

XIS Reference Book

X-Ray Imaging Software

Version 3.2



Table of Contents

1	General.....	8
1.1	About the program XIS	8
1.2	Liability policy.....	8
1.3	Copyright.....	8
1.4	Getting Started - The first image.....	9
1.4.1	Introduction.....	9
1.4.2	General considerations.....	9
1.4.3	Connection of the X-Ray Detector.....	9
1.4.4	The first image.....	10
1.5	How to perform corrections.....	11
1.6	Mathematical description of corrections.....	12
1.6.1	Offset correction	12
1.6.2	Gain correction	12
1.6.3	Multi Gain correction (Gain Sequence)	12
1.6.4	Pixel correction.....	12
1.7	Sorting schemes overview	13
1.8	How to acquire data from several sensors parallel.....	13
1.9	How to use function keys.....	13
1.10	What is new in XIS version 3.2	14
2	XRD Frame Grabber	15
2.1	Framegrabber types.....	15
2.1.1	PCI Framegrabber XRD-FG.....	15
2.1.2	PCI-X Framegrabber XRD-FGX Opto	16
3	Installation	17
3.1	Installation of X-Ray Imaging Software / Installation of the Framegrabber.....	17
3.1.1	Hardware Installation of the frame grabber	17
3.1.2	Software Installation on Windows® XP, Windows® 2000.....	17
3.2	Trouble Shooting.....	18
3.2.1	Setup:.....	18
3.2.2	XIS: Initialization:	18
4	X-Ray Imaging Software	19
4.1	Overview of the Menu Commands.....	19
4.1.1	File Menu Commands	19
4.1.2	Edit Menu Commands.....	19
	Acquire Menu Commands.....	19
4.1.3	Detector Menu Commands.....	20
4.1.4	View Menu Commands	20
4.1.5	Window Menu Commands	20
4.1.6	Options Menu Commands.....	20
4.1.7	Help menu commands.....	20
4.2	File menu	21
4.2.1	New command.....	21
4.2.2	Open command.....	21
4.2.3	Close command.....	22
4.2.4	Print command	22
4.2.5	Print Preview	22

- 4.2.6 Print Setup..... 23
- 4.2.7 Save command..... 23
- 4.2.8 Save As command 23
- 4.2.9 Save Correction files 23
- 4.2.10 Exit command..... 23
- 4.2.11 Import command..... 23
- 4.2.12 1, 2, 3, 4 command..... 23
- 4.3 Edit menu..... 24
 - 4.3.1 Edit build sequence 24
 - 4.3.2 Edit Math 24
 - 4.3.3 Copy 25
 - 4.3.4 Copy Frames 25
 - 4.3.5 Average 25
 - 4.3.6 Select by Value..... 25
 - 4.3.7 Select All..... 25
 - 4.3.8 Deselect All..... 25
 - 4.3.9 Set Value 25
 - 4.3.10 Select Rect 26
 - 4.3.11 Create Pixel Map 26
- 4.4 Acquire menu..... 27
 - 4.4.1 Single Shot 27
 - 4.4.2 Sequence Command..... 27
 - 4.4.3 Continuous Command..... 28
 - 4.4.4 Get Offset Image 28
 - 4.4.5 Get Gain/Offset Image..... 29
 - 4.4.6 Get All Offset Images 29
 - 4.4.7 Link Offset Correction..... 29
 - 4.4.8 Link Gain Correction..... 29
 - 4.4.9 Link Gain Sequence Correction..... 29
 - 4.4.10 Link Pixel Correction..... 30
 - 4.4.11 Acquire Build GainSequence..... 30
 - 4.4.12 Convert to Gain Image 31
 - 4.4.13 End of Acquisition..... 31
- 4.5 Detector 32
 - 4.5.1 Timings menu 32
 - 4.5.2 Detector Mode 32
- 4.6 View menu 32
 - 4.6.1 Status Bar..... 32
 - 4.6.2 Toolbar 32
 - 4.6.3 Acquisition Bar..... 32
 - 4.6.4 LUT - Look-Up-Table 32
 - 4.6.5 Player 33
 - 4.6.6 ZoomBox 33

View Palette..... 34

- 4.6.7 View Control Box..... 34
- 4.7 Window menu 36
 - 4.7.1 New Window..... 36
 - 4.7.2 Cascade 36
 - 4.7.3 Tile Horizontal..... 36
 - 4.7.4 Tile Vertical..... 36
 - 4.7.5 Window Arrange Icons 36
 - 4.7.6 1, 2, ... command..... 36
- 4.8 Options menu..... 36
 - 4.8.1 Acquisition 36
 - 4.8.2 View..... 38
 - 4.8.3 Options Active Sensor..... 38
 - 4.8.4 Options Sensor..... 39
 - 4.8.5 Detector Options..... 40
- 4.9 Help..... 40
 - 4.9.1 Contents 40
 - 4.9.2 Index..... 40
 - 4.9.3 Using Help 40
 - 4.9.4 About 40
- 4.10 Toolbar..... 41
- 4.11 Acquisition Toolbar 41

4.12	Status Bar	42
4.13	Title Bar.....	42
4.13.1	Scroll bars.....	42
4.13.2	Size command (System menu)	42
4.13.3	Move command (Control menu).....	43
4.13.4	Minimize command (application Control menu)	43
4.13.5	Maximize command (System menu)	43
4.13.6	Next Window command (document Control menu).....	43
4.13.7	Previous Window command (document Control menu)	43
4.13.8	Close command (Control menus).....	43
4.13.9	Restore command (Control menu).....	43
4.13.10	Switch to command (application Control menu).....	43
4.14	Standard Dialogs	44
4.14.1	File Selection Dialog.....	44
4.14.2	Overwrite data dialog.....	45
4.14.3	Choose Directory Dialog.....	45
5	Application.....	46
5.1	Mathematical Expressions	46
5.1.1	Parsing expression:	46
5.2	Image Correction	49
5.2.1	Use of the Offset Correction	49
5.2.2	Use of the Gain/Offset Correction	50
5.2.3	Use of the Multiple Gain Correction.....	50
5.2.4	Use and generation of the Pixel Correction.....	51
5.2.5	Correct already acquired images.....	51
5.3	Acquisition Control Modes	51
5.4	Warning table by using the detector and its status	52
6	Details for the Hardware	53
6.1	Readout schema.....	53
6.1.1	Free Running.....	53
6.1.2	External Trigger	54
6.1.3	Internal Trigger	54
6.1.4	How to use the internal trigger mode.....	54
6.1.5	Trigger-modes	55
6.1.6	External trigger inputs/outputs.....	56
6.2	How to use the detector gain setting.....	57
6.3	How to use the detector binning setting.....	57
6.4	File format	58
6.5	Description of Hardware Header.....	59
6.6	Row types	59
6.7	Interrupt sources	59
6.8	Polling mode	60
6.9	Sorting.....	61
6.9.1	Sorting schemes overview.....	61
6.9.2	Sorting RTM 128	61
6.9.3	Sorting RID 256	62
6.9.4	Sorting RID 128-400.....	63
6.9.5	RID 1024-100	64
6.9.6	Sorting RID 512-400 A0	65
6.9.7	Sorting XRD 512-400 A1/A2 // XRD 0840	66
6.9.8	Sorting XRD 512-400 E.....	67
6.9.9	Sorting XRD 1640 A // XRD 1620 AJ // XRD 0820.....	68
6.9.10	Sorting XRD 1620/21 AM/AN	70
6.9.11	Sorting XRD 1620/40 AN CS.....	71
7	X-Ray Imaging Software Library	72
7.1	X-Ray Imaging Library Overview	72
7.2	XISL Functions.....	73
7.2.1	Acquisition Descriptor.....	73
7.2.2	Acquisition_EnumSensors.....	73
7.2.3	Acquisition_GetNextSensor.....	73
7.2.4	Acquisition_Init	74

7.2.5	Acquisition_GetCommChannel	76
7.2.6	Acquisition_DefineDestBuffers	76
7.2.7	Acquisition_SetCallbacksAndMessages	77
7.2.8	Acquisition_Acquire_Image	78
7.2.9	Acquisition_Acquire_Image_PreloadCorr	79
7.2.10	Acquisition_Abort	79
7.2.11	Acquisition_AbortCurrentFrame	79
7.2.12	Acquisition_SetCameraMode	80
7.2.13	Acquisition_SetCorrData	80
7.2.14	Acquisition_Acquire_OffsetImage	81
7.2.15	Acquisition_Acquire_OffsetImage_PreloadCorr	81
7.2.16	Acquisition_Acquire_GainImage	82
7.2.17	Acquisition_Acquire_GainImage_PreloadCorr	82
7.2.18	Acquisition_CreatePixelMap	83
7.2.19	Acquisition_DoOffsetCorrection	83
7.2.20	Acquisition_DoGainCorrection	84
7.2.21	Acquisition_DoOffsetGainCorrection	84
7.2.22	Acquisition_DoPixelCorrection	85
7.2.23	Acquisition_IsAcquiringData	85
7.2.24	Acquisition_GetErrorCode	85
7.2.25	Acquisition_Close	86
7.2.26	Acquisition_CloseAll	86
7.2.27	Acquisition_SetReady	86
7.2.28	Acquisition_GetReady	86
7.2.29	Acquisition_GetConfiguration	87
7.2.30	Acquisition_GetIntTimes	88
7.2.31	Acquisition_SetAcqData	88
7.2.32	Acquisition_GetAcqData	89
7.2.33	Acquisition_GetActFrame	89
7.2.34	Acquisition_SetFrameSyncMode	90
7.2.35	Acquisition_SetFrameSync	90
7.2.36	Acquisition_SetTimerSync	90
7.2.37	Acquisition_SetFrameSyncTimeMode	91
7.2.38	CHwHeaderInfo	92
7.2.39	Acquisition_GetHwHeaderInfo	93
7.2.40	Acquisition_GetWinHandle	93
7.2.41	Acquisition_CreateGainMap	94
7.2.42	Acquisition_Acquire_Image_Ex	94
7.2.43	Acquisition_SetCorrData_Ex	95
7.2.44	Acquisition_GetCorrData_Ex	96
7.2.45	Acquisition_DoOffsetGainCorrection_Ex	96
7.2.46	Acquisition_SetCameraGain	97
7.2.47	Acquisition_Acquire_GainImage_EX_ROI	98
7.2.48	Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr	99
7.2.49	CHwHeaderInfoEx	99
7.2.50	Acquisition_GetHwHeaderInfoEx	101
7.2.51	Acquisition_GetLatestFrameHeader	101
7.2.52	Acquisition_ResetFrameCnt	102
7.2.53	Acquisition_SetCameraBinningMode	102
7.2.54	Acquisition_GetCameraBinningMode	102
7.2.55	Acquisition_SetCameraTriggerMode	103
7.2.56	Acquisition_GetCameraTriggerMode	103
7.3	Demo application	104
7.4	XISL error codes	115
7.4.1	Frame Grabber error codes	116

List of Figures

Figure 1:	Connections of the XRD 1620	9
Figure 2:	Multi Gain correction	12
Figure 3:	Image of the XRD-FG Frame Grabber	15
Figure 4:	Image of the XRD-FGX Opto Frame Grabber	16
Figure 5:	D-Sub connector front view	16
Figure 6:	New File Dialog	21

Figure 7: Build Sequence Dialog	24
Figure 8: Select Pixel by Value Dialog.....	25
Figure 9: Enter new Value Dialog	26
Figure 10: Acquire Sequence Dialog	27
Figure 11: Continuous Acquisition Dialog	28
Figure 12: Build GainSequence Dialog	30
Figure 13: Zoom Box	33
Figure 14: Control Box	35
Figure 15: Dialog for the Acquisition Options.....	37
Figure 16: View Options Dialog	38
Figure 17: Sensor Options Dialog.....	39
Figure 18: Set Detector Options	40
Figure 19: Choose File Dialog	44
Figure 20: Overwrite Data Dialog.....	45
Figure 21: Choose Directory Dialog.....	45
Figure 22: Mathematical Expressions Dialog.....	46
Figure 23: General timing diagram in continuous mode	53
Figure 24: General timing diagram in frame-wise triggered mode.....	55
Figure 25: timing diagram for 'Data Delivered on Demand' triggered mode.....	55
Figure 26 External Trigger inputs/outputs of the XRD 512-400 and XRD 1640 AL/AG detector series	56
Figure 27: External Trigger inputs/outputs of the XRD-FG Frame Grabber	56
Figure 28: Trigger inputs/outputs of the XRD 08xx yN and XRD 162xx yN detectors.....	57
Figure 29: Interrupt sources at a) sequence acquisition mode (4 frames) b) continuous mode.....	60
Figure 30: Sorting overview of the RTM 128	62
Figure 31: Sorting overview of the RID 128-400.....	63
Figure 32: Sorting overview of the RID 1024-100.....	64
Figure 33: Sorting overview of the RID 512-400 A0	65
Figure 34: Sorting overview of the XRD 512-400 A1/A2.....	66
Figure 35: Sorting overview of the XRD 512-400 E	67
Figure 36: Sorting overview of the XRD 1640 A / 1620 AJ	68
Figure 37: Sorting overview of the XRD 1620 AN / 1621 AN.....	70
Figure 38: Sorting overview of the XRD 1620/40 AN CS.....	71

List of Tables

Table 1: X-Ray adjustments.....	10
Table 2: Sorting Schemes.....	13
Table 3: Function Keys	13
Table 4: LVDS connector Pin assignment (GP_** use LVDS signalling standard).....	16
Table 5: File menu	19
Table 6: Edit menu.....	19
Table 7: Acquire menu.....	19
Table 8: Detector menu	20
Table 9 : View menu	20
Table 10: Window menu	20
Table 11: Options menu.....	20
Table 12: Help menu.....	20
Table 13: Settings of the New File Dialog	21
Table 14: Settings of the Print Dialog	22
Table 15: Settings of the Print Preview Dialog.....	22
Table 16: Settings of the Sequence Dialog.....	24
Table 17: Settings of the Select Pixel by Value Dialog	25
Table 18: Settings of the Acquire Sequence Dialog	27
Table 19: Displaying different Timings.....	32
Table 20: Zoom settings of the Control Box.....	34
Table 21: ROI settings of the Control Box	35
Table 22: Settings of the View Options Dialog.....	38
Table 23: Windows Toolbar	41
Table 24: Acquisition Toolbar	41
Table 25: Status Bar	42
Table 26: Switch Command Dialog.....	44

Table 27: Parsing symbols.....	46
Table 28: Type Operators	47
Table 29: SUM - function parameters	47
Table 30: Description of the trigger input/output signals	56
Table 31 PIN assignment of Trigger signal for the XRD 08xx yN and XRD 16xx yN detectors (LVDS Signals).....	57
Table 32: LVDS connector Pin assignment (GP_** use LVDS signalling standard).....	57
Table 33: Overview of the five Detector Gain Settings Table	57
Table 34: Overview of the detector binning modes.....	58
Table 35: Single image and file sequence format	58
Table 36: File header description.....	58
Table 37: Type of Data	58
Table 38: Timing diagram of the row types	59
Table 39: Sorting schemes overview	61
Table 40: Sorting overview of the RTM 128.....	61
Table 41: Sorting overview of the RID 256	62
Table 42: Sorting overview of the RID 128-400	63
Table 43: Sorting overview of the RID 1024-100	64
Table 44: Sorting overview of the RID 512-400 A0.....	65
Table 45: Sorting overview of the XRD 512-400 A1/A2	66
Table 46: Sorting overview of the XRD 512-400 E	67
Table 47: Sorting overview of the XRD 1640 A / 1620 AJ	69
Table 49: sorting overview of the XRD 1620/40 AN CS	71
Table 50: Overview of the X-Ray Imaging Software Library	72
Table 51: Supported Frame Grabber device and their channel type	74
Table 52: Sorting Flags	75
Table 53: Supported Frame Grabber device and their channel type	76
Table 54: Supported Acquisition Sequence Options	78
Table 55: Description of the SyncMode Options.....	87
Table 56: Trigger option Flags	90
Table 57: Description of the ChwHeaderInfo structure	92
Table 58 : Description of the SyncMode Options.....	95
Table 59: Detector gain values	97
Table 60: Description of the CHwHeaderInfoEx structure	100
Table 61: Description of the SyncMode Options.....	115
Table 62: Description of the Frame Grabber Error Codes	116

1 General

1.1 About the program XIS

The program XIS X-Ray Imaging Software is capable to demonstrate the functions of the **digital amorphous Silicon X-Ray Detector** (former **RID Radiation Image Detector**). It can be used with any type of XRD through the installed PCI or PCI-X frame grabber board. It detects automatically the size of the sensor and receives images of the detector in a 16 bit digitized data format (65535 steps). The images are presented on the screen with 256 grey levels.

Offset, Gain/Offset and Gain Sequence/ Offset Images are used to take out the specific dark current of the detector and the specific irradiation of the x-ray source of the images. The use of the Offset, Gain/Offset and Gain Sequence/ Offset Images to correct images is recommended for best resolution quality.

A **Pixel Correction** allows the 'software repair' of defect pixels to enhance the image quality.

The detector allows the setting of different frame times. The shortest possible frame time is selected at startup, but any timing can be set to optimize the system capabilities.

The program XIS is embedded in the Windows®2000 and Windows®XP environment and can be handled in the same manner as other standard WINDOWS programs. It allows limited image processing to present the acquired images in best image quality by using general WINDOWS routines or specific commands only available in the XIS software.

To present acquired images or sequences of images, the program allows the storage of images in various formats. Hard copies of the images can be printed with any installed printer. It is recommended to use a printer allowing to resolve 256 grey levels for optimum presentation.

