

XISL: X-ray Imaging Software Library

Generated by Doxygen 1.7.6.1

Tue Jun 11 2019 15:27:58

Contents

1	Deprecated List	1
2	Module Index	3
2.1	Modules	3
3	Class Index	5
3.1	Class List	5
4	Module Documentation	7
4.1	Init Functions for XRpad	7
4.1.1	Function Documentation	8
4.1.1.1	Acquisition_GetVersionInfo	8
4.1.1.2	Acquisition_Set_OnboardOptions	8
4.1.1.3	Acquisition_Set_OnboardOptionsDualEnergy	9
4.1.1.4	Acquisition_Set_OnboardOptionsPostOffset	10
4.1.1.5	Acquisition_Set_OnboardOptionsPostOffsetEx	11
4.1.1.6	Acquisition_wpe_ActivateNetworkConfig	12
4.1.1.7	Acquisition_wpe_getAvailableSystems	12
4.1.1.8	Acquisition_wpe_GetNetworkConfigs	13
4.2	Special XRpad Functions	14
4.2.1	Function Documentation	19
4.2.1.1	Acq_wpe_DualEnergy_LoadCorrectionImageToBuffer	19
4.2.1.2	Acq_wpe_LoadCorrectionImageToBuffer	20
4.2.1.3	Acq_wpe_SetImageTransferInterface	20
4.2.1.4	Acq_wpe_SystemControl	21
4.2.1.5	Acquisition_AcknowledgeImage	21

4.2.1.6	Acquisition_AckSDCardForceFsck	22
4.2.1.7	Acquisition_AckSDCardForceFsckError	22
4.2.1.8	Acquisition_CloseFile	22
4.2.1.9	Acquisition_CreateFakeShockCriticalLevel	23
4.2.1.10	Acquisition_CreateFakeShockWarningLevel	23
4.2.1.11	Acquisition_DisableEventCallback	23
4.2.1.12	Acquisition_DisableSyslogSaving	24
4.2.1.13	Acquisition_Enable_EMI_Data_Readout	24
4.2.1.14	Acquisition_Enable_TestPattern	24
4.2.1.15	Acquisition_FactoryResetShock	25
4.2.1.16	Acquisition_FreeFTPFileBuffer	25
4.2.1.17	Acquisition_FTP_CloseSession	25
4.2.1.18	Acquisition_FTP_InitSession	26
4.2.1.19	Acquisition_Get_Current_Voltage	26
4.2.1.20	Acquisition_GetAutoDeepSleepIdleLocations	26
4.2.1.21	Acquisition_GetAutoPowerOnLocations	27
4.2.1.22	Acquisition_GetBatteryStatus	27
4.2.1.23	Acquisition_GetChargeMode	27
4.2.1.24	Acquisition_GetChargeModePAcqDesc	28
4.2.1.25	Acquisition_GetDefaultBootConfiguration	28
4.2.1.26	Acquisition_GetFTPFile	29
4.2.1.27	Acquisition_GetGridSensorStatus	29
4.2.1.28	Acquisition_GetLocation	29
4.2.1.29	Acquisition_GetMissedImageCount	30
4.2.1.30	Acquisition_GetNetwork	30
4.2.1.31	Acquisition_GetNetworkSpeed	30
4.2.1.32	Acquisition_GetPowerstate	31
4.2.1.33	Acquisition_GetSDCardInfo	31
4.2.1.34	Acquisition_GetSDCardTimeout	31
4.2.1.35	Acquisition_GetTemperature	32
4.2.1.36	Acquisition_GetTemperatureThresholds	32
4.2.1.37	Acquisition_GetWLAN_ChannelList	33
4.2.1.38	Acquisition_GetWLAN_CountryCode	33

4.2.1.39	Acquisition_IdentifyDevice	33
4.2.1.40	Acquisition_IsPreviewImage	34
4.2.1.41	Acquisition_IsSpartanIDLE	34
4.2.1.42	Acquisition_OpenMissedImage	34
4.2.1.43	Acquisition_Resend_All_Messages	35
4.2.1.44	Acquisition_Reset_OnboardOptions	35
4.2.1.45	Acquisition_ResetOnboardShockEvent	35
4.2.1.46	Acquisition_ResetTemperatureTimeout	36
4.2.1.47	Acquisition_Set_FPGA_Power_Mode	36
4.2.1.48	Acquisition_Set_OnboardOffsetImageAcquisition	36
4.2.1.49	Acquisition_Set_OnboardOptionPreview	37
4.2.1.50	Acquisition_SetAEDOptions	37
4.2.1.51	Acquisition_SetAutoDeepSleepIdleLocations	38
4.2.1.52	Acquisition_SetAutoPowerOnLocations	38
4.2.1.53	Acquisition_SetChargeMode	39
4.2.1.54	Acquisition_SetChargeModePAcqDesc	39
4.2.1.55	Acquisition_SetDefaultBootConfiguration	40
4.2.1.56	Acquisition_SetDualEnergyParams	40
4.2.1.57	Acquisition_SetEventCallback	40
4.2.1.58	Acquisition_SetFakeTemperature	41
4.2.1.59	Acquisition_SetFTPFile	41
4.2.1.60	Acquisition_SetIdleTimeout	42
4.2.1.61	Acquisition_SetNetworkSpeed	42
4.2.1.62	Acquisition_SetPhototimedParams	43
4.2.1.63	Acquisition_SetPrivateKey	43
4.2.1.64	Acquisition_SetSDCardForceFsch	43
4.2.1.65	Acquisition_SetSDCardTimeout	44
4.2.1.66	Acquisition_SetSpartanXISLToInitState	44
4.2.1.67	Acquisition_SetSystemTime	45
4.2.1.68	Acquisition_SetTailTimeforTriggerMode	45
4.2.1.69	Acquisition_SetTemperatureThresholds	46
4.2.1.70	Acquisition_SetTemperatureTimeout	46
4.2.1.71	Acquisition_SetWLAN_CountryCode	46

4.2.1.72	Acquisition_Test_SDCardPerformance	47
4.2.1.73	Acquisition_VerifyGenuineness	47
4.2.1.74	Acquisition_wpe_ChangeNetworkConfig	48
4.2.1.75	Acquisition_wpe_FillDefaultNetworkConfiguration	49
4.2.1.76	Acquisition_wpe_ForceIP	50
4.2.1.77	Acquisition_wpe_GetExamFlag	51
4.2.1.78	Acquisition_wpe_ReadCameraRegisters	51
4.2.1.79	Acquisition_wpe_ReadCameraRegistersPAcqDesc	52
4.2.1.80	Acquisition_wpe_SetMaxOnboardCorrValue	52
4.2.1.81	Acquisition_wpe_SetUniqueImageTag	52
4.2.1.82	Acquisition_xrpd_GetDetectorType	53
4.2.1.83	Acquisition_xrpd_GetExamFlag	53
4.2.1.84	Acquisition_xrpd_ReadCameraRegisters	54
4.2.1.85	Acquisition_xrpd_ReadCameraRegistersHACQDESC	54
4.2.1.86	Acquisition_xrpd_ReadCameraRegistersPAcqDesc	54
4.2.1.87	Acquisition_xrpd_WriteCameraRegistersHACQDESC	55
4.3	Common Init functions	56
4.3.1	Function Documentation	57
4.3.1.1	Acquisition_CallFPGASetCameraModelInternal	57
4.3.1.2	Acquisition_Close	57
4.3.1.3	Acquisition_CloseAll	58
4.3.1.4	Acquisition_EnableLogging	58
4.3.1.5	Acquisition_EnumSensors	58
4.3.1.6	Acquisition_GetCommChannel	59
4.3.1.7	Acquisition_GetConfiguration	59
4.3.1.8	Acquisition_GetHwHeaderInfo	60
4.3.1.9	Acquisition_GetHwHeaderInfoEx	61
4.3.1.10	Acquisition_GetIntTimes	61
4.3.1.11	Acquisition_GetLogLevel	62
4.3.1.12	Acquisition_GetNextSensor	62
4.3.1.13	Acquisition_Global_Cleanup	63
4.3.1.14	Acquisition_Global_Init	63
4.3.1.15	Acquisition_Init	64

4.3.1.16	Acquisition_SetCallbacksAndMessages	65
4.3.1.17	Acquisition_SetLogLevel	66
4.3.1.18	Acquisition_SetLogOutput	67
4.3.1.19	Acquisition_TogglePerformanceLogging	67
4.4	Init Functions for GbIF	68
4.4.1	Function Documentation	69
4.4.1.1	Acquisition_GbIF_CheckNetworkSpeed	69
4.4.1.2	Acquisition_GbIF_DiscoverDetectors	70
4.4.1.3	Acquisition_GbIF_DiscoveredDetectorByIndex	70
4.4.1.4	Acquisition_GbIF_DiscoveredDetectorCount	71
4.4.1.5	Acquisition_GbIF_ForceIP	71
4.4.1.6	Acquisition_GbIF_GetConnectionSettings	71
4.4.1.7	Acquisition_GbIF_GetDetectorProperties	72
4.4.1.8	Acquisition_GbIF_GetDevice	72
4.4.1.9	Acquisition_GbIF_GetDeviceCnt	73
4.4.1.10	Acquisition_GbIF_GetDeviceList	73
4.4.1.11	Acquisition_GbIF_GetDeviceParams	74
4.4.1.12	Acquisition_GbIF_GetFilterDrvState	74
4.4.1.13	Acquisition_GbIF_GetPacketDelay	74
4.4.1.14	Acquisition_GbIF_GetTransmissionMode	75
4.4.1.15	Acquisition_GbIF_GetVersion	75
4.4.1.16	Acquisition_GbIF_Init	76
4.4.1.17	Acquisition_GbIF_SetConnectionSettings	77
4.4.1.18	Acquisition_GbIF_SetDiscoveryTimeout	77
4.4.1.19	Acquisition_GbIF_SetPacketDelay	78
4.4.1.20	Acquisition_GbIF_SetPortRange	78
4.4.1.21	Acquisition_GbIF_SetTransmissionMode	78
4.4.1.22	Acquisition_wpe_GetVersionNEW	79
4.5	Acquire and correction functions	80
4.5.1	Function Documentation	83
4.5.1.1	Acquisition_Abort	83
4.5.1.2	Acquisition_AbortCurrentFrame	83
4.5.1.3	Acquisition_Acquire_GainImage	83

4.5.1.4	Acquisition_Acquire_GainImage_Ex	84
4.5.1.5	Acquisition_Acquire_GainImage_Ex_ROI	85
4.5.1.6	Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr	86
4.5.1.7	Acquisition_Acquire_GainImage_PreloadCorr	86
4.5.1.8	Acquisition_Acquire_Image	86
4.5.1.9	Acquisition_Acquire_Image_Ex	88
4.5.1.10	Acquisition_Acquire_Image_PreloadCorr	89
4.5.1.11	Acquisition_Acquire_OffsetImage	89
4.5.1.12	Acquisition_Acquire_OffsetImage_Ex	90
4.5.1.13	Acquisition_Acquire_OffsetImage_PreloadCorr	90
4.5.1.14	Acquisition_CreateGainMap	91
4.5.1.15	Acquisition_CreateGainMap32	91
4.5.1.16	Acquisition_CreateOnboardPixelMaskFrom16BitPixelMask	91
4.5.1.17	Acquisition_CreatePixelMap	92
4.5.1.18	Acquisition_DefineDestBuffers	92
4.5.1.19	Acquisition_DoOffsetCorrection	93
4.5.1.20	Acquisition_DoOffsetGainCorrection	93
4.5.1.21	Acquisition_DoOffsetGainCorrection32	94
4.5.1.22	Acquisition_DoOffsetGainCorrection_Ex	94
4.5.1.23	Acquisition_DoPixelCorrection	94
4.5.1.24	Acquisition_GetCorrData	95
4.5.1.25	Acquisition_GetCorrData_Ex	95
4.5.1.26	Acquisition_GetLatestFrameHeader	96
4.5.1.27	Acquisition_SetCorrData	96
4.5.1.28	Acquisition_SetCorrData_Ex	97
4.6	Setting up the Device	98
4.6.1	Function Documentation	99
4.6.1.1	Acquisition_GetCameraBinningMode	99
4.6.1.2	Acquisition_GetCameraFOVMode	99
4.6.1.3	Acquisition_GetCameraROI	100
4.6.1.4	Acquisition_GetCameraTriggerMode	100
4.6.1.5	Acquisition_GetIpAddress	100
4.6.1.6	Acquisition_ResetFrameCnt	101

4.6.1.7	Acquisition_SetCameraBinningMode	101
4.6.1.8	Acquisition_SetCameraFOVMode	102
4.6.1.9	Acquisition_SetCameraGain	102
4.6.1.10	Acquisition_SetCameraMode	103
4.6.1.11	Acquisition_SetCameraROI	103
4.6.1.12	Acquisition_SetCameraTriggerMode	103
4.6.1.13	Acquisition_SetDACOffset	104
4.6.1.14	Acquisition_SetDACOffsetBinningFPS	104
4.6.1.15	Acquisition_SetFrameSyncMode	105
4.6.1.16	Acquisition_SetFrameSyncTimeMode	106
4.6.1.17	Acquisition_SetTimerSync	106
4.6.1.18	Acquisition_SetTriggerOutSignalOptions	107
4.7	Callback function	109
4.7.1	Function Documentation	109
4.7.1.1	Acquisition_GetAcqData	109
4.7.1.2	Acquisition_GetActFrame	110
4.7.1.3	Acquisition_GetReady	110
4.7.1.4	Acquisition_GetWinHandle	110
4.7.1.5	Acquisition_SetAcqData	111
4.7.1.6	Acquisition_SetReady	111
4.8	Common error handling	113
4.8.1	Function Documentation	113
4.8.1.1	Acquisition_GetErrorCode	113
4.8.1.2	Acquisition_wpe_GetErrorCode	113
4.8.1.3	Acquisition_wpe_GetErrorCodeEx	113
4.9	functions provided by Acq.h	115
4.9.1	Function Documentation	117
4.9.1.1	Acq_wpe_GetSystemInformation	117
4.9.1.2	Acq_WPE_Init	117
4.9.1.3	Acquisition_ActivateServiceMode	117
4.9.1.4	Acquisition_CreateXISFileInMemory	117
4.9.1.5	Acquisition_DeleteFile	118
4.9.1.6	Acquisition_DoOffsetCorrection32	118

4.9.1.7	Acquisition_DoOffsetGainCorrection_Ex32	118
4.9.1.8	Acquisition_GetConnectionStatus	119
4.9.1.9	Acquisition_GetDACOffsetFloorValueFromFlash	119
4.9.1.10	Acquisition_GetDetectorProperties	119
4.9.1.11	Acquisition_GetFileInfo	120
4.9.1.12	Acquisition_GetHwHeader	120
4.9.1.13	Acquisition_GetRotationAngle	120
4.9.1.14	Acquisition_GetTriggerOutStatus	121
4.9.1.15	Acquisition_GetVersion	121
4.9.1.16	Acquisition_GetXISFileBufferSize	122
4.9.1.17	Acquisition_IsAcquiringData	122
4.9.1.18	Acquisition_LoadFile	123
4.9.1.19	Acquisition_LoadXISFileToMemory	123
4.9.1.20	Acquisition_SaveFile	123
4.9.1.21	Acquisition_SaveRawData	124
4.9.1.22	Acquisition_SetConsoleLogging	124
4.9.1.23	Acquisition_SetDACOffsetFloorValueByMode	125
4.9.1.24	Acquisition_SetDACOffsetFloorValueInFlash	125
4.9.1.25	Acquisition_SetDACOffsetFloorValueInFlashInternal	125
4.9.1.26	Acquisition_SetFileLogging	126
4.9.1.27	Acquisition_SetFPGACameraMode	126
4.9.1.28	Acquisition_SetRotationAngle	127
4.9.1.29	Acquisition_wpe_GetVersion	127
4.9.1.30	Acquisition_wpe_GetVersionEx	127
4.10	enumerations provided by Acq.h	129
4.10.1	Define Documentation	134
4.10.1.1	HIS_ALL_OK	134
4.10.1.2	HIS_ERROR_ABORT	134
4.10.1.3	HIS_ERROR_ABORTCURRFRAME	135
4.10.1.4	HIS_ERROR_ACKNOWLEDGE_IMAGE	135
4.10.1.5	HIS_ERROR_ACQ	135
4.10.1.6	HIS_ERROR_ACQ_ALREADY_RUNNING	135
4.10.1.7	HIS_ERROR_ACQABORT	135

4.10.1.8 HIS_ERROR_ACQUISITION	135
4.10.1.9 HIS_ERROR_ALREADY_EXISTS	135
4.10.1.10 HIS_ERROR_AVERAGED_LOST	135
4.10.1.11 HIS_ERROR_BAD_SORTING_PARAM	135
4.10.1.12 HIS_ERROR_BOARDINIT	135
4.10.1.13 HIS_ERROR_BUFFERSPACE_NOT_SUFF	136
4.10.1.14 HIS_ERROR_CONFLICT	136
4.10.1.15 HIS_ERROR_CORRBUFFER_INCOMPATIBLE	136
4.10.1.16 HIS_ERROR_CREATE_MEMORYMAPPING	136
4.10.1.17 HIS_ERROR_CREATE_MUTEX	136
4.10.1.18 HIS_ERROR_CURL	136
4.10.1.19 HIS_ERROR_DESC_NOT_LOCAL	136
4.10.1.20 HIS_ERROR_DOES_NOT_EXIST	136
4.10.1.21 HIS_ERROR_EMI_NOT_SET	136
4.10.1.22 HIS_ERROR_ENABLE_INTERRUPTS	136
4.10.1.23 HIS_ERROR_ENABLE_ONBOARD_GAINOFFSET	137
4.10.1.24 HIS_ERROR_ENABLE_ONBOARD_MEAN	137
4.10.1.25 HIS_ERROR_ENABLE_ONBOARD_OFFSET	137
4.10.1.26 HIS_ERROR_ENABLE_ONBOARD_PREVIEW	137
4.10.1.27 HIS_ERROR_FRAME_INV	137
4.10.1.28 HIS_ERROR_FUNC_NOTIMPL	137
4.10.1.29 HIS_ERROR_GET_AVAILABLE_SYSTEMS	137
4.10.1.30 HIS_ERROR_GET_CHARGE_MODE	137
4.10.1.31 HIS_ERROR_GET_NUM_BOARDS	137
4.10.1.32 HIS_ERROR_GET_ONBOARD_OFFSET	137
4.10.1.33 HIS_ERROR_GETHWHEADERINFO	138
4.10.1.34 HIS_ERROR_GETOSVERSION	138
4.10.1.35 HIS_ERROR_HEADER_TIMEOUT	138
4.10.1.36 HIS_ERROR_HW_ALREADY_OPEN_BY_ANOTHER_PROCESS	138
4.10.1.37 HIS_ERROR_HW_BOARD_CHANNEL_ALREADY_USED	138
4.10.1.38 HIS_ERROR_HWHEADER_INV	138
4.10.1.39 HIS_ERROR_ILLEGAL_INDEX	138
4.10.1.40 HIS_ERROR_INIT_DET_OPTIONS	138

4.10.1.41 HIS_ERROR_INVALID_FILENAME	138
4.10.1.42 HIS_ERROR_INVALID_FUNC_CALL	138
4.10.1.43 HIS_ERROR_INVALID_HANDLE	139
4.10.1.44 HIS_ERROR_INVALID_PARAM	139
4.10.1.45 HIS_ERROR_INVALIDACQDESC	139
4.10.1.46 HIS_ERROR_INVALIDBUFFERNR	139
4.10.1.47 HIS_ERROR_LOAD_COORECTIONIMAGETOBUFFER	139
4.10.1.48 HIS_ERROR_LOADDRIVER	139
4.10.1.49 HIS_ERROR_MEMORY	139
4.10.1.50 HIS_ERROR_MEMORY_MAPPING	139
4.10.1.51 HIS_ERROR_MISSING_VERSION_INFORMATION	139
4.10.1.52 HIS_ERROR_NO_BOARD_IN_SUBNET	139
4.10.1.53 HIS_ERROR_NO_FPGA_ACK	140
4.10.1.54 HIS_ERROR_NOCAMERA	140
4.10.1.55 HIS_ERROR_NODESC_AVAILABLE	140
4.10.1.56 HIS_ERROR_NOT_DISCOVERED	140
4.10.1.57 HIS_ERROR_NOT_INITIALIZED	140
4.10.1.58 HIS_ERROR_NOT_SUPPORTED	140
4.10.1.59 HIS_ERROR_NR_OF_BOARDS_CHANGED	140
4.10.1.60 HIS_ERROR_ONBOARDVAVGFAILED	140
4.10.1.61 HIS_ERROR_OPEN_FILE	140
4.10.1.62 HIS_ERROR_READ_DATA	140
4.10.1.63 HIS_ERROR_RESET_ZYNQ	141
4.10.1.64 HIS_ERROR_RETRIEVE_ENHANCED_HEADER	141
4.10.1.65 HIS_ERROR_SET_CHARGE_MODE	141
4.10.1.66 HIS_ERROR_SET_EVENT_CALLBACK	141
4.10.1.67 HIS_ERROR_SET_IDLE_TIMEOUT	141
4.10.1.68 HIS_ERROR_SET_IMAGE_TAG	141
4.10.1.69 HIS_ERROR_SET_IMAGE_TAG_LENGTH	141
4.10.1.70 HIS_ERROR_SET_ONBOARD_BINNING	141
4.10.1.71 HIS_ERROR_SET_PACKET_DELAY	141
4.10.1.72 HIS_ERROR_SET_PROC_SCRIPT	141
4.10.1.73 HIS_ERROR_SETBAUDRATE	142

4.10.1.74 HIS_ERROR_SETCAMERAMODE	142
4.10.1.75 HIS_ERROR_SETDISCOVERYTIMEOUT	142
4.10.1.76 HIS_ERROR_SETEXAMFLAG	142
4.10.1.77 HIS_ERROR_SETFRMSYNC	142
4.10.1.78 HIS_ERROR_SETFRMSYNCMODE	142
4.10.1.79 HIS_ERROR_SETHEADERSIZE	142
4.10.1.80 HIS_ERROR_SETLINETRIG_MODE	142
4.10.1.81 HIS_ERROR_SETREGISTERTIMEOUT	142
4.10.1.82 HIS_ERROR_SETTIMERSYNC	142
4.10.1.83 HIS_ERROR_SLOW_SYSTEM	143
4.10.1.84 HIS_ERROR_TIMEOUT	143
4.10.1.85 HIS_ERROR_TRANSMISSION_MODE	143
4.10.1.86 HIS_ERROR_UNABLE_TO_ACCESS_DETECTOR_FLASH	143
4.10.1.87 HIS_ERROR_UNABLE_TO_CLOSE_BOARD	143
4.10.1.88 HIS_ERROR_UNABLE_TO_OPEN_BOARD	143
4.10.1.89 HIS_ERROR_UNKNOWN_IP_MAC_NAME	143
4.10.1.90 HIS_ERROR_VXD_REGISTER_DMA_ADDRESS	143
4.10.1.91 HIS_ERROR_VXD_REGISTER_IRQ	143
4.10.1.92 HIS_ERROR_VXD_REGISTER_STAT_ADDR	143
4.10.1.93 HIS_ERROR_VXD_REGISTER_STATADR	144
4.10.1.94 HIS_ERROR_VXD_UNMASK_IRQ	144
4.10.1.95 HIS_ERROR_VXDGETDMAADR	144
4.10.1.96 HIS_ERROR_VXDNOTFOUND	144
4.10.1.97 HIS_ERROR_VXDNOTOPEN	144
4.10.1.98 HIS_ERROR_VXDUNKNOWNERROR	144
4.10.1.99 HIS_ERROR_WLAN_RESTART	144
4.10.1.100 HIS_ERROR_WRITE_DATA	144
4.10.1.101 HIS_ERROR_WRONG_CAMERA_MODE	144
4.10.1.102 HIS_ERROR_WRONGBOARDSELECT	144
4.10.1.103 HIS_ERROR_WSA	145
4.10.1.104 HIS_ERROR_XRPD_BATTERY_COM	145
4.10.1.105 HIS_ERROR_XRPD_CONNECT	145
4.10.1.106 HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT	145

4.10.1.107	HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_CRIT	145
4.10.1.108	HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_WARN	145
4.10.1.109	HIS_ERROR_XRPD_DETECTOR_IN_DEEP_SLEEP	145
4.10.1.110	HIS_ERROR_XRPD_DISABLE_SYSLOG_SAVING	145
4.10.1.111	HIS_ERROR_XRPD_FACTORY_RESET_SHOCK_EVENT	145
4.10.1.112	HIS_ERROR_XRPD_GET_AUTOPOWERONLOCATIONS	145
4.10.1.113	HIS_ERROR_XRPD_GET_CURRENT_VOLTAGE	146
4.10.1.114	HIS_ERROR_XRPD_GET_DEEPSLEEPIDLELOCATIONS	146
4.10.1.115	HIS_ERROR_XRPD_GET_DEF_BOOT_CFG	146
4.10.1.116	HIS_ERROR_XRPD_GET_DET_OPTIONS	146
4.10.1.117	HIS_ERROR_XRPD_GET_DET_TYPE	146
4.10.1.118	HIS_ERROR_XRPD_GET_EPC_REGISTER	146
4.10.1.119	HIS_ERROR_XRPD_GET_SDCARD_INFO	146
4.10.1.120	HIS_ERROR_XRPD_GET_SDCARD_TIMEOUT	146
4.10.1.121	HIS_ERROR_XRPD_GET_TEMP_VALUES	146
4.10.1.122	HIS_ERROR_XRPD_GET_TEMPERATURE_THRESHOLDS	146
4.10.1.123	HIS_ERROR_XRPD_GET_WLAN_CC	147
4.10.1.124	HIS_ERROR_XRPD_GET_WLAN_ChannelList	147
4.10.1.125	HIS_ERROR_XRPD_NO_EVENT_INTERFACE	147
4.10.1.126	HIS_ERROR_XRPD_NO_EVENTCALLBACK_DEFINED	147
4.10.1.127	HIS_ERROR_XRPD_NO_LOCATION	147
4.10.1.128	HIS_ERROR_XRPD_NO_NETWORK	147
4.10.1.129	HIS_ERROR_XRPD_NOT_CONNECTED	147
4.10.1.130	HIS_ERROR_XRPD_REQUEST_POWERSTATE	147
4.10.1.131	HIS_ERROR_XRPD_RESEND_ALL_MSG	147
4.10.1.132	HIS_ERROR_XRPD_RESET_SHOCK	147
4.10.1.133	HIS_ERROR_XRPD_RESET_TEMPERATURE_TIMEOUT	148
4.10.1.134	HIS_ERROR_XRPD_SDCARDPERFORMANCE	148
4.10.1.135	HIS_ERROR_XRPD_SESSION_ERROR	148
4.10.1.136	HIS_ERROR_XRPD_SET_AUTOPOWERONLOCATIONS	148
4.10.1.137	HIS_ERROR_XRPD_SET_CPUFREQ_GOVERNOR	148
4.10.1.138	HIS_ERROR_XRPD_SET_DATE_TIME	148
4.10.1.139	HIS_ERROR_XRPD_SET_DEEPSLEEPIDLELOCATIONS	148

4.10.1.140	HIS_ERROR_XRPD_SET_DEF_BOOT_CFG	148
4.10.1.141	HIS_ERROR_XRPD_SET_EPC_REGISTER	148
4.10.1.142	HIS_ERROR_XRPD_SET_EVENT	148
4.10.1.143	HIS_ERROR_XRPD_SET_FORCE_FSCK	149
4.10.1.144	HIS_ERROR_XRPD_SET_NETWORK	149
4.10.1.145	HIS_ERROR_XRPD_SET_PRIVATE_KEY	149
4.10.1.146	HIS_ERROR_XRPD_SET_SDCARD_TIMEOUT	149
4.10.1.147	HIS_ERROR_XRPD_SET_TEMP_FAKE_MODE	149
4.10.1.148	HIS_ERROR_XRPD_SET_TEMPERATURE_THRESHOLDS	149
4.10.1.149	HIS_ERROR_XRPD_SET_TEMPERATURE_TIMEOUT	149
4.10.1.150	HIS_ERROR_XRPD_SET_WLAN_CC	149
4.10.1.151	HIS_ERROR_XRPD_VERIFY_GENUINENESS	149
4.10.2	Typedef Documentation	149
4.10.2.1	ACQDESCPOS	149
4.10.2.2	HACQDESC	150
4.10.2.3	OnboardBinningMode	150
4.10.2.4	ProcScriptOperation	150
4.10.2.5	XIS_Acquisition_Event	150
4.10.2.6	XIS_Battery_Event	150
4.10.2.7	XIS_Detector_Event	150
4.10.2.8	XIS_Detector_TRIGOUT_SignalMode	150
4.10.2.9	XIS_DetectorTriggerMode	150
4.10.2.10	XIS_Event	150
4.10.2.11	XIS_FileType	150
4.10.2.12	XIS_Init_Flags	150
4.10.2.13	XIS_Library_Event	150
4.10.2.14	XIS_Sensor_Event	150
4.10.2.15	XIS_Transmission_Mode	150
4.10.2.16	XIS_Xrpd_Event	151
4.10.2.17	XislFileEntryType	151
4.10.2.18	XislFileHandle	151
4.10.2.19	XislFileStorageLocation	151
4.10.2.20	XislFtpSession	151

