XISL: X-ray Imaging Software Library

Generated by Doxygen 1.7.6.1

Mon Nov 21 2016 14:43:42

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Module Documentation

3.1 Init Functions for XRpad

Functions

- HIS_RETURN Acquisition_GetVersionInfo (HACQDESC hAcqDesc, XRpad_-VersionInfo ∗versionInfo)

    Acquisition_GetVersionInfo Retrieves version information of the detector.
- HIS_RETURN Acquisition_Set_OnboardOptions (HACQDESC hAcqDesc, BOO-L bEnableAck, BOOL bOffset, BOOL bGain, BOOL bPixel)

    Activate image acquisition options for onboard offset, gain, mean correction and acknowledgement of frames.
- HIS_RETURN Acquisition_Set_OnboardOptionsPostOffset (HACQDESC hAcq-Desc, BOOL bNoOnboardCorr, BOOL bSendPreviewFrist, BOOL bSendFULL-First, BOOL bEnableAckFirst, BOOL bEnableAckSecond, BOOL bEnableOffset-First, BOOL bEnablePostOffsetCorr, BOOL bGain, BOOL bPixel)

    This function defines the image acquisition options for PhotoTimed mode including preview on/off offset/gain/pixel correction on/off and which image will be send to the client.
- HIS_RETURN Acquisition_Set_OnboardOptionsPostOffsetEx (HACQDESC h-AcqDesc, BOOL bNoOnboardCorr, BOOL bSendPreviewFrist, BOOL bSendF-ULLFirst, BOOL bEnableAckFirst, BOOL bEnableAckSecond, BOOL bEnable-OffsetFirst, BOOL bEnablePostOffsetCorr, BOOL bGain, BOOL bPixel, BOOL bStoreOffsetToSD)

    This function can be used to verify the pgototimed mode. It defines the image acquisition options for PhotoTimed mode including preview on/off offset/gain/pixel correction on/off and which image will be send to the client.
- HIS_RETURN Acquisition_wpe_ActivateNetworkConfig (const char ∗ipAddress, int configIndex)

    This function activates the selected network configuration of the detector.
- HIS_RETURN Acquisition_wpe_getAvailableSystems (struct discoveryReply ∗reply, int ∗numDevices, int timeout, int port)

This function sends a network broadcast and retrieves all available XRpad detectors in the Network.

- HIS_RETURN Acquisition_wpe_GetNetworkConfigs (const char ∗ipAddress, struct networkConfiguration ∗configs, int ∗arrayLength, int ∗activeConfig)

  This function retieves the Network settings of a specific Detector defined by its ip address.
- HIS_RETURN Acquisition_wpe_ReadCameraRegisters (const char ∗ipAddress, unsigned long ∗buffer)

  This function retrieves the Status register of a specific Detector defined by its ip address.

3.1.1 Function Documentation

3.1.1.1 HIS_RETURN **Acquisition_GetVersionInfo** (**HACQDESC** ***hAcqDesc,*** **XRpad_VersionInfo** ∗ ***versionInfo***)

Acquisition_GetVersionInfo Retrieves version information of the detector.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*versionInfo*	Pointer to XRpad_VersionInfo structure, which will be filled with the version information.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.1.1.2 HIS_RETURN **Acquisition_Set_OnboardOptions** (**HACQDESC** ***hAcqDesc,*** BOOL ***bEnableAck,*** BOOL ***bOffset,*** BOOL ***bGain,*** BOOL ***bPixel***)

Activate image acquisition options for onboard offset, gain, mean correction and acknowledgement of frames.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
bEnableAck	Enables or disables the Ack mechanism. Image will be stored on sd card for redundance/safety when not acknowledged
bOffset	Enables or disables onboard offset corr.
bGain	Enables or disables onboard gain corr
bPixel	Enables or disables onboard pixel corr

Note

Acq_wpe_LoadCorrectionImageToBuffer must be used to upload the correction data from sd-card to memory

Acquisition_wpe_SetUniqueImageTag must be used to tag the image for acknowledgement
Acquisition_AcknowledgeImage must be used to acknowledge the image if activated. otherwise it will be stored on the sdcard

Returns

An ErroCode is returned when unsuccessfull otherwise HIS_ALL_OK

3.1.1.3 HIS_RETURN Acquisition_Set_OnboardOptionsPostOffset (HACQDESC *hAcqDesc*, BOOL *bNoOnboardCorr*, BOOL *bSendPreviewFrist*, BOOL *bSendFULLFirst*, BOOL *bEnableAckFirst*, BOOL *bEnableAckSecond*, BOOL *bEnableOffsetFirst*, BOOL *bEnablePostOffsetCorr*, BOOL *bGain*, BOOL *bPixel*)

This function defines the image acquisition options for PhotoTimed mode including preview on/off offset/gain/pixel correction on/off and which image will be send to the client.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descripto
	bNoOnboardCorr	if this parameters is set all onboard corrections and preview generation are disabled
	bSendPreviewFrist	if this parameters is set all a preview for the first image is generated and will be send to the client
	bSendFULLFirst	if this parameters is set the first image will be send to the client in full size
	bEnableAckFirst	if this parameters is set first full size image will be stored to sd-card if not acknowledged
	bEnableAckSecond	if this parameters is set second full size image will be stored to sd-card if not acknowledged
	bEnableOffsetFirst	if this parameters is set the first image and the preview image (if selected) will be offset corrected with a predefined offset image
	bEnablePostOffsetCorr	if this parameters is set the first image will be offset corrected with the second image
	bGain	if this parameters is set the images image will be gain corrected with the predefined gain image
	bPixel	if this parameters is set the images image will be pixel corrected with the predefined pixel mask image

Returns

An ErroCode is returned when unsuccessfull otherwise HIS_ALL_OK

∗note the clients image buffer defined using Acquisition_DefineDestBuffers must have the size to retrieve all selected images (max 3xFull size for Preview, Bright, Offset/-Brightoc)

3.1.1.4 HIS_RETURN Acquisition_Set_OnboardOptionsPostOffsetEx (HACQDESC *hAcqDesc*, BOOL *bNoOnboardCorr*, BOOL *bSendPreviewFrist*, BOOL *bSendFULLFirst*, BOOL *bEnableAckFirst*, BOOL *bEnableAckSecond*, BOOL *bEnableOffsetFirst*, BOOL *bEnablePostOffsetCorr*, BOOL *bGain*, BOOL *bPixel*, BOOL *bStoreOffsetToSD*)

This function can be used to verify the pgototimed mode. It defines the image acquisition options for PhotoTimed mode including preview on/off offset/gain/pixel correction on/off and which image will be send to the client.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor
	bNo-Onboard-Corr	if this parameters is set all onboard corrections and preview generation are disabled
	bSend-PreviewFrist	if this parameters is set all a preview for the first image is generated and will be send to the client
	bSendFULL-First	if this parameters is set the first image will be send to the client in full size
	bEnableAck-First	if this parameters is set first full size image will be stored to sd-card if not acknowledged
	bEnableAck-Second	if this parameters is set second full size image will be stored to sd-card if not acknowledged
	bEnable-OffsetFirst	if this parameters is set the first image and the preview image (if selected) will be offset corrected with a predefined offset image
	bEnable-PostOffset-Corr	if this parameters is set the first image will be offset corrected with the second image
	bGain	if this parameters is set the images image will be gain corrected with the predefined gain image
	bPixel	if this parameters is set the images image will be pixel corrected with the predefined pixel mask image
	bStore-OffsetToSD	if this parameters is set second full size image will be stored to sd-card (for debug/test only)

Returns

An ErroCode is returned when unsuccessfull otherwise HIS_ALL_OK

∗note the clients image buffer defined using Acquisition_DefineDestBuffers must have the size to retrieve all selected images (max 3xFull size for Preview, Bright, Offset/-

Brightoc) ∗note this function is for test/debug only. If bStoreOffsetToSD is enabled the other Acknowledge/StoreToSD functions will be disabled

3.1.1.5 HIS_RETURN **Acquisition_wpe_ActivateNetworkConfig** (const char ∗ *ipAddress*, int *configIndex*)

This function activates the selected network configuration of the detector.

Parameters

ipAddress	IP adress of the detectors network interface LAN/WLAN
configIndex	config index to activate

Returns

HIS_ALL_OK function successfull otherwise an error code

3.1.1.6 HIS_RETURN **Acquisition_wpe_getAvailableSystems** (struct **discoveryReply** ∗ *reply*, int ∗ *numDevices*, int *timeout*, int *port*)

This function sends a network broadcast and retrieves all available XRpad detectors in the Network.

Parameters

reply	array of discoveryReply structures
numDevices	pointer to a value containing the number of allocated structures when called
timeout	The timeout shall be greater or erqual to 0 (zero). The physical unit of timeout is ms. The value 0 means to make use of the WPE default timeout of about 2000ms.
port	The port value shall be 0 or 57635. The value 0 means to make use of the default port. The value 57635 is the default port. Actually the PKI detectors do not support other ports than the default port.

The parameter is reserved for future use and exists to kepp the interface unchanged.

Note

Please refer to Acq.h for the parameter definitions. The user has to allocate the discoveryReply array before calling the function.

- numDevices has to contain the number of allocated structures and will contain the retrieved number of devices after the call. An array length of 10 is recommended
- Timeout and port must be 0 to use the predefined default values

Returns

HIS_ALL_OK function successful otherwise an error code

3.1.1.7 HIS_RETURN **Acquisition_wpe_GetNetworkConfigs** (const char ∗ *ipAddress,* struct **networkConfiguration** ∗ *configs,* int ∗ *arrayLength,* int ∗ *activeConfig*)

This function retieves the Network settings of a specific Detector defined by its ip address.

Parameters

ipAddress	IP adress of the detectors LAN/WLAN interface
configs	array of networkConfiguration structures
arrayLength	number of allocated strucures
activeConfig	currently activated configuration

Note

Please refer to Acq.h for the parameter definitions.

- The user has to allocate the networkConfiguration array before calling the function.
- arrayLength has to contain the number of allocated structures and will contain the retieved number of devices after the call. An array length of 20 is recommended
- activeConfig will contain the current activated configuration

Returns

HIS_ALL_OK function successful otherwise an error code

3.1.1.8 HIS_RETURN **Acquisition_wpe_ReadCameraRegisters** (const char ∗ *ipAddress,* unsigned long ∗ *buffer*)

This function retrieves the Status register of a specific Detector defined by its ip address.

Parameters

ipAddress	IP ddress of the detector LAN/WLAN
buffer	Buffer to retrieve the contend of the camera register

Note

Please refer to acq.h for the parameter definitions.

- The user has to allocate the buffer for the register first
- Buffer size has to be EPC_REGISTER_LENGTH

- Buffer will contain Status Register as struct EPC_REGISTER

Returns

HIS_ALL_OK status register could be retrieved otherwise an error code

3.2 Special XRpad Functions

Functions

- HIS_RETURN Acq_wpe_LoadCorrectionImageToBuffer (HACQDESC hAcqDesc, const char ∗pccCorrectionFilePath, ProcScriptOperation Operation)

  Load a correction file from detector SDCARD to a specific correction buffer.
- HIS_RETURN Acq_wpe_SetImageTransferInterface (const char ∗ipAddress, XRpad_DataInterfaceControlEnum eDataInterface)

  This function is used to set the desired detector interface for image transfer.
- HIS_RETURN Acq_wpe_SystemControl (const char ∗ipAddress, XRpad_SystemControlEnum eAction)

  This function is used to control the status of the detector. It can be used e.g. for reboot, shutdown.
- HIS_RETURN Acquisition_AcknowledgeImage (HACQDESC hAcqDesc, const char ∗tag)

  Acquisition_AcknowledgeImage Acknowledges the successful receipt of an image.
- HIS_RETURN Acquisition_AckSDCardForceFsck (HACQDESC hAcqDesc)

  If the filesystem was checked and no errors were found, a message will be send to the client. The reception of the message can be acknowledged/removed with this method. No further message will be send.
- HIS_RETURN Acquisition_AckSDCardForceFsckError (HACQDESC hAcqDesc)

  In the case check of filesystem found errors and fixed them, there will be files like FSCK0000.REC on the SD card and a message will be send to the client. This method won't touch these files. => They have to be handled "manually" through ftp. The reception of the message can be acknowledged with this method. No further message will be send.
- HIS_RETURN Acquisition_CloseFile (XislFileHandle fileHandle)

  Acquisition_CloseFile Closes a file and releases allocated memory.
- HIS_RETURN Acquisition_CreateFakeShockCriticalLevel (HACQDESC hAcqDesc)

  Simulates a shock event at critical level.
- HIS_RETURN Acquisition_CreateFakeShockWarningLevel (HACQDESC hAcqDesc)

  Simulates a shock event at warning level.
- HIS_RETURN Acquisition_DisableEventCallback (HACQDESC hAcqDesc)

  Disconnects the event callback mechanism from from the detector.
- HIS_RETURN Acquisition_Enable_EMI_Data_Readout (HACQDESC hAcqDesc, unsigned int uiOnOff)

  Enables/Disables the EMI data readout and transfer.
- HIS_RETURN Acquisition_FactoryResetShock (HACQDESC hAcqDesc)

  Resets all shock events.
- HIS_RETURN Acquisition_FreeFTPFileBuffer (void ∗pdatabuffer)

  This function to frees a filebuffer allocated by Acquisition_GetFTPFile.
- HIS_RETURN Acquisition_FTP_CloseSession (XislFtpSession session)

Acquisition_FTP_CloseSesion Closes an FTP session.

- HIS_RETURN Acquisition_FTP_InitSession (HACQDESC hAcqDesc, XislFtpSession ∗session)

  Acquisition_FTP_InitSession Initializes a FTP session between client and detector (if supported).

- HIS_RETURN Acquisition_Get_Current_Voltage (HACQDESC hAcqDesc, DETECTOR_CURRENT_VOLTAGE ∗pstructCurrentVoltage)

  Retrieves the internal Currents and Voltages from the detector.

- HIS_RETURN Acquisition_GetAutoPowerOnLocations (HACQDESC hAcqDesc, unsigned int ∗autopoweronlocations)

  Retrieves the current locations from which the detector switches on automatically when the location changed.

- HIS_RETURN Acquisition_GetBatteryStatus (HACQDESC hAcqDesc, XRpad_BatteryStatus ∗batteryStatus)

  Retrieves the battery status.

- HIS_RETURN Acquisition_GetChargeMode (HACQDESC hAcqDesc, unsigned char ∗charge_mode_req, unsigned char ∗charge_mode_charger)

  Retrieves the battery charge mode.

- HIS_RETURN Acquisition_GetChargeModePAcqDesc (PAcquisitionDesc pAcqDesc, unsigned char ∗charge_mode_req, unsigned char ∗charge_mode_charger)

  Retrieves the battery charge mode.

- HIS_RETURN Acquisition_GetFTPFile (const char ∗ipAddress, const char ∗filename, void ∗∗databuffer, long ∗filesize)

  This function retieves data from a file from the detector into a memory buffer.

- HIS_RETURN Acquisition_GetGridSensorStatus (HACQDESC hAcqDesc, unsigned int ∗uiStatus)

  This function retrieves the status of the Grid Sensors.

- HIS_RETURN Acquisition_GetLocation (HACQDESC hAcqDesc, unsigned int ∗location)

  Retrieves the location info of the detector.

- HIS_RETURN Acquisition_GetMissedImageCount (XislFtpSession session, UINT ∗count)

  Acquisition_FTP_GetMissedFileCount Retrieves the count of missed images stored on the detector.

- HIS_RETURN Acquisition_GetNetwork (HACQDESC hAcqDesc, unsigned int ∗network)

  Retrieves the network link speed of the detector.

- HIS_RETURN Acquisition_GetPowerstate (HACQDESC hAcqDesc, unsigned int ∗powerstate)

  Retrieves the current powerstate info of the detector.

- HIS_RETURN Acquisition_GetProvidedEnhancedFeatures (HACQDESC hAcqDesc, unsigned int ∗puiEnhancesFeatures)

  Acquisition_GetProvidedEnhancedFeatures retrieves wether the Detector Provides and Enhanced features.

- HIS_RETURN Acquisition_GetSDCardInfo (HACQDESC hAcqDesc, unsigned int ∗total, unsigned int ∗avail)

  Retrieves the sdcard info about available storage space on the sd-card from the detector.
- HIS_RETURN Acquisition_GetSDCardTimeout (HACQDESC hAcqDesc, unsigned short ∗sdcard_timeout)

  Retrieves the SD card timeout value of the detector.
- HIS_RETURN Acquisition_GetTemperatureThresholds (HACQDESC hAcqDesc, unsigned int ∗threshold_warning, unsigned int ∗threshold_critical)

  Retrieves the warning level and the critical temperature thresholds of the detector.
- HIS_RETURN Acquisition_IdentifyDevice (HACQDESC hAcqDesc)

  trigger device to identify itself by flashing the display
- HIS_RETURN Acquisition_IsPreviewImage (HACQDESC hAcqDesc, unsigned int ∗uiIsPreview)

  This function retrieves the information whether the latest received frame is a preview image.
- HIS_RETURN Acquisition_OpenMissedImage (XislFtpSession session, UINT index, XislFileHandle ∗fileHandle)

  Acquisition_FTP_OpenMissedImage Retrieves a handle of a missed image file.
- HIS_RETURN Acquisition_Resend_All_Messages (HACQDESC hAcqDesc)

  resets all messages to be resend
- HIS_RETURN Acquisition_Reset_OnboardOptions (HACQDESC hAcqDesc)

  Reset all onboard features like preview and corrections.
- HIS_RETURN Acquisition_ResetOnboardShockEvent (HACQDESC hAcqDesc, unsigned int latestShock_Timestamp)

  Resets the onboard shockevent. Log files will not be deleted.
- HIS_RETURN Acquisition_ResetTemperatureTimeout (HACQDESC hAcqDesc)

  Reset the timeout the detector waits before a thermal shutdown.
- HIS_RETURN Acquisition_Set_FPGA_Power_Mode (HACQDESC hAcqDesc, unsigned int uiMode)

  switches the Analog control FPGA On (IDLE or off DEEP_SLEEP) and waits for the device to be ready
- HIS_RETURN Acquisition_Set_OnboardOffsetImageAcquisition (HACQDESC hAcqDesc, BOOL bEnable, BOOL bSend, BOOL bStoreSD)

  Activate onboard offset acquisition to acquire an image into the onboard offset correction buffer.
- HIS_RETURN Acquisition_Set_OnboardOptionPreview (HACQDESC hAcqDesc, BOOL bEnablePreview, BOOL bPreviewOptionSendFull, OnboardBinningMode eMode, unsigned int uiSelectedScript)

  This functions Enables/Disables Onboard Preview generation for single shot acquisition.
- HIS_RETURN Acquisition_SetAutoPowerOnLocations (HACQDESC hAcqDesc, unsigned int autopoweronlocations)

  sets the locations from which the detecor switches on automatically when location changed

- HIS_RETURN Acquisition_SetChargeMode (HACQDESC hAcqDesc, unsigned char charge_mode)

  Sets the battery charge mode.
- HIS_RETURN Acquisition_SetChargeModePAcqDesc (PAcquisitionDesc pAcq-Desc, unsigned char charge_mode)

  Sets the battery charge mode.
- HIS_RETURN Acquisition_SetEventCallback (HACQDESC hAcqDesc, XIS_-EventCallback EventCallback, void ∗userData)

  Sets a callback function to retrieve events.
- HIS_RETURN Acquisition_SetFakeTemperature (HACQDESC hAcqDesc, BOOL bEnableFakeMode, int iFakeTemperature)

  Acquisition_SetFakeTemperature Enables or disables a fake temperature mode of the virtual temperature sensor on XRpad detectors.
- HIS_RETURN Acquisition_SetFTPFile (const char ∗ipAddress, const char ∗filename, void ∗databuffer, long filesize)

  This function stores a databuffer on the sd card folder /mnt/sdcard.
- HIS_RETURN Acquisition_SetIdleTimeout (HACQDESC hAcqDesc, unsigned short timeout)

  This function sets the idle timeout period.
- HIS_RETURN Acquisition_SetNetworkSpeed (HACQDESC hAcqDesc, unsigned int network)

  sets the max speed of the network of the detector
- HIS_RETURN Acquisition_SetPhototimedParams (HACQDESC hAcqDesc, unsigned short usNrOfScrubs, unsigned short usMaxDelay)

  This function configures detector Parameters for Phototimed mode.
- HIS_RETURN Acquisition_SetPrivateKey (HACQDESC hAcqDesc, unsigned char(∗key_old)[64], unsigned char(∗key_new)[64])

  Updates the private key stored on the device.
- HIS_RETURN Acquisition_SetSDCardForceFsck (HACQDESC hAcqDesc)

  Set flag to force check of filesystem on SD card. After setting the flag the detector has to be rebooted to execute the fsck.
- HIS_RETURN Acquisition_SetSDCardTimeout (HACQDESC hAcqDesc, unsigned short sdcard_timeout)

  Sets the timeout after which an unacknowledged image is stored to the SD card.
- HIS_RETURN Acquisition_SetSystemTime (HACQDESC hAcqDesc, char ∗c-DateTime)

  sets the system time of the detector.
- HIS_RETURN Acquisition_SetTailTimeforTriggerMode (HACQDESC hAcqDesc, unsigned short usTailTime, XIS_DetectorTriggerMode eTriggerMode)

  set the tail time in front of the readout in msec
- HIS_RETURN Acquisition_SetTemperatureThresholds (HACQDESC hAcqDesc, unsigned int threshold_warning, unsigned int threshold_critical)

  Sets the warning level and the critical threshold for the surface temperature of the detector.
- HIS_RETURN Acquisition_SetTemperatureTimeout (HACQDESC hAcqDesc, unsigned short timeout)

This function sets the timeout period the detector waits before the thermal shutdown.

- HIS_RETURN Acquisition_VerifyGenuineness (HACQDESC hAcqDesc, char(∗msg)[128], size_t ∗msg_len, unsigned char(∗md)[20])

  Acquisition_VerifyGenuineness Verifies if the device is genuine.
- HIS_RETURN Acquisition_wpe_ChangeNetworkConfig (const char ∗ipAddress, int configIndex, struct networkConfiguration ∗config)

  Change the network configuration via API call.
- HIS_RETURN Acquisition_wpe_FillDefaultNetworkConfiguration (struct network-Configuration ∗config)

  this function will fill a structure with default network settings
- HIS_RETURN Acquisition_wpe_ForceIP (const char ∗macAddress, struct networkConfiguration ∗config, int port, int ∗isAnswered)

  Send a device a "force IP" request as broadcast to be used a temporary IP-adress.
- HIS_RETURN Acquisition_wpe_GetExamFlag (const char ∗ipAddress, unsigned long ∗pExamFlag)

  This function retrieved the status of the exam flag which is set during a running acquisition.
- HIS_RETURN Acquisition_wpe_SetMaxOnboardCorrValue (HACQDESC hAcq-Desc, unsigned short usMax, unsigned short usReplace)

  Set Max Value for onboard corrections.
- HIS_RETURN Acquisition_wpe_SetUniqueImageTag (HACQDESC hAcqDesc, const char ∗imageTag)

  Sets an Image Tag via wpe200 library.

3.2.1 Function Documentation

3.2.1.1 HIS_RETURN Acq_wpe_LoadCorrectionImageToBuffer (HACQDESC *hAcqDesc,* const char ∗ *pccCorrectionFilePath,* ProcScriptOperation *Operation*)

Load a correction file from detector SDCARD to a specific correction buffer.

Parameters

hAcqDesc	handle to the detector
pcc-Correction-FilePath	Full path of the correction file on the SDCARD including /mnt/sdcard
Operation	0-OFFSET 1-GAIN 2-MEAN

Note

The correction will not be enabled with this function.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.2.1.2 HIS_RETURN **Acq_wpe_SetImageTransferInterface** (const char ∗ *ipAddress*, **XRpad_DataInterfaceControlEnum** *eDataInterface*)

This function is used to set the desired detector interface for image transfer.

Parameters

ipAddress	IP-Address of the device to control
eData-Interface	Interface to use (0 - LAN 1 - WLAN)

Note

This function may return -10004 since the device will not be reachable in case of shutdown/reboot
This function overwrites the current settings for the interface temporarily

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.2.1.3 HIS_RETURN **Acq_wpe_SystemControl** (const char ∗ *ipAddress*, **XRpad_SystemControlEnum** *eAction*)

This function is used to control the status of the detector. It can be used e.g. for reboot, shutdown.

Parameters

ipAddress	IP-Address of the device to control
eAction	Action to execute

Note

This function may return -10004 since the device will not be reachable in case of shutdown/reboot
WPE_SYSTEM_CONTROL_REBOOT - requieres AcquisitonClose and re init
XRpad_SYSTEM_CONTROL_SET_DEEP_SLEEP / XRpad_SYSTEM_CONTRO-L_SET_IDLE If switching to deep sleep and switching to idle is used
it has to be checked whether the device is back in IDLE stated by using Acquisition-_GetHWHeader(..) and the image tag must be set to a value <>"". Use -Acquisition_Set_FPGA_Power_Mode to controll DEEPSLEEP<> IDLE instead to ensure device is back to idle.
The function will return HIS_ERROR_ACQ_ALREADY_RUNNING when called during a running acquisition.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.2.1.4 HIS_RETURN Acquisition_AcknowledgeImage (HACQDESC *hAcqDesc*, const char ∗ *tag*)

Acquisition_AcknowledgeImage Acknowledges the successful receipt of an image.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
in	*tag*	The tag of the image to be acknowledged.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.5 HIS_RETURN Acquisition_AckSDCardForceFsck (HACQDESC *hAcqDesc*)

If the filesystem was checked and no errors were found, a message will be send to the client. The reception of the message can be acknowledged/removed with this method. No further message will be send.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.6 HIS_RETURN Acquisition_AckSDCardForceFsckError (HACQDESC *hAcqDesc*)

In the case check of filesystem found errors and fixed them, there will be files like FSCK0000.REC on the SD card and a message will be send to the client. This method won't touch these files. => They have to be handled "manually" through ftp. The reception of the message can be acknowledged with this method. No further message will be send.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.7 HIS_RETURN Acquisition_CloseFile (XislFileHandle *fileHandle*)

Acquisition_CloseFile Closes a file and releases allocated memory.

Parameters

`in`	*fileHandle*	A valid file handle.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Warning

The fileHandle is no longer valid after this operation. Further usage of this handle may lead to undefined behavior.

3.2.1.8 HIS_RETURN Acquisition_CreateFakeShockCriticalLevel (HACQDESC *hAcqDesc*)

Simulates a shock event at critical level.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.9 HIS_RETURN Acquisition_CreateFakeShockWarningLevel (HACQDESC *hAcqDesc*)

Simulates a shock event at warning level.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.10 HIS_RETURN Acquisition_DisableEventCallback (HACQDESC *hAcqDesc*)

Disconnects the event callback mechanism from from the detector.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Note

Beware that the detector will shut down after the configured timout after this disconnect. To re-enable the callback-mechanism, call Acquisition_SetEventCallback again.

Returns

Returns always HIS_ALL_OK.

3.2.1.11 HIS_RETURN Acquisition_Enable_EMI_Data_Readout (HACQDESC *hAcqDesc,* unsigned int *uiOnOff*)

Enables/Disables the EMI data readout and transfer.

Note

Use Acquisition_GbIF_DiscoverDevices in a preceding call to discover the network.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
in	*uiOnOff*	Defines whether the EMI readout shall be enabled 1 or disabled 0

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.2.1.12 HIS_RETURN Acquisition_FactoryResetShock (HACQDESC *hAcqDesc*)

Resets all shock events.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Log file will not be deleted

3.2.1.13 HIS_RETURN Acquisition_FreeFTPFileBuffer (void ∗ *pdatabuffer*)

This function to frees a filebuffer allocated by Acquisition_GetFTPFile.

Parameters

pdatabuffer	Pointer to buffer that should be freed up.

Note

customer/client app has to set the pointer to null

Returns

Allways HIS_ALL_OK

3.2.1.14 HIS_RETURN Acquisition_FTP_CloseSession (XislFtpSession *session*)

Acquisition_FTP_CloseSesion Closes an FTP session.

Parameters

`in`	*session*	A valid FTP session descriptor.

Returns

Always return HIS_ALL_OK-

3.2.1.15 HIS_RETURN Acquisition_FTP_InitSession (HACQDESC *hAcqDesc,* XislFtpSession ∗ *session*)

Acquisition_FTP_InitSession Initializes a FTP session between client and detector (if supported).

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*session*	If the function succeeds, this parameter retrieves a descriptor of the current FTP session.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.16 HIS_RETURN Acquisition_Get_Current_Voltage (HACQDESC *hAcqDesc*, DETECTOR_CURRENT_VOLTAGE ∗ *pstructCurrentVoltage*)

Retrieves the internal Currents and Voltages from the detector.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*pstruct-Current-Voltage*	Will retrieve internal detector Voltages and Currents

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.17 HIS_RETURN Acquisition_GetAutoPowerOnLocations (HACQDESC *hAcqDesc*, unsigned int ∗ *autopoweronlocations*)

Retrieves the current locations from which the detector switches on automatically when the location changed.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*au-topoweron-locations*	Will retrieve the bitwise coded location for auto power on from the detector

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.18 HIS_RETURN Acquisition_GetBatteryStatus (HACQDESC *hAcqDesc*, XRpad_BatteryStatus ∗ *batteryStatus*)

Retrieves the battery status.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*battery-Status*	Pointer to an XRpad_BatteryStatus structure to retrieve the battery status.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.19 HIS_RETURN **Acquisition_GetChargeMode** (**HACQDESC** *hAcqDesc*, unsigned char ∗ *charge_mode_req*, unsigned char ∗ *charge_mode_charger*)

Retrieves the battery charge mode.

The charge_mode_req requested by the software defines the max charge mode The charge_mode_charger defines the max charge allowed by the charger.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*charge_-mode_req*	The value of charge mode requested by the software, 0..3 0 no charging, 3 max charging
out	*charge_-mode_-charger*	The value of charge mode set by the charger, 0..3 0 no charging, 3 max charging

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.20 HIS_RETURN **Acquisition_GetChargeModePAcqDesc** (PAcquisitionDesc *pAcqDesc*, unsigned char ∗ *charge_mode_req*, unsigned char ∗ *charge_mode_charger*)

Retrieves the battery charge mode.

The charge_mode_req requested by the software defines the max charge mode The charge_mode_charger defines the max charge allowed by the charger.

Parameters

in	*pAcqDesc*	pointer to a valid Acquisition Descriptor.
out	*charge_-mode_req*	The value of charge mode requested by the software, 0..3 0 no charging, 3 max charging
out	*charge_-mode_-charger*	The value of charge mode set by the charger, 0..3 0 no charging, 3 max charging

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.21 HIS_RETURN **Acquisition_GetFTPFile** (const char ∗ *ipAddress*, const char ∗ *filename*, void ∗∗ *databuffer*, long ∗ *filesize*)

This function retieves data from a file from the detector into a memory buffer.

