, bool bStart=true, CxImageSource *pImage↩ Source=NULL, QString *pErrMsg=NULL)=0

  Loads chain from XML, attaches all views and starts it. If `pImageSource` is not NULL then the chain will be appended to it and the image source from the XML will be omitted.
- virtual void showChainableConfiguration (CxChainable *pChainable, QWidget *pParent)=0

  Displays a modal configuration dialog for given widget if it exists.
- virtual CxMdiView * viewPtr (int iViewId)=0

  Pointer to view by its id.
- virtual CxChain * viewChain (int iViewId)

  Pointer to chain the view is belonging to.
- virtual bool captureImageInView (int iViewId=-1)=0

  Duplicates the image in active view (as in menu File - Capture)
- virtual bool runAcquisition (bool bStart, XICORE_HANDLE hCamera=NULL)=0

  Start or stop acquisition. Camera can be NULL in case of one camera connected.
- virtual XICORE_HANDLE cameraHandleFromView (int iViewId)=0

  Camera used in the given view (returns NULL when no camera invoved)
- virtual QList< int > viewIdsForCameraHandle (XICORE_HANDLE hCamera)=0

  All views associated with given camera handle.
- virtual SxPicBuf * capturePictureWithGUI (QWidget *pParent, XICORE_HANDLE hCamera, const QString &sWndTitle, const QString &sInstructions, bool bEnableAveraging, int &iAverageCount, CxImageMetadata *pDstMetadata=NULL)=0
- virtual bool openFile (const QString &sFilename, bool bAddToMRU=true)=0

  Open file as new view. Optionally add the file name to recently used list.
- virtual bool createViewWithImage (CxImageData *pImgData, const QString &sTitle)=0

  Creates new view using the image. View takes ownership of the data and will free it when needed.
- virtual const CxImageData * viewCurrentImage (int iViewId)=0

  Image currently displayed in view.
- virtual TxVecObjID addVectorObject (int iViewId, CxVectorObject *pObj)=0

Adds vector object to view. You should store the returned ID instead of object pointer.

- virtual CxVectorObject * vectorObjectPtr (int iViewId, TxVecObjID idObj)=0

  In case you want to manipulate the object directly.
- virtual void removeVectorObject (int iViewId, TxVecObjID idObj)=0

  Deletes the object.
- virtual QVector< TxVecObjID > listAllVectorObjects (int iViewId)=0

  Returns the list of all objects in the scene.
- virtual void overrideViewCursor (int iViewId, QCursor *pCursor)=0

  Pass NULL to stop overriding.
- virtual bool mapPositionToImage (int iViewId, const QPoint &ptScreenPos, QPointF *pPtInImage)=0

  Recalculate to position inside the image. Returns true when inside.
- virtual TxBinaryLayerID addBinaryLayer (int iViewId, CxBinaryLayer *pLayer, bool bAutoColor=true)=0

  Add new binary layer.
- virtual CxBinaryLayer * binaryLayerPtr (int iViewid, TxBinaryLayerID idLayer)=0

  Returns a pointer to the binary layer with given id.
- virtual void removeBinaryLayer (int iViewId, TxBinaryLayerID idLayer)=0

  Deletes the binary layers.
- virtual QList< TxBinaryLayerID > listAllBinaryLayers (int iViewId)=0

  Returns a list of all binary layers in the view.
- virtual QList< TxBinaryLayerID > listVisibleBinaryLayers (int iViewId)=0

  Returns a list of visible binary layers in the view.
- virtual void setCustomPlotColors (QCustomPlot *pPlot)=0

  Sets app-style colors to graph grawing component.
- virtual void playSound (ExSound eSound)=0

  Plays the application predefined sound.
- virtual void playSound (const QString &sFilename)=0

  Plays a sound in WAV format.

Static Public Member Functions

- static IxAppDelegate * instance ()

  Returns the pointer to actual application delegate.
- static QByteArray **fileSystemRepresentation** (const QString &sFilename)
- static QString **stringFromFileSystemRepresentation** (const QByteArray &sEncodedFilename)
- static void setInstance (IxAppDelegate *pInstance)

  Reserved for internal use.