4.10.2.21	XRpad_BatteryHealth	151
4.10.2.22	XRpad_BatteryPresence	151
4.10.2.23	XRpad_BatteryStatus	151
4.10.2.24	XRpad_ChargeMode	151
4.10.2.25	XRpad_DataInterfaceControlEnum	151
4.10.2.26	XRpad_ShockEvent	151
4.10.2.27	XRpad_ShockSensorReport	151
4.10.2.28	XRpad_TempSensor	151
4.10.2.29	XRpad_TempSensorReport	151
4.10.2.30	XRpad_VersionInfo	151
4.10.3	Enumeration Type Documentation	151
4.10.3.1	OnboardBinningMode	151
4.10.3.2	ProcScriptOperation	152
4.10.3.3	XIS_Acquisition_Event	152
4.10.3.4	XIS_Battery_Event	152
4.10.3.5	XIS_Detector_Event	152
4.10.3.6	XIS_Detector_TRIGOUT_SignalMode	153
4.10.3.7	XIS_DetectorTriggerMode	153
4.10.3.8	XIS_Event	153
4.10.3.9	XIS_FileType	154
4.10.3.10	XIS_Init_Flags	154
4.10.3.11	XIS_Library_Event	154
4.10.3.12	XIS_Sensor_Event	155
4.10.3.13	XIS_Transmission_Mode	155
4.10.3.14	XIS_Xrpd_Event	155
4.10.3.15	XislFileEntryType	155
4.10.3.16	XislFileStorageLocation	155
4.10.3.17	XislLoggingLevels	156
4.10.3.18	XRpad_BatteryHealth	156
4.10.3.19	XRpad_BatteryPresence	156
4.10.3.20	XRpad_ChargeMode	156
4.10.3.21	XRpad_DataInterfaceControlEnum	157
4.10.3.22	XRpad_SystemControlEnum	157

5	Class Documentation	159
5.1	CHwHeaderInfo Struct Reference	159
5.1.1	Member Data Documentation	160
5.1.1.1	bAddRow	160
5.1.1.2	bAddRow	160
5.1.1.3	bPwrSave	160
5.1.1.4	bPwrSave	160
5.1.1.5	bSyncMode	160
5.1.1.6	bSyncMode	160
5.1.1.7	dwAccess	160
5.1.1.8	dwAccess	160
5.1.1.9	dwAcqMode	160
5.1.1.10	dwAcqMode	160
5.1.1.11	dwBias	161
5.1.1.12	dwBias	161
5.1.1.13	dwDataSorting	161
5.1.1.14	dwDataSorting	161
5.1.1.15	dwDataType	161
5.1.1.16	dwDataType	161
5.1.1.17	dwFrmFillRowIntervalls	161
5.1.1.18	dwFrmFillRowIntervalls	161
5.1.1.19	dwFrmNrRows	161
5.1.1.20	dwFrmNrRows	161
5.1.1.21	dwFrmRowType	161
5.1.1.22	dwFrmRowType	161
5.1.1.23	dwGain	161
5.1.1.24	dwGain	161
5.1.1.25	dwHeaderID	161
5.1.1.26	dwHeaderID	161
5.1.1.27	dwLeakRows	161
5.1.1.28	dwLeakRows	161
5.1.1.29	dwNrColumns	161
5.1.1.30	dwNrColumns	161

5.1.1.31	dwNrOfFillingRows	161
5.1.1.32	dwNrOfFillingRows	161
5.1.1.33	dwNrRows	161
5.1.1.34	dwNrRows	161
5.1.1.35	dwOffset	162
5.1.1.36	dwOffset	162
5.1.1.37	dwPROMID	162
5.1.1.38	dwPROMID	162
5.1.1.39	dwTiming	162
5.1.1.40	dwTiming	162
5.1.1.41	dwZoomBRColumn	162
5.1.1.42	dwZoomBRColumn	162
5.1.1.43	dwZoomBRRow	162
5.1.1.44	dwZoomBRRow	162
5.1.1.45	dwZoomULColumn	162
5.1.1.46	dwZoomULColumn	162
5.1.1.47	dwZoomULRow	162
5.1.1.48	dwZoomULRow	162
5.2	CHwHeaderInfoEx Struct Reference	162
5.2.1	Member Data Documentation	164
5.2.1.1	wAccess	164
5.2.1.2	wAccess	164
5.2.1.3	wBias	164
5.2.1.4	wBias	164
5.2.1.5	wBinningMode	164
5.2.1.6	wBinningMode	164
5.2.1.7	wCameratype	164
5.2.1.8	wCameratype	164
5.2.1.9	wClock	164
5.2.1.10	wClock	164
5.2.1.11	wCommand1	164
5.2.1.12	wCommand1	164
5.2.1.13	wCommand2	164

5.2.1.14	wCommand2	164
5.2.1.15	wCommand3	164
5.2.1.16	wCommand3	164
5.2.1.17	wCommand4	164
5.2.1.18	wCommand4	164
5.2.1.19	wDataSorting	164
5.2.1.20	wDataSorting	164
5.2.1.21	wDummy	165
5.2.1.22	wDummy	165
5.2.1.23	wFrameCnt	165
5.2.1.24	wFrameCnt	165
5.2.1.25	wFrmNrRows	165
5.2.1.26	wFrmNrRows	165
5.2.1.27	wFrmRowType	165
5.2.1.28	wFrmRowType	165
5.2.1.29	wGain	165
5.2.1.30	wGain	165
5.2.1.31	wHeaderID	165
5.2.1.32	wHeaderID	165
5.2.1.33	wLeakRows	165
5.2.1.34	wLeakRows	165
5.2.1.35	wNrColumns	165
5.2.1.36	wNrColumns	165
5.2.1.37	wNrRows	165
5.2.1.38	wNrRows	165
5.2.1.39	wPROMID	165
5.2.1.40	wPROMID	165
5.2.1.41	wRealInttime_microSec	165
5.2.1.42	wRealInttime_microSec	165
5.2.1.43	wRealInttime_milliSec	165
5.2.1.44	wRealInttime_milliSec	165
5.2.1.45	wResolutionX	166
5.2.1.46	wResolutionX	166

5.2.1.47	wResolutionY	166
5.2.1.48	wResolutionY	166
5.2.1.49	wRowTime	166
5.2.1.50	wRowTime	166
5.2.1.51	wStatus	166
5.2.1.52	wStatus	166
5.2.1.53	wTiming	166
5.2.1.54	wTiming	166
5.2.1.55	wUgComp	166
5.2.1.56	wUgComp	166
5.2.1.57	wZoomBRColumn	166
5.2.1.58	wZoomBRColumn	166
5.2.1.59	wZoomBRRow	166
5.2.1.60	wZoomBRRow	166
5.2.1.61	wZoomULColumn	166
5.2.1.62	wZoomULColumn	166
5.2.1.63	wZoomULRow	166
5.2.1.64	wZoomULRow	166
5.3	DETECTOR_BATTERY Struct Reference	166
5.3.1	Member Data Documentation	167
5.3.1.1	capacity	167
5.3.1.2	charge	167
5.3.1.3	current	167
5.3.1.4	cycle_count	167
5.3.1.5	energy	167
5.3.1.6	serial_no	167
5.3.1.7	status	167
5.3.1.8	temperature	167
5.3.1.9	voltage	167
5.4	DETECTOR_CURRENT_VOLTAGE Struct Reference	167
5.4.1	Member Data Documentation	167
5.4.1.1	imA1	167
5.4.1.2	imA2	167

5.4.1.3	imA3	168
5.4.1.4	iV1	168
5.4.1.5	iV2	168
5.4.1.6	iV3	168
5.5	deviceInfo Struct Reference	168
5.5.1	Detailed Description	168
5.5.2	Member Data Documentation	168
5.5.2.1	device_version	168
5.5.2.2	manufacturer_name	168
5.5.2.3	manufacturer_specific	169
5.5.2.4	model_name	169
5.5.2.5	serial_number	169
5.5.2.6	spec_version	169
5.5.2.7	user_name	169
5.6	discoveryReply Struct Reference	169
5.6.1	Detailed Description	169
5.6.2	Member Data Documentation	169
5.6.2.1	deviceInfo	169
5.6.2.2	gvcp_ip	170
5.6.2.3	lanInfo	170
5.6.2.4	wlanInfo	170
5.7	discoveryReplyEx Struct Reference	170
5.7.1	Detailed Description	170
5.7.2	Member Data Documentation	170
5.7.2.1	deviceInfo	170
5.7.2.2	gvcp_ip	170
5.7.2.3	lanInfo	171
5.7.2.4	messageCount	171
5.7.2.5	messages	171
5.7.2.6	wlanInfo	171
5.8	discoveryReplyMsg Struct Reference	171
5.8.1	Detailed Description	171
5.8.2	Member Data Documentation	171

5.8.2.1	local_ip	171
5.8.2.2	remote_ip	171
5.9	EPC_REGISTER Struct Reference	171
5.9.1	Member Data Documentation	172
5.9.1.1	active_network_config	172
5.9.1.2	battery	172
5.9.1.3	channel	172
5.9.1.4	exam_flag	172
5.9.1.5	lan_status_register	172
5.9.1.6	power_state	172
5.9.1.7	rtc_value	172
5.9.1.8	sdcard_state	172
5.9.1.9	sdcard_usage	172
5.9.1.10	signal_strength	172
5.9.1.11	spartan_id	172
5.9.1.12	spartan_register	172
5.9.1.13	temperature_error_level	172
5.9.1.14	temperature_value	172
5.9.1.15	temperature_warning_level	172
5.9.1.16	version	173
5.9.1.17	wlan_status_register	173
5.10	FPGAType Struct Reference	173
5.10.1	Member Data Documentation	173
5.10.1.1	wTiming	173
5.10.1.2	wValue0	173
5.10.1.3	wValue1	173
5.10.1.4	wValue2	173
5.10.1.5	wValue3	173
5.10.1.6	wValue4	173
5.10.1.7	wValue5	173
5.10.1.8	wValue6	173
5.11	GBIF_Detector_Properties Struct Reference	173
5.11.1	Member Data Documentation	174

5.11.1.1	cDetectorType	174
5.11.1.2	cDeviceIdentifier	174
5.11.1.3	cDummy	174
5.11.1.4	cManufacturingDate	174
5.11.1.5	cPlaceOfManufacture	174
5.11.1.6	cUniqueDeviceIdentifier	174
5.12	GBIF_DEVICE_PARAM Struct Reference	174
5.12.1	Member Data Documentation	174
5.12.1.1	cDeviceName	174
5.12.1.2	cDeviceName	174
5.12.1.3	cGBIFFirmwareVersion	174
5.12.1.4	cGBIFFirmwareVersion	174
5.12.1.5	cManufacturerName	175
5.12.1.6	cManufacturerName	175
5.12.1.7	cModelName	175
5.12.1.8	cModelName	175
5.12.1.9	dwIPCurentBootOptions	175
5.12.1.10	dwIPCurentBootOptions	175
5.12.1.11	ucAdapterIP	175
5.12.1.12	ucAdapterMask	175
5.12.1.13	ucGateway	175
5.12.1.14	ucIP	175
5.12.1.15	ucMacAddress	175
5.12.1.16	ucSubnetMask	175
5.13	networkAdapterConfiguration Struct Reference	175
5.13.1	Detailed Description	176
5.13.2	Member Data Documentation	176
5.13.2.1	bridged	176
5.13.2.2	dns	176
5.13.2.3	enabled	176
5.13.2.4	gateway	176
5.13.2.5	hw_accel	176
5.13.2.6	ifname	176

5.13.2.7	ipaddr	176
5.13.2.8	macaddr	177
5.13.2.9	netmask	177
5.13.2.10	not_used	177
5.13.2.11	proto	177
5.14	networkConfiguration Struct Reference	177
5.14.1	Detailed Description	178
5.14.2	Member Data Documentation	178
5.14.2.1	gbif_enabled	178
5.14.2.2	hostname	178
5.14.2.3	lan	178
5.14.2.4	name	178
5.14.2.5	notUsed	178
5.14.2.6	path	178
5.14.2.7	readonly	178
5.14.2.8	sshd_enabled	178
5.14.2.9	wifi	178
5.14.2.10	wlan	179
5.15	networkInfo Struct Reference	179
5.15.1	Detailed Description	179
5.15.2	Member Data Documentation	179
5.15.2.1	broadcast	179
5.15.2.2	ip	179
5.15.2.3	mac	179
5.15.2.4	mask	179
5.16	RTC_STRUCT Struct Reference	180
5.16.1	Member Data Documentation	180
5.16.1.1	day	180
5.16.1.2	hour	180
5.16.1.3	minute	180
5.16.1.4	month	180
5.16.1.5	second	180
5.16.1.6	year	180

5.17	wifiConfiguration Struct Reference	180
5.17.1	Detailed Description	180
5.17.2	Member Data Documentation	181
5.17.2.1	agmode	181
5.17.2.2	channel	181
5.17.2.3	description	181
5.17.2.4	mode	181
5.17.2.5	ssid	181
5.18	wifiConfigurationEx Struct Reference	181
5.18.1	Detailed Description	182
5.18.2	Member Data Documentation	182
5.18.2.1	agmode	182
5.18.2.2	channel	182
5.18.2.3	description	182
5.18.2.4	mode	182
5.18.2.5	passphrase	182
5.18.2.6	scan_ssid	182
5.18.2.7	ssid	182
5.19	WinHeaderType Struct Reference	182
5.19.1	Member Data Documentation	183
5.19.1.1	BRX	183
5.19.1.2	BRX	183
5.19.1.3	BRY	184
5.19.1.4	BRY	184
5.19.1.5	Correction	184
5.19.1.6	Correction	184
5.19.1.7	FileSize	184
5.19.1.8	FileSize	184
5.19.1.9	FileType	184
5.19.1.10	FileType	184
5.19.1.11	HeaderSize	184
5.19.1.12	HeaderSize	184
5.19.1.13	HeaderVersion	184

5.19.1.14 HeaderVersion	184
5.19.1.15 ImageHeaderSize	184
5.19.1.16 ImageHeaderSize	184
5.19.1.17 IntegrationTime	185
5.19.1.18 NrOfFrames	185
5.19.1.19 NrOfFrames	185
5.19.1.20 TypeOfNumbers	185
5.19.1.21 TypeOfNumbers	185
5.19.1.22 ULX	185
5.19.1.23 ULX	185
5.19.1.24 ULY	185
5.19.1.25 ULY	185
5.19.1.26 x	185
5.19.1.27 x	185
5.20 WinHeaderType101 Struct Reference	185
5.20.1 Member Data Documentation	186
5.20.1.1 BRX	186
5.20.1.2 BRY	186
5.20.1.3 Correction	186
5.20.1.4 FileSize	186
5.20.1.5 FileType	186
5.20.1.6 HeaderSize	187
5.20.1.7 HeaderVersion	187
5.20.1.8 ImageHeaderSize	187
5.20.1.9 IntegrationTime	187
5.20.1.10 NrOfFrames	187
5.20.1.11 TypeOfNumbers	187
5.20.1.12 ULX	187
5.20.1.13 ULY	187
5.20.1.14 wMedianValue	187
5.20.1.15 x	187
5.21 WinImageHeaderType Struct Reference	187
5.21.1 Member Data Documentation	188

5.21.1.1	dwPROMID	188
5.21.1.2	dwPROMID	188
5.21.1.3	fAmpere	188
5.21.1.4	fKVolt	188
5.21.1.5	n_avframes	188
5.21.1.6	n_avframes	188
5.21.1.7	strPrefilter	188
5.21.1.8	strProject	188
5.21.1.9	strSystemused	188
5.22	XRpad_BatteryStatus Struct Reference	188
5.22.1	Member Data Documentation	189
5.22.1.1	authenticated	189
5.22.1.2	charge_mode	189
5.22.1.3	charge_state	189
5.22.1.4	cycle_count	189
5.22.1.5	design_capacity	189
5.22.1.6	health	189
5.22.1.7	presence	189
5.22.1.8	remaining_capacity	190
5.22.1.9	temperature	190
5.23	XRpad_ShockEvent Struct Reference	190
5.23.1	Member Data Documentation	190
5.23.1.1	critical_sensor1	190
5.23.1.2	critical_sensor2	190
5.23.1.3	critical_sensor3	190
5.23.1.4	timestamp	190
5.23.1.5	warning_sensor1	190
5.23.1.6	warning_sensor2	190
5.23.1.7	warning_sensor3	190
5.24	XRpad_ShockSensorReport Struct Reference	190
5.24.1	Member Data Documentation	191
5.24.1.1	largest	191
5.24.1.2	latest	191

5.25	XRpad_TempSensor Struct Reference	191
5.25.1	Member Data Documentation	191
5.25.1.1	index	191
5.25.1.2	is_virtual	191
5.25.1.3	name	191
5.25.1.4	temperature	191
5.25.1.5	warn_level	191
5.26	XRpad_TempSensorReport Struct Reference	191
5.26.1	Member Data Documentation	191
5.26.1.1	sensor_count	191
5.26.1.2	sensors	191
5.26.1.3	shutdown_time	191
5.26.1.4	system_warn_level	192
5.27	XRpad_VersionInfo Struct Reference	192
5.27.1	Member Data Documentation	192
5.27.1.1	hwdriver	192
5.27.1.2	linux_kernel	192
5.27.1.3	misp_firmware	192
5.27.1.4	pld_firmware	192
5.27.1.5	software	192
5.27.1.6	spartan_firmware	192
5.27.1.7	subversion	192
5.27.1.8	wlan	192
5.27.1.9	xrpd	192
5.27.1.10	zynq_firmware	192
5.28	XrpdVariant Union Reference	192
5.28.1	Member Data Documentation	193
5.28.1.1	const_ptr	193
5.28.1.2	dbl	193
5.28.1.3	flt	193
5.28.1.4	ptr	193
5.28.1.5	u64	193

Chapter 1

Deprecated List

Member [Acquisition_GbIF_GetFilterDrvState](#) (HACQDESC hAcqDesc)

Cannot be used with current library versions.

Member [Acquisition_GetNetwork](#) (HACQDESC hAcqDesc, unsigned int *network)

Use [Acquisition_GetNetworkSpeed\(\)](#)

Member [Acquisition_Test_SDCardPerformance](#) (HACQDESC hAcqDesc, unsigned int buffer-size, double *wbitrate, unsigned int *wmicroseconds, double *rbitrate, unsigned int *rmicroseconds)

Function is considered to be deprecated!

Member [Acquisition_wpe_GetVersionNEW](#) (int *major, int *minor, int *release, int *build)

Please use [Acquisition_wpe_GetVersionEx\(\)](#)

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Acquire and correction functions	80
Callback function	109
Common Init functions	56
Common error handling	113
Init Functions for GbIF	68
Init Functions for XRpad	7
Setting up the Device	98
Special XRpad Functions	14
enumerations provided by Acq.h	129
functions provided by Acq.h	115

Chapter 3

Class Index

3.1 Class List

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

CHwHeaderInfo	159
CHwHeaderInfoEx	162
DETECTOR_BATTERY	166
DETECTOR_CURRENT_VOLTAGE	167
deviceInfo	168
discoveryReply	169
discoveryReplyEx	170
discoveryReplyMsg	171
EPC_REGISTER	171
FPGAType	173
GBIF_Detector_Properties	173
GBIF_DEVICE_PARAM	174
networkAdapterConfiguration	175
networkConfiguration	177
networkInfo	179
RTC_STRUCT	180
wifiConfiguration	180
wifiConfigurationEx	181
WinHeaderType	182
WinHeaderType101	185
WinImageHeaderType	187
XRpad_BatteryStatus	188
XRpad_ShockEvent	190
XRpad_ShockSensorReport	190
XRpad_TempSensor	191
XRpad_TempSensorReport	191
XRpad_VersionInfo	192
XrpdVariant	192

Chapter 4

Module Documentation

4.1 Init Functions for XRpad

Functions

- HIS_RETURN [Acquisition_GetVersionInfo](#) (HACQDESC hAcqDesc, [XRpad_VersionInfo](#) *version-Info)
Acquisition_GetVersionInfo Retrieves version information of the detector.
- HIS_RETURN [Acquisition_Set_OnboardOptions](#) (HACQDESC hAcqDesc, BOOL bEnableAck, BOOL bOffset, BOOL bGain, BOOL bPixel)
Activate image acquisition options for onboard offset, gain, pixel correction and acknowledgement of frames.
- HIS_RETURN [Acquisition_Set_OnboardOptionsDualEnergy](#) (HACQDESC hAcqDesc, BOOL bEnablePreviewFirst, BOOL bEnablePreviewSecond, BOOL bEnablePreviewOffset, BOOL bEnableOffset, BOOL bOnboardPxlCorr, BOOL bEnableAckFirst, BOOL bEnableFirst, BOOL bEnableAckSecond, BOOL bEnableSecond, BOOL bEnableAckThird, BOOL bEnableAckFourth, [OnboardBinningMode](#) ePreviewBinningMode)
This function can be used to verify the dual energy mode. It defines the image acquisition options for dual energy mode including preview on/off offset/pixel correction on/off and which images will be send to the client.
- 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.
- 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 PhotoTimed 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 *num-Devices, 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 [network-Configuration](#) *configs, int *arrayLength, int *activeConfig)
This function retrieves the Network settings of a specific Detector defined by its IP address.

4.1.1 Function Documentation

4.1.1.1 HIS_RETURN Acquisition_GetVersionInfo (HACQDESC hAcqDesc, XRpad_VersionInfo * versionInfo)

Acquisition_GetVersionInfo Retrieves version information of the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>versionInfo</i>	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.

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

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

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>bEnableAck</i>	Enables or disables the acknowledgment mechanism. Image will be stored on SD card for redundancy/safety when no acknowledgment is sent
<i>bOffset</i>	Enables or disables onboard offset correction
<i>bGain</i>	Enables or disables onboard gain correction
<i>bPixel</i>	Enables or disables onboard pixel correction

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 bEnableAck is set to TRUE Otherwise it will be stored on the SD card

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.1.1.3 HIS_RETURN Acquisition_Set_OnboardOptionsDualEnergy (HACQDESC hAcqDesc, BOOL bEnablePreviewFirst, BOOL bEnablePreviewSecond, BOOL bEnablePreviewOffset, BOOL bEnableOffset, BOOL bOnboardPxlCorr, BOOL bEnableAckFirst, BOOL bEnableFirst, BOOL bEnableAckSecond, BOOL bEnableSecond, BOOL bEnableAckThird, BOOL bEnableAckFourth, OnboardBinningMode ePreviewBinningMode)

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

In the dual energy mode the following four images will be acquired: # first bright image # second bright image # first dark image # second dark image

The on board offset correction (bEnableOffset) will perform the following for the third and fourth image: * (first bright image (uncorrected)) - (first dark image) * (second bright image (uncorrected)) - (second dark image)

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
in	<i>bEnable-PreviewFirst</i>	Enables preview image of first bright image
in	<i>bEnable-PreviewSecond</i>	Enables preview image of second bright image
in	<i>bEnable-PreviewOffset</i>	Enables on board offset correction for preview and full of first and second bright images
in	<i>bEnableOffset</i>	Enables on board offset correction for two final images
in	<i>bOnboardPxl-Corr</i>	Enables on board bad pixel correction for all images
in	<i>bEnableAck-First</i>	Enables image acknowledge mechanism for first bright image
in	<i>bEnableFirst</i>	Enables transmit of first bright image
in	<i>bEnableAck-Second</i>	Enables image acknowledge mechanism for second bright image
in	<i>bEnable-Second</i>	Enables transmit of second bright image
in	<i>bEnableAck-Third</i>	Enables image acknowledge mechanism for either first dark or offset corrected bright image
in	<i>bEnableAck-Fourth</i>	Enables image acknowledge mechanism for either second dark or offset corrected bright image
in	<i>ePreview-BinningMode</i>	Selection of on board binning mode for preview image. Just ONBOARD-BINNING2x2 and ONBOARDBINNING4x4 are allowed.

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure

Note

The clients image buffer defined by calling Acquisition_DefineDestBuffers must have the size to retrieve all selected images (max six full size for 2 Previews, 2 Brights and 2 Finals/Darks)

Just LSB of BOOL input parameters in considered by function.

Do not mix up dual enery mode specific "LoadCorrectionImageToBuffer()" and/or "Acquisition_Set_OnboardOptions()" functions with others mode related function(s).

4.1.1.4 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
	<i>bNoOnboard-Corr</i>	If this option is set then no onboard correction is done at all. That applies to not only the AED post offset correction but also the regular onboard correction, i.e. the flags bGain, bPixel and bEnableOffsetFirst are ignored. Further, no preview is generated even if it has been requested, i.e. bSendPreviewFirst will be ignored.
	<i>bSendPreview-Frist</i>	If this flag is set then the very first image that is being sent is a preview image. The preview will always be a 4x4 preview, independent of what has been set with Acquisition_Set_OnboardOptionPreview.
	<i>bSendFULL-First</i>	If this option is set then the full image is sent to the client computer. - A preview image, if requested, will always be sent before the full size image.
	<i>bEnableAck-First</i>	If this option is set, then the first image needs to be acknowledged by calling Acquisition_AcknowledgeImage. If the image is not being acknowledged, then the image is stored to the detector's SD card. If this parameter is set first full size image will be stored to SD card if not acknowledged.
	<i>bEnableAck-Second</i>	This is the same as bEnableAckFirst but for the second image. If this parameter is set second full size image will be stored to SD card if not acknowledged.
	<i>bEnableOffset-First</i>	If this option is set, then the preview image (if bSendPreviewFirst is set) and the first full image will be offset corrected with a "pre-offset" loaded into the correction buffer using Acq_wpe_LoadCorrectionImageToBuffer
	<i>bEnablePost-OffsetCorr</i>	If this option is set, then the first full size image will be offset corrected with the second image which is the "post-offset" image.
	<i>bGain</i>	If this option is set, then the preview image (if bSendPreviewFirst is set) and the full size images will be gain corrected with a gain image loaded into the correction buffer using Acq_wpe_LoadCorrectionImageToBuffer.

	<i>bPixel</i>	If this option is set, then the preview image (if <i>bSendPreviewFirst</i> is set) and the full size images will be corrected for bad pixels using the pixel correction image loaded into the correction buffer using <i>Acq_wpe_Load-CorrectionImageToBuffer</i> .
--	---------------	---

Returns

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

Note

- The clients image buffer defined using *Acquisition_DefineDestBuffers* must have the size to retrieve all selected images (max three full size for Preview, Bright, Offset/Brightoc)
- If *bEnableOffsetFirst* is enabled, then the image will be offset corrected with a pre-offset image. In this case, a post-offset image is being sent afterwards if *usMode* is set to 1 (see *Acquisition_SetAEDOptions*). If *usMode* is set to 0 then no post-offset image is sent.
- If *bEnableOffsetFirst* is enabled, then the image will be offset corrected with a pre-offset image. (E.g.: In this case, a post-offset image is being sent afterwards if *usMode* is set to 1 (see - *Acquisition_SetAEDOptions*). If *usMode* is set to 0 then no post-offset image is sent)
- If *bEnablePostOffsetCorr* is set then the image will be offset corrected with a post-offset image. In this case, the post-offset image will not be sent to the client.
- If "post-offset" is enabled when calling *Acquisition_SetAEDOptions* then a post-offset image is sent whenever the full image is NOT post-offset corrected. A full image is not post-offset corrected when either *bEnablePostOffsetCorr* is FALSE or when *bNoOnboardCorr* is TRUE, i.e. onboard corrections have been turned off. If the full image has been post-offset corrected or if acquiring a post-offset image has not been enabled then no post-offset image will be sent.

4.1.1.5 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 PhotoTimed 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

<i>in</i>	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
	<i>bNoOnboard-Corr</i>	If this parameter is set all onboard corrections and preview generation are disabled
	<i>bSendPreview-Frist</i>	If this parameter is set all a preview for the first image is generated and will be send to the client
	<i>bSendFULL-First</i>	If this parameter is set the first image will be send to the client in full size

	<i>bEnableAck-First</i>	If this parameter is set first full size image will be stored to SD card if not acknowledged
	<i>bEnableAck-Second</i>	If this parameter is set second full size image will be stored to SD card if not acknowledged
	<i>bEnableOffset-First</i>	If this parameter is set the first image and the preview image (if selected) will be offset corrected with a predefined offset image
	<i>bEnablePost-OffsetCorr</i>	If this parameter is set the first image will be offset corrected with the second image
	<i>bGain</i>	If this parameter is set the images image will be gain corrected with the predefined gain image
	<i>bPixel</i>	If this parameter is set the images image will be pixel corrected with the predefined pixel mask image
	<i>bStoreOffset-ToSD</i>	If this parameter is set second full size image will be stored to SD card (for debug/test only)

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure

Note

The clients image buffer defined by calling Acquisition_DefineDestBuffers must have the size to retrieve all selected images (max three full size for Preview, Bright, Offset/Brightoc)
This function is for test/debug only. If bStoreOffsetToSD is enabled the other Acknowledge/StoreToSD functions will be disabled

4.1.1.6 HIS_RETURN Acquisition_wpe_ActivateNetworkConfig (const char * ipAddress, int configIndex)

This function activates the selected network configuration of the detector.