Parameters

ipAddress	Pointer to char array of ip address
filename	Pointer to value to retrieve the actual Field Of View mode.
databuffer	Pointer to databuffer
filesize	Pointer to long value of retrieved filesize

Note

default directory is /mnt/sdcard/

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.2.1.22 HIS_RETURN **Acquisition_GetGridSensorStatus** (HACQDESC *hAcqDesc*, unsigned int ∗ *uiStatus*)

This function retrieves the status of the Grid Sensors.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*uiStatus*	pointer to a unsigned int value to retrieve the current status

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

0 - No Grid 1 - Grid Type1 2 - Grid Type2 3 - Grid Type3

3.2.1.23 HIS_RETURN **Acquisition_GetLocation** (HACQDESC *hAcqDesc*, unsigned int ∗ *location*)

Retrieves the location info of the detector.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*location*	Will retrieve location of detector

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.24 HIS_RETURN **Acquisition_GetMissedImageCount** (**XislFtpSession** *session,* UINT ∗ *count*)

Acquisition_FTP_GetMissedFileCount Retrieves the count of missed images stored on the detector.

Parameters

in	*session*	A valid FTP session descriptor.
out	*count*	If the function succeeds, this parameter retrieves the count of missed images stored on the detector.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.25 HIS_RETURN **Acquisition_GetNetwork** (**HACQDESC** *hAcqDesc,* unsigned int ∗ *network*)

Retrieves the network link speed of the detector.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*network*	will retrieve network link speed of detector

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.26 HIS_RETURN **Acquisition_GetPowerstate** (**HACQDESC** *hAcqDesc,* unsigned int ∗ *powerstate*)

Retrieves the current powerstate info of the detector.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*powerstate*	Will retrieve powerstate of the detector

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

powerstate 4 - IDLE; 3 - DEEP SLEEP

3.2.1.27 HIS_RETURN Acquisition_GetProvidedEnhancedFeatures (HACQDESC *hAcqDesc*, unsigned int ∗ *puiEnhancesFeatures*)

Acquisition_GetProvidedEnhancedFeatures retrieves wether the Detector Provides and Enhanced features.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*pui-Enhances-Features*	pointer to a vaule retrieving the whether enhanced modes are provided

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.28 HIS_RETURN Acquisition_GetSDCardInfo (HACQDESC *hAcqDesc*, unsigned int ∗ *total*, unsigned int ∗ *avail*)

Retrieves the sdcard info about available storage space on the sd-card from the detector.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*total*	Will retrieve total storage in MB
out	*avail*	Will retrieve available storage in MB

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.29 HIS_RETURN Acquisition_GetSDCardTimeout (HACQDESC *hAcqDesc*, unsigned short ∗ *sdcard_timeout*)

Retrieves the SD card timeout value of the detector.

See Acquisition_SetSDCardTimeout for more information.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*sdcard_-timeout*	Pointer to the variable to retrieve SD card timeout value in seconds.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.30 HIS_RETURN Acquisition_GetTemperatureThresholds (HACQDESC *hAcqDesc*, unsigned int ∗ *threshold_warning*, unsigned int ∗ *threshold_critical*)

Retrieves the warning level and the critical temperature thresholds of the detector.

See Acquisition_SetTemperatureThresholds for more information.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*threshold_-warning*	Pointer to the variable to retrieve the warning level temperature in millidegree Celsius.
`out`	*threshold_-critical*	Pointer to the variable to retrieve the critical temperature threshold in millidegree Celsius.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.31 HIS_RETURN Acquisition_IdentifyDevice (HACQDESC *hAcqDesc*)

trigger device to identify itself by flashing the display

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.32 HIS_RETURN **Acquisition_IsPreviewImage** (**HACQDESC** ***hAcqDesc,*** **unsigned int** ∗ ***uiIsPreview***)

This function retrieves the information whether the latest received frame is a preview image.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`out`	*uiIsPreview*	Flag whether the imasge was a preview - 1 otherwise - 0

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

This function must be called in the EndframeCallback since the information whether the image is a preview will be overwritten when the next image was retrieved

3.2.1.33 HIS_RETURN **Acquisition_OpenMissedImage** (**XislFtpSession** ***session,*** **UINT** ***index,*** **XislFileHandle** ∗ ***fileHandle***)

Acquisition_FTP_OpenMissedImage Retrieves a handle of a missed image file.

Parameters

`in`	*session*	A valid FTP session descriptor.
`in`	*index*	Index between 0 and (count - 1), where count is retrieved by Acquisition_FTP_GetMissedImageCount.
`out`	*fileHandle*	If the function succeeds, this parameter retrieves the handle of the missed image file.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Warning

This function allocates a memory buffer to carry the whole data of the file. Call Acquisition_CloseFile to release this memory. Otherwise it will leak.

3.2.1.34 HIS_RETURN **Acquisition_Resend_All_Messages** (**HACQDESC** ***hAcqDesc***)

resets all messages to be resend

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.35 HIS_RETURN **Acquisition_Reset_OnboardOptions (HACQDESC *hAcqDesc*)**

Reset all onboard features like preview and corrections.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.36 HIS_RETURN **Acquisition_ResetOnboardShockEvent (HACQDESC *hAcqDesc*, unsigned int *latestShock_Timestamp*)**

Resets the onboard shockevent. Log files will not be deleted.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
	latestShock-_Timestamp	has to be the timestamp of the latest shock that is going to be resetted. to avoid a race condition

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.37 HIS_RETURN **Acquisition_ResetTemperatureTimeout (HACQDESC *hAcqDesc*)**

Reset the timeout the detector waits before a thermal shutdown.

If the temperature of the detector is above the critical threshold, it waits for a predefined time before it shuts down. This shutdown timeout is reset with this function. You can use Acquisition_SetTemperatureTimeout to set the timeout value.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.38 HIS_RETURN **Acquisition_Set_FPGA_Power_Mode** (**HACQDESC** *hAcqDesc*, unsigned int *uiMode*)

switches the Analog control FPGA On (IDLE or off DEEP_SLEEP) and waits for the device to be ready

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*uiMode*	0 - Off 1 - On

Returns

Returns HIS_ALL_OK on success or an appropriate error code including HIS_ERROR_NO_FPGA_ACK otherwise.

Note

Currenlty only valid for network connected detectors

3.2.1.39 HIS_RETURN **Acquisition_Set_OnboardOffsetImageAcquisition** (**HACQDESC** *hAcqDesc*, BOOL *bEnable*, BOOL *bSend*, BOOL *bStoreSD*)

Activate onboard offset acquisition to acquire an image into the onboard offset correction buffer.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
bEnable	Enable Store next image to onboard offset correction buffer
bSend	Send out the offset image to client
bStoreSD	Store the image to SDcard when not acknowledged

Returns

An ErroCode is returned when unsuccessfull

3.2.1.40 HIS_RETURN **Acquisition_Set_OnboardOptionPreview** (**HACQDESC** *hAcqDesc*, BOOL *bEnablePreview*, BOOL *bPreviewOptionSendFull*, **OnboardBinningMode** *eMode*, unsigned int *uiSelectedScript*)

This functions Enables/Disables Onboard Preview generation for single shot acquisition.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
bEnable-Preview	Enables or disables the preview option. Binned Image will be send in front of the full size image for preview.
bPreview-OptionSend-Full	Enables or disables the send Fullsize Image option.
eMode	Selected Binning mode for Preview
uiSelected-Script	Must be 0 in current implementation

Returns

An ErroCode is returned when unsuccessfull

Note

If Preview is enabled the destination buffer must have the size of 2 full size images

3.2.1.41 HIS_RETURN Acquisition_SetAutoPowerOnLocations (HACQDESC *hAcqDesc*, unsigned int *autopoweronlocations*)

sets the locations from which the detecor switches on automatically when location changed

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*au-topoweron-locations*	Bitmask to define the locations

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

locations are ored location0 - 1, location1 - 2, location3 - 4, .. , location7 - 128; all on 255

3.2.1.42 HIS_RETURN Acquisition_SetChargeMode (HACQDESC *hAcqDesc*, unsigned char *charge_mode*)

Sets the battery charge mode.

To avoid charging influences on the image quality and to reduce heating up of the device the charge mode can be modified.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*charge_-mode*	The value of charge mode, 0..3 0 no charging, 3 max charging

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.43 HIS_RETURN Acquisition_SetChargeModePAcqDesc (PAcquisitionDesc *pAcqDesc,* unsigned char *charge_mode*)

Sets the battery charge mode.

To avoid charging influences on the image quality and to reduce heating up of the device the charge mode can be modified.

Parameters

`in`	*pAcqDesc*	pointer to a valid Acquisition Descriptor.
`in`	*charge_-mode*	The value of charge mode, 0..3 0 no charging, 3 max charging

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.44 HIS_RETURN Acquisition_SetEventCallback (HACQDESC *hAcqDesc,* XIS_EventCallback *EventCallback,* void ∗ *userData*)

Sets a callback function to retrieve events.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*Event-Callback*	Pointer to the callback function which retrieves the events.
	userData	pointer to the user data

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.45 HIS_RETURN Acquisition_SetFakeTemperature (HACQDESC *hAcqDesc*, BOOL *bEnableFakeMode*, int *iFakeTemperature*)

Acquisition_SetFakeTemperature Enables or disables a fake temperature mode of the virtual temperature sensor on XRpad detectors.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*bEnable-FakeMode*	TRUE enables the fake mode, FALSE disables it.
`in`	*iFake-Temperature*	Temperature to set in 1/1000 degree Celsius. (42500 == 42.5 C). This parameter is ignored if bEnableFakeMode is FALSE.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.46 HIS_RETURN Acquisition_SetFTPFile (const char ∗ *ipAddress*, const char ∗ *filename*, void ∗ *databuffer*, long *filesize*)

This function stores a databuffer on the sd card folder /mnt/sdcard.

Parameters

ipAddress	Pointer to char array of ip address
filename	Pointer to value to retrieve the actual Field Of View mode.
databuffer	Pointer to databuffer
filesize	Pointer to long value of uploaded filesize

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.2.1.47 HIS_RETURN Acquisition_SetIdleTimeout (HACQDESC *hAcqDesc*, unsigned short *timeout*)

This function sets the idle timeout period.

To reduce power consumption and extend battery lifetime, the XRpad detectors shut down automatically after 10 minutes in idle state.

Use this function to adjust the idle-timeout period, after which the automatic shutdown is initiated.

Note

The shutdown timeout is NOT permanently stored and must be (re-)set after every (re-)boot of the detector.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*timeout*	The value of the idle timeout period in seconds. The timeout value must be at least 20 seconds.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.48 HIS_RETURN Acquisition_SetNetworkSpeed (HACQDESC *hAcqDesc,* unsigned int *network*)

sets the max speed of the network of the detector

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*network*	Will provide network of detector

Note

this may take several seconds to settle

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.49 HIS_RETURN Acquisition_SetPhototimedParams (HACQDESC *hAcqDesc,* unsigned short *usNrOfScrubs,* unsigned short *usMaxDelay*)

This function configures detector Parameters for Phototimed mode.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*usNrOf-Scrubs*	after bright image readout before offset readout (default is 4 min 1)
`in`	*usMaxDelay*	Max Exposure Delay in millisceonds (default is 5000)

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

currently only available fro XRpad2

3.2.1.50 HIS_RETURN Acquisition_SetPrivateKey (HACQDESC *hAcqDesc,* unsigned char(∗) *key_old[64],* unsigned char(∗) *key_new[64]*)

Updates the private key stored on the device.

The private key must be kept secret to ensure genuineness. It must have exactly 512 bits ($\hat{=}$ 64 bytes). Note that it is transmitted unencrypted. Please note that you need the current private key as authorization. This implies that you will not be able to update the key, once you loose your current one.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
in	*key_old*	Pointer to an array containing the old key in raw binary data.
in	*key_new*	Pointer to an array containing the new key in raw binary data.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

currently only available with XRpad2

3.2.1.51 HIS_RETURN Acquisition_SetSDCardForceFsck (HACQDESC *hAcqDesc*)

Set flag to force check of filesystem on SD card. After setting the flag the detector has to be rebooted to execute the fsck.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.52 HIS_RETURN Acquisition_SetSDCardTimeout (HACQDESC *hAcqDesc*, unsigned short *sdcard_timeout*)

Sets the timeout after which an unacknowledged image is stored to the SD card.

The timeout applies only to images that have to be acknowledged by the application. When the image is stored, the client is notified by an event as far an event- callback function is specified by using Acquisition_SetEventCallback. An image may be stored before this timeout is reached under the following circumstances:

- There are more than two internal buffers already in use. The oldest image is then stored to SD, automatically
- The handle was closed using Acquisition_Close or Acquisition_CloseAll
- The connection to the detector was lost/timed out

  This might happen

  1. if the detector shuts down for any reason
  2. if the client application shuts down for any reason

     Note

     Please note that, due to SD card performance, there is some delay between the timeout and the moment the file is fully written.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*sdcard_-timeout*	SD card timeout value in seconds.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.53 HIS_RETURN Acquisition_SetSystemTime (HACQDESC *hAcqDesc*, char ∗ *cDateTime*)

sets the system time of the detector.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*cDateTime*	Pointer to a datetime string

format: YYYY.MM.DD-hh:mm:ss (24h format), the timezone of the detector is always UTC

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.54 HIS_RETURN Acquisition_SetTailTimeforTriggerMode (HACQDESC *hAcqDesc*, unsigned short *usTailTime*, XIS_DetectorTriggerMode *eTriggerMode*)

set the tail time in front of the readout in msec

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
in	*usTailTime*	Tail time to set in msec
in	*eTrigger-Mode*	TriggerMode for which the tail time shall be set

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Default for Framewise is 0msec deafult for DDD, AED, Phototimed is 20msec. Only availbale for XRpad detectors

3.2.1.55 HIS_RETURN Acquisition_SetTemperatureThresholds (HACQDESC *hAcqDesc*, unsigned int *threshold_warning*, unsigned int *threshold_critical*)

Sets the warning level and the critical threshold for the surface temperature of the detector.

The surface temperature of the detector is determined by several temperature sensors inside the device. Use this function to set the threshold to set the warning level and the critical threshold.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
in	*threshold_-warning*	Warning temperature in milldegree celsius. If the surface temperature is above this temperatur, the API will trigger a warning event.
in	*threshold_-critical*	Critical temperature in milldegree celsius. If the surface temperature is above this temperatur, the detector will shut down after a predefined timeout.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.56 HIS_RETURN Acquisition_SetTemperatureTimeout (HACQDESC *hAcqDesc*, unsigned short *timeout*)

This function sets the timeout period the detector waits before the thermal shutdown.

If the temperature of the detector is above the critical threshold, it waits for a predefined time before it shuts down. This shutdown timeout might be reset by using Acquisition_-ResetTemperatureTimeout or is always resetted if an image is acquired.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
in	*timeout*	Timeout value in seconds. A value less than 20 seconds is not permitted.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.57 HIS_RETURN Acquisition_VerifyGenuineness (HACQDESC *hAcqDesc*, char(∗) *msg[128]*, size_t ∗ *msg_len*, unsigned char(∗) *md[20]*)

Acquisition_VerifyGenuineness Verifies if the device is genuine.

The genuineness is verified by utilizing the HMAC-SHA1 algorithm. The host name of the device serves as message and is delivered as output parameter msg. The key is by default an array of 64 bytes of zeros. The key might be changed by using Acquisition_-SetPrivateKey.

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
out	*msg*	Pointer to an int[128] array. Retrieves the message. This is equal to the host name of the device. This string is NULL-terminated. The terminating NULL-byte itself is not included in the calculation of the hash.
out	*msg_len*	The message length.
out	*md*	The message digest (hash) of msg as described above in raw binary data.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.58 HIS_RETURN **Acquisition_wpe_ChangeNetworkConfig** (const char ∗ *ipAddress,* int *configIndex,* struct **networkConfiguration** ∗ *config*)

Change the network configuration via API call.

Parameters

ipAddress	ID address of the detector
configIndex	Configuration index to change
config	networkConfiguration structure to store

Note

config (networkConfiguration) has some readonly members, please check reference for that!!

Returns

returns HIS_ALL_OK when successfull, An ErroCode is returned when unsuccessfull

3.2.1.59 HIS_RETURN **Acquisition_wpe_FillDefaultNetworkConfiguration** (struct **networkConfiguration** ∗ *config*)

this function will fill a structure with default network settings

Parameters

config	pointer to the "struct networkConfiguration" to use.

Returns

HIS_ALL_OK if ok or a error code.
The error code is a HIS_ERROR_... or the abs of an WPE_ERR_... code.

3.2.1.60 HIS_RETURN **Acquisition_wpe_ForceIP** (const char ∗ *macAddress,* struct **networkConfiguration** ∗ *config,* int *port,* int ∗ *isAnswered*)

Send a device a "force IP" request as broadcast to be used a temporary IP-adress.

Parameters

macAddress	the devices MAC address using the format "00:0A:35:11:DE:D1"
config	pointer to the "struct networkConfiguration" to use.
port	The port number to use, if 0, the default port is used.
isAnswered	return 1 if the request was answered or 0 if not

Note

config (networkConfiguration) has some readonly members, please check reference for that!!

Returns

HIS_ALL_OK if ok or a negative error code

3.2.1.61 HIS_RETURN **Acquisition_wpe_GetExamFlag** (const char ∗ *ipAddress*, unsigned long ∗ *pExamFlag*)

This function retrieved the status of the exam flag which is set during a running acquisition.

Parameters

ipAddress	IP-Address of the device to control
pExamFlag	pointer to an unsigned long value to retrieve the exam flag status (1 - exam ongoing; any other - no exam ongoingn

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.2.1.62 HIS_RETURN **Acquisition_wpe_SetMaxOnboardCorrValue** (HACQDESC *hAcqDesc*, unsigned short *usMax*, unsigned short *usReplace*)

Set Max Value for onboard corrections.

Parameters

`in`	*hAcqDesc*	Handle of a valid Acquisition Descriptor.
`in`	*usMax*	Max Values for corrections of current value exeedes usMax its replaced by usReplace
`in`	*usReplace*	used to replace values $>$ usMax

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.2.1.63 HIS_RETURN **Acquisition_wpe_SetUniqueImageTag** (HACQDESC *hAcqDesc*, const char ∗ *imageTag*)

Sets an Image Tag via wpe200 library.

If available for this detector, an image tag is set for the next image. The tag is used for the Ack-Mechanism, available via Acquisition_Set_OnboardOptions.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
imageTag	Image tag to be used as filename for autosave when not acknowledged and to acknowledge the image

Returns

An ErroCode is returned when unsuccessfull

Warning

Always use a unique identifier as image tag. If the image tag is ambiguous, the image may not be explicit identified and may remain on the SD Card storage of the detector. The SD Card might run out of memory in that case.

3.3 Common Init functions

Functions

- HIS_RETURN Acquisition_Close (HACQDESC hAcqDesc)

  Hardware and XISL are closed by this routine. The acquisition descriptor is no longer valid.
- HIS_RETURN Acquisition_CloseAll ()

  This function closes / shuts down all connections to detectors via frame grabbers and IP connections currently allocated by the XISL. All acquisition descriptor structures returned by other functions are invalid after this function call.
- HIS_RETURN Acquisition_EnableLogging (BOOL onOff)

  The purpose of this function is to toggle (enable/disable) the internal logging of the Xisl.
- HIS_RETURN Acquisition_EnumSensors (UINT ∗pdwNumSensors, BOOL b-EnableIRQ, BOOL bAlwaysOpen)

  This function enumerates all currently connected sensors. All recognized sensors are initialized automatically. To get the HACQDESC of every sensor, use Acquisition_-GetNextSensor. For a programming example see the initialization part of the XIS-L demonstration. Note: In case of Network/IP Sensors only sensors with standart-gateway equal to zero are initialized automatically.
- HIS_RETURN Acquisition_GetCommChannel (HACQDESC hAcqDesc, UINT ∗pdwChannelType, int ∗pnChannelNr)

  This function returns the type of the communication device that is used to transfer data from the detector into the PC RAM.
- HIS_RETURN Acquisition_GetConfiguration (HACQDESC hAcqDesc, UINT ∗dwFrames, UINT ∗dwRows, UINT ∗dwColumns, UINT ∗dwDataType, UINT ∗dwSortFlags, BOOL ∗bIRQEnabled, DWORD ∗dwAcqType, DWORD ∗dw-SystemId, DWORD ∗dwSyncMode, DWORD ∗dwHwAccess)

  This function retrieves all important acquisition parameters, that can be set by -Acquisition_Init or that are set by the self configuration mechanisms of the XISL.
- HIS_RETURN Acquisition_GetHwHeaderInfo (HACQDESC hAcqDesc, CHw-HeaderInfo ∗pInfo)

  This function returns the contents of the camera's hardware header in a CHwHeader-Info structure.
- HIS_RETURN Acquisition_GetHwHeaderInfoEx (HACQDESC hAcqDesc, CHw-HeaderInfo ∗pInfo, CHwHeaderInfoEx ∗pInfoEx)

  This function acquires the frame header of the connected detector. If dwHeaderID in the CHwHeaderInfo structure is 14 (1621detectors) pInfoEx will retrieve the extended header, otherwise the structure will be filled with 0xFFFF.
- HIS_RETURN Acquisition_GetIntTimes (HACQDESC hAcqDesc, double ∗dblInt-Time, int ∗nIntTimes)

  This function retrieves the current integration times.
- HIS_RETURN Acquisition_GetNextSensor (ACQDESCPOS ∗Pos, HACQDESC ∗phAcqDesc)

  You can use this function to iterate through all recognized sensors in the system. for a programming example see the initialization part of the XISL demonstration.

- HIS_RETURN Acquisition_Init (HACQDESC ∗phAcqDesc, DWORD dwBoardType, int nChannelNr, BOOL bEnableIRQ, UINT Rows, UINT Columns, UINT dwSortFlag, BOOL bSelfInit, BOOL bAlwaysOpen)

  The Acquisition_Init function initializes the frame grabber board and the corresponding driver. It enables desired hardware interrupts, prepares acquisition threads, defines callback functions to react on acquisition status changes and tests for sufficient memory space for DMA (direct memory access).
- HIS_RETURN Acquisition_SetCallbacksAndMessages (HACQDESC hAcqDesc, HWND hWnd, UINT dwErrorMsg, UINT dwLoosingFramesMsg, void(CALLBACK ∗lpfnEndFrameCallback)(HACQDESC), void(CALLBACK ∗lpfnEndAcqCallback)(HACQDESC))

  The Acquisition_SetCallbacksAndMessages function defines callback functions to react on acquisition status changes. For a programming example see the initialization part of the XISL demonstration.
- HIS_RETURN Acquisition_SetLogLevel (XislLoggingLevels xislLogLvl)

  This function is to set the logging level of the Xisl logging.
- ∗∗∗∗∗∗HIS_RETURN Acquisition_SetLogOutput (const char ∗filePath, BOOL consoleOnOff)

  ∗∗/
- HIS_RETURN Acquisition_TogglePerformanceLogging (BOOL onOff)

  This function is used to toggle the performance logging. The performance logged will measure the time it takes for a particular function call to execute.

3.3.1 Function Documentation

3.3.1.1 HIS_RETURN Acquisition_Close (HACQDESC *hAcqDesc*)

Hardware and XISL are closed by this routine. The acquisition descriptor is no longer valid.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.2 HIS_RETURN Acquisition_CloseAll ( )

This function closes / shuts down all connections to detectors via frame grabbers and IP connections currently allocated by the XISL. All acquisition descriptor structures returned by other functions are invalid after this function call.

Returns

Value If the function is successful it returns zero, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.3 HIS_RETURN **Acquisition_EnableLogging** (BOOL *onOff*)

The purpose of this function is to toggle (enable/disable) the internal logging of the Xisl.

Parameters

onOff	Boolean value used to determine whether to turn internal logging on or off

Returns

If the function is successful it returns zero, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.4 HIS_RETURN **Acquisition_EnumSensors** (UINT ∗ *pdwNumSensors,* BOOL *bEnableIRQ,* BOOL *bAlwaysOpen*)

This function enumerates all currently connected sensors. All recognized sensors are initialized automatically. To get the HACQDESC of every sensor, use Acquisition_Get-NextSensor. For a programming example see the initialization part of the XISL demonstration. Note: In case of Network/IP Sensors only sensors with standart-gateway equal to zero are initialized automatically.

Parameters

pdwNum-Sensors	Address of a 4 byte integer that receives the number of recognized sensors.
bEnableIRQ	If you want to run the acquisition in polling mode set this parameter to zero. If you want to enable hardware interrupts set the parameter to one.
bAlways-Open	If this parameter is TRUE the XISL is capturing all communication port regardless if this port is already opened by other processes running on the system. The use of this option is only recommended in debug versions of your applications because it is not possible to free all system resources allocated by another process.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.5 HIS_RETURN Acquisition_GetCommChannel (HACQDESC *hAcqDesc,* UINT ∗ *pdwChannelType,* int ∗ *pnChannelNr*)

This function returns the type of the communication device that is used to transfer data from the detector into the PC RAM.

Parameters

hAcqDesc	Pointer to HACQDESC.
pdw-Channel-Type	Address of a 4 byte integer that receives an id of the currently open communication device.
pnChannel-Nr	Address of a 4 byte integer that receives the number of communication channel. If the above mentioned communication device is a frame grabber this number is unique to identify the grabber if more than one grabber of one type is installed on the system. (see frame grabber installation description). If the communication device is an RS232 interface then this number contains the COM port number

Note

- 0 HIS_BOARD_TYPE_NONE no device (not valid)
- 1 HIS_BOARD_TYPE_ELTEC XRD-FG or XRD-FGe Frame Grabber
- 8 HIS_BOARD_TYPE_ELTEC_XRD_FGX XRD-FGX frame grabber
- 16 HIS_BOARD_TYPE_ELTEC_XRD_FGE_Opto XRD-FGe Opto
- 32 HIS_BOARD_TYPE_ELTEC_GbIF GigabitEthernet
- 96 HIS_BOARD_TYPE_ELTEC_EMBEDDED Embedded Detector (e.g. X-Rpad)

Returns

If the initialization is successful HIS_ALL_OK is returned, otherwise the return value is greater. Call Acquisition_GetErrorCode to get extended information.

3.3.1.6 HIS_RETURN Acquisition_GetConfiguration (HACQDESC *hAcqDesc,* UINT ∗ *dwFrames,* UINT ∗ *dwRows,* UINT ∗ *dwColumns,* UINT ∗ *dwDataType,* UINT ∗ *dwSortFlags,* BOOL ∗ *bIRQEnabled,* DWORD ∗ *dwAcqType,* DWORD ∗ *dwSystemId,* DWORD ∗ *dwSyncMode,* DWORD ∗ *dwHwAccess*)

This function retrieves all important acquisition parameters, that can be set by -Acquisition_Init or that are set by the self configuration mechanisms of the XISL.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwFrames	Number of frames of acquisition buffer.
dwRows	Number of rows of the sensor.

dwColumns	Number of columns of the sensor.
dwDataType	Type of data of acquisition buffer (should always be two = unsigned short).
dwSortFlags	Type of sorting. This value depends on camera and used sensor (see sorting schemes).
bIRQ-Enabled	Retrieves a flag that indicates if interrupts are enabled (see Acquisition-_Init,. hardware interrupts) or if the hardware is running in polling mode. In interrupt mode this parameter is equal to one and zero in the other case.
dwAcqType	Only for internal use.
dwSystemId	PROM identification number of the used camera. This number is only important if the camera operates objectionably and you need any support from PerkinElmer.
dwSync-Mode	This parameter receives the sync mode, see table.
dwHw-Access	This parameter receives the hardware access parameter, that is programmed into the camera. If you have to use this parameter (that depends from your contract with PerkinElmer) please contact PerkinElmer for the possible values.

Note

- HIS_SYNCMODE_FREE_RUNNING The sensor is operating in free running mode.
- HIS_SYNCMODE_EXTERNAL_TRIGGER The sensor is operating in triggered mode. Frames are only send if an external trigger signal is applied to the camera.
- HIS_SYNCMODE_INTERNAL_TIMER The synchronization signal can be generated by the internal timer of the frame grabber (see Acquisition_SetTimerSync)
- HIS_SYNCMODE_SOFT_TRIGGER The synchronization signal can be generated by software (see Acquisition_SetFrameSync).

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.7 HIS_RETURN Acquisition_GetHwHeaderInfo (HACQDESC *hAcqDesc,* CHwHeaderInfo ∗ *pInfo*)

This function returns the contents of the camera's hardware header in a CHwHeaderInfo structure.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pInfo	Pointer to a Structure of type CHwHeaderInfo that contains the contents of the camera's hardware header necessary for self configuration features.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.8 HIS_RETURN **Acquisition_GetHwHeaderInfoEx (HACQDESC** ***hAcqDesc,*** **CHwHeaderInfo** ∗ ***pInfo,*** **CHwHeaderInfoEx** ∗ ***pInfoEx*** **)**

This function acquires the frame header of the connected detector. If dwHeaderID in the CHwHeaderInfo structure is 14 (1621detectors) pInfoEx will retrieve the extended header, otherwise the structure will be filled with 0xFFFF.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pInfo	Pointer to Structure of type CHwHeaderInfo to retrieve the detector's hardware header.
pInfoEx	Pointer to Structure of type CHwHeaderInfoEx to retrieve the detector's hardware header when available. Can be NULL.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

∗ //val 2009-08-13 wait for frame end before retrieving the header

3.3.1.9 HIS_RETURN **Acquisition_GetIntTimes (HACQDESC** ***hAcqDesc,*** **double** ∗ ***dblIntTime,*** **int** ∗ ***nIntTimes*** **)**

This function retrieves the current integration times.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dblIntTime	Pointer to an array of 8 byte floating point numbers. This array must contain at least 8 entries.
nIntTimes	This parameter contains the number of maximum entries in the array of 8 byte floating point numbers pointed to by ∗dblIntTime. After return of the function this variable provides the number of available integration times.

```
double dblIntTimes[8];
int nIntTimes = 8;
if (Acquisition_GetIntTimes(hAcqDesc, dblIntTimes, &nIntTimes)!=HIS_ALL_OK)
{
   //error handling
}
printf("Number of available integration times: %d\n", nIntTimes);
for (int i=0; i<nIntTimes; i++)
{
    printf("%d: %f\n",i, dblIntTimes[i]);
}
```

Returns

Value If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.3.1.10 HIS_RETURN **Acquisition_GetNextSensor** (**ACQDESCPOS** ∗ *Pos*, **HACQDESC** ∗ *phAcqDesc*)

You can use this function to iterate through all recognized sensors in the system. for a programming example see the initialization part of the XISL demonstration.