4.66.1 Detailed Description

Class for plugins to communicate with the application. Use IxAppDelegate::instance() to get the pointer.

Warning

This class should always be used in the main (GUI) thread. You should never use it in CxChainable::processData(). In this case the application behaviour is undefined.

4.66.2 Member Enumeration Documentation

4.66.2.1 enum IxAppDelegate::ExSound

Enumerator

eSoundCapture When image is captured or autosaved (instant save).

eSoundRecord When instant video recordint is started or stopped.

4.66.3 Member Function Documentation

4.66.3.1 virtual SxPicBuf* IxAppDelegate::capturePictureWithGUI (QWidget * *pParent*, XICORE_HANDLE *hCamera*, const QString & *sWndTitle*, const QString & *sInstructions*, bool *bEnableAveraging*, int & *iAverageCount*, CxImageMetadata * *pDstMetadata* = NULL) [pure virtual]

Show dialog that lets user to capture a image from camera, returning the picture. Caller must free and delete the picture). If you leave pParent NULL, mainWnd is used as parent.

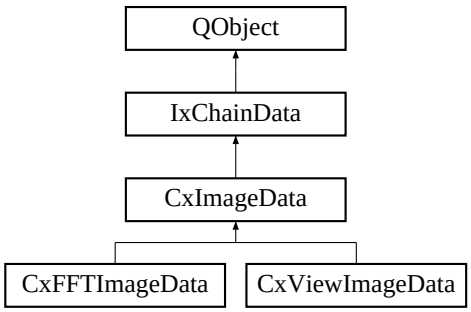
The documentation for this class was generated from the following file:

- AppDelegate.h

4.67 IxChainData Class Reference

The base class for all data exchanged by CxChainable objects.

Inheritance diagram for IxChainData:



Public Member Functions

- virtual qint32 frameNo () const =0

  Gets the frame number of the data. For "static" data (e.g. static image) it is allways zero.

- virtual IxChainData * clone (TxMemPoolHandle hMemPool, QObject *pMemPoolClient) const =0

  Creates the deep copy of this object. If the `hMemPool` is not NULL and it is a valid memory pool, then the data will be allocated from the memory pool. If the `hMemPool` is NULL then the data will be allocated from heap (operator new).

- virtual IxChainData * clone (QObject *pMemPoolClient) const =0

  Creates the deep copy of this object from the same memory source as the original data.

- virtual bool isFromMemoryPool (TxMemPoolHandle hMemPool) const =0

  Checks if the data is from given memory pool.

4.67.1 Detailed Description

The base class for all data exchanged by CxChainable objects.

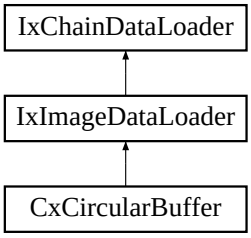
The documentation for this class was generated from the following file:

- Data.h

4.68 IxChainDataLoader Class Reference

The base class for any kind of chain data loading.

Inheritance diagram for IxChainDataLoader:



Public Member Functions

- virtual bool load ()=0

  Loads data from some source.
- virtual qint32 sequenceSize ()=0

  In case the source has multiple data items, their count is returnes here.
- virtual IxChainData ∗ loadedData (qint32 iSeqIdx=0)=0

  Gets the pointer to the current data (after the load() method was called).
- virtual QString inputFile ()=0

  If the loader load data from file then this method returns the the file's name.
- virtual void **setInputFile** (const QString &sFileName)=0

4.68.1 Detailed Description

The base class for any kind of chain data loading.

4.68.2 Member Function Documentation

4.68.2.1 virtual IxChainData∗ IxChainDataLoader::loadedData (qint32 *iSeqIdx* = `0`) `[pure virtual]`

Gets the pointer to the current data (after the load() method was called).

Remarks

The caller of this method is the owner of the returned data so that it has to take care about its deletion using the `delete` operator.

Implemented in CxCircularBuffer.