1.2 Liability policy

PerkinElmer endeavours to provide a quality product and takes full responsibility for the correct function of the XIS-Software with its own digital X-Ray detectors. PerkinElmer is not responsible for any misuse of the Software, or for the performance with any detector other than with PerkinElmer detectors. PerkinElmer also takes no responsibility for any hardware or software failures caused by installing and using this software on a third-party computer system. The software is solely intended to operate the PKI digital X-Ray detectors with PKI XRD-FG[X] Framegrabbers for product evaluation purposes. PerkinElmer is not liable for property damage, physical injury, or data corruption. PerkinElmer reserves the right to make changes or modifications to this product without notice.

PerkinElmer Inc. SHALL NOT BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING, BUT NOT LIMITED TO, LOSS OF USE, REVENUE, OR PROSPECTIVE PROFITS RESULTING FROM THE USE OF THIS DOCUMENT OR THE PRODUCT TO WHICH IT RELATES. ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY DISCLAIMED.

1.3 Copyright

This software is proprietary to PerkinElmer and dedicated for use in combination with the PKI digital X-Ray detectors with PKI XRD-FG[X] Framegrabbers for product evaluation purposes only and without licensing agreement. This document and the product to which it relates are protected by copyright law from unauthorized reproduction.

Notice to U.S. Government End Users

The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. 2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. 12.212 or 48 C.F.R. 227.7202, as applicable. Consistent with 48 C.F.R. 12.212 or 48 C.F.R. 227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to the U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished rights reserved under the copyright laws of the United States.

1.4 Getting Started - The first image

1.4.1 Introduction

This chapter describes the procedures to get the first x-ray images with adequate quality. It explains how one can use correction files with the appropriate settings of the detector integration time and x-ray source parameters.

1.4.2 General considerations

The detector can, in principal, produce images without any correction. These images contain the offset of the readout electronics, the individual offsets of each pixel (dark current) as well as the electronics and pixel gain differences, apart from the x-ray source non-uniformities.

Each column is connected to one channel of the readout electronics with the specific channel offset. This results in a dark image with vertical stripes caused by the individual channel offsets. The dark image may also contain pixels which are brighter than the others caused by a higher dark current. The detector is arranged in groups of 128 (256) readout channels. The groups can deviate in their gain such that one can distinguish blocks of 128 (256) channels in a bright image caused by this gain difference. The panel itself may contain pixels and perhaps row or columns which are defective (totally black or white). To eliminate these detector specific effects and to obtain good quality results each image will be 'offset' and 'gain' corrected and if it is required the defective pixels will also be corrected. The creation of the correction files is described in the chapter "How to perform corrections". The mathematical procedures which are applied to each pixel are described in Mathematical description of corrections.

1.4.3 Connection of the X-Ray Detector

Before starting the connection of the X-Ray Detector ensure that the frame grabber and the software is installed correctly (Chapter 3). If not please begin with the frame grabber and software installation.

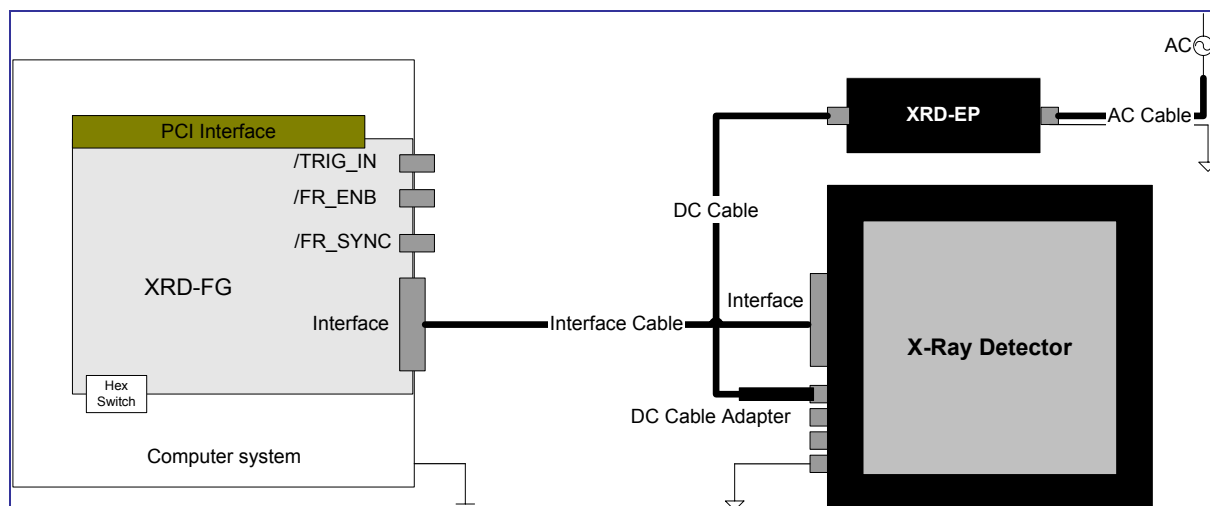


Figure 1: Connections of the XRD 1620

The computer with the frame grabber and the monitor need to be grounded through the protection earth conductors within the power cords. The required potential equalization of the Detector and the XRD-EP power supply has to be managed through the labelled connectors and the potential equalization between both devices is managed through the XRD-EPS DC Cable. For safety reasons only original cables and connectors should be used.

The X-Ray detector should be connected as described in the following manner and as shown in Figure 1

1. Shut down the computer
2. Connect the **Detector** and the **XRD-EP** with the potential equalization
3. Connect the **Frame Grabber** and the **Detector** via **Interface Cable**
4. Connect the **XRD EPS DC Cable** with the Power Connector of the **Detector**
5. Connect the **XRD-EP** and the **XRD EPS DC Cable**
6. Connect the **XRD-EP** with the mains via **AC Cable**.
7. Switch on the **XRD-EP**
8. Switch on the Computer

1.4.4 The first image

The detector is powered on and all cable connections are performed. At startup the frame grabber board will be initialized and afterwards a dialog appears to select the mode of the frame grabber board. "Yes" enables the Interrupt Mode and "No" the Polling. In both cases the system attempts to initialize the frame grabber board(s) and the connected detectors. The "Cancel" button starts the program without initialization. The initialization can take some time depending on the number of frame grabber boards and detectors. If more than one detector is connected a dialog appears that contains a list of all recognized detectors and an active detector has to be selected. All following actions correspond to that active sensor. Now the system is ready to acquire the first image.

The Acquire\Single Shot command acquires a single image. If the detector was not irradiated only a dark image is displayed. The image can be enhanced by the brightness and contrast settings. As explained above the uncorrected dark image contains vertical stripes caused by the electronics offset. By choosing the Continuous acquisition mode the image is refreshed on the display in the selected frame rate.

In the next step the detector should run in the continuous mode and the x-ray source should be switched on to irradiate the detector. The brightness and contrast should be set to default (F2-KEY: 0-65535). If the gray image is displayed the parameters of the x-ray source and detector are in a good range. If a white or a black image is displayed the x-ray source has to be adjusted in the following way:

Displayed image -->	white	black	
Action -->	decrease	increase	-the x-ray tube current
	increase	decrease	-the distance
	append	remove	-additional filters
	decrease	increase	-the x-ray tube voltage

Table 1: X-Ray adjustments

1.5 How to perform corrections

The XRD works as an independent detector to acquire X-ray images. After starting the XIS.EXE software, the detector is automatically initialized and sends out images in its fastest frame time (TIMING0) among all available ones (Timings menu: list of possible frame times).

The X-Ray Detectors (XRD) need an Offset correction to take into account the dark current of each pixel. In particular, during the warm-up phase of the detector, the Offset is not stable and during this time period the detector use is not recommended. During operation it is recommended to refresh periodically the Offset.

Additionally, a Gain correction is necessary to homogenize differences in pixel sensitivities or to take into account the X-ray beam illumination, therefore it is very important that the whole image area is illuminated homogeneously. The Gain correction should be carried out in an optimum dynamic range of the sensor (70-80 % of the full scale range FSR) or in the dynamic range of interest. The radiation intensity used to create the gain image can depend on the application, e.g. if the typical grey level is about 10.000 ADU and the remaining area is saturated, it is recommended to use a gain image created at 10.000 ADU. The use of an Offset and Gain calibration eliminates offset dependency and therefore any stored Gain correction file can be used for a specific frame time for longer time periods.

The image performance can be enhanced by using the Multiple Gain Correction. For each dynamic range of interest a separate offset corrected and averaged bright image is used as an interpolation point. The maximum number of interpolation points depends on the installed computer memory. It is important that each bright image is completely and homogeneously illuminated.

The Pixel correction allows a 'software repair' of defective pixels to enhance image quality. Improper pixel values are replaced with the averaged value of the surrounding eight adjacent pixels where defective pixels are not used. The pixel correction is only performed on specific pixels, mapped in the file PXMLMASK.HIS. The delivery package includes the PXMLMASK.HIS file for the specific detector, the user can also generate their own correction file. Please be aware that the number of pixels used for the mean correction should be minimized. The pixel correction procedure requires CPU time and depending on its speed, a slower presentation of the acquired images on screen might occur in relation to the selected timing mode of the system. The main screen of the XIS software sends out a warning message displayed on the screen if all of the acquired images are not accepted by the computer.

1.6 Mathematical description of corrections

All correction files are saved in the HIS file format. Offset and pixel correction files use unsigned 16 bit integer to store the pixel data while gain files use unsigned 32 bit integers.

1.6.1 Offset correction

To create the offset correction image an average image of a sequence of dark images have to be acquired. These image data are subtracted from the incoming pixel data at acquisition time.

1.6.2 Gain correction

To create the gain image an average image of a sequence of Offset-corrected bright images have to be acquired. Afterwards the median value of the pixel data of the whole sensor is evaluated and the entries in the gain image are derived by the following formula

$$\text{entry} = \text{median} * 65536 / (\text{bright_value} - \text{offset_value})$$

For performance reasons the correction is done using integer arithmetic. To improve the precision of the calculation the gain data bits are shifted to left by 16 bits. The gain correction is performed by multiplying the offset correct pixel data by the gain data and shift the result back to right by 16 bits.

1.6.3 Multi Gain correction (Gain Sequence)

To create a Sequence for the Multi-Gain correction an offset corrected averaged bright image has to be acquired for each range of interest.

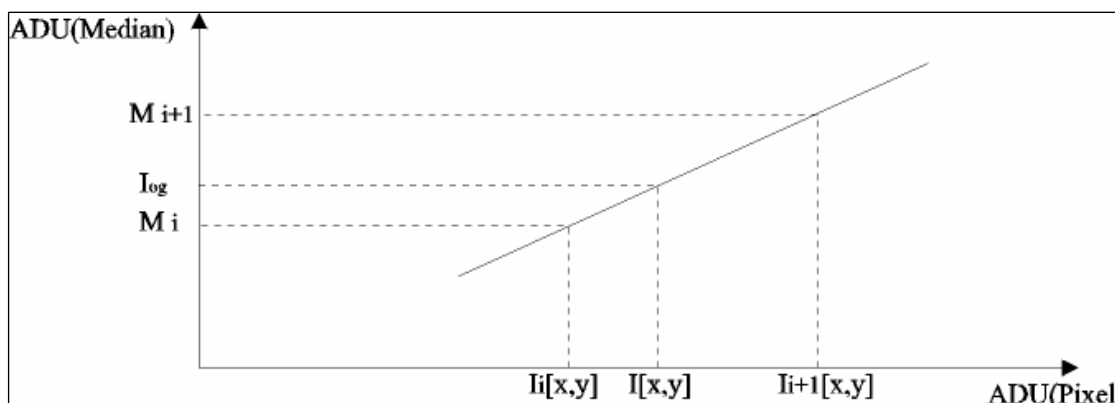


Figure 2: Multi Gain correction

- M_i: Median Value of the offset corrected Image i
- I_i[x,y]: Intensity of pixel (x,y) of the offset corrected image i
- I[x,y]: Intensity of pixel (x,y) of the offset corrected input image
- I_{og}[x,y]: Intensity of pixel (x,y) of the offset-gain corrected output image

The pixel value I_{og}[x,y] is calculated with the following formula:

$$I_{og}[x, y] = M_i + \frac{(M_{i+1} - M_i)}{I_{i+1}[x, y] - I_i[x, y]} * (I[x, y] - I_i[x, y])$$

1.6.4 Pixel correction

The pixel correction is done by the XISL function Acquisition_DoPixelCorrection. The pixel correction information are stored in a simple HIS file, where the bad pixels are set to a value of 65535 and the others are set to zero. When the pixel correction image is linked to the acquisition unit the XIS derives the correction data used by the above mentioned function from the pixel correction file. For that purpose the pixel marked as defects are corrected by the average of the neighboring good pixels. A helper function that converts pixel data in HIS format to a suitable map for Acquisition_DoPixelCorrection is Acquisition_CreatePixelMap.

1.7 Sorting schemes overview

Depending on the sensor and detector type the data come in different orders from the detector. The XISL sort the data in an internal buffer with highly optimized routines written in machine code. Normally the user don't need to care about sorting because the data returned in the acquisition buffer defined in Acquisition_DefineDestBuffers are in the correct order. If the sensor and detector type is unknown the XIS comes up with a detector type dialog at initialization, where the correct sorting has to be entered. The following detector types and sortings are supported:

RID128	1
RID256	2
RID128-400	3
RID1024-100	4
RID512-400 A0	5
XRD512-400 A1/A2, XRD 0840	6
XRD512-400 E	7
XRD 1640 A , XRD 0820	8
XRD 1620 AJ	8
XRD 1680 A	9
XRD 1620/21 AM/AN, XRD 1621	11
XRD 16x0 AN CS	12

Table 2: Sorting Schemes

1.8 How to acquire data from several sensors parallel

This feature of the XIS can be used by plugging in two or more frame grabber boards and connecting them to suitable detectors. But it is not recommended to use different types of frame grabbers together. Each frame grabber has to be set to a unique sensor number at the hex switch. During the XIS initialization the active sensor has to be selected. Choose Acquire/Continuous to start a continuous data acquisition or acquire correction images or link the correction data before. By the dialog Active Sensor in the Options menu another sensor from the list can be selected. The following actions correspond to the new active sensor. But now new correction files have to be acquired or to be linked for the new sensor. The command Window/Tile shows both acquisition windows how they acquire data parallel.

1.9 How to use function keys

Function keys are available in the acquisition mode for an easy image presentation. The function key **F2** allows the display of the full range of the image, presented in 256 gray levels. The function key **F7** allows to switch to on-line Offset, **F8** to on-line Offset/Gain, **Shift F8** to on-line the Multi-Gain and **F9** is used for on-line Median correction.

F1	HELP	Displays the Help menu
F2	DEFAULT	Set to default display mode. Reset brightness and contrast settings.
F7	OFFSET	Enable/disable Offset correction.
F8	OFFSET/GAIN	Enable/disable Gain/Offset correction.
Shift F8	OFFSET/GAINSEQUENCE	Enable/disable GainSequence/Offset correction.
F9	BAD PIXEL	Enable/disable Pixel correction.
<ESC>	STOP	Stop Acquisition
F11	HEADER	Display of the last acquired HwHeader
Shift F11	Frame Counter	Reset of the detector frame-counter

Table 3: Function Keys

1.10 What is new in XIS version 3.2

- ▶ The new detector series **XRD 1621 xN** is integrated.
- ▶ The new detector type **XRD 1620/40 xN CS** is integrated
- ▶ Supports the new optical Framegrabber Device **XRD-FGX Opto** including onboard corrections.
- ▶ **XIS/XISL** is running under Windows® 2000 and Windows® XP.
- ▶ New Library-Functions
Acquisition_Acquire_Image/OffsetImage/GainImage_PreloadCorr(..) to provide shorter response times with the new optical Framegrabber.
- ▶ New Library-Function Acquisition_Acquire_GainImage_Ex_ROI (Gain with Region of Interest)
- ▶ HeaderID 14 with Camera-type 1 or 2 (**XRD 1621 xN**, **XRD 0820 xN**)
Camera-type 1 supports detector binning
Camera-type 2 supports binning and special trigger - modes.
- ▶ **XRD FGX Opto** supports onboard averaging up to 512 frames.
Onboard averaging will be done automatically when the amount of frames to average is one to the power of 2 (1, 2, 4, 8, ..512). Skip frames will not work in this mode.
- ▶ New Library-Functions (for Cameratype 1 and 2, HeaderID 14)
 - Acquisition_ResetFramecounter(..)
 - Acquisition_GetBinningMode(..)
 - Acquisition_SetBinningMode(..)
 - Acquisition_GetTriggerMode(..)
 - Acquisition_SetTriggerMode(..)
 - Acquisition_GetLatestFrameHeader(..)
 - Acquisition_GetHwHeaderInfoEx(..)
 - Acquisition_SetFrameSyncTimeMode(..)
- ▶ Extended Hardware Header for **XRD 16y1 xN**, **XRD 0820 xN** including Frame Counter, measured integration time and Trigger-lost-Flag
- ▶ New **XIS** functions:
 - F11 key to display the last acquired HwHeader.
 - Shift F11 to reset the detector frame-counter.
 - Change detector Binning Mode with Options=>detector options
- ▶ Extended demo main.c source in the SDK to show how to use the new functions for the 1621

2 XRD Frame Grabber

2.1 Framegrabber types

The software supports the **XRD FG** and the **XRD FGX Opto** Frame grabber developed by PerkinElmer Optoelectronics. The **XRD FGX Opto** works with the operation systems Windows®XP and Windows®2000.