Parameters

Pos	Pointer to an unsigned 4 byte integer that receives informations that are needed for subsequent calls of this function. To receive the acquisition descriptor (HACQDESC) for the first recognized sensor set Pos to N-ULL. In case of the 64bit library ACQDESCPOS is a void pointer like defined in Acq.h of the driver SDK.
phAcqDesc	Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call Acquisition_Init the first time set hAcqDesc to NULL, in subsequent calls use the former returned value.

Returns

If the initialization is successful zero is returned, otherwise the return value is greater. Call Acquisition_GetErrorCode to get extended information.

3.3.1.11 HIS_RETURN **Acquisition_Init** (**HACQDESC** ∗ *phAcqDesc*, DWORD *dwBoardType*, int *nChannelNr*, BOOL *bEnableIRQ*, UINT *Rows*, UINT *Columns*, UINT *dwSortFlag*, BOOL *bSelfInit*, BOOL *bAlwaysOpen*)

The Acquisition_Init function initializes the frame grabber board and the corresponding driver. It enables desired hardware interrupts, prepares acquisition threads, defines callback functions to react on acquisition status changes and tests for sufficient memory space for DMA (direct memory access).

Parameters

phAcqDesc	Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call Acquisition_Init the first time set hAcqDesc to NULL, in subsequent calls use the former returned value.
dwBoard-Type	This parameter defines on which communication device the sensor is located. Only one type of frame grabber can be used at the same time.
nChannelNr	This parameter defines the device number. Its possible values depend from dwBoardType and the number of the installed components. For instance if you installed 2 frame grabber boards and you want to acquire data from that one, on that the hardware board selector is set to three, set dwChannelNr equal to 3.
bEnableIRQ	If you want to run the acquisition in polling mode set this parameter to zero. If you want to enable hardware interrupts set the parameter to one.
Rows	Number of sensor columns and rows
Columns	Number of sensor columns and rows
dwSortFlag	Depending on the sensor different sorting schemes are needed because the data come in incorrect order from the detector. dwSortFlags can be one of the following values: The sorting is done automatically by XISL during acquisition. The sorting routines are written in machine code and are therefore very fast.
bSelfInit	If bSelfInit is set to true the function retrieves the detector parameters (Rows, Columns,SortFlags) automatically. If bSelfInit is set to false the configuration parameters supplied by Rows, Columns, dwSortFlags are used.
bAlways-Open	If this parameter is TRUE the XISL is capturing the requested communication port regardless if this port is already opened by other processes running on the system. The use of this option is only recommended in debug versions of your applications because it is not possible to free all system resources allocated by another process.

Note

It enables desired hardware interrupts, prepares acquisition threads, defines callback functions to react on acquisition status changes and tests for sufficient memory space for DMA (direct memory access). Supported interfaces and channel types:

- 1 HIS_BOARD_TYPE_ELTEC The communication interface to the detector is an XRD-FG or XRD-FGe frame grabber.
- 8 HIS_BOARD_TYPE_ELTEC_XRD_FGX The communication interface to the detector is an XRD-FGX frame grabber.
- 16 HIS_BOARD_TYPE_ELTEC_XRD_FGE_Opto The communication interface to the detector is an XRD-FGe Opto frame grabber.
- 32 HIS_BOARD_TYPE_ELTEC_GbIF The detector communicates over GigabitEthernet with the host PC. (For GbIF and XRpad Detectors - Acquisition_GbIF_Init must be used)
- 96 HIS_BOARD_TYPE_ELTEC_EMBEDDED The detector communicates over GigabitEthernet or WLAN with the host PC. (For GbIF and XRpad -

Detectors Acquisition_GbIF_Init must be used) This value is an ored combination of HIS_BOARD_TYPE_ELTEC_GbIF and HIS_BOARD_TYPE_ELTEC_WPE showing that extended Functionality to the GbIF is available. The value will be set automatically after initialization and can be retrieved by - Acquisition_GetCommChannel(..) Sorting Flags:

- HIS_SORT_NOSORT (0x0) no sorting / XRpad series / 0822 / 1622 / 1642 / 1611
- HIS_SORT_QUAD (0x1) RID128
- HIS_SORT_COLUMN (0x2 RID256
- HIS_SORT_COLUMNQUAD (0x3) RID128-400
- HIS_SORT_QUAD_INVERSE (0x4) RID1024-100
- HID_SORT_QUAD_TILE (0x5) RID512-400 A0
- HIS_SORT_QUAD_TILE_INVERSE (0x6) XRD512-400 A1/A2 XRD 0840
- HIS_SORT_QUAD_TILE_INVERSE_SCRAMBLE (0x7) XRD 512-400 E
- HIS_SORT_OCT_TITLE_INVERSE (0x8) XRD 1640 A , XRD 1620 A , XRD 0820
- HIS_SORT_HEX_TILE_INVERSE (11) XRD 1620/21 AM/AN
- HIS_SORT_HEX_CS(12) XRD 1620/40 AN CS
- HIS_SORT_TOP_BOTTOM(15) XRD 4343RF

Returns

HIS_ALL_OK function successful otherwise an error code

3.3.1.12 HIS_RETURN Acquisition_SetCallbacksAndMessages (HACQDESC *hAcqDesc*, HWND *hWnd*, UINT *dwErrorMsg*, UINT *dwLoosingFramesMsg*, void(CALLBACK ∗lpfnEndFrameCallback)(HACQDESC) *,* void(CALLBACK ∗lpfnEndAcqCallback)(HACQDESC))

The Acquisition_SetCallbacksAndMessages function defines callback functions to react on acquisition status changes. For a programming example see the initialization part of the XISL demonstration.

Parameters

hAcqDesc	Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call Acquisition_Init the first time set hAcqDesc to NULL, in subsequent calls use the former returned value.
hWnd	If the HSL recognizes an end of DMA transfer and it is ready with sorting, it checks if the application called Acquisition_SetReady after redrawing. If the application did not call the function, an user defined message (dwLoosingFramesMsg) is posted to hWnd for further handling. If an error occurred during acquisition also a user defined message (dwErrorMsg) is posted to hWnd.

dwErrorMsg	Defines a user message that is posted to hWnd if an error occurs during acquisition.
dwLoosing-FramesMsg	Defines a user message that is posted to hWnd if Acquisition_SetReady wasn't called by the application at the end of sorting.
lpfnEnd-Frame-Callback	Defines a function pointer that is called after the XISL did the sorting. The prototype for the function is given by: In this routine you can do corrections, on-line image processing and redrawing of your data images. Be careful with sending messages from this callback to your application. lpfnEndFrameCallback and lpfnEndAcqCallback are called from a separate thread which is dissimilar to the applications main thread. That should cause problems if you send messages to your main thread via SendMessage. If this causes problems use PostMessage instead. If this parameter is set to NULL it is ignored
lpfnEndAcq-Callback	Defines a function pointer that is called after the XISL did the sorting. The prototype for the function is given by: In this routine you can perform any clean up at acquisition end. If this parameter is set to NULL, it is ignored.

Note

The prototype for the function is given by: void CALLBACK OnEndAcqCallback(H-ACQDESC hAcqDesc); In this routine you can perform any clean up at acquisition end. If this parameter is set to NULL, it is ignored.

Returns

HIS_ALL_OK if successful

3.3.1.13 HIS_RETURN Acquisition_SetLogLevel (XislLoggingLevels *xislLogLvl*)

This function is to set the logging level of the Xisl logging.

Parameters

xislLogLvl	Enum value of which logging level to be used (i.e. LEVEL_DEBUG to display Debug logging and below)

Returns

If the function is successful it returns zero, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.14 ∗∗ ∗ ∗∗ ∗ HIS_RETURN Acquisition_SetLogOutput (const char ∗ *filePath,* BOOL *consoleOnOff*)

∗∗∗/

**/

This function will create a log file based on the file name provided and will enable console logging if desired.

Parameters

filePath	Customizable file name for logging output file. If the filepath parameter is NULL a default logging file will be created.
consoleOn-Off	Parameter used to enable or disable logging output to the console

Returns

If the function is successful it returns zero, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.3.1.15 HIS_RETURN Acquisition_TogglePerformanceLogging (BOOL *onOff*)

This function is used to toggle the performance logging. The performance logged will measure the time it takes for a particular function call to execute.

Parameters

onOff	Boolean value used to determine whether to turn internal logging on or off

Returns

If the function is successful it returns zero, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.4 Init Functions for GbIF

Functions

- HIS_RETURN Acquisition_GbIF_CheckNetworkSpeed (HACQDESC hAcqDesc, WORD ∗wTiming, long ∗lPacketDelay, long lMaxNetworkLoadPercent)

  This function determines which Timing and Packetdelay the Detector can be set for the current system and network configuration. Please note that this function is intended for one detector only. If more than one detector is connected to the network adapter (detectors in LAN), the parameter lMaxNetworkPercent must divided by the number of connected sensors. Since the network load might change during operation this function cannot guarantee optimal performance. Use Acquisition_GbIF_SetPacketDelay and Acquisition_SetCameraMode(..) to apply the determined settings. For X-Rpad detectors a fixed value is returned for LAN/WLAN transmission. (80% network load are recommended)
- HIS_RETURN Acquisition_GbIF_DiscoverDetectors ()

  Discover all NICs for PKI devices via IP broadcast.
- HIS_RETURN Acquisition_GbIF_DiscoveredDetectorByIndex (long lIndex, GBIF_DEVICE_PARAM ∗pDevice)

  Retrieves a GBIF_DEVICE_PARAM structure by index.
- HIS_RETURN Acquisition_GbIF_DiscoveredDetectorCount (long ∗pDeviceCount)

  Retrieves the count of the discovered devices.
- HIS_RETURN Acquisition_GbIF_ForceIP (GBIF_STRING_DATATYPE ∗cMAC, GBIF_STRING_DATATYPE ∗cDefIP, GBIF_STRING_DATATYPE ∗cDefSubNetMask, GBIF_STRING_DATATYPE ∗cStdGateway)

  To configure the device it can be helpful to force the device temporarily to connect with a certain IP_Adress, usually one out of the same subnet and with the same StdGateway like the network card of your computer system. With restart of the detector the device will loose the temporary IP and behave as configured (e.g. IP per DHCP or LLA). This function in not available for the XRpad.
- HIS_RETURN Acquisition_GbIF_GetConnectionSettings (GBIF_STRING_DATATYPE ∗ucMAC, unsigned long ∗ulBootOptions, GBIF_STRING_DATATYPE ∗ucDefIP, GBIF_STRING_DATATYPE ∗ucDefSubNetMask, GBIF_STRING_DATATYPE ∗ucStdGateway)

  This function retrieves the connection parameters of a GbIF detector.
- HIS_RETURN Acquisition_GbIF_GetDetectorProperties (HACQDESC hAcqDesc, GBIF_Detector_Properties ∗pDetectorProperties)

  This function fills the GBIF_Detector_Properties structure, which contains permanently stored information of the connected device.
- HIS_RETURN Acquisition_GbIF_GetDevice (GBIF_STRING_DATATYPE ∗ucAddress, DWORD dwAddressType, GBIF_DEVICE_PARAM ∗pDevice)

  This function retrieves the device specified by the passed MAC address.
- HIS_RETURN Acquisition_GbIF_GetDeviceCnt (long ∗plNrOfboards)

  This function retrieves the total number of sensors found in the network by a network broadcast. If more than one network adapter are installed, a broadcast will be performed on all of them.

- HIS_RETURN Acquisition_GbIF_GetDeviceList (GBIF_DEVICE_PARAM ∗pGbIF_DEVICE_PARAM, int nDeviceCnt)

  This function retrieves a list of GbIF detector devices found by network broadcast. If multiple network adapters are used in the host system, all of them are checked whether GbIF detectors are connected.

- HIS_RETURN Acquisition_GbIF_GetDeviceParams (HACQDESC hAcqDesc, GBIF_DEVICE_PARAM ∗pDevice)

  This function retrieves the device parameters of a board that has already been opened.

- HIS_RETURN Acquisition_GbIF_GetFilterDrvState (HACQDESC hAcqDesc)

  Returns the Status of the GbIF filter Driver (if installed) otherwise an Error Code.

- HIS_RETURN Acquisition_GbIF_GetPacketDelay (HACQDESC hAcqDesc, long ∗lPacketdelay)

  Retrieve the InterPacket Delay, which is set for the current data connection.

- HIS_RETURN Acquisition_GbIF_GetVersion (int ∗pMajor, int ∗pMinor, int ∗pRelease, char ∗pStrVersion, int iStrLength)

  Retrieve version information about the used GbIF library.

- HIS_RETURN Acquisition_GbIF_Init (HACQDESC ∗phAcqDesc, int nChannelNr, BOOL bEnableIRQ, UINT uiRows, UINT uiColumns, BOOL bSelfInit, BOOL bAlwaysOpen, long lInitType, GBIF_STRING_DATATYPE ∗ucAddress)

  The function Acquisition_GbIF_Init initializes the Ethernet connected detectors and the corresponding drivers. It prepares acquisition threads, defines callback functions to react on acquisition status changes.

- HIS_RETURN Acquisition_GbIF_SetConnectionSettings (GBIF_STRING_DATATYPE ∗cMAC, unsigned long ulBootOptions, GBIF_STRING_DATATYPE ∗cDefIP, GBIF_STRING_DATATYPE ∗cDefSubNetMask, GBIF_STRING_DATATYPE ∗cStdGateway)

  This function provides the parameters to configure how the detector connects to the network adapter. This function in not available for the XRpad.

- HIS_RETURN Acquisition_GbIF_SetDiscoveryTimeout (long timeout)

  Sets the discovery timeout in milliseconds.

- HIS_RETURN Acquisition_GbIF_SetPacketDelay (HACQDESC hAcqDesc, long lPacketdelay)

  The Inter-Packet Delay can be set flexibly to balance out the workload of the IP connection between detector and network adapter. It is recommended to be configured depending on the network load. The value can be retrieved by calling Acquisition_GbIF_CheckNetworkSpeed.

- HIS_RETURN Acquisition_GbIF_SetPortRange (long lStartPort, long lNrOfports)

  enables frame acknowledgement

- HIS_RETURN Acquisition_wpe_GetVersion (int ∗major, int ∗minor, int ∗release, int ∗build)

  This function returns the version of the used wpe200 library if available and loadable.

3.4.1 Function Documentation

3.4.1.1 HIS_RETURN **Acquisition_GbIF_CheckNetworkSpeed** (HACQDESC *hAcqDesc*, WORD ∗ *wTiming*, long ∗ *lPacketDelay*, long *lMaxNetworkLoadPercent*)

This function determines which Timing and Packetdelay the Detector can be set for the current system and network configuration. Please note that this function is intended for one detector only. If more than one detector is connected to the network adapter (detectors in LAN), the parameter lMaxNetworkPercent must divided by the number of connected sensors. Since the network load might change during operation this function cannot guarantee optimal performance. Use Acquisition_GbIF_SetPacketDelay and - Acquisition_SetCameraMode(..) to apply the determined settings. For XRpad detectors a fixed value is returned for LAN/WLAN transmission. (80% network load are recommended)

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
wTiming	Pointer to suggested Free Running Timing for actual System - Performance
lPacketDelay	Pointer to calculated max InterPacketDelay for suggested timing
lMax-Network-LoadPercent	Percentage of Network load for which the Packet Delay shall be cheked. To allow a stable data transmissions with some space for packet resend select∼80 percent

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.2 HIS_RETURN **Acquisition_GbIF_DiscoverDetectors** ( )

Discover all NICs for PKI devices via IP broadcast.

Note

Use Acquisition_GbIF_DiscoveredDevel_GetGBifConnectionStatusiceCount and - Acquisition_GbIF_DiscoveredDeviceByIndex in subsequent calls to retrieve information about the discovered devices.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.4.1.3 HIS_RETURN **Acquisition_GbIF_DiscoveredDetectorByIndex** (long *lIndex*, **GBIF_DEVICE_PARAM** ∗ *pDevice*)

Retrieves a GBIF_DEVICE_PARAM structure by index.

Note

Use Acquisition_GbIF_DiscoverDevices in a preceding call to discover the network.

Parameters

lIndex	Index of the devcice to retrieve the parameters. Valid indices are in the range 0 <= index <= (count - 1).
pDevice	Pointer to a GBIF_DEVICE_PARAM structure to retrieve the parameters ot the device.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.4.1.4 HIS_RETURN **Acquisition_GbIF_DiscoveredDetectorCount** (long ∗ *pDeviceCount*)

Retrieves the count of the discovered devices.

Note

Use Acquisition_GbIF_DiscoverDevices in a preceding call to discover the network.

Parameters

pDevice-Count	Pointer to a variable that retrieves the device count.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.4.1.5 HIS_RETURN **Acquisition_GbIF_ForceIP** (GBIF_STRING_DATATYPE ∗ *cMAC,* GBIF_STRING_DATATYPE ∗ *cDefIP,* GBIF_STRING_DATATYPE ∗ *cDefSubNetMask,* GBIF_STRING_DATATYPE ∗ *cStdGateway*)

To configure the device it can be helpful to force the device temporarily to connect with a certain IP _Adress, usually one out of the same subnet and with the same StdGateway like the network card of your computer system. With restart of the detector the device will loose the temporary IP and behave as configured (e.g. IP per DHCP or LLA). This function in not available for the XRpad.

Parameters

cMAC	MAC address of the device to retrieve a temporary IP.
cDefIP	Temporary IP.

cDefSubNet-Mask	Temporary Subnet Mask.
cStd-Gateway	Temporary Gateway.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.6 HIS_RETURN **Acquisition_GbIF_GetConnectionSettings** (GBIF_STRING_DATATYPE ∗ *ucMAC*, unsigned long ∗ *ulBootOptions*, GBIF_STRING_DATATYPE ∗ *ucDefIP*, GBIF_STRING_DATATYPE ∗ *ucDefSubNetMask*, GBIF_STRING_DATATYPE ∗ *ucStdGateway*)

This function retrieves the connection parameters of a GbIF detector.

Parameters

ucMAC	MAC address of the device having the requested settings.
ulBoot-Options	OR-able flag indicating the type of connection bitwisely
ucDefIP	Retrieves the IP address the device is connected with.
ucDefSub-NetMask	Retrieves the Subnet the device is in.
ucStd-Gateway	Retrieves the Standard Gateway of the connection to the device.

Note

- HIS_GbIF_IP_STATIC - Use Static IP address stored in the sensor
- HIS_GbIF_IP_LLA - Sensor will propose a Local Link Address
- HIS_GbIF_IP_DHCP - Sensor will receive IP address by DHCP server

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.7 HIS_RETURN **Acquisition_GbIF_GetDetectorProperties** (**HACQDESC** *hAcqDesc*, **GBIF_Detector_Properties** ∗ *pDetectorProperties*)

This function fills the GBIF_Detector_Properties structure, which contains permanently stored information of the connected device.

Parameters

hAcqDesc	Acquisition descriptor structure.
pDetector-Properties	Pointer to DetectorProperties structure. The memory for the object must be allocated before calling this function.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.8 HIS_RETURN **Acquisition_GbIF_GetDevice** (GBIF_STRING_DATATYPE ∗ *ucAddress,* DWORD *dwAddressType,* **GBIF_DEVICE_PARAM** ∗ *pDevice*)

This function retrieves the device specified by the passed MAC address.

Parameters

ucAddress	Address (array of 16 characters) to open the specified board. It can represent the MAC address, IP address or device name of the sensor.
dwAddress-Type	To identify of which type the parameter ucAddress is, lInitType can have the following values All values are defined within the file acq.h. IP-Address, MAC-Address and Detector Name can be retrieved by -Acquisition_GbIF_GetDeviceList.
pDevice	Struct which retrieves attributes and configuration of the device defined by ucAddress.

Note

- HIS_GbIF_IP The sensor is selected by the IP-Address passed in cAddress
- HIS_GbIF_MAC The sensor is selected by the MAC-Adress passed in c-Address
- HIS_GbIF_NAME The sensor is selected by the Detector-Name passed in cAddress

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.9 HIS_RETURN **Acquisition_GbIF_GetDeviceCnt** (long ∗ *plNrOfboards*)

This function retrieves the total number of sensors found in the network by a network broadcast. If more than one network adapter are installed, a broadcast will be performed on all of them.

Parameters

plNr-Ofboards	Retrieves the number of sensors found in the network broadcasting.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.10 HIS_RETURN **Acquisition_GbIF_GetDeviceList** (**GBIF_DEVICE_PARAM** ∗ *pGbIF_DEVICE_PARAM,* int *nDeviceCnt*)

This function retrieves a list of GbIF detector devices found by network broadcast. If multiple network adapters are used in the host system, all of them are checked whether GbIF detectors are connected.

Parameters

pGbIF_DEVICE_PARAM	Returns a list of GBIF_DEVICE_PARAM elements, with nDeviceCnt entries. For that there has to be memory allocated of the size nDeviceCnt∗sizeof(GBIF_DEVICE_PARAM) before calling the function
nDeviceCnt	is the number of devices, which are found within the network. This parameter has to be known before calling Acquisition_GbIF_GetDeviceList and has to be passed to the function. It can be retrieved by calling Acquisition_GbIF_GetDeviceCnt.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.11 HIS_RETURN **Acquisition_GbIF_GetDeviceParams** (**HACQDESC** *hAcqDesc,* **GBIF_DEVICE_PARAM** ∗ *pDevice*)

This function retrieves the device parameters of a board that has already been opened.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pDevice	Pointer to structure which retrieves attributes and configuration of the device defined by hAcqDesc.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.12 HIS_RETURN **Acquisition_GbIF_GetFilterDrvState** (**HACQDESC** *hAcqDesc*)

Returns the Status of the GbIF filter Driver (if installed) otherwise an Error Code.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure

Returns

If the function is successful it returns the status of the Filter driver

- 1 for active
- -1 for disabled / not installed
- if the function is not included in gbif.dll or the HACQDESC is not valid an ErroCode is returned

Note

The Filter driver is fdeprecated and can not be used with current library versions

3.4.1.13 HIS_RETURN Acquisition_GbIF_GetPacketDelay (HACQDESC *hAcqDesc*, long ∗ *lPacketdelay*)

Retrieve the InterPacket Delay, which is set for the current data connection.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
lPacketdelay	Retrieves the currently set value for Inter-Packet Delay

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.14 HIS_RETURN Acquisition_GbIF_GetVersion (int ∗ *pMajor*, int ∗ *pMinor*, int ∗ *pRelease*, char ∗ *pStrVersion*, int *iStrLength*)

Retrieve version information about the used GbIF library.

Parameters

pMajor	Pointer to a variable to retrieve the major version number.
pMinor	Pointer to a variable to retrieve the minor version number.
pRelease	Pointer to a variable to retrieve the release number.
pStrVersion	Pointer to an array of characters to retrieve a unique hash id string.
iStrLength	Length of the pStrVersion array.

Note

Since the hash could become 40 characters long (even if it is very unlikely that this will ever happen), we consider a length of 64 characters for the pStrVersion array.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.4.1.15 HIS_RETURN Acquisition_GbIF_Init (HACQDESC ∗ *phAcqDesc,* int *nChannelNr,* BOOL *bEnableIRQ,* UINT *uiRows,* UINT *uiColumns,* BOOL *bSelfInit,* BOOL *bAlwaysOpen,* long *lInitType,* GBIF_STRING_DATATYPE ∗ *ucAddress*)

The function Acquisition_GbIF_Init initializes the Ethernet connected detectors and the corresponding drivers. It prepares acquisition threads, defines callback functions to react on acquisition status changes.

Parameters

phAcqDesc	Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call Acquisition_GbIF_Init the first time set hAcq-Desc to NULL, in subsequent calls use the former returned value.
nChannelNr	This parameter defines a number to refer to the initialized device. For each GbIF sensor to be initialized an individual channel number has to be assigned. If you try to access multiple sensors using the same channel number, only the first one will be successfully initialized.
bEnableIRQ	To run the acquisition in polling mode set this parameter to zero. To enable hardware interrupts set the parameter to one.
uiRows	Number of rows of the sensor.
uiColumns	Number of columns of the sensor.
bSelfInit	If bSelfInit is set to true the function retrieves the detector parameters (Rows, Columns,SortFlags) automatically. If bSelfInit is set to false the configuration parameters supplied by Rows, Columns, dwSortFlags are used.
bAlways-Open	If this parameter is TRUE the XISL is capturing the requested communication port regardless if this port is already opened by other processes running on the system. The use of this option is only recommended in debug versions of your applications because it is not possible to free all system resources allocated by another process.
lInitType	To identify of which type the parameter cAddress is, lInitType can have the following values: • HIS_GbIF_FIRST_CAM 0 • HIS_GbIF_IP 1 • HIS_GbIF_MAC 2 • HIS_GbIF_NAME 3 All values are defined within the file acq.h. I-P-Address, MAC-Address and Detector Name can be retrieved by Acquisition_GbIF_GetDeviceList
ucAddress	Address (array of 16 characters) to open the specified board. It can represent the MAC address, IP address or device name of the sensor.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.4.1.16 HIS_RETURN Acquisition_GbIF_SetConnectionSettings (GBIF_STRING_DATATYPE ∗ *cMAC,* unsigned long *ulBootOptions,* GBIF_STRING_DATATYPE ∗ *cDefIP,* GBIF_STRING_DATATYPE ∗ *cDefSubNetMask,* GBIF_STRING_DATATYPE ∗ *cStdGateway*)

This function provides the parameters to configure how the detector connects to the network adapter. This function in not available for the XRpad.

Parameters

cMAC	MAC address of the device to be configured.
ulBoot-Options	• OR-able flag to set the type of connection bitwisely:
cDefIP	In case ulBootOptions is equal to HIS_GbIF_IP_STATIC, cDefIP can be used to set a new value as static IP. For that, cDefIP has to contain an IP address when calling the function.
cDefSubNet-Mask	In case ulBootOptions is equal to HIS_GbIF_IP_STATIC, cDefSubNet-Mask can be used to set a new value as static IP. For that, cDefSubNet-Mask has to contain an IP address when calling the function.
cStd-Gateway	In case ulBootOptions is equal to HIS_GbIF_IP_STATIC, cStdGateway can be used to set a new value as static IP. For that, cStdGateway has to contain an IP address when calling the function. When cStdGateway is zero the detector will be able to be initialized by Acquisition_Enum-Sensors(..)

Note

- HIS_GbIF_IP_STATIC - Use Static IP address stored in the sensor
- HIS_GbIF_IP_LLA - Sensor will propose a Local Link Address
- HIS_GbIF_IP_DHCP - Sensor will receive IP address by DHCP server

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.17 HIS_RETURN Acquisition_GbIF_SetDiscoveryTimeout (long *timeout*)

Sets the discovery timeout in milliseconds.

The discovery timeout is the time the library waits for responses of devices in the network, after sending out a broadcast. You should increase this value, if you encounter problems by discovering devices. This applies especially for noisy WLan environments.

The default value is 750 milliseconds.

Parameters

`in`	*timeout*	Discovery timeout in milliseconds. Must be a positive value.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.4.1.18 HIS_RETURN **Acquisition_GbIF_SetPacketDelay** (**HACQDESC** *hAcqDesc,* long *lPacketdelay*)

The Inter-Packet Delay can be set flexibly to balance out the workload of the IP connection between detector and network adapter. It is recommended to be configured depending on the network load. The value can be retrieved by calling Acquisition_GbIF_CheckNetworkSpeed.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
lPacketdelay	Value for Inter-Packet Delay, which is to be set in the unit TICKS

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.4.1.19 HIS_RETURN **Acquisition_GbIF_SetPortRange** (long *lStartPort,* long *lNrOfports*)

enables frame acknowledgement

Parameters

hAcqDesc	
Enable	

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

acknowledges last frame

Parameters

hAcqDesc	
reserved	

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

set port range to be used bith Ethernet detecors to lStartPort, lNrOfports

Parameters

lStartPort	Starting Port (should be 1024)
lNrOfports	Number of Devices max (should be 16)

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.4.1.20 HIS_RETURN Acquisition_wpe_GetVersion (int ∗ *major,* int ∗ *minor,* int ∗ *release,* int ∗ *build*)

This function returns the version of the used wpe200 library if available and loadable.

Parameters

major	major version
minor	minor version
release	release version
build	build version

Returns

HIS_ALL_OK function successful otherwise an error code

3.5 Acquire and correction functions

Functions

- HIS_RETURN Acquisition_Abort (HACQDESC hAcqDesc)

  This routine aborts a currently running acquisition.

- HIS_RETURN Acquisition_AbortCurrentFrame (HACQDESC hAcqDesc)

  This routine aborts the transfer of the current frame from the detector to the frame grabber. The detector will be reset then and a new frame transfer will be started immediately. The detector should run in the same mode as the image correction files have been obtained. Therefore it is not recommended to use this function during a measurement.

- HIS_RETURN Acquisition_Acquire_GainImage (HACQDESC hAcqDesc, WORD ∗pwOffsetData, DWORD ∗pdwGainData, UINT nRows, UINT nColumns, UINT nFrames)

  This function acquires nFrames which are all offset corrected by data stored in pOffsetData. After that the gain data are added in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). After averaging the data are further processed for subsequent gain correction of image data. The last acquired data at frame end time are available via pGainData. At the end of the acquisition time the gain data are also accessible via pGainData. The offset data are necessary to derive a valid gain image. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

- HIS_RETURN Acquisition_Acquire_GainImage_Ex (HACQDESC hAcqDesc, WORD ∗pOffsetData, DWORD ∗pGainData, UINT nRows, UINT nCols, UINT nFrames, UINT dwOpt)

  This function acquires nFrames which are all offset corrected by data stored in pOffsetData. After that the gain data are added in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). After averaging the data are further processed for subsequent gain correction of image data. The last acquired data at frame end time are available via pGainData. At the end of the acquisition time the gain data are also accessible via pGainData. The offset data are necessary to derive a valid gain image. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

- HIS_RETURN Acquisition_Acquire_GainImage_Ex_ROI (HACQDESC hAcqDesc, WORD ∗pwOffsetData, DWORD ∗pdwGainData, UINT nRows, UINT nColumns, UINT nFrames, UINT dwOpt, UINT uiULX, UINT uiULY, UINT uiBRX, UINT uiBRY, UINT uiMode)

  This function is similar to the Acquisiton_Acquire_GainImage(..)-function. The function provides the possibility to use a well-defined region of interest for the median determination. The median is used to calculate the gain of single pixels. (see Mathematical description of corrections) The function should be used to acquire the gain image when not the whole panel is illuminated.