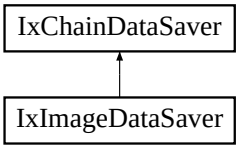
The documentation for this class was generated from the following file:

- DataLoader.h

4.69 IxChainDataSaver Class Reference

The base class for any kind of class for saving IxChainData.

Inheritance diagram for IxChainDataSaver:



Public Member Functions

- virtual void setSequenceSize (qint32 iSeqSize)

  In case the source has multiple data items, we let saver know the number of items.
- virtual bool save (const IxChainData *pData, qint32 iSeqIdx=0)=0

  Saves the current data. May be part of sequence.

4.69.1 Detailed Description

The base class for any kind of class for saving IxChainData.

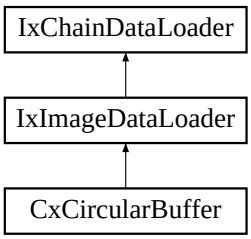
The documentation for this class was generated from the following file:

- DataSaver.h

4.70 IxImageDataLoader Class Reference

The base class for image data loading.

Inheritance diagram for IxImageDataLoader:



Public Member Functions

- virtual double loadedFps ()=0

  In case the source is a sequence, this should get its fps. Return 0 if uncertain.
- virtual TxImageFormatList supportedImageFormats ()=0

  Image formats capable of loading.

4.70.1 Detailed Description

The base class for image data loading.

Remarks

The implementations of this class should be registered by CxRtti so that the application could use them.

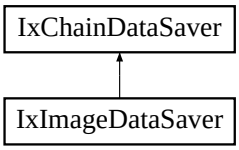
The documentation for this class was generated from the following file:

- DataLoader.h

4.71 IxImageDataSaver Class Reference

The base class for any kind of image data saver.

Inheritance diagram for IxImageDataSaver:



Public Member Functions

- virtual TxImageFormatList supportedImageFormats ()=0

  Image formats capable of saving.
- virtual void **setOutputFile** (const QString &sFileName)=0
- virtual bool **querySavedImageFormat** (const SxPicBufInfo &aInput, SxPicBufInfo &aOutput)
- virtual void **setSaverOptions** (const TxImageFormatOptions &aOpt)
- virtual QWidget * **widgetForSaveDlg** (QWidget *pParent)
- virtual TxImageFormatOptions **optionsFromWidgetInSaveDlg** (QWidget *pWidget)
- virtual TxImageFormatOptions **codecOptionsFromWigetInSaveDlg** (QWidget *pWidget, TxFourCC &fcc↩ Codec)

4.71.1 Detailed Description

The base class for any kind of image data saver.

Remarks

The implementations of this class should be registered by CxRtti so that the application could use them.

The documentation for this class was generated from the following file:

- DataSaver.h

4.72 IxVideoAccessor Class Reference

The class for accessing the raw video data buffers.

Public Member Functions

- virtual qint32 framesCount ()=0

  Number of frames of the video sequence.
- virtual size_t frameDataSize (qint32 iSeqIndex)=0

  Size of the frame with the given index in bytes.
- virtual bool frameData (qint32 iSeqIndex, void *pData, size_t &ruiSize)=0

  Copies the raw frame data with given index to the preallocated `pData`.
- virtual qint32 lastProcessedFrame ()=0

  Returns the index of last processed (decoded) frame.
- virtual void keyFrames (QList< qint32 > &rKeyFrames)=0

  Returns a list of keyframes of the video sequence.

4.72.1 Detailed Description

The class for accessing the raw video data buffers.

Remarks

This class should implemented by video loaders. It is passed to IxVideoDecoder instances to access the video data.

The documentation for this class was generated from the following file:

- VideoAccessor.h

4.73 IxVideoDecoder Class Reference

The base class of all video decoders.

Public Member Functions

- virtual QString name ()=0

  Localized name of the decoder visible in GUI.
- virtual TxFourCC **fourCC** ()=0
- virtual bool decode (qint32 iFrameIndex, IxVideoAccessor *pVideo, SxPicBuf &rDecodedPic)=0

  The FourCC identifier of the codec.