Parameters

<i>ipAddress</i>	IP adress of the detectors network interface LAN/WLAN
<i>configIndex</i>	Index of the network configuration to activate

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure

4.1.1.7 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

<i>reply</i>	Array of discoveryReply structures
<i>numDevices</i>	Pointer to a value containing the number of allocated structures when called
<i>timeout</i>	The timeout shall be greater or equal 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.
<i>port</i>	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 keep 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. After the call it will contain the retrieved number of devices. An array length of 10 is recommended
- Timeout and port must be 0 to use the predefined default values

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure

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

This function retrieves the Network settings of a specific Detector defined by its IP address.

Parameters

<i>ipAddress</i>	IP address of the detectors LAN/WLAN interface.
<i>configs</i>	Array of networkConfiguration structures.
<i>arrayLength</i>	Number of allocated structures.
<i>activeConfig</i>	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 retrieved number of devices after the call. An array length of 20 is recommended
- activeConfig will contain the current activated configuration

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure

4.2 Special XRpad Functions

Functions

- HIS_RETURN [Acq_wpe_DualEnergy_LoadCorrectionImageToBuffer](#) (HACQDESC hAcqDesc, const char *pccCorrectionFilePath, const [ProcScriptOperation](#) Operation)
Load a correction file from the detector's SD card to a dual energy mode specific correction buffer.
- HIS_RETURN [Acq_wpe_LoadCorrectionImageToBuffer](#) (HACQDESC hAcqDesc, const char *pccCorrectionFilePath, [ProcScriptOperation](#) Operation)
Load a correction file from the detector's SD card to a specific correction buffer.
- HIS_RETURN [Acq_wpe_SetImageTransferInterface](#) (const char *ipAddress, [XRpad_DataInterface-
ControlEnum](#) 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)
Acknowledges the successful receipt of an image.
- HIS_RETURN [Acquisition_AckSDCardForceFsk](#) (HACQDESC hAcqDesc)
After the filesystem was checked per request and no errors were found, a message will be send to the client continuously. The reception of the message can be acknowledged/removed with this method. No further message will be send.
- HIS_RETURN [Acquisition_AckSDCardForceFskError](#) (HACQDESC hAcqDesc)
If errors there found in the filesystem and these errors were fixed, then there will be files like FSCK0000.-REC on the SD card and a message will be send to the client continuously. This method won't touch these files. Instead, 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)
Disables the event callback mechanism.
- HIS_RETURN [Acquisition_DisableSyslogSaving](#) (HACQDESC hAcqDesc)
Disables the rotating storage of the syslog files.
- HIS_RETURN [Acquisition_Enable_EMI_Data_Readout](#) (HACQDESC hAcqDesc, unsigned int uiOnOff)
Enables/disables the EMI data readout and transfer.
- HIS_RETURN [Acquisition_Enable_TestPattern](#) (HACQDESC hAcqDesc, unsigned int uiOnOff)
Enables/disables the test pattern on XRpad2 generated by the detector's FPGA.
- HIS_RETURN [Acquisition_FactoryResetShock](#) (HACQDESC hAcqDesc)
Resets all shock events.

- HIS_RETURN [Acquisition_FreeFTPFileBuffer](#) (void *pdatatbuffer)
This function releases a filebuffer allocated by Acquisition_GetFTPFile.
- HIS_RETURN [Acquisition_FTP_CloseSession](#) (XisIFtpSession session)
Acquisition_FTP_CloseSession closes an FTP session.
- HIS_RETURN [Acquisition_FTP_InitSession](#) (HACQDESC hAcqDesc, XisIFtpSession *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_GetAutoDeepSleepIdleLocations](#) (HACQDESC hAcqDesc, unsigned int *autoDeepSleepLocations, unsigned int *autoIdleLocations)
Retrieves the current locations from which the detector switches from idle to deep sleep and from deep sleep to idle when location changed.
- 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_GetDefaultBootConfiguration](#) (HACQDESC hAcqDesc, int *default_boot_cfg)
Function to check state of the functionality of a default boot configuration selection.
- HIS_RETURN [Acquisition_GetFTPFile](#) (const char *ipAddress, const char *filename, void **pdatatbuffer, long *filesize)
This function retrieves data from a file from the detector into a memory buffer.
- HIS_RETURN [Acquisition_GetGridSensorStatus](#) (HACQDESC hAcqDesc, unsigned int *ui-Status)
This function retrieves the status of the Grid Sensors.
- HIS_RETURN [Acquisition_GetLocation](#) (HACQDESC hAcqDesc, unsigned int *location)
Retrieves the location information of the detector.
- HIS_RETURN [Acquisition_GetMissedImageCount](#) (XisIFtpSession 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)
This function is deprecated and is only provided for compatibility. Use Acquisition_GetNetworkSpeed instead.
- HIS_RETURN [Acquisition_GetNetworkSpeed](#) (HACQDESC hAcqDesc, unsigned int *network)
Retrieves the network link speed of the detector. For a GigE detector this is the rated network speed, e.g. 1000 for a gigabit network.

- HIS_RETURN [Acquisition_GetPowerstate](#) (HACQDESC hAcqDesc, unsigned int *powerstate)
Retrieves the current powerstate information of the detector.
- HIS_RETURN [Acquisition_GetSDCardInfo](#) (HACQDESC hAcqDesc, unsigned int *total, unsigned int *avail)
Retrieves the SD card information 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_GetTemperature](#) (HACQDESC hAcqDesc, unsigned int(*current_value)[8])
Retrieves the temperature values strait from the xrpdc instead of EPC register.
- 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_GetWLAN_ChannelList](#) (HACQDESC hAcqDesc, char *wlan_channel_list_str, size_t wlan_channel_list_len)
Gets the actual WLAN channel list from the detector.
- HIS_RETURN [Acquisition_GetWLAN_CountryCode](#) (HACQDESC hAcqDesc, char *wlan_cc_str)
Retrieves the current WLAN country code from the detector.
- HIS_RETURN [Acquisition_IdentifyDevice](#) (HACQDESC hAcqDesc)
Trigger device to identify itself by flashing the LCD display for approximately 15 seconds.
- HIS_RETURN [Acquisition_IsPreviewImage](#) (HACQDESC hAcqDesc, unsigned int *uilsPreview)
This function retrieves the information whether the latest received frame is a preview image.
- unsigned int [Acquisition_IsSpartanIDLE](#) (PAcquisitionDesc pAcqDesc)
Returns, if XRpad2 detector is in deep sleep HIS_ERROR_XRPD_DETECTOR_IN_DEEP_SLEEP or in IDLE mode HIS_ALL_OK.
- HIS_RETURN [Acquisition_OpenMissedImage](#) (XisIFtpSession session, UINT index, XisFileHandle *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 resent.
- HIS_RETURN [Acquisition_Reset_OnboardOptions](#) (HACQDESC hAcqDesc)
Reset all onboard features like preview and corrections.
- HIS_RETURN [Acquisition_ResetOnboardShockEvent](#) (HACQDESC hAcqDesc, unsigned int latest-Shock_Timestamp)
Resets the onboard shookevent. 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 ui-Mode)
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 b-Enable, 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_SetAEDOptions](#) (HACQDESC hAcqDesc, unsigned short usMode, unsigned short usOffsetImageDelay, unsigned short usSelectReadoutSection)

This function configures detector Parameters for AED mode.
- HIS_RETURN [Acquisition_SetAutoDeepSleepIdleLocations](#) (HACQDESC hAcqDesc, unsigned int autoDeepSleepLocations, unsigned int autoIdleLocations)

Sets the locations in which the detector switches from idle to deep sleep and from deep sleep to idle when location changed.
- HIS_RETURN [Acquisition_SetAutoPowerOnLocations](#) (HACQDESC hAcqDesc, unsigned int autopoweronlocations)

Sets the locations from which the detector switches on automatically when the location changed.
- HIS_RETURN [Acquisition_SetChargeMode](#) (HACQDESC hAcqDesc, unsigned char charge_mode)

Sets the battery charge mode.
- HIS_RETURN [Acquisition_SetChargeModePAcqDesc](#) (PAcquisitionDesc pAcqDesc, unsigned char charge_mode)

Sets the battery charge mode.
- HIS_RETURN [Acquisition_SetDefaultBootConfiguration](#) (HACQDESC hAcqDesc, int default_boot_cfg)

Function to activate or deactivated the functionality of a default boot configuration selection.
- HIS_RETURN [Acquisition_SetDualEnergyParams](#) (HACQDESC hAcqDesc, unsigned short usNrOfScrubs, unsigned short usMaxDelay)

This function configures detector Parameters for dual energy 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 with the path /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 maximum 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.
- unsigned int [Acquisition_SetSpartanXISLTolInitState](#) (PAcquisitionDesc pAcqDesc)
set XISL and Spartan to InitState using current Header Information
- HIS_RETURN [Acquisition_SetSystemTime](#) (HACQDESC hAcqDesc, const char *cDateTime)
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_SetWLAN_CountryCode](#) (HACQDESC hAcqDesc, const char *wlan_cc_str)
Sets a new WLAN country code to the detector.
- HIS_RETURN [Acquisition_Test_SDCardPerformance](#) (HACQDESC hAcqDesc, unsigned int buffersize, double *wbitrate, unsigned int *wmicroseconds, double *rbitrate, unsigned int *rmicroseconds)
This function triggers the XRpad daemon on the detector to perform a SD card performance determination. The XRpad daemon creates a test file with data and file size of buffersize bytes. File writing and file reading is monitored. Monitoring results are available via wbitrate, wmicroseconds, rbitrate and rmicroseconds.
- 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 [networkConfiguration](#) *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 use temporary network settings (e.g. a temporary IP address)
- HIS_RETURN [Acquisition_wpe_GetExamFlag](#) (const char *ipAddress, unsigned long *pExamFlag)
This function retrieves the status of the exam flag which is set during a running acquisition.

- 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.
- HIS_RETURN [Acquisition_wpe_ReadCameraRegistersPAcqDesc](#) (PAcquisitionDesc pAcqDesc, unsigned int offset, unsigned int length, unsigned long *buffer)
Retrieves the epc from wpe dll.
- HIS_RETURN [Acquisition_wpe_SetMaxOnboardCorrValue](#) (HACQDESC hAcqDesc, unsigned short usMax, unsigned short usReplace)
Set maximum value for onboard corrections. If the value of a corrected pixel exceeds usMax then it will be replaced with the value usReplace.
- HIS_RETURN [Acquisition_wpe_SetUniqueImageTag](#) (HACQDESC hAcqDesc, const char *image-Tag)
Sets an Image Tag via wpe200 library.
- HIS_RETURN [Acquisition_xrpd_GetDetectorType](#) (HACQDESC hAcqDesc, char *response_buffer, size_t response_buffer_len)
Reads the detector type from the current connected detector.
- HIS_RETURN [Acquisition_xrpd_GetExamFlag](#) (HACQDESC hAcqDesc, unsigned long *pExam-Flag)
This function retrieved the status of the exam flag which is set during a running acquisition.
- HIS_RETURN [Acquisition_xrpd_ReadCameraRegisters](#) (HACQDESC hAcqDesc, unsigned long *buffer)
This function retrieves the Status register of a specific Detector defined by its ip address.
- HIS_RETURN [Acquisition_xrpd_ReadCameraRegistersHACQDESC](#) (HACQDESC hAcqDesc, unsigned int offset, unsigned int length, unsigned long *buffer)
Retrieves the epc from xrpd instead wpe call.
- HIS_RETURN [Acquisition_xrpd_ReadCameraRegistersPAcqDesc](#) (PAcquisitionDesc pAcqDesc, unsigned int offset, unsigned int length, unsigned long *buffer)
Retrieves the epc via xrpd.
- HIS_RETURN [Acquisition_xrpd_WriteCameraRegistersHACQDESC](#) (HACQDESC hAcqDesc, unsigned int offset, unsigned int length, unsigned int *buffer)
Retrieves the epc from xrpd instead wpe call.

4.2.1 Function Documentation

4.2.1.1 HIS_RETURN Acq_wpe_DualEnergy_LoadCorrectionImageToBuffer (HACQDESC hAcqDesc, const char * pccCorrectionFilePath, const ProcScriptOperation Operation)

Load a correction file from the detector's SD card to a dual energy mode specific correction buffer.

Parameters

<i>hAcqDesc</i>	Handle to the detector
<i>pccCorrection-FilePath</i>	Full path of the correction file on the SDCARD including /mnt/sdcard
<i>Operation</i>	ProcScriptOperation OFFSET, OFFSET_2 or MEAN (pixel correction)

Note

The correction will not be enabled with this function.
Do not mix up dual enery mode specific "LoadCorrectionImageToBuffer()" and/or "Acquisition_Set - OnboardOptions()" functions with others mode related function(s).

Returns

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

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

Load a correction file from the detector's SD card to a specific correction buffer.

Parameters

<i>hAcqDesc</i>	Handle to the detector
<i>pccCorrection-FilePath</i>	Full path of the correction file on the SDCARD including /mnt/sdcard
<i>Operation</i>	see enum ProcScriptOperation

Note

The correction will not be enabled with this function.

Returns

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

4.2.1.3 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

<i>ipAddress</i>	IP-Address of the device to control
<i>eDataInterface</i>	Interface to use (0 - LAN 1 - WLAN)

Note

This function may return -10004 if the device is not reachable in case of a 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.

4.2.1.4 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

<i>ipAddress</i>	IP address of the device to control.
<i>eAction</i>	Action to execute.

Note

- This function may return -10004 if the device is not reachable in case of a shutdown/reboot
- For the action WPE_SYSTEM_CONTROL_REBOOT first call Acquisition_Close or Acquisition_CloseAll. Afterwards the detector has to be initialized again.
- When the actions XRpad_SYSTEM_CONTROL_SET_DEEP_SLEEP / XRpad_SYSTEM_CONTROL_SET_IDLE are used to switch to deep sleep and to switch to idle is used it has to be checked whether the device is back in IDLE stated by using Acquisition_GetHWHeader. Also the image tag must be a non empty string. Instead of these actions, it is recommended to use Acquisition_Set_FPGA_Power_Mode to switch between deep sleep and 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.

4.2.1.5 HIS_RETURN Acquisition_AcknowledgeImage (HACQDESC *hAcqDesc*, const char * *tag*)

Acknowledges the successful receipt of an image.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>tag</i>	The tag of the image to be acknowledged.

Returns

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

4.2.1.6 HIS_RETURN Acquisition_AckSDCardForceFsck (HACQDESC *hAcqDesc*)

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

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Note

If an error was found the error message will be send continuously. The message can be acknowledged using Acquisition_AckSDCardForceFsckError.

Returns

Returns HIS_ALL_OK on success of acknowledging a performed file system check without errors on detector. Returns HIS_ERROR_VXD_REGISTER_IRQ on success of performing XISL call without having a performed file system check on detector. Or an appropriate error code otherwise.

4.2.1.7 HIS_RETURN Acquisition_AckSDCardForceFsckError (HACQDESC *hAcqDesc*)

If errors there found in the filesystem and these errors were fixed, then there will be files like FSCK0000.-REC on the SD card and a message will be send to the client continuously. This method won't touch these files. Instead, 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success of acknowledging a performed file system check with found FS errors on detector. Returns HIS_ERROR_VXD_REGISTER_IRQ on success of performing XISL call without having a performed file system check with found FS errors on detector. Or an appropriate error code otherwise.

4.2.1.8 HIS_RETURN Acquisition_CloseFile (XislFileHandle *fileHandle*)

Acquisition_CloseFile closes a file and releases allocated memory.

Parameters

in	<i>fileHandle</i>	A valid file handle.
----	-------------------	----------------------

Returns

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

Warning

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

4.2.1.9 HIS_RETURN Acquisition_CreateFakeShockCriticalLevel (HACQDESC hAcqDesc)

Simulates a shock event at critical level.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.10 HIS_RETURN Acquisition_CreateFakeShockWarningLevel (HACQDESC hAcqDesc)

Simulates a shock event at warning level.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.11 HIS_RETURN Acquisition_DisableEventCallback (HACQDESC hAcqDesc)

Disables the event callback mechanism.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.12 HIS_RETURN Acquisition_DisableSyslogSaving (HACQDESC *hAcqDesc*)

Disables the rotating storage of the syslog files.

If this function is not called, the detector will compress the syslog files using lzop and save them in a tarball on the SD card. To prevent the SD card from running out of memory, only the tarballs of the last 10 boots are stored. If you call this function after Acquisition_GbIF_Init, the detector will NOT save the syslog AND it will remove all existing logs from the SD card.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor. This setting is not persistent. You have to call it every time after init.
----	-----------------	--

Attention

THIS COMMAND WILL NOT ONLY DISABLE THE SAVING OF THE SYSLOG, BUT ALSO REMOVE ALL EXISTING SYSLOGS FROM THE SD CARD.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

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

Enables/disables the EMI data readout and transfer.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>uiOnOff</i>	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.

4.2.1.14 HIS_RETURN Acquisition_Enable_TestPattern (HACQDESC *hAcqDesc*, unsigned int *uiOnOff*)

Enables/disables the test pattern on XRpad2 generated by the detector's FPGA.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>uiOnOff</i>	Defines whether the generator shall be enabled 1 or disabled 0

Returns

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

4.2.1.15 HIS_RETURN Acquisition_FactoryResetShock (HACQDESC *hAcqDesc*)

Resets all shock events.

Parameters

<i>in</i>	<i>hAcqDesc</i>	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

4.2.1.16 HIS_RETURN Acquisition_FreeFTPFileBuffer (void * *pdatabuffer*)

This function releases a filebuffer allocated by Acquisition_GetFTPFile.

Parameters

<i>pdatabuffer</i>	Pointer to buffer that should be freed up.
--------------------	--

Note

Customer/client application has to set the pointer to NULL.

Returns

Always HIS_ALL_OK

4.2.1.17 HIS_RETURN Acquisition_FTP_CloseSession (XisIFtpSession *session*)

Acquisition_FTP_CloseSession closes an FTP session.

Parameters

<i>in</i>	<i>session</i>	A valid FTP session descriptor.
-----------	----------------	---------------------------------

Returns

Always HIS_ALL_OK.

4.2.1.18 HIS_RETURN Acquisition_FTP_InitSession (HACQDESC *hAcqDesc*, XisIFtpSession * *session*)

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

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>session</i>	If the function succeeds, this parameter contains a descriptor of the current FTP session.

Returns

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

4.2.1.19 HIS_RETURN Acquisition_Get_Current_Voltage (HACQDESC *hAcqDesc*, DETECTOR_CURRENT_VOLTAGE * *pstructCurrentVoltage*)

Retrieves the internal currents and voltages from the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>pstructCurrent-Voltage</i>	Will retrieve internal detector voltages and currents

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.20 HIS_RETURN Acquisition_GetAutoDeepSleepIdleLocations (HACQDESC *hAcqDesc*, unsigned int * *autoDeepSleepLocations*, unsigned int * *autoldleLocations*)

Retrieves the current locations from which the detector switches from idle to deep sleep and from deep sleep to idle when location changed.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>autoDeep-SleepLocations</i>	Will retrieve the bitwise coded locations in which the detector switches to deep sleep when location change is detected
out	<i>autoldle-Locations</i>	Will retrieve the bitwise coded locations in which the detector switches to idle state when location change is detected

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Go to idle is prioritized in case confliction settings in the bitmasks

4.2.1.21 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>autopoweronlocations</i>	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.

4.2.1.22 HIS_RETURN Acquisition_GetBatteryStatus (HACQDESC *hAcqDesc*, XRpad_BatteryStatus * *batteryStatus*)

Retrieves the battery status.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition descriptor.
out	<i>batteryStatus</i>	Pointer to an XRpad_BatteryStatus structure to retrieve the battery status.

Returns

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

4.2.1.23 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 maximum charge mode The *charge_mode_charger* defines the maximum charge allowed by the charger.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>charge_mode_req</i>	The value of charge mode requested by the software, 0..3 0 no charging, 3 maximum charging
out	<i>charge_mode_charger</i>	The value of charge mode set by the charger, 0..3 0 no charging, 3 maximum charging

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.24 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	<i>pAcqDesc</i>	pointer to a valid Acquisition Descriptor.
out	<i>charge_mode_req</i>	The value of charge mode requested by the software, 0..3 0 no charging, 3 max charging
out	<i>charge_mode_charger</i>	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.

4.2.1.25 HIS_RETURN Acquisition_GetDefaultBootConfiguration (HACQDESC *hAcqDesc*, int * *default_boot_cfg*)

Function to check state of the functionality of a default boot configuration selection.

Parameters

in	<i>hAcqDesc</i>	an acquisition descr structure
out	<i>default_boot_cfg</i>	*default_boot_cfg < 0 -- functionality is not active *default_boot_cfg == 0 -- value is not returned. 0 < *default_boot_cfg < 16 -- number/identifier of selected configuration

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

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

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

Parameters

<i>ipAddress</i>	Pointer to char array of IP address.
<i>filename</i>	Pointer to value to retrieve the actual field of view mode.
<i>databuffer</i>	Pointer to databuffer.
<i>filesize</i>	Pointer to long value of retrieved filesize.

Note

Default directory is /mnt/sdcard/

Returns

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

4.2.1.27 HIS_RETURN Acquisition_GetGridSensorStatus (HACQDESC *hAcqDesc*, unsigned int * *uiStatus*)

This function retrieves the status of the Grid Sensors.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>uiStatus</i>	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

It is possible to distinguish between four grids via the two sensors (combinatorics). E.g.:

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

4.2.1.28 HIS_RETURN Acquisition_GetLocation (HACQDESC *hAcqDesc*, unsigned int * *location*)

Retrieves the location information of the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>location</i>	Will retrieve location of detector.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.29 HIS_RETURN Acquisition_GetMissedImageCount (XisIFtpSession *session*, UINT * *count*)

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

Parameters

in	<i>session</i>	A valid FTP session descriptor.
out	<i>count</i>	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 on failure.

4.2.1.30 HIS_RETURN Acquisition_GetNetwork (HACQDESC *hAcqDesc*, unsigned int * *network*)

This function is deprecated and is only provided for compatibility. Use Acquisition_GetNetworkSpeed instead.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>network</i>	Will retrieve network link speed of detector.

Deprecated Use [Acquisition_GetNetworkSpeed\(\)](#)

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.31 HIS_RETURN Acquisition_GetNetworkSpeed (HACQDESC *hAcqDesc*, unsigned int * *network*)

Retrieves the network link speed of the detector. For a GigE detector this is the rated network speed, e.g. 1000 for a gigabit network.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>network</i>	Will retrieve network link speed of detector.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.32 HIS_RETURN Acquisition_GetPowerstate (HACQDESC *hAcqDesc*, unsigned int * *powerstate*)

Retrieves the current powerstate information of the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>powerstate</i>	Will retrieve powerstate of the detector

Note

powerstate will be one of the values defined by XRPAD_SystemControlEnum.

Returns

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

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

Retrieves the SD card information about available storage space on the SD card from the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>total</i>	Will retrieve total storage in MB.
out	<i>avail</i>	Will retrieve available storage in MB.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.34 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>sdcard_timeout</i>	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.

4.2.1.35 HIS_RETURN Acquisition_GetTemperature (HACQDESC *hAcqDesc*, unsigned int(*) *current_value[8]*)

Retrieves the temperature values strait from the xrpd instead of EPC register.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>current_value</i>	Pointer to the variable to retrieve the temperature values in millidegree Celsius.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.36 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>threshold_warning</i>	Pointer to the variable to retrieve the warning level temperature in millidegree Celsius.
out	<i>threshold_critical</i>	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.

4.2.1.37 HIS_RETURN Acquisition_GetWLAN_ChannelList (HACQDESC *hAcqDesc*, char * *wlan_channel_list_str*, size_t *wlan_channel_list_len*)

Gets the actual WLAN channel list from the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>wlan_channel_list_str</i>	Gets the WLAN channel list string. The string is a JSON string. The string buffer needs to be provided by the caller.
in	<i>wlan_channel_list_len</i>	The buffer size of <i>wlan_channel_list_str</i> .

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.38 HIS_RETURN Acquisition_GetWLAN_CountryCode (HACQDESC *hAcqDesc*, char * *wlan_cc_str*)

Retrieves the current WLAN country code from the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>wlan_cc_str</i>	Retrieves the 2 letter WLAN country code from the detector. The string will be 0 terminated. It is recommended that <i>wlan_cc_str</i> is at least 4 characters long to accommodate possible 3 letter codes plus the string termination character.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.39 HIS_RETURN Acquisition_IdentifyDevice (HACQDESC *hAcqDesc*)

Trigger device to identify itself by flashing the LCD display for approximately 15 seconds.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.40 HIS_RETURN Acquisition_IsPreviewImage (HACQDESC *hAcqDesc*, unsigned int * *uilsPreview*)

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

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>uilsPreview</i>	Flag whether the image 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 is retrieved.

4.2.1.41 unsigned int Acquisition_IsSpartanIDLE (PAcquisitionDesc *pAcqDesc*)

Returns, if XRpad2 detector is in deep sleep HIS_ERROR_XRPD_DETECTOR_IN_DEEP_SLEEP or in IDLE mode HIS_ALL_OK.

Parameters

in	<i>pAcqDesc</i>	pointer to a valid Acquisition Descriptor structure.
----	-----------------	--

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.42 HIS_RETURN Acquisition_OpenMissedImage (XisIFtpSession *session*, UINT *index*, XisIFileHandle * *fileHandle*)

Acquisition_FTP_OpenMissedImage retrieves a handle of a missed image file.

Parameters

in	<i>session</i>	A valid FTP session descriptor.
in	<i>index</i>	Index between 0 and (count - 1), where count is retrieved by Acquisition_FTP_GetMissedImageCount.
out	<i>fileHandle</i>	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 on failure.

Warning

This function allocates a memory buffer to carry the data of the file. Call Acquisition_CloseFile to release this memory to avoid memory leaks.

4.2.1.43 HIS_RETURN Acquisition_Resend_All_Messages (HACQDESC hAcqDesc)

Resets all messages to be resent.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.44 HIS_RETURN Acquisition_Reset_OnboardOptions (HACQDESC hAcqDesc)

Reset all onboard features like preview and corrections.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.45 HIS_RETURN Acquisition_ResetOnboardShockEvent (HACQDESC hAcqDesc, unsigned int latestShock_Timestamp)

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

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
	<i>latestShock_Timestamp</i>	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.

4.2.1.46 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.47 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>uiMode</i>	0 - Off or 1 - On

Returns

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

Note

Currently only valid for network connected detectors

4.2.1.48 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

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

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.49 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

<i>hAcqDesc</i>	Handle of a valid acquisition descriptor.
<i>bEnablePreview</i>	Enables or disables the preview option. If enabled, an acquisition will first send a binned preview image before sending the image in full resolution.
<i>bPreviewOptionSendFull</i>	Enables or disables sending the full size image.
<i>eMode</i>	Selected binning mode for preview
<i>uiSelectedScript</i>	Must be 0 in current implementation

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

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

4.2.1.50 HIS_RETURN Acquisition_SetAEDOptions (HACQDESC *hAcqDesc*, unsigned short *usMode*, unsigned short *usOffsetImageDelay*, unsigned short *usSelectReadoutSection*)

This function configures detector Parameters for AED mode.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>usMode</i>	Defines which AED mode shall be used (0-single AED image 1-AED image followed by an offset image after a defined delay time)

in	<i>usOffsetImage-Delay</i>	Reserved! Set always to 0 to use detector default setting! Defines the maximum delay time between AED image and the Offset image. Valid Range (500-65535).
in	<i>usSelect-Readout-Section</i>	Defines which readout section shall be used for the AED trigger generation (0 - full area, 1 - half odd groups, 2 - half even groups, 3 - center region, 4 - all off).

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Currently only available for XRpad2.

- If usMode is 0 then no post offset will be acquired. This is the way AED worked before (and still does using the same function calls). If usMode is 1 then a post offset image will be acquired.
- For usMode 1, an additional acquisition buffer has to be provided.
- Groups are zero indexed, hence the first group is an even group.

4.2.1.51 HIS_RETURN Acquisition_SetAutoDeepSleepIdleLocations (HACQDESC hAcqDesc, unsigned int autoDeepSleepLocations, unsigned int autoidleLocations)

Sets the locations in which the detector switches from idle to deep sleep and from deep sleep to idle when location changed.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>autoDeep-SleepLocations</i>	Will set the bitwise coded locations in which the detector switches to deep sleep when location change is detected. Set to 256 to disable
in	<i>autoidle-Locations</i>	Will set the bitwise coded locations in which the detector switches to idle state when location change is detected. Set to 256 to disable

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Go to idle is prioritized in case confliction settings in the bitmasks.