- HIS_RETURN Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr (HACQDESC hAcqDesc, DWORD ∗pdwGainData, UINT nRows, UINT nColumns, UINT nFrames, UINT dwOpt, UINT uiULX, UINT uiULY, UINT uiBRX, UINT uiBRY, UINT uiMode)

> *The function provides the same functionality as Acquisition_Acquire_GainImage_EX-_ROI(. . .) except loading the image correction data. Please use Acquisition_SetCorr-Data_Ex(..) to set the correction data before.*

- HIS_RETURN Acquisition_Acquire_GainImage_PreloadCorr (HACQDESC h-AcqDesc, DWORD ∗pGainData, UINT nRows, UINT nCols, UINT nFrames)

  > *The function provides the same functionality as Acquisition_Acquire_Gain_Image(. . .) except loading the image correction data. Use Acquisition_SetCorrData_Ex before to set Offset Correction data only.∗.*

- HIS_RETURN Acquisition_Acquire_Image (HACQDESC hAcqDesc, UINT dw-Frames, UINT dwSkipFrms, UINT dwOpt, unsigned short ∗pwOffsetData, DW-ORD ∗pdwGainData, DWORD ∗pdwPxlCorrList)

  > *This function acquires dwFrames frames and performs offset, gain and pixel corrections automatically. The routine returns immediately. If you want to be informed about frame end or acquisition end, then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.*

- HIS_RETURN Acquisition_Acquire_Image_Ex (HACQDESC hAcqDesc, UINT dwFrames, UINT dwSkipFrms, UINT dwOpt, unsigned short ∗pwOffsetData, UI-NT dwGainFrames, unsigned short ∗pwGainData, unsigned short ∗pwGainAvg-Data, DWORD ∗pdwGainData, DWORD ∗pdwPxlCorrList)

  > *This function acquires dwFrames frames and performs offset, gain and pixel corrections automatically. The routine returns immediately. If you want to be informed about frame end or acquisition end, then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.*

- HIS_RETURN Acquisition_Acquire_Image_PreloadCorr (HACQDESC hAcq-Desc, UINT dwFrames, UINT dwSkipFrms, UINT dwOpt)

  > *The function provides the same functionality as Acquisition_Acquire_Image(. . .) except loading the image correction data. Use Acquisition_SetCorrData_Ex before to set the correction data.*

- HIS_RETURN Acquisition_Acquire_OffsetImage (HACQDESC hAcqDesc, WO-RD ∗pwOffsetData, UINT nRows, UINT nColumns, UINT nFrames)

  > *This function acquires nFrames, adds them in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). The last acquired data at frame end time are available via pOffsetData. At the end of the acquisition time the averaged data are also accessible via pOffsetData. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.*

- HIS_RETURN Acquisition_Acquire_OffsetImage_Ex (HACQDESC hAcqDesc, -WORD ∗pwOffsetData, UINT nRows, UINT nColumns, UINT nFrames, UINT dw-Opt)

  > *This function acquires nFrames, adds them in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). The last acquired data at frame end time are available via pOffsetData. At the end of the acquisition time the averaged data are also accessible via pOffsetData. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.*

- HIS_RETURN Acquisition_Acquire_OffsetImage_PreloadCorr (HACQDESC h-AcqDesc, WORD ∗pwOffsetData, UINT nRows, UINT nColumns, UINT nFrames, UINT dwOpt)

> *The function provides the same functionality as Acquisition_Acquire_Offset_-Image(. . .) except loading the image correction data. Use Acquisition_SetCorr-Data_Ex before to all correction data to NULL.*

- HIS_RETURN Acquisition_CreateGainMap (WORD ∗pGainData, WORD ∗p-GainAVG, int nCount, int nFrame)

  This function creates a list of median values for gain correction using an bright offet corr image to be used with the function Acquisition_Acquire_Image_Ex, Acquisition_Set-CorrData_Ex or Acquisition_DoOffsetGainCorrection_Ex. One value for each image in the pGainData-sequence.

- HIS_RETURN Acquisition_CreateGainMap32 (unsigned long ∗pGainData, un-signed long ∗pGainAVG, int nCount, int nFrame)

  This function creates a list of median values to be used with the function Acquisition_-Acquire_Image_Ex . One value for each image in the pGainData-sequence.

- HIS_RETURN Acquisition_CreateOnboardPixelMaskFrom16BitPixelMask (un-signed short ∗uspPixelMaskSrc, DWORD dwRows, DWORD dwColumns, un-signed char ∗bpOnboardPixelMask)

  This function creates an on-board style (i.e. 1 bit/pixel) pixel mask from a standard (16 bit/pixel) pixel mask image to be used on XRpad2 detectors.

- HIS_RETURN Acquisition_CreatePixelMap (WORD ∗pwData, int nDataRows, int nDataColumns, int ∗pCorrList, int ∗pnCorrListSize)

  This function specifies the size of or creates a pixel correction list (depending on pwData) that one can use in Acquisition_Acquire_Image or Acquisition_DoPixel-Correction.

- HIS_RETURN Acquisition_DefineDestBuffers (HACQDESC hAcqDesc, un-signed short ∗pProcessedData, UINT nFrames, UINT nRows, UINT nColumns)

  This function defines the pointers of the destination buffer for Acquisition_Acquire_-Image. The data are written into this buffer after sorting. The buffer must have a proper size depending on acquisition mode. To acquire one image the buffer must have the size sensor rows ∗ sensor columns ∗2. To acquire a sequence the buffer must have the size sensor rows ∗ sensor columns ∗2 ∗ frames. In the case of continuous acquisition the buffer must have the size sensor rows ∗ sensor columns ∗2 ∗ frames of the ring buffer.

- HIS_RETURN Acquisition_DoOffsetCorrection (WORD ∗pSource, WORD ∗p-Dest, WORD ∗pOffsetData, int nCount)

  This function performs an offset correction for the data defined in pSource. A suitable place to call this function is the end of frame callback function or the end of acquisition callback.

- HIS_RETURN Acquisition_DoOffsetGainCorrection (WORD ∗pSource, WORD ∗pDest, WORD ∗pOffsetData, DWORD ∗pGainData, int nCount)

  This function performs an offset and a gain correction for the data defined in pSource at once. A suitable place to call this function is the end of frame callback function or the end acq callback.

- HIS_RETURN Acquisition_DoOffsetGainCorrection32 (unsigned long ∗pSource, unsigned long ∗pDest, unsigned long ∗pOffsetData, unsigned long ∗pGainData, int nCount)

  Similar to Acquisition_DoOffsetGainCorrection.

- HIS_RETURN Acquisition_DoOffsetGainCorrection_Ex (WORD ∗pSource, WO-RD ∗pDest, WORD ∗pOffsetData, WORD ∗pGainData, WORD ∗pGainAVG, int nCount, int nFrame)

This function performs an offset and a multi gain correction using offset corrected bright images as gain for the data defined in pSource at once. A suitable place to call this function is the end of frame callback function or the end of acq callback.

- HIS_RETURN Acquisition_DoPixelCorrection (WORD ∗pData, int ∗pCorrList)

  This function performs a defect pixel correction on the data defined by pData using the pCorrList. A suitable place to call this function is the end of frame callback or after the end acq callback.
- HIS_RETURN Acquisition_GetCorrData (HACQDESC hAcqDesc, unsigned short ∗∗ppwOffsetData, DWORD ∗∗ppdwGainData, DWORD ∗∗ppdwPxlCorrList)

  This function returns the adresses of the applied correction buffers.
- HIS_RETURN Acquisition_GetCorrData_Ex (HACQDESC hAcqDesc, unsigned short ∗∗ppwOffsetData, unsigned short ∗∗ppwGainData, unsigned short ∗∗ppwGainAvgData, UINT ∗∗nGainFrames, DWORD ∗∗ppdwGainData, DWORD ∗∗ppdwPxlCorrList)

  This function retrieves the current correction data during a running acquisition.
- HIS_RETURN Acquisition_GetLatestFrameHeader (HACQDESC hAcqDesc, CHwHeaderInfo ∗pInfo, CHwHeaderInfoEx ∗pInfoEx)

  Use this function to retrieve the last acuiered frame header of the connected detector. If dwHeaderID in the CHwHeaderInfo structure is 14 pInfoEx will retrieve the extended header, otherwise the structure will be filled with 0xFFFF.
- HIS_RETURN Acquisition_SetCorrData (HACQDESC hAcqDesc, unsigned short ∗pwOffsetData, DWORD ∗pdwGainData, DWORD ∗pdwPxlCorrList)

  This function switches the correction buffers during a running acquisition. It is also possible to switch off corrections by setting the pOffsetData, pGainData and pPixelCorrList to NULL.
- HIS_RETURN Acquisition_SetCorrData_Ex (HACQDESC hAcqDesc, unsigned short ∗pwOffsetData, unsigned short ∗pwGainData, unsigned short ∗pwGainAvgData, UINT nGainFrames, DWORD ∗pdwGainData, DWORD ∗pdwPxlCorrList)

  This function switches the correction buffers during a running acquisition. It is also possible to switch off corrections by setting the pOffsetData, pGainData pwGainData, pwGainAvgData, nGainFrames, and pPixelCorrList to NULL.

3.5.1 Function Documentation

3.5.1.1 HIS_RETURN Acquisition_Abort (HACQDESC *hAcqDesc*)

This routine aborts a currently running acquisition.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.2 HIS_RETURN Acquisition_AbortCurrentFrame (HACQDESC *hAcqDesc*)

This routine aborts the transfer of the current frame from the detector to the frame grabber. The detector will be reset then and a new frame transfer will be started immediately. The detector should run in the same mode as the image correction files have been obtained. Therefore it is not recommended to use this function during a measurement.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.3 HIS_RETURN Acquisition_Acquire_GainImage (HACQDESC *hAcqDesc,* WORD ∗ *pwOffsetData,* DWORD ∗ *pdwGainData,* UINT *nRows,* UINT *nColumns,* UINT *nFrames*)

This function acquires nFrames which are all offset corrected by data stored in pOffsetData. After that the gain data are added in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). After averaging the data are further processed for subsequent gain correction of image data. The last acquired data at frame end time are available via pGainData. At the end of the acquisition time the gain data are also accessible via pGainData. The offset data are necessary to derive a valid gain image. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pwOffset-Data	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). It is recommended to acquire the Offset shortly before calling Acquisition_Acquire_GainImage
pdwGain-Data	Pointer to buffer that receives the gain data.
nRows	Number of rows of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nColumns	Number of columns of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nFrames	Number of frames to acquire.

Note

For the optical frame grabbers its recommended to use the Gainsequence correction using offset corrected bright images as a gain (even a single point gain correction is used) since the onboard correction works with offset corrected bright images. This will increase the speed to upload the images to the onboard correction buffer

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.4 HIS_RETURN Acquisition_Acquire_GainImage_Ex (HACQDESC *hAcqDesc*, WORD ∗ *pOffsetData*, DWORD ∗ *pGainData*, UINT *nRows*, UINT *nCols*, UINT *nFrames*, UINT *dwOpt*)

This function acquires nFrames which are all offset corrected by data stored in pOffsetData. After that the gain data are added in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). After averaging the data are further processed for subsequent gain correction of image data. The last acquired data at frame end time are available via pGainData. At the end of the acquisition time the gain data are also accessible via pGainData. The offset data are necessary to derive a valid gain image. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pOffsetData	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). It is recommended to acquire the Offset shortly before calling Acquisition_Acquire_GainImage.
pGainData	Pointer to buffer that receives the gain data.
nRows	Number of rows of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nCols	Number of columns of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nFrames	Number of frames to acquire.
dwOpt	Options

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.5 HIS_RETURN **Acquisition_Acquire_GainImage_Ex_ROI** (**HACQDESC** *hAcqDesc,* WORD ∗ *pwOffsetData,* DWORD ∗ *pdwGainData,* UINT *nRows,* UINT *nColumns,* UINT *nFrames,* UINT *dwOpt,* UINT *uiULX,* UINT *uiULY,* UINT *uiBRX,* UINT *uiBRY,* UINT *uiMode*)

This function is similar to the Acquisiton_Acquire_GainImage(..)-function. The function provides the possibility to use a well-defined region of interest for the median determination. The median is used to calculate the gain of single pixels. (see Mathematical description of corrections) The function should be used to acquire the gain image when not the whole panel is illuminated.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
pwOffset-Data	Pointer that contains offset data. (see Acquisition_Acquire_Offset-Image). It is recommended to acquire the Offset shortly before calling Acquisition_Acquire_GainImage
pdwGain-Data	Pointer to buffer that receives the gain data
nRows	Number of rows and columns of the offset data buffer. If the values are not suitable to the current connected sensor the function return with an error.
nColumns	Number of rows and columns of the offset data buffer. If the values are not suitable to the current connected sensor the function return with an error.
nFrames	Number of frames to acquire.
dwOpt	must be 0
uiULX	UpperLeftX define a rect which is used to calculate the Median for the pixel-gain calculation
uiULY	UpperLeftY define a rect which is used to calculate the Median for the pixel-gain calculation
uiBRX	BottomRightX define a rect which is used to calculate the Median for the pixel-gain calculation
uiBRY	BottomRightY define a rect which is used to calculate the Median for the pixel-gain calculation
uiMode	see table

Note

- 0 - normal Gain whole image used for median determination.
- 1 - Median for pixel-gain calculation from ROI (defined by uiULX. . .)
- 2 - Median for pixel-gain calculation from ROI each pixel-gain outside ROI will be set to 1

- 3 - Median for pixel-gain calculation from ROI each pixel-gain outside ROI will be set to 0

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

∗// val 20051012

3.5.1.6 HIS_RETURN Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr (HACQDESC *hAcqDesc*, DWORD ∗ *pdwGainData*, UINT *nRows*, UINT *nColumns*, UINT *nFrames*, UINT *dwOpt*, UINT *uiULX*, UINT *uiULY*, UINT *uiBRX*, UINT *uiBRY*, UINT *uiMode*)

The function provides the same functionality as Acquisition_Acquire_GainImage_EX_-ROI(. . .) except loading the image correction data. Please use Acquisition_SetCorr-Data_Ex(..) to set the correction data before.

Note

Please find the parameter description above

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.7 HIS_RETURN Acquisition_Acquire_GainImage_PreloadCorr (HACQDESC *hAcqDesc*, DWORD ∗ *pGainData*, UINT *nRows*, UINT *nCols*, UINT *nFrames*)

The function provides the same functionality as Acquisition_Acquire_Gain_Image(. . .) except loading the image correction data. Use Acquisition_SetCorrData_Ex before to set Offset Correction data only.∗.

Note

Please find the parameter description at Acquisition_Acquire_GainImage(..)

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.8 HIS_RETURN Acquisition_Acquire_Image (HACQDESC *hAcqDesc,* UINT *dwFrames,* UINT *dwSkipFrms,* UINT *dwOpt,* unsigned short $*$ *pwOffsetData,* DWORD $*$ *pdwGainData,* DWORD $*$ *pdwPxlCorrList*)

This function acquires dwFrames frames and performs offset, gain and pixel corrections automatically. The routine returns immediately. If you want to be informed about frame end or acquisition end, then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwFrames	Number of frames to acquire is one of the sequence options is set for dwOpt. If the continuous option is set this value gives the number of frames in a ring buffer that is used for continuous data acquisition.
dwSkipFrms	Number of frames to skip before a frames is copied into the acquisition buffer.
dwOpt	Options for sequence acquisition.
pwOffset-Data	Pointer that contains offset data. (see Acquisition_Acquire_Offset-Image). The Offset must be actual. It is recommended to acquire them shortly before calling Acquisition_Acquire. If you don't want to perform an offset correction set this parameter to NULL.
pdwGain-Data	Pointer that contains gain data. (see Acquisition_Acquire_GainImage). If you don't want to perform a gain correction set this parameter to NU-LL.
pdwPxlCorr-List	Pointer to a buffer that contains pixel correction data.pdwPixelData points to a linear array of data. Its size is given through ((number of wrong pixels) $*$ 10 + 1) $*$ sizeof(int). The first entry in a group of nine contains the offset of the pixel from the base pointer of the data array. The other eight entries equal the offset of the correction pixels from the base pointer. If you want to use less than eight pixels for correction, then set the remaining entries to -1. The value of the pixel is replaced by the mean value of the correction pixels. If you don't want to perform a pixel correction set this parameter to NULL. An easier way to create a pixel map is the use of the XISL function Acquisition_CreatePixelMap.

Note

- HIS_SEQ_TWO_BUFFERS 0x1 Storage of the sequence into two buffers. Secure image acquisition by separated data transfer and later performed image correction.
- HIS_SEQ_ONE_BUFFER 0x2 Storage of the sequence into one buffer. Direct acquisition and linked correction into one buffer.
- HIS_SEQ_AVERAGE 0x4 All acquired single images are directly added into one buffer and after acquisition divided by the number of frames, including linked correction files.
- HIS_SEQ_DEST_ONE_FRAME 0x8 Sequence of frames using the same image buffer
- HIS_SEQ_COLLATE 0x10 Skip frames after acquiring frames in a ring buffer

- HIS_SEQ_CONTINUOUS 0x100 Continuous acquisition Frames are continuously acquired into a ring buffer of dwFrames

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.9 HIS_RETURN Acquisition_Acquire_Image_Ex (HACQDESC *hAcqDesc,* UINT *dwFrames,* UINT *dwSkipFrms,* UINT *dwOpt,* unsigned short ∗ *pwOffsetData,* UINT *dwGainFrames,* unsigned short ∗ *pwGainData,* unsigned short ∗ *pwGainAvgData,* DWORD ∗ *pdwGainData,* DWORD ∗ *pdwPxlCorrList*)

This function acquires dwFrames frames and performs offset, gain and pixel corrections automatically. The routine returns immediately. If you want to be informed about frame end or acquisition end, then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwFrames	Number of frames to acquire is one of the sequence options is set for dwOpt. If the continuous option is set this value gives the number of frames in a ring buffer that is used for continuous data acquisition.
dwSkipFrms	Number of frames to skip before a frames is copied into the acquisition buffer.
dwOpt	Options for sequence acquisition.
pwOffset-Data	Pointer that contains offset data. (see Acquisition_Acquire_Offset-Image). The Offset must be actual. It is recommended to acquire them shortly before calling Acquisition_Acquire. If you don't want to perform an offset correction set this parameter to NULL.
dwGain-Frames	Number of frames in pwGainData
pwGainData	pointer to the median-list created by Acquisition_CreateGainMap
pwGainAvg-Data	List of Medians representing the frames of the pwGainData sequence
pdwGain-Data	Pointer that contains gain data. (see Acquisition_Acquire_GainImage). If you don't want to perform a gain correction set this parameter to NULL
pdwPxlCorr-List	Pointer to a buffer that contains pixel correction data.pdwPixelData points to a linear array of data. Its size is given through ((number of wrong pixels) ∗ 10 + 1) ∗ sizeof(int). The first entry in a group of nine contains the offset of the pixel from the base pointer of the data array. The other eight entries equal the offset of the correction pixels from the base pointer. If you want to use less than eight pixels for correction, then set the remaining entries to -1. The value of the pixel is replaced by the mean value of the correction pixels. If you don't want to perform a pixel correction set this parameter to NULL. An easier way to create a pixel map is the use of the XISL function Acquisition_CreatePixelMap.

Note

- HIS_SEQ_TWO_BUFFERS 0x1 Storage of the sequence into two buffers. Secure image acquisition by separated data transfer and later performed image correction.
- HIS_SEQ_ONE_BUFFER 0x2 Storage of the sequence into one buffer. Direct acquisition and linked correction into one buffer.
- HIS_SEQ_AVERAGE 0x4 All acquired single images are directly added into one buffer and after acquisition divided by the number of frames, including linked correction files.
- HIS_SEQ_DEST_ONE_FRAME 0x8 Sequence of frames using the same image buffer
- HIS_SEQ_COLLATE 0x10 Skip frames after acquiring frames in a ring buffer
- HIS_SEQ_CONTINUOUS 0x100 Continuous acquisition Frames are continuously acquired into a ring buffer of dwFrames

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.10 HIS_RETURN **Acquisition_Acquire_Image_PreloadCorr (HACQDESC *hAcqDesc,* UINT *dwFrames,* UINT *dwSkipFrms,* UINT *dwOpt*)**

The function provides the same functionality as Acquisition_Acquire_Image(. . .) except loading the image correction data. Use Acquisition_SetCorrData_Ex before to set the correction data.

Note

Please find the parameter description at Acquisition_Acquire_Image(..)

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.11 HIS_RETURN **Acquisition_Acquire_OffsetImage (HACQDESC *hAcqDesc,* WORD ∗ *pwOffsetData,* UINT *nRows,* UINT *nColumns,* UINT *nFrames*)**

This function acquires nFrames, adds them in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). The last acquired data at frame end time are available via pOffsetData. At the end of the acquisition time the averaged data are also accessible via pOffsetData. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pwOffset-Data	Pointer to a acquisition buffer for offset data.
nRows	Number of rows of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nColumns	Number of columns of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nFrames	Number of frames to acquire.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.12 HIS_RETURN Acquisition_Acquire_OffsetImage_Ex (HACQDESC *hAcqDesc*, WORD ∗ *pwOffsetData*, UINT *nRows*, UINT *nColumns*, UINT *nFrames*, UINT *dwOpt*)

This function acquires nFrames, adds them in a 32 bit buffer and after acquisition the data values are divided by nFrames (averaging). The last acquired data at frame end time are available via pOffsetData. At the end of the acquisition time the averaged data are also accessible via pOffsetData. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init the suitable Callback functions and post from there a corresponding message to your application.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pwOffset-Data	Pointer to a acquisition buffer for offset data.
nRows	Number of rows of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nColumns	Number of columns of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nFrames	Number of frames to acquire.
dwOpt	

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.13 HIS_RETURN **Acquisition_Acquire_OffsetImage_PreloadCorr** (HACQDESC *hAcqDesc,* WORD ∗ *pwOffsetData,* UINT *nRows,* UINT *nColumns,* UINT *nFrames,* UINT *dwOpt*)

The function provides the same functionality as Acquisition_Acquire_Offset_Image(. . .) except loading the image correction data. Use Acquisition_SetCorrData_Ex before to all correction data to NULL.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pwOffset-Data	Pointer to a acquisition buffer for offset data.
nRows	Number of rows of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nColumns	Number of columns of the offset data buffer. If the value is not suitable to the current connected sensor the function return with an error.
nFrames	Number of frames to acquire.
dwOpt	

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.5.1.14 HIS_RETURN **Acquisition_CreateGainMap** (WORD ∗ *pGainData,* WORD ∗ *pGainAVG,* int *nCount,* int *nFrame*)

This function creates a list of median values for gain correction using an bright offet corr image to be used with the function Acquisition_Acquire_Image_Ex, Acquisition_-SetCorrData_Ex or Acquisition_DoOffsetGainCorrection_Ex. One value for each image in the pGainData-sequence.

Parameters

pGainData	pointer to a sequence of offset corrected data. The images in the array must be sorted by median.
pGainAVG	pointer to a nFrame sized array of type word
nCount	Number of pixels to per image.
nFrame	Number of Frames in pGainData

Returns

If the function is successful it returns true, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.15 HIS_RETURN **Acquisition_CreateGainMap32** (unsigned long ∗ *pGainData,* unsigned long ∗ *pGainAVG,* int *nCount,* int *nFrame*)

This function creates a list of median values to be used with the function Acquisition_-Acquire_Image_Ex . One value for each image in the pGainData-sequence.

Parameters

pGainData	Pointer to a sequence of offset corrected data. The images in the array must be sorted by median.
pGainAVG	Pointer to a nFrame sized array of type word
nCount	Number of pixels to per image.
nFrame	Number of Frames in pGainData

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.16 HIS_RETURN **Acquisition_CreateOnboardPixelMaskFrom16BitPixelMask** (unsigned short ∗ *uspPixelMaskSrc,* DWORD *dwRows,* DWORD *dwColumns,* unsigned char ∗ *bpOnboardPixelMask*)

This function creates an on-board style (i.e. 1 bit/pixel) pixel mask from a standard (16 bit/pixel) pixel mask image to be used on XRpad2 detectors.

Parameters

uspPixel-MaskSrc	Input (16 bit/pixel) pixel mask image buffer
dwRows	Number of rows in the image
dwColumns	Number of columns in the image
bpOnboard-PixelMask	Output, on-board style (1 bit/pixel) pixel mask image buffer

Returns

HIS_ALL_OK

3.5.1.17 HIS_RETURN Acquisition_CreatePixelMap (WORD ∗ *pwData,* int *nDataRows,* int *nDataColumns,* int ∗ *pCorrList,* int ∗ *pnCorrListSize*)

This function specifies the size of or creates a pixel correction list (depending on pw-Data) that one can use in Acquisition_Acquire_Image or Acquisition_DoPixelCorrection.

Parameters

pwData	Pointer to a data source buffer. Defective pixels are marked by setting their value to 1 (0xFFFF). All other pixel values are recognized as good ones.
nDataRows	Number of rows of the data source.
nData-Columns	Number of columns of the data source.
pCorrList	Pointer to an array (pixel map buffer) that receives the pixel correction data. If pCorrlist is set to NULL then only the required size of the pixel map buffer is returned in pnCorrListSize.
pnCorrList-Size	Pointer to an integer that receives the required size of the pixel map buffer if pCorrList is set to NULL otherwise it contains the size of the pixel buffer at function call time.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.18 HIS_RETURN Acquisition_DefineDestBuffers (HACQDESC *hAcqDesc,* unsigned short ∗ *pProcessedData,* UINT *nFrames,* UINT *nRows,* UINT *nColumns*)

This function defines the pointers of the destination buffer for Acquisition_Acquire_-Image. The data are written into this buffer after sorting. The buffer must have a proper size depending on acquisition mode. To acquire one image the buffer must have the size sensor rows ∗ sensor columns ∗2. To acquire a sequence the buffer must have the size sensor rows ∗ sensor columns ∗2 ∗ frames. In the case of continuous acquisition the buffer must have the size sensor rows ∗ sensor columns ∗2 ∗ frames of the ring buffer.

Parameters

hAcqDesc	Pointer to HACQDESC.
pProcessed-Data	Pointer to the destination buffer.
nFrames	Number of frames of the destination buffer. It must be greater than zero.

nRows	Number of rows of the destination buffer. If this number is not suitable to the sensor the function return with an error code. If you need extended information call Acquisition_GetErrorCode.
nColumns	Number of columns of the destination buffer. If this number is not suitable to the sensor the function return with an error code. If you need extended information call Acquisition_GetErrorCode.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.19 HIS_RETURN **Acquisition_DoOffsetCorrection** (WORD ∗ *pSource,* WORD ∗ *pDest,* WORD ∗ *pOffsetData,* int *nCount*)

This function performs an offset correction for the data defined in pSource. A suitable place to call this function is the end of frame callback function or the end of acquisition callback.

Parameters

pSource	Pointer to data source buffer.
pDest	Pointer to data destination buffer. This parameter can be equal to p-Source.
pOffsetData	Pointer that contains offset data. (see Acquisition_Acquire_Offset-Image). These data should be lately acquired, so that it is up-to-date. It is recommended to acquire them shortly before calling Acquisition_Do-OffsetCorrection.
nCount	Number of pixels to correct.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.5.1.20 HIS_RETURN **Acquisition_DoOffsetGainCorrection** (WORD ∗ *pSource,* WORD ∗ *pDest,* WORD ∗ *pOffsetData,* DWORD ∗ *pGainData,* int *nCount*)

This function performs an offset and a gain correction for the data defined in pSource at once. A suitable place to call this function is the end of frame callback function or the end acq callback.

Parameters

pSource	Pointer to data source buffer.
pDest	Pointer to data destination buffer. This parameter can be equal to p-Source.

pOffsetData	Pointer that contains offset data. (see Acquisition_Acquire_Offset-Image). These data should be lately acquired, so that it is up-to-date. It is recommended to acquire them shortly before calling Acquisition_Do-OffsetCorrection.
pGainData	Pointer that contains gain data. (see Acquisition_Acquire_GainImage).
nCount	Number of data entries to correct.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.5.1.21 HIS_RETURN **Acquisition_DoOffsetGainCorrection32** (unsigned long ∗ ***pSource,*** unsigned long ∗ ***pDest,*** unsigned long ∗ ***pOffsetData,*** unsigned long ∗ ***pGainData,*** int ***nCount***)

Similar to Acquisition_DoOffsetGainCorrection.

3.5.1.22 HIS_RETURN **Acquisition_DoOffsetGainCorrection_Ex** (WORD ∗ ***pSource,*** WORD ∗ ***pDest,*** WORD ∗ ***pOffsetData,*** WORD ∗ ***pGainData,*** WORD ∗ ***pGainAVG,*** int ***nCount,*** int ***nFrame***)

This function performs an offset and a multi gain correction using offset corrected bright images as gain for the data defined in pSource at once. A suitable place to call this function is the end of frame callback function or the end of acq callback.

Parameters

pSource	Pointer to data source buffer.
pDest	Pointer to data destination buffer. This parameter can be equal to p-Source.
pOffsetData	Pointer that contains offset data. (see Acquisition_Acquire_Offset-Image). These data should be lately acquired, so that it is up-to-date. It is recommended to acquire them shortly before calling Acquisition_Do-OffsetCorrection.
pGainData	Pointer to a sequence of offset corrected data. The images in the array must be sorted by median.
pGainAVG	Pointer to the median-list created by Acquisition_CreateGainMap
nCount	Number of pixels to correct.
nFrame	Number of frames in pGainData

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.5.1.23 HIS_RETURN Acquisition_DoPixelCorrection (WORD ∗ *pData,* int ∗ *pCorrList*)

This function performs a defect pixel correction on the data defined by pData using the pCorrList. A suitable place to call this function is the end of frame callback or after the end acq callback.