2.1.1 PCI Framegrabber XRD-FG

IB – XRD Interface Bus

1 – /TRIG IN

2 – /FR_EN (Begin of frame)

3 – /FR_SYNC

4 – Status LEDs

5 – HEX Switch

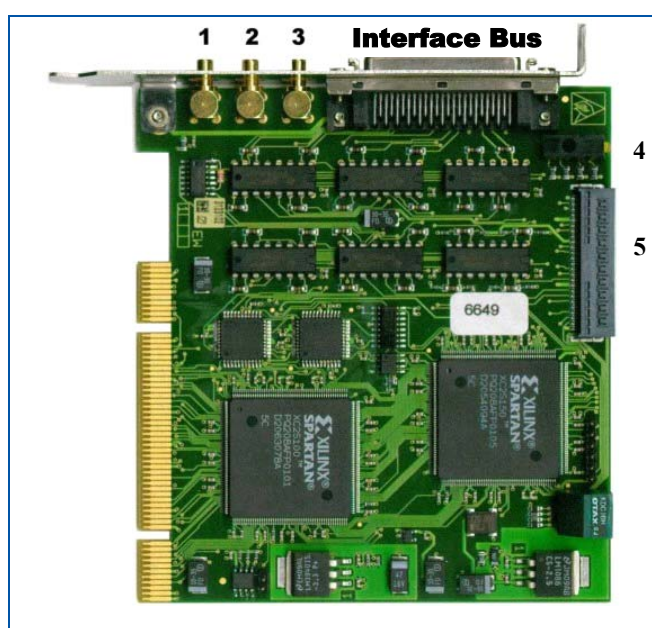


Figure 3: Image of the **XRD-FG** Frame Grabber

2.1.2 PCI-X Framegrabber XRD-FGX Opto

PCI-X Rev 2.0 133MHz 3,3V

- 1 – HEX Switch
- 2 – D-Sub connector (details below)
- 3 – Optical Tranceiver Out (IP68)
- 4 – Optical Tranceiver In (IP68)



Figure 4: Image of the **XRD-FGX Opto** Frame Grabber

The new Detector types XRD 1621/41AN are equipped with an optical Interface only.
Features are:

- Data Rate up to 15fps @ 2k x 2k x 16bit
- Hardware Image correction (offset/gain/mean) w/o loss of frames up to 10 calibration points for non linear correction.
- New Glass fiber Interface (galvanic isolation, robust, plug type IP68)
- New D-Sub trigger in/out connector with LVDS signals.

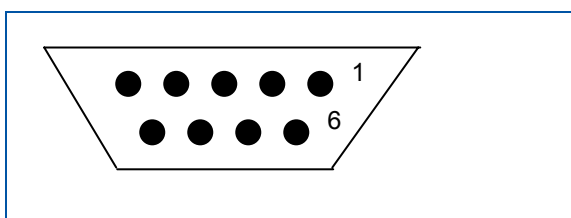


Figure 5 D-Sub connector front view:

PIN	DESCRIPTION	
1	GP_OUT0+	frame enable
2	GP_OUT1+	frame sync
3	GP_IN0+	trigger in
4	GP_IN1+	na
5	GND	
6	GP_OUT0-	frame enable
7	GP_OUT1-	frame sync
8	GP_IN0-	trigger in
9	GP_IN1-	na

Table 4: LVDS connector Pin assignment (GP_** use LVDS signalling standard)

3 Installation

3.1 Installation of X-Ray Imaging Software / Installation of the Framegrabber

3.1.1 Hardware Installation of the frame grabber

To install the frame grabber it is recommended to shut down the computer and unplug the power supply. Failure to do so may cause severe damage to both your motherboard and the frame grabber. In most cases the mainboard has an onboard LED which shows the power OFF mode or the soft-off mode (Power is still on). Hold the grabbers by the edges and try not to touch the chips, leads or connectors. Please place the frame grabber on a grounded antistatic pad whenever the grabbers are separated from the system.

It is recommended to use an exclusive IRQ port, please read your mainboard guide for more information. If more than one frame grabber has to be used in the system the switch on the left side of the grabber has to be set to a unique number for every board.

Please read the readme.txt on the installation CD for the latest information before installing the frame grabber driver and the application software.

1. Shut down the computer
2. Unplug the power supply and remove the computer system's cover
3. Turn the switch of the grabber to a unique number for every board
4. Carefully align the frame grabber's connectors and press firmly
5. Secure the card(s) on the slot with a screw
6. Replace the computer system's cover
7. Restart the computer system
8. Log on to Windows using the administrator account

3.1.2 Software Installation on Windows®XP, Windows®2000

1. After LOG ON the Hardware Wizard notice the new frame grabber as an multimedia device
2. Plug in the XIS Installation CD-ROM
3. Follow the hardware wizard to install the XRD-FG(X) as a new device
4. After installation of the XRD-FG(X) and XISL drivers by the Wizard start the XIS setup from the appearing menu or if the Setup does not start automatically start the START.EXE in the root directory of the CD
5. The XIS SETUP program will lead you through the installation process
6. Restart the computer system
7. The XIS is now ready to start
8. If the initialization of the frame grabber and the detector is successful a corresponding message appears in the status bar.

3.2 Trouble Shooting

3.2.1 Setup:

Colour Resolution:

The setup informs you, that the selected color resolution is another than 256 colors. Please choose "Settings/Control Panel" in the Windows "Start" menu and double click the icon "Display". Several property sheets appear. Select "Settings" and enter at least "256 color" in the "Color Palette" list box.

3.2.2 XIS: Initialization:

XIS Error 2 // Eltec Error –303

IRQ-Problem

Change the IRQ-Settings in the System Settings Control Panel or in the BIOS. Use an other PCI-Slot without IRQ-Sharing or remove not absolutely necessary PCI-boards or use the polling mode.

XIS Error 2 // Eltec Error –41:

Virtual device driver not present
No framegrabber driver is loaded.

XIS Error 23:

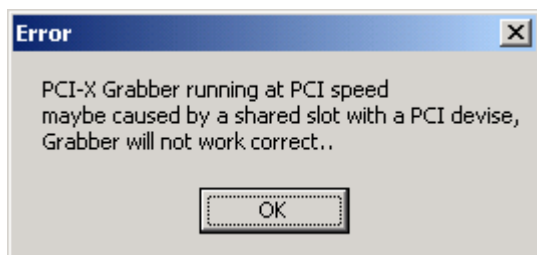
Hardware header invalid

- ▶ Check the connections of the detector and the frame grabber board.
- ▶ Check if there are older libraries of an earlier installation in the path the delete or rename it.
- ▶ Try the detector setup by the Option/Acquisition dialog
- ▶ Write down the header information and contact your vendor

Optical Grabber only:

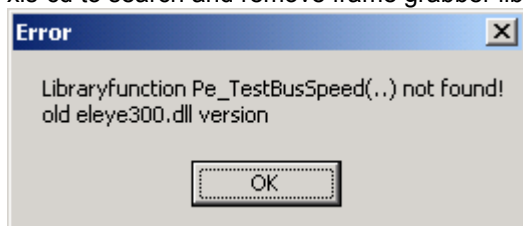
PCI-FGX Opto – PCI Problem

- ▶ This Message occurs when the PCI-X Slot is only in 32Bit mode or running with 66MHz in place of 100/133MHz. This maybe caused by a PCI card connected to a PCI-X slot. So the PCI-X Grabber will not work correct. Please try using another PCI-X slot with 100/133MHz to avoid this sharing problem.



PCI-FGX Opto – Driver Version Problem

- ▶ This Message occurs when the Library function Pe_TestbusSpeed is not found. Please try to install the latest driver. Be sure that all files of the old driver are removed / uninstalled before. You can use the **new DriverDelInstaller.exe** which is located on the xis-cd to search and remove frame grabber libraries from the system32 directory.



4 X-Ray Imaging Software

4.1 Overview of the Menu Commands

4.1.1 File Menu Commands

The File menu offers the following commands:

New	Creates a new document.
Open	Opens an existing document.
Close	Closes an opened document.
Save	Saves an opened document using the same file name.
Save As	Saves an opened document to a specified file name.
Save Correction files	Saves all loaded correction files.
Import	Imports an existing uncompressed document
Print	Prints a document.
Print Preview	Displays the document on the screen as it would appear printed.
Print Setup	Selects a printer and printer connection.
Exit	Exits XIS.

Table 5: File menu

4.1.2 Edit Menu Commands

The Edit menu offers the following commands:

Copy	Copies data from the document to the clipboard.
Math	A dialog comes up that gives you the possibility to manipulate sensor data by sophisticated build in functions.
Copy Frames	Copies frames from one document to an empty document.
Average	Averages over the images of a sequence.
Select by value	Selects image pixels by their data values.
Select all	Selects all image pixels.
Deselect all	Deselects all image pixels.
Select Rect	Selects a rectangle of pixels by coordinates
Set value	Sets all selected pixels to a specified value.
Create pixel map	All selected pixels are marked as defects and a pixel correction map is created.

Table 6: Edit menu

Acquire Menu Commands

The Acquire menu offers the following commands, which enable the user to acquire images of the detector and to perform suggested corrections to achieve best image quality:

Continuous	Acquires and displays images continuously in the selected timing mode. (Default TIMING0: shortest acquisition time of the detector.)
Single Shot	Acquires a single image.
Sequence	Acquires a sequence of images.
Link Offset Corr	Linking of a stored Offset correction file to acquire images corrected with the loaded Offset file.
Link Gain Corr	Linking of a stored Gain correction file to acquire images corrected with the loaded Gain file.
Link Gain Sequence	Linking of a stored Gain-Sequence correction file to acquire images corrected with the loaded Gain-Sequence file.
Link Pixel Corr	Linking of a stored Pixel correction file to acquire images corrected with the loaded Pixel file.
Get Offset Image	Acquires a new Offset image for later Offset corrected image acquisition.
Get Gain/Offset Image	Acquires a new Offset corrected Gain image for later Gain/Offset corrected image acquisition.
Get All Offset Images	Acquires all Offset images for later Offset corrected image acquisition. (All frame times available in the Timings menu are automatically acquired)
Build Gain Sequence	Creates a Gain-Sequence correction file to be used with the Gain Sequence correction.
Convert to Gain Image	Creates a new gain image from an offset and a bright image. In the bright image a region of interest can be selected to optimize the gain image regarding best presentation of this area.

Table 7: Acquire menu

4.1.3 Detector Menu Commands

The Detector menu offers the following commands:

Mode	Selects the Detector mode. The detector is able to operate the following modes: free running, triggered by external sources triggered by an customized internal timing
Timings	Selects the currently active frame time for detector operation.

Table 8: Detector menu

4.1.4 View Menu Commands

The View menu offers the following commands:

Acquisition Bar	Shows or hides the acquisition toolbar
Toolbar	Shows or hides the toolbar.
Status Bar	Shows or hides the status bar.
LUT	Shows or hides the LUT window.
Control	Shows or hides the Control window.
Player	Shows or hides the Player bar (available using Acquire Sequence command).
Zoom Box	Shows or hides the Zoom Box window

Table 9 : View menu

4.1.5 Window Menu Commands

The Window menu offers the following commands, which enable arranging multiple views of multiple documents in the application window:

New Window	Creates a new window that views the same document.
Cascade	Arranges windows in an overlapped fashion.
Tile	Arranges windows in non-overlapped tiles.
Arrange Icons	Arranges icons of closed windows.
Window 1, 2, ...	Goes to specified window.

Table 10: Window menu

4.1.6 Options Menu Commands

The Options menu offers the following submenus:

Acquisition	Allows to enter specific acquisition options.
View	Allows to enter specific view options (zooming etc.).
Active Sensor	Allows to change the currently active sensor if more then one sensor is connected to the system.
Sensor	Gives information and allows to set some options for the active sensor

Table 11: Options menu

4.1.7 Help menu commands

The Help menu offers the following commands, which provide you assistance with this application:

Help Topics	Offers you an index of topics on which you can get help.
About	Displays the version number of this application.

Table 12: Help menu

4.2 File menu

4.2.1 New command

This command creates a new document in XIS. The type of the file can be selected in the New File dialog box.

Shortcuts

Toolbar: 

Keys: CTRL+N

The following dialog appears if the user creates a new document

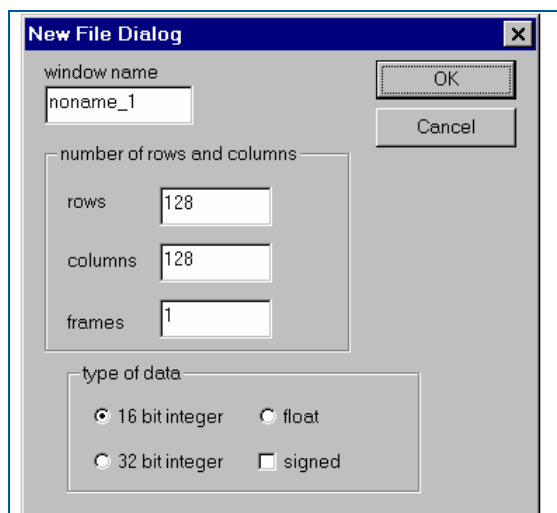


Figure 6: New File Dialog

Window name	This name specifies the name of the new document.
rows, columns, frames	This input numbers specify the number of columns, rows and frames of the new document. All data values are set to zero.
type of data	The XIS supports different data types for data acquisition and evaluation. The suitable type could be specified here. All operations done by the XIS are type safe. If a data type isn't suitable for the desired operation a warning is given out and the data type is changed allocating a new suitable data buffer. In this case all old data are lost.

Table 13: Settings of the New File Dialog

4.2.2 Open command

This command opens an existing document in a new window. Multiple documents can be opened at once. Use the Window menu to switch among the multiple open documents.

See Window 1, 2, ... command.

Shortcuts

Toolbar: 

Keys: CTRL+O

4.2.3 Close command

The Close command closes all windows containing the active document. XIS suggests that the changes of document have saved before closing it. If a document is closed without saving, all changes made since the last time of saving are lost. Before closing an untitled document, XIS displays the Save As dialog box and suggests to name and save the document.

A document can also be closed by using the Close icon on the document's window, as shown below:



4.2.4 Print command

This command prints a document. First a Print dialog box appears to specify the range of pages to be printed, the number of copies, the destination printer, and other printer setup options.

Shortcuts

Toolbar: 

Keys: CTRL+P

The following options can be used:

Printer	This is the active printer and printer connection.	
Setup	Displays a Print Setup dialog box to change the printer and printer connection.	
Print Range	Specify the pages you want to print:	
	All	Prints the entire document.
	Selection	Prints the currently selected text.
	Pages	Prints the range of pages you specify in the From and To boxes.
Copies	Specify the number of copies you want to print for the above page range.	
Collate Copies	Prints copies in page number order, instead of separated multiple copies of each page.	
Print Quality	Select the quality of the printing.	
	Generally, lower quality printing takes less time to produce.	

Table 14: Settings of the Print Dialog

4.2.5 Print Preview

This command displays the active document as it would appear when printed. When this command is called, the main window will be replaced with a print preview window in which one or two pages will be displayed in their printed format. The print preview toolbar offers you the following options:

Print	Bring up the print dialog box, to start a print job.
Next Page	Preview the next printed page.
Prev Page	Preview the previous printed page.
One Page / Two Page	Preview one or two printed pages at a time.
Zoom In	Take a closer look at the printed page.
Zoom Out	Take a larger look at the printed page.
Close	Return from print preview to the editing window.

Table 15: Settings of the Print Preview Dialog

4.2.6 Print Setup

This command presents a Print Setup dialog box to specify the printer and its connection.

4.2.7 Save command

The Save command saves the active document to its current name and directory. Saving a document for the first time, XIS displays the Save As dialog box to name the document. To change the name and directory of an existing document before saving, choose the Save As command.

Shortcuts

Toolbar: 

Keys: CTRL+S

4.2.8 Save As command

Use this command to save and name the active document. XIS displays the Save As dialog box to name the document.

To save a document with its existing name and directory, use the Save command.

4.2.9 Save Correction files

Use this command to save all currently loaded correction files. The files are stored in the correction folder defined in the Options/Acquisition dialog.

4.2.10 Exit command

Use this command to end the XIS session. The same effect has the Close command on the application Control menu. XIS prompts to save documents with unsaved changes.

Shortcuts

Mouse: Double-click the application's Control menu button.



Keys: ALT+F4

4.2.11 Import command

This command opens an existing file containing raw data in a new window. In the appearing dialog the number of columns, rows, frames and the data type can be selected.

4.2.12 1, 2, 3, 4 command

Use the numbers and filenames listed at the bottom of the File menu to open the last four documents you closed. Choose the number that corresponds with the document you want to open.

4.3 Edit menu

4.3.1 Edit build sequence

This menu entry opens the "Build Sequence" dialog to edit available sequences of images by inserting or appending data from other images or to create a new sequence from existing images.