4.73.1 Detailed Description

The base class of all video decoders.

Remarks

The implementations of this class should be registered by CxRtti so that the application could use them.

4.73.2 Member Function Documentation

4.73.2.1 virtual bool IxVideoDecoder::decode (qint32 *iFrameIndex*, IxVideoAccessor * *pVideo*, SxPicBuf & *rDecodedPic*) [pure virtual]

The FourCC identifier of the codec.

The methed decodes a video frame given by `iFrameIndex`.

Remarks

If the `rDecodedPicBuf.m_pData==NULL`, the decoder will allocate it but the image must be initialized correctly (width, height, channels, datatype...)

The documentation for this class was generated from the following file:

- VideoDecoder.h

4.74 IxVideoEncoder Class Reference

The base class of all video encoders.

Public Member Functions

- virtual QString name ()=0

  Localized name name visible in GUI.
- virtual TxFourCC fourCC ()=0

  The FourCC identifier of the codec.
- virtual bool encode (const SxPicBuf &rPic, void ∗pEncoded, size_t &ruiSize, bool ∗pbKeyFrame=NULL)=0

  Encodes next image (rPic) in the stream, output is stored to pEncoded buffer with size ruiSize.
- virtual size_t **maxEncodedSize** (const SxPicBufInfo &rPicInfo)=0
- virtual void queryOuputFormat (const SxPicBufInfo &rInput, SxPicBufInfo &rOutput)=0

  For the given input format it sets the output format.
- virtual void **setEncoderOptions** (const TxImageFormatOptions &aOpt)
- virtual QWidget ∗ **widgetForSaveDlg** (QWidget ∗pParent)
- virtual TxImageFormatOptions **optionsFromWidgetInSaveDlg** (QWidget ∗pWidget)

4.74.1 Detailed Description

The base class of all video encoders.

Remarks

The implementations of this class should be registered by CxRtti so that the application could use them.

4.74.2 Member Function Documentation

4.74.2.1 virtual bool IxVideoEncoder::encode (const SxPicBuf & *rPic,* void ∗ *pEncoded,* size_t & *ruiSize,* bool ∗ *pbKeyFrame* = NULL) [pure virtual]

Encodes next image (rPic) in the stream, output is stored to pEncoded buffer with size ruiSize.

Parameters

pbKeyFrame	is optional. When not NULL and set to true on input, then the encoder should force this image to be a key frame. On the output it gives the information whether image was encoded as the key frame.

The documentation for this class was generated from the following file:

- VideoEncoder.h

4.75 CxHsiCameraParams::SxBandIdx Struct Reference

Public Attributes

- int **zone**
- int **band**

The documentation for this struct was generated from the following file:

- HsiCameraParams.h

4.76 SxCameraInfo Struct Reference

Information about camera device.

Public Attributes

- QString m_sName

  Camera model name (XI_PRM_DEVICE_NAME)
- QString m_sType

  Device type (XI_PRM_DEVICE_TYPE)
- int m_iDeviceModelId

  Camera model ID (XI_PRM_DEVICE_MODEL_ID)
- QString m_sSerial

  Serial number. We expect it to be unique. (XI_PRM_DEVICE_SN)
- QString m_sSensorSerial

  Serial number of the sensor. We expect it to be unique. (XI_PRM_DEVICE_SENS_SN)
- QString m_sUserId

  Camera user id (as set by user, and loaded later opening the camera)
- QString m_sInstancePath

  Instance path of the device. (XI_PRM_DEVICE_INSTANCE_PATH)
- ExColorFilterArray m_eColorFilterArray

  Color filter array type of RAW data (XI_PRM_COLOR_FILTER_ARRAY)

4.76.1 Detailed Description

Information about camera device.