Parameters

pData	Pointer to data.
pCorrList	Pointer that contains correction data. pCorrList points to a linear array of data. Its size is given through ((number of pixels) ∗ 9 + 1). The first entry in a group of nine contains the offset of the pixel from the base pointer of the data array. The other eight entries equal the offset of the correction pixels from the base pointer. If you want to use less than eight pixels for correction, then set the remaining entries to -1. The value of the pixel is replaced by the mean value of the correction pixels. The end of the pixel correction list is indicated by a value of 1 as the last entry in the pixel map.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.24 HIS_RETURN Acquisition_GetCorrData (HACQDESC *hAcqDesc,* unsigned short ∗∗ *ppwOffsetData,* DWORD ∗∗ *ppdwGainData,* DWORD ∗∗ *ppdwPxlCorrList*)

This function returns the adresses of the applied correction buffers.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
ppwOffset-Data	Point to offset data.
ppdwGain-Data	Pointer to gain data.
ppdwPxl-CorrList	Pointer to pixel correction list.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.25 HIS_RETURN Acquisition_GetCorrData_Ex (HACQDESC *hAcqDesc,* unsigned short ∗∗ *ppwOffsetData,* unsigned short ∗∗ *ppwGainData,* unsigned short ∗∗ *ppwGainAvgData,* UINT ∗∗ *nGainFrames,* DWORD ∗∗ *ppdwGainData,* DWORD ∗∗ *ppdwPxlCorrList*)

This function retrieves the current correction data during a running acquisition.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
ppwOffset-Data	Point to offset data. (see Acquisition_Acquire_OffsetImage).
ppwGain-Data	Pointer to a sequence of offset corrected data. The images in the array must be sorted by median
ppwGain-AvgData	Pointer to the median-list created by Acquisition_CreateGainMap
nGain-Frames	UINT ∗pointer retrieving the number of frames in pwGainData
ppdwGain-Data	Pointer that contains gain data (see Acquisition_Acquire_GainImage).
ppdwPxl-CorrList	Pointer to a pixel correction list (see Acquisition_DoPixelCorrection)

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.26 HIS_RETURN Acquisition_GetLatestFrameHeader (HACQDESC *hAcqDesc,* CHwHeaderInfo ∗ *pInfo,* CHwHeaderInfoEx ∗ *pInfoEx*)

Use this function to retrieve the last acuiered frame header of the connected detector. If dwHeaderID in the CHwHeaderInfo structure is 14 pInfoEx will retrieve the extended header, otherwise the structure will be filled with 0xFFFF.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pInfo	Pointer to Structure of type CHwHeaderInfo to retrieve the detector's hardware header
pInfoEx	Pointer to Structure of type CHwHeaderInfoEx to retrieve the detector's hardware header when available. Can be NULL

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.27 HIS_RETURN Acquisition_SetCorrData (HACQDESC *hAcqDesc*, unsigned short ∗ *pwOffsetData*, DWORD ∗ *pdwGainData*, DWORD ∗ *pdwPxlCorrList*)

This function switches the correction buffers during a running acquisition. It is also possible to switch off corrections by setting the pOffsetData, pGainData and pPixelCorrList to NULL.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
pwOffset-Data	Point to offset data (see Acquisition_Acquire_OffsetImage).
pdwGain-Data	Pointer that contains gain data (see Acquisition_Acquire_GainImage).
pdwPxlCorr-List	Pointer to a pixel correction list (see Acquisition_DoPixelCorrection).

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.5.1.28 HIS_RETURN Acquisition_SetCorrData_Ex (HACQDESC *hAcqDesc*, unsigned short ∗ *pwOffsetData*, unsigned short ∗ *pwGainData*, unsigned short ∗ *pwGainAvgData*, UINT *nGainFrames*, DWORD ∗ *pdwGainData*, DWORD ∗ *pdwPxlCorrList*)

This function switches the correction buffers during a running acquisition. It is also possible to switch off corrections by setting the pOffsetData, pGainData pwGainData, pwGainAvgData, nGainFrames, and pPixelCorrList to NULL.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
pwOffset-Data	Pointer to offset data (see Acquisition_Acquire_OffsetImage).
pwGainData	Pointer to a sequence of offset corrected data. The images in the array must be sorted by median.
pwGainAvg-Data	Pointer to the median-list created by Acquisition_CreateGainMap
nGain-Frames	number of frames in pwGainData
pdwGain-Data	Pointer that contains gain data (see Acquisition_Acquire_GainImage).

pdwPxlCorr-List	Pointer to a pixel correction list (see Acquisition_DoPixelCorrection)

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6 Setting up the Device

Functions

- HIS_RETURN Acquisition_GetCameraBinningMode (HACQDESC hAcqDesc, -WORD ∗wMode)

  Use this function to retrieve the detectors binning mode.
- HIS_RETURN Acquisition_GetCameraFOVMode (HACQDESC hAcqDesc, WO-RD ∗wMode)

  Use this function to retrieve the detectors FOV mode.
- HIS_RETURN Acquisition_GetCameraROI (HACQDESC hAcqDesc, unsigned short ∗usActivateGrp)

  This function returnes the currently activated Region of Interest in Sectional Readout Mode.
- HIS_RETURN Acquisition_GetCameraTriggerMode (HACQDESC hAcqDesc, W-ORD ∗wMode)

  Retreives TriggerMode for Detectors with Header >14 and Detectortype >2.
- HIS_RETURN Acquisition_GetIpAdress (HACQDESC hAcqDesc, const char ∗∗ipAddress)

  This function retrieves the current ip address of the detector (if any).
- HIS_RETURN Acquisition_ResetFrameCnt (HACQDESC hAcqDesc)

  This function is used to set internal frame counter to zero. Note: If this function is used during active acquisition the detector Readout time may be affected. (Only XISL version > 3-2-0-9, HeaderId 14 and Cameratype 1 and 2)
- HIS_RETURN Acquisition_SetCameraBinningMode (HACQDESC hAcqDesc, -WORD wMode)

  Use this function to set the detectors binning mode. Not all detectors support image binning. Please refer to the detector manual for a table of supported binning modes.
- HIS_RETURN Acquisition_SetCameraFOVMode (HACQDESC hAcqDesc, WO-RD wMode)

  This function selects the field of view for XRD 4343RF detectors.
- HIS_RETURN Acquisition_SetCameraGain (HACQDESC hAcqDesc, WORD w-Mode)

  This function can be used to select the gain of the detector.
- HIS_RETURN Acquisition_SetCameraMode (HACQDESC hAcqDesc, UINT dw-Mode)

  This function sets the acquisition timing mode of the detector. Currently eight fixed frame times of the detector are provided.
- HIS_RETURN Acquisition_SetCameraROI (HACQDESC hAcqDesc, unsigned short usActivateGrp)

  This function selects a Defined Region of Interest for Readout.
- HIS_RETURN Acquisition_SetCameraTriggerMode (HACQDESC hAcqDesc, W-ORD wMode)

  The function sets the internal trigger scheme for Detectors.
- HIS_RETURN Acquisition_SetDACoffset (HACQDESC hAcqDesc, WORD wDA-CoffsetValue)

Use this function to sets the DAC for offset floor level within the detector for XRD 4343 detectors.

- HIS_RETURN Acquisition_SetDACoffsetBinningFPS (HACQDESC hAcqDesc, -WORD wBinningMode, double dblFps, WORD ∗pwValueToFPGA)

  XRD 4343 only. Use this function to sets the DAC offset value within the detector by passing the expected frames/sec and the used binning mode This function may only be used for evalutation purpose. Please contact the application team to get information about how to set the DACOffset.
- HIS_RETURN Acquisition_SetFrameSyncMode (HACQDESC hAcqDesc, DWO-RD dwMode)

  This function sets the synchronization mode of the detector.
- HIS_RETURN Acquisition_SetFrameSyncTimeMode (HACQDESC hAcqDesc, unsigned int uiMode, unsigned int dwDelayTime)

  This function sets the synchronization mode of the detector to triggered mode and sets the delay time for "DDD/AED " triggered mode with defined integration time.
- HIS_RETURN Acquisition_SetTimerSync (HACQDESC hAcqDesc, DWORD ∗dwCycleTime)

  This function configures the CycleTime for internal triggered mode.
- HIS_RETURN Acquisition_SetTriggerOutSignalOptions (HACQDESC hAcq-Desc, unsigned short usTiggerOutSignalMode, unsigned short usEP_Seq-Length, unsigned short usEP_FirstBrightFrm, unsigned short usEP_LastBright-Frm, unsigned short usEP_Delay1, unsigned short usEP_Delay2, unsigned short usDDD_Delay, int iTriggerOnRisingEdgeEnable, int iSaveAsDefault)

  This function defines behavior of the '/TrigOut' - signal of the detector trigger connector and defines the exposure delay.

3.6.1 Function Documentation

3.6.1.1 HIS_RETURN Acquisition_GetCameraBinningMode (HACQDESC *hAcqDesc*, WORD ∗ *wMode*)

Use this function to retrieve the detectors binning mode.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	Pointer to value to retrieve the actual binning mode.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.2 HIS_RETURN Acquisition_GetCameraFOVMode (HACQDESC *hAcqDesc*, WORD ∗ *wMode*)

Use this function to retrieve the detectors FOV mode.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	Pointer to value to retrieve the actual Field Of View mode.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.3 HIS_RETURN Acquisition_GetCameraROI (HACQDESC *hAcqDesc,* unsigned short ∗ *usActivateGrp*)

This function returnes the currently activated Region of Interest in Sectional Readout Mode.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
usActivate-Grp	pointer to unsigend short value representing a bitwise Selection of -Groups to Readout and Transmit

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.4 HIS_RETURN Acquisition_GetCameraTriggerMode (HACQDESC *hAcqDesc,* WORD ∗ *wMode*)

Retreives TriggerMode for Detectors with Header >14 and Detectortype >2.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	Command 42: • 0 - DDD • 1 - DDD without clearance scan • 2 - Start Stop • [3] - Trigger Frames (Triggern 1.Art)

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.5 HIS_RETURN **Acquisition_GetIpAdress** (HACQDESC *hAcqDesc*, const char $**$ *ipAddress*)

This function retrieves the current ip address of the detector (if any).

Parameters

`in`	*hAcqDesc*	Pointer to acquisition descriptor structure
`out`	*ipAddress*	Pointer to character array. This will be point to a statically allocated chunk of memory, that may be invalidated after subsequent or concurrent calls.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.6.1.6 HIS_RETURN **Acquisition_ResetFrameCnt** (HACQDESC *hAcqDesc*)

This function is used to set internal frame counter to zero. Note: If this function is used during active acquisition the detector Readout time may be affected. (Only XISL version $>$ 3-2-0-9, HeaderId 14 and Cameratype 1 and 2)

Parameters

hAcqDesc	Pointer to acquisition descriptor structure

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.7 HIS_RETURN **Acquisition_SetCameraBinningMode** (HACQDESC *hAcqDesc*, WORD *wMode*)

Use this function to set the detectors binning mode. Not all detectors support image binning. Please refer to the detector manual for a table of supported binning modes.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	Binning Mode to be set • value 1 : no binning (default) • value 2 : 2x2 binning • value 3 : 4x4 binning (3x3 R&F) • value 4 : 1x2 binning • value 5 : 1x4 binning

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.6.1.8 HIS_RETURN Acquisition_SetCameraFOVMode (HACQDESC *hAcqDesc,* WORD *wMode*)

This function selects the field of view for XRD 4343RF detectors.

2013-04-22 Val GetTriggerStatus GbIF

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	Binning Mode to be set to value, see table

Note

- value 1 : no FOV (default) 2880 x 2880 binning1 1440 x 1440 binning2 960 x 960 binning3
- value 2 : 1920 x 1920 binning1 960 x 960 binning2 640 x 640 binning3
- value 3 : 1440 x 1440 binning1 720 x 736 binning2 480 x 480 binning3
- value 4 : 1440 x 2880 binning1 720 x 1440 binning2 480 x 960 binning3
- value 5 : 480 x 2880 binning1 240 x 1440 binning2 160 x 960 binning3
- Refer to manual for a FOV mode table.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.9 HIS_RETURN Acquisition_SetCameraGain (HACQDESC *hAcqDesc,* WORD *wMode*)

This function can be used to select the gain of the detector.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	Gain factor to set. For the XRpad and XRD 4343RF please refer to the detector reference manual. For the AM-Type the values of all capacities are added. All bitwise combinations are valid. For example: 3 => 1.3p-F. For the xN/xO/xP-Type the Value in the table is set when the detector provides the functionality (refer to detector specification).

Note

Detector xN/xO/xP

- 0 0.25pF
- 1 0.5pF
- 2 1 pF
- 3 2 pF
- 4 4 pF
- 5 8 pF

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.10 HIS_RETURN Acquisition_SetCameraMode (HACQDESC *hAcqDesc,* UINT *dwMode*)

This function sets the acquisition timing mode of the detector. Currently eight fixed frame times of the detector are provided.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwMode	Must be a number between 0 and 7. The corresponding frame time depends on the used detector setting.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.6.1.11 HIS_RETURN Acquisition_SetCameraROI (HACQDESC *hAcqDesc,* unsigned short *usActivateGrp*)

This function selects a Defined Region of Interest for Readout.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
usActivate-Grp	Bitwise Selection of Groups to Readout and Transmit

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.12 HIS_RETURN **Acquisition_SetCameraTriggerMode** (**HACQDESC** *hAcqDesc,* WORD *wMode*)

The function sets the internal trigger scheme for Detectors.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wMode	see table

Note

Command 42:

- 0 - WM / DDD
- 1 - WM / DDD without clearance scan
- 2 - Start Stop
- [3] - Trigger Frames (Triggern 1.Art)
- 4 - AutoTrigger Frames
- 5 - Trigger on Row Tag (Framewise with Filter on Trigger input) / RnF frame-Wise flow controlled
- 6 - single shot with post Offset by single trigger (second trigger aborts readout of bright image) - XRPAD2 only
- 7 - dual energy with post offset - XRPAD2 only The selected mode will be used once the detector is set to triggered mode using Acquisition_SetFrame-SyncMode(..)

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.6.1.13 HIS_RETURN **Acquisition_SetDACoffset** (**HACQDESC** *hAcqDesc,* WORD *wDACoffsetValue*)

Use this function to sets the DAC for offset floor level within the detector for XRD 4343 detectors.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wDACoffset-Value	wADCOffsetValue to be set. This value has to be between 1 and 4095

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.14 HIS_RETURN Acquisition_SetDACoffsetBinningFPS (HACQDESC *hAcqDesc,* WORD *wBinningMode,* double *dblFps,* WORD ∗ *pwValueToFPGA*)

XRD 4343 only. Use this function to sets the DAC offset value within the detector by passing the expected frames/sec and the used binning mode This function may only be used for evalutation purpose. Please contact the application team to get information about how to set the DACOffset.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
wBinning-Mode	current binning mode
dblFps	number of frames per sec
pwValueTo-FPGA	pointer to a variable to retrieve the value send to the fpga

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.15 HIS_RETURN Acquisition_SetFrameSyncMode (HACQDESC *hAcqDesc,* DWORD *dwMode*)

This function sets the synchronization mode of the detector.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwMode	This parameter can values see table

Note

dwMode

- HIS_SYNCMODE_FREE_RUNNING - no sync, the detector will run without trigger / power save mode in XRpad detectors
- HIS_SYNCMODE_EXTERNAL_TRIGGER - the detector waits for an external signal for readout
- HIS_SYNCMODE_INTERNAL_TIMER - the detector will be triggered by an internal generator (interval to be set with Acquisition_SetTimerSync)

- HIS_SYNCMODE_SOFT_TRIGGER - the detector will be triggered by an internal signal generated via software (Acquisition_SetFrameSync)
- HIS_SYNCMODE_AUTO_TRIGGER - the detector will read out an image when xr-ray exposure is detected
- HIS_SYNCMODE_EXTERNAL_TRIGGER_FG - the external trigger port of the framegrabber will be used as trigger input

HIS_SYNCMODE_EXTERNAL_TRIGGER_FG is requiered by RnF and 1611 detecors to route the external triggersignal attached to the frame grabber to the detector

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.16 HIS_RETURN Acquisition_SetFrameSyncTimeMode (HACQDESC *hAcqDesc*, unsigned int *uiMode*, unsigned int *dwDelayTime*)

This function sets the synchronization mode of the detector to triggered mode and sets the delay time for "DDD/AED " triggered mode with defined integration time.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
uiMode	Timing Mode must be 0 .. 7
dwDelay-Time	Additional delay time in milliseconds (can be 0-ms up to (5000 ms-int-Time)

```
// set special triggermode to 'Data Delivered on demand without clearance scan'
Acquisition_SetCameraTriggerMode (hAcqDesc,1);


// set Framegrabber to Soft_Trigger-Mode
Acquisition_SetFrameSyncMode(hAcqDesc,HIS_SYNCMODE_SOFT_TRIGGER);

//set detector to timing 0 delay 1sec
Acquisition_SetFrameSyncTimeMode(hAcqDesc,0,1000);

hevEndAcq = CreateEvent(NULL, FALSE, FALSE, NULL);
if ((nRet=Acquisition_Acquire_Image(hAcqDesc, 1, 0, HIS_SEQ_ONE_BUFFER, NULL,
      NULL, NULL))!=HIS_ALL_OK)
{
    //error handling
}

//  start the readout
Acquisition_SetFrameSync(hAcqDesc);
```

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.17 HIS_RETURN Acquisition_SetTimerSync (HACQDESC *hAcqDesc*, DWORD ∗ *dwCycleTime*)

This function configures the CycleTime for internal triggered mode.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwCycle-Time	Pointer to a 32 bit integer that provides the required cycle time in microsec. After returning of the function this parameter contains the realized cycle time. Before calling this function you have to set the frame grabber to a suitable synchronization mode by a call of Acquisition_-SetFrameSyncMode. Some frame grabbers can realize synchronization cycles only in discreet steps. Thats why the function returns the realized cycle time in dwCycleTime.

The parameter dwCycleTime should typically be smaller or equal to 5000000 microsec.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.6.1.18 HIS_RETURN Acquisition_SetTriggerOutSignalOptions (HACQDESC *hAcqDesc*, unsigned short *usTiggerOutSignalMode*, unsigned short *usEP_SeqLength*, unsigned short *usEP_FirstBrightFrm*, unsigned short *usEP_LastBrightFrm*, unsigned short *usEP_Delay1*, unsigned short *usEP_Delay2*, unsigned short *usDDD_Delay*, int *iTriggerOnRisingEdgeEnable*, int *iSaveAsDefault*)

This function defines behavior of the '/TrigOut' - signal of the detector trigger connector and defines the exposure delay.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
usTigger-OutSignal-Mode	see table
usEP_Seq-Length	Defines the Sequence Length for EP mode
usEP_First-BrightFrm	Defines the First Frame in the Sequence where the TrigOut Signal is activated
usEP_Last-BrightFrm	Defines the Last Frame in the Sequence where the TrigOut Signal is activated

usEP_-Delay1	Defines the Delay1 from Begin of Frame in the EP_Frames until the TrigOut Signal gets activated
usEP_-Delay2	Defines the Delay2 from Begin of Frame in the EP_Frames until the TrigOut Signal gets deactivated
usDDD_-Delay	Additional Delay in DDD/AED/Phototimed Mode
iTriggerOn-RisingEdge-Enable	• 0 -Trigger falling Edge

- 1 -trigger on rising edge

Parameters

iSaveAs-Default	1 - Save usTiggerOutSignalMode and iTriggerOnRisingEdgeEnable in EEPROM to be availabel after the next power cycle

Note

trigger out signal modes: 0 - FRM_EN_PWM 1 - FRM_EN_PWM_INV 2 - EP 3 - EP_INV 4 - DDD_Pulse 5 - DDD_Pulse_INV 6 - GND 7 - VCC

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.7 Callback function

Functions

- HIS_RETURN Acquisition_GetAcqData (HACQDESC hAcqDesc, DWORD ∗dw-AcqData)

  This routine returns a pointer to value that can be set by Acquisition_SetAcqData. - These two functions are useful to avoid global variables to put through parameters to the end of frame and end of acquisition callback functions setted by Acquisition_Init.
- HIS_RETURN Acquisition_GetActFrame (HACQDESC hAcqDesc, DWORD ∗dwActAcqFrame, DWORD ∗dwActSecBuffFrame)

  This function retrieves the current acquisition frames.
- HIS_RETURN Acquisition_GetReady (HACQDESC hAcqDesc)

  This function checks whether the passed Acquisition Descriptor is valid.
- HIS_RETURN Acquisition_GetWinHandle (HACQDESC hAcqDesc, HWND ∗h-Wnd)

  This function retrieves the current acquisition window handle if defined before.
- HIS_RETURN Acquisition_SetAcqData (HACQDESC hAcqDesc, DWORD dw-AcqData)

  This routine sets a value that can be received with Acquisition_GetAcqData. These two functions are useful to avoid global variables to put through parameters to the end of frame and end of acquisition callback functions setted by Acquisition_Init.
  You can use this function to pass parameters to your callback routines. Thereby it is possible to avoid global variables. The recommended way of use is to define a data structure/value/class and to pass the address of an instance of it by Acquisition_Set-AcqData. This has the advantage that a variety of parameters, which are defined within the structure/value/class, are accessible within the callback functions.
- HIS_RETURN Acquisition_SetReady (HACQDESC hAcqDesc, BOOL bFlag)

  This function must be called when the application finished redrawing of the new acquired data. A good place to call this function is the end of frame callback function defined in Acquisition_Init or the message handler to for the redraw message after redrawing.

3.7.1 Function Documentation

3.7.1.1 HIS_RETURN **Acquisition_GetAcqData (HACQDESC** ***hAcqDesc,*** DWORD ∗ ***dwAcqData***)

This routine returns a pointer to value that can be set by Acquisition_SetAcqData. - These two functions are useful to avoid global variables to put through parameters to the end of frame and end of acquisition callback functions setted by Acquisition_Init.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwAcqData	Data to be received. To put through more than one parameters define a structure with the required parameters and cast the pointer to dwData.

Note

For the 64bit version of the XiSL.dll the return type of this parameter has changed to void∗

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.7.1.2 HIS_RETURN **Acquisition_GetActFrame** (**HACQDESC** ***hAcqDesc,*** DWORD ∗ ***dwActAcqFrame,*** DWORD ∗ ***dwActSecBuffFrame***)

This function retrieves the current acquisition frames.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwActAcq-Frame	Actual frame of acquisition buffer. The acquisition buffer is allocated internally by the frame grabber driver and is not accessible externally. However, the index can be used to verify the consistency of image sequences; it runs repeatently from 1 to 8.
dwActSec-BuffFrame	Actual frame for second buffer that is needed to acquire sequences of averaged images. It is the frame count of the acquisition buffer defined by Acqusition_DefineDestBuffers.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.7.1.3 HIS_RETURN **Acquisition_GetReady** (**HACQDESC** ***hAcqDesc***)

This function checks whether the passed Acquisition Descriptor is valid.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.7.1.4 HIS_RETURN **Acquisition_GetWinHandle** (**HACQDESC** ***hAcqDesc,*** HWND ∗ ***hWnd***)

This function retrieves the current acquisition window handle if defined before.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
hWnd	Retrieves the window handle defined in Acquisition_Init.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.7.1.5 HIS_RETURN Acquisition_SetAcqData (HACQDESC *hAcqDesc,* DWORD *dwAcqData*)

This routine sets a value that can be received with Acquisition_GetAcqData. These two functions are useful to avoid global variables to put through parameters to the end of frame and end of acquisition callback functions setted by Acquisition_Init.

You can use this function to pass parameters to your callback routines. Thereby it is possible to avoid global variables. The recommended way of use is to define a data structure/value/class and to pass the address of an instance of it by Acquisition_Set-AcqData. This has the advantage that a variety of parameters, which are defined within the structure/value/class, are accessible within the callback functions.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwAcqData	32bit: Data to be accessible within callback functions. dwData can represent a single value as well as an address. In that case a structure/-value/class has to be defined with the required parameters and cast dwData to the pointer. The value/ address can be retrieved in the End-FrameCallback/ EndAcqCallback by using Acquisition_GetAcqData(..). 64bit: Data to be accessable within callback functions: Pass a pointer to a user defined structure/value/class which can be retrieved in the End-FrameCallback/ EndAcqCallback by using Acquisition_GetAcqData(..)

Note

For the 64bit version of the XiSL.dll the return type of this parameter has changed to void∗

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.7.1.6 HIS_RETURN Acquisition_SetReady (HACQDESC *hAcqDesc,* BOOL *bFlag*)

This function must be called when the application finished redrawing of the new acquired data. A good place to call this function is the end of frame callback function defined in Acquisition_Init or the message handler to for the redraw message after redrawing.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
bFlag	Boolean value. Set to zero to signal XISL set redrawing isn't ready, otherwise set to one.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.8 Common error handling

Functions

- HIS_RETURN Acquisition_GetErrorCode (HACQDESC hAcqDesc, DWORD ∗dwHISErr, DWORD ∗dwBoardErr)

  The function returns extended information if an error occurred during a XISL function call.
- HIS_RETURN Acquisition_wpe_GetErrorCode ()

  This function retrieves the last error code from the wpe200.dll.
- HIS_RETURN Acquisition_wpe_GetErrorCodeEx (char ∗pBuffer, long len)

  This function retrieves the last error string from the wpe200.dll.

3.8.1 Function Documentation

3.8.1.1 HIS_RETURN **Acquisition_GetErrorCode** (HACQDESC *hAcqDesc,* DWORD ∗ *dwHISErr,* DWORD ∗ *dwBoardErr*)

The function returns extended information if an error occurred during a XISL function call.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
dwHISErr	Retrieves an error code regarding the XISL itself.
dwBoardErr	Retrieves an error code regarding the acquisition board. Please consult the corresponding documentation of your data acquisition board.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.8.1.2 HIS_RETURN **Acquisition_wpe_GetErrorCode** (void)

This function retrieves the last error code from the wpe200.dll.

Returns

Latest error code

3.8.1.3 HIS_RETURN **Acquisition_wpe_GetErrorCodeEx** (char ∗ *pBuffer,* long *len*)

This function retrieves the last error string from the wpe200.dll.

Parameters

pBuffer	Pointer to a character array to receive the latest error message. (char[256] recommended).
len	length of the passed character array

Returns

HIS_ALL_OK when function call was successfull otherwise an error code

3.9 functions provided by Acq.h

Functions

- HIS_RETURN Acq_wpe_GetSystemInformation (const char ∗ipAddress, char ∗buffer, int bufferLen)

  This function is used to retrieve system information from the detector.
- HIS_RETURN Acq_WPE_Init ()

  Acq_WPE_Init called when dll is loaded, intialize wpe, global objects, etc...
- HIS_RETURN Acquisition_ActivateServiceMode (HACQDESC hAcqDesc, BOOL bActivate)

  This function activates the Service Mode, which means Service Data is written into acquired images. This makes it easy to provide images for support.
- HIS_RETURN Acquisition_CreateXISFileInMemory (void ∗pMemoryFileBuffer, void ∗pDataBuffer, UINT dwRows, UINT dwColumns, UINT dwFrames, BOOL uiOnboardFileHeader, XIS_FileType filetype)

  This function creates an XIS image file (including header) in the given memory buffer.
- HIS_RETURN Acquisition_DeleteFile (XislFileHandle fileHandle)

  Acquisition_DeleteFile Deletes a file from a location.
- HIS_RETURN Acquisition_DoOffsetCorrection32 (unsigned long ∗pSource, unsigned long ∗pDest, unsigned long ∗pOffsetData, int nCount)

  Similar to Acquisition_DoOffsetCorrection.
- HIS_RETURN Acquisition_DoOffsetGainCorrection_Ex32 (unsigned long ∗pSource, unsigned long ∗pDest, unsigned long ∗pOffsetData, unsigned long ∗pGainData, unsigned long ∗pGainAVG, int nCount, int nFrame)

  Similar to Acquisition_DoOffsetGainCorrection_Ex.
- HIS_RETURN Acquisition_GetConnectionStatus (HACQDESC hAcqDesc)

  retrievs the status of the current connection (if supported).
- HIS_RETURN Acquisition_GetDACOffsetFloorValueFromFlash (HACQDESC hAcqDesc, unsigned int uiMode, WORD ∗pwValue)

  This function reads the desired offset value from the detector memory for the selected mode. It must be set using Acquisition_SetDACoffset or Acquisition_SetDACOffsetFloorValueByMode.
- HIS_RETURN Acquisition_GetDetectorProperties (HACQDESC hAcqDesc, GBIF_Detector_Properties ∗pDetectorProperties)

  This function retrieved the GBIF_Detector_Properties structure, which contains permanently stored information of the connected device.
- HIS_RETURN Acquisition_GetFileInfo (XislFileHandle fileHandle, XislFileInfo ∗fileInfo)

  Acquisition_GetFileInfo Retrieves basic information about a file.
- HIS_RETURN Acquisition_GetRotationAngle (HACQDESC hAcqDesc, long ∗lRotAngle)

  Get onboard Rotation Setting for FG-E Opto.
- HIS_RETURN Acquisition_GetTriggerOutStatus (HACQDESC hAcqDesc, int ∗iTriggerStatus)

  This function retrieves the triggerout status from detector.

- HIS_RETURN Acquisition_GetVersion (int ∗major, int ∗minor, int ∗release, int ∗build)

  reads the version of the xisl dll
- HIS_RETURN Acquisition_GetXISFileBufferSize (size_t ∗pFileSize, UINT dwRows, UINT dwColumns, UINT dwFrames, BOOL uiOnboardFileHeader, XIS_FileType filetype)

  This function returns the requiered size(in bytes) of an XIS image file with header information.
- HIS_RETURN Acquisition_IsAcquiringData (HACQDESC hAcqDesc)

  This function tests if the XISL is currently acquiring data.
- HIS_RETURN Acquisition_LoadFile (XislFileHandle fileHandle, unsigned char ∗∗buffer)

  Acquisition_LoadFile Loads a file from several locations.
- HIS_RETURN Acquisition_LoadXISFileToMemory (const char ∗filename, void ∗pMemoryFileBuffer, size_t bufferSize)

  This function loads an XIS Image file into the given data buffer.
- HIS_RETURN Acquisition_SaveFile (const char ∗filename, void ∗pImageBuffer, UINT dwRows, UINT dwColumns, UINT dwFrames, BOOL uiOnboardFileHeader, XIS_FileType filetype)

  This function saves the given data buffer to file with an appropriately generated HIS-type header.
- HIS_RETURN Acquisition_SaveRawData (const char ∗filename, const unsigned char ∗buffer, size_t bufferSize)

  This function saves the given buffer to file. This will either save a raw data file or an XIS image file if Acquisition_CreateXISFileInMemory is used.
- HIS_RETURN Acquisition_SetConsoleLogging (BOOL enableConsole)

  This function will be used to enable or disable logging to the console window.
- HIS_RETURN Acquisition_SetDACOffsetFloorValueByMode (HACQDESC hAcqDesc, unsigned int uiMode)

  This function changes the detector DAC Offset setting for the selected mode.
- HIS_RETURN Acquisition_SetDACOffsetFloorValueInFlash (HACQDESC hAcqDesc, unsigned int uiMode, WORD wValue)

  This function writes the desired offset value to the detector memory for the selected mode. It must be set using Acquisition_SetDACoffset or Acquisition_SetDACOffsetFloorValueByMode.
- HIS_RETURN Acquisition_SetDACOffsetFloorValueInFlashInternal (HACQDESC hAcqDesc, unsigned int uiMode, WORD wValue)

  This function writes the desired offset value to the detector memory for the selected mode. It must be set using Acquisition_SetDACoffset or Acquisition_SetDACOffsetFloorValueByMode.
- HIS_RETURN Acquisition_SetFileLogging (const char ∗filename, BOOL enableLogging)

  This function will create a log file based on the file name provided and will enable console logging if desired.
- HIS_RETURN Acquisition_SetFPGACameraMode (HACQDESC hAcqDesc, FPGAType FPGACommand, BOOL bInverse)

This function sends an FPGA command to the detector. This functions is for internal use only.