The "Build Sequence" dialog looks like the following image:

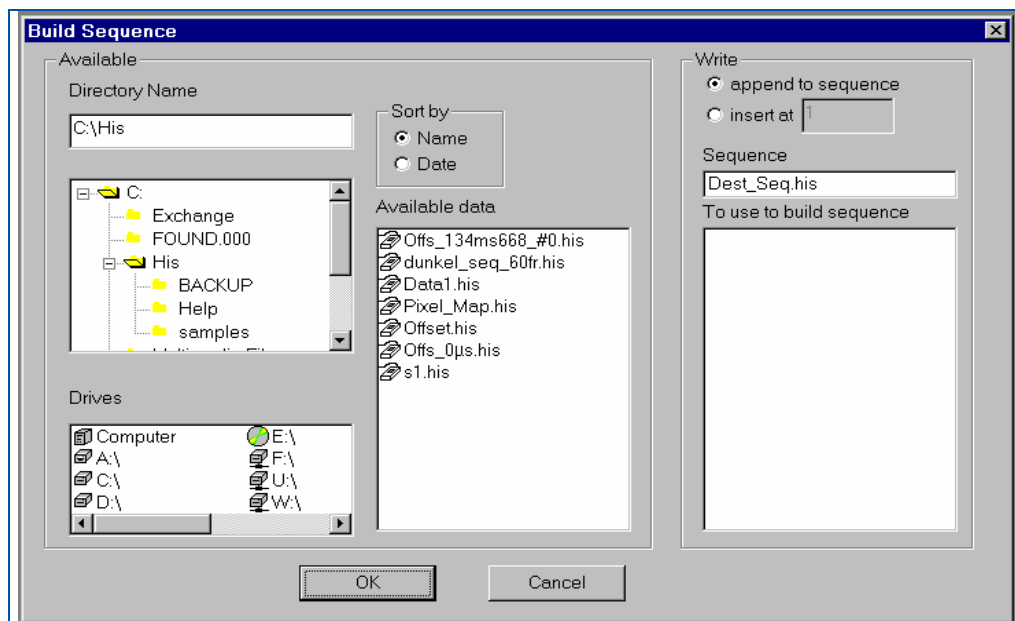


Figure 7: Build Sequence Dialog

The available group box shows all available drives, directories and files. The currently loaded images are listed under the drive "computer" in the drives list box. To list stored images click on the items of the drive box and the entries of the above tree control that displays the directory structure.

Sort by radio buttons	Change the sorting of the available images by the corresponding radio buttons
append to sequence	Check this radio button to append images to a sequence.
insert at button	Check this radio button to insert images at a specified location in the data stream. The last frame of the sequence is specified by the "insert at" edit box.
Sequence edit box	Drag and drop a sequence file from the "available data" box or edit the name. If the file name isn't loaded and not available on the storage media a new data window is created and the data are appended.
To use to build sequence box	This list contains the files which will be appended to or inserted into the sequence. Drag and drop files from the "available data" list box.

Table 16: Settings of the Sequence Dialog

4.3.2 Edit Math

The Edit Math menu entry opens the "enter expression" dialog box. The built-in parser supports several mathematical image calculations, e.g. addition of image data to numbers or other image data, the subtraction of images and numbers, the division and multiplication of images and numbers and averaging of different images and numbers.


The status of the parser is shown in the Status Bar.

The details of the parser are described in the chapter "Mathematical expressions".

4.3.3 Copy

This command copies selected data into the clipboard. If no data selected, the Copy command is unavailable. Copying data to the clipboard replaces the previously stored data.

Shortcuts

Toolbar: 

Keys: CTRL+C

4.3.4 Copy Frames

This command copies a number of frames into a new document. In the appearing dialog the range of frames can be specified.

4.3.5 Average

This command derives the average of the frames provided in the active document and copy the result into a new empty document.

4.3.6 Select by Value

This command selects all pixels of the active document that are within or out of a specified range. The selected pixels appear on the screen in the manner, which is adjusted by the control box. All following actions are related to this selection.

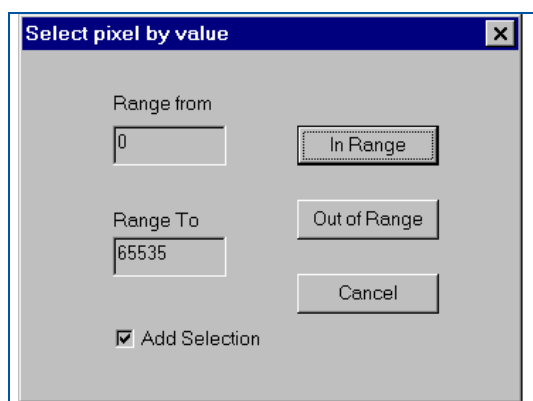


Figure 8: Select Pixel by Value Dialog

Range from	This edit control specifies the lower limit of the selection range.
Range to	This edit control specifies the upper limit of the selection range.
Add Selection	By this button the new selection is added to an older one.
In Range	By this button all pixels within the specified range are selected.
Out of Range	By this button all pixels out of the specified range are selected.
Cancel	This button aborts the selection process.

Table 17: Settings of the Select Pixel by Value Dialog

4.3.7 Select All

This command selects all pixels of the active document. The selected pixels appear on the screen in the manner, which is adjusted by the control box. All following actions are related to this selection.

4.3.8 Deselect All

This command deselects all pixels of the active document.

4.3.9 Set Value

The Set Value command allows the change of pixel values manual. This function is useful for editing pixel maps (see create pixel map).

After selecting this menu entry the following dialog appears:

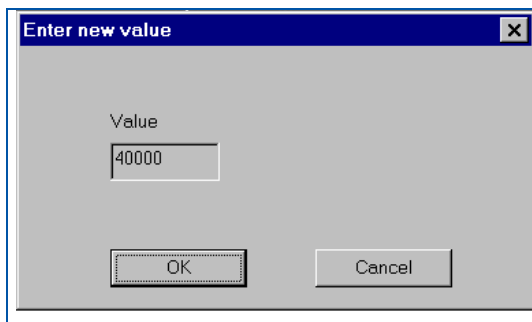


Figure 9: Enter new Value Dialog

- Value** Enter the new value of the pixels.
Ok Press this button to accept the new value.
Cancel This button aborts the selection process.

4.3.10 Select Rect

Selects a rectangle of pixels by coordinates

4.3.11 Create Pixel Map

This command creates a pixel map from a data file.

A pixel map is a normal data document that can be used for a pixel correction. Defective pixels are distinguished from working pixel by their values in the pixel map. Defective pixels have values of 65535, the values of working pixels are different from this value. This menu entry creates an empty data document and all selected pixels of the source document are marked as defect pixels in the new created pixel map.

4.4 Acquire menu

4.4.1 Single Shot

One single image of the detector at the selected frame time (see Timings menu) is acquired by the Single Shot command. The image buffer contains the data of the acquired image.

The Function keys are not available during single shot acquisition, only the ESCAPE key can be pressed to interrupt this action. The Image corrections are Offline available (Offset, Gain or Pixel correction) for details see the chapter "image corrections".

Before the next shot a dialog appears to choose the manner of further processing of the actual image frame (see also Overwrite Data). If the next data's are acquired into a new window, the last actual correction setting is performed during the next image acquisition cycle. To change the current correction setting the Windows menu can be used to close the correction files or the Acquire menu to link other files.



4.4.2 Sequence Command

By this command a defined image sequence at the selected frame time (see Timings menu) is acquired. The image buffer contains the data of all specified acquired images. The Function keys are not available during sequence acquisition, only the ESCAPE key can be pressed to interrupt this action. The image corrections (Offset, Gain or Pixel correction) can only be enabled or disabled before starting the sequence. If the image corrections are disabled it is possible to use the corrections Offline, for details see the chapter "image corrections".

The sequence mode offers different settings to optimize the sequence of images, which can be set in the appearing dialog (see figure). For example the sequence can be stored in one or two buffers or alternatively an average image of the sequence can be build. Details for the settings are displayed in the following table.

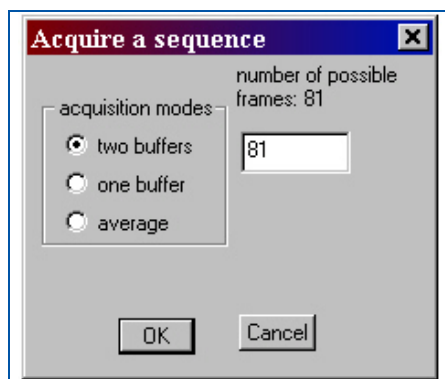


Figure 10: Acquire Sequence Dialog

Two Buffers	Storage of the sequence into two buffers. The transferred data's and the later performed image corrections are separated. Warning: Procedure uses double memory size.
One Buffer	Storage of the sequence into one buffer. Direct acquisition and linked correction into one buffer. Warning: Frames might be lost if slower CPU's are used.
Average	All acquired single images are directly added into one buffer and after acquisition divided by the number of frames, including linked correction files.
Skip frames	During One Buffer and Average acquisition it is possible to skip frames.
Number of frames	Depending on the PC memory and the number of opened windows the number of images to be acquired as a sequence is limited.

Table 18: Settings of the Acquire Sequence Dialog

Before next acquiring of images a dialog appears to choose the manner of further processing of the actual sequence (see also Overwrite Data). If the next data's are

acquired into a new window, the last actual correction setting is performed during the next image acquisition cycle. To change the current correction setting the Windows menu can be used to close the correction files or the Acquire menu to link other files.

Shortcuts



Toolbar:

Keys: CTRL+SHIFT+S

4.4.3 Continuous Command

The **Continuous** command acquires continuously images of the detector at the selected frame time (see Timings menu). The image buffer contains always the data's of the last acquired image cycle. The next acquired image clears the current frame buffer and overwrites the old image.

The call of the **Continuous** command appears in a dialog where the number of images per cycle can be edit. The number of frames per cycle can be selected between one and the displayed maximum number of possible frames, which are depending of RAM and open windows. But for memory reasons it is recommended to use the default number. The number of skipped frames can also be edit in the dialog.

By the Function keys the online corrections can be enabled or disabled (Offset, Gain or Pixel correction). The image presentation can be changed using the View menu. To stop the continuous acquisition the user can press the ESCAPE key or call **End of Acquisition**. The last acquired image is displayed and can be further processed. If more than one frame is in the buffer images of this cycle can be selected by the **Player** command in the View menu. The image presentation can be changed using the View Options.

Before next acquiring of images a dialog appears to choose the manner of further processing of the actual sequence (see also Overwrite Data). If the next data's are acquired into a new window, the last actual correction setting is performed during the next image acquisition cycle. To change the current correction setting the Windows menu can be used to close the correction files or the Acquire menu to link other files.

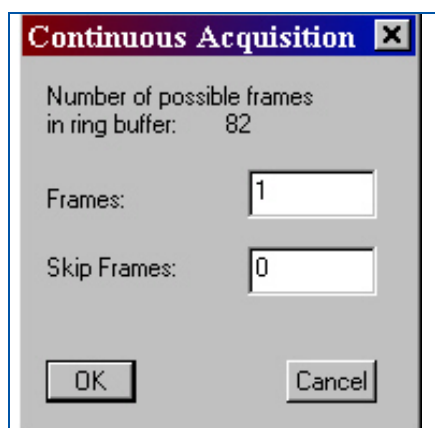


Figure 11: Continuous Acquisition Dialog

Shortcuts



Toolbar:

Keys: CTRL+SHIFT+C

4.4.4 Get Offset Image

The **Get Offset Image** command generates a new Offset Image file of the detector at the selected frame time (see Timings menu). The number of frames to average for the Offset image can be edit in the appearing dialog. The new correction image is used to perform

later the correction of the new acquired images. Please be aware, that a corrected image should not be corrected twice.

See also: Use of the Offset Correction.

The Offset image is automatically linked and used for the next acquired images. To switch between correction and non correction of an running acquisition the Function keys can be used. To stop generally the image correction close the linked correction files by the Window menu commands.

Shortcut: CTRL+SHIFT+G

4.4.5 Get Gain/Offset Image

The **Get Gain/Offset Image** command generates a new Gain-Offset Image file of the detector at the selected frame time (see Timings menu). The number of frames to average for the new Gain image can be edit in the appearing dialog. The new correction image is used to perform later the correction on the new acquired images. Please be aware, that a corrected image should not be corrected twice.

See also: Use of the Gain/Offset Correction.

The Gain/Offset image is automatically linked and used for the next acquired images. To switch between correction and non correction of an running acquisition the Function keys can be used. To stop generally the image correction close the linked correction files by the Window menu commands.

Shortcut: CTRL+SHIFT+G

4.4.6 Get All Offset Images

This command generates a set of new Offset Image files of the detector at **all** available frame times. The number of frames to average for the new Offset image files can be edit in the appearing dialog. The new correction image files are used to perform later the correction on the new acquired images. Please be aware, that a corrected image should not be corrected twice.

The Offset image files are automatically linked concerning their frame time and used for the next acquired images. To switch between correction and non correction of an running acquisition the Function keys can be used. To stop generally the image correction close the linked correction files by the Window menu commands.

4.4.7 Link Offset Correction

The **Link Offset Correction** command loads defined Offset correction files of the detector for the selected frame time (see Timings menu). The linked file is used to perform directly the correction for the actual frame. Please be aware, that a corrected image should not be corrected again. The file is selected by the File Selection dialog box. To acquire Offset correction files see Get Offset Image.

This file is also used for the next acquired images. To switch between correction and non correction of an running acquisition the Function keys can be used. To stop generally the image correction close the linked correction files by the Window menu commands.

4.4.8 Link Gain Correction

The **Link Gain Correction** command loads defined Gain/Offset correction files of the detector for the selected frame time (see Timings menu). The linked file is used to perform directly the correction for the actual frame. Please be aware, that a corrected image should not be corrected again. The file is selected by the File Selection dialog box. To acquire Gain/Offset correction files see Get Gain/Offset Image.

This file is also used for the next acquired images. To switch between correction and non correction of an running acquisition the Function keys can be used. To stop generally the image correction close the linked correction files by the Window menu commands.

4.4.9 Link Gain Sequence Correction

The Link Gain Sequence Correction command loads defined GainSequence/Offset correction files of the detector for the selected frame time (see Timings menu). The linked

file is used to perform directly the correction for the actual frame. Please be aware, that a corrected image should not be corrected again. The file is selected by the File Selection dialog box. To acquire Offset correction files see Get Offset Image. The Gain Sequence correction file must be created with the command Acquire Build GainSequence command.

NOTE: This file is also used for the next acquired images. To switch between correction and non correction of the acquired images the Function Keys can be used or to stop generally the correction close the linked correction files by using the Window menu commands.

4.4.10 Link Pixel Correction

The Link Pixel Correction command loads a defined pixel correction file of the detector. The linked file is used to perform directly the correction for the actual frame. Please be aware, that a corrected image should not be corrected again. The file is selected by the File Selection dialog box.

This file is also used for the next acquired images. To switch between correction and non correction of an running acquisition the Function keys can be used. To stop generally the image correction close the linked correction files by the Window menu commands.

4.4.11 Acquire Build GainSequence

This menu entry opens the "Build Sequence" dialog to create a sequence of offset-corrected bright images to be used with the GainSequence-correction. The selected images will be sorted by median. Beware that the XRD-FGX can handle a sequence with up to 10Frames.

The "Build Sequence" dialog looks like the following image

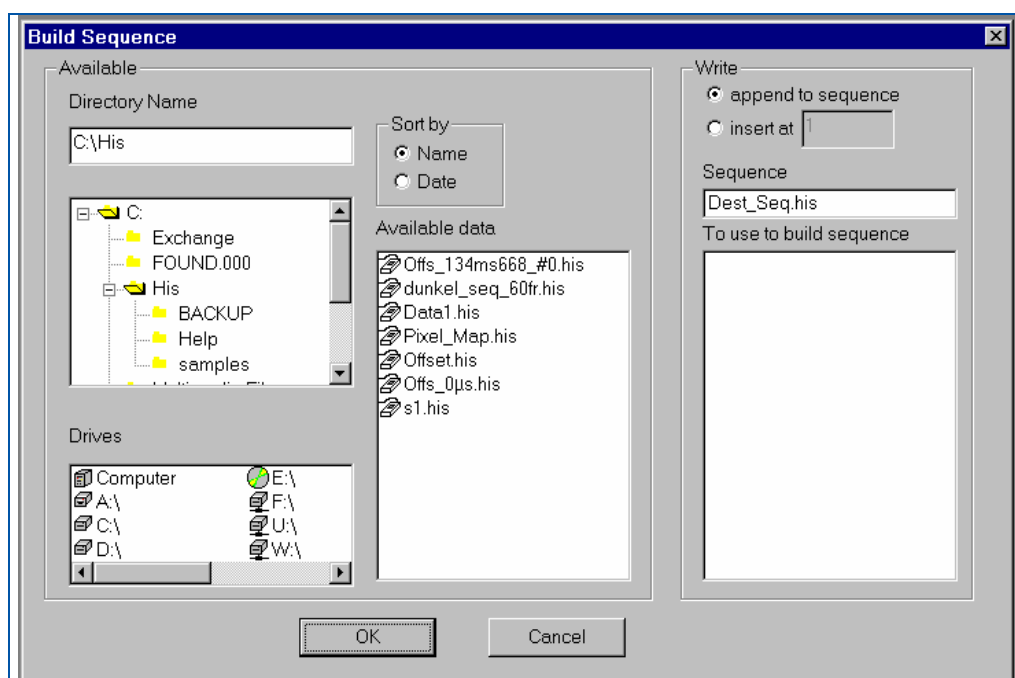


Figure 12: Build GainSequence Dialog

The available group box shows all available drives, directories and files. The currently loaded images are listed under the drive "computer" in the drives list box. To list stored images click on the items of the drive box and the entries of the above tree control that displays the directory structure.

The "write" group provides all controls regarding the destination sequence.

Sort by radio buttons	Change the sorting of the available images by the corresponding radio buttons
append to sequence	Ignored!
insert at button	Ignored!
Sequence edit box	Drag and drop a sequence file from the "available data" box or edit the name. If

	the file name isn't loaded and not available on the storage media a new data window is created and the data are appended.
To use to build sequence box	This list contains the files which will appended to or inserted into the sequence. Drag and drop files from the "available data" list box.

Press the Ok-Button to create a Sequence of the selected files sorted by Median.

4.4.12 Convert to Gain Image

This function creates a new gain image from a dark image file and a bright image file. Compared to the automatic creation of the gain image by Get Gain/Offset Image this routine has the advantage that the user can select a region of interest in the bright image to optimize the gain image regarding best presentation of data in this area. Afterwards the new gain image has to be linked by Link Gain Correction.