4.2.1.52 HIS_RETURN Acquisition_SetAutoPowerOnLocations (HACQDESC hAcqDesc, unsigned int autopoweronlocations)

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

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>autopoweronlocations</i>	Bitmask to define the locations.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Locations are ORed, hence multiple locations can be defined. location0 - 1, location1 - 2, location3 - 4, .. , location7 - 128; All locations 255

4.2.1.53 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>charge_mode</i>	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.

4.2.1.54 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	<i>pAcqDesc</i>	Pointer to a valid acquisition descriptor.
in	<i>charge_mode</i>	The value of charge mode, 0..3 0 no charging, 3 maximum charging.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.55 HIS_RETURN Acquisition_SetDefaultBootConfiguration (HACQDESC *hAcqDesc*, int *default_boot_cfg*)

Function to activate or deactivated the functionality of a default boot configuration selection.

Parameters

in	<i>hAcqDesc</i>	an acquisition descr structure
in	<i>default_boot_cfg</i>	default_boot_cfg < 0 -- deactivates the functionality default_boot_cfg == 0 -- value is ignored. The function returns immediately without error code. 0 < default_boot_cfg < 16 -- number/identifier of selected configuration

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.56 HIS_RETURN Acquisition_SetDualEnergyParams (HACQDESC *hAcqDesc*, unsigned short *usNrOfScrubs*, unsigned short *usMaxDelay*)

This function configures detector Parameters for dual energy mode.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>usNrOfScrubs</i>	After bright images readout before offset readouts (default is 4 min 1)
in	<i>usMaxDelay</i>	Max Exposure Delay in milliseconds (default is 5000)

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Currently only available for XRpad2

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

Sets a callback function to retrieve events.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>EventCallback</i>	Pointer to the callback function which retrieves the events.
	<i>userData</i>	Pointer to the user data

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

The user data will be passed into the callback function when it is being called. This allows users to pass data into the callback function instead of having to declare that data globally.

4.2.1.58 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>bEnableFakeMode</i>	TRUE enables the fake mode, FALSE disables it.
in	<i>iFakeTemperature</i>	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.

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

This function stores a databuffer on the SD card with the path /mnt/sdcard.

Parameters

in	<i>ipAddress</i>	Pointer to char array containing the IP address of the detector.
in	<i>filename</i>	Pointer to the filename under which the data buffer shall be stored on the SD card. The main path /mnt/sdcard should not be included.
in	<i>databuffer</i>	Pointer to databuffer.
in	<i>filesize</i>	Pointer to long value containing the size of databuffer.

Returns

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

Note

In order to store a valid HIS file, such a file first needs to be created in memory. See the function `Acquisition_CreateXISFileInMemory`.

4.2.1.60 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 permanently stored!

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>timeout</i>	The value of the idle timeout period in seconds. The timeout value must be at least 20 seconds. The maximum value is 65534 sec as a valid time. 65535 will disable the automatic shutdown.

Returns

Returns `HIS_ALL_OK` on success or an appropriate error code otherwise.

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

Sets the maximum speed of the network of the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>network</i>	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.

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

This function configures detector Parameters for Phototimed mode.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>usNrOfScrubs</i>	After bright image readout before offset readout (default is 4 min 1)
in	<i>usMaxDelay</i>	Max Exposure Delay in milliseconds (default is 5000)

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Note

Currently only available for XRpad2

4.2.1.63 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 ($^{\wedge}$ = 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>key_old</i>	Pointer to an array containing the old key in raw binary data.
in	<i>key_new</i>	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 for XRpad2

4.2.1.64 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

<i>in</i>	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
-----------	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.65 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 by calling functions like Acquisition_Set_OnboardOptions and Acquisition_Set_OnboardOptionsPostOffset. When the image is stored, the client is notified by an event as far as 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

Parameters

<i>in</i>	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
<i>in</i>	<i>sdcard_timeout</i>	SD card timeout value in seconds.

Note

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

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.66 unsigned int Acquisition_SetSpartanXISLTolInitState (PAcquisitionDesc *pAcqDesc*)

set XISL and Spartan to InitState using current Header Information

Parameters

in	<i>pAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.67 HIS_RETURN Acquisition_SetSystemTime (HACQDESC hAcqDesc, const char * cDateTime)

Sets the system time of the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>cDateTime</i>	Pointer to a datetime string. If this pointer is NULL, the UTC date and time is computed from the host's local system time.

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 on failure.

4.2.1.68 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>usTailTime</i>	Tail time to set in milliseconds.
in	<i>eTriggerMode</i>	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 default for DDD, AED, Phototimed is 20msec. Only available for XRpad detectors.

4.2.1.69 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>threshold_warning</i>	Warning temperature in millidegree Celsius. If the surface temperature is above this temperature, the API will trigger a warning event.
in	<i>threshold_critical</i>	Critical temperature in millidegree Celsius. If the surface temperature is above this temperature, the detector will shut down after a predefined timeout.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.70 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>timeout</i>	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.

4.2.1.71 HIS_RETURN Acquisition_SetWLAN_CountryCode (HACQDESC *hAcqDesc*, const char * *wlan_cc_str*)

Sets a new WLAN country code to the detector.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>wlan_cc_str</i>	NULL terminated, 2 letter WLAN country code from the detector. It is recommended that <i>wlan_cc_str</i> is at least 4 characters long to accommodate possible 3 letter codes plus string termination.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.72 HIS_RETURN Acquisition_Test_SDCardPerformance (HACQDESC *hAcqDesc*, unsigned int *bufferize*, double * *wbitrate*, unsigned int * *wmicroseconds*, double * *rbitrate*, unsigned int * *rmicroseconds*)

This function triggers the XRpad daemon on the detector to perform a SD card performance determination. The XRpad daemon creates a test file with data and file size of *bufferize* bytes. File writing and file reading is monitored. Monitoring results are available via *wbitrate*, *wmicroseconds*, *rbitrate* and *rmicroseconds*.

Note

rbitrate and *rmicroseconds* are for internal use only.

Deprecated Function is considered to be deprecated!

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
<i>bufferize</i>	test file size in bytes.
<i>wbitrate</i>	determined bitrate for writing
<i>wmicroseconds</i>	elapsed time while writing test file in micro seconds
<i>rbitrate</i>	determined bitrate for reading
<i>rmicroseconds</i>	elapsed time while reading test file in micro seconds

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.73 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	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
out	<i>msg</i>	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	<i>msg_len</i>	The message length.
out	<i>md</i>	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.

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

Change the network configuration via API call.

Parameters

<i>ipAddress</i>	IP address of the detector
<i>configIndex</i>	Configuration Index of network configuration setting to be changed.
<i>config</i>	networkConfiguration structure to store.

Note

- config ([networkConfiguration](#)) has some readonly members. The function call will fail if those values are modified.
- It is best practice to first read the configuration using Acquisition_wpe_GetNetworkConfigs, modifying the desired values and then use this function to update them. Alternatively, [Acquisition_wpe_FillDefaultNetworkConfiguration\(\)](#) can be called to fill a structure with default values.
- Using this function will only update the settings but not make them active. To make a network configuration active, call Acquisition_wpe_ActivateNetworkConfig.
- Factory settings cannot be overwritten!

Attention

Limitations exist when WLAN is disabled on the detector:

- Provided that following conditions are met:
 1. WLAN is disabled on the detector
 2. in the new configuration to use
 - WLAN mode is Access Point (AP)
 - channel is not 0 (auto) **OR** agmode is not HT20
- Then the call of this function will not be successful without one of the following workarounds:

1. in the provided configuration, set the WLAN mode to client
 2. in the provided configuration, enable the automatic channel selection in HT20 mode
 - (a) set the WLAN channel number to 0 (auto channel selection)
 - (b) set the WLAN agmode to HT20
 3. temporary enable WLAN with automatic channel selection in HT20 mode, afterwards call this function with the original desired settings
 - (a) activate a network configuration with following modified parameters
 - i. set the WLAN enabled flag to 1
 - ii. set the WLAN channel number to 0 (auto channel selection)
 - iii. set the WLAN agmode to HT20
 - (b) then call this function again with the original desired values
- Programming example:
 - in the XISL demonstration application, see the demo of network change limitations and workarounds when WLAN is disabled
 - Root cause of limitation:
 - when calling this function, plausibility checks are done against the provided parameters
 - the validity of the provided WLAN channel number is checked by verifying if the WLAN driver supports it in the current WLAN mode
 - if WLAN is disabled on the detector, there is no WLAN driver loaded.
Due to this, the validity of the provided channel number cannot be checked
 - This is interpreted as a failure, and an error code is returned

Returns

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

4.2.1.75 HIS_RETURN Acquisition_wpe_FillDefaultNetworkConfiguration (struct networkConfiguration * config)

This function will fill a structure with default network settings.

Parameters

<i>config</i>	Pointer to the "struct networkConfiguration" to use.
---------------	--

Returns

Returns HIS_ALL_OK on success or an appropriate error code on failure

4.2.1.76 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 use temporary network settings (e.g. a temporary IP address)

Parameters

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

Note

- config ([networkConfiguration](#)) has some readonly members, please check reference for that.

Attention

Limitations exist when WLAN is disabled on the detector:

- Provided that following conditions are met:
 1. WLAN is disabled on the detector
 2. in the new configuration to use
 - WLAN mode is Access Point (AP)
 - channel is not 0 (auto) **OR** agmode is not HT20
- Then the call of this function will not be successful without one of the following workarounds:
 1. in the provided configuration, set the WLAN mode to client
 2. in the provided configuration, enable the automatic channel selection in HT20 mode
 - (a) set the WLAN channel number to 0 (auto channel selection)
 - (b) set the WLAN agmode to HT20
 3. temporary enable WLAN with automatic channel selection in HT20 mode, afterwards call this function with the original desired settings
 - (a) activate a network configuration with following modified parameters
 - i. set the WLAN enabled flag to 1
 - ii. set the WLAN channel number to 0 (auto channel selection)
 - iii. set the WLAN agmode to HT20
 - (b) then call this function again with the original desired values
- Programming example:
 - in the XISL demonstration application, see the demo of network change limitations and workarounds when WLAN is disabled

- Root cause of limitation:
 - when calling this function, plausibility checks are done against the provided parameters
 - the validity of the provided WLAN channel number is checked by verifying if the WLAN driver supports it in the current WLAN mode
 - if WLAN is disabled on the detector, there is no WLAN driver loaded.
Due to this, the validity of the provided channel number cannot be checked
 - This is interpreted as a failure, and an error code is returned

Returns

Returns HIS_ALL_OK on success or a negative error code on failure.

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

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

Parameters

<i>ipAddress</i>	IP-Address of the device to control
<i>pExamFlag</i>	Pointer to an unsigned long value to retrieve the exam flag status (1 - exam ongoing; any other - no exam ongoing)

Returns

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

4.2.1.78 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

<i>ipAddress</i>	IP address of the detector LAN/WLAN
<i>buffer</i>	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 the status register as struct [EPC_REGISTER](#)

Returns

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

4.2.1.79 HIS_RETURN Acquisition_wpe_ReadCameraRegistersPAcqDesc (PAcquisitionDesc pAcqDesc, unsigned int offset, unsigned int length, unsigned long * buffer)

Retrieves the epc from wpe dll.

Parameters

<i>pAcqDesc</i>	Handle of a valid Acquisition descriptor.
<i>offset</i>	Start address in epc
<i>length</i>	How many data to read
<i>buffer</i>	EPC data as pointer

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.80 HIS_RETURN Acquisition_wpe_SetMaxOnboardCorrValue (HACQDESC hAcqDesc, unsigned short usMax, unsigned short usReplace)

Set maximum value for onboard corrections. If the value of a corrected pixel exceeds usMax then it will be replaced with the value usReplace.

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
in	<i>usMax</i>	Maximum pixel value which is evaluated to be corrected before the correction is applied.
in	<i>usReplace</i>	Used to replace values > usMax.

Note

If 0x0000h was set, 0xFFFFh will be used instead for the both input arguments.

Returns

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

4.2.1.81 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 acknowledgement mechanism, available via Acquisition_Set_OnboardOptions and Acquisition_Set_OnboardOptions-PostOffset. When an image is that needs to be acknowledges but is not acknowledged, then it will be stored with the name given by the image tag followed by an underscore and a random number.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>imageTag</i>	Image tag to be used as filename for autosave when not acknowledged and to acknowledge the image (max characters 118)

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

Warning

Always use a unique identifier as image tag. If the image tag is ambiguous OR is a substring of a tag previously used, then all the images with this same or similar image tag will be acknowledged and deleted from the SD card when an acknowledgment is received.

4.2.1.82 HIS_RETURN Acquisition_xrpd_GetDetectorType (HACQDESC *hAcqDesc*, char * *response_buffer*, size_t *response_buffer_len*)

Reads the detector type from the current connected detector.

Parameters

in	<i>hAcqDesc</i>	- Handle of a valid Acquisition Descriptor
out	<i>response_buffer</i>	- buffer to which the response will be written. Must not be NULL.
in	<i>response_buffer_len</i>	- buffer size of response_buffer. Must be big enough to store all of the retrieved data (recommended minimum value: 128)

Returns

HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.83 HIS_RETURN Acquisition_xrpd_GetExamFlag (HACQDESC *hAcqDesc*, unsigned long * *pExamFlag*)

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

Parameters

<i>hAcqDesc</i>	handle to an existing connection
<i>pExamFlag</i>	pointer to an unsigned long value to retrieve the exam flag status (1 - exam ongoing; any other - no exam ongoing)

Returns

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

4.2.1.84 HIS_RETURN Acquisition_xrpd_ReadCameraRegisters (HACQDESC *hAcqDesc*, unsigned long * *buffer*)

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

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
<i>buffer</i>	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

4.2.1.85 HIS_RETURN Acquisition_xrpd_ReadCameraRegistersHACQDESC (HACQDESC *hAcqDesc*, unsigned int *offset*, unsigned int *length*, unsigned long * *buffer*)

Retrieves the epc from xrpd instead wpe call.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition descriptor.
<i>offset</i>	Start address in epc
<i>length</i>	How many data to read
<i>buffer</i>	EPC data as pointer

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.86 HIS_RETURN Acquisition_xrpd_ReadCameraRegistersPACqDesc (PAcquisitionDesc *pAcqDesc*, unsigned int *offset*, unsigned int *length*, unsigned long * *buffer*)

Retrieves the epc via xrpd.

Parameters

<i>pAcqDesc</i>	Pointer to Acquisition Descriptor.
<i>offset</i>	Start address in epc
<i>length</i>	How many data to read
<i>buffer</i>	EPC data as pointer

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise.

4.2.1.87 HIS_RETURN Acquisition_xrpd_WriteCameraRegistersHACQDESC (HACQDESC *hAcqDesc*, unsigned int *offset*, unsigned int *length*, unsigned int * *buffer*)

Retrieves the epc from xrpd instead wpe call.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition descriptor.
<i>offset</i>	Start address in epc.
<i>length</i>	How many data to read.
<i>buffer</i>	EPC data as pointer.

Returns

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

4.3 Common Init functions

Functions

- unsigned int [Acquisition_CallFPGASetCameraModelInternal](#) (PAcquisitionDesc pAcqDesc, unsigned char *CommandString, long lLength)
Wrapper for eltec set camera mode.
- 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 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_GetNextSensor. For a programming example see the initialization part of the XISL demonstration. Note: In case of Network/IP sensors, only sensors with standard-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 *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.
- HIS_RETURN [Acquisition_GetHwHeaderInfo](#) (HACQDESC hAcqDesc, CHwHeaderInfo *pInfo)
This function returns the contents of the camera's hardware header in a CHwHeaderInfo structure.
- 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.
- HIS_RETURN [Acquisition_GetIntTimes](#) (HACQDESC hAcqDesc, double *dblIntTime, int *nIntTimes)
This function retrieves the current integration times.
- HIS_RETURN [Acquisition_GetLogLevel](#) (XislLoggingLevels *xislLogLvl)
This function is to get the logging level of the Xisl logging.
- HIS_RETURN [Acquisition_GetNextSensor](#) (ACQDESCPOS *Pos, HACQDESC *phAcqDesc)
You can use this function to iterate through all recognized sensors in the system.
- HIS_RETURN [Acquisition_Global_Cleanup](#) (void)
Acquisition_Global_Cleanup Counterpart of Acquisition_Global_Init. Cleans up global resources.

- HIS_RETURN [Acquisition_Global_Init](#) ([XIS_Init_Flags](#) flags)

Acquisition_Global_Init Initialize global resources.

- 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)

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

- 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.

4.3.1 Function Documentation

- #### 4.3.1.1 unsigned int [Acquisition_CallFPGASetCameraModelInternal](#) ([PAcquisitionDesc](#) pAcqDesc, unsigned char * *CommandString*, long *ILength*)

Wrapper for eltec set camera mode.

Parameters

in	<i>pAcqDesc</i>	Pointer to a valid Acquisition Descriptor structure.
in	<i>Command-String</i>	char* Must have length 8 - command to send to the detector.
in	<i>ILength</i>	For future use.

Returns

Returns HIS_ALL_OK on success or an appropriate error code otherwise

- #### 4.3.1.2 HIS_RETURN [Acquisition_Close](#) ([HACQDESC](#) hAcqDesc)

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

Parameters

<i>hAcqDesc</i>	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.

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

4.3.1.4 HIS_RETURN Acquisition_EnableLogging (BOOL *onOff*)

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

Parameters

<i>onOff</i>	Boolean value used to determine whether to turn internal logging on or off
--------------	--

Returns

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

4.3.1.5 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_GetNextSensor. For a programming example see the initialization part of the XISL demonstration. Note: In case of Network/IP sensors, only sensors with standard-gateway equal to zero are initialized automatically.

Parameters

<i>pdwNumSensors</i>	Address of a 4 byte integer that receives the number of recognized sensors.
<i>bEnableIRQ</i>	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.
<i>bAlwaysOpen</i>	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

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

4.3.1.6 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

<i>hAcqDesc</i>	Pointer to HACQDESC.
<i>pdwChannelType</i>	Address of a 4 byte integer that receives an id of the currently open communication device.
<i>pnChannelNr</i>	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. XRpad)

Returns

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

4.3.1.7 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwFrames</i>	Number of frames of acquisition buffer.
<i>dwRows</i>	Number of rows of the sensor.
<i>dwColumns</i>	Number of columns of the sensor.
<i>dwDataType</i>	Type of data of acquisition buffer (should always be two = unsigned short).
<i>dwSortFlags</i>	Type of sorting. This value depends on camera and used sensor (see sorting schemes).
<i>bIRQEnabled</i>	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.
<i>dwAcqType</i>	Only for internal use.
<i>dwSystemId</i>	PROM identification number of the used camera. This number is only important if the camera operates objectionably and you need any support from the manufacturer.
<i>dwSyncMode</i>	This parameter receives the sync mode, see table.
<i>dwHwAccess</i>	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 the manufacturer) 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 sent 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

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

4.3.1.8 HIS_RETURN Acquisition_GetHwHeaderInfo (HACQDESC hAcqDesc, CHwHeaderInfo * pInfo)

This function returns the contents of the camera's hardware header in a [CHwHeaderInfo](#) structure.

Parameters

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

Returns

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

4.3.1.9 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

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

Returns

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

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

This function retrieves the current integration times.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dblIntTime</i>	Pointer to an array of 8 byte floating point numbers. This array must contain at least 8 entries. The returned integration times are in microseconds.
<i>nIntTimes</i>	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 Returns HIS_ALL_OK on success or an appropriate error code on failure.

4.3.1.11 HIS_RETURN Acquisition_GetLogLevel (XislLoggingLevels * xislLogLvl)

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

Parameters

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

Due to the internal interfaces the return value does not distinguish between LEVEL_ALL and LEVEL_TRACE. If the actual logging level is LEVEL_ALL or LEVEL_TRACE the function always returns LEVEL_TRACE.

Returns

Returns HIS_ALL_OK if successful or appropriate error code.

4.3.1.12 HIS_RETURN Acquisition_GetNextSensor (ACQDESCPOS * Pos, HACQDESC * phAcqDesc)

You can use this function to iterate through all recognized sensors in the system.

Usage example:

```
ACQDESCPOS pos = 0;
do {
    HACQDESC hAcqDesc = (HACQDESC)0x0;
    if (Acquisition_GetNextSensor(&pos, &hAcqDesc) == HIS_ALL_OK) {
        std::cout << "Found a sensor with handle: " << hAcqDesc <<
        std::endl;
    }
} while (pos != 0);
```

For a more detailed programming example, see the initialization part of the XISL demonstration.

Parameters

<code>in, out</code>	<i>Pos</i>	<p>Pointer to an unsigned 4 byte integer that receives informations that are needed for subsequent calls of this function.</p> <ul style="list-style-type: none"> Expected input: <ul style="list-style-type: none"> pointer must not be NULL initialize pointed value to zero to call this function for the first time For subsequent calls of this function, do NOT change the value of this parameter Provided output value: <ul style="list-style-type: none"> not zero if the next sensor was found zero otherwise
<code>out</code>	<i>phAcqDesc</i>	<p>Pointer to a Handle of a structure that contains all needed parameters for acquisition (HACQDESC). Its value must not be used if the call of this function was not successful.</p>

Returns

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

4.3.1.13 HIS_RETURN Acquisition_Global_Cleanup (void)

Acquisition_Global_Cleanup Counterpart of Acquisition_Global_Init. Cleans up global resources.

Note

Call this function when only 1 thread in your program is running.

Returns

Returns HIS_ALL_OK if successful or appropriate error code.

4.3.1.14 HIS_RETURN Acquisition_Global_Init (XIS_Init_Flags flags)

Acquisition_Global_Init Initialize global resources.

Note

Call this function when only 1 thread in your program is running.

Remember to call [Acquisition_Global_Cleanup\(\)](#) when your program finishes and only 1 thread is running.

Parameters

<i>in</i>	<i>flags</i>	Combined value of enum XIS_Init_Flags. Defines which libraries to be initialized.
-----------	--------------	---

Returns

Returns HIS_ALL_OK if successful or appropriate error code.

4.3.1.15 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

<i>phAcqDesc</i>	Handle of a structure that contains all needed parameters for acquisition (HACQDESC-C). If you call Acquisition_Init the first time set hAcqDesc to NULL, in subsequent calls use the former returned value.
<i>dwBoardType</i>	This parameter defines on which communication device the sensor is located. Only one type of frame grabber can be used at the same time.
<i>nChannelNr</i>	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.
<i>bEnableIRQ</i>	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.
<i>Rows</i>	Number of sensor rows.
<i>Columns</i>	Number of sensor columns.
<i>dwSortFlag</i>	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.
<i>bSelfInit</i>	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.
<i>bAlwaysOpen</i>	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
 - HIS_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_TILE_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

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

4.3.1.16 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

<i>hAcqDesc</i>	Handle of a structure that contains all needed parameters for acquisition (HACQDESC).
<i>hWnd</i>	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.
<i>dwErrorMsg</i>	Defines a user message that is posted to hWnd if an error occurs during acquisition.
<i>dwLoosing-FramesMsg</i>	Defines a user message that is posted to hWnd if Acquisition_SetReady wasn't called by the application at the end of sorting.
<i>lpfnEndFrame-Callback</i>	Defines a function pointer that is called after the XISL did the sorting. 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. lpfnEndFrame-Callback 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
<i>lpfnEndAcq-Callback</i>	Defines a function pointer that is called after the XISL did the sorting. 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(HACQDESC hAcqDesc); In this routine you can perform any clean up at acquisition end. If this parameter is set to NULL, it is ignored.

Returns

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

4.3.1.17 HIS_RETURN Acquisition_SetLogLevel (XislLoggingLevels xislLogLevel)

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

Parameters

<i>xislLogLevel</i>	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.

4.3.1.18 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

<i>filePath</i>	Customizable file name for logging output file. If the filepath parameter is NULL a default logging file will be created.
<i>consoleOnOff</i>	Parameter used to enable or disable logging output to the console

Returns

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

4.3.1.19 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

<i>onOff</i>	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.

4.4 Init Functions for GbIF

Functions

- HIS_RETURN [Acquisition_GbIF_CheckNetworkSpeed](#) (HACQDESC hAcqDesc, WORD *wTiming, long *lPacketDelay, long lMaxNetworkLoadPercent)

This function determines which timing and packet delay 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 be 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_GbIF_SetCameraMode\(..\)](#) to apply the determined settings. For XRpad detectors a fixed value is returned for LAN/WLAN transmission. (80% network load are recommended)

- HIS_RETURN [Acquisition_GbIF_DiscoverDetectors](#) ()

Discover all NICs for detectors 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 address, 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 lose the temporary IP and behave as configured (e.g. IP per DHCP or LLA). This function is 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 is 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_GetTransmissionMode](#) (HACQDESC hAcqDesc, XIS_Transmission-
 _Mode *pTransmissionMode)
Acquisition_GbIF_SetTransmissionMode Set the transmission mode for the device.
- HIS_RETURN [Acquisition_GbIF_GetVersion](#) (int *pMajor, int *pMinor, int *pRelease, char *pStr-
 Version, int iStrLength)
Retrieve version information about the used GbIF library.
- HIS_RETURN [Acquisition_GbIF_Init](#) (HACQDESC *phAcqDesc, int nChannelNr, BOOL bEnableI-
 RQ, UINT uiRows, UINT uiColumns, BOOL bSelfInit, BOOL bAlwaysOpen, long lInitType, GBIF_S-
 TRING_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, un-
 signed long ulBootOptions, GBIF_STRING_DATATYPE *cDefIP, GBIF_STRING_DATATYPE *c-
 DefSubNetMask, GBIF_STRING_DATATYPE *cStdGateway)
*This function provides the parameters to configure how the detector connects to the network adapter. This
 function is 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 adds a delay between network packets which reduces the theoretical throughput
 but can lead to a better and more reliable connection. 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)
Set port range to be used with ethernet detecors to lStartPort, lNrOfports.
- HIS_RETURN [Acquisition_GbIF_SetTransmissionMode](#) (HACQDESC hAcqDesc, XIS_Transmission-
 _Mode transmissionMode)
Acquisition_GbIF_SetTransmissionMode Set the transmission mode for the device.
- HIS_RETURN [Acquisition_wpe_GetVersionNEW](#) (int *major, int *minor, int *release, int *build)
This function returns the version of the used wpe200 library if available and loadable.

4.4.1 Function Documentation

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

This function determines which timing and packet delay 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 be 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
<i>wTiming</i>	Pointer to suggested Free Running Timing for actual system performance.
<i>lPacketDelay</i>	Pointer to calculated maximum InterPacketDelay for suggested timing.
<i>lMaxNetwork-LoadPercent</i>	Percentage of network load for which the Packet Delay shall be checked. To allow a stable data transmission with some space for packet resend select ~80 percent.

Returns

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

4.4.1.2 HIS_RETURN Acquisition_GbIF_DiscoverDetectors ()

Discover all NICs for detectors via IP broadcast.

Note

Use `Acquisition_GbIF_DiscoveredDetectorCount` and `Acquisition_GbIF_DiscoveredDetectorByIndex` in subsequent calls to retrieve information about the discovered devices.

Returns

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

4.4.1.3 HIS_RETURN Acquisition_GbIF_DiscoveredDetectorByIndex (long lIndex, GBIF_DEVICE_PARAM * pDevice)

Retrieves a [GBIF_DEVICE_PARAM](#) structure by index.

Note

Use `Acquisition_GbIF_DiscoverDetectors` in a preceding call to discover the network.

Parameters

<i>lIndex</i>	Index of the device to retrieve the parameters. Valid indices are in the range $0 \leq \text{index} \leq (\text{count} - 1)$.
<i>pDevice</i>	Pointer to a GBIF_DEVICE_PARAM structure to retrieve the parameters of the device.

Returns

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

4.4.1.4 HIS_RETURN Acquisition_GbIF_DiscoveredDetectorCount (long * pDeviceCount)

Retrieves the count of the discovered devices.

Note

Use Acquisition_GbIF_DiscoverDetectors in a preceding call to discover the network.

Parameters

<i>pDeviceCount</i>	Pointer to a variable that retrieves the device count.
---------------------	--

Returns

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

4.4.1.5 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 address, 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 is not available for the XRpad.

Parameters

<i>cMAC</i>	MAC address of the device to retrieve a temporary IP.
<i>cDefIP</i>	Temporary IP address.
<i>cDefSubNetMask</i>	Temporary subnet mask.
<i>cStdGateway</i>	Temporary gateway.