- HIS_RETURN Acquisition_SetRotationAngle (HACQDESC hAcqDesc, long lRotAngle)

  This function activates the onboard image rotation angle on optical frame-grabbers for images up to 2048x2048 pixels.

- HIS_RETURN Acquisition_wpe_GetVersionNEW (int ∗major, int ∗minor, int ∗release, int ∗build)

  Retrieve version information about the used wpe library.

3.9.1 Function Documentation

3.9.1.1 HIS_RETURN **Acq_wpe_GetSystemInformation** (const char ∗ *ipAddress,* char ∗ *buffer,* int *bufferLen*)

This function is used to retrieve system information from the detector.

Parameters

ipAddress	IP-Address of the device to control
buffer	buffer to hold the return text
bufferLen	length of the buffer

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.9.1.2 HIS_RETURN **Acq_WPE_Init** ( )

Acq_WPE_Init called when dll is loaded, intialize wpe, global objects, etc...

Returns

HIS_RETURN

3.9.1.3 HIS_RETURN **Acquisition_ActivateServiceMode** (**HACQDESC** *hAcqDesc,* BOOL *bActivate*)

This function activates the Service Mode, which means Service Data is written into acquired images. This makes it easy to provide images for support.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
bActivate	Show Service Data: bActivate = TRUE. otherwise: bActivate = FALSE.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.9.1.4 HIS_RETURN **Acquisition_CreateXISFileInMemory** (void ∗ *pMemoryFileBuffer*, void ∗ *pDataBuffer*, UINT *dwRows*, UINT *dwColumns*, UINT *dwFrames*, BOOL *uiOnboardFileHeader*, XIS_FileType *filetype*)

This function creates an XIS image file (including header) in the given memory buffer.

Parameters

pMemory-FileBuffer	Output buffer that will contain the XIS image file (with header)
pDataBuffer	Input image data buffer (without header information)
dwRows	Number of rows in the image
dwColumns	Number of columns in the image
dwFrames	Number of frames in the image
uiOnboard-FileHeader	A boolean value indicating whether the file has the on-board style header (TRUE) or not (FALSE)
filetype	The data-type of the pixels in the image.

Returns

HIS_ALL_OK

3.9.1.5 HIS_RETURN **Acquisition_DeleteFile** (XislFileHandle *fileHandle*)

Acquisition_DeleteFile Deletes a file from a location.

Parameters

`in`	*fileHandle*	A valid file handle.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

3.9.1.6 HIS_RETURN **Acquisition_DoOffsetCorrection32** (unsigned long ∗ *pSource*, unsigned long ∗ *pDest*, unsigned long ∗ *pOffsetData*, int *nCount*)

Similar to Acquisition_DoOffsetCorrection.

3.9.1.7 HIS_RETURN **Acquisition_DoOffsetGainCorrection_Ex32** (unsigned long ∗ *pSource*, unsigned long ∗ *pDest*, unsigned long ∗ *pOffsetData*, unsigned long ∗ *pGainData*, unsigned long ∗ *pGainAVG*, int *nCount*, int *nFrame*)

Similar to Acquisition_DoOffsetGainCorrection_Ex.

3.9.1.8 HIS_RETURN **Acquisition_GetConnectionStatus** (**HACQDESC** *hAcqDesc*)

retrievs the status of the current connection (if supported).

Parameters

in	*hAcqDesc*	Handle of a valid Acquisition Descriptor.

Returns

Returns HIS_ALL_OK on success or an appropriate error code including HIS_ERROR_NO_FPGA_ACK otherwise.

Note

Currenlty only valid for network connected detectors

3.9.1.9 HIS_RETURN **Acquisition_GetDACOffsetFloorValueFromFlash** (**HACQDESC** *hAcqDesc*, unsigned int *uiMode*, WORD ∗ *pwValue*)

This function reads the desired offset value from the detector memory for the selected mode. It must be set using Acquisition_SetDACoffset or Acquisition_SetDACOffsetFloorValueByMode.

Parameters

hAcqDesc	Acquisition descriptor structure.
uiMode	Mode to read (0-63).
pwValue	pwValue Pointer to a value to retrieve the offset value

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

Note

currently only available for 4343RF detectors

3.9.1.10 HIS_RETURN Acquisition_GetDetectorProperties (HACQDESC *hAcqDesc*, GBIF_Detector_Properties ∗ *pDetectorProperties*)

This function retrieved the GBIF_Detector_Properties structure, which contains permanently stored information of the connected device.

Parameters

hAcqDesc	Acquisition descriptor structure.
pDetector-Properties	Pointer to instance of GBIF_Detector_Properties

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.9.1.11 HIS_RETURN Acquisition_GetFileInfo (XislFileHandle *fileHandle*, XislFileInfo ∗ *fileInfo*)

Acquisition_GetFileInfo Retrieves basic information about a file.

Parameters

`in`	*fileHandle*	A valid file handle.
`out`	*fileInfo*	If the function succeeds, this parameter retrieves the handle of the missed image file.

Returns

Returns always HIS_ALL_OK. This function cannot fail.

3.9.1.12 HIS_RETURN Acquisition_GetRotationAngle (HACQDESC *hAcqDesc*, long ∗ *lRotAngle*)

Get onboard Rotation Setting for FG-E Opto.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
lRotAngle	Retrieves rotation angle.

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.9.1.13 HIS_RETURN Acquisition_GetTriggerOutStatus (HACQDESC *hAcqDesc,* int ∗ *iTriggerStatus*)

This function retrieves the triggerout status from detector.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
iTrigger-Status	see table

Note

- -1 Error, progably timeout

- 0 triger in was low
- 1 triger in was high only available for XRpad detectors

Returns

HIS_ALL_OK trigger status could be retrieved otherwise an error code

3.9.1.14 HIS_RETURN Acquisition_GetVersion (int ∗ *major,* int ∗ *minor,* int ∗ *release,* int ∗ *build*)

reads the version of the xisl dll

Parameters

major	Pointer to a variable to retrieve the major version number.
minor	Pointer to a variable to retrieve the minor version number.
release	Pointer to a variable to retrieve the release number.
build	Pointer to a variable to retrieve the build number.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.9.1.15 HIS_RETURN Acquisition_GetXISFileBufferSize (size_t ∗ *pFileSize,* UINT *dwRows,* UINT *dwColumns,* UINT *dwFrames,* BOOL *uiOnboardFileHeader,* XIS_FileType *filetype*)

This function returns the requiered size(in bytes) of an XIS image file with header information.

Parameters

pFileSize	Requested file size in bytes
dwRows	Number of rows in the image
dwColumns	Number of columns in the image
dwFrames	Number of frames in the image
uiOnboard-FileHeader	A boolean value indicating whether the file has the on-board style header (TRUE) or not (FALSE)
filetype	The data-type of the pixels in the image.

Returns

HIS_ALL_OK

3.9.1.16 HIS_RETURN Acquisition_IsAcquiringData (HACQDESC *hAcqDesc*)

This function tests if the XISL is currently acquiring data.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor

Returns

If the function is successful it returns 1 if an aquisition is running - 0 if no acquisition is running otherwise an error code.

3.9.1.17 HIS_RETURN Acquisition_LoadFile (XislFileHandle *fileHandle,* unsigned char ∗∗ *buffer*)

Acquisition_LoadFile Loads a file from several locations.

Parameters

`in`	*fileHandle*	A valid file handle.
`out`	*buffer*	Pointer to buffer to retrieve the location of the data.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

This function is blocking. Depending on size and location of a file, the calling thread might block for an unpredictable period.

3.9.1.18 HIS_RETURN Acquisition_LoadXISFileToMemory (const char ∗ *filename*, void ∗ *pMemoryFileBuffer*, size_t *bufferSize*)

This function loads an XIS Image file into the given data buffer.

Parameters

filename	Path to the input file
pMemory-FileBuffer	Output image data buffer. This should be approriately allocated to fit all the image and header data.
bufferSize	The size of the output buffer (in bytes) this should be equal to the size of the image and header.

Returns

HIS_ALL_OK

3.9.1.19 HIS_RETURN Acquisition_SaveFile (const char ∗ *filename*, void ∗ *pImageBuffer*, UINT *dwRows*, UINT *dwColumns*, UINT *dwFrames*, BOOL *uiOnboardFileHeader*, XIS_FileType *filetype*)

This function saves the given data buffer to file with an appropriately generated HIS-type header.

Parameters

filename	Path to the output file
pImage-Buffer	Input image data buffer (without header information)
dwRows	Number of rows in the image
dwColumns	Number of columns in the image
dwFrames	Number of frames in the image
uiOnboard-FileHeader	A boolean value indicating whether the file is to have the on-board style header (TRUE) or not (FALSE)
filetype	The data-type of the pixels in the image.

Returns

HIS_ALL_OK

3.9.1.20 HIS_RETURN Acquisition_SaveRawData (const char ∗ *filename*, const unsigned char ∗ *buffer*, size_t *bufferSize*)

This function saves the given buffer to file. This will either save a raw data file or an XIS image file if Acquisition_CreateXISFileInMemory is used.

Parameters

filename	Path to the output file
buffer	Input image data buffer (with or without header information)
bufferSize	The size of the data buffer (including any headers used) in bytes

Returns

HIS_ALL_OK

3.9.1.21 HIS_RETURN Acquisition_SetConsoleLogging (BOOL *enableConsole*)

This function will be used to enable or disable logging to the console window.

Parameters

enable-Console	Parameter used to enable or disable logging output to the console

Returns

If the function is successful it returns zero, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.9.1.22 HIS_RETURN Acquisition_SetDACOffsetFloorValueByMode (HACQDESC *hAcqDesc,* unsigned int *uiMode*)

This function changes the detector DAC Offset setting for the selected mode.

Parameters

hAcqDesc	Acquisition descriptor structure.
uiMode	Mode that should be changed (0-63).

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

Note

currently only available for 4343RF detectors; the value for the selected mode is retrieved from detector flash

3.9.1.23 HIS_RETURN Acquisition_SetDACOffsetFloorValueInFlash (HACQDESC *hAcqDesc*, unsigned int *uiMode*, WORD *wValue*)

This function writes the desired offset value to the detector memory for the selected mode. It must be set using Acquisition_SetDACoffset or Acquisition_SetDACOffset-FloorValueByMode.

Parameters

hAcqDesc	Acquisition descriptor structure.
uiMode	Mode that should be changed (0-63).
wValue	(0-4095) wValue value to set for the mode

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

Note

currently only available for 4343RF detectors; only modes $>=$ 32 are changeable

3.9.1.24 HIS_RETURN Acquisition_SetDACOffsetFloorValueInFlashInternal (HACQDESC *hAcqDesc*, unsigned int *uiMode*, WORD *wValue*)

This function writes the desired offset value to the detector memory for the selected mode. It must be set using Acquisition_SetDACoffset or Acquisition_SetDACOffset-FloorValueByMode.

Parameters

hAcqDesc	Acquisition descriptor structure.
uiMode	Mode that should be changed (0-63).
wValue	wValue value to set for the mode

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

Note

currently only available for 4343RF detectors

3.9.1.25 HIS_RETURN Acquisition_SetFileLogging (const char $*$ *filename*, BOOL *enableLogging*)

This function will create a log file based on the file name provided and will enable console logging if desired.

Parameters

filename	Customizable file name for logging output file. If the filepath parameter is NULL a default logging file will be created.
enable-Logging	Parameter used to enable or disable logging output to the console

Returns

If the function is successful it returns zero, otherwise an error code.

3.9.1.26 HIS_RETURN Acquisition_SetFPGACameraMode (HACQDESC *hAcqDesc,* FPGAType *FPGACommand,* BOOL *bInverse*)

This function sends an FPGA command to the detector. This functions is for internal use only.

Parameters

hAcqDesc	Handle of a valid Acquisition Descriptor
FPGA-Command	command to send to the detector
bInverse	do a bitwise invers on the data field

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code.

3.9.1.27 HIS_RETURN Acquisition_SetRotationAngle (HACQDESC *hAcqDesc,* long *lRotAngle*)

This function activates the onboard image rotation angle on optical frame-grabbers for images up to 2048x2048 pixels.

Parameters

hAcqDesc	Pointer to acquisition descriptor structure
lRotAngle	Rotation angle: must be 0, 90 or -90

Returns

If the function is successful it returns HIS_ALL_OK, otherwise an error code. To get extended information call Acquisition_GetErrorCode.

3.9.1.28 HIS_RETURN Acquisition_wpe_GetVersionNEW (int ∗ *major,* int ∗ *minor,* int ∗ *release,* int ∗ *build*)

Retrieve version information about the used wpe library.

Parameters

major	Pointer to a variable to retrieve the major version number.
minor	Pointer to a variable to retrieve the minor version number.
release	Pointer to a variable to retrieve the release number.
build	Pointer to a variable to retrieve the build number.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure.

3.10 enumerations provided by Acq.h

Classes

- struct XRpad_BatteryStatus
- struct XRpad_ShockEvent
- struct XRpad_ShockSensorReport
- struct XRpad_TempSensor
- struct XRpad_TempSensorReport
- struct XRpad_VersionInfo

Defines

- #define HIS_ALL_OK 0
- #define HIS_ERROR_ABORT 46
- #define HIS_ERROR_ABORTCURRFRAME 21
- #define HIS_ERROR_ACKNOWLEDGE_IMAGE 133
- #define HIS_ERROR_ACQ 43
- #define HIS_ERROR_ACQ_ALREADY_RUNNING 5
- #define HIS_ERROR_ACQABORT 12
- #define HIS_ERROR_ACQUISITION 13
- #define HIS_ERROR_ALREADY_EXISTS 74
- #define HIS_ERROR_AVERAGED_LOST 49
- #define HIS_ERROR_BAD_SORTING_PARAM 50
- #define HIS_ERROR_BOARDINIT 2
- #define HIS_ERROR_BUFFERSPACE_NOT_SUFF 29
- #define HIS_ERROR_CORRBUFFER_INCOMPATIBLE 4
- #define HIS_ERROR_CREATE_MEMORYMAPPING 35
- #define HIS_ERROR_CREATE_MUTEX 42
- #define HIS_ERROR_CURL 65
- #define HIS_ERROR_DESC_NOT_LOCAL 44
- #define HIS_ERROR_DOES_NOT_EXIST 75
- #define HIS_ERROR_EMI_NOT_SET 115
- #define HIS_ERROR_ENABLE_INTERRUPTS 108
- #define HIS_ERROR_ENABLE_ONBOARD_GAINOFFSET 68
- #define HIS_ERROR_ENABLE_ONBOARD_MEAN 67
- #define HIS_ERROR_ENABLE_ONBOARD_OFFSET 66
- #define HIS_ERROR_ENABLE_ONBOARD_PREVIEW 69
- #define HIS_ERROR_FRAME_INV 31
- #define HIS_ERROR_FUNC_NOTIMPL 40
- #define HIS_ERROR_GET_CHARGE_MODE 139
- #define HIS_ERROR_GET_NUM_BOARDS 33
- #define HIS_ERROR_GET_ONBOARD_OFFSET 64
- #define HIS_ERROR_GETHWHEADERINFO 22
- #define HIS_ERROR_GETOSVERSION 16
- #define HIS_ERROR_HEADER_TIMEOUT 56

- #define HIS_ERROR_HW_ALREADY_OPEN_BY_ANOTHER_PROCESS 34
- #define HIS_ERROR_HW_BOARD_CHANNEL_ALREADY_USED 146
- #define HIS_ERROR_HWHEADER_INV 23
- #define HIS_ERROR_ILLEGAL_INDEX 60
- #define HIS_ERROR_INVALID_FILENAME 77
- #define HIS_ERROR_INVALID_FUNC_CALL 20
- #define HIS_ERROR_INVALID_HANDLE 73
- #define HIS_ERROR_INVALID_PARAM 45
- #define HIS_ERROR_INVALIDACQDESC 7
- #define HIS_ERROR_INVALIDBUFFERNR 72
- #define HIS_ERROR_LOAD_COORECTIONIMAGETOBUFFER 71
- #define HIS_ERROR_LOADDRIVER 39
- #define HIS_ERROR_MEMORY 1
- #define HIS_ERROR_MEMORY_MAPPING 41
- #define HIS_ERROR_MISSING_VERSION_INFORMATION 143
- #define HIS_ERROR_NO_BOARD_IN_SUBNET 52
- #define HIS_ERROR_NO_FPGA_ACK 57
- #define HIS_ERROR_NOCAMERA 3
- #define HIS_ERROR_NODESC_AVAILABLE 28
- #define HIS_ERROR_NOT_DISCOVERED 62
- #define HIS_ERROR_NOT_INITIALIZED 61
- #define HIS_ERROR_NR_OF_BOARDS_CHANGED 58
- #define HIS_ERROR_ONBOARDAVGFAILED 63
- #define HIS_ERROR_OPEN_FILE 76
- #define HIS_ERROR_READ_DATA 26
- #define HIS_ERROR_RETRIEVE_ENHANCED_HEADER 107
- #define HIS_ERROR_SET_CHARGE_MODE 118
- #define HIS_ERROR_SET_IDLE_TIMEOUT 117
- #define HIS_ERROR_SET_IMAGE_TAG 104
- #define HIS_ERROR_SET_IMAGE_TAG_LENGTH 106
- #define HIS_ERROR_SET_ONBOARD_BINNING 70
- #define HIS_ERROR_SET_PROC_SCRIPT 105
- #define HIS_ERROR_SETBAUDRATE 27
- #define HIS_ERROR_SETCAMERAMODE 30
- #define HIS_ERROR_SETDISCOVERYTIMEOUT 78
- #define HIS_ERROR_SETEXAMFLAG 59
- #define HIS_ERROR_SETFRMSYNC 17
- #define HIS_ERROR_SETFRMSYNCMODE 18
- #define HIS_ERROR_SETLINETRIG_MODE 24
- #define HIS_ERROR_SETTIMERSYNC 19
- #define HIS_ERROR_SLOW_SYSTEM 32
- #define HIS_ERROR_TIMEOUT 6
- #define HIS_ERROR_UNABLE_TO_ACCESS_DETECTOR_FLASH 55
- #define HIS_ERROR_UNABLE_TO_CLOSE_BOARD 54
- #define HIS_ERROR_UNABLE_TO_OPEN_BOARD 53
- #define HIS_ERROR_UNKNOWN_IP_MAC_NAME 51

- #define HIS_ERROR_VXD_REGISTER_DMA_ADDRESS 36
- #define HIS_ERROR_VXD_REGISTER_IRQ 14
- #define HIS_ERROR_VXD_REGISTER_STAT_ADDR 37
- #define HIS_ERROR_VXD_REGISTER_STATADR 15
- #define HIS_ERROR_VXD_UNMASK_IRQ 38
- #define HIS_ERROR_VXDGETDMAADR 11
- #define HIS_ERROR_VXDNOTFOUND 8
- #define HIS_ERROR_VXDNOTOPEN 9
- #define HIS_ERROR_VXDUNKNOWNERROR 10
- #define HIS_ERROR_WRITE_DATA 25
- #define HIS_ERROR_WRONG_CAMERA_MODE 48
- #define HIS_ERROR_WRONGBOARDSELECT 47
- #define HIS_ERROR_XRPD_CONNECT 134
- #define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT 112
- #define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_CRIT 119
- #define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_WARN 120
- #define HIS_ERROR_XRPD_FACTORY_RESET_SHOCK_EVENT 121
- #define HIS_ERROR_XRPD_GET_AUTOPOWERONLOCATIONS 137
- #define HIS_ERROR_XRPD_GET_CURRENT_VOLTAGE 147
- #define HIS_ERROR_XRPD_GET_SDCARD_INFO 113
- #define HIS_ERROR_XRPD_GET_SDCARD_TIMEOUT 142
- #define HIS_ERROR_XRPD_GET_TEMPERATURE_THRESHOLDS 129
- #define HIS_ERROR_XRPD_NO_EVENT_INTERFACE 111
- #define HIS_ERROR_XRPD_NO_EVENTCALLBACK_DEFINED 130
- #define HIS_ERROR_XRPD_NO_LOCATION 116
- #define HIS_ERROR_XRPD_NO_NETWORK 122
- #define HIS_ERROR_XRPD_NOT_CONNECTED 144
- #define HIS_ERROR_XRPD_REQUEST_POWERSTATE 136
- #define HIS_ERROR_XRPD_RESEND_ALL_MSG 132
- #define HIS_ERROR_XRPD_RESET_SHOCK 135
- #define HIS_ERROR_XRPD_RESET_TEMPERATURE_TIMEOUT 127
- #define HIS_ERROR_XRPD_SDCARDPERFORMANCE 145
- #define HIS_ERROR_XRPD_SESSION_ERROR 109
- #define HIS_ERROR_XRPD_SET_AUTOPOWERONLOCATIONS 138
- #define HIS_ERROR_XRPD_SET_CPUFREQ_GOVERNOR 148
- #define HIS_ERROR_XRPD_SET_DATE_TIME 131
- #define HIS_ERROR_XRPD_SET_EVENT 110
- #define HIS_ERROR_XRPD_SET_FORCE_FSCK 140
- #define HIS_ERROR_XRPD_SET_NETWORK 123
- #define HIS_ERROR_XRPD_SET_PRIVATE_KEY 125
- #define HIS_ERROR_XRPD_SET_SDCARD_TIMEOUT 141
- #define HIS_ERROR_XRPD_SET_TEMP_FAKE_MODE 114
- #define HIS_ERROR_XRPD_SET_TEMPERATURE_THRESHOLDS 128
- #define HIS_ERROR_XRPD_SET_TEMPERATURE_TIMEOUT 126
- #define HIS_ERROR_XRPD_VERIFY_GENUINENESS 124

Typedefs

- typedef UINT ACQDESCPOS
- typedef HANDLE HACQDESC
- typedef enum OnboardBinningMode OnboardBinningMode
- typedef enum ProcScriptOperation ProcScriptOperation
- typedef enum XIS_Acquisition_Event XIS_Acquisition_Event
- typedef enum XIS_Battery_Event XIS_Battery_Event
- typedef enum XIS_Detector_Event XIS_Detector_Event
- typedef enum XIS_Detector_TRIGOUT_SignalMode XIS_Detector_TRIGOUT_-SignalMode
- typedef enum XIS_DetectorTriggerMode XIS_DetectorTriggerMode
- typedef enum XIS_Event XIS_Event
- typedef enum XIS_FileType XIS_FileType
- typedef enum XIS_Library_Event XIS_Library_Event
- typedef enum XIS_Sensor_Event XIS_Sensor_Event
- typedef enum XislFileEntryType XislFileEntryType
- typedef void ∗ XislFileHandle
- typedef enum XislFileStorageLocation XislFileStorageLocation
- typedef void ∗ XislFtpSession
- typedef enum XRpad_BatteryHealth XRpad_BatteryHealth
- typedef enum XRpad_BatteryPresence XRpad_BatteryPresence
- typedef struct XRpad_BatteryStatus XRpad_BatteryStatus
- typedef enum XRpad_ChargeMode XRpad_ChargeMode
- typedef enum XRpad_DataInterfaceControlEnum XRpad_DataInterfaceControl-Enum
- typedef struct XRpad_ShockEvent XRpad_ShockEvent
- typedef struct XRpad_ShockSensorReport XRpad_ShockSensorReport
- typedef struct XRpad_TempSensor XRpad_TempSensor
- typedef struct XRpad_TempSensorReport XRpad_TempSensorReport
- typedef struct XRpad_VersionInfo XRpad_VersionInfo

Enumerations

- enum OnboardBinningMode { ONBOARDBINNING2x1 = 0, ONBOARDBINNIN-G2x2 = 1, ONBOARDBINNING4x1 = 2, ONBOARDBINNING4x4 = 3, ONBOA-RDBINNING3x3 = 4, ONBOARDBINNING9to4 = 5 }
- enum ProcScriptOperation { PREBINNING, PREMEAN, PRESTOREBUFFER, OFFSET, GAIN, MEAN, PREVIEW, BINNING, STOREBUFFER, STORESD, SEND }
- enum XIS_Acquisition_Event { XAE_TRIGOUT = 0x00000002, XAE_READOUT = 0x00000004 }
- enum XIS_Battery_Event { XBE_BATTERY_REPORT = 0x00000001, XBE_BA-TTERY_WARNING = 0x00000002 }
- enum XIS_Detector_Event { XDE_BUFFERS_IN_USE = 0x00000001, XDE_ST-ORED_IMAGE = 0x00000002, XDE_DROPPED_IMAGE = 0x00000003 }

- enum XIS_Detector_TRIGOUT_SignalMode { TRIGOUT_SIGNAL_FRM_EN_P-WM, TRIGOUT_SIGNAL_FRM_EN_PWM_INV, TRIGOUT_SIGNAL_EP, TRIG-OUT_SIGNAL_EP_INV, TRIGOUT_SIGNAL_DDD_Pulse, TRIGOUT_SIGNAL-_DDD_Pulse_INV, TRIGOUT_SIGNAL_GND, TRIGOUT_SIGNAL_VCC }
- enum XIS_DetectorTriggerMode { TRIGGERMODE_DDD, TRIGGERMODE_D-DD_WO_CLEARANCE, TRIGGERMODE_STARTSTOP, TRIGGERMODE_FR-AMEWISE, TRIGGERMODE_AED, TRIGGERMODE_ROWTAG, TRIGGERM-ODE_DDD_POST_OFFSET, TRIGGERMODE_DDD_DUAL_POST_OFFSET }
- enum XIS_Event { XE_ACQUISITION_EVENT = 0x00000001, XE_SENSOR_-EVENT = 0x00000002, XE_SDCARD_EVENT = 0x00000004, XE_BATTERY_-EVENT = 0x00000005, XE_LOCATION_EVENT = 0x00000006, XE_NETWOR-K_EVENT = 0x00000007, XE_DETECTOR_EVENT = 0x00000008, XE_LIBRA-RY_EVENT = 0x00000009, XE_SDCARD_FSCK_EVENT = 0x0000000A }
- enum XIS_FileType { PKI_RESERVED = 1, PKI_DOUBLE = 2, PKI_SHORT = 4, PKI_SIGNED = 8, PKI_ERRORMAPONBOARD = 16, PKI_LONG = 32, PKI_SI-GNEDSHORT = PKI_SHORT | PKI_SIGNED, PKI_SIGNEDLONG = PKI_LONG | PKI_SIGNED, PKI_FAULTMASK = PKI_LONG | PKI_RESERVED }
- enum XIS_Library_Event { XLE_HIS_ERROR_PACKET_LOSS = 0x00000001 }
- enum XIS_Sensor_Event { XSE_HALL = 0x00000001, XSE_SHOCK = 0x00000010, XSE_TEMPERATURE = 0x00000020, XSE_TEMPERATUR-E_BACK_TO_NORMAL = 0x00000021, XSE_THERMAL_SHUTDOWN = 0x00000022 }
- enum XislFileEntryType { XFT_File = 1, XFT_Directory = 2, XFT_Link = 4, XFT-_Other = 0x80000000, XFT_Any = 0xFFFFFFFF }
- enum XislFileStorageLocation { XFSL_Local = 0, XFSL_FTP = 1 }
- enum XislLoggingLevels { LEVEL_TRACE = 0, LEVEL_DEBUG, LEVEL_INFO, LEVEL_WARN, LEVEL_ERROR, LEVEL_FATAL, LEVEL_ALL, LEVEL_NONE }
- enum XRpad_BatteryHealth { XRpad_BATTERY_OK = 0x0000, XRpad_COM-MUNICATION_ERROR = 0x0001, XRpad_TERMINATE_DISCHARGE_ALARM = 0x0002, XRpad_UNDERVOLTAGE_ALARM = 0x0004, XRpad_OVERVOLT-AGE_ALARM = 0x0008, XRpad_OVERTEMPERATURE_ALARM = 0x0010, X-Rpad_BATTERY_UNKNOWN_ERROR = 0x0080 }
- enum XRpad_BatteryPresence { XRpad_NO_BATTERY = 0, XRpad_BATTER-Y_INSERTED = 1, XRpad_DUMMY_INSERTED = 2 }
- enum XRpad_ChargeMode { XRpad_NOT_CHARGING = 0, XRpad_CHARGI-NG_SLOW = 1, XRpad_CHARGING_NORMAL = 2, XRpad_CHARGING_FAST = 3, XRpad_FULLY_CHARGED = 4, XRpad_DISCHARGING = 5 }
- enum XRpad_DataInterfaceControlEnum { XRpad_DATA_VIA_LAN = 0, XRpad-_DATA_VIA_WLAN = 1 }
- enum XRpad_SystemControlEnum { XRpad_SYSTEM_CONTROL_REBOOT = 0, XRpad_SYSTEM_CONTROL_RESTART_NETWORK = 1, XRpad_SYSTEM-_CONTROL_SHUTDOWN = 2, XRpad_SYSTEM_CONTROL_SET_DEEP_SL-EEP = 3, XRpad_SYSTEM_CONTROL_SET_IDLE = 4 }

3.10.1 Define Documentation

3.10.1.1 #define HIS_ALL_OK 0

No error

3.10.1.2 #define HIS_ERROR_ABORT 46

Error during abort acquisition function.

3.10.1.3 #define HIS_ERROR_ABORTCURRFRAME 21

Aborting current frame failed.

3.10.1.4 #define HIS_ERROR_ACKNOWLEDGE_IMAGE 133

Error acknowledging the image.

3.10.1.5 #define HIS_ERROR_ACQ 43

Error starting the acquisition.

3.10.1.6 #define HIS_ERROR_ACQ_ALREADY_RUNNING 5

Acquisition is already running.

3.10.1.7 #define HIS_ERROR_ACQABORT 12

An unexpected acquisition abort occurred.

3.10.1.8 #define HIS_ERROR_ACQUISITION 13

error occurred during data acquisition.

3.10.1.9 #define HIS_ERROR_ALREADY_EXISTS 74

Error Invalid filename, file already exists.

3.10.1.10 #define HIS_ERROR_AVERAGED_LOST 49

The number of images for frame grabber onboard averaging must be 2 to the power of n.

3.10.1.11 #define HIS_ERROR_BAD_SORTING_PARAM 50

Parameter for (onboard) sorting not valid.

3.10.1.12 #define HIS_ERROR_BOARDINIT 2

Unable to initialize board.

3.10.1.13 #define HIS_ERROR_BUFFERSPACE_NOT_SUFF 29

Buffer space not sufficient.

3.10.1.14 #define HIS_ERROR_CORRBUFFER_INCOMPATIBLE 4

Your correction files do not have a proper size.

3.10.1.15 #define HIS_ERROR_CREATE_MEMORYMAPPING 35

Error creating memory mapped file.

3.10.1.16 #define HIS_ERROR_CREATE_MUTEX 42

Could not create Mutex.