The call of Convert to Gain Image comes to a dialog where the dark, the bright and the new gain image has to be named. The available data files can be shown by selecting one edit box and pressing the insert key on the keyboard. In the appearing File Selection dialog the files can be found and entered.

4.4.13 End of Acquisition

This command stops the continuous acquisition. The last acquired image is displayed and can be further processed.

Shortcut: Esc

4.5 Detector

4.5.1 Timings menu

The Timings menu enables the setting of different frame times for the image acquisition of the detector. Eight different frame times are available. TIMING0 is the shortest possible frame time of the specific detector. The detector starts automatically in the first timing.

Example of the default timings menu for the XRD/RID 512-400 A:

(TIMING 0)	134 ms	(Shortest available frame time of the detector.)
(TIMING 1)	200 ms	
(TIMING 2)	400 ms	
(TIMING 3)	800 ms	
(TIMING 4)	1600 ms	
(TIMING 5)	3200 ms	
(TIMING 6)	6400 ms	
(TIMING 7)	12800 ms	

Table 19: Displaying different Timings

4.5.2 Detector Mode

Three different acquisition modes are available. They are called "free running", "external triggered" and "internal triggered".

- The free running mode means that the detector sends out continuously frames according to the selected frame time. This is the default mode.
- The external triggered mode means that the detector sends a frame after triggering by an external pulse and ignores all other incoming trigger pulses until the selected frame time has elapsed. After that the detector can be triggered by a new pulse.
- The internal triggered mode means that each frame time between fastest timing and 5 seconds can be selected and the frame grabber triggers the detector by this frame time.

4.6 View menu

4.6.1 Status Bar

The left area of the status bar shows online information for the menu or Tool Bar entry where the cursor is above or shows the status of an executed command. A check mark appears next to the menu item when the Status Bar is displayed.

For details see the chapter Status Bar.

4.6.2 Toolbar

The Toolbar command displays and hides the Toolbar, which includes buttons for some of the most common commands in XIS, such as File Open. A check mark appears next to the menu item when the Toolbar is displayed.

For details see the chapter Toolbar

4.6.3 Acquisition Bar

The Acquisition Toolbar command displays and hides the Acquisition Toolbar, which includes buttons for some of the most special commands in XIS, such as Acquisition. A check mark appears next to the menu item when the Toolbar is displayed.

For details see the chapter Acquisition Toolbar

4.6.4 LUT - Look-Up-Table

Hides or Shows the LUT - Look-Up-Table.

The Lock-Up-Table represents the currently selected LUT range in a graphic bar within 256 gray levels. The lowest intensity is represented black, the brightest intensity white. The fading uses the 16 bit range for the actual values.

NOTE:

Exceptionally the values for Gain/Offset correction images are represented in a 32 bit integer mode.

4.6.5 Player

This command is only available if sequences are loaded or acquired.

To show the images and to select a specific frame of the sequence the **Next** and **Previous** button are used. In contrast to the step by step representation the **Play** button starts a continuous playing of all images of the sequence.



4.6.6 ZoomBox

The ZoomBox is a utility to zoom into the area located at respective the cursor position.

The zoom can be intensified using the mouse wheel¹. Starting at a size of 101*101 pixels being up-scaled, the zoom level can be changed up to 5*5 pixels maximum zoom.

Below the section showing the scaled pixels the ZoomBox provides other features.

Considering image data these are

Position	Current cursor position.
Value	Pixel value at the cursor position.
Window Size	Size n^2 of the currently zoomed section.
Median	Median of the currently zoomed n^2 section.
Sigma	Sigma of the currently zoomed n^2 section.

In case the document considered is a fault mask provided values are:

Position	Current cursor position.
Value	Fault which is set at the current cursor position.
Window Size	Size n^2 of the currently zoomed section.
Fault Density	Fault Density of the currently zoomed n^2 pixels (which is the ratio of good and fault pixels within the n^2 window).

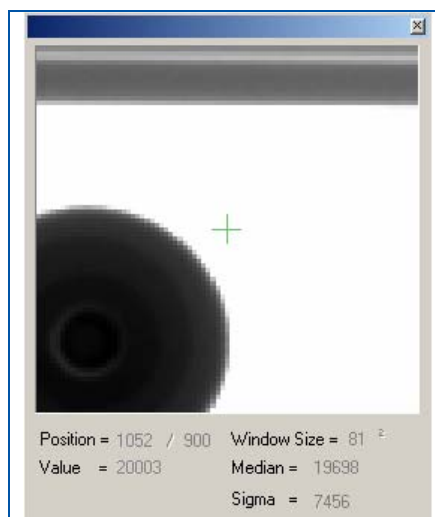



Figure 13: Zoom Box

The ZoomBox can also be activated by clicking the Toolbar button 

¹ During usage of the ZoomBox the mouse wheel functionality for image scrolling is deactivated. The image can still be scrolled by moving the scroll bar at the window rim though.

View Palette

The View Palette command allows the change of the color palette, whereby the palette file format is binary. The first entry is a 32 bit integer that gives the palette version. Currently it's always 100 hexadecimal. The second entry is a 32 bit integer that gives the number of entries. If the graphic display runs in 8 bit mode this number can be smaller or equal to 236. If it is greater only the first 236 entries are accepted. In higher color modes (high and true color) the number of entries can rise up to 65535.

The entries are 32 bit integers as well. The first byte gives the red, the second the green and the third the blue value (see PALETTEENTRY structure in the Win32 SDK). The fourth byte is set to zero.

4.6.7 View Control Box

The View Control Box appears automatically when an image is acquired opened or imported. To hide and recall the Control Box the menu entry **View** in the View menu can be used.

The detector acquires images of 16Bit, but Windows based computers can only display 8Bit. By the View Control Box the interested grey levels can be selected and displayed on the screen. The selection can be done automatically or manually (zooming).

The Control Box contains the following features:

Brightness	Select brightness for image presentation.
Contrast	Select contrast for image presentation.
LUT Range	Select Bright and Dark values for image presentation.
Full Range	This box represents always the minimum (0) and maximum (65535) values in 16 bit range.
Invert	Inverts the actual image presentation.
Track Range	Automatic tracking within a region of interest based on evaluated minimum and maximum value of this region. It is recommendable to use this function in the Continuous Acquisition mode for viewing interesting sample parts under X-ray illumination.
Zoom	If this radio button is checked the program evaluates the minimum and maximum pixel values of the selected region of interest and presents this value range in 256 gray levels.
Equalize	Performs image enhancement by equalizing all gray levels in a region of interest.
Full	Allows to jump into the full presentation mode of the LUT Range.

Table 20: Zoom settings of the Control Box

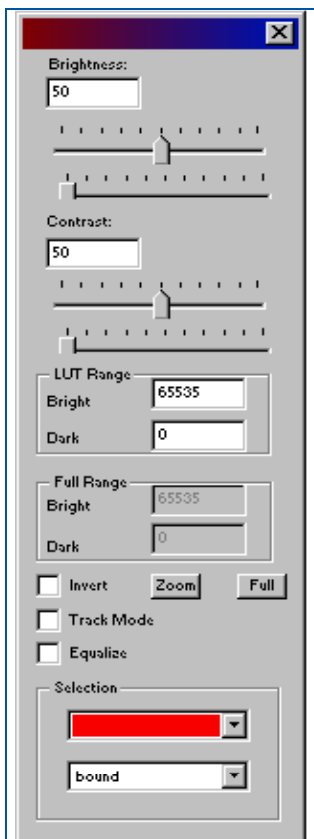


Figure 14: Control Box

To mark **the region of interest** within the image the mouse pointer has to be moved to the starting point and the left button has to be pressed while moving to the end point.

By moving from the upper left to a desired lower right point, the user selects a rectangular shaped window, presented in inverted values.

By moving from lower left to upper right a horizontal line is selected.

By moving from upper right to lower left, a vertical line is selected.

The color and the mode presenting the region of interest can be changed in the listbox on the bottom of the color control window. The different modes are:

Fill	The region of interest is filled with the selected color, it is default
Bound	The region of interest is bounded by the selected color.
And	The selected color will be combined with the corresponding pixel values by a logical and operation within the region of interest.
Or	The selected color will be combined with the corresponding pixel values by a logical or operation within the region of interest.
Hide	Region of interest is hidden.

Table 21: ROI settings of the Control Box

4.7 Window menu

4.7.1 New Window

The New Window command opens a new window with the same contents as the active window. Multiple document windows can be displayed in different parts or views of a document at the same time. If the contents in one window are changed, all other windows containing the same document reflect those changes. When a new window is created, it becomes the active window and is displayed on top of all other open windows.

4.7.2 Cascade

This command arranges multiple opened windows in an overlapped fashion.

4.7.3 Tile Horizontal

This command arranges vertically multiple opened windows in a non-overlapped fashion.

4.7.4 Tile Vertical

This command arranges multiple opened windows side by side.

4.7.5 Window Arrange Icons

This command arranges the icons for minimized windows at the bottom of the main window. If there is an open document window at the bottom of the main window, some or all of the icons displayed below this document window are not visible.

4.7.6 1, 2, ... command

XIS displays a list of open document windows at the bottom of the Window menu. A check mark appears in front of the document name of the active window. Choose a document from this list to make its window active.

4.8 Options menu

4.8.1 Acquisition

This command restarts the sensor initialization and allows to select the initialization type (automatic or manual). After choosing this menu entry, a dialog comes up, which asks to initialize the sensor automatically or manually. The selection of automatic initialization leads to a dialog asking for the mode in which the frame grabber will run. Choosing "yes" runs the frame grabber in Interrupt Mode and "No" runs the frame grabber in Polling Mode. The "Cancel" button starts the XIS without initialization of the detector. If the "Open always" radio button is checked, the XIS opens the requested communication channel regardless if it has been already captured by another process running on the system. It isn't recommended to use this option except for debugging because the XIS can't free all resources in one process that were allocated by another process.

For the automatic initialization, the XIS scans the PCI(X) bus for plugged-in frame grabbers and tries to detect connected detectors. If it recognizes more than one detector, it asks you to select a default one. The initialization process is ready if the message "initialization successful" appears in the status bar.

During the manual initialization, the XIS asks for the communication channel to open, more than one are possible. The communication channel can be set by the hex-switch of the frame grabber board, if only one frame grabber inside the default channel is zero. The following dialog expects parameters to initialize the sensors connected to these channels:

4.8.1.1 Size of the detector

Define upper and bottom left and right X, Y dimension of the detector.

4.8.1.2 Sorting

The following sorting schemes are possible (see also sorting overview):

- No Sort
- RID 128
- RID 256
- RID 128-400
- RID 1024-100 or MX1024
- XRD 0820/40
- XRD/RID 512-400
- XRD 1640 A / 1620 AJ
- XRD 1680 A
- XRD 162X AM/AN
- XRD 16XX AN CS

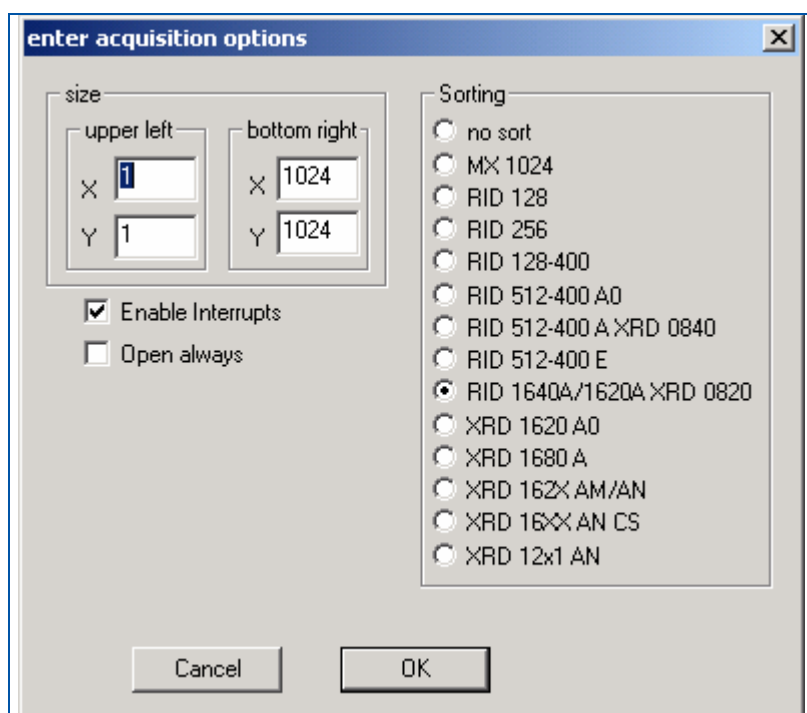


Figure 15: Dialog for the Acquisition Options

4.8.1.3 Enable Interrupts

The setting of different interrupt sources allow an efficient data transfer between I/O board and the memory of the PC (for further explanation see interrupt sources). If no interrupts are enabled the detector is running in polling mode. It is recommended to use the interrupted mode for an frame synchronization.

4.8.1.4 Open always

This option opens the requested communication channel regardless if it has been already captured by another process running on the system. It isn't recommended to use this option except for debugging because the XISL can't free all resources in one process that were allocated by another process. If you initialized more than one sensor the XIS asks now a default detector to that all the further instructions correspond.

4.8.2 View

The View command comes to the “Enter View Options” dialog, where the presentation of the actual window can be optimized.

Toolbar: 

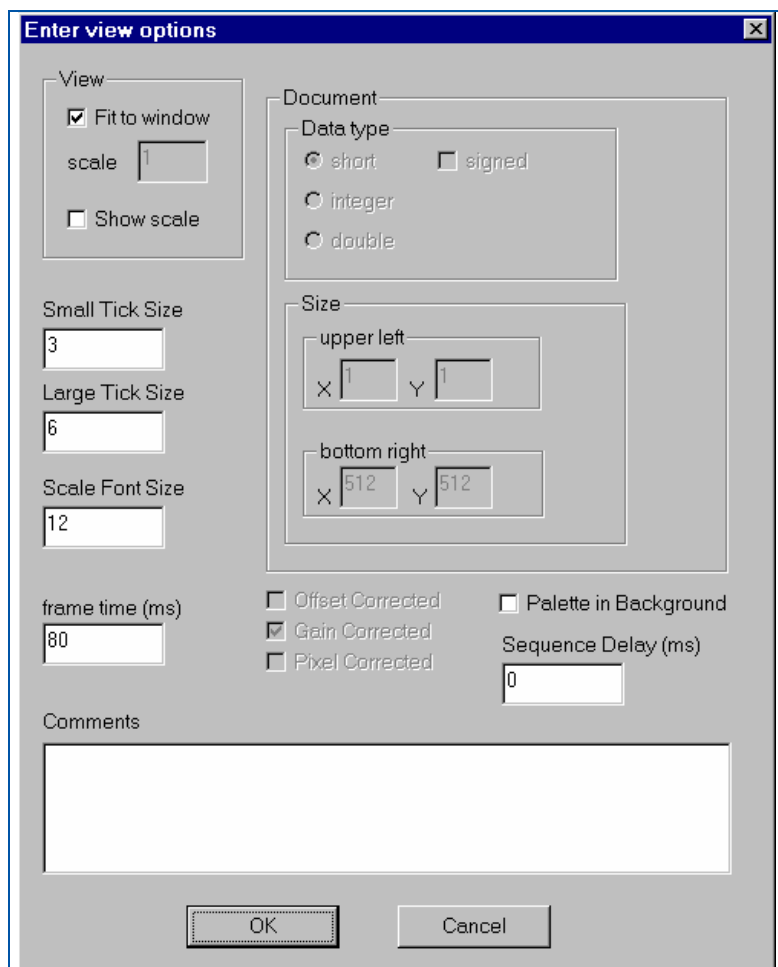


Figure 16: View Options Dialog

View - Fit to window	Defines the scale of the image window.
View - Show Scaling	To be switched ON for scaling the edges of the image Small Tick2 (default) Large Tick4 (default) Scale Font 12 (default)
Document - Data type	Information of the used data type. Short Integer Double Signed
Document – Size	Information on upper and bottom left and right X, Y dimension of the detector.

Table 22: Settings of the View Options Dialog

4.8.3 Options Active Sensor

The appearing dialog shows the connected sensors in a list box. The active sensor is highlighted. The sensors are identified by the frame grabber type they are plugged in and a unique number. To change the active sensor for acquisition another sensor can be selected from the list box.

Toolbar: 

4.8.4 Options Sensor

This menu entry comes to the “Sensor Options” dialog contains information about the active sensor.

- The communication channel edit box gives information about the frame grabber board which is connected to the active sensor.
- In the correction directory the correction files are stored by the Save correction files command. If the Auto Correction Load radio button is checked suitable correction files are loaded from the correction directory at startup and if a different timing is selected.