Returns

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

4.4.1.6 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

<i>ucMAC</i>	MAC address of the device having the requested settings.
<i>ulBootOptions</i>	OR-able flag indicating the type of connection bitwisely
<i>ucDefIP</i>	Retrieves the IP address the device is connected with.
<i>ucDefSubNet-Mask</i>	Retrieves the subnet the device is in.
<i>ucStdGateway</i>	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

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

4.4.1.7 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

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

Returns

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

4.4.1.8 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

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

Note

- HIS_GbIF_IP The sensor is selected by the IP address passed in ucAddress
- HIS_GbIF_MAC The sensor is selected by the MAC address passed in ucAddress
- HIS_GbIF_NAME The sensor is selected by the detector name passed in ucAddress

Returns

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

4.4.1.9 HIS_RETURN Acquisition_GbIF_GetDeviceCnt (long * pINrOfboards)

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

Parameters

<i>pINrOfboards</i>	Retrieves the number of sensors found in the network broadcasting.
---------------------	--

Returns

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

4.4.1.10 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

<i>pGbIF_DEVICE_PARAM</i>	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
<i>nDeviceCnt</i>	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

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

4.4.1.11 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

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

Returns

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

4.4.1.12 HIS_RETURN Acquisition_GbIF_GetFilterDrvState (HACQDESC *hAcqDesc*)

Returns the status of the GbIF filter driver (if installed) otherwise an error code.

Parameters

<i>hAcqDesc</i>	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

Deprecated Cannot be used with current library versions.

Note

The filter driver is deprecated and can not be used with current library versions.

4.4.1.13 HIS_RETURN Acquisition_GbIF_GetPacketDelay (HACQDESC *hAcqDesc*, long * *IPacketdelay*)

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

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>IPacketdelay</i>	Retrieves the currently set value for Inter-Packet Delay

Returns

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

4.4.1.14 HIS_RETURN Acquisition_GbIF_GetTransmissionMode (HACQDESC *hAcqDesc*, XIS_Transmission_Mode * *pTransmissionMode*)

Acquisition_GbIF_SetTransmissionMode Set the transmission mode for the device.

Some devices support different transmission modes. This only applies for GigE detectors. Use this function to retrieve the transmission mode of the device. See enum XIS_Transmission_Mode and refer to the manual of your device for further information.

Parameters

in	<i>hAcqDesc</i>	Handle to a valid acquisition descriptor
out	<i>pTransmission-Mode</i>	Pointer to a variable the mode is passed to.

Returns

Returns HIS_ALL_OK on success or an appropriate error code in case of an error.

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

Retrieve version information about the used GbIF library.

Parameters

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

4.4.1.16 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

<i>phAcqDesc</i>	Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call Acquisition_GbIF_Init the first time set hAcqDesc to NULL, in subsequent calls use the former returned value.
<i>nChannelNr</i>	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.
<i>bEnableIRQ</i>	To run the acquisition in polling mode set this parameter to zero. To enable hardware interrupts set the parameter to one.
<i>uiRows</i>	Number of rows of the sensor.
<i>uiColumns</i>	Number of columns of the sensor.
<i>bSelfInit</i>	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.
<i>bAlwaysOpen</i>	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.
<i>lInitType</i>	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. IP address, MAC address and detector name can be retrieved by Acquisition_GbIF_GetDeviceList
<i>ucAddress</i>	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

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

4.4.1.17 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 is not available for the XRpad.

Parameters

<i>cMAC</i>	MAC address of the device to be configured.
<i>ulBootOptions</i>	OR-able flag to set the type of connection bitwisely.
<i>cDefIP</i>	In case <i>ulBootOptions</i> is equal to HIS_GbIF_IP_STATIC, <i>cDefIP</i> can be used to set a new value as static IP. For that, <i>cDefIP</i> has to contain an IP address when calling the function.
<i>cDefSubNetMask</i>	In case <i>ulBootOptions</i> is equal to HIS_GbIF_IP_STATIC, <i>cDefSubNetMask</i> can be used to set a new value as static IP. For that, <i>cDefSubNetMask</i> has to contain an IP address when calling the function.
<i>cStdGateway</i>	In case <i>ulBootOptions</i> is equal to HIS_GbIF_IP_STATIC, <i>cStdGateway</i> can be used to set a new value as static IP. For that, <i>cStdGateway</i> has to contain an IP address when calling the function. When <i>cStdGateway</i> is zero the detector will be able to be initialized by Acquisition_EnumSensors(..)

Note

- HIS_GbIF_IP_STATIC - Use the 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

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

4.4.1.18 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 discovering devices. This applies especially for noisy WLAN environments.

The default value is 750 milliseconds.

Parameters

<i>in</i>	<i>timeout</i>	Discovery timeout in milliseconds. Must be a positive value.
-----------	----------------	---

Returns

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

4.4.1.19 HIS_RETURN Acquisition_GbIF_SetPacketDelay (HACQDESC *hAcqDesc*, long *IPacketdelay*)

The Inter-Packet Delay can be set flexibly to balance out the workload of the IP connection between detector and network adapter. It adds a delay between network packets which reduces the theoretical throughput but can lead to a better and more reliable connection. It is recommended to be configured depending on the network load. The value can be retrieved by calling Acquisition_GbIF_CheckNetworkSpeed.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>IPacketdelay</i>	Value for Inter-Packet Delay, which is to be set in the unit TICKS

Returns

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

4.4.1.20 HIS_RETURN Acquisition_GbIF_SetPortRange (long *IStartPort*, long *INrOfports*)

Set port range to be used with ethernet detectors to IStartPort, INrOfports.

Parameters

<i>IStartPort</i>	Starting port (should be 1024)
<i>INrOfports</i>	Number of maximum devices (should be 16)

Note

The port and the range is setup properly per default. It is not recommended to change the values.

Returns

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

4.4.1.21 HIS_RETURN Acquisition_GbIF_SetTransmissionMode (HACQDESC *hAcqDesc*, XIS_Transmission_Mode *transmissionMode*)

Acquisition_GbIF_SetTransmissionMode Set the transmission mode for the device.

Some devices support different transmission modes. This only applies for GigE detectors. Use this function to change the transmission mode of the device. See enum `XIS_Transmission_Mode` and refer to the manual of your device for further information.

Parameters

in	<i>hAcqDesc</i>	Handle to a valid acquisition descriptor
in	<i>transmission-Mode</i>	The mode to be set

Returns

Returns `HIS_ALL_OK` on success or an appropriate error code in case of an error.

4.4.1.22 HIS_RETURN Acquisition_wpe_GetVersionNEW (int * *major*, int * *minor*, int * *release*, int * *build*)

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

Parameters

<i>major</i>	Major version
<i>minor</i>	Minor version
<i>release</i>	Release version
<i>build</i>	Build version

Note

Please use [Acquisition_wpe_GetVersionEx\(\)](#) instead.

Deprecated Please use [Acquisition_wpe_GetVersionEx\(\)](#)

Returns

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

4.5 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 integration period of the current frame. The detector will be immediately start a new readout. If the abort current frame is called during an active readout it will abort after the readout has been finished. The detector should run in the same mode as the image correction files have been obtained. Please note that the integration time is random depending on the used readout/trigger mode.

- HIS_RETURN [Acquisition_Acquire_GainImage](#) (HACQDESC hAcqDesc, WORD *pOffsetData, - 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. See mathematical description of corrections in XIS reference manual for a description of the gain data format. At the end of the acquisition time the gain data are also accessible via pdwGainData. 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. See mathematical description of corrections in XIS reference manual for a description of the gain data format. At the end of the acquisition time the gain data are also accessible via pdwGainData. 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 *pOffsetData, 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 Acquisition_Acquire_GainImage(..)-function. The function provides the possibility to use a well-defined region of interest (ROI) 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, D-WORD *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_SetCorrData_Ex(..) to set the correction data before.

- 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.

- 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 if pwOffsetData, pdwGainData and pdwPxlCorrList are not NULL. The routine returns immediately. If you want to be informed about frame end or acquisition end, then define in Acquisition_Init 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, 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 if pwOffsetData, pdwGainData and pdwPxlCorrList are not NULL. 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 hAcqDesc, UINT dwFrames, UINT dwSkipFrms, UINT dwOpt)

The function provides the same functionality as Acquisition_Acquire_Image except it does not load new correction data. Use Acquisition_SetCorrData_Ex before to set the correction data.

- 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 pwOffsetData. At the end of the acquisition time the averaged data are also accessible via pwOffsetData. 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 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 pwOffsetData. At the end of the acquisition time the averaged data are also accessible via pwOffsetData. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in Acquisition_Init suitable Callback functions and post from there a corresponding message to your application.

- 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 it does not load new correction data. Use Acquisition_SetCorrData_Ex before to all correction data to NULL.

- HIS_RETURN [Acquisition_CreateGainMap](#) (WORD *pGainData, WORD *pGainAVG, int nCount, int nFrame)

This function creates a list of median values for gain correction using a bright offset corrected 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.

- 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.

- 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.

- 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_DoPixelCorrection.

- 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.*

- 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.

- 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. 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, 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 acquisition 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 **ppdwPxICorrList)

This function returns the addresses 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 **ppdwPxICorrList)

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 acquired 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 *pdwPxICorrList)

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`. Use this function to change the correction data between frames after calling `Acquisition_Acquire_Image` or one of its derivatives.

- **HIS_RETURN Acquisition_SetCorrData_Ex** (**HACQDESC** `hAcqDesc`, unsigned short `*pwOffsetData`, unsigned short `*pwGainData`, unsigned short `*pwGainAvgData`, `UINT` `nGainFrames`, `DWORD` `*pdwGainData`, `DWORD` `*pdwPxI CorrList`)

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`.

4.5.1 Function Documentation

4.5.1.1 HIS_RETURN Acquisition_Abort (HACQDESC hAcqDesc)

This routine aborts a currently running acquisition.

Parameters

<code>hAcqDesc</code>	Handle of a valid Acquisition Descriptor.
-----------------------	---

Returns

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

4.5.1.2 HIS_RETURN Acquisition_AbortCurrentFrame (HACQDESC hAcqDesc)

This routine aborts the integration period of the current frame. The detector will be immediately start a new readout. If the abort current frame is called during an active readout it will abort after the readout has been finished. The detector should run in the same mode as the image correction files have been obtained. Please note that the integration time is random depending on the used readout/trigger mode.

Parameters

<code>hAcqDesc</code>	Handle of a valid Acquisition Descriptor
-----------------------	--

Returns

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

4.5.1.3 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.

See mathematical description of corrections in XIS reference manual for a description of the gain data format. At the end of the acquisition time the gain data are also accessible via `pdwGainData`. 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>pOffsetData</i>	Pointer that contains offset data. (see <code>Acquisition_Acquire_OffsetImage</code>). It is recommended to acquire the Offset shortly before calling <code>Acquisition_Acquire_GainImage</code>
<i>pdwGainData</i>	Pointer to buffer that receives the gain data.
<i>nRows</i>	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.
<i>nColumns</i>	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.
<i>nFrames</i>	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

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

4.5.1.4 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. See mathematical description of corrections in XIS reference manual for a description of the gain data format. At the end of the acquisition time the gain data are also accessible via `pdwGainData`. 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>pOffsetData</i>	Pointer that contains offset data. (see <code>Acquisition_Acquire_OffsetImage</code>). It is recommended to acquire the Offset shortly before calling <code>Acquisition_Acquire_GainImage</code> .
<i>pGainData</i>	Pointer to buffer that receives the gain data.

<i>nRows</i>	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.
<i>nCols</i>	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.
<i>nFrames</i>	Number of frames to acquire.
<i>dwOpt</i>	Options

Returns

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

4.5.1.5 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 Acquisition_Acquire_GainImage(..)-function. The function provides the possibility to use a well-defined region of interest (ROI) 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

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>pwOffsetData</i>	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). It is recommended to acquire the Offset shortly before calling Acquisition_Acquire_GainImage
<i>pdwGainData</i>	Pointer to buffer that receives the gain data
<i>nRows</i>	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.
<i>nColumns</i>	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.
<i>nFrames</i>	Number of frames to acquire.
<i>dwOpt</i>	must be 0
<i>uiULX</i>	UpperLeftX define a rect which is used to calculate the median for the pixel-gain calculation
<i>uiULY</i>	UpperLeftY define a rect which is used to calculate the median for the pixel-gain calculation
<i>uiBRX</i>	BottomRightX define a rect which is used to calculate the median for the pixel-gain calculation
<i>uiBRY</i>	BottomRightY define a rect which is used to calculate the median for the pixel-gain calculation
<i>uiMode</i>	See table

Note

- 0 - normal Gain whole image used for median determination.
- 1 - Median for pixel-gain calculation from ROI (defined by uiULX following)

- 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

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

4.5.1.6 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_SetCorrData_Ex(..) to set the correction data before.

Note

Please find the parameter description at Acquisition_Acquire_GainImage_Ex_ROI.

Returns

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

4.5.1.7 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_Ex_ROI.

Returns

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

4.5.1.8 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 if pwOffsetData, pdwGainData and pdwPxlCorrList are not NULL. The routine returns immediately. If you want to be informed about frame end or acquisition end, then define in Acquisition_Init suitable Callback functions and post from there a corresponding message to your application.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwFrames</i>	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.
<i>dwSkipFrms</i>	Number of frames to skip before a frame is copied into the acquisition buffer.
<i>dwOpt</i>	Options for sequence acquisition.
<i>pdwOffsetData</i>	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). The Offset must be actual. It is recommended to acquire them shortly before calling Acquisition_Acquire. If you do not want to perform an offset correction set this parameter to NULL.
<i>pdwGainData</i>	Pointer that contains gain data. (see Acquisition_Acquire_GainImage). If you do not want to perform a gain correction set this parameter to NULL.
<i>pdwPxlCorrList</i>	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 do not 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

If multiple calls are made to Acquisition_Acquire_Image without changing the correction data, then consider using Acquisition_Acquire_Image_PreloadCorr instead. This will avoid reloading the correction files which can be time consuming.

- 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 entire sequence into one buffer that accommodates all frames. 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

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

4.5.1.9 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 * pdwPxI CorrList)

This function acquires dwFrames frames and performs offset, gain and pixel corrections automatically if pwOffsetData, pdwGainData and pdwPxI CorrList are not NULL. 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwFrames</i>	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.
<i>dwSkipFrms</i>	Number of frames to skip before a frame is copied into the acquisition buffer.
<i>dwOpt</i>	Options for sequence acquisition.
<i>pwOffsetData</i>	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). The Offset must be actual. It is recommended to acquire them shortly before calling Acquisition_Acquire. If you do not want to perform an offset correction set this parameter to NULL.
<i>dwGainFrames</i>	Number of frames in pwGainData
<i>pwGainData</i>	pointer to the median-list created by Acquisition_CreateGainMap
<i>pwGainAvg-Data</i>	List of Medians representing the frames of the pwGainData sequence
<i>pdwGainData</i>	Pointer that contains gain data. (see Acquisition_Acquire_GainImage). If you do not want to perform a gain correction set this parameter to NULL
<i>pdwPxI CorrList</i>	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 do not 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

If multiple calls are made to Acquisition_Acquire_Image without changing the correction data, then consider using Acquisition_Acquire_Image_PreloadCorr instead. This will avoid reloading the correction files which can be time consuming.

- 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 entire sequence into one buffer that accommodates all frames. 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

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

4.5.1.10 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 it does not load new correction data. Use Acquisition_SetCorrData_Ex before to set the correction data.

Note

Please find the parameter description at [Acquisition_Acquire_Image\(\)](#)

Returns

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

4.5.1.11 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 pwOffsetData. At the end of the acquisition time the averaged data are also accessible via pwOffsetData. 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

<i>hAcqDesc</i>	Handle of a valid Acquisition descriptor.
<i>pwOffsetData</i>	Pointer to an acquisition buffer for offset data.
<i>nRows</i>	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.
<i>nColumns</i>	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.
<i>nFrames</i>	Number of frames to acquire.

Returns

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

4.5.1.12 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 *pwOffsetData*. At the end of the acquisition time the averaged data are also accessible via *pwOffsetData*. The routine returns immediately. If you want to be informed about frame end or acquisition end then define in *Acquisition_Init* suitable Callback functions and post from there a corresponding message to your application.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition descriptor.
<i>pwOffsetData</i>	Pointer to an acquisition buffer for offset data.
<i>nRows</i>	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.
<i>nColumns</i>	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.
<i>nFrames</i>	Number of frames to acquire.
<i>dwOpt</i>	For internal use only, shall always be 0.

Returns

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

4.5.1.13 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 it does not load new correction data. Use *Acquisition_SetCorrData_Ex* before to all correction data to NULL.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>pwOffsetData</i>	Pointer to a acquisition buffer for offset data.
<i>nRows</i>	Number of rows of the offset data buffer. If the value does not match with the currently connected sensor the function returns with an error.
<i>nColumns</i>	Number of columns of the offset data buffer. If the value does not match with the currently connected sensor the function return with an error.
<i>nFrames</i>	Number of frames to acquire.
<i>dwOpt</i>	For internal use only, shall always be 0

Returns

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

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

This function creates a list of median values for gain correction using a bright offset corrected 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

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

Note

: This functions has an XISL unusual return code in case of success.

Returns

Returns 1 on success or HIS_ERROR_CREATE_MEMORYMAPPING error code.

4.5.1.15 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

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

Returns

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

4.5.1.16 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

<i>uspPixelMask-Src</i>	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.
<i>dwRows</i>	Number of rows in the image
<i>dwColumns</i>	Number of columns in the image
<i>bpOnboard-PixelMask</i>	Output, on-board style (1 bit/pixel) pixel mask image buffer

Returns

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

4.5.1.17 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 *pwData*) that one can use in Acquisition_Acquire_Image or Acquisition_DoPixelCorrection.

Parameters

<i>pwData</i>	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.
<i>nDataRows</i>	Number of rows of the data source.
<i>nDataColumns</i>	Number of columns of the data source.
<i>pCorrList</i>	Pointer to an array (pixel map buffer) that receives the pixel correction data. If <i>pCorrList</i> is set to NULL then only the required size of the pixel map buffer is returned in <i>pnCorrListSize</i> .
<i>pnCorrListSize</i>	Pointer to an integer that receives the required size of the pixel map buffer if <i>pCorrList</i> is set to NULL otherwise it contains the size of the pixel buffer at function call time.

Returns

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

4.5.1.18 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

<i>hAcqDesc</i>	Pointer to HACQDESC.
<i>pProcessed-Data</i>	Pointer to the destination buffer.
<i>nFrames</i>	Number of frames of the destination buffer. It must be greater than zero.
<i>nRows</i>	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.
<i>nColumns</i>	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

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

4.5.1.19 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

<i>pSource</i>	Pointer to data source buffer.
<i>pDest</i>	Pointer to data destination buffer. This parameter can be equal to <i>pSource</i> .
<i>pOffsetData</i>	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). These data should be lately acquired, so that it is up-to-date. It is recommended to acquire them shortly before calling Acquisition_DoOffsetCorrection.
<i>nCount</i>	Number of pixels to correct.

Returns

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

4.5.1.20 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*. A suitable place to call this function is the end of frame callback function or the end acq callback.

Parameters

<i>pSource</i>	Pointer to data source buffer.
<i>pDest</i>	Pointer to data destination buffer. This parameter can be equal to <i>pSource</i> .

<i>pOffsetData</i>	Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). These data should be lately acquired, so that it is up-to-date. It is recommended to acquire them shortly before calling Acquisition_DoOffsetCorrection.
<i>pGainData</i>	Pointer that contains gain data. (see Acquisition_Acquire_GainImage).
<i>nCount</i>	Number of data entries to correct.

Returns

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

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

Similar to Acquisition_DoOffsetGainCorrection.

4.5.1.22 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 acquisition callback.

Parameters

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

Returns

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

4.5.1.23 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

<i>pData</i>	Pointer to data.
<i>pCorrList</i>	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. It is recommended to use Acquisition_CreatePixelMap to create this list.

Returns

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

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

This function returns the addresses of the applied correction buffers.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>ppwOffsetData</i>	Point to offset data.
<i>ppdwGainData</i>	Pointer to gain data.
<i>ppdwPxlCorr-List</i>	Pointer to pixel correction list.

Returns

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

4.5.1.25 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>ppwOffsetData</i>	Point to offset data. (see Acquisition_Acquire_OffsetImage).
<i>ppwGainData</i>	Pointer to a sequence of offset corrected data. The images in the array must be sorted by median
<i>ppwGainAvg-Data</i>	Pointer to the median-list created by Acquisition_CreateGainMap

<i>nGainFrames</i>	UINT *pointer retrieving the number of frames in pwGainData
<i>ppdwGainData</i>	Pointer that contains gain data (see Acquisition_Acquire_GainImage).
<i>ppdwPxlCorr-List</i>	Pointer to a pixel correction list (see Acquisition_DoPixelCorrection)

Returns

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

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

Use this function to retrieve the last acquired 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

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

Returns

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

4.5.1.27 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. Use this function to change the correction data between frames after calling Acquisition_Acquire_Image or one of its derivatives.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>pwOffsetData</i>	Point to offset data (see Acquisition_Acquire_OffsetImage).
<i>pdwGainData</i>	Pointer that contains gain data (see Acquisition_Acquire_GainImage).
<i>pdwPxlCorrList</i>	Pointer to a pixel correction list (see Acquisition_DoPixelCorrection).

Returns

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

4.5.1.28 HIS_RETURN Acquisition_SetCorrData_Ex (HACQDESC *hAcqDesc*, unsigned short * *pwOffsetData*, unsigned short * *pwGainData*, unsigned short * *pwGainAvgData*, UINT *nGainFrames*, DWORD * *pdwGainData*, DWORD * *pdwPxI CorrList*)

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

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>pwOffsetData</i>	Pointer to offset data (see Acquisition_Acquire_OffsetImage).
<i>pwGainData</i>	Pointer to a sequence of offset corrected data. The images in the array must be sorted by median.
<i>pwGainAvg-Data</i>	Pointer to the median-list created by Acquisition_CreateGainMap
<i>nGainFrames</i>	number of frames in <i>pwGainData</i>
<i>pdwGainData</i>	Pointer that contains gain data (see Acquisition_Acquire_GainImage).
<i>pdwPxI CorrList</i>	Pointer to a pixel correction list (see Acquisition_DoPixelCorrection)

Returns

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

4.6 Setting up the Device

Functions

- HIS_RETURN [Acquisition_GetCameraBinningMode](#) (HACQDESC hAcqDesc, WORD *wMode)
Use this function to retrieve the detectors binning mode. See the description of Acquisition_SetCameraBinningMode for possible returned binning modes.
- HIS_RETURN [Acquisition_GetCameraFOVMode](#) (HACQDESC hAcqDesc, WORD *wMode)
Use this function to retrieve the detectors FOV mode. See the description of Acquisition_SetCameraFOVMode for possible returned FOV modes.
- HIS_RETURN [Acquisition_GetCameraROI](#) (HACQDESC hAcqDesc, unsigned short *usActivateGrp)
This function returns the currently activated Region of Interest in Sectional Readout Mode. See the description of Acquisition_SetCameraROI for possible returned ROI values.
- HIS_RETURN [Acquisition_GetCameraTriggerMode](#) (HACQDESC hAcqDesc, WORD *wMode)
Retrieves trigger mode for detectors with Header > 14 and Detectortype > 2.
- HIS_RETURN [Acquisition_GetIpAddress](#) (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 the 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 and binning options.
- HIS_RETURN [Acquisition_SetCameraFOVMode](#) (HACQDESC hAcqDesc, WORD wMode)
This function selects the field of view for XRD 4343RF detectors.
- HIS_RETURN [Acquisition_SetCameraGain](#) (HACQDESC hAcqDesc, WORD wMode)
This function can be used to select the gain of the detector.
- 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.
- 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, WORD wMode)
The function sets the internal trigger scheme for Detectors.
- 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 and 3025/4336/4343 XRpad2.
- 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 evaluation purposes. Please contact the application team to get information about how to set the DACOffset.
- HIS_RETURN [Acquisition_SetFrameSyncMode](#) (HACQDESC hAcqDesc, DWORD dwMode)

This function sets the synchronization mode of the detector.

- HIS_RETURN [Acquisition_SetFrameSyncTimeMode](#) ([HACQDESC](#) hAcqDesc, unsigned int ui-Mode, 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](#) 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.

4.6.1 Function Documentation

4.6.1.1 HIS_RETURN Acquisition_GetCameraBinningMode (HACQDESC hAcqDesc, WORD * wMode)

Use this function to retrieve the detectors binning mode. See the description of [Acquisition_SetCameraBinningMode](#) for possible returned binning modes.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>wMode</i>	Pointer to value to retrieve the actual binning mode.

Returns

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

4.6.1.2 HIS_RETURN Acquisition_GetCameraFOVMode (HACQDESC hAcqDesc, WORD * wMode)

Use this function to retrieve the detectors FOV mode. See the description of [Acquisition_SetCameraFOVMode](#) for possible returned FOV modes.

Parameters

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

Returns

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

4.6.1.3 HIS_RETURN Acquisition_GetCameraROI (HACQDESC *hAcqDesc*, unsigned short * *usActivateGrp*)

This function returns the currently activated Region of Interest in Sectional Readout Mode. See the description of Acquisition_SetCameraROI for possible returned ROI values.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>usActivateGrp</i>	Pointer to unsigned short value representing a bitwise Selection of Groups to Readout and Transmit

Returns

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

4.6.1.4 HIS_RETURN Acquisition_GetCameraTriggerMode (HACQDESC *hAcqDesc*, WORD * *wMode*)

Retrieves trigger mode for detectors with Header >14 and Detectortype >2.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>wMode</i>	Command 42: 0 - DDD 1 - DDD without clearance scan 2 - Start Stop [3] - Trigger frames (trigger first type)

Returns

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

4.6.1.5 HIS_RETURN Acquisition_GetIpAddress (HACQDESC *hAcqDesc*, const char ** *ipAddress*)

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

Parameters

in	<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
out	<i>ipAddress</i>	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

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

4.6.1.6 HIS_RETURN Acquisition_ResetFrameCnt (HACQDESC hAcqDesc)

This function is used to set the 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

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
-----------------	---

Returns

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

4.6.1.7 HIS_RETURN Acquisition_SetCameraBinningMode (HACQDESC hAcqDesc, WORD wMode)

Use this function to set the detectors binning mode and binning options.

Attention

The support of binning modes and additional options is detector dependent.
Please refer to the detector manual for a table of supported binning modes and options.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>wMode</i>	Used for selection of binning mode and additional options <ul style="list-style-type: none"> • bits 0 to 7: binning mode selection value <ul style="list-style-type: none"> – value 1: no binning (default) – value 2 to 5: detector dependent binning mode • bits 8 to 15: bit mask with additional binning options. The options are mutually exclusive. If no option is set, the default option will be used automatically. <ul style="list-style-type: none"> – bit 8: use averaged binning (default option) – bit 9: use accumulated binning

Returns

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

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

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

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure.
<i>wMode</i>	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

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

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

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

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>wMode</i>	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 => 1.3pF. 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

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

4.6.1.10 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwMode</i>	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.

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

This function selects a Defined Region of Interest for Readout.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
<i>usActivateGrp</i>	Bitwise Selection of Groups to Readout and Transmit.

Returns

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

Note

This paragraph describes the Sectional Readout which is available for detectors with Cameratyp e 10 and 12. With this function predefined sections of the detector field of view can be selected and combined to one adjacent readout Section. Each of the four sections has a size of 1024 columns and 255 rows (XRD 0822 xP) respectively 128 Rows (XRD 1642 xP). Location and size of the region can be set bitwise with the API function.

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

The function sets the internal trigger scheme for Detectors.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>wMode</i>	see table

Note

Command 42:

- 0 - WM / DDD
- 1 - WM / DDD without clearance scan
- 2 - Start Stop
- 3 - Trigger frames (trigger first type)
- 4 - AutoTrigger frames
- 5 - Trigger on row tag (framewise with filter on trigger input) / RnF frameWise 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_SetFrameSyncMode\(\)](#)

Returns

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

4.6.1.13 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 and 3025/4336/4343 XRpad2.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>wDACOffsetValue</i>	wADCOffsetValue to be set. This value has to be between 1 and 4095

Returns

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

4.6.1.14 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 evaluation purposes. Please

contact the application team to get information about how to set the DACOffset.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure.
<i>wBinningMode</i>	Current binning mode.
<i>dblFps</i>	Number of frames per second.
<i>pwValueToFPGA</i>	Pointer to a variable to retrieve the value send to the FPGA.