3.10.1.17 #define HIS_ERROR_CURL 65

Error CURL.

3.10.1.18 #define HIS_ERROR_DESC_NOT_LOCAL 44

Acquisition descriptor is not local.

3.10.1.19 #define HIS_ERROR_DOES_NOT_EXIST 75

Error Invalid filename type does not exist.

3.10.1.20 #define HIS_ERROR_EMI_NOT_SET 115

Error the requested EMI readout mode was not reported by the detector.

3.10.1.21 #define HIS_ERROR_ENABLE_INTERRUPTS 108

Error enabling XRPD interrupts.

3.10.1.22 #define HIS_ERROR_ENABLE_ONBOARD_GAINOFFSET 68

Error setting onboard gain corr mode.

3.10.1.23 #define HIS_ERROR_ENABLE_ONBOARD_MEAN 67

Error setting onboard mean corr mode.

3.10.1.24 #define HIS_ERROR_ENABLE_ONBOARD_OFFSET 66

Error setting onboard offset corr mode.

3.10.1.25 #define HIS_ERROR_ENABLE_ONBOARD_PREVIEW 69

Error setting onboard preview mode.

3.10.1.26 #define HIS_ERROR_FRAME_INV 31

Frame invalid.

3.10.1.27 #define HIS_ERROR_FUNC_NOTIMPL 40

Function is not implemented.

3.10.1.28 #define HIS_ERROR_GET_CHARGE_MODE 139

Error retrieving the requested charge mode from the detector.

3.10.1.29 #define HIS_ERROR_GET_NUM_BOARDS 33

Error during getting number of boards.

3.10.1.30 #define HIS_ERROR_GET_ONBOARD_OFFSET 64

Error getting onboard offset.

3.10.1.31 #define HIS_ERROR_GETHWHEADERINFO 22

Getting hardware header failed.

3.10.1.32 #define HIS_ERROR_GETOSVERSION 16

Getting version of operating system failed.

3.10.1.33 #define HIS_ERROR_HEADER_TIMEOUT 56

No frame header received from Detector.

3.10.1.34 #define HIS_ERROR_HW_ALREADY_OPEN_BY_ANOTHER_PROCESS 34

Communication channel already opened by another process.

3.10.1.35 #define HIS_ERROR_HW_BOARD_CHANNEL_ALREADY_USED 146

Requested channel is already openend.

3.10.1.36 #define HIS_ERROR_HWHEADER_INV 23

Hardware header is invalid.

3.10.1.37 #define HIS_ERROR_ILLEGAL_INDEX 60

Error Function called with an illegal index number.

3.10.1.38 #define HIS_ERROR_INVALID_FILENAME 77

Error Invalid filename for image tag or log file.

3.10.1.39 #define HIS_ERROR_INVALID_FUNC_CALL 20

Invalid function call.

3.10.1.40 #define HIS_ERROR_INVALID_HANDLE 73

Error Invalid SHOCKID.

3.10.1.41 #define HIS_ERROR_INVALID_PARAM 45

Invalid Parameter.

3.10.1.42 #define HIS_ERROR_INVALIDACQDESC 7

Acquisition descriptor invalid.

3.10.1.43 #define HIS_ERROR_INVALIDBUFFERNR 72

Error Invalid pointer/buffer passed as parameter.

3.10.1.44 #define HIS_ERROR_LOAD_COORECTIONIMAGETOBUFFER 71

Error Loading image from SD to onboard buffer.

3.10.1.45 #define HIS_ERROR_LOADDRIVER 39

Unable to load driver.

3.10.1.46 #define HIS_ERROR_MEMORY 1

Memory couldn’t be allocated.

3.10.1.47 #define HIS_ERROR_MEMORY_MAPPING 41

Unable to create memory mapping.

3.10.1.48 #define HIS_ERROR_MISSING_VERSION_INFORMATION 143

Error not connected to on detector XRPD process.

3.10.1.49 #define HIS_ERROR_NO_BOARD_IN_SUBNET 52

Detector could not be found in the Subnet.

3.10.1.50 #define HIS_ERROR_NO_FPGA_ACK 57

Command not acknowledged.

3.10.1.51 #define HIS_ERROR_NOCAMERA 3

Got a time out. May be no detector present.

3.10.1.52 #define HIS_ERROR_NODESC_AVAILABLE 28

No acquisition descriptor available.

3.10.1.53 #define HIS_ERROR_NOT_DISCOVERED 62

Error No detectors discovered yet.

3.10.1.54 #define HIS_ERROR_NOT_INITIALIZED 61

Error Function or function environment not correctly initialised.

3.10.1.55 #define HIS_ERROR_NR_OF_BOARDS_CHANGED 58

Number of boards within network changed during broadcast.

3.10.1.56 #define HIS_ERROR_ONBOARDAVGFAILED 63

Error onbaord averaging failed.

3.10.1.57 #define HIS_ERROR_OPEN_FILE 76

Error Invalid filename for image tag or log file.

3.10.1.58 #define HIS_ERROR_READ_DATA 26

Reading data failed.

3.10.1.59 #define HIS_ERROR_RETRIEVE_ENHANCED_HEADER 107

Error retrieving the enhanced header.

3.10.1.60 #define HIS_ERROR_SET_CHARGE_MODE 118

Error setting the software requested charge mode.

3.10.1.61 #define HIS_ERROR_SET_IDLE_TIMEOUT 117

Error setting the on detector idle timeout.

3.10.1.62 #define HIS_ERROR_SET_IMAGE_TAG 104

Error setting the onboard image tag.

3.10.1.63 #define HIS_ERROR_SET_IMAGE_TAG_LENGTH 106

Error Image tag length exceeded 128char (including path: autosave/).

3.10.1.64 #define HIS_ERROR_SET_ONBOARD_BINNING 70

Error setting onboard binning mode.

3.10.1.65 #define HIS_ERROR_SET_PROC_SCRIPT 105

Error setting the onboard process script.

3.10.1.66 #define HIS_ERROR_SETBAUDRATE 27

Setting baud rate failed.

3.10.1.67 #define HIS_ERROR_SETCAMERAMODE 30

Setting detector mode failed.

3.10.1.68 #define HIS_ERROR_SETDISCOVERYTIMEOUT 78

Error setting gbif discovery timeout.

3.10.1.69 #define HIS_ERROR_SETEXAMFLAG 59

Unable to set the exam flag.

3.10.1.70 #define HIS_ERROR_SETFRMSYNC 17

Can not set frame sync.

3.10.1.71 #define HIS_ERROR_SETFRMSYNCMODE 18

Can not set frame sync mode.

3.10.1.72 #define HIS_ERROR_SETLINETRIG_MODE 24

Setting line trigger mode failed.

3.10.1.73 #define HIS_ERROR_SETTIMERSYNC 19

Can not set timer sync.

3.10.1.74 #define HIS_ERROR_SLOW_SYSTEM 32

System to slow.

3.10.1.75 #define HIS_ERROR_TIMEOUT 6

Got a time out from hardware.

3.10.1.76 #define HIS_ERROR_UNABLE_TO_ACCESS_DETECTOR_FLASH 55

Unable to access the flash memory of Detector.

3.10.1.77 #define HIS_ERROR_UNABLE_TO_CLOSE_BOARD 54

Unable to close connection to Network Detector.

3.10.1.78 #define HIS_ERROR_UNABLE_TO_OPEN_BOARD 53

Unable to open connection to Network Detector.

3.10.1.79 #define HIS_ERROR_UNKNOWN_IP_MAC_NAME 51

Connection to Network Detector cannot be opened due to invalid IP address / MAC / Detector name.

3.10.1.80 #define HIS_ERROR_VXD_REGISTER_DMA_ADDRESS 36

Error registering DMA address.

3.10.1.81 #define HIS_ERROR_VXD_REGISTER_IRQ 14

Unable to register interrupt.

3.10.1.82 #define HIS_ERROR_VXD_REGISTER_STAT_ADDR 37

Error registering static address.

3.10.1.83 #define HIS_ERROR_VXD_REGISTER_STATADR 15

Register status address failed.

3.10.1.84 #define HIS_ERROR_VXD_UNMASK_IRQ 38

Unable to unmask interrupt.

3.10.1.85 #define HIS_ERROR_VXDGETDMAADR 11

VxD Error: GetDmaAddr failed.

3.10.1.86 #define HIS_ERROR_VXDNOTFOUND 8

Unable to find VxD.

3.10.1.87 #define HIS_ERROR_VXDNOTOPEN 9

Unable to open VxD.

3.10.1.88 #define HIS_ERROR_VXDUNKNOWNERROR 10

Unknown error during VxD loading.

3.10.1.89 #define HIS_ERROR_WRITE_DATA 25

Writing data failed.

3.10.1.90 #define HIS_ERROR_WRONG_CAMERA_MODE 48

Change of Detector Mode during Acquisition.

3.10.1.91 #define HIS_ERROR_WRONGBOARDSELECT 47

The wrong board is selected.

3.10.1.92 #define HIS_ERROR_XRPD_CONNECT 134

Error connecting to the on detector XRPD process.

3.10.1.93 #define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT 112

Error creating fake shock events.

3.10.1.94 #define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_CRIT 119

Error creating critical level fake shock events.

3.10.1.95 #define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_WARN 120

Error creating warning level fake shock events.

3.10.1.96 #define HIS_ERROR_XRPD_FACTORY_RESET_SHOCK_EVENT 121

Error resetting the shock events to factory values.

3.10.1.97 #define HIS_ERROR_XRPD_GET_AUTOPOWERONLOCATIONS 137

Error retrieving the auto power on locations from the detector.

3.10.1.98 #define HIS_ERROR_XRPD_GET_CURRENT_VOLTAGE 147

Error retrieving the Voltage or Current from detector.

3.10.1.99 #define HIS_ERROR_XRPD_GET_SDCARD_INFO 113

Error retrieving the sd card info.

3.10.1.100 #define HIS_ERROR_XRPD_GET_SDCARD_TIMEOUT 142

Error getting the sd card timeout from the detector.

3.10.1.101 #define HIS_ERROR_XRPD_GET_TEMPERATURE_THRESHOLDS 129

Error getting the temperature thresholds from the detector.

3.10.1.102 #define HIS_ERROR_XRPD_NO_EVENT_INTERFACE 111

Error No interface to communicate event messages active.

3.10.1.103 #define HIS_ERROR_XRPD_NO_EVENTCALLBACK_DEFINED 130

Error no eventcallback defined for irq messages.

3.10.1.104 #define HIS_ERROR_XRPD_NO_LOCATION 116

Error retrieving the location info from the detector.

3.10.1.105 #define HIS_ERROR_XRPD_NO_NETWORK 122

Error getting the on detector LAN network speed.

3.10.1.106 #define HIS_ERROR_XRPD_NOT_CONNECTED 144

Error not connected to on detector XRPD process.

3.10.1.107 #define HIS_ERROR_XRPD_REQUEST_POWERSTATE 136

Error setting the power state on the detector.

3.10.1.108 #define HIS_ERROR_XRPD_RESEND_ALL_MSG 132

Error triggering the resend of all current messages by the XRPD.

3.10.1.109 #define HIS_ERROR_XRPD_RESET_SHOCK 135

Error resetting the shock event.

3.10.1.110 #define HIS_ERROR_XRPD_RESET_TEMPERATURE_TIMEOUT 127

Error resetting the temperature timeout counter.

3.10.1.111 #define HIS_ERROR_XRPD_SDCARDPERFORMANCE 145

Error retrieving the SD card performance.

3.10.1.112 #define HIS_ERROR_XRPD_SESSION_ERROR 109

Error XRPD session Error.

3.10.1.113 #define HIS_ERROR_XRPD_SET_AUTOPOWERONLOCATIONS 138

Error setting the auto power on locations on the detector.

3.10.1.114 #define HIS_ERROR_XRPD_SET_CPUFREQ_GOVERNOR 148

Unable to set on detector CPU govenor.

3.10.1.115 #define HIS_ERROR_XRPD_SET_DATE_TIME 131

Error setting on detectors date and time.

3.10.1.116 #define HIS_ERROR_XRPD_SET_EVENT 110

Error No interface to communicate event messages active.

3.10.1.117 #define HIS_ERROR_XRPD_SET_FORCE_FSCK 140

Error requesting an fscheck on next boot.

3.10.1.118 #define HIS_ERROR_XRPD_SET_NETWORK 123

Error setting the on detector LAN network speed.

3.10.1.119 #define HIS_ERROR_XRPD_SET_PRIVATE_KEY 125

Error setting the private key for genuiness.

3.10.1.120 #define HIS_ERROR_XRPD_SET_SDCARD_TIMEOUT 141

Error setting the sd card timeout on the detector.

3.10.1.121 #define HIS_ERROR_XRPD_SET_TEMP_FAKE_MODE 114

Error activating the fake temperatur mode on the detector.

3.10.1.122 #define HIS_ERROR_XRPD_SET_TEMPERATURE_THRESHOLDS 128

Error setting the temperature thresholds on the detector.

3.10.1.123 #define HIS_ERROR_XRPD_SET_TEMPERATURE_TIMEOUT 126

Error setting the temperature timeout.

3.10.1.124 #define HIS_ERROR_XRPD_VERIFY_GENUINENESS 124

Error verifying the private key for genuiness.

3.10.2 Typedef Documentation

3.10.2.1 typedef UINT ACQDESCPOS

3.10.2.2 typedef HANDLE HACQDESC

AcquisitionDesc defines a data structure that is used by all functions of XISL. It contains all required parameters for the acquisition. Access to the data fields is only possible via the XISL API functions. HACQDESC defines a HANLDE to the acquisition descriptor.

See also

AcquisitionDesc

3.10.2.3 typedef enum OnboardBinningMode OnboardBinningMode

3.10.2.4 typedef enum ProcScriptOperation ProcScriptOperation

3.10.2.5 typedef enum XIS_Acquisition_Event XIS_Acquisition_Event

3.10.2.6 typedef enum XIS_Battery_Event XIS_Battery_Event

3.10.2.7 typedef enum XIS_Detector_Event XIS_Detector_Event

3.10.2.8 typedef enum XIS_Detector_TRIGOUT_SignalMode XIS_Detector_TRIGOUT_SignalMode

3.10.2.9 typedef enum XIS_DetectorTriggerMode XIS_DetectorTriggerMode

3.10.2.10 typedef enum XIS_Event XIS_Event

3.10.2.11 typedef enum XIS_FileType XIS_FileType

3.10.2.12 typedef enum XIS_Library_Event XIS_Library_Event

3.10.2.13 typedef enum XIS_Sensor_Event XIS_Sensor_Event

3.10.2.14 typedef enum XislFileEntryType XislFileEntryType

3.10.2.15 typedef void∗ XislFileHandle

3.10.2.16 typedef enum XislFileStorageLocation XislFileStorageLocation

3.10.2.17 typedef void∗ XislFtpSession

3.10.2.18 typedef enum XRpad_BatteryHealth XRpad_BatteryHealth

3.10.2.19 typedef enum XRpad_BatteryPresence XRpad_BatteryPresence

3.10.2.20 typedef struct XRpad_BatteryStatus XRpad_BatteryStatus

3.10.2.21 typedef enum XRpad_ChargeMode XRpad_ChargeMode

3.10.2.22 typedef enum XRpad_DataInterfaceControlEnum XRpad_DataInterfaceControlEnum

Possible image transfer channels for XRpad

3.10.2.23 typedef struct XRpad_ShockEvent XRpad_ShockEvent

3.10.2.24 typedef struct XRpad_ShockSensorReport XRpad_ShockSensorReport

3.10.2.25 typedef struct XRpad_TempSensor XRpad_TempSensor

3.10.2.26 typedef struct XRpad_TempSensorReport XRpad_TempSensorReport

3.10.2.27 typedef struct XRpad_VersionInfo XRpad_VersionInfo

3.10.3 Enumeration Type Documentation

3.10.3.1 enum OnboardBinningMode

Enumerator:

ONBOARDBINNING2x1

ONBOARDBINNING2x2

ONBOARDBINNING4x1

ONBOARDBINNING4x4

ONBOARDBINNING3x3

ONBOARDBINNING9to4

3.10.3.2 enum ProcScriptOperation

Enumerator:

PREBINNING

PREMEAN

PRESTOREBUFFER

OFFSET

GAIN

MEAN

PREVIEW

BINNING

STOREBUFFER

STORESD

SEND

3.10.3.3 enum XIS_Acquisition_Event

Enumerator:

XAE_TRIGOUT

XAE_READOUT

3.10.3.4 enum XIS_Battery_Event

Enumerator:

XBE_BATTERY_REPORT

XBE_BATTERY_WARNING

3.10.3.5 enum XIS_Detector_Event

Enumerator:

XDE_BUFFERS_IN_USE

XDE_STORED_IMAGE

XDE_DROPPED_IMAGE

3.10.3.6 enum XIS_Detector_TRIGOUT_SignalMode

Enumerator:

TRIGOUT_SIGNAL_FRM_EN_PWM

TRIGOUT_SIGNAL_FRM_EN_PWM_INV

TRIGOUT_SIGNAL_EP

TRIGOUT_SIGNAL_EP_INV

TRIGOUT_SIGNAL_DDD_Pulse

TRIGOUT_SIGNAL_DDD_Pulse_INV

TRIGOUT_SIGNAL_GND

TRIGOUT_SIGNAL_VCC

3.10.3.7 enum XIS_DetectorTriggerMode

Enumerator:

TRIGGERMODE_DDD

TRIGGERMODE_DDD_WO_CLEARANCE

TRIGGERMODE_STARTSTOP

TRIGGERMODE_FRAMEWISE

TRIGGERMODE_AED

TRIGGERMODE_ROWTAG

TRIGGERMODE_DDD_POST_OFFSET

TRIGGERMODE_DDD_DUAL_POST_OFFSET

3.10.3.8 enum XIS_Event

Enumerator:

XE_ACQUISITION_EVENT

XE_SENSOR_EVENT

XE_SDCARD_EVENT

XE_BATTERY_EVENT

XE_LOCATION_EVENT

XE_NETWORK_EVENT

XE_DETECTOR_EVENT

XE_LIBRARY_EVENT

XE_SDCARD_FSCK_EVENT

3.10.3.9 enum XIS_FileType

Enumerator:

PKI_RESERVED

PKI_DOUBLE

PKI_SHORT

PKI_SIGNED

PKI_ERRORMAPONBOARD

PKI_LONG

PKI_SIGNEDSHORT

PKI_SIGNEDLONG

PKI_FAULTMASK

3.10.3.10 enum XIS_Library_Event

Enumerator:

XLE_HIS_ERROR_PACKET_LOSS Frame is lost. not all network packages could be received.

3.10.3.11 enum XIS_Sensor_Event

Enumerator:

XSE_HALL

XSE_SHOCK

XSE_TEMPERATURE

XSE_TEMPERATURE_BACK_TO_NORMAL

XSE_THERMAL_SHUTDOWN

3.10.3.12 enum XislFileEntryType

Enumerator:

XFT_File

XFT_Directory

XFT_Link

XFT_Other

XFT_Any

3.10.3.13 enum XislFileStorageLocation

Enumerator:

XFSL_Local

XFSL_FTP

3.10.3.14 enum XislLoggingLevels

Enumerator:

LEVEL_TRACE

LEVEL_DEBUG

LEVEL_INFO

LEVEL_WARN

LEVEL_ERROR

LEVEL_FATAL

LEVEL_ALL

LEVEL_NONE

3.10.3.15 enum XRpad_BatteryHealth

Enumerator:

XRpad_BATTERY_OK

XRpad_COMMUNICATION_ERROR

XRpad_TERMINATE_DISCHARGE_ALARM

XRpad_UNDERVOLTAGE_ALARM

XRpad_OVERVOLTAGE_ALARM

XRpad_OVERTEMPERATURE_ALARM

XRpad_BATTERY_UNKNOWN_ERROR

3.10.3.16 enum XRpad_BatteryPresence

Enumerator:

XRpad_NO_BATTERY

XRpad_BATTERY_INSERTED

XRpad_DUMMY_INSERTED

3.10.3.17 enum XRpad_ChargeMode

Enumerator:

- ***XRpad_NOT_CHARGING***
- ***XRpad_CHARGING_SLOW***
- ***XRpad_CHARGING_NORMAL***
- ***XRpad_CHARGING_FAST***
- ***XRpad_FULLY_CHARGED***
- ***XRpad_DISCHARGING***

3.10.3.18 enum XRpad_DataInterfaceControlEnum

Possible image transfer channels for XRpad

Enumerator:

- ***XRpad_DATA_VIA_LAN***
- ***XRpad_DATA_VIA_WLAN***

3.10.3.19 enum XRpad_SystemControlEnum

Possible system control actions for XRpad

Enumerator:

- ***XRpad_SYSTEM_CONTROL_REBOOT***
- ***XRpad_SYSTEM_CONTROL_RESTART_NETWORK***
- ***XRpad_SYSTEM_CONTROL_SHUTDOWN***
- ***XRpad_SYSTEM_CONTROL_SET_DEEP_SLEEP***
- ***XRpad_SYSTEM_CONTROL_SET_IDLE***

Chapter 4

Class Documentation

4.1 CHwHeaderInfo Struct Reference

Public Attributes

- int bAddRow
- BOOL bAddRow
- int bPwrSave
- BOOL bPwrSave
- int bSyncMode
- BOOL bSyncMode
- unsigned long dwAccess
- DWORD dwAccess
- unsigned long dwAcqMode
- DWORD dwAcqMode
- unsigned long dwBias
- DWORD dwBias
- unsigned long dwDataSorting
- DWORD dwDataSorting
- unsigned long dwDataType
- DWORD dwDataType
- unsigned long dwFrmFillRowIntervalls
- DWORD dwFrmFillRowIntervalls
- unsigned long dwFrmNrRows
- DWORD dwFrmNrRows
- unsigned long dwFrmRowType
- DWORD dwFrmRowType
- unsigned long dwGain
- DWORD dwGain
- unsigned long dwHeaderID
- DWORD dwHeaderID
- unsigned long dwLeakRows

- DWORD dwLeakRows
- unsigned long dwNrColumns
- DWORD dwNrColumns
- unsigned long dwNrOfFillingRows
- DWORD dwNrOfFillingRows
- unsigned long dwNrRows
- DWORD dwNrRows
- unsigned long dwOffset
- DWORD dwOffset
- unsigned long dwPROMID
- DWORD dwPROMID
- unsigned long dwTiming
- DWORD dwTiming
- unsigned long dwZoomBRColumn
- DWORD dwZoomBRColumn
- unsigned long dwZoomBRRow
- DWORD dwZoomBRRow
- unsigned long dwZoomULColumn
- DWORD dwZoomULColumn
- unsigned long dwZoomULRow
- DWORD dwZoomULRow

4.1.1 Member Data Documentation

4.1.1.1 int **CHwHeaderInfo::bAddRow**

4.1.1.2 BOOL **CHwHeaderInfo::bAddRow**

4.1.1.3 int **CHwHeaderInfo::bPwrSave**

4.1.1.4 BOOL **CHwHeaderInfo::bPwrSave**

4.1.1.5 int **CHwHeaderInfo::bSyncMode**

4.1.1.6 BOOL **CHwHeaderInfo::bSyncMode**

4.1.1.7 unsigned long **CHwHeaderInfo::dwAccess**

4.1.1.8 DWORD **CHwHeaderInfo::dwAccess**

4.1.1.9 unsigned long **CHwHeaderInfo::dwAcqMode**

4.1.1.10 DWORD **CHwHeaderInfo::dwAcqMode**

4.1.1.11 unsigned long **CHwHeaderInfo::dwBias**

4.1.1.12 DWORD CHwHeaderInfo::dwBias

4.1.1.13 unsigned long CHwHeaderInfo::dwDataSorting

4.1.1.14 DWORD CHwHeaderInfo::dwDataSorting

4.1.1.15 unsigned long CHwHeaderInfo::dwDataType

4.1.1.16 DWORD CHwHeaderInfo::dwDataType

4.1.1.17 unsigned long CHwHeaderInfo::dwFrmFillRowIntervalls

4.1.1.18 DWORD CHwHeaderInfo::dwFrmFillRowIntervalls

4.1.1.19 unsigned long CHwHeaderInfo::dwFrmNrRows

4.1.1.20 DWORD CHwHeaderInfo::dwFrmNrRows

4.1.1.21 unsigned long CHwHeaderInfo::dwFrmRowType

4.1.1.22 DWORD CHwHeaderInfo::dwFrmRowType

4.1.1.23 unsigned long CHwHeaderInfo::dwGain

4.1.1.24 DWORD CHwHeaderInfo::dwGain

4.1.1.25 unsigned long CHwHeaderInfo::dwHeaderID

4.1.1.26 DWORD CHwHeaderInfo::dwHeaderID

4.1.1.27 unsigned long CHwHeaderInfo::dwLeakRows

4.1.1.28 DWORD CHwHeaderInfo::dwLeakRows

4.1.1.29 unsigned long CHwHeaderInfo::dwNrColumns

4.1.1.30 DWORD CHwHeaderInfo::dwNrColumns

4.1.1.31 unsigned long CHwHeaderInfo::dwNrOfFillingRows

4.1.1.32 DWORD CHwHeaderInfo::dwNrOfFillingRows

4.1.1.33 unsigned long CHwHeaderInfo::dwNrRows

4.1.1.34 DWORD CHwHeaderInfo::dwNrRows

4.1.1.35 unsigned long CHwHeaderInfo::dwOffset

4.1.1.36 DWORD CHwHeaderInfo::dwOffset

4.1.1.37 unsigned long CHwHeaderInfo::dwPROMID

4.1.1.38 DWORD CHwHeaderInfo::dwPROMID

4.1.1.39 unsigned long CHwHeaderInfo::dwTiming

4.1.1.40 DWORD CHwHeaderInfo::dwTiming

4.1.1.41 unsigned long CHwHeaderInfo::dwZoomBRColumn

4.1.1.42 DWORD CHwHeaderInfo::dwZoomBRColumn

4.1.1.43 unsigned long CHwHeaderInfo::dwZoomBRRow

4.1.1.44 DWORD CHwHeaderInfo::dwZoomBRRow

4.1.1.45 unsigned long CHwHeaderInfo::dwZoomULColumn

4.1.1.46 DWORD CHwHeaderInfo::dwZoomULColumn

4.1.1.47 unsigned long CHwHeaderInfo::dwZoomULRow

4.1.1.48 DWORD CHwHeaderInfo::dwZoomULRow

4.2 CHwHeaderInfoEx Struct Reference

Public Attributes

- unsigned short wAccess
- WORD wAccess
- unsigned short wBias
- WORD wBias
- unsigned short wBinningMode
- WORD wBinningMode
- unsigned short wCameratype
- WORD wCameratype
- unsigned short wClock
- WORD wClock
- unsigned short wCommand1
- WORD wCommand1
- unsigned short wCommand2
- WORD wCommand2
- unsigned short wCommand3
- WORD wCommand3
- unsigned short wCommand4

- WORD wCommand4
- unsigned short wDataSorting
- WORD wDataSorting
- unsigned short wDummy
- WORD wDummy
- unsigned short wFrameCnt
- WORD wFrameCnt
- unsigned short wFrmNrRows
- WORD wFrmNrRows
- unsigned short wFrmRowType
- WORD wFrmRowType
- unsigned short wGain
- WORD wGain
- unsigned short wHeaderID
- WORD wHeaderID
- unsigned short wLeakRows
- WORD wLeakRows
- unsigned short wNrColumns
- WORD wNrColumns
- unsigned short wNrRows
- WORD wNrRows
- unsigned short wPROMID
- WORD wPROMID
- unsigned short wRealInttime_microSec
- WORD wRealInttime_microSec
- unsigned short wRealInttime_milliSec
- WORD wRealInttime_milliSec
- unsigned short wResolutionX
- WORD wResolutionX
- unsigned short wResolutionY
- WORD wResolutionY
- unsigned short wRowTime
- WORD wRowTime
- unsigned short wStatus
- WORD wStatus
- unsigned short wTiming
- WORD wTiming
- unsigned short wUgComp
- WORD wUgComp
- unsigned short wZoomBRColumn
- WORD wZoomBRColumn
- unsigned short wZoomBRRow
- WORD wZoomBRRow
- unsigned short wZoomULColumn
- WORD wZoomULColumn
- unsigned short wZoomULRow
- WORD wZoomULRow

4.2.1 Member Data Documentation

4.2.1.1 unsigned short CHwHeaderInfoEx::wAccess

4.2.1.2 WORD CHwHeaderInfoEx::wAccess

4.2.1.3 unsigned short CHwHeaderInfoEx::wBias

4.2.1.4 WORD CHwHeaderInfoEx::wBias

4.2.1.5 unsigned short CHwHeaderInfoEx::wBinningMode

4.2.1.6 WORD CHwHeaderInfoEx::wBinningMode

4.2.1.7 unsigned short CHwHeaderInfoEx::wCameratype

4.2.1.8 WORD CHwHeaderInfoEx::wCameratype

4.2.1.9 unsigned short CHwHeaderInfoEx::wClock

4.2.1.10 WORD CHwHeaderInfoEx::wClock

4.2.1.11 unsigned short CHwHeaderInfoEx::wCommand1

4.2.1.12 WORD CHwHeaderInfoEx::wCommand1

4.2.1.13 unsigned short CHwHeaderInfoEx::wCommand2

4.2.1.14 WORD CHwHeaderInfoEx::wCommand2

4.2.1.15 unsigned short CHwHeaderInfoEx::wCommand3

4.2.1.16 WORD CHwHeaderInfoEx::wCommand3

4.2.1.17 unsigned short CHwHeaderInfoEx::wCommand4

4.2.1.18 WORD CHwHeaderInfoEx::wCommand4

4.2.1.19 unsigned short CHwHeaderInfoEx::wDataSorting

4.2.1.20 WORD CHwHeaderInfoEx::wDataSorting

4.2.1.21 unsigned short CHwHeaderInfoEx::wDummy

4.2.1.22 WORD CHwHeaderInfoEx::wDummy

4.2.1.23 unsigned short CHwHeaderInfoEx::wFrameCnt

4.2.1.24 WORD CHwHeaderInfoEx::wFrameCnt

4.2.1.25 unsigned short CHwHeaderInfoEx::wFrmNrRows

4.2.1.26 WORD CHwHeaderInfoEx::wFrmNrRows

4.2.1.27 unsigned short CHwHeaderInfoEx::wFrmRowType

4.2.1.28 WORD CHwHeaderInfoEx::wFrmRowType

4.2.1.29 unsigned short CHwHeaderInfoEx::wGain

4.2.1.30 WORD CHwHeaderInfoEx::wGain

4.2.1.31 unsigned short CHwHeaderInfoEx::wHeaderID

4.2.1.32 WORD CHwHeaderInfoEx::wHeaderID

4.2.1.33 unsigned short CHwHeaderInfoEx::wLeakRows

4.2.1.34 WORD CHwHeaderInfoEx::wLeakRows

4.2.1.35 unsigned short CHwHeaderInfoEx::wNrColumns

4.2.1.36 WORD CHwHeaderInfoEx::wNrColumns

4.2.1.37 unsigned short CHwHeaderInfoEx::wNrRows

4.2.1.38 WORD CHwHeaderInfoEx::wNrRows

4.2.1.39 unsigned short CHwHeaderInfoEx::wPROMID

4.2.1.40 WORD CHwHeaderInfoEx::wPROMID

4.2.1.41 unsigned short CHwHeaderInfoEx::wRealInttime_microSec

4.2.1.42 WORD CHwHeaderInfoEx::wRealInttime_microSec

4.2.1.43 unsigned short CHwHeaderInfoEx::wRealInttime_milliSec

4.2.1.44 WORD CHwHeaderInfoEx::wRealInttime_milliSec

4.2.1.45 unsigned short CHwHeaderInfoEx::wResolutionX

4.2.1.46 WORD CHwHeaderInfoEx::wResolutionX

4.2.1.47 unsigned short CHwHeaderInfoEx::wResolutionY

4.2.1.48 WORD **CHwHeaderInfoEx::wResolutionY**

4.2.1.49 unsigned short **CHwHeaderInfoEx::wRowTime**

4.2.1.50 WORD **CHwHeaderInfoEx::wRowTime**

4.2.1.51 unsigned short **CHwHeaderInfoEx::wStatus**

4.2.1.52 WORD **CHwHeaderInfoEx::wStatus**

4.2.1.53 unsigned short **CHwHeaderInfoEx::wTiming**

4.2.1.54 WORD **CHwHeaderInfoEx::wTiming**

4.2.1.55 unsigned short **CHwHeaderInfoEx::wUgComp**

4.2.1.56 WORD **CHwHeaderInfoEx::wUgComp**

4.2.1.57 unsigned short **CHwHeaderInfoEx::wZoomBRColumn**

4.2.1.58 WORD **CHwHeaderInfoEx::wZoomBRColumn**

4.2.1.59 unsigned short **CHwHeaderInfoEx::wZoomBRRow**

4.2.1.60 WORD **CHwHeaderInfoEx::wZoomBRRow**

4.2.1.61 unsigned short **CHwHeaderInfoEx::wZoomULColumn**

4.2.1.62 WORD **CHwHeaderInfoEx::wZoomULColumn**

4.2.1.63 unsigned short **CHwHeaderInfoEx::wZoomULRow**

4.2.1.64 WORD **CHwHeaderInfoEx::wZoomULRow**

4.3 DETECTOR_BATTERY Struct Reference

Public Attributes

- DWORD capacity
- DWORD charge
- DWORD current
- DWORD cycle_count
- DWORD energy
- DWORD serial_no
- DWORD status
- DWORD temperature
- DWORD voltage

4.3.1 Member Data Documentation

4.3.1.1 DWORD DETECTOR_BATTERY::capacity

4.3.1.2 DWORD DETECTOR_BATTERY::charge

4.3.1.3 DWORD DETECTOR_BATTERY::current

4.3.1.4 DWORD DETECTOR_BATTERY::cycle_count

4.3.1.5 DWORD DETECTOR_BATTERY::energy

4.3.1.6 DWORD DETECTOR_BATTERY::serial_no

4.3.1.7 DWORD DETECTOR_BATTERY::status

4.3.1.8 DWORD DETECTOR_BATTERY::temperature

4.3.1.9 DWORD DETECTOR_BATTERY::voltage

4.4 DETECTOR_CURRENT_VOLTAGE Struct Reference

Public Attributes

- int imA1
- int imA2
- int imA3
- int iV1
- int iV2
- int iV3

4.4.1 Member Data Documentation

4.4.1.1 int DETECTOR_CURRENT_VOLTAGE::imA1

4.4.1.2 int DETECTOR_CURRENT_VOLTAGE::imA2

4.4.1.3 int DETECTOR_CURRENT_VOLTAGE::imA3

4.4.1.4 int DETECTOR_CURRENT_VOLTAGE::iV1

4.4.1.5 int DETECTOR_CURRENT_VOLTAGE::iV2

4.4.1.6 int DETECTOR_CURRENT_VOLTAGE::iV3

4.5 deviceInfo Struct Reference

Public Attributes

- char device_version [16]

  The device version.
- char manufacturer_name [32]

  The manufactures name.
- char manufacturer_specific [48]

  Some manufacture info.
- char model_name [32]

  The model name.
- char serial_number [16]

  The serial number.
- char spec_version [16]

  GigE vision spec version.
- char user_name [16]

  Device specific.