The documentation for this struct was generated from the following file:

- CameraParameters.h

4.77 CxVideoCodecManager::SxCodecItem Struct Reference

Public Attributes

- QString **m_sRttiClass**
- QString m_sName

  Localized name of the codec.
- TxFourCC **m_fcc**

The documentation for this struct was generated from the following file:

- VideoCodecManager.h

4.78 CxCameraParam::SxEnumItem Struct Reference

Struct holding all info for each item of enumerated parameter (ecpEnum)

Public Member Functions

- **SxEnumItem** (const CxValue &vVal, const QString &sCaption)

Public Attributes

- CxValue **m_vValue**
- QString **m_sCaption**

4.78.1 Detailed Description

Struct holding all info for each item of enumerated parameter (ecpEnum)

The documentation for this struct was generated from the following file:

- CameraParameters.h

4.79 CxMemoryManager::SxLimitDef Struct Reference

Public Member Functions

- bool **operator**< (const SxLimitDef &other) const

Public Attributes

- QString **m_sOS**
- QString **m_sBits**
- QMap< QString, QString > **m_mapParams**
- quint64 **m_uiLimitBytes**

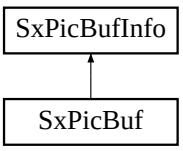
The documentation for this struct was generated from the following file:

- MemoryManager.h

4.80 SxPicBuf Struct Reference

Picture buffer.

Inheritance diagram for SxPicBuf:



Public Types

- enum ExDataOwner { edoNone, edoThisPicBuf, edoMemoryPool }

Public Member Functions

- SxPicBuf ()

  Default constructor, initialies all to zero.
- size_t dataSize () const

Returns the size of pixel data (m_pData).

- void setPicFormat (const SxPicBufInfo &rInfo)

  Sets the widht, height, format accoring to the rInfo parameter.
- bool isFromMemoryPool (TxMemPoolHandle hMemPool) const

  Data comes from some memory pool.
- void setAsShallowCopyOf (const SxPicBuf &rOther)

  Use with caution while hacking. Should not be needed in 99% of cases.
- void copyPicAdditionalData (const SxPicBuf &rOther)

  Copy frame number, time stamps and other info values.

Public Attributes

- void * m_pData
- size_t m_uiAllocSize
- enum SxPicBuf::ExDataOwner **m_eDataOwner**
- TxMemPoolHandle m_hMemoryPoolOwner
- QObject * m_pMemoryPoolClient
- qint32 m_iFrameNo
- quint64 m_uiTimeStamp
- qint32 m_iXiApiFrameNo

Protected Member Functions

- SxPicBuf (const SxPicBuf &rOther)

  Copy constructor is forbidden to use. See operator=.
- SxPicBuf & operator= (const SxPicBuf &rOther)

  operator= is forbidden to use. It is dangerous to copy pointer allocated data.

4.80.1 Detailed Description

Picture buffer.

4.80.2 Member Enumeration Documentation

4.80.2.1 enum SxPicBuf::ExDataOwner

Enumerator

edoNone You should delete the m_pData yourself

edoThisPicBuf Data belong to this picbuf.

edoMemoryPool Data were allocated inside memory pool, m_hMemoryPoolOwner

4.80.3 Member Data Documentation

4.80.3.1 TxMemPoolHandle SxPicBuf::m_hMemoryPoolOwner

Data is part of this memory pool. Do not free.

4.80.3.2 qint32 SxPicBuf::m_iFrameNo

Frame number is set when image comes from camera or an image sequence. Else zero.

4.80.3.3 qint32 SxPicBuf::m_iXiApiFrameNo

Frame number as set by xiApi. May be different to our own counter m_iFrameNo. -1 for undefined.

4.80.3.4 void* SxPicBuf::m_pData

Pointer to pixel data.

4.80.3.5 QObject* SxPicBuf::m_pMemoryPoolClient

Who allocated the buffer in the pool (may have requests).

4.80.3.6 size_t SxPicBuf::m_uiAllocSize

The allocated size of data (i.e. available size)

4.80.3.7 quint64 SxPicBuf::m_uiTimeStamp

The timestamp in micro seconds.

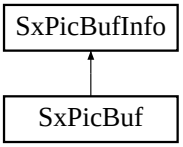
The documentation for this struct was generated from the following file:

- PicBuf.h

4.81 SxPicBufInfo Class Reference

Information about picture buffer.