Returns

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

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

This function sets the synchronization mode of the detector.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwMode</i>	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 required by RnF and 1611 detectors to route the external trigger signal attached to the frame grabber to the detector

Returns

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

4.6.1.16 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

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>uiMode</i>	Timing Mode must be 0 .. 7
<i>dwDelayTime</i>	Additional delay time in milliseconds (can be 0-ms up to (5000 ms-intTime)

```
// 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

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

4.6.1.17 HIS_RETURN Acquisition_SetTimerSync (HACQDESC hAcqDesc, DWORD * dwCycleTime)

This function configures the CycleTime for internal triggered mode.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwCycleTime</i>	Pointer to a 32 bit integer that provides the required cycle time in microseconds. 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 discrete steps. That's why the function returns the realized cycle time in dwCycleTime.

The parameter dwCycleTime should typically be smaller or equal to 5,000,000 microseconds.

Returns

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

4.6.1.18 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>usTiggerOutSignalMode</i>	See table
<i>usEP.SeqLength</i>	Defines the sequence length for EP mode
<i>usEP.FirstBrightFrm</i>	Defines the first frame in the sequence where the TrigOut signal is activated
<i>usEP.LastBrightFrm</i>	Defines the last frame in the sequence where the TrigOut signal is activated
<i>usEP.Delay1</i>	Defines the Delay1 from begin of frame in the EP_Frames until the TrigOut signal gets activated
<i>usEP.Delay2</i>	Defines the Delay2 from begin of frame in the EP_Frames until the TrigOut signal gets deactivated
<i>usDDD.Delay</i>	Additional delay in DDD/AED/Phototimed mode
<i>iTriggerOnRisingEdgeEnable</i>	0 -Trigger falling edge 1 -trigger on rising edge
<i>iSaveAsDefault</i>	1 - Save <i>usTiggerOutSignalMode</i> and <i>iTriggerOnRisingEdgeEnable</i> in EEPROM to be available 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

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

4.7 Callback function

Functions

- 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 set 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 *hWnd)
This function retrieves the current acquisition window handle if defined before.
- 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 set 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_SetAcqData. 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.

4.7.1 Function Documentation

4.7.1.1 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 set by Acquisition_Init.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwAcqData</i>	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

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

4.7.1.2 HIS_RETURN Acquisition_GetActFrame (HACQDESC *hAcqDesc*, DWORD * *dwActAcqFrame*, DWORD * *dwActSecBuffFrame*)

This function retrieves the current acquisition frames.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwActAcqFrame</i>	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 repeatedly from 1 to 8.
<i>dwActSecBuffFrame</i>	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 Acquisition_DefineDest-Buffers.

Returns

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

4.7.1.3 HIS_RETURN Acquisition_GetReady (HACQDESC *hAcqDesc*)

This function checks whether the passed Acquisition Descriptor is valid.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
-----------------	---

Returns

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

4.7.1.4 HIS_RETURN Acquisition_GetWinHandle (HACQDESC *hAcqDesc*, HWND * *hWnd*)

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

Parameters

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

Returns

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

4.7.1.5 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_SetAcqData. This has the advantage that a variety of parameters, which are defined within the structure/value/class, are accessible within the callback functions.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>dwAcqData</i>	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 EndFrameCallback/ EndAcqCallback by using Acquisition_GetAcqData(..). 64bit: Data to be accessible within callback functions: Pass a pointer to a user defined structure/value/class which can be retrieved in the EndFrameCallback/ 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

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

4.7.1.6 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

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

Returns

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

4.8 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.

4.8.1 Function Documentation

4.8.1.1 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

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

Returns

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

4.8.1.2 HIS_RETURN Acquisition_wpe_GetErrorCode (void)

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

Returns

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

4.8.1.3 HIS_RETURN Acquisition_wpe_GetErrorCodeEx (char * pBuffer, long len)

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

Parameters

<i>pBuffer</i>	Pointer to a character array to receive the latest error message. (char[256] recommended).
<i>len</i>	Length of the passed character array.

Returns

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

4.9 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](#) ()
Called when dll is loaded, initialize 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)
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)
Retrieves 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 retrieves the [GBIF_Detector_Properties](#) structure, which contains permanently stored information of the connected device.
- HIS_RETURN [Acquisition_GetFileInfo](#) (XislFileHandle fileHandle, XislFileInfo *fileInfo)
Retrieves basic information about a file.
- HIS_RETURN [Acquisition_GetHwHeader](#) (HACQDESC hAcqDesc, unsigned char *pData, unsigned int uiSize)
*This function returns the contents of the camera's hardware header in a char Array
It is designed only for internal usage*
- 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 required 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)
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 enable/disable logging to a file. When enabled, the log will be written to the file with the provided file name.
- 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_GetVersion](#) (int *major, int *minor, int *release, int *build)
Retrieve version information about the used wpe library.
- HIS_RETURN [Acquisition_wpe_GetVersionEx](#) (int *pMajor, int *pMinor, int *pRelease, char *pStrVersion, int iStrLength)
Retrieve version information about the used wpe library.

4.9.1 Function Documentation

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

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

Parameters

<i>ipAddress</i>	IP address of the device to control.
<i>buffer</i>	Buffer to hold the return text.
<i>bufferLen</i>	Length of the buffer.

Returns

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

4.9.1.2 HIS_RETURN Acq_WPE_Init ()

Called when dll is loaded, initialize wpe, global objects, etc.

Returns

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

4.9.1.3 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>bActivate</i>	Show Service Data: <i>bActivate</i> = TRUE otherwise: <i>bActivate</i> = FALSE

Returns

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

4.9.1.4 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

<i>pMemoryFile-Buffer</i>	Output buffer that will contain the XIS image file (with header). To determine the size of this buffer call <code>Acquisition_GetXISFileBufferSize</code> .
<i>pDataBuffer</i>	Input image data buffer (without header information).

ATTENTION: Special case for `PKI_ERRORMAPONBOARD` The function expects a WORD value per pixel as input. The values are transformed to the bitmaskarray. A WORD value needs to be 0xFFFF to get a bit of the resulting bitmaskarray set. Otherwise the associated bit is unset.

Parameters

<i>dwRows</i>	Number of rows in the image.
<i>dwColumns</i>	Number of columns in the image.
<i>dwFrames</i>	Number of frames in the image.
<i>uiOnboardFile-Header</i>	A boolean value indicating whether the file has the on-board style header (TRUE) or not (FALSE).
<i>filetype</i>	The data-type of the pixels in the image.

Returns

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

4.9.1.5 HIS_RETURN Acquisition_DeleteFile (XisIFileHandle fileHandle)

Deletes a file from a location.

Parameters

<i>in</i>	<i>fileHandle</i>	A valid file handle.
-----------	-------------------	----------------------

Returns

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

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

Similar to `Acquisition_DoOffsetCorrection`.

4.9.1.7 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`.

4.9.1.8 HIS_RETURN Acquisition_GetConnectionStatus (HACQDESC *hAcqDesc*)

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

Parameters

in	<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
----	-----------------	---

Returns

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

Note

Currently only valid for network connected detectors.

4.9.1.9 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

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
<i>uiMode</i>	Mode to read (0-63).
<i>pwValue</i>	Pointer to a value to retrieve the offset value.

Returns

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

Note

Currently only available for 4343RF detectors

4.9.1.10 HIS_RETURN Acquisition_GetDetectorProperties (HACQDESC *hAcqDesc*, GBIF_Detector_Properties * *pDetectorProperties*)

This function retrieves the [GBIF_Detector_Properties](#) structure, which contains permanently stored information of the connected device.

Parameters

<i>hAcqDesc</i>	Acquisition descriptor structure.
<i>pDetector-Properties</i>	Pointer to instance of GBIF_Detector_Properties

Returns

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

4.9.1.11 HIS_RETURN Acquisition_GetFileInfo (XislFileHandle *fileHandle*, XislFileInfo * *fileInfo*)

Retrieves basic information about a file.

Parameters

in	<i>fileHandle</i>	A valid file handle.
out	<i>fileInfo</i>	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 on failure.

4.9.1.12 HIS_RETURN Acquisition_GetHwHeader (HACQDESC *hAcqDesc*, unsigned char * *pData*, unsigned int *uiSize*)

This function returns the contents of the camera's hardware header in a char Array

It is designed only for internal usage

Parameters

<i>hAcqDesc</i>	
<i>pData</i>	Pointer to an allocated List of size uiSize to be filled with el_GetHeader
<i>uiSize</i>	size of allocated Memory in pData

Returns

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

4.9.1.13 HIS_RETURN Acquisition_GetRotationAngle (HACQDESC *hAcqDesc*, long * *IRotAngle*)

Get onboard rotation setting for FG-E Opto.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure
<i>IRotAngle</i>	Retrieves rotation angle.

Returns

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

4.9.1.14 HIS_RETURN Acquisition_GetTriggerOutStatus (HACQDESC *hAcqDesc*, int * *iTriggerStatus*)

This function retrieves the triggerout status from detector.

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure.
<i>iTriggerStatus</i>	See table

Note

- -1 Error, probably timeout
- 0 trigger in was low
- 1 trigger in was high

Note

Only available for XRpad detectors.

Returns

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

4.9.1.15 HIS_RETURN Acquisition_GetVersion (int * *major*, int * *minor*, int * *release*, int * *build*)

Reads the version of the XISL.dll.

Parameters

<i>major</i>	Pointer to a variable to retrieve the major version number.
<i>minor</i>	Pointer to a variable to retrieve the minor version number.
<i>release</i>	Pointer to a variable to retrieve the release number.
<i>build</i>	Pointer to a variable to retrieve the build number.

Returns

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

4.9.1.16 HIS_RETURN Acquisition_GetXISFileBufferSize (*size_t* * *pFileSize*, *UINT* *dwRows*, *UINT* *dwColumns*, *UINT* *dwFrames*, *BOOL* *uiOnboardFileHeader*, *XIS_FileType* *filetype*)

This function returns the required size(in bytes) of an XIS image file with header information.

Parameters

<i>pFileSize</i>	Requested file size in bytes
<i>dwRows</i>	Number of rows in the image
<i>dwColumns</i>	Number of columns in the image
<i>dwFrames</i>	Number of frames in the image
<i>uiOnboardFile-Header</i>	A boolean value indicating whether the file has the on-board style header (TRUE) or not (FALSE)
<i>filetype</i>	The data-type of the pixels in the image.

Returns

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

Note

For XIS_FileType PKI_ERRORMAPONBOARD just the first frame is considered.

4.9.1.17 HIS_RETURN Acquisition_IsAcquiringData (*HACQDESC* *hAcqDesc*)

This function tests if the XISL is currently acquiring data.

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor.
-----------------	---

Returns

Return value	Description
0	FALSE (No ongoing image acquisition related to the handle.)
1	TRUE (Ongoing image acquisition related to the handle.)
7	HIS_ERROR_INVALIDACQDESC (Handle of provided valid Acquisition Descriptor is invalid.)

4.9.1.18 HIS_RETURN Acquisition_LoadFile (XislFileHandle *fileHandle*, unsigned char ** *buffer*)

Loads a file from several locations.

Parameters

in	<i>fileHandle</i>	A valid file handle.
out	<i>buffer</i>	Pointer to buffer to retrieve the location of the data.

Returns

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

Note

This function is blocking. Depending on size and location of a file, the calling thread might block for an unpredictable period.

4.9.1.19 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

<i>filename</i>	Path to the input file.
<i>pMemoryFile-Buffer</i>	Output image data buffer. This should be appropriately allocated to fit all the image and header data.
<i>bufferSize</i>	The size of the output buffer (in bytes) this should be equal to the size of the image and header.

Returns

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

4.9.1.20 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

<i>filename</i>	Path to the output file.
<i>pImageBuffer</i>	Input image data buffer (without header information).
<i>dwRows</i>	Number of rows in the image.
<i>dwColumns</i>	Number of columns in the image.

<i>dwFrames</i>	Number of frames in the image.
<i>uiOnboardFileHeader</i>	A boolean value indicating whether the file is to have the on-board style header (TRUE) or not (FALSE).
<i>filetype</i>	The data-type of the pixels in the image.

Note

If *uiOnboardFileHeader* is true only types *PKI_SHORT* and *PKI_ERRORMAPONBOARD* are allowed otherwise *PKI_SHORT* and *PKI_LONG* are allowed.

Returns

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

4.9.1.21 **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

<i>filename</i>	Path to the output file.
<i>buffer</i>	Input image data buffer (with or without header information).
<i>bufferSize</i>	The size of the data buffer (including any headers used) in bytes.

Returns

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

4.9.1.22 **HIS_RETURN Acquisition_SetConsoleLogging** (*BOOL enableConsole*)

This function will be used to enable or disable logging to the console window.

Parameters

<i>enableConsole</i>	Parameter used to enable or disable logging output to the console.
----------------------	--

Returns

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

4.9.1.23 HIS_RETURN Acquisition_SetDACOffsetFloorValueByMode (HACQDESC *hAcqDesc*, unsigned int *uiMode*)

This function changes the detector DAC Offset setting for the selected mode.

Parameters

<i>hAcqDesc</i>	Acquisition descriptor structure.
<i>uiMode</i>	Mode that should be changed (0-63).

Returns

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

Note

Currently only available for 4343RF detectors; the value for the selected mode is retrieved from detector flash.

4.9.1.24 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.

Parameters

<i>hAcqDesc</i>	Acquisition descriptor structure.
<i>uiMode</i>	Mode that should be changed (0-63).
<i>wValue</i>	(0-4095) wValue Value to set for the mode.

Returns

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

Note

Currently only available for 4343RF detectors; only modes ≥ 32 are changeable.

4.9.1.25 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.

Parameters

<i>hAcqDesc</i>	Acquisition descriptor structure.
<i>uiMode</i>	Mode that should be changed (0-63).
<i>wValue</i>	wValue Value to set for the mode.

Returns

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

Note

Currently only available for 4343RF detectors

4.9.1.26 HIS_RETURN Acquisition_SetFileLogging (const char * *filename*, BOOL *enableLogging*)

This function will enable/disable logging to a file. When enabled, the log will be written to the file with the provided file name.

Parameters

<i>filename</i>	If file logging is enabled: Customizable name of file to which the log data will be written. If file logging is disabled: ignored
<i>enableLogging</i>	Parameter used to enable or disable file logging.

Returns

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

4.9.1.27 HIS_RETURN Acquisition_SetFPGACameraMode (HACQDESC *hAcqDesc*, FPGAType *FPGACommand*, BOOL *blInverse*)

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

Parameters

<i>hAcqDesc</i>	Handle of a valid Acquisition Descriptor
<i>FPGA-Command</i>	Command to send to the detector
<i>blInverse</i>	Do a bitwise invers on the data field

Returns

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

4.9.1.28 HIS_RETURN Acquisition_SetRotationAngle (HACQDESC *hAcqDesc*, long *IRotAngle*)

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

Parameters

<i>hAcqDesc</i>	Pointer to acquisition descriptor structure.
<i>IRotAngle</i>	Rotation angle: must be 0, 90 or -90

Returns

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

4.9.1.29 HIS_RETURN Acquisition_wpe_GetVersion (int * *major*, int * *minor*, int * *release*, int * *build*)

Retrieve version information about the used wpe library.

Parameters

<i>major</i>	Pointer to a variable to retrieve the major version number.
<i>minor</i>	Pointer to a variable to retrieve the minor version number.
<i>release</i>	Pointer to a variable to retrieve the release number.
<i>build</i>	Pointer to a variable to retrieve the build number.

Note

Deprecated, please use [Acquisition_wpe_GetVersionEx\(\)](#)

Returns

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

4.9.1.30 HIS_RETURN Acquisition_wpe_GetVersionEx (int * *pMajor*, int * *pMinor*, int * *pRelease*, char * *pStrVersion*, int *iStrLength*)

Retrieve version information about the used wpe library.

Parameters

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

4.10 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_CONFLICT](#) 167
- #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_AVAILABLE_SYSTEMS](#) 154
- #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_INIT_DET_OPTIONS` 176
- `#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_NOT_SUPPORTED` 165
- `#define HIS_ERROR_NR_OF_BOARDS_CHANGED` 58
- `#define HIS_ERROR_ONBOARDVGFFAILED` 63
- `#define HIS_ERROR_OPEN_FILE` 76
- `#define HIS_ERROR_READ_DATA` 26
- `#define HIS_ERROR_RESET_ZYNQ` 172
- `#define HIS_ERROR_RETRIEVE_ENHANCED_HEADER` 107
- `#define HIS_ERROR_SET_CHARGE_MODE` 118
- `#define HIS_ERROR_SET_EVENT_CALLBACK` 169
- `#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_PACKET_DELAY` 153
- `#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_SETHEADERSIZE` 160
- `#define HIS_ERROR_SETLINETRIG_MODE` 24
- `#define HIS_ERROR_SETREGISTERTIMEOUT` 161
- `#define HIS_ERROR_SETTIMERSYNC` 19
- `#define HIS_ERROR_SLOW_SYSTEM` 32
- `#define HIS_ERROR_TIMEOUT` 6
- `#define HIS_ERROR_TRANSMISSION_MODE` 166
- `#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_WLAN_RESTART` 168
- `#define HIS_ERROR_WRITE_DATA` 25
- `#define HIS_ERROR_WRONG_CAMERA_MODE` 48
- `#define HIS_ERROR_WRONGBOARDSELECT` 47
- `#define HIS_ERROR_WSA` 164
- `#define HIS_ERROR_XRPD_BATTERY_COM` 163
- `#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_DETECTOR_IN_DEEP_SLEEP` 157
- `#define HIS_ERROR_XRPD_DISABLE_SYSLOG_SAVING` 152
- `#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_DEEPSLEEPIDLELOCATIONS` 155
- `#define HIS_ERROR_XRPD_GET_DEF_BOOT_CFG` 170
- `#define HIS_ERROR_XRPD_GET_DET_OPTIONS` 175
- `#define HIS_ERROR_XRPD_GET_DET_TYPE` 174
- `#define HIS_ERROR_XRPD_GET_EPC_REGISTER` 159
- `#define HIS_ERROR_XRPD_GET_SDCARD_INFO` 113
- `#define HIS_ERROR_XRPD_GET_SDCARD_TIMEOUT` 142
- `#define HIS_ERROR_XRPD_GET_TEMP_VALUES` 158
- `#define HIS_ERROR_XRPD_GET_TEMPERATURE_THRESHOLDS` 129
- `#define HIS_ERROR_XRPD_GET_WLAN_CC` 149
- `#define HIS_ERROR_XRPD_GET_WLAN_ChannelList` 151

- #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_DEEPSLEEPIDLELOCATIONS](#) 156
- #define [HIS_ERROR_XRPD_SET_DEF_BOOT_CFG](#) 171
- #define [HIS_ERROR_XRPD_SET_EPC_REGISTER](#) 162
- #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_SET_WLAN_CC](#) 150
- #define [HIS_ERROR_XRPD_VERIFY_GENUINENESS](#) 124

Typedefs

- typedef uint [ACQDESCPOS](#)
- typedef handle [HACQDESC](#)
- typedef enum [OnboardBinningMode](#) [OnboardBinningMode](#)
- typedef enum [ProcScriptOperation](#) [ProcScriptOperation](#)
enum used for addressing XRpad2 on board correction stages.
- 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_Init_Flags](#) [XIS_Init_Flags](#)
- typedef enum [XIS_Library_Event](#) [XIS_Library_Event](#)
- typedef enum [XIS_Sensor_Event](#) [XIS_Sensor_Event](#)

- typedef enum [XIS_Transmission_Mode](#) [XIS_Transmission_Mode](#)
Transmission modes.
- typedef enum [XIS_Xrpd_Event](#) [XIS_Xrpd_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_DataInterfaceControlEnum](#)
- 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, [ONBOARDBINNING2x2](#) = 1, [ONBOARDBINNING4x1](#) = 2, [ONBOARDBINNING4x4](#) = 3, [ONBOARDBINNING3x3](#) = 4, [ONBOARDBINNING9to4](#) = 5 }
- enum [ProcScriptOperation](#) { [PREBINNING](#), [PREMEAN](#), [PRESTOREBUFFER](#), [OFFSET](#), [GAIN](#), [MEAN](#), [PREVIEW](#), [BINNING](#), [STOREBUFFER](#), [STORESD](#), [SEND](#), [OFFSET_2](#), [GAIN_2](#) }
enum used for addressing XRpad2 on board correction stages.
- enum [XIS_Acquisition_Event](#) { [XAE_TRIGOUT](#) = 0x00000002, [XAE_READOUT](#) = 0x00000004, [XAE_TRIGGERED](#) = 0x00000008, [XAE_AED_READY](#) = 0x00000010 }
- enum [XIS_Battery_Event](#) { [XBE_BATTERY_REPORT](#) = 0x00000001, [XBE_BATTERY_WARNING](#) = 0x00000002 }
- enum [XIS_Detector_Event](#) { [XDE_BUFFERS_IN_USE](#) = 0x00000001, [XDE_STORED_IMAGE](#) = 0x00000002, [XDE_DROPPED_IMAGE](#) = 0x00000003, [XDE_POWER_STATE_CHANGED](#) = 0x00000004, [XDE_POWER_STATE_CHANGE_FAILED](#) = 0x00000005 }
- enum [XIS_Detector_TRIGOUT_SignalMode](#) { [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](#) }
- enum [XIS_DetectorTriggerMode](#) { [TRIGGERMODE_DDD](#), [TRIGGERMODE_DDD_WO_CLEARANCE](#), [TRIGGERMODE_STARTSTOP](#), [TRIGGERMODE_FRAMEWISE](#), [TRIGGERMODE_AED](#), [TRIGGERMODE_ROWTAG](#), [TRIGGERMODE_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_NETWORK_EVENT](#) = 0x00000007, [XE_DETECTOR_EVENT](#) = 0x00000008, [XE_LIBRARY_EVENT](#) = 0x00000009, [XE_SDCARD_FSCK_EVENT](#) = 0x0000000A, [XE_XRPD_EVENT](#) = 0x0000000B }

- enum `XIS_FileType` { `PKI_RESERVED` = 1, `PKI_DOUBLE` = 2, `PKI_SHORT` = 4, `PKI_SIGNED` = 8, `PKI_ERRORMAPONBOARD` = 16, `PKI_LONG` = 32, `PKI_SIGNEDSHORT` = `PKI_SHORT` | `PKI_SIGNED`, `PKI_SIGNEDLONG` = `PKI_LONG` | `PKI_SIGNED`, `PKI_FAULTMASK` = `PKI_LONG` | `PKI_RESERVED` }
- enum `XIS_Init_Flags` { `XIF_XISL` = 0x00000001, `XIF_GBIF` = 0x00000002, `XIF_WPE200` = 0x00000004, `XIF_CURL` = 0x00000008, `XIF_ALL` = 0xFFFFFFFF }
- enum `XIS_Library_Event` { `XLE_HIS_ERROR_PACKET_LOSS` = 0x00000001 }
- enum `XIS_Sensor_Event` { `XSE_HALL` = 0x00000001, `XSE_SHOCK` = 0x00000010, `XSE_TEMPERATURE` = 0x00000020, `XSE_TEMPERATURE_BACK_TO_NORMAL` = 0x00000021, `XSE_THERMAL_SHUTDOWN` = 0x00000022 }