4.5.1 Detailed Description

The device info

4.5.2 Member Data Documentation

4.5.2.1 char **deviceInfo::device_version**[16]

The device version.

4.5.2.2 char **deviceInfo::manufacturer_name**[32]

The manufactures name.

4.5.2.3 char **deviceInfo::manufacturer_specific**[48]

Some manufacture info.

4.5.2.4 char **deviceInfo::model_name**[32]

The model name.

4.5.2.5 char **deviceInfo::serial_number**[16]

The serial number.

4.5.2.6 char **deviceInfo::spec_version**[16]

GigE vision spec version.

4.5.2.7 char **deviceInfo::user_name**[16]

Device specific.

4.6 discoveryReply Struct Reference

Public Attributes

- struct deviceInfo deviceInfo

  The device information.
- char gvcp_ip [16]

  Which IP address is used for image transfer.
- struct networkInfo lanInfo

  The LAN network setup.
- struct networkInfo wlanInfo

  The WLAN network setup.

4.6.1 Detailed Description

The original discovery reply

4.6.2 Member Data Documentation

4.6.2.1 struct **deviceInfo discoveryReply::deviceInfo**

The device information.

4.6.2.2 char **discoveryReply::gvcp_ip**[16]

Which IP address is used for image transfer.

4.6.2.3 struct **networkInfo discoveryReply::lanInfo**

The LAN network setup.

4.6.2.4 struct networkInfo discoveryReply::wlanInfo

The WLAN network setup.

4.7 discoveryReplyEx Struct Reference

Public Attributes

- struct deviceInfo deviceInfo

  The device information.
- char gvcp_ip [16]

  Which IP address is used for image transfer.
- struct networkInfo lanInfo

  The LAN network setup.
- unsigned messageCount

  How many messages this reply carries.
- struct discoveryReplyMsg messages [32]

  Info about the received reply packets.
- struct networkInfo wlanInfo

  The WLAN network setup.

4.7.1 Detailed Description

The extended discovery reply Can carry additional information

4.7.2 Member Data Documentation

4.7.2.1 struct deviceInfo discoveryReplyEx::deviceInfo

The device information.

4.7.2.2 char discoveryReplyEx::gvcp_ip[16]

Which IP address is used for image transfer.

4.7.2.3 struct networkInfo discoveryReplyEx::lanInfo

The LAN network setup.

4.7.2.4 unsigned discoveryReplyEx::messageCount

How many messages this reply carries.

4.7.2.5 struct **discoveryReplyMsg discoveryReplyEx::messages**[32]

Info about the received reply packets.

4.7.2.6 struct **networkInfo discoveryReplyEx::wlanInfo**

The WLAN network setup.

4.8 discoveryReplyMsg Struct Reference

Public Attributes

- char local_ip [16]
- char remote_ip [16]

4.8.1 Detailed Description

Information about a single discovery Reply network packet

4.8.2 Member Data Documentation

4.8.2.1 char **discoveryReplyMsg::local_ip**[16]

4.8.2.2 char **discoveryReplyMsg::remote_ip**[16]

4.9 EPC_REGISTER Struct Reference

Public Attributes

- DWORD active_network_config
- DETECTOR_BATTERY battery
- DWORD channel
- DWORD exam_flag
- DWORD lan_status_register
- DWORD power_state
- RTC_STRUCT rtc_value
- DWORD sdcard_state
- DWORD sdcard_usage
- DWORD signal_strength
- DWORD spartan_id
- CHwHeaderInfoEx spartan_register
- DWORD temperature_error_level [8]
- DWORD temperature_value [8]

- DWORD temperature_warning_level [8]
- DWORD version
- DWORD wlan_status_register

4.9.1 Member Data Documentation

4.9.1.1 DWORD EPC_REGISTER::active_network_config

4.9.1.2 DETECTOR_BATTERY EPC_REGISTER::battery

4.9.1.3 DWORD EPC_REGISTER::channel

4.9.1.4 DWORD EPC_REGISTER::exam_flag

4.9.1.5 DWORD EPC_REGISTER::lan_status_register

4.9.1.6 DWORD EPC_REGISTER::power_state

4.9.1.7 RTC_STRUCT EPC_REGISTER::rtc_value

4.9.1.8 DWORD EPC_REGISTER::sdcard_state

4.9.1.9 DWORD EPC_REGISTER::sdcard_usage

4.9.1.10 DWORD EPC_REGISTER::signal_strength

4.9.1.11 DWORD EPC_REGISTER::spartan_id

4.9.1.12 CHwHeaderInfoEx EPC_REGISTER::spartan_register

4.9.1.13 DWORD EPC_REGISTER::temperature_error_level[8]

4.9.1.14 DWORD EPC_REGISTER::temperature_value[8]

4.9.1.15 DWORD EPC_REGISTER::temperature_warning_level[8]

4.9.1.16 DWORD EPC_REGISTER::version

4.9.1.17 DWORD EPC_REGISTER::wlan_status_register

4.10 FPGAType Struct Reference

Public Attributes

- unsigned char wTiming
- unsigned char wValue0

- unsigned char wValue1
- unsigned char wValue2
- unsigned char wValue3
- unsigned char wValue4
- unsigned char wValue5
- unsigned char wValue6

4.10.1 Member Data Documentation

4.10.1.1 unsigned char **FPGAType::wTiming**

4.10.1.2 unsigned char **FPGAType::wValue0**

4.10.1.3 unsigned char **FPGAType::wValue1**

4.10.1.4 unsigned char **FPGAType::wValue2**

4.10.1.5 unsigned char **FPGAType::wValue3**

4.10.1.6 unsigned char **FPGAType::wValue4**

4.10.1.7 unsigned char **FPGAType::wValue5**

4.10.1.8 unsigned char **FPGAType::wValue6**

4.11 GBIF_Detector_Properties Struct Reference

Public Attributes

- char cDetectorType [32]
- char cDeviceIdentifier [16]
- char cDummy [48]
- char cManufacturingDate [8]
- char cPlaceOfManufacture [8]
- char cUniqueDeviceIdentifier [16]

4.11.1 Member Data Documentation

4.11.1.1 char **GBIF_Detector_Properties::cDetectorType**

4.11.1.2 char **GBIF_Detector_Properties::cDeviceIdentifier**[16]

4.11.1.3 char **GBIF_Detector_Properties::cDummy**

4.11.1.4 char **GBIF_Detector_Properties::cManufacturingDate**

4.11.1.5 char GBIF_Detector_Properties::cPlaceOfManufacture

4.11.1.6 char GBIF_Detector_Properties::cUniqueDeviceIdentifier[16]

4.12 GBIF_DEVICE_PARAM Struct Reference

Public Attributes

- char cDeviceName [16]
- CHAR cDeviceName [16]
- char cGBIFFirmwareVersion [32]
- CHAR cGBIFFirmwareVersion [32]
- char cManufacturerName [32]
- CHAR cManufacturerName [32]
- char cModelName [32]
- CHAR cModelName [32]
- unsigned long dwIPCurrentBootOptions
- DWORD dwIPCurrentBootOptions
- GBIF_STRING_DATATYPE ucAdapterIP [GBIF_IP_MAC_NAME_CHAR_ARRAY_LENGTH]
- GBIF_STRING_DATATYPE ucAdapterMask [GBIF_IP_MAC_NAME_CHAR_ARRAY_LENGTH]
- GBIF_STRING_DATATYPE ucGateway [GBIF_IP_MAC_NAME_CHAR_ARRAY_LENGTH]
- GBIF_STRING_DATATYPE ucIP [GBIF_IP_MAC_NAME_CHAR_ARRAY_LENGTH]
- GBIF_STRING_DATATYPE ucMacAddress [GBIF_IP_MAC_NAME_CHAR_ARRAY_LENGTH]
- GBIF_STRING_DATATYPE ucSubnetMask [GBIF_IP_MAC_NAME_CHAR_ARRAY_LENGTH]

4.12.1 Member Data Documentation

4.12.1.1 char GBIF_DEVICE_PARAM::cDeviceName[16]

4.12.1.2 CHAR GBIF_DEVICE_PARAM::cDeviceName[16]

4.12.1.3 char GBIF_DEVICE_PARAM::cGBIFFirmwareVersion[32]

4.12.1.4 CHAR GBIF_DEVICE_PARAM::cGBIFFirmwareVersion[32]

4.12.1.5 char GBIF_DEVICE_PARAM::cManufacturerName[32]

4.12.1.6 CHAR GBIF_DEVICE_PARAM::cManufacturerName[32]

4.12.1.7 char GBIF_DEVICE_PARAM::cModelName[32]

4.12.1.8 CHAR GBIF_DEVICE_PARAM::cModelName[32]

4.12.1.9 unsigned long GBIF_DEVICE_PARAM::dwIPCurrentBootOptions

4.12.1.10 DWORD GBIF_DEVICE_PARAM::dwIPCurrentBootOptions

4.12.1.11 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucAdapterIP

4.12.1.12 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucAdapterMask

4.12.1.13 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucGateway

4.12.1.14 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucIP

4.12.1.15 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucMacAddress

4.12.1.16 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucSubnetMask

4.13 networkAdapterConfiguration Struct Reference

Public Attributes

- int bridged

  Is the device in a bridge, this is not changeable by API.
- char dns [16]

  DNS server.
- int enabled

  Enabled flag, this is not changeable for LAN by API, but changeable for WLAN.
- char gateway [16]

  Gateway.
- int hw_accel

  Used image transfer, this is not changeable by API.
- char ifname [16]

  Interface name (eth0), this is not changeable by API.
- char ipaddr [16]

  IP address.
- char macaddr [18]

  MAC address, this is not changeable by API.
- char netmask [16]

  Netmask.
- char not_used [110]

  To fill up the struct to 320 byte.
- char proto [16]

  "static" or "dhcp", only these two options are available

4.13.1 Detailed Description

Structure for holding the adapter part of a configuration

4.13.2 Member Data Documentation

4.13.2.1 int **networkAdapterConfiguration::bridged**

Is the device in a bridge, this is not changeable by API.

4.13.2.2 char **networkAdapterConfiguration::dns**[16]

DNS server.

4.13.2.3 int **networkAdapterConfiguration::enabled**

Enabled flag, this is not changeable for LAN by API, but changeable for WLAN.

4.13.2.4 char **networkAdapterConfiguration::gateway**[16]

Gateway.

4.13.2.5 int **networkAdapterConfiguration::hw_accel**

Used image transfer, this is not changeable by API.

4.13.2.6 char **networkAdapterConfiguration::ifname**[16]

Interface name (eth0), this is not changeable by API.

4.13.2.7 char **networkAdapterConfiguration::ipaddr**[16]

IP address.

4.13.2.8 char **networkAdapterConfiguration::macaddr**[18]

MAC address, this is not changeable by API.

4.13.2.9 char **networkAdapterConfiguration::netmask**[16]

Netmask.

4.13.2.10 char **networkAdapterConfiguration::not_used**[110]

To fill up the struct to 320 byte.

4.13.2.11 char **networkAdapterConfiguration::proto**[16]

"static" or "dhcp", only these two options are available

4.14 networkConfiguration Struct Reference

Public Attributes

- int gbif_enabled
  Is the GBif enabled.
- char hostname [80]
  The hostname.
- struct networkAdapterConfiguration lan
  LAN.
- char name [80]
  The configuration name.
- char notUsed [184]
  For later extensions.
- char path [128]
  The configurations path, this is not changeable by API.
- int readonly
  Is the configuration readonly, this is not changeable by API.
- int sshd_enabled
  SSH daemon enabled.
- struct wifiConfigurationEx wifi
  Wifi configuration, extended version.
- struct networkAdapterConfiguration wlan
  WLAN.

4.14.1 Detailed Description

Structure for holding the complete network configuration.

4.14.2 Member Data Documentation

4.14.2.1 int **networkConfiguration::gbif_enabled**

Is the GBif enabled.

4.14.2.2 char **networkConfiguration::hostname**[80]

The hostname.

4.14.2.3 struct **networkAdapterConfiguration networkConfiguration::lan**

LAN.

4.14.2.4 char **networkConfiguration::name**[80]

The configuration name.

4.14.2.5 char **networkConfiguration::notUsed**[184]

For later extensions.

4.14.2.6 char **networkConfiguration::path**[128]

The configurations path, this is not changeable by API.

4.14.2.7 int **networkConfiguration::readonly**

Is the configuration readonly, this is not changeable by API.

4.14.2.8 int **networkConfiguration::sshd_enabled**

SSH daemon enabled.

4.14.2.9 struct **wifiConfigurationEx networkConfiguration::wifi**

Wifi configuration, extended version.

4.14.2.10 struct **networkAdapterConfiguration networkConfiguration::wlan**

WLAN.

4.15 networkInfo Struct Reference

Public Attributes

- char broadcast [16]

The IP broadcast as string.
- char ip [16]

  The IP (v4) address as string.
- char mac [18]

  The MAC address as string.
- char mask [16]

  The IP mask as string.

4.15.1 Detailed Description

The network information

4.15.2 Member Data Documentation

4.15.2.1 char **networkInfo::broadcast**[16]

The IP broadcast as string.

4.15.2.2 char **networkInfo::ip**[16]

The IP (v4) address as string.

4.15.2.3 char **networkInfo::mac**[18]

The MAC address as string.

4.15.2.4 char **networkInfo::mask**[16]

The IP mask as string.

4.16 RTC_STRUCT Struct Reference

Public Attributes

- DWORD day
- DWORD hour
- DWORD minute
- DWORD month
- DWORD second
- DWORD year

4.16.1 Member Data Documentation

4.16.1.1 DWORD **RTC_STRUCT::day**

4.16.1.2 DWORD **RTC_STRUCT::hour**

4.16.1.3 DWORD **RTC_STRUCT::minute**

4.16.1.4 DWORD **RTC_STRUCT::month**

4.16.1.5 DWORD **RTC_STRUCT::second**

4.16.1.6 DWORD **RTC_STRUCT::year**

4.17 wifiConfiguration Struct Reference

Public Attributes

- char agmode [32]

  agmode
- int channel

  Channel.
- char description [64]

  Contains the description in case of a station.
- char mode [32]

  Accesspoint or client.
- char ssid [64]

  Own SSID if mode == "ap" or the accesspoints ssid.

4.17.1 Detailed Description

Wifi configurations

4.17.2 Member Data Documentation

4.17.2.1 char **wifiConfiguration::agmode**[32]

agmode

4.17.2.2 int **wifiConfiguration::channel**

Channel.

4.17.2.3 char **wifiConfiguration::description**[64]

Contains the description in case of a station.

4.17.2.4 char **wifiConfiguration::mode**[32]

Accesspoint or client.

4.17.2.5 char **wifiConfiguration::ssid**[64]

Own SSID if mode == "ap" or the accesspoints ssid.

4.18 wifiConfigurationEx Struct Reference

Public Attributes

- char agmode [32]

  agmode
- int channel

  Channel, only valid options will be accepted, otherwise will return with error.
- char description [64]

  Contains the description in case of a station.
- char mode [32]

  Accesspoint or client.
- char passphrase [68]

  new Passphrase (station or accesspoint) (may be 64 byte)
- int scan_ssid

  only of station mode: scan the SSID
- char ssid [64]

  Own SSID if mode == "ap" or the accesspoints ssid.

4.18.1 Detailed Description

Wifi configurations, extended version

4.18.2 Member Data Documentation

4.18.2.1 char **wifiConfigurationEx::agmode**[32]

agmode

4.18.2.2 int wifiConfigurationEx::channel

Channel, only valid options will be accepted, otherwise will return with error.

4.18.2.3 char wifiConfigurationEx::description[64]

Contains the description in case of a station.

4.18.2.4 char wifiConfigurationEx::mode[32]

Accesspoint or client.

4.18.2.5 char wifiConfigurationEx::passphrase[68]

new Passphrase (station or accesspoint) (may be 64 byte)

4.18.2.6 int wifiConfigurationEx::scan_ssid

only of station mode: scan the SSID

4.18.2.7 char wifiConfigurationEx::ssid[64]

Own SSID if mode == "ap" or the accesspoints ssid.

4.19 WinHeaderType Struct Reference

Public Attributes

- unsigned short BRX
- WORD BRX
- unsigned short BRY
- WORD BRY
- unsigned short Correction
- WORD Correction
- unsigned long FileSize
- UINT FileSize
- unsigned short FileType
- WORD FileType
- unsigned short HeaderSize
- WORD HeaderSize
- unsigned short HeaderVersion
- WORD HeaderVersion
- unsigned short ImageHeaderSize

- WORD ImageHeaderSize
- double IntegrationTime
- unsigned short NrOfFrames
- WORD NrOfFrames
- unsigned short TypeOfNumbers
- WORD TypeOfNumbers
- unsigned short ULX
- WORD ULX
- unsigned short ULY
- WORD ULY
- unsigned char x [WINRESTSIZE]
- BYTE x [WINRESTSIZE]

4.19.1 Member Data Documentation

4.19.1.1 unsigned short WinHeaderType::BRX

4.19.1.2 WORD WinHeaderType::BRX

4.19.1.3 unsigned short WinHeaderType::BRY

4.19.1.4 WORD WinHeaderType::BRY

4.19.1.5 unsigned short WinHeaderType::Correction

4.19.1.6 WORD WinHeaderType::Correction

4.19.1.7 unsigned long WinHeaderType::FileSize

4.19.1.8 UINT WinHeaderType::FileSize

4.19.1.9 unsigned short WinHeaderType::FileType

4.19.1.10 WORD WinHeaderType::FileType

4.19.1.11 unsigned short WinHeaderType::HeaderSize

4.19.1.12 WORD WinHeaderType::HeaderSize

4.19.1.13 unsigned short WinHeaderType::HeaderVersion

4.19.1.14 WORD WinHeaderType::HeaderVersion

4.19.1.15 unsigned short WinHeaderType::ImageHeaderSize

4.19.1.16 WORD WinHeaderType::ImageHeaderSize

4.19.1.17 double WinHeaderType::IntegrationTime

4.19.1.18 unsigned short WinHeaderType::NrOfFrames

4.19.1.19 WORD WinHeaderType::NrOfFrames

4.19.1.20 unsigned short WinHeaderType::TypeOfNumbers

4.19.1.21 WORD WinHeaderType::TypeOfNumbers

4.19.1.22 unsigned short WinHeaderType::ULX

4.19.1.23 WORD WinHeaderType::ULX

4.19.1.24 unsigned short WinHeaderType::ULY

4.19.1.25 WORD WinHeaderType::ULY

4.19.1.26 unsigned char WinHeaderType::x[WINRESTSIZE]

4.19.1.27 BYTE WinHeaderType::x[WINRESTSIZE]

4.20 WinHeaderType101 Struct Reference

Public Attributes

- WORD BRX
- WORD BRY
- WORD Correction
- UINT FileSize
- WORD FileType
- WORD HeaderSize
- WORD HeaderVersion
- WORD ImageHeaderSize
- double IntegrationTime
- WORD NrOfFrames
- WORD TypeOfNumbers
- WORD ULX
- WORD ULY
- WORD wMedianValue
- BYTE x [WINRESTSIZE101]

4.20.1 Member Data Documentation

4.20.1.1 WORD WinHeaderType101::BRX

4.20.1.2 WORD WinHeaderType101::BRY

4.20.1.3 WORD WinHeaderType101::Correction

4.20.1.4 UINT WinHeaderType101::FileSize

4.20.1.5 WORD WinHeaderType101::FileType

4.20.1.6 WORD WinHeaderType101::HeaderSize

4.20.1.7 WORD WinHeaderType101::HeaderVersion

4.20.1.8 WORD WinHeaderType101::ImageHeaderSize

4.20.1.9 double WinHeaderType101::IntegrationTime

4.20.1.10 WORD WinHeaderType101::NrOfFrames

4.20.1.11 WORD WinHeaderType101::TypeOfNumbers

4.20.1.12 WORD WinHeaderType101::ULX

4.20.1.13 WORD WinHeaderType101::ULY

4.20.1.14 WORD WinHeaderType101::wMedianValue

4.20.1.15 BYTE WinHeaderType101::x[WINRESTSIZE101]

4.21 WinImageHeaderType Struct Reference

Public Attributes

- unsigned long dwPROMID
- DWORD dwPROMID
- float fAmpere
- float fKVolt
- unsigned short n_avframes
- WORD n_avframes
- char strPrefilter [9]
- char strProject [6]
- char strSystemused [3]

4.21.1 Member Data Documentation

4.21.1.1 unsigned long WinImageHeaderType::dwPROMID

4.21.1.2 DWORD WinImageHeaderType::dwPROMID

4.21.1.3 float WinImageHeaderType::fAmpere

4.21.1.4 float WinImageHeaderType::fKVolt

4.21.1.5 unsigned short WinImageHeaderType::n_avframes

4.21.1.6 WORD WinImageHeaderType::n_avframes

4.21.1.7 char WinImageHeaderType::strPrefilter

4.21.1.8 char WinImageHeaderType::strProject

4.21.1.9 char WinImageHeaderType::strSystemused

4.22 XislLoggingErrorHandler Class Reference

Public Member Functions

- virtual void error (const log4cplus::tstring &err)
- virtual void reset ()
- XislLoggingErrorHandler ()
- virtual ∼XislLoggingErrorHandler ()

4.22.1 Constructor & Destructor Documentation

4.22.1.1 XislLoggingErrorHandler::XislLoggingErrorHandler ( )

4.22.1.2 XislLoggingErrorHandler::∼XislLoggingErrorHandler ( ) `[virtual]`

4.22.2 Member Function Documentation

4.22.2.1 void XislLoggingErrorHandler::error (const log4cplus::tstring & *err*) `[virtual]`

4.22.2.2 void XislLoggingErrorHandler::reset ( ) `[virtual]`

4.23 XRpad_BatteryStatus Struct Reference

Public Attributes

- int authenticated
- XRpad_ChargeMode charge_mode
- int charge_state

- int cycle_count
- int design_capacity
- int health
- XRpad_BatteryPresence presence
- int remaining_capacity
- int temperature

4.23.1 Member Data Documentation

4.23.1.1 int **XRpad_BatteryStatus::authenticated**

4.23.1.2 **XRpad_ChargeMode XRpad_BatteryStatus::charge_mode**

4.23.1.3 int **XRpad_BatteryStatus::charge_state**

4.23.1.4 int **XRpad_BatteryStatus::cycle_count**

4.23.1.5 int **XRpad_BatteryStatus::design_capacity**

4.23.1.6 int **XRpad_BatteryStatus::health**

4.23.1.7 **XRpad_BatteryPresence XRpad_BatteryStatus::presence**

4.23.1.8 int **XRpad_BatteryStatus::remaining_capacity**

4.23.1.9 int **XRpad_BatteryStatus::temperature**

4.24 XRpad_ShockEvent Struct Reference

Public Attributes

- unsigned int critical_sensor1
- unsigned int critical_sensor2
- unsigned int critical_sensor3
- unsigned int timestamp
- unsigned int warning_sensor1
- unsigned int warning_sensor2
- unsigned int warning_sensor3

4.24.1 Member Data Documentation

4.24.1.1 unsigned int **XRpad_ShockEvent::critical_sensor1**

4.24.1.2 unsigned int **XRpad_ShockEvent::critical_sensor2**

4.24.1.3 unsigned int **XRpad_ShockEvent::critical_sensor3**

4.24.1.4 unsigned int **XRpad_ShockEvent::timestamp**

4.24.1.5 unsigned int **XRpad_ShockEvent::warning_sensor1**

4.24.1.6 unsigned int **XRpad_ShockEvent::warning_sensor2**

4.24.1.7 unsigned int **XRpad_ShockEvent::warning_sensor3**

4.25 XRpad_ShockSensorReport Struct Reference

Public Attributes

- XRpad_ShockEvent largest
- XRpad_ShockEvent latest

4.25.1 Member Data Documentation

4.25.1.1 **XRpad_ShockEvent XRpad_ShockSensorReport::largest**

4.25.1.2 **XRpad_ShockEvent XRpad_ShockSensorReport::latest**

4.26 XRpad_TempSensor Struct Reference

Public Attributes

- char index
- BOOL is_virtual
- char name [32]
- double temperature
- unsigned char warn_level

4.26.1 Member Data Documentation

4.26.1.1 char **XRpad_TempSensor::index**

4.26.1.2 BOOL **XRpad_TempSensor::is_virtual**

4.26.1.3 char **XRpad_TempSensor::name**[32]

4.26.1.4 double **XRpad_TempSensor::temperature**

4.26.1.5 unsigned char **XRpad_TempSensor::warn_level**

4.27 XRpad_TempSensorReport Struct Reference

Public Attributes

- unsigned char sensor_count
- XRpad_TempSensor sensors [16]
- unsigned int shutdown_time
- unsigned char system_warn_level

4.27.1 Member Data Documentation

4.27.1.1 unsigned char **XRpad_TempSensorReport::sensor_count**

4.27.1.2 **XRpad_TempSensor XRpad_TempSensorReport::sensors**[16]

4.27.1.3 unsigned int **XRpad_TempSensorReport::shutdown_time**

4.27.1.4 unsigned char **XRpad_TempSensorReport::system_warn_level**

4.28 XRpad_VersionInfo Struct Reference

Public Attributes

- char hwdriver [32]
- char linux_kernel [32]
- char msp_firmware [32]
- char pld_firmware [32]
- char software [32]
- char spartan_firmware [32]
- char subversion [256]
- char wlan [32]
- char xrpd [32]
- char zynq_firmware [32]

4.28.1 Member Data Documentation

4.28.1.1 char **XRpad_VersionInfo::hwdriver**[32]

4.28.1.2 char **XRpad_VersionInfo::linux_kernel**[32]

4.28.1.3 char **XRpad_VersionInfo::msp_firmware**[32]

4.28.1.4 char **XRpad_VersionInfo::pld_firmware**[32]

4.28.1.5 char **XRpad_VersionInfo::software**[32]

4.28.1.6 char **XRpad_VersionInfo::spartan_firmware**[32]

4.28.1.7 char **XRpad_VersionInfo::subversion**[256]

4.28.1.8 char **XRpad_VersionInfo::wlan**[32]

4.28.1.9 char **XRpad_VersionInfo::xrpd**[32]

4.28.1.10 char **XRpad_VersionInfo::zynq_firmware**[32]