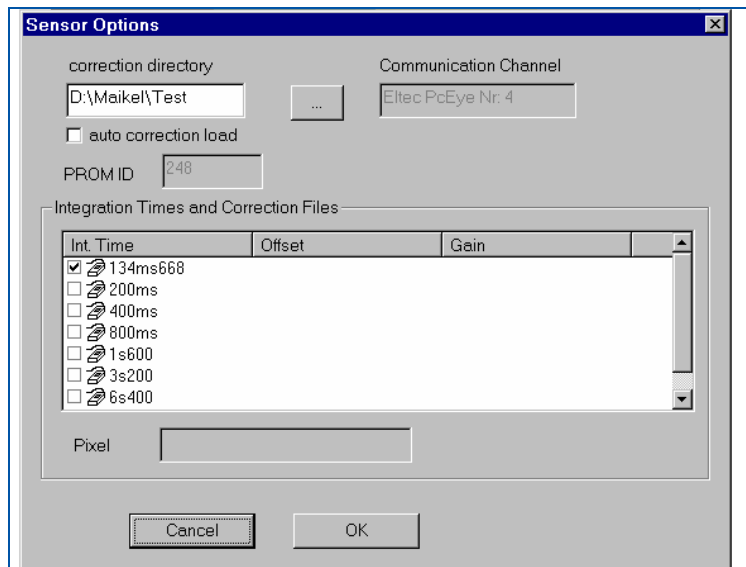


Figure 17: Sensor Options Dialog

- The PROM ID is the identifier of the actual used detector PROM. This information is important to get support at hardware problems concerning the detector or the frame grabber.
- In the group “Integration Times and Correction Files” all available integration times are listed. The selected one is checked in the list control. If correction files are attached for the different timings there will be corresponding entry in the list control. If a pixel correction file is attached it is displayed in the “Pixel” edit box.

4.8.5 Detector Options

This menu entry comes to the “Set Detector Options” dialog containing information about the sensor settings and allows changing them.

- ▶ Binning-mode, Trigger-mode, Gain, Timing and Special TriggerMode are changeable as described in the Section 6-“Details for the Hardware”.
- ▶ Use ‘Data Delivered on Demand Options’ to change the additional delay in DDD-Trigger Mode

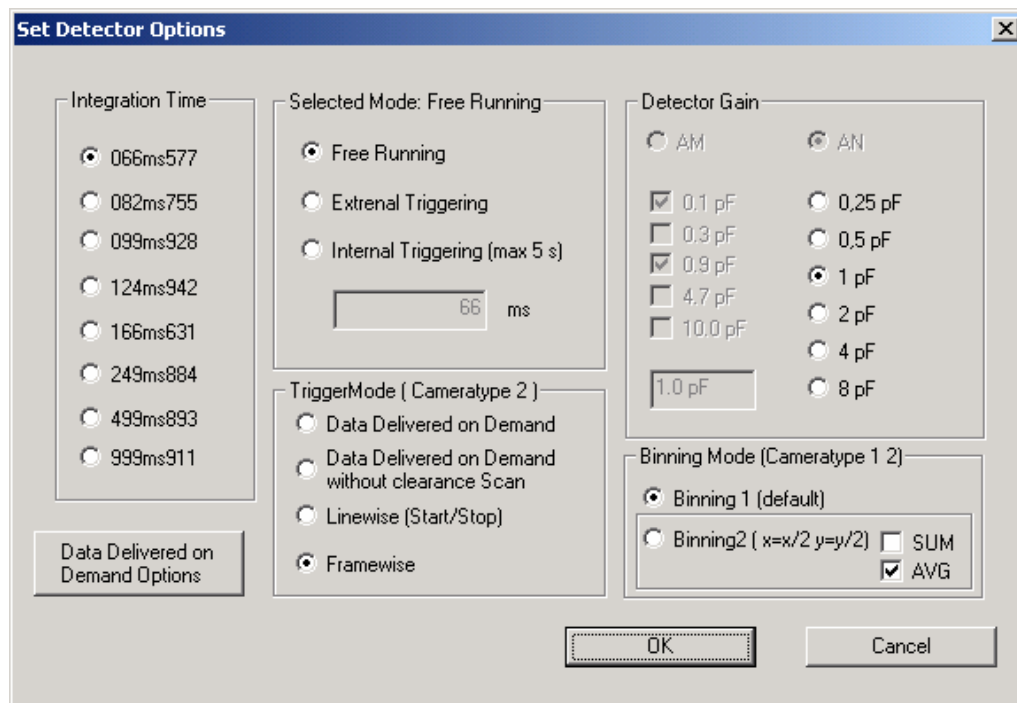


Figure 18: Set Detector Options

4.9 Help

4.9.1 Contents

Displays the contents of XIS help file.

4.9.2 Index

This command displays the main screen of Help. The main screen offers step-by-step instructions for using XIS and various types of reference information. Once opened Help a call of the Contents button returns to the main screen.

4.9.3 Using Help

This command displays instructions about using Help.

4.9.4 About

Displays the version number and copyright notice of XIS.

4.10 Toolbar



The toolbar is displayed across the top of the application window and below the menu bar. The toolbar provides quick mouse access to many tools used in XIS. The Toolbar can be hide or displayed by the command Toolbar in the View menu (ALT, V, T).








Click	Application
	Open a new document.
	Open an existing document. XIS displays the Open dialog to find and select the files.
	Save the active document or template with its current name. If the file is not named XIS displays the Save As dialog box.
	Print the active document.
	Copy the selection to the clipboard.
	Insert the contents of the clipboard at the insertion point.
	The Context Help obtains help for some commands. When the Toolbar's Context Help button is active, the mouse cursor will change to an arrow and question mark. The help topic shows online information for the menu or Tool Bar entry where the cursor is above Shortcut SHIFT+F1

Table 23: Windows Toolbar

4.11 Acquisition Toolbar



The toolbar is displayed across the top of the application window and below the menu bar. The toolbar provides quick mouse access to many tools used in XIS. The Toolbar can be hide or displayed by the command Toolbar in the View menu (ALT, V, A).







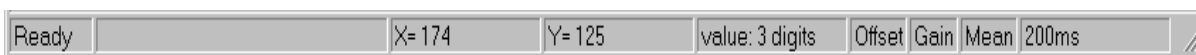
Click	Application
	Starts the continuous Acquisition. Shortcut: CTR+SHIFT+C
	Opens the Sequence Dialog
	Starts a Single Shot
	Opens the Player Dialog.
	Opens the View Dialog
	Opens the Active Sensor Dialog

Table 24: Acquisition Toolbar

4.12 Status Bar



The status bar is displayed at the bottom of the XIS window. To display or hide the status bar the Status Bar command in the View menu can be used.

The left area of the status bar shows online information for the menu or Tool bar entry where the cursor is above or shows the status of an executed command.

The right areas of the status bar give extended information regarding acquisition status, pixel values and frame times.

Indicator	Description
Box 1:	Describes the main status of the software. (Online Help)
Box 2:	Warnings by using detector and its messages.
Box 3:	Selected X-value of the detector array.
Box 4:	Selected Y-value of the detector array.
Box 5:	Actual value in digits between 0 - 65535.
Box 6:	Marker for Offset ON/OFF.
Box 7:	Marker for Gain/Offset ON/OFF.
Box 7a:	Marker for GainSeq/Offset ON/OFF.
Box 8:	Marker for Pixel correction ON/OFF.
Box 9:	Selected frame time.

Table 25: Status Bar

4.13 Title Bar



The title bar is located along the top of a window. It contains the name of the application and document. To move the window, drag the title bar. Note: You can also move dialog boxes by dragging their title bars.

A title bar contains the following elements:

- Application Control-menu button
- Document Control-menu button
- Maximize button
- Minimize button
- Name of the application
- Name of the document
- Restore button

4.13.1 Scroll bars

Displayed at the right and bottom edges of the document window. The scroll boxes inside the scroll bars indicate your vertical and horizontal location in the document. You can use the mouse to scroll to other parts of the document.

4.13.2 Size command (System menu)

This command displays a four-headed arrow to size the active window with the arrow keys. This command is unavailable if the window is maximized.



After the pointer changes to the four-headed arrow:

1. Press one of the DIRECTION keys (left, right, up, or down arrow key) to move the pointer to the border you want to move.
2. Press a DIRECTION key to move the border.
3. Press ENTER when the window is the size you want.

Shortcut

Mouse: Drag the size bars at the corners or edges of the window.

4.13.3 Move command (Control menu)

This command displays a four-headed arrow to move the active window or dialog box with the arrow keys. This command is unavailable if the window is maximized.



Shortcut

Keys: CTRL+F7

4.13.4 Minimize command (application Control menu)

Use this command to reduce the XIS window to an icon.

Shortcut



Mouse: Click the minimize icon on the title bar.

Keys: ALT+F9

4.13.5 Maximize command (System menu)

Use this command to enlarge the active window to fill the available space.

Shortcut



Mouse: Click the maximize icon on the title bar; or double-click the title bar.

Keys: CTRL+F10 enlarges a document window.

4.13.6 Next Window command (document Control menu)

Use this command to switch to the next open document window. XIS determines which window is next according to the order in which you opened the windows.

Shortcut

Keys: CTRL+F6

4.13.7 Previous Window command (document Control menu)

Use this command to switch to the previous open document window. XIS determines which window is previous according to the order in which you opened the windows.

Shortcut

Keys: SHIFT+CTRL+F6

4.13.8 Close command (Control menus)

Use this command to close the active window or dialog box.

Double-clicking a Control-menu box is the same as choosing the Close command.



Note: If multiple windows opened for a single document, the Close command on the document Control menu closes only one window at a time. All windows can be closed at once with the Close command on the File menu.

Shortcuts

Keys: CTRL+F4 closes a document window

ALT+F4 closes the current window or dialog box

4.13.9 Restore command (Control menu)

Use this command to return the active window to its size and position before you chose the Maximize or Minimize command.

4.13.10 Switch to command (application Control menu)

Use this command to display a list of all open applications. Use this "Task List" to switch to or close an application on the list.

Keys: CTRL+ESC

Dialog Box Options of the Switch command

Task List	The next application can be selected.
Switch To	Makes the selected application active.
End Task	Closes the selected application.
Cancel	Closes the Task List box.
Cascade	Arranges open applications as overlapped windows whereby the title bars are visible. This option has not effect on applications reduced to icons.
Tile	Arranges open applications into not overlapping windows. This option has no effect on applications reduced to icons.
Arrange Icons	Arranges the icons of all minimized applications across the bottom of the screen.

Table 26: Switch Command Dialog

4.14 Standard Dialogs

4.14.1 File Selection Dialog

This dialog shows all available data files stored on hard disk or loaded in the RAM. This dialog appears by the commands Link Offset Correction, Link Gain Correction and Link Pixel Correction.

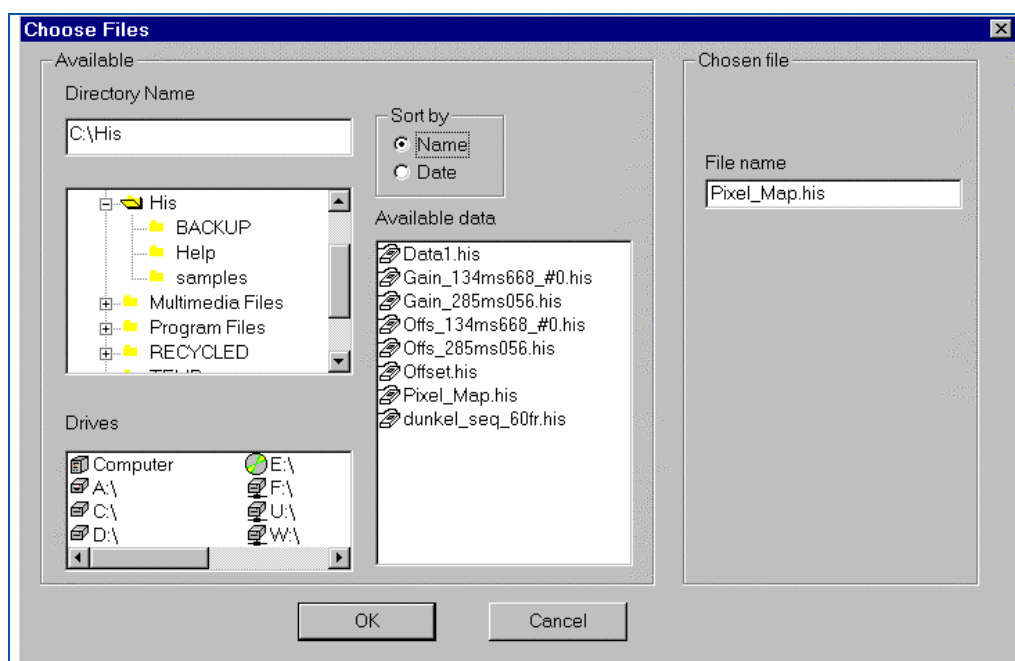


Figure 19: Choose File Dialog

The available group box shows all available drives, directories and files. The currently loaded data's are listed below the drive "computer".

By pressing one of the "sort" radio buttons one can influence the way the available data are sorted in the "available data" list box.

Double click the selected file from the "available data" box and the file will be displayed in the "Chosen file" box. The process will be executed by pressing the "Ok" button or the "enter" key

4.14.2 Overwrite data dialog

This dialog appears if new data has to be acquired into an unsaved window.

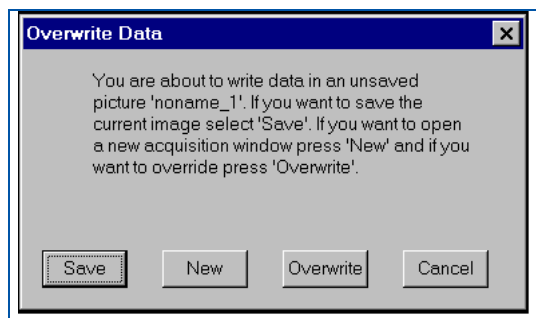


Figure 20: Overwrite Data Dialog

Save	Saves the current data and writes the new data in the window buffer
New	A dialog appears to define the new document type
Overwrite	Overwrites the existing data without any saving
Cancel	Cancels the dialog

4.14.3 Choose Directory Dialog

This dialog appears if a directory has to be selected.

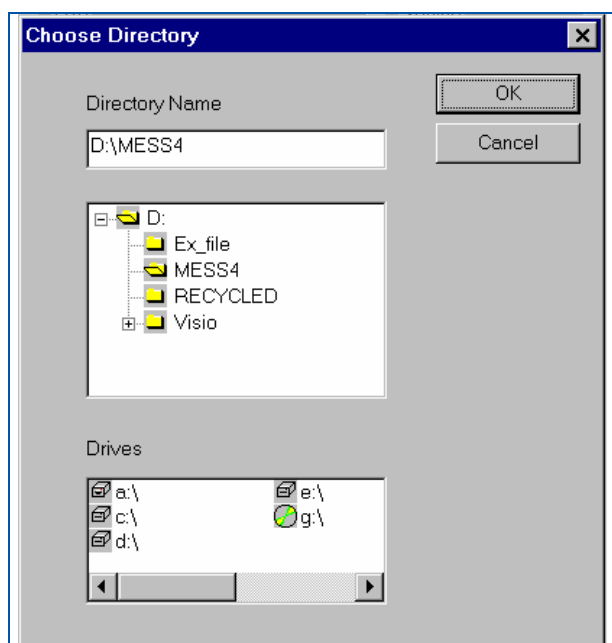


Figure 21: Choose Directory Dialog

directory name	This edit line specifies the selected directory. It's connected to a directory tree and all current available folders are listed.
drives	All current available drives are listed here. If a drive is selected the directory name and its directory tree is actualized.

5 Application

5.1 Mathematical Expressions

This dialog enables the easy input of mathematical expressions for image processing.

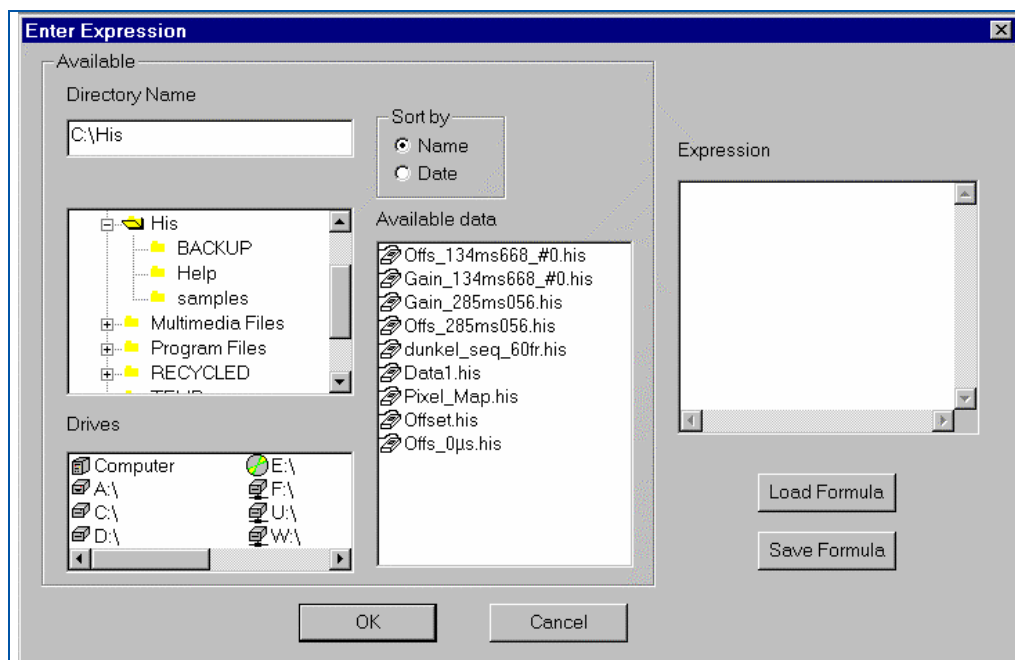


Figure 22: Mathematical Expressions Dialog

The available group box shows all available drives, directories and files. The currently loaded data's are listed below the drive "computer". To list stored images click on the items of the drive box and the entries of the above tree control displaying the directory structure. By pressing one of the "sort" radio buttons one can influence the way the available data are sorted in the "available data" list box.

The mathematical expression can be added in the "Expression" edit box. Images can simply inserted by drag and drop files from the "available data" list to the edit box. The file name is inserted at the current cursor position. Soft line breaks can be set by pressing the "enter" key while holding the "Control" key.