- enum `XIS_Transmission_Mode` { `XTM_LIVE` = 0x00000001, `XTM_SAFE` = 0x00000002 }

Transmission modes.

- enum `XIS_Xrpd_Event` { `XXE_CONNECTION_LOST` = 0x00000001, `XXE_RECONNECTED` = 0x00000002 }
- 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_COMMUNICATION_ERROR` = 0x0001, `XRpad_TERMINATE_DISCHARGE_ALARM` = 0x0002, `XRpad_UNDERVOLTAGE_ALARM` = 0x0004, `XRpad_OVERVOLTAGE_ALARM` = 0x0008, `XRpad_OVERTEMPERATURE_ALARM` = 0x0010, `XRpad_BATTERY_UNKNOWN_ERROR` = 0x0080 }
- enum `XRpad_BatteryPresence` { `XRpad_NO_BATTERY` = 0, `XRpad_BATTERY_INSERTED` = 1, `XRpad_DUMMY_INSERTED` = 2, `XRpad_BATTERY_COM_ERR` = 3 }
- enum `XRpad_ChargeMode` { `XRpad_NOT_CHARGING` = 0, `XRpad_CHARGING_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_SLEEP` = 3, `XRpad_SYSTEM_CONTROL_SET_IDLE` = 4, `XRpad_SYSTEM_CONTROL_RESTART_WLAN` = 9 }

4.10.1 Define Documentation

4.10.1.1 #define HIS_ALL_OK 0

No error

4.10.1.2 #define HIS_ERROR_ABORT 46

Error during abort acquisition function.

4.10.1.3 #define HIS_ERROR_ABORTCURRFRAME 21

Aborting current frame failed.

4.10.1.4 #define HIS_ERROR_ACKNOWLEDGE_IMAGE 133

Error acknowledging the image.

4.10.1.5 #define HIS_ERROR_ACQ 43

Error starting the acquisition.

4.10.1.6 #define HIS_ERROR_ACQ_ALREADY_RUNNING 5

Acquisition is already running.

4.10.1.7 #define HIS_ERROR_ACQABORT 12

An unexpected acquisition abort occurred.

4.10.1.8 #define HIS_ERROR_ACQUISITION 13

error occurred during data acquisition.

4.10.1.9 #define HIS_ERROR_ALREADY_EXISTS 74

Error Invalid filename, file already exists.

4.10.1.10 #define HIS_ERROR_AVERAGED_LOST 49

The number of images for frame grabber onboard averaging must be 2 to the power of n.

4.10.1.11 #define HIS_ERROR_BAD_SORTING_PARAM 50

Parameter for (onboard) sorting not valid.

4.10.1.12 #define HIS_ERROR_BOARDINIT 2

Unable to initialize board.

4.10.1.13 #define HIS_ERROR_BUFFERSPACE_NOT_SUFF 29

Buffer space not sufficient.

4.10.1.14 #define HIS_ERROR_CONFLICT 167

The configuration/function call is conflicting with another option.

4.10.1.15 #define HIS_ERROR_CORRBUFFER_INCOMPATIBLE 4

Your correction files do not have a proper size.

4.10.1.16 #define HIS_ERROR_CREATE_MEMORYMAPPING 35

Error creating memory mapped file.

4.10.1.17 #define HIS_ERROR_CREATE_MUTEX 42

Could not create Mutex.

4.10.1.18 #define HIS_ERROR_CURL 65

Error CURL.

4.10.1.19 #define HIS_ERROR_DESC_NOT_LOCAL 44

Acquisition descriptor is not local.

4.10.1.20 #define HIS_ERROR_DOES_NOT_EXIST 75

Error Invalid filename type does not exist.

4.10.1.21 #define HIS_ERROR_EMI_NOT_SET 115

Error the requested EMI readout mode was not reported by the detector.

4.10.1.22 #define HIS_ERROR_ENABLE_INTERRUPTS 108

Error enabling XRPD interrupts.

4.10.1.23 `#define HIS_ERROR_ENABLE_ONBOARD_GAINOFFSET 68`

Error setting onboard gain corr mode.

4.10.1.24 `#define HIS_ERROR_ENABLE_ONBOARD_MEAN 67`

Error setting onboard mean corr mode.

4.10.1.25 `#define HIS_ERROR_ENABLE_ONBOARD_OFFSET 66`

Error setting onboard offset corr mode.

4.10.1.26 `#define HIS_ERROR_ENABLE_ONBOARD_PREVIEW 69`

Error setting onboard preview mode.

4.10.1.27 `#define HIS_ERROR_FRAME_INV 31`

Frame invalid.

4.10.1.28 `#define HIS_ERROR_FUNC_NOTIMPL 40`

Function is not implemented.

4.10.1.29 `#define HIS_ERROR_GET_AVAILABLE_SYSTEMS 154`

Could not retrieve available systems.

4.10.1.30 `#define HIS_ERROR_GET_CHARGE_MODE 139`

Error retrieving the requested charge mode from the detector.

4.10.1.31 `#define HIS_ERROR_GET_NUM_BOARDS 33`

Error during getting number of boards.

4.10.1.32 `#define HIS_ERROR_GET_ONBOARD_OFFSET 64`

Error getting onboard offset.

4.10.1.33 `#define HIS_ERROR_GETHWHEADERINFO 22`

Getting hardware header failed.

4.10.1.34 `#define HIS_ERROR_GETOSVERSION 16`

Getting version of operating system failed.

4.10.1.35 `#define HIS_ERROR_HEADER_TIMEOUT 56`

No frame header received from Detector.

4.10.1.36 `#define HIS_ERROR_HW_ALREADY_OPEN_BY_ANOTHER_PROCESS 34`

Communication channel already opened by another process.

4.10.1.37 `#define HIS_ERROR_HW_BOARD_CHANNEL_ALREADY_USED 146`

Requested channel is already openend.

4.10.1.38 `#define HIS_ERROR_HWHEADER_INV 23`

Hardware header is invalid.

4.10.1.39 `#define HIS_ERROR_ILLEGAL_INDEX 60`

Error Function called with an illegal index number.

4.10.1.40 `#define HIS_ERROR_INIT_DET_OPTIONS 176`

Error initializing the detector options data

4.10.1.41 `#define HIS_ERROR_INVALID_FILENAME 77`

Error Invalid filename for image tag or log file.

4.10.1.42 `#define HIS_ERROR_INVALID_FUNC_CALL 20`

Invalid function call.

4.10.1.43 `#define HIS_ERROR_INVALID_HANDLE 73`

Error Invalid SHOCKID.

4.10.1.44 `#define HIS_ERROR_INVALID_PARAM 45`

Invalid Parameter.

4.10.1.45 `#define HIS_ERROR_INVALIDACQDESC 7`

Acquisition descriptor invalid.

4.10.1.46 `#define HIS_ERROR_INVALIDBUFFERNR 72`

Error Invalid pointer/buffer passed as parameter.

4.10.1.47 `#define HIS_ERROR_LOAD_COORECTIONIMAGETOBUFFER 71`

Error Loading image from SD to onboard buffer.

4.10.1.48 `#define HIS_ERROR_LOADDRIVER 39`

Unable to load driver.

4.10.1.49 `#define HIS_ERROR_MEMORY 1`

Memory couldn't be allocated.

4.10.1.50 `#define HIS_ERROR_MEMORY_MAPPING 41`

Unable to create memory mapping.

4.10.1.51 `#define HIS_ERROR_MISSING_VERSION_INFORMATION 143`

Error not connected to on detector XRPD process.

4.10.1.52 `#define HIS_ERROR_NO_BOARD_IN_SUBNET 52`

Detector could not be found in the Subnet.

4.10.1.53 `#define HIS_ERROR_NO_FPGA_ACK` 57

Command not acknowledged.

4.10.1.54 `#define HIS_ERROR_NOCAMERA` 3

Got a time out. May be no detector present.

4.10.1.55 `#define HIS_ERROR_NODESC_AVAILABLE` 28

No acquisition descriptor available.

4.10.1.56 `#define HIS_ERROR_NOT_DISCOVERED` 62

Error No detectors discovered yet.

4.10.1.57 `#define HIS_ERROR_NOT_INITIALIZED` 61

Error Function or function environment not correctly initialised.

4.10.1.58 `#define HIS_ERROR_NOT_SUPPORTED` 165

Function is not supported by your device or setup.

4.10.1.59 `#define HIS_ERROR_NR_OF_BOARDS_CHANGED` 58

Number of boards within network changed during broadcast.

4.10.1.60 `#define HIS_ERROR_ONBOARD AVG FAILED` 63

Error onbaord averaging failed.

4.10.1.61 `#define HIS_ERROR_OPEN_FILE` 76

Error Invalid filename for image tag or log file.

4.10.1.62 `#define HIS_ERROR_READ_DATA` 26

Reading data failed.

4.10.1.63 #define HIS_ERROR_RESET_ZYNQ 172

Resetting the Zynq FPGA failed

4.10.1.64 #define HIS_ERROR_RETRIEVE_ENHANCED_HEADER 107

Error retrieving the enhanced header.

4.10.1.65 #define HIS_ERROR_SET_CHARGE_MODE 118

Error setting the software requested charge mode.

4.10.1.66 #define HIS_ERROR_SET_EVENT_CALLBACK 169

Unable to set event callback. Could be caused by multiple threads running the function in parallel.

4.10.1.67 #define HIS_ERROR_SET_IDLE_TIMEOUT 117

Error setting the on detector idle timeout.

4.10.1.68 #define HIS_ERROR_SET_IMAGE_TAG 104

Error setting the onboard image tag.

4.10.1.69 #define HIS_ERROR_SET_IMAGE_TAG_LENGTH 106

Error Image tag length exceeded 128char (including path: autosave/).

4.10.1.70 #define HIS_ERROR_SET_ONBOARD_BINNING 70

Error setting onboard binning mode.

4.10.1.71 #define HIS_ERROR_SET_PACKET_DELAY 153

Could not set on detector packet delay.

4.10.1.72 #define HIS_ERROR_SET_PROC_SCRIPT 105

Error setting the onboard process script.

4.10.1.73 `#define HIS_ERROR_SETBAUDRATE 27`

Setting baud rate failed.

4.10.1.74 `#define HIS_ERROR_SETCAMERAMODE 30`

Setting detector mode failed.

4.10.1.75 `#define HIS_ERROR_SETDISCOVERYTIMEOUT 78`

Error setting gbif discovery timeout.

4.10.1.76 `#define HIS_ERROR_SETEXAMFLAG 59`

Unable to set the exam flag.

4.10.1.77 `#define HIS_ERROR_SETFRMSYNC 17`

Can not set frame sync.

4.10.1.78 `#define HIS_ERROR_SETFRMSYNCMODE 18`

Can not set frame sync mode.

4.10.1.79 `#define HIS_ERROR_SETHEADERSIZE 160`

unable to set header size.

4.10.1.80 `#define HIS_ERROR_SETLINETRIG_MODE 24`

Setting line trigger mode failed.

4.10.1.81 `#define HIS_ERROR_SETREGISTERTIMEOUT 161`

unable to set register Timeout size.

4.10.1.82 `#define HIS_ERROR_SETTIMERSYNC 19`

Can not set timer sync.

4.10.1.83 `#define HIS_ERROR_SLOW_SYSTEM 32`

System too slow.

4.10.1.84 `#define HIS_ERROR_TIMEOUT 6`

Got a time out from hardware.

4.10.1.85 `#define HIS_ERROR_TRANSMISSION_MODE 166`

Failed to enable live transmission mode.

4.10.1.86 `#define HIS_ERROR_UNABLE_TO_ACCESS_DETECTOR_FLASH 55`

Unable to access the flash memory of Detector.

4.10.1.87 `#define HIS_ERROR_UNABLE_TO_CLOSE_BOARD 54`

Unable to close connection to Network Detector.

4.10.1.88 `#define HIS_ERROR_UNABLE_TO_OPEN_BOARD 53`

Unable to open connection to Network Detector.

4.10.1.89 `#define HIS_ERROR_UNKNOWN_IP_MAC_NAME 51`

Connection to Network Detector cannot be opened due to invalid IP address / MAC / Detector name.

4.10.1.90 `#define HIS_ERROR_VXD_REGISTER_DMA_ADDRESS 36`

Error registering DMA address.

4.10.1.91 `#define HIS_ERROR_VXD_REGISTER_IRQ 14`

Unable to register interrupt.

4.10.1.92 `#define HIS_ERROR_VXD_REGISTER_STAT_ADDR 37`

Error registering static address.

4.10.1.93 #define HIS_ERROR_VXD_REGISTER_STATADR 15

Register status address failed.

4.10.1.94 #define HIS_ERROR_VXD_UNMASK_IRQ 38

Unable to unmask interrupt.

4.10.1.95 #define HIS_ERROR_VXDGETDMAADR 11

VxD Error: GetDmaAddr failed.

4.10.1.96 #define HIS_ERROR_VXDNOTFOUND 8

Unable to find VxD.

4.10.1.97 #define HIS_ERROR_VXDNOTOPEN 9

Unable to open VxD.

4.10.1.98 #define HIS_ERROR_VXDUNKNOWNERROR 10

Unknown error during VxD loading.

4.10.1.99 #define HIS_ERROR_WLAN_RESTART 168

WLAN Restart failed.

4.10.1.100 #define HIS_ERROR_WRITE_DATA 25

Writing data failed.

4.10.1.101 #define HIS_ERROR_WRONG_CAMERA_MODE 48

Change of Detector Mode during Acquisition.

4.10.1.102 #define HIS_ERROR_WRONGBOARDSELECT 47

The wrong board is selected.

4.10.1.103 **#define HIS_ERROR_WSA 164**

WSAStartup or WSACleanup failed.

4.10.1.104 **#define HIS_ERROR_XRPD_BATTERY_COM 163**

unable to communicate with battery.

4.10.1.105 **#define HIS_ERROR_XRPD_CONNECT 134**

Error connecting to the on detector XRPD process.

4.10.1.106 **#define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT 112**

Error creating fake shock events.

4.10.1.107 **#define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_CRIT 119**

Error creating critical level fake shock events.

4.10.1.108 **#define HIS_ERROR_XRPD_CREATE_FAKE_SHOCK_EVENT_WARN 120**

Error creating warning level fake shock events.

4.10.1.109 **#define HIS_ERROR_XRPD_DETECTOR_IN_DEEP_SLEEP 157**

Function returned error since detector is in deep sleep and does not support the function in that state.

4.10.1.110 **#define HIS_ERROR_XRPD_DISABLE_SYSLOG_SAVING 152**

Unable to disable the saving of the syslog.

4.10.1.111 **#define HIS_ERROR_XRPD_FACTORY_RESET_SHOCK_EVENT 121**

Error resetting the shock events to factory values.

4.10.1.112 **#define HIS_ERROR_XRPD_GET_AUTOPOWERONLOCATIONS 137**

Error retrieving the auto power on locations from the detector.

4.10.1.113 **#define HIS_ERROR_XRPD_GET_CURRENT_VOLTAGE 147**

Error retrieving the Voltage or Current from detector.

4.10.1.114 **#define HIS_ERROR_XRPD_GET_DEEPSLEEPIDLELOCATIONS 155**

Error retrieving the deep-sleep idle change locations from the detector.

4.10.1.115 **#define HIS_ERROR_XRPD_GET_DEF_BOOT_CFG 170**

Error getting default boot configuration

4.10.1.116 **#define HIS_ERROR_XRPD_GET_DET_OPTIONS 175**

Error getting the detector options

4.10.1.117 **#define HIS_ERROR_XRPD_GET_DET_TYPE 174**

Error getting the detector type

4.10.1.118 **#define HIS_ERROR_XRPD_GET_EPC_REGISTER 159**

unable to get epc register.

4.10.1.119 **#define HIS_ERROR_XRPD_GET_SDCARD_INFO 113**

Error retrieving the sd card info.

4.10.1.120 **#define HIS_ERROR_XRPD_GET_SDCARD_TIMEOUT 142**

Error getting the sd card timeout from the detector.

4.10.1.121 **#define HIS_ERROR_XRPD_GET_TEMP_VALUES 158**

unable to get temp values from detector.

4.10.1.122 **#define HIS_ERROR_XRPD_GET_TEMPERATURE_THRESHOLDS 129**

Error getting the temperature thresholds from the detector.

4.10.1.123 **#define HIS_ERROR_XRPD_GET_WLAN_CC** 149

Unable to get on detector WLAN country code info.

4.10.1.124 **#define HIS_ERROR_XRPD_GET_WLAN_ChannelList** 151

Unable to get on detector WLAN channel list info.

4.10.1.125 **#define HIS_ERROR_XRPD_NO_EVENT_INTERFACE** 111

Error No interface to communicate event messages active.

4.10.1.126 **#define HIS_ERROR_XRPD_NO_EVENTCALLBACK_DEFINED** 130

Error no eventcallback defined for irq messages.

4.10.1.127 **#define HIS_ERROR_XRPD_NO_LOCATION** 116

Error retrieving the location info from the detector.

4.10.1.128 **#define HIS_ERROR_XRPD_NO_NETWORK** 122

Error getting the on detector LAN network speed.

4.10.1.129 **#define HIS_ERROR_XRPD_NOT_CONNECTED** 144

Error not connected to on detector XRPD process.

4.10.1.130 **#define HIS_ERROR_XRPD_REQUEST_POWERSTATE** 136

Error setting the power state on the detector.

4.10.1.131 **#define HIS_ERROR_XRPD_RESEND_ALL_MSG** 132

Error triggering the resend of all current messages by the XRPD.

4.10.1.132 **#define HIS_ERROR_XRPD_RESET_SHOCK** 135

Error resetting the shock event.

4.10.1.133 `#define HIS_ERROR_XRPD_RESET_TEMPERATURE_TIMEOUT` 127

Error resetting the temperature timeout counter.

4.10.1.134 `#define HIS_ERROR_XRPD_SDCARDPERFORMANCE` 145

Error retrieving the SD card performance.

4.10.1.135 `#define HIS_ERROR_XRPD_SESSION_ERROR` 109

Error XRPD session Error.

4.10.1.136 `#define HIS_ERROR_XRPD_SET_AUTOPOWERONLOCATIONS` 138

Error setting the auto power on locations on the detector.

4.10.1.137 `#define HIS_ERROR_XRPD_SET_CPUFREQ_GOVERNOR` 148

Unable to set on detector CPU governor.

4.10.1.138 `#define HIS_ERROR_XRPD_SET_DATE_TIME` 131

Error setting on detectors date and time.

4.10.1.139 `#define HIS_ERROR_XRPD_SET_DEEPSLEEPIDLELOCATIONS` 156

Error setting the deep-sleep idle change locations from the detector.

4.10.1.140 `#define HIS_ERROR_XRPD_SET_DEF_BOOT_CFG` 171

Error setting default boot configuration

4.10.1.141 `#define HIS_ERROR_XRPD_SET_EPC_REGISTER` 162

unable to set epc register.

4.10.1.142 `#define HIS_ERROR_XRPD_SET_EVENT` 110

Error No interface to communicate event messages active.

4.10.1.143 **#define HIS_ERROR_XRPD_SET_FORCE_FSCK 140**

Error requesting an fscheck on next boot.

4.10.1.144 **#define HIS_ERROR_XRPD_SET_NETWORK 123**

Error setting the on detector LAN network speed.

4.10.1.145 **#define HIS_ERROR_XRPD_SET_PRIVATE_KEY 125**

Error setting the private key for genuiness.

4.10.1.146 **#define HIS_ERROR_XRPD_SET_SDCARD_TIMEOUT 141**

Error setting the sd card timeout on the detector.

4.10.1.147 **#define HIS_ERROR_XRPD_SET_TEMP_FAKE_MODE 114**

Error activating the fake temperatur mode on the detector.

4.10.1.148 **#define HIS_ERROR_XRPD_SET_TEMPERATURE_THRESHOLDS 128**

Error setting the temperature thresholds on the detector.

4.10.1.149 **#define HIS_ERROR_XRPD_SET_TEMPERATURE_TIMEOUT 126**

Error setting the temperature timeout.

4.10.1.150 **#define HIS_ERROR_XRPD_SET_WLAN_CC 150**

Unable to get on detector WLAN country code.

4.10.1.151 **#define HIS_ERROR_XRPD_VERIFY_GENUINENESS 124**

Error verifying the private key for genuiness.

4.10.2 Typedef Documentation

4.10.2.1 **typedef UINT ACQDESCPOS**

4.10.2.2 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 HANDLE to the acquisition descriptor.

See also

AcquisitionDesc

4.10.2.3 typedef enum OnboardBinningMode OnboardBinningMode

4.10.2.4 typedef enum ProcScriptOperation ProcScriptOperation

enum used for addressing XRpad2 on board correction stages.

4.10.2.5 typedef enum XIS_Acquisition_Event XIS_Acquisition_Event

4.10.2.6 typedef enum XIS_Battery_Event XIS_Battery_Event

4.10.2.7 typedef enum XIS_Detector_Event XIS_Detector_Event

4.10.2.8 typedef enum XIS_Detector_TRIGOUT_SignalMode XIS_Detector_TRIGOUT_SignalMode

4.10.2.9 typedef enum XIS_DetectorTriggerMode XIS_DetectorTriggerMode

4.10.2.10 typedef enum XIS_Event XIS_Event

4.10.2.11 typedef enum XIS_FileType XIS_FileType

4.10.2.12 typedef enum XIS_Init_Flags XIS_Init_Flags

4.10.2.13 typedef enum XIS_Library_Event XIS_Library_Event

4.10.2.14 typedef enum XIS_Sensor_Event XIS_Sensor_Event

4.10.2.15 typedef enum XIS_Transmission_Mode XIS_Transmission_Mode

Transmission modes.

Note

Not all devices support each mode. Only XTM_LIVE is supported by all devices.

4.10.2.16 typedef enum XIS_Xrpd_Event XIS_Xrpd_Event

4.10.2.17 typedef enum XislFileEntryType XislFileEntryType

4.10.2.18 typedef void* XislFileHandle

4.10.2.19 typedef enum XislFileStorageLocation XislFileStorageLocation

4.10.2.20 typedef void* XislFtpSession

4.10.2.21 typedef enum XRpad_BatteryHealth XRpad_BatteryHealth

4.10.2.22 typedef enum XRpad_BatteryPresence XRpad_BatteryPresence

4.10.2.23 typedef struct XRpad_BatteryStatus XRpad_BatteryStatus

4.10.2.24 typedef enum XRpad_ChargeMode XRpad_ChargeMode

4.10.2.25 typedef enum XRpad_DataInterfaceControlEnum XRpad_DataInterfaceControlEnum

Possible image transfer channels for XRpad

4.10.2.26 typedef struct XRpad_ShockEvent XRpad_ShockEvent

4.10.2.27 typedef struct XRpad_ShockSensorReport XRpad_ShockSensorReport

4.10.2.28 typedef struct XRpad_TempSensor XRpad_TempSensor

4.10.2.29 typedef struct XRpad_TempSensorReport XRpad_TempSensorReport

4.10.2.30 typedef struct XRpad_VersionInfo XRpad_VersionInfo

4.10.3 Enumeration Type Documentation

4.10.3.1 enum OnboardBinningMode

Enumerator:

ONBOARDBINNING2x1

ONBOARDBINNING2x2

ONBOARDBINNING4x1

ONBOARDBINNING4x4

ONBOARDBINNING3x3

ONBOARDBINNING9to4

4.10.3.2 enum ProcScriptOperation

enum used for addressing XRpad2 on board correction stages.

Enumerator:

PREBINNING reserved. don not use!
PREMEAN reserved. don not use!
PRESTOREBUFFER reserved. don not use!
OFFSET offset correction
GAIN gain correction (do not use in dual enmergy mode!)
MEAN pixel correction
PREVIEW reserved. don not use!
BINNING reserved. don not use!
STOREBUFFER reserved. don not use!
STORESD reserved. don not use!
SEND reserved. don not use!
OFFSET_2 offset correction of second bright image in dual energy mode
GAIN_2 reserved. don not use!

4.10.3.3 enum XIS_Acquisition_Event

Enumerator:

XAE_TRIGOUT
XAE_READOUT
XAE_TRIGGERED
XAE_AED_READY

4.10.3.4 enum XIS_Battery_Event

Enumerator:

XBE_BATTERY_REPORT
XBE_BATTERY_WARNING

4.10.3.5 enum XIS_Detector_Event

Enumerator:

XDE_BUFFERS_IN_USE
XDE_STORED_IMAGE

XDE_DROPPED_IMAGE

XDE_POWER_STATE_CHANGED value 2 success: new power state

XDE_POWER_STATE_CHANGE_FAILED value 2 error: code XRPD_ERROR_SWITCH_PWR_STATE_BLOCKED_BY_EXAM 0x0052, any other: internal error

4.10.3.6 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

4.10.3.7 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

4.10.3.8 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
XE_XRPD_EVENT

4.10.3.9 enum XIS_FileType

Enumerator:

PKI_RESERVED
PKI_DOUBLE
PKI_SHORT
PKI_SIGNED
PKI_ERRORMAPONBOARD
PKI_LONG
PKI_SIGNEDSHORT
PKI_SIGNEDLONG
PKI_FAULTMASK

4.10.3.10 enum XIS_Init_Flags

Enumerator:

XIF_XISL Initialize libXISL.
XIF_GBIF Initialize libgbif.
XIF_WPE200 Initialize libwpe200 (currently not used)
XIF_CURL Initialize libcurl.
XIF_ALL Initialize everything.

4.10.3.11 enum XIS_Library_Event

Enumerator:

XLE_HIS_ERROR_PACKET_LOSS Frame is lost. not all network packages could be received.

4.10.3.12 enum XIS_Sensor_Event

Enumerator:

XSE_HALL
XSE_SHOCK
XSE_TEMPERATURE
XSE_TEMPERATURE_BACK_TO_NORMAL
XSE_THERMAL_SHUTDOWN

4.10.3.13 enum XIS_Transmission_Mode

Transmission modes.

Note

Not all devices support each mode. Only XTM_LIVE is supported by all devices.

Enumerator:

XTM_LIVE All images are sent out directly as they are acquired.
XTM_SAFE Images are sent out when the previous was completely received.

4.10.3.14 enum XIS_Xrpd_Event

Enumerator:

XXE_CONNECTION_LOST Connection to XRpad Daemon unexpectedly closed.
XXE_RECONNECTED Connection re-established automatically.

4.10.3.15 enum XisIFileEntryType

Enumerator:

XFT_File
XFT_Directory
XFT_Link
XFT_Other
XFT_Any

4.10.3.16 enum XisIFileStorageLocation

Enumerator:

XFSL_Local
XFSL_FTP

4.10.3.17 enum XislLoggingLevels

Enumerator:

LEVEL_TRACE
LEVEL_DEBUG
LEVEL_INFO
LEVEL_WARN
LEVEL_ERROR
LEVEL_FATAL
LEVEL_ALL
LEVEL_NONE

4.10.3.18 enum XRpad_BatteryHealth

Enumerator:

XRpad_BATTERY_OK All OK.
XRpad_COMMUNICATION_ERROR Communication error on SMBUS.
XRpad_TERMINATE_DISCHARGE_ALARM Battery is nearly empty but still discharging.
XRpad_UNDERVOLTAGE_ALARM Battery voltage is too low will be stopped.
XRpad_OVERVOLTAGE_ALARM Battery voltage is too high charging will be stopped.
XRpad_OVERTEMPERATURE_ALARM Battery temperature is too high charging will be stopped.
XRpad_BATTERY_UNKNOWN_ERROR Unknown error reported by battery controller.

4.10.3.19 enum XRpad_BatteryPresence

Enumerator:

XRpad_NO_BATTERY No battery detected.
XRpad_BATTERY_INSERTED Battery detected.
XRpad_DUMMY_INSERTED Dummy battery detected (unused)
XRpad_BATTERY_COM_ERR Communication error on SMBUS.

4.10.3.20 enum XRpad_ChargeMode

Enumerator:

XRpad_NOT_CHARGING Charging disabled.
XRpad_CHARGING_SLOW Charging slowly.
XRpad_CHARGING_NORMAL Charging normal.
XRpad_CHARGING_FAST Charging fast.
XRpad_FULLY_CHARGED Battery is fully charged.
XRpad_DISCHARGING Battery is discharging.

4.10.3.21 enum XRpad_DataInterfaceControlEnum

Possible image transfer channels for XRpad

Enumerator:

XRpad_DATA_VIA_LAN use LAN

XRpad_DATA_VIA_WLAN use WLAN

4.10.3.22 enum XRpad_SystemControlEnum

Possible system control actions for XRpad

Enumerator:

XRpad_SYSTEM_CONTROL_REBOOT restart XRpad

XRpad_SYSTEM_CONTROL_RESTART_NETWORK restart XRpad Network

XRpad_SYSTEM_CONTROL_SHUTDOWN shutdown XRpad

XRpad_SYSTEM_CONTROL_SET_DEEP_SLEEP power down analog circuitry and sensor FPGA

XRpad_SYSTEM_CONTROL_SET_IDLE power up analog circuitry and sensor FPGA

XRpad_SYSTEM_CONTROL_RESTART_WLAN restart XRpad2 WLAN only

Chapter 5

Class Documentation

5.1 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](#)

5.1.1 Member Data Documentation

5.1.1.1 int CHwHeaderInfo::bAddRow

5.1.1.2 BOOL CHwHeaderInfo::bAddRow

5.1.1.3 int CHwHeaderInfo::bPwrSave

5.1.1.4 BOOL CHwHeaderInfo::bPwrSave

5.1.1.5 int CHwHeaderInfo::bSyncMode

5.1.1.6 BOOL CHwHeaderInfo::bSyncMode

5.1.1.7 unsigned long CHwHeaderInfo::dwAccess

5.1.1.8 DWORD CHwHeaderInfo::dwAccess

5.1.1.9 unsigned long CHwHeaderInfo::dwAcqMode

5.1.1.10 DWORD CHwHeaderInfo::dwAcqMode

- 5.1.1.11 unsigned long CHwHeaderInfo::dwBias
- 5.1.1.12 DWORD CHwHeaderInfo::dwBias
- 5.1.1.13 unsigned long CHwHeaderInfo::dwDataSorting
- 5.1.1.14 DWORD CHwHeaderInfo::dwDataSorting
- 5.1.1.15 unsigned long CHwHeaderInfo::dwDataType
- 5.1.1.16 DWORD CHwHeaderInfo::dwDataType
- 5.1.1.17 unsigned long CHwHeaderInfo::dwFrmFillRowIntervalls
- 5.1.1.18 DWORD CHwHeaderInfo::dwFrmFillRowIntervalls
- 5.1.1.19 unsigned long CHwHeaderInfo::dwFrmNrRows
- 5.1.1.20 DWORD CHwHeaderInfo::dwFrmNrRows
- 5.1.1.21 unsigned long CHwHeaderInfo::dwFrmRowType
- 5.1.1.22 DWORD CHwHeaderInfo::dwFrmRowType
- 5.1.1.23 unsigned long CHwHeaderInfo::dwGain
- 5.1.1.24 DWORD CHwHeaderInfo::dwGain
- 5.1.1.25 unsigned long CHwHeaderInfo::dwHeaderID
- 5.1.1.26 DWORD CHwHeaderInfo::dwHeaderID
- 5.1.1.27 unsigned long CHwHeaderInfo::dwLeakRows
- 5.1.1.28 DWORD CHwHeaderInfo::dwLeakRows
- 5.1.1.29 unsigned long CHwHeaderInfo::dwNrColumns
- 5.1.1.30 DWORD CHwHeaderInfo::dwNrColumns
- 5.1.1.31 unsigned long CHwHeaderInfo::dwNrOfFillingRows
- 5.1.1.32 DWORD CHwHeaderInfo::dwNrOfFillingRows
- 5.1.1.33 unsigned long CHwHeaderInfo::dwNrRows
- 5.1.1.34 DWORD CHwHeaderInfo::dwNrRows

- 5.1.1.35 unsigned long CHwHeaderInfo::dwOffset
- 5.1.1.36 DWORD CHwHeaderInfo::dwOffset
- 5.1.1.37 unsigned long CHwHeaderInfo::dwPROMID
- 5.1.1.38 DWORD CHwHeaderInfo::dwPROMID
- 5.1.1.39 unsigned long CHwHeaderInfo::dwTiming
- 5.1.1.40 DWORD CHwHeaderInfo::dwTiming
- 5.1.1.41 unsigned long CHwHeaderInfo::dwZoomBRColumn
- 5.1.1.42 DWORD CHwHeaderInfo::dwZoomBRColumn
- 5.1.1.43 unsigned long CHwHeaderInfo::dwZoomBRRow
- 5.1.1.44 DWORD CHwHeaderInfo::dwZoomBRRow
- 5.1.1.45 unsigned long CHwHeaderInfo::dwZoomULColumn
- 5.1.1.46 DWORD CHwHeaderInfo::dwZoomULColumn
- 5.1.1.47 unsigned long CHwHeaderInfo::dwZoomULRow
- 5.1.1.48 DWORD CHwHeaderInfo::dwZoomULRow

5.2 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](#)

5.2.1 Member Data Documentation

5.2.1.1 unsigned short `CHwHeaderInfoEx::wAccess`

5.2.1.2 WORD `CHwHeaderInfoEx::wAccess`

5.2.1.3 unsigned short `CHwHeaderInfoEx::wBias`

5.2.1.4 WORD `CHwHeaderInfoEx::wBias`

5.2.1.5 unsigned short `CHwHeaderInfoEx::wBinningMode`

5.2.1.6 WORD `CHwHeaderInfoEx::wBinningMode`

5.2.1.7 unsigned short `CHwHeaderInfoEx::wCameratype`

5.2.1.8 WORD `CHwHeaderInfoEx::wCameratype`