Inheritance diagram for SxPicBufInfo:



Public Member Functions

- SxPicBufInfo ()
  Default constructor.
- quint32 bytesPerPixel () const
  Returns the number of bytes per pixel.

Public Attributes

- quint32 m_uiWidth
- quint32 m_uiHeight
- quint32 m_uiStride
- quint32 m_uiChannels
- quint32 m_uiBpc
- ExImageDataType m_eDataType

- ExDataStorageFormat m_eFormat
- ExColorFilterArray m_eColorFilterArray

4.81.1 Detailed Description

Information about picture buffer.

4.81.2 Member Data Documentation

4.81.2.1 ExColorFilterArray SxPicBufInfo::m_eColorFilterArray

Bayer pattern

4.81.2.2 ExImageDataType SxPicBufInfo::m_eDataType

Data type of the image.

4.81.2.3 ExDataStorageFormat SxPicBufInfo::m_eFormat

Format of the image.

4.81.2.4 quint32 SxPicBufInfo::m_uiBpc

Bits per channel.

4.81.2.5 quint32 SxPicBufInfo::m_uiChannels

Number of color channels.

4.81.2.6 quint32 SxPicBufInfo::m_uiHeight

Height in pixels.

4.81.2.7 quint32 SxPicBufInfo::m_uiStride

Line width in bytes.

4.81.2.8 quint32 SxPicBufInfo::m_uiWidth

Width in pixels.

The documentation for this class was generated from the following file:

- PicBuf.h

Chapter 5

File Documentation

5.1 Chainable.h File Reference

Classes

- class CxChainable

  A base class of all chainable objects.
- class CxImageProvider

  The base class for any chainable object which provides images.
- class CxImageSource

  The base class of any image provider which is an image source, i.e. it has no precedessors and it provides images.

Macros

- #define CHAINABLE_RUN_WHILE_TRUE

  The macro must be used to start a loop in the CxChainable::run() in a case that the method is implemented as an infinite loop (CxChainable::eRunIsInfiniteLoop).
- #define CHAINABLE_RUN_END_WHILE

  The macro must be used to finish a loop in the CxChainable::run() in a case that the method is implemented as an infinite loop (CxChainable::eRunIsInfiniteLoop).

5.1.1 Macro Definition Documentation

5.1.1.1 #define CHAINABLE_RUN_END_WHILE

Value:

```
}\
    if(__bStateChanged__)\
    {\
       if(__eState__ == eStopped)\
       {\
          LOG_TRACE(ERR_MSG(CxChainable::run(), "Releasing the m_semStartStop (stopped)."))
      ;\
          m_semStartStop.release();\
       }\
    }
```

The macro must be used to finish a loop in the CxChainable::run() in a case that the method is implemented as an infinite loop (CxChainable::eRunIsInfiniteLoop).

See also

CHAINABLE_RUN_WHILE_TRUE

5.1.1.2 #define CHAINABLE_RUN_WHILE_TRUE

Value:

```
Q_ASSERT(m_eFlags.testFlag(eRunIsInfiniteLoop));\
    LOG_TRACE(ERR_MSG(CxChainable::run(), "Releasing the m_semStartStop (started)."));\
    m_semStartStop.release();\
    bool __bStateChanged__ = false;\
    CxChainable::ExChainableState __eState__ = eUndefined;\
    while(true)\
    {\
        {\
           QMutexLocker lock(&m_lock);\
           if(m_eReqState == eStopped)\
           {\
               m_eState = eStopped;\
               __eState__ = m_eState;\
               __bStateChanged__ = true;\
               m_eReqState = eUndefined;\
               break;\
           }\
        }\
        if(chain()->stopped())\
        {\
            continue;\
        }
```

The macro must be used to start a loop in the CxChainable::run() in a case that the method is implemented as an infinite loop (CxChainable::eRunIsInfiniteLoop).

See also

CHAINABLE_RUN_END_WHILE