5.1.1 Parsing expression:

Reserved symbols are:

+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
(open parentheses
)	close parentheses
=	Assign operator
...	dots used to specify a range of frames in a data sequence
[open square brackets to specify a range of frames in a data sequence
]	close square brackets to specify a range of frames in a data sequence
(USHORT)	cast operator, converts an arbitrary data type to the required data type (unsigned short)
SUM	summation function
AVG	Average function.
ABS	Absolute function

Table 27: Parsing symbols

+ operator:

If both operands are numbers the result is also a number. If one operand is a number and the other is an image then the number is added to every data entry. If both operands are images the data are added pixel by pixel.

- operator:

If both operands are numbers the result is also a number. If one operand is a number and the other is an image then the number is subtracted from every data entry. If both operands are images the data are subtracted pixel by pixel.

*** operator:**

If both operands are numbers the result is also a number. If one operand is a number and the other is an image then the number is multiplied with every data entry. If both operands are images the data are multiplied pixel by pixel. A matrix multiplication is not performed!!!

/ operator

If both operands are numbers the result is also a number. If one operand is a number and the other is an image then the number is divided by every data entry. If both operands are images the data are divided pixel by pixel. If a division by zero is recognized the parser returns with an error message.

= operator

If the result of an arithmetic expression is an image then this result has to be assigned to a result window.

For instance: `Dest.his = data1+40000`

is a valid expression. To the data entries of "data1" a value of "40000" is added and the result is written into "Dest.his".

The expression `Dest.his = 40000`

returns an error because the result is a number.

The expression `data1+50000-data2`

also causes a parse error because the result is an image and the assignment to a result window is absent.

(Type) (cast operator)

This operator converts an arbitrary data format into the required ones. Type can be one of the following words:

SHORT	signed 16 bit integer
USHORT	unsigned 16 bit integer
LONG	signed 32 bit integer
ULONG	unsigned 32 bit integer
DOUBLE	8 byte floating point number

Table 28: Type Operators

`Dest = (ULONG) (data1+data2/16-SUM(FRAMES, data5[2...4]))`

`[a...b]` operator (range operator)

Syntax: `Data.his[a...b]`

This operator returns a sequence of b-a frames extracted from the source sequence (here `Data.his`) starting at frame a and ending at frame b.

`Dest = data1[3...7]`

SUM function:

The sum function derives the sum of numbers, the sum of different images, the sum of rows or the sum of columns of different images. The entries in this function are separated by comma. The first parameter has to be one of keywords.

FRAMES	Derives the sum of different frames.
ROWS	Derives the sum of different rows. If the following arguments represent more then one frame, the result is a sequence of frames containing the summed rows of the frames.
COLUMNS	Derives the sum of different columns. If the following arguments represent more then one frame, the result is a sequence of frames containing the summed columns of frames.

Table 29: SUM - function parameters

If an entry is a sequence of several frames the sum of the sequence is evaluated and after that the resulting images are summed. Valid expressions are for instance:

Dest.his = SUM(FRAMES, data1[3..6], data2, 40000, -30000)

Dest.his = SUM(ROWS, data1[3..6], data2, 40000, -30000)

Dest.his = SUM(COLUMNS, data1[3..6], data2, 40000, -30000)

AVG function:

The average function derives the average of numbers as well as the average of different images or both. The entries in this function are separated by comma. If an entry is a sequence of several frames the average of the sequence is evaluated and after that the resulting images are averaged. Valid expressions are for instance:

Dest.his = AVG(data1, data2, 40000, -30000)

ABS function:

The absolute function derives the absolute values of numbers or data entries. A valid expression is for instance:

Dest.his = ABS(data1)

General remarks:

All data or numbers are converted to floating point numbers before the expression is evaluated. The result images are representations of floating point data's.

5.2 Image Correction

The X-Ray Detectors (**XRD**) need an Offset correction to take into account the dark current of each pixel. In particular, during the warm-up phase of the detector, the Offset is not stable and during this time period the detector use is not recommended. During operation it is recommended to refresh periodically the Offset.

Additionally, a Gain correction is necessary to homogenize differences in pixel sensitivities or to take into account the X-ray beam illumination, therefore it is very important that the whole image area is illuminated homogeneously. The Gain correction should be carried out in an optimum dynamic range of the sensor (70-80 % of the full scale range FSR) or in the dynamic range of interest. The radiation intensity used to create the gain image can depend on the application, e.g. if the typical grey level is about 10.000 ADU and the remaining area is saturated, it is recommended to use a gain image created at 10.000 ADU. The use of an Offset and Gain calibration eliminates offset dependency and therefore any stored Gain correction file can be used for a specific frame time for longer time periods.

The image performance can be enhanced by using the Multiple Gain Correction. For each dynamic range of interest a separate offset corrected and averaged gain image is used as an interpolation point. The maximum number of interpolation points depends on the installed computer memory and the **XRD FGX Opto** frame grabber can operate with up to 10 gain images. It is important that each gain image is completely and homogeneously illuminated. The radiation intensity used to create the gain images can depend on the application, e.g. if the typical grey level is between 5.000 ADU and 10.000 ADU and the remaining area is saturated, it is recommended to use gain images created at 5.000 ADU, 10.000 ADU and 45.000 ADU.

The Pixel correction allows a ‘software repair’ of underperforming pixels to enhance image quality. Underperforming pixel values are replaced with the averaged value of the surrounding eight adjacent pixels where underperforming pixels are not used. The pixel correction is only performed on specific pixels, mapped in the file PXMLMASK.HIS. The delivery package includes the PXMLMASK.HIS file for the specific detector, the user can also generate their own correction file. Please be aware that the number of pixels used for the mean correction should be minimized. The pixel correction procedure requires CPU time (**XRD FG** Frame Grabber) and depending on its speed, a slower presentation of the acquired images on screen might occur in relation to the selected timing mode of the system. The main screen of the **XIS** software sends out a warning message displayed on the screen if all of the acquired images are not accepted by the computer.

All Corrections are very similar for the **XRD FG** and **XRD FGX Opto** Frame Grabbers. All procedures can be used in the same manner and the XRD driver will automatically select the software or hardware correction.

5.2.1 Use of the Offset Correction

The offset correction of images is recommended to eliminate the influence of pixel dark currents in the acquired image. To get an Offset correction file the following steps have to be performed:

1. Select the desired integration time, readout mode and gain setting.
2. Switch off the X-ray source so that the detector only transfers its "dark image".
3. Wait a few seconds until the detector achieves an equilibrium.
4. Start the Get Offset Image. / Start All Offset Images.
5. Select a number of frames.
It is recommended to use between 20 to 100 frame cycles which will be averaged. The averaged image is qualified as the new Offset Image of the selected frame time and automatically linked to later acquired images.
6. Control the new acquired image using the Options/View command and/or Brightness, Contrast or LUT range.
7. The Offset correction file can be saved if desired.

Note: A warning appears if the program is left without saving new acquired Offset correction files.

The Offset correction should be repeated periodically. In particular during the warming-up period of the system, the dark current of the pixels may change considerably.

To interrupt the procedure the <ESC> key can be used.

NOTE: If the item **Get All Offset Images** is used, step 4 is automatically performed for all available frame times. Please check the total time necessary before selecting the number of frames to avoid longer waiting periods.

5.2.2 Use of the Gain/Offset Correction

The Gain/Offset correction of images is recommended to eliminate the influence of pixel sensitivities and influences of the used X-ray source in the acquired image. To get a Gain/Offset correction file the following steps have to be performed:

1. Select the desired integration time, readout mode and gain setting.
2. Acquire a new Offset correction image.
3. Switch on the X-Ray source and control the brightness of the acquired image in the desired read out settings. The detector's acquired intensity should be between 70-80 % of FSR or in the range of the ROI. The gain intensity depends on the application, but the whole image area should be illuminated homogenously.
4. Start Get Gain/Offset Image.
5. Select a number of frames.
It is recommended to use between 20 to 100 frame cycles which will be averaged. The averaged image is qualified as the new Gain/Offset Image of the selected frame time and automatically linked to later acquired images.
6. Control the new acquired image by using the Options/View command and/or Brightness, Contrast or LUT range.
7. Store the Offset correction file if desired.
Note: A warning appears if the program is left without saving new acquired Offset correction files.

To interrupt the procedure the <ESC> key can be used.

NOTE: The Gain image is automatically Offset corrected with the currently linked Offset correction file. To get best quality of the correction file, please perform an new Offset correction before starting Gain/Offset correction.

5.2.3 Use of the Multiple Gain Correction

The Multiple Gain Correction is recommended to eliminate the influence of pixel sensitivities and influences of the used X-ray source in the acquired image. To get a Multiple Gain Correction file the following steps have to be performed:

1. Select the desired integration time, readout mode and gain setting
2. Acquire a new Offset correction image.
8. Switch on the X-Ray source and control the brightness of the acquired image in the desired read out settings. The detector's acquired intensity should be in the range of the ROI. the whole image should be illuminated homogenously.
3. Start the Acquire Sequence
4. Select a number of frames and the average mode.
It is recommended to use between 20 to 100 frame cycles which will be averaged.
5. Control the new acquired image by using the Options/View command and/or Brightness, Contrast or LUT range.
6. Store the Offset corrected bright image
7. Repeat the steps 3-7 for each intensity of interest.
8. Create a Gain-Sequence with Acquire/Build Gain Sequence
9. Start a new acquisition.
10. Link the created Gain-Sequence file with Acquire/Link Gain Sequence
11. Store the Offset correction file if desired.

Note: A warning appears if the program is left without saving new acquired Offset correction files.

5.2.4 Use and generation of the Pixel Correction

The Pixel Correction of images is recommended to eliminate the influence of underperforming pixels of the detector in acquired images. To get an Pixel correction file the following steps have to be performed:

1. Select the desired integration time, readout mode and gain setting.
2. Link correction files.
3. Switch on the X-ray source and in the continuous mode, control the brightness of the acquired image. The detector's acquired intensity should be between 70-80 % of its maximum signal.
4. Start an image acquisition as for the Get Gain/Offset Image (no sample in front of the detector).
5. Control the new acquired image by using the Options/View command and/or Brightness, Contrast or LUT range.
6. The window should show an homogenous corrected image. Intensity deviations are a sign of not fully working pixels.
7. Change the x-ray source such that the detector's acquired intensity should be about 50 % of its maximum signal.
8. Go to Select by Value in the Edit Menu.
9. Enter desired range of good pixels (e.g. 15000-45000 out of 0-65535)
10. Select "Out of range" button. (All selected pixels are marked.)
11. Call Create Pixel Map in the Edit Menu.
12. The Pixel Map is created and can be stored as new PXMLMASK.HIS.

NOTE: The new PXMLMASK.HIS is automatically linked to new acquisitions and the acquired start-up image (see 4.) is also corrected.

5.2.5 Correct already acquired images

It is possible to correct already acquired uncorrected images. Select the desired image by the Window Command and use one of the Link Commands (Link Offset Correction, Link Gain Correction, Link Gain Sequence Correction or Link Pixel Correction). The active image is automatically corrected.

These setting are also used for the next acquisitions.

NOTE: It is not recommended to close linked correction files during a running acquisition. This can lead in a closing application.

5.3 Acquisition Control Modes

Three different acquisition control modes are available. The **Free Running** mode is the default mode which means that the detector sends out continuously frames according to the selected frame time. The **External Trigger** mode means that the detector sends out a frame after triggering by an external pulse and ignores all other incoming trigger pulse until the selected frame time has elapsed. After that the detector can be triggered by a new pulse. Details of the trigger pulse are described in the chapter External Trigger. The third mode is the **Internal Trigger** mode. In this case a fixed pulse frequency between the fasted free running mode of the detector and 5 seconds in steps of 1 μ s can be selected and these pulses are sending via frame grabber board to the detector.

The control mode can be selected in the submenu **Mode** and the integration time can be selected in the submenu **Timings** of the menu **Detector**. The selected integration time and mode are marked by a check mark on the right side of these items. The trigger pulse can be send as well to the PC frame grabber boards **XRD FG** and **XRD FGX Opto** as to the detector (see: External Trigger).

The trigger modes are recommended if a pulsed x-ray source is used. If the x-ray pulse appears during the readout time of the detector the information are split into two frames. Also if these frames are summarized there could be artefacts which are not correctable. The trigger mode realizes an expose during the delay of the readout structure.

To start external triggering the following steps have to be performed

1. Connect the trigger cable with the detector or with the frame grabber board

2. Power on the detector
3. Startup **XIS**
4. Select an initialization of the detector (Interrupt or Polling Mode)
5. Select the desired integration time (Timing Menu)
6. Select the **External Trigger Mode**
7. Start the get correction image commands (Offset and Gain/Offset)
8. Send trigger pulses
9. Link the pixel correction if desired
10. Start the desired acquisition mode (Continuous or Sequence)
11. Send trigger pulses

In case of internal triggering the steps are similar to the free running mode. To start the internal trigger mode the mode has to be selected and the frequency has to be inserted in the appearing dialog. After that the detector sends out frames in the desired frequency and the frames can be acquired continuously, in a sequence or as a single shot.

Note: In the external trigger mode the detector is waiting for a new acquisition until the trigger pulse is sent. During the desired frame time a new trigger pulse has no effect. The correction files have to be created with the same frequency of trigger pulses for best results.

Example: If the detector is set to the integration time 400ms and the trigger pulse is sending every 200ms, the detector nevertheless runs with 400ms. But if the pulse appears every 450ms the detector runs with 450ms. The selected integration time should always below the desired period time of the trigger pulse frequency. The lower limit is the shortest free running timing (Timing 0) which can be selected.

5.4 Warning table by using the detector and its status

Warning: You are loosing frames

Based on CPU speed and used correction mode, the monitor can not present all received images from the detector.

=> Change to a longer integration time per frame or use less on-line corrections.

Black or white value out of range

If the user selects values out of the allowed range of 0 - 65535 digits. This is not allowed.

Board initialization failed

Starting the software, the detector could not be initialized.

=> Check power cords and interface cables and restart Acquisition in the Options menu.

Board initialization successful

The I/O board was successfully initialized, continue with Acquiring images.

Not all functions available

This message appears if no detector or no I/O board can be detected. The XIS software can be used for image presentation of stored images and further processing of these images.

In the manual initialization of the detector some additional features are also not available.

Acquisition done

This message appears if a started acquisition of images is done. The user can continue using further XIS commands.

Ready

This message appears if one of XIS commands is done. The user can continue using further XIS commands.

6 Details for the Hardware

6.1 Readout schema

6.1.1 Free Running

The first timing (Timing 0) of each detector is the fastest readout time. This means that the detector needs a minimum of 134.668 ms (XRD512-400 AL1) for one frame. Each pixel is readout every 134.668 ms and during this time the pixel collects also radiation. For details of the readout schema see the chapter Sorting. The structure of higher timings is a first readout and a following delay. The time of delay is the time of the selected integration time minus the time of the first timing (Timing 0). As an example the timing two of the XRD512-400 AL1 is 400 ms, this means the first 134.668 ms is a readout of the detector and the following 265.332 ms is a delay and the detector is only integrating radiation.

New fast detector versions like XRD 1620 AJx have different row types for the timings to enhance the image quality at lower speed. For more details concerning the different readout timings see the detector manual or can be calculated as described on page Description of Hardware Header. In these cases the detector uses the whole frame time for readout. To realize a delay using the fastest readout the internal trigger mode can be used with a frame time of the selected timing plus delay time. For more details see the paragraph Internal Trigger.

Note:

If a pulsed x-ray source is used it is recommended to expose during the delay of the detector. If the x-ray pulse appears during the readout time the information is split into two frames. Also if these frames are summarized there could be artefacts which are not correctable. To realize an exposure during the delay the detector allows the triggering of the x-ray source and the detector itself.

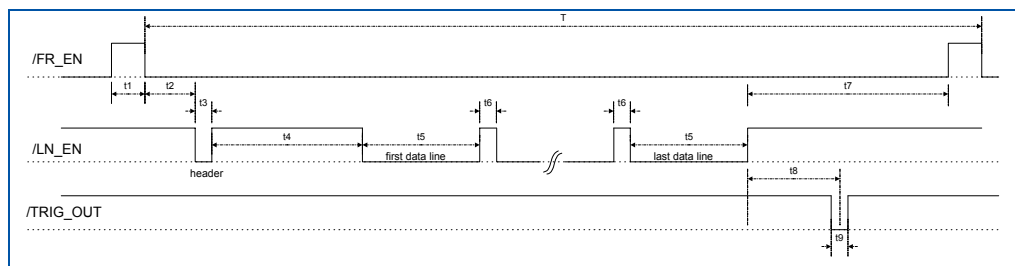


Figure 23: General timing diagram in continuous mode

6.1.2 External Trigger

Triggering the detector is the attempt to synchronise the detector to other devices e.g. x-ray sources having their specific schemes of radiating x-ray pulses. The current mode is triggering the detector on a frame by frame base which means that the detector send a frame after triggering by an external pulse and ignores all other incoming pulses until the selected frame time has elapsed. After that the detector can be triggered by a new pulse. In order to trigger the detector a 20µs wide low active trigger pulse has to be transmitted to the device. The trigger signal has to be generated externally and can then be connected to either pin one of the 7-pin round connector (/TRIG_IN) located directly at the detector device or connected to the sub-click located on the rear side of the **XRD-FG** (/TRIG_IN converted to RS422 signal as /FR_SYNC) respectively the D-Sub connector of the **XRD-FGX Opto** (/TRIG_IN as LVDS signal converted to /FR_SYNC).

Trigger pulses are accepted from both sources. Prior to this the detector has to be set per command into the external trigger mode. The waveform of the trigger pulse as shown in chapter 'Trigger-modes' describes the triggering mode on a frame by frame base. The period of the trigger waveform determines the integration. The /TRIG_OUT signal with a pulse width of 62.5 ns indicates the start of a new frame and can be used to synchronise the x-ray source.

6.1.3 Internal Trigger

The internal trigger mode works similar to the external trigger mode and is also triggering the detector on a frame by frame base which means that the detector send a frame after triggering by an external pulse and ignores all other incoming pulses until the selected frame time has elapsed. After that the detector can be triggered by a new pulse.

The trigger pulse is supported by the frame grabber and is a fixed pulse frequency between the fastest free running mode of the detector and 5 seconds in steps of 1µs. The frame grabber sends the signal /FR_SYNC over the XRD Interface Bus or the Optical Interface to the detector. The /TRIG_OUT signal with a pulse width of 62.5 ns indicates the start of a new frame and can be used to synchronise the x-ray source.

6.1.4 How to use the internal trigger mode

The following method describes how the internal trigger can be used to implement different integration times or to use one of the readout schemes with an customized delay. If the interface is connected and the power is switched on the detector runs in the first free running mode. To use the internal trigger mode the following steps have to be performed:

The detector has to be set to the desired readout time using the Timings menu

XISL: Acquisition_SetCameraMode(hAcqDesc, 0..7)

-> The detector runs continuously (free running) in the desired timing and readout scheme.

The detector has to be set to the internal trigger mode using the Detector Mode menu

XISL: Acquisition_SetFrameSyncMode(hAcqDesc, HIS_SYNCMODE_INTERNAL_TIMER).

-> The detector aborts the current frame and waits for a trigger signal.

-> The shortest repeat time of an trigger pulse is the selected timing (readout time).

Select the frame rate (readout plus delay time) in the appearing dialog (readout time ... 5s)

XISL: Acquisition_SetTimerSync(hAcqDesc, frame_rate).

The frame grabber sends the /FR_SYNC signal directly via the HIIB to the detector in the selected frame rate.

-> The detector runs "continuously" with the selected frame rate.

Start the acquisition

XISL: Acquisition_AquireImage(hAcqDesc, frames, ...)

-> The frame grabber grabs the data into the memory.

6.1.5 Trigger-modes

6.1.5.1 Frame-wise (default)

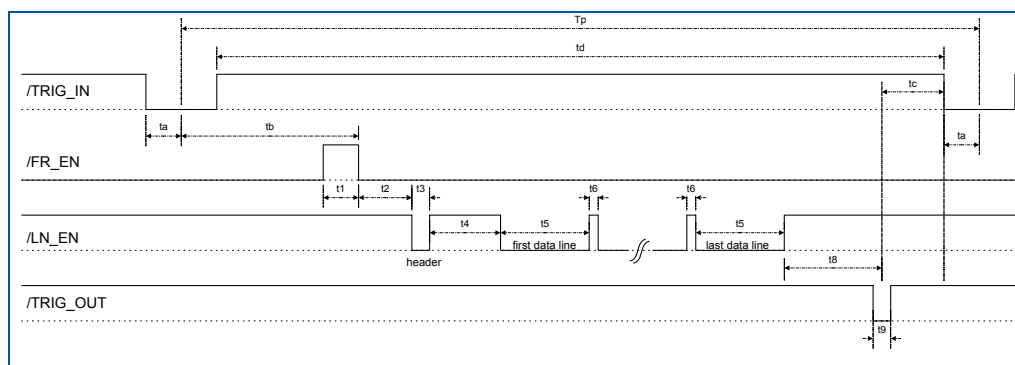


Figure 24: General timing diagram in frame-wise triggered mode

This is the most general trigger mode. Once the detector has been set to trigger mode the detector will synchronize to trigger events and perform a complete detector scan and transmit one image. This is a frame wise trigger mode which allows the operator to take control of the integration time of the detector. Trigger events during the scan process are ignored. Thus the shortest possible integration is equivalent to the integration time of the fastest mode in free running.

6.1.5.2 Data Delivered on Demand' triggered mode

Trigger with DDD mode is available for detectors with HeaderID 14 and CameraType 2

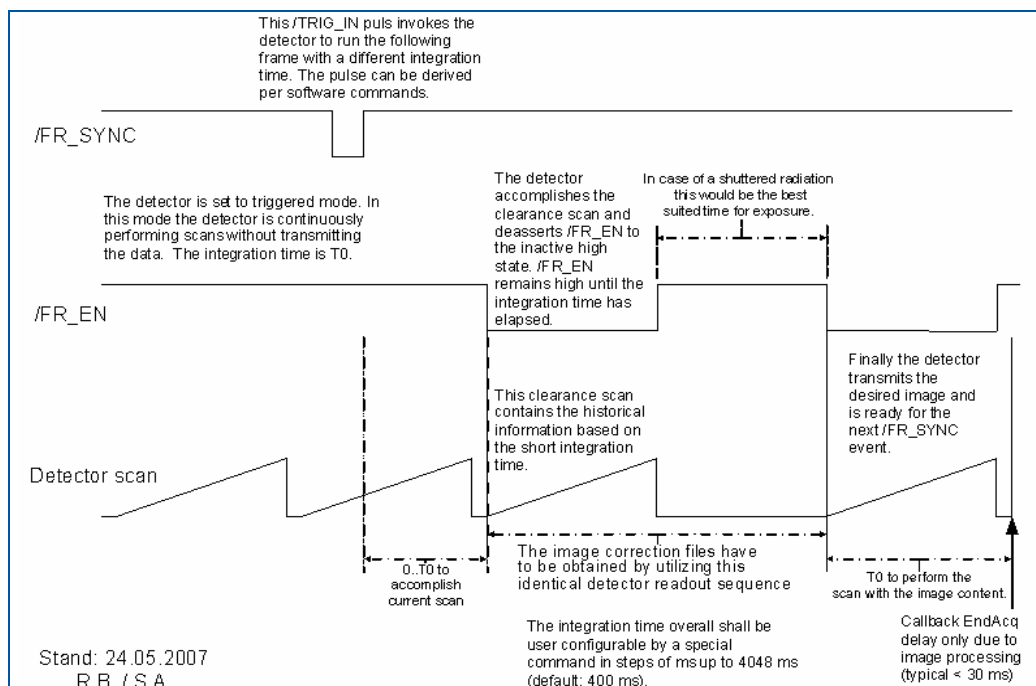


Figure 25: timing diagram for 'Data Delivered on Demand' triggered mode

The detector is running in a "silent" mode, like free running mode but without transferring image data. If the user application (Software, Frame Grabber or external Source) sends a triggers signal to the detector, the detector accomplishes the current frame. The next frame is processed also with the fastest integration time and a delay time appears which is the best time suited for exposure in case of pulsed or shuttered radiation. After this delay the detector performs the image scan and transfers the desired data. After this the detector returns to the silent mode until the next trigger pulse is sent.

6.1.5.3 Start/Stop trigger-mode

As long as the /TRIG_IN signal is at the inactive high state the detector will keep continuously scanning lines. As soon as /TRIG_IN receives low state (active) the current line scan will be completed and the scan will be halted. Once the /TRIG_IN gets released the detector will continue the scanning at the current position. During the halt phase the x-ray burst pulses are applied. The period between consecutive halt phases should be optimised to allow a certain number of line scans to be performed. This method allows synchronising the detector to the pulse regime of the x-ray source in such a fashion, that no x-rays are applied during line scans where the gates of the TFT's are being turned on and off.

6.1.6 External trigger inputs/outputs

The /TRIG IN, /TRIG OUT, /LN_EN and /FR_EN signals are low active /TTL signals for the XRD 512-400 and XRD 1640 AL/AG detectors and LVDS Signals for the new XRD 08x0 AN and XRD 16xx AN detectors. The meaning of these signals are described in the Table 30. Detailed descriptions of the trigger ports are in the detector manuals.

/FR_ENB	indicates a new frame (detector output)
/FR_SYNC	frame synchronization signal, generated by the frame grabber to externally control the frame rate (detector input)
/LN_ENB	indicates a new line (detector output)
/TRIG_IN	External trigger signal to synchronize the frame rate (detector input)
/TRIG_OUT	The /TRIG_OUT signal indicates the start of a new frame and can be used to synchronise the x-ray source (detector output)

Table 30: Description of the trigger input/output signals

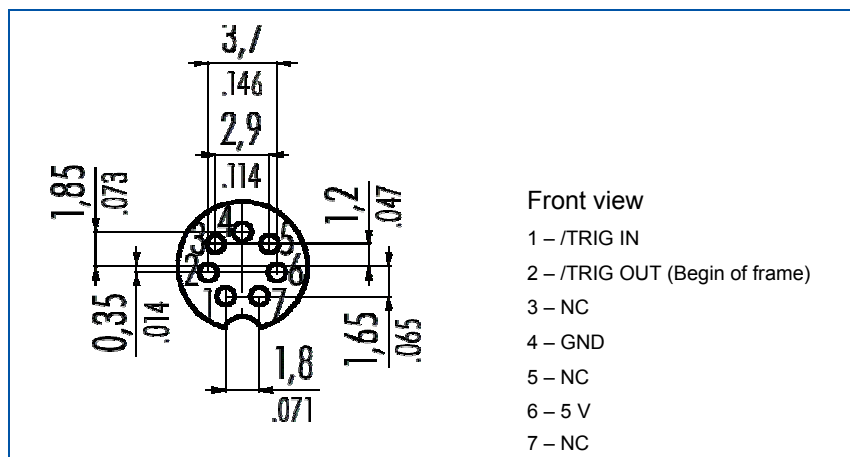


Figure 26 External Trigger inputs/outputs of the XRD 512-400 and XRD 1640 AL/AG detector series

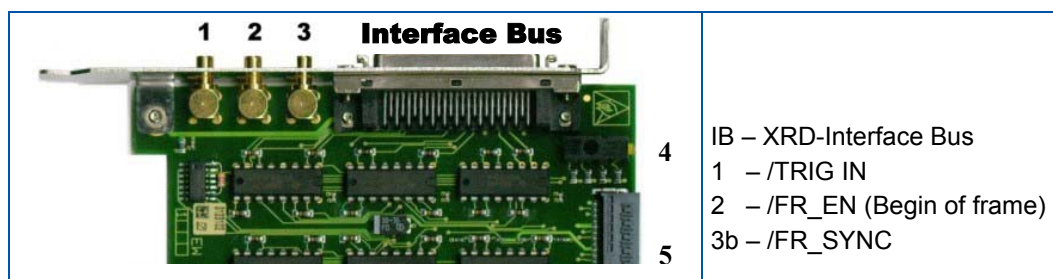


Figure 27: External Trigger inputs/outputs of the XRD-FG Frame Grabber

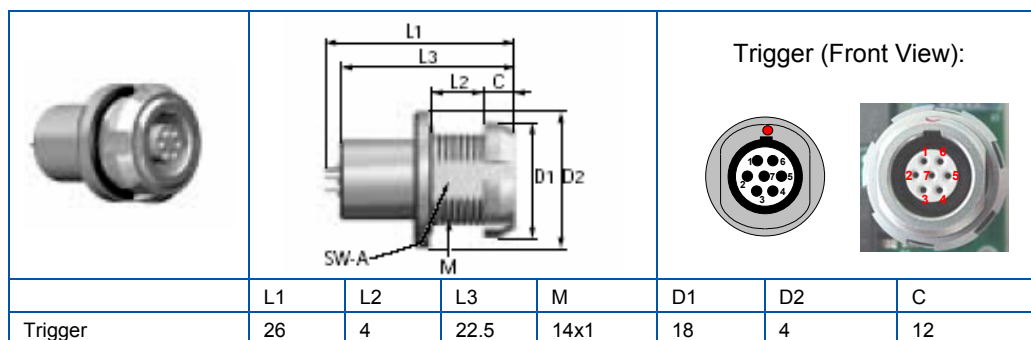


Figure 28: Trigger inputs/outputs of the **XRD 08xx yN** and **XRD 162xx yN** detectors

PIN	Colour	Connection
1	Black	TRIGIN_+
2	Brown	TRIG_IN_-
3	Red	TRIGOUT_+
4	Orange	TRIGOUT_-
5	Yellow	FGND
6	Blue	5PF
7	Magenta	

Table 31 PIN assignment of Trigger signal for the **XRD 08xx yN** and **XRD 16xx yN** detectors (LVDS Signals)

PIN	DESCRIPTION	
1	GP_OUT0+	frame enable
2	GP_OUT1+	frame sync
3	GP_IN0+	trigger in
4	GP_IN1+	na
5	GND	
6	GP_OUT0-	frame enable
7	GP_OUT1-	frame sync
8	GP_IN0-	trigger in
9	GP_IN1-	na

Table 32: LVDS connector Pin assignment (GP_** use LVDS signalling standard)

6.2 How to use the detector gain setting

This paragraph describes the change of the detector gain. The **XRD AN** type detectors are supporting five different gain settings. When the detector is powered on the XRD AN runs with 1pF capacity. The detector gain will be set bitwise by the **XIS** dialog

Options/Detector Options or by the library function

Acquisition_SetCameraGain(*hAcqDesc*, 1..5). The following table will give an overview of the different gain settings. It is very important for the image quality that the correction files were obtained in the same gain setting as in the current working operating mode.

Selected Value	Capacity
0	0.25pF (200µm only)
1	0.5 pF
2	1 pF
3	2 pF
4	4 pF
5	8 pF

Table 33: Overview of the five Detector Gain Settings Table

6.3 How to use the detector binning setting

This paragraph describes the change of the detector binning modes. The **XRD 0820/1621 AN detectors** are supporting different binning mode settings. When the detector is powered on it runs without binning (default mode). The detector binning will be set by the **XIS** dialog **Options/Detector Options** or by the library function Acquisition_SetCameraBinningMode(..). The following table gives an overview of the different binning modes.

Note: It is very important for the image quality that the correction files are obtained in the same binning mode, gain setting and integration time as for the measurements.

Selected Value Bit	Binning mode
0	No binning
1	2 x 2 binning
8	Averaged binning
9	Accumulated binning

Table 34: Overview of the detector binning modes

6.4 File format

Each pixel is digitized in 16 bit resolution (2 byte information) and saved as 16 bit unsigned integer for acquired frames. Gain images are stored as unsigned 32 bit integers and there are also double precision floating point data (8 byte) representations supported by the software. All data can be signed or unsigned in general. The data are preceded by a file header of 68 byte and an image header of 32 byte. If we consider a sequence of n images acquired by a 512x512 detector we get the file sizes listed in the table below:

File		Single Image		Sequence	
Header		68	byte	68	byte
Image Header		32	byte	32	byte
Image Data	512x512x2 byte	524288	byte	524288	byte * n

Table 35: Single image and file sequence format

The **file header** allows the reading of the data by any other software module and has the following description:

Information	Description
File header	68 byte
WORD FileType	// File ID (0x7000)
WORD HeaderSize	// Size of this file header in bytes
WORD HeaderVersion	// yy.y
ULONG FileSize	// Size of the whole file in bytes
WORD ImageHeaderSize	// Size of the image header in bytes
WORD ULX, ULY, BRX, BRY	// bounding rectangle of the image
WORD NrOfFrames	// number of frames
WORD Correction	// 0 = none, 1 = Offset, 2 = Offset+Gain
double IntegrationTime	// frame time in microseconds
WORD TypeOfNumbers	// short, long integer, double, signed/unsigned, inverted, fault map, Offset/Gain correction data, pixel correction data
BYTE x[WINRESTSIZE]	// fill up to 68 byte

Table 36: File header description

The file header has a size of 68 byte. It consists of several entries to describe the organization of the stored data. Most of the entries are self-explanatory except the TypeOfNumbers entry, which can have any combination (OR) of the following values:

double precision floating point number	2
16 bit integer	4
data are signed	8
32 bit integer	32

Table 37: Type of Data

One can get the values of the bounding rectangle by ULX, ULY, BRX, BRY. The number of rows and columns results from the formula:

$$\begin{aligned} \text{rows} &= \text{BRY} - \text{ULY} + 1 \\ \text{columns} &= \text{BRX} - \text{ULX} + 1 \end{aligned}$$

6.5 Description of Hardware Header

The hardware header is transferred by the detector at initialization time of the frame grabber board and at the end of acquisition time.

For a detailed description of the Hardware-Header see Table 57: Description of the ChwHeaderInfo structure and Table 60: Description of the CHwHeaderInfoEx structure
To get all Timings use the Acquisition_GetIntTimes(..)-function while Detector is in FreeRunning-Mode.

6.6 Row types

The following pages describe the available row read out schemes and the corresponding value of RowType (see Hardware Header).

All time values are given in microseconds. In sync mode the timer of the first row starts at synchronization time with a possible time tolerance of 32 nanoseconds. The trigger input is low active. The minimal pulse duration is 1 microsecond and the trigger event is done at the rising edge.

RowType	Row time
0	300
1	300
2	312.5
3	390.625
4	781.25
5	310
6	387.5
7	781.25
8	2500
9	524.0
10	524.0
11	310.0
12	3333
14	2500
15	2500
16	2500
17	2500
18	1536
19	261.0
20	472.0
21	315.9
22	945.0
23	630.0
24	264.0
25	528.0
26	266.0
27	796.0
28	524.0

Table 38: Timing diagram of the row types

6.7 Interrupt sources

Interrupts allow the application to wait passively for changes in the acquisition status of the hardware.

There are four interrupt sources:

start of DMA, end of DMA, end of sequence, end of acquisition

These interrupts occur if the acquisition status changes to allow the application to react.

The acquisition mode (Continuous, Single Shot, Sequence) influences the data flow and therefore the acquisition status. The following diagrams illustrate the data flow and the corresponding interrupts.