5.2.1.9 unsigned short `CHwHeaderInfoEx::wClock`

5.2.1.10 WORD `CHwHeaderInfoEx::wClock`

5.2.1.11 unsigned short `CHwHeaderInfoEx::wCommand1`

5.2.1.12 WORD `CHwHeaderInfoEx::wCommand1`

5.2.1.13 unsigned short `CHwHeaderInfoEx::wCommand2`

5.2.1.14 WORD `CHwHeaderInfoEx::wCommand2`

5.2.1.15 unsigned short `CHwHeaderInfoEx::wCommand3`

5.2.1.16 WORD `CHwHeaderInfoEx::wCommand3`

5.2.1.17 unsigned short `CHwHeaderInfoEx::wCommand4`

5.2.1.18 WORD `CHwHeaderInfoEx::wCommand4`

5.2.1.19 unsigned short `CHwHeaderInfoEx::wDataSorting`

5.2.1.20 WORD `CHwHeaderInfoEx::wDataSorting`

- 5.2.1.21 unsigned short CHwHeaderInfoEx::wDummy
- 5.2.1.22 WORD CHwHeaderInfoEx::wDummy
- 5.2.1.23 unsigned short CHwHeaderInfoEx::wFrameCnt
- 5.2.1.24 WORD CHwHeaderInfoEx::wFrameCnt
- 5.2.1.25 unsigned short CHwHeaderInfoEx::wFrmNrRows
- 5.2.1.26 WORD CHwHeaderInfoEx::wFrmNrRows
- 5.2.1.27 unsigned short CHwHeaderInfoEx::wFrmRowType
- 5.2.1.28 WORD CHwHeaderInfoEx::wFrmRowType
- 5.2.1.29 unsigned short CHwHeaderInfoEx::wGain
- 5.2.1.30 WORD CHwHeaderInfoEx::wGain
- 5.2.1.31 unsigned short CHwHeaderInfoEx::wHeaderID
- 5.2.1.32 WORD CHwHeaderInfoEx::wHeaderID
- 5.2.1.33 unsigned short CHwHeaderInfoEx::wLeakRows
- 5.2.1.34 WORD CHwHeaderInfoEx::wLeakRows
- 5.2.1.35 unsigned short CHwHeaderInfoEx::wNrColumns
- 5.2.1.36 WORD CHwHeaderInfoEx::wNrColumns
- 5.2.1.37 unsigned short CHwHeaderInfoEx::wNrRows
- 5.2.1.38 WORD CHwHeaderInfoEx::wNrRows
- 5.2.1.39 unsigned short CHwHeaderInfoEx::wPROMID
- 5.2.1.40 WORD CHwHeaderInfoEx::wPROMID
- 5.2.1.41 unsigned short CHwHeaderInfoEx::wRealInttime_microSec
- 5.2.1.42 WORD CHwHeaderInfoEx::wRealInttime_microSec
- 5.2.1.43 unsigned short CHwHeaderInfoEx::wRealInttime_milliSec
- 5.2.1.44 WORD CHwHeaderInfoEx::wRealInttime_milliSec

- 5.2.1.45 unsigned short CHwHeaderInfoEx::wResolutionX
- 5.2.1.46 WORD CHwHeaderInfoEx::wResolutionX
- 5.2.1.47 unsigned short CHwHeaderInfoEx::wResolutionY
- 5.2.1.48 WORD CHwHeaderInfoEx::wResolutionY
- 5.2.1.49 unsigned short CHwHeaderInfoEx::wRowTime
- 5.2.1.50 WORD CHwHeaderInfoEx::wRowTime
- 5.2.1.51 unsigned short CHwHeaderInfoEx::wStatus
- 5.2.1.52 WORD CHwHeaderInfoEx::wStatus
- 5.2.1.53 unsigned short CHwHeaderInfoEx::wTiming
- 5.2.1.54 WORD CHwHeaderInfoEx::wTiming
- 5.2.1.55 unsigned short CHwHeaderInfoEx::wUgComp
- 5.2.1.56 WORD CHwHeaderInfoEx::wUgComp
- 5.2.1.57 unsigned short CHwHeaderInfoEx::wZoomBRColumn
- 5.2.1.58 WORD CHwHeaderInfoEx::wZoomBRColumn
- 5.2.1.59 unsigned short CHwHeaderInfoEx::wZoomBRRow
- 5.2.1.60 WORD CHwHeaderInfoEx::wZoomBRRow
- 5.2.1.61 unsigned short CHwHeaderInfoEx::wZoomULColumn
- 5.2.1.62 WORD CHwHeaderInfoEx::wZoomULColumn
- 5.2.1.63 unsigned short CHwHeaderInfoEx::wZoomULRow
- 5.2.1.64 WORD CHwHeaderInfoEx::wZoomULRow

5.3 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](#)

5.3.1 Member Data Documentation

5.3.1.1 DWORD DETECTOR_BATTERY::capacity

5.3.1.2 DWORD DETECTOR_BATTERY::charge

5.3.1.3 DWORD DETECTOR_BATTERY::current

5.3.1.4 DWORD DETECTOR_BATTERY::cycle_count

5.3.1.5 DWORD DETECTOR_BATTERY::energy

5.3.1.6 DWORD DETECTOR_BATTERY::serial_no

5.3.1.7 DWORD DETECTOR_BATTERY::status

5.3.1.8 DWORD DETECTOR_BATTERY::temperature

5.3.1.9 DWORD DETECTOR_BATTERY::voltage

5.4 DETECTOR_CURRENT_VOLTAGE Struct Reference

Public Attributes

- int [imA1](#)
- int [imA2](#)
- int [imA3](#)
- int [iV1](#)
- int [iV2](#)
- int [iV3](#)

5.4.1 Member Data Documentation

5.4.1.1 int DETECTOR_CURRENT_VOLTAGE::imA1

5.4.1.2 int DETECTOR_CURRENT_VOLTAGE::imA2

5.4.1.3 int DETECTOR_CURRENT_VOLTAGE::imA3

5.4.1.4 int DETECTOR_CURRENT_VOLTAGE::iV1

5.4.1.5 int DETECTOR_CURRENT_VOLTAGE::iV2

5.4.1.6 int DETECTOR_CURRENT_VOLTAGE::iV3

5.5 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.

5.5.1 Detailed Description

The device info

5.5.2 Member Data Documentation

5.5.2.1 char deviceInfo::device_version[16]

The device version.

5.5.2.2 char deviceInfo::manufacturer_name[32]

The manufactures name.

5.5.2.3 char deviceInfo::manufacturer_specific[48]

Some manufacture info.

5.5.2.4 char deviceInfo::model_name[32]

The model name.

5.5.2.5 char deviceInfo::serial_number[16]

The serial number.

5.5.2.6 char deviceInfo::spec_version[16]

GigE vision spec version.

5.5.2.7 char deviceInfo::user_name[16]

Device specific.

5.6 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.

5.6.1 Detailed Description

The original discovery reply

5.6.2 Member Data Documentation

5.6.2.1 struct deviceInfo discoveryReply::deviceInfo

The device information.

5.6.2.2 char discoveryReply::gvcp_ip[16]

Which IP address is used for image transfer.

5.6.2.3 struct networkInfo discoveryReply::lanInfo

The LAN network setup.

5.6.2.4 struct networkInfo discoveryReply::wlanInfo

The WLAN network setup.

5.7 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.

5.7.1 Detailed Description

The extended discovery reply Can carry additional information

5.7.2 Member Data Documentation

5.7.2.1 struct deviceInfo discoveryReplyEx::deviceInfo

The device information.

5.7.2.2 char discoveryReplyEx::gvcp_ip[16]

Which IP address is used for image transfer.

5.7.2.3 struct `networkInfo` `discoveryReplyEx::lanInfo`

The LAN network setup.

5.7.2.4 unsigned `discoveryReplyEx::messageCount`

How many messages this reply carries.

5.7.2.5 struct `discoveryReplyMsg` `discoveryReplyEx::messages[32]`

Info about the received reply packets.

5.7.2.6 struct `networkInfo` `discoveryReplyEx::wlanInfo`

The WLAN network setup.

5.8 `discoveryReplyMsg` Struct Reference

Public Attributes

- char [local_ip](#) [16]
- char [remote_ip](#) [16]

5.8.1 Detailed Description

Information about a single discovery Reply network packet

5.8.2 Member Data Documentation

5.8.2.1 char `discoveryReplyMsg::local_ip[16]`

5.8.2.2 char `discoveryReplyMsg::remote_ip[16]`

5.9 `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](#)

5.9.1 Member Data Documentation

5.9.1.1 DWORD [EPC_REGISTER::active_network_config](#)

5.9.1.2 [DETECTOR_BATTERY](#) [EPC_REGISTER::battery](#)

5.9.1.3 DWORD [EPC_REGISTER::channel](#)

5.9.1.4 DWORD [EPC_REGISTER::exam_flag](#)

5.9.1.5 DWORD [EPC_REGISTER::lan_status_register](#)

5.9.1.6 DWORD [EPC_REGISTER::power_state](#)

5.9.1.7 [RTC_STRUCT](#) [EPC_REGISTER::rtc_value](#)

5.9.1.8 DWORD [EPC_REGISTER::sdcard_state](#)

5.9.1.9 DWORD [EPC_REGISTER::sdcard_usage](#)

5.9.1.10 DWORD [EPC_REGISTER::signal_strength](#)

5.9.1.11 DWORD [EPC_REGISTER::spartan_id](#)

5.9.1.12 [CHwHeaderInfoEx](#) [EPC_REGISTER::spartan_register](#)

5.9.1.13 DWORD [EPC_REGISTER::temperature_error_level](#)[8]

5.9.1.14 DWORD [EPC_REGISTER::temperature_value](#)[8]

5.9.1.15 DWORD [EPC_REGISTER::temperature_warning_level](#)[8]

5.9.1.16 `DWORD EPC_REGISTER::version`

5.9.1.17 `DWORD EPC_REGISTER::wlan_status_register`

5.10 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](#)

5.10.1 Member Data Documentation

5.10.1.1 unsigned char `FPGAType::wTiming`

5.10.1.2 unsigned char `FPGAType::wValue0`

5.10.1.3 unsigned char `FPGAType::wValue1`

5.10.1.4 unsigned char `FPGAType::wValue2`

5.10.1.5 unsigned char `FPGAType::wValue3`

5.10.1.6 unsigned char `FPGAType::wValue4`

5.10.1.7 unsigned char `FPGAType::wValue5`

5.10.1.8 unsigned char `FPGAType::wValue6`

5.11 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]

5.11.1 Member Data Documentation

5.11.1.1 char GBIF_Detector_Properties::cDetectorType

5.11.1.2 char GBIF_Detector_Properties::cDeviceIdentifier[16]

5.11.1.3 char GBIF_Detector_Properties::cDummy

5.11.1.4 char GBIF_Detector_Properties::cManufacturingDate

5.11.1.5 char GBIF_Detector_Properties::cPlaceOfManufacture

5.11.1.6 char GBIF_Detector_Properties::cUniqueDeviceIdentifier[16]

5.12 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]

5.12.1 Member Data Documentation

5.12.1.1 char GBIF_DEVICE_PARAM::cDeviceName[16]

5.12.1.2 CHAR GBIF_DEVICE_PARAM::cDeviceName[16]

5.12.1.3 char GBIF_DEVICE_PARAM::cGBIFFirmwareVersion[32]

5.12.1.4 CHAR GBIF_DEVICE_PARAM::cGBIFFirmwareVersion[32]

- 5.12.1.5 char GBIF_DEVICE_PARAM::cManufacturerName[32]
- 5.12.1.6 CHAR GBIF_DEVICE_PARAM::cManufacturerName[32]
- 5.12.1.7 char GBIF_DEVICE_PARAM::cModelName[32]
- 5.12.1.8 CHAR GBIF_DEVICE_PARAM::cModelName[32]
- 5.12.1.9 unsigned long GBIF_DEVICE_PARAM::dwIPCurrentBootOptions
- 5.12.1.10 DWORD GBIF_DEVICE_PARAM::dwIPCurrentBootOptions
- 5.12.1.11 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucAdapterIP
- 5.12.1.12 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucAdapterMask
- 5.12.1.13 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucGateway
- 5.12.1.14 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucIP
- 5.12.1.15 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucMacAddress
- 5.12.1.16 GBIF_STRING_DATATYPE GBIF_DEVICE_PARAM::ucSubnetMask

5.13 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

5.13.1 Detailed Description

Structure for holding the adapter part of a configuration

5.13.2 Member Data Documentation

5.13.2.1 int `networkAdapterConfiguration::bridged`

Is the device in a bridge, this is not changeable by API.

5.13.2.2 char `networkAdapterConfiguration::dns[16]`

DNS server.

5.13.2.3 int `networkAdapterConfiguration::enabled`

Enabled flag, this is not changeable for LAN by API, but changeable for WLAN.

5.13.2.4 char `networkAdapterConfiguration::gateway[16]`

Gateway.

5.13.2.5 int `networkAdapterConfiguration::hw_accel`

Used image transfer, this is not changeable by API.

5.13.2.6 char `networkAdapterConfiguration::ifname[16]`

Interface name (eth0), this is not changeable by API.

5.13.2.7 char `networkAdapterConfiguration::ipaddr[16]`

IP address.

5.13.2.8 char networkAdapterConfiguration::macaddr[18]

MAC address, this is not changeable by API.

5.13.2.9 char networkAdapterConfiguration::netmask[16]

Netmask.

5.13.2.10 char networkAdapterConfiguration::not_used[110]

To fill up the struct to 320 byte.

5.13.2.11 char networkAdapterConfiguration::proto[16]

"static" or "dhcp", only these two options are available

5.14 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.

5.14.1 Detailed Description

Structure for holding the complete network configuration.

5.14.2 Member Data Documentation

5.14.2.1 `int networkConfiguration::gbif_enabled`

Is the GBif enabled.

5.14.2.2 `char networkConfiguration::hostname[80]`

The hostname.

5.14.2.3 `struct networkAdapterConfiguration networkConfiguration::lan`

LAN.

5.14.2.4 `char networkConfiguration::name[80]`

The configuration name.

5.14.2.5 `char networkConfiguration::notUsed[184]`

For later extensions.

5.14.2.6 `char networkConfiguration::path[128]`

The configurations path, this is not changeable by API.

5.14.2.7 `int networkConfiguration::readonly`

Is the configuration readonly, this is not changeable by API.

5.14.2.8 `int networkConfiguration::sshd_enabled`

SSH daemon enabled.

5.14.2.9 `struct wifiConfigurationEx networkConfiguration::wifi`

Wifi configuration, extended version.

5.14.2.10 struct networkAdapterConfiguration networkConfiguration::wlan

WLAN.

5.15 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.

5.15.1 Detailed Description

The network information

5.15.2 Member Data Documentation

5.15.2.1 char networkInfo::broadcast[16]

The IP broadcast as string.

5.15.2.2 char networkInfo::ip[16]

The IP (v4) address as string.

5.15.2.3 char networkInfo::mac[18]

The MAC address as string.

5.15.2.4 char networkInfo::mask[16]

The IP mask as string.

5.16 RTC_STRUCT Struct Reference

Public Attributes

- DWORD [day](#)
- DWORD [hour](#)
- DWORD [minute](#)
- DWORD [month](#)
- DWORD [second](#)
- DWORD [year](#)

5.16.1 Member Data Documentation

5.16.1.1 DWORD RTC_STRUCT::day

5.16.1.2 DWORD RTC_STRUCT::hour

5.16.1.3 DWORD RTC_STRUCT::minute

5.16.1.4 DWORD RTC_STRUCT::month

5.16.1.5 DWORD RTC_STRUCT::second

5.16.1.6 DWORD RTC_STRUCT::year

5.17 wifiConfiguration Struct Reference

Public Attributes

- char [agmode](#) [32]
agmode
- int [channel](#)
Channel, only valid options will be accepted, use 0 for automatically select channel, otherwise will return with error.
- 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.

5.17.1 Detailed Description

Wifi configurations

5.17.2 Member Data Documentation

5.17.2.1 char wifiConfiguration::agmode[32]

agmode

5.17.2.2 int wifiConfiguration::channel

Channel, only valid options will be accepted, use 0 for automatically select channel, otherwise will return with error.

5.17.2.3 char wifiConfiguration::description[64]

Contains the description in case of a station.

5.17.2.4 char wifiConfiguration::mode[32]

Accesspoint or client.

5.17.2.5 char wifiConfiguration::ssid[64]

Own SSID if mode == "ap" or the accesspoints ssid.

5.18 wifiConfigurationEx Struct Reference

Public Attributes

- char [agmode](#) [32]
agmode
- int [channel](#)
Channel, only valid options will be accepted, use 0 for automatically select channel, 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.

5.18.1 Detailed Description

Wifi configurations, extended version

5.18.2 Member Data Documentation

5.18.2.1 char wifiConfigurationEx::agmode[32]

agmode

5.18.2.2 int wifiConfigurationEx::channel

Channel, only valid options will be accepted, use 0 for automatically select channel, otherwise will return with error.

5.18.2.3 char wifiConfigurationEx::description[64]

Contains the description in case of a station.

5.18.2.4 char wifiConfigurationEx::mode[32]

Accesspoint or client.

5.18.2.5 char wifiConfigurationEx::passphrase[68]

new Passphrase (station or accesspoint) (may be 64 byte)

5.18.2.6 int wifiConfigurationEx::scan_ssid

only of station mode: scan the SSID

5.18.2.7 char wifiConfigurationEx::ssid[64]

Own SSID if mode == "ap" or the accesspoints ssid.

5.19 WinHeaderType Struct Reference

Public Attributes

- unsigned short [BRX](#)
- WORD [BRX](#)

- unsigned short [BRY](#)
- WORD [BRY](#)
- Bounding rectangle of the image.*
- unsigned short [Correction](#)
- WORD [Correction](#)
- 0 = none, 1 = offset, 2 = gain, 4 = bad pixel, (ored) can be 0*
- unsigned long [FileSize](#)
- UINT [FileSize](#)
- Size of the whole file in Bytes (HeaderSize+ImageHeaderSize+Frames*rows*columns*dattypesize)*
- unsigned short [FileType](#)
- WORD [FileType](#)
- File ID (0x7000)*
- unsigned short [HeaderSize](#)
- WORD [HeaderSize](#)
- Size of this file header in Bytes.*
- unsigned short [HeaderVersion](#)
- WORD [HeaderVersion](#)
- Must be 100, default XIS file header not to be used for onboard.*
- unsigned short [ImageHeaderSize](#)
- WORD [ImageHeaderSize](#)
- Size of the Image header in Bytes 32 Bytes, values can be all zero.*
- double [IntegrationTime](#)
- Frame time in microseconds can by 0.*
- unsigned short [NrOfFrames](#)
- WORD [NrOfFrames](#)
- Nr of Frames in seq.*
- unsigned short [TypeOfNumbers](#)
- WORD [TypeOfNumbers](#)
- Refer to enum XIS_FileType.*
- unsigned short [ULX](#)
- WORD [ULX](#)
- unsigned short [ULY](#)
- WORD [ULY](#)
- unsigned char [x](#) [WINRESTSIZE]
- BYTE [x](#) [WINRESTSIZE]
- Fill up this struct to have a size of 68 bytes (FileHeaderSize)*

5.19.1 Member Data Documentation

5.19.1.1 unsigned short WinHeaderType::BRX

5.19.1.2 WORD WinHeaderType::BRX

5.19.1.3 unsigned short WinHeaderType::BRY

5.19.1.4 WORD WinHeaderType::BRY

Bounding rectangle of the image.

5.19.1.5 unsigned short WinHeaderType::Correction

5.19.1.6 WORD WinHeaderType::Correction

0 = none, 1 = offset, 2 = gain, 4 = bad pixel, (ored) can be 0

5.19.1.7 unsigned long WinHeaderType::FileSize

5.19.1.8 UINT WinHeaderType::FileSize

Size of the whole file in Bytes (HeaderSize+ImageHeaderSize+Frames*rows*columns*datatypesize)

5.19.1.9 unsigned short WinHeaderType::FileType

5.19.1.10 WORD WinHeaderType::FileType

File ID (0x7000)

5.19.1.11 unsigned short WinHeaderType::HeaderSize

5.19.1.12 WORD WinHeaderType::HeaderSize

Size of this file header in Bytes.

5.19.1.13 unsigned short WinHeaderType::HeaderVersion

5.19.1.14 WORD WinHeaderType::HeaderVersion

Must be 100, default XIS file header not to be used for onboard.

5.19.1.15 unsigned short WinHeaderType::ImageHeaderSize

5.19.1.16 WORD WinHeaderType::ImageHeaderSize

Size of the Image header in Bytes 32 Bytes, values can be all zero.

5.19.1.17 double WinHeaderType::IntegrationTime

Frame time in microseconds can by 0.

5.19.1.18 unsigned short WinHeaderType::NrOfFrames

5.19.1.19 WORD WinHeaderType::NrOfFrames

Nr of Frames in seq.

5.19.1.20 unsigned short WinHeaderType::TypeOfNumbers

5.19.1.21 WORD WinHeaderType::TypeOfNumbers

Refer to enum XIS_FileType.

5.19.1.22 unsigned short WinHeaderType::ULX

5.19.1.23 WORD WinHeaderType::ULX

5.19.1.24 unsigned short WinHeaderType::ULY

5.19.1.25 WORD WinHeaderType::ULY

5.19.1.26 unsigned char WinHeaderType::x[WINRESTSIZE]

5.19.1.27 BYTE WinHeaderType::x[WINRESTSIZE]

Fill up this struct to have a size of 68 bytes (FileHeaderSize)

5.20 WinHeaderType101 Struct Reference

Public Attributes

- WORD [BRX](#)
- WORD [BRY](#)
 - Bounding rectangle of the image.*
- WORD [Correction](#)
 - 0 = none, 1 = offset, 2 = gain, 4 = bad pixel, (ored) can be 0*
- UINT [FileSize](#)
 - Size of the whole file in Bytes (HeaderSize+ImageHeaderSize+Frames*rows*columns*dattypesize)*
- WORD [FileType](#)
 - File ID (0x7000)*

- WORD [HeaderSize](#)
Size of this file header in Bytes.
- WORD [HeaderVersion](#)
Must be 101 for Onboard file Header used by XRpad[2] using this file type for onboard corrections.
- WORD [ImageHeaderSize](#)
Size of the Image header in Bytes 2048 Bytes, values can be all zero.
- double [IntegrationTime](#)
Frame time in microseconds.
- WORD [NrOfFrames](#)
Nr of Frames in seq.
- WORD [TypeOfNumbers](#)
Refer to enum `XIS_FileType` must be `PKI_ERRORMAPONBOARD` (1bit per pixel images filled up to full byte) or `PKI_SHORT` (2Bytes per Pixel) for onboard usage.
- WORD [ULX](#)
- WORD [ULY](#)
- WORD [wMedianValue](#)
Median of the image / shall be 0. use 0 for onboard corrections. median for gain corr will be automatically calculated.
- BYTE [x](#) [WINRESTSIZE101]
Fill up this struct to have a size of 68 bytes (FileHeaderSize)

5.20.1 Member Data Documentation

5.20.1.1 WORD WinHeaderType101::BRX

5.20.1.2 WORD WinHeaderType101::BRY

Bounding rectangle of the image.

5.20.1.3 WORD WinHeaderType101::Correction

0 = none, 1 = offset, 2 = gain, 4 = bad pixel, (ored) can be 0

5.20.1.4 UINT WinHeaderType101::FileSize

Size of the whole file in Bytes (HeaderSize+ImageHeaderSize+Frames*rows*columns*datatypesize)

5.20.1.5 WORD WinHeaderType101::FileType

File ID (0x7000)

5.20.1.6 WORD WinHeaderType101::HeaderSize

Size of this file header in Bytes.

5.20.1.7 WORD WinHeaderType101::HeaderVersion

Must be 101 for Onboard file Header used by XRpad[2] using this file type for onboard corrections.

5.20.1.8 WORD WinHeaderType101::ImageHeaderSize

Size of the Image header in Bytes 2048 Bytes, values can be all zero.

5.20.1.9 double WinHeaderType101::IntegrationTime

Frame time in microseconds.

5.20.1.10 WORD WinHeaderType101::NrOfFrames

Nr of Frames in seq.

5.20.1.11 WORD WinHeaderType101::TypeOfNumbers

Refer to enum XIS_FileType must be PKI_ERRORMAPONBOARD (1bit per pixel images filled up to full byte) or PKI_SHORT (2Bytes per Pixel) for onboard usage.

5.20.1.12 WORD WinHeaderType101::ULX**5.20.1.13 WORD WinHeaderType101::ULY****5.20.1.14 WORD WinHeaderType101::wMedianValue**

Median of the image / shall be 0. use 0 for onboard corrections. median for gain corr will be automatically calculated.

5.20.1.15 BYTE WinHeaderType101::x[WINRESTSIZE101]

Fill up this struct to have a size of 68 bytes (FileHeaderSize)

5.21 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]

5.21.1 Member Data Documentation

5.21.1.1 unsigned long WinImageHeaderType::dwPROMID

5.21.1.2 DWORD WinImageHeaderType::dwPROMID

5.21.1.3 float WinImageHeaderType::fAmpere

5.21.1.4 float WinImageHeaderType::fKVolt

5.21.1.5 unsigned short WinImageHeaderType::n_avframes

5.21.1.6 WORD WinImageHeaderType::n_avframes

5.21.1.7 char WinImageHeaderType::strPrefilter

5.21.1.8 char WinImageHeaderType::strProject

5.21.1.9 char WinImageHeaderType::strSystemused

5.22 XRpad_BatteryStatus Struct Reference

Public Attributes

- int [authenticated](#)
Retrieves the information if an OEM battery is used.
- [XRpad_ChargeMode](#) [charge_mode](#)
Retrieves the current charge mode (XRpad_ChargeMode).
- int [charge_state](#)
Retrieves the information about the charge status in percent.
- int [cycle_count](#)
Retrieves number of charging cycles.
- int [design_capacity](#)

Displays the information of the original capacity of this battery.

- int [health](#)

Displays the overall status of the battery which is an ored combination of the XRpad_BatteryHealth enum values provided by the battery controller.

- [XRpad_BatteryPresence](#) [presence](#)

Provides the information if a battery is present (XRpad_BatteryPresence).

- int [remaining_capacity](#)

Retrieves the information about the remaining charge in mA.

- int [temperature](#)

Retrieves the temperature of the battery.

5.22.1 Member Data Documentation

5.22.1.1 int XRpad_BatteryStatus::authenticated

Retrieves the information if an OEM battery is used.

5.22.1.2 XRpad_ChargeMode XRpad_BatteryStatus::charge_mode

Retrieves the current charge mode (XRpad_ChargeMode).

5.22.1.3 int XRpad_BatteryStatus::charge_state

Retrieves the information about the charge status in percent.

5.22.1.4 int XRpad_BatteryStatus::cycle_count

Retrieves number of charging cycles.

5.22.1.5 int XRpad_BatteryStatus::design_capacity

Displays the information of the original capacity of this battery.

5.22.1.6 int XRpad_BatteryStatus::health

Displays the overall status of the battery which is an ored combination of the XRpad_BatteryHealth enum values provided by the battery controller.

5.22.1.7 XRpad_BatteryPresence XRpad_BatteryStatus::presence

Provides the information if a battery is present (XRpad_BatteryPresence).

5.22.1.8 int XRpad_BatteryStatus::remaining_capacity

Retrieves the information about the remaining charge in mA.

5.22.1.9 int XRpad_BatteryStatus::temperature

Retrieves the temperature of the battery.

5.23 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](#)

5.23.1 Member Data Documentation

5.23.1.1 unsigned int XRpad_ShockEvent::critical_sensor1

5.23.1.2 unsigned int XRpad_ShockEvent::critical_sensor2

5.23.1.3 unsigned int XRpad_ShockEvent::critical_sensor3

5.23.1.4 unsigned int XRpad_ShockEvent::timestamp

5.23.1.5 unsigned int XRpad_ShockEvent::warning_sensor1

5.23.1.6 unsigned int XRpad_ShockEvent::warning_sensor2

5.23.1.7 unsigned int XRpad_ShockEvent::warning_sensor3

5.24 XRpad_ShockSensorReport Struct Reference

Public Attributes

- [XRpad_ShockEvent](#) largest
- [XRpad_ShockEvent](#) latest

5.24.1 Member Data Documentation

5.24.1.1 `XRpad_ShockEvent XRpad_ShockSensorReport::largest`

5.24.1.2 `XRpad_ShockEvent XRpad_ShockSensorReport::latest`

5.25 XRpad_TempSensor Struct Reference

Public Attributes

- char [index](#)
- BOOL [is_virtual](#)
- char [name](#) [32]
- double [temperature](#)
- unsigned char [warn_level](#)

5.25.1 Member Data Documentation

5.25.1.1 `char XRpad_TempSensor::index`

5.25.1.2 `BOOL XRpad_TempSensor::is_virtual`

5.25.1.3 `char XRpad_TempSensor::name[32]`

5.25.1.4 `double XRpad_TempSensor::temperature`

5.25.1.5 `unsigned char XRpad_TempSensor::warn_level`

5.26 XRpad_TempSensorReport Struct Reference

Public Attributes

- unsigned char [sensor_count](#)
- [XRpad_TempSensor](#) [sensors](#) [16]
- unsigned int [shutdown_time](#)
- unsigned char [system_warn_level](#)

5.26.1 Member Data Documentation

5.26.1.1 `unsigned char XRpad_TempSensorReport::sensor_count`

5.26.1.2 `XRpad_TempSensor XRpad_TempSensorReport::sensors[16]`

5.26.1.3 `unsigned int XRpad_TempSensorReport::shutdown_time`

5.26.1.4 unsigned char XRpad_TempSensorReport::system_warn_level

5.27 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]

5.27.1 Member Data Documentation

5.27.1.1 char XRpad_VersionInfo::hwdriver[32]

5.27.1.2 char XRpad_VersionInfo::linux_kernel[32]

5.27.1.3 char XRpad_VersionInfo::msp_firmware[32]

5.27.1.4 char XRpad_VersionInfo::pld_firmware[32]

5.27.1.5 char XRpad_VersionInfo::software[32]

5.27.1.6 char XRpad_VersionInfo::spartan_firmware[32]

5.27.1.7 char XRpad_VersionInfo::subversion[256]

5.27.1.8 char XRpad_VersionInfo::wlan[32]

5.27.1.9 char XRpad_VersionInfo::xrpd[32]

5.27.1.10 char XRpad_VersionInfo::zynq_firmware[32]

5.28 XrpdVariant Union Reference

Public Attributes

- const void * [const_ptr](#)
- double [dbl](#)

- float [flt](#)
- void * [ptr](#)
- uint64_t [u64](#)

5.28.1 Member Data Documentation

5.28.1.1 `const void*` `XrpdVariant::const_ptr`

5.28.1.2 `double` `XrpdVariant::dbl`

5.28.1.3 `float` `XrpdVariant::flt`

5.28.1.4 `void*` `XrpdVariant::ptr`

5.28.1.5 `uint64_t` `XrpdVariant::u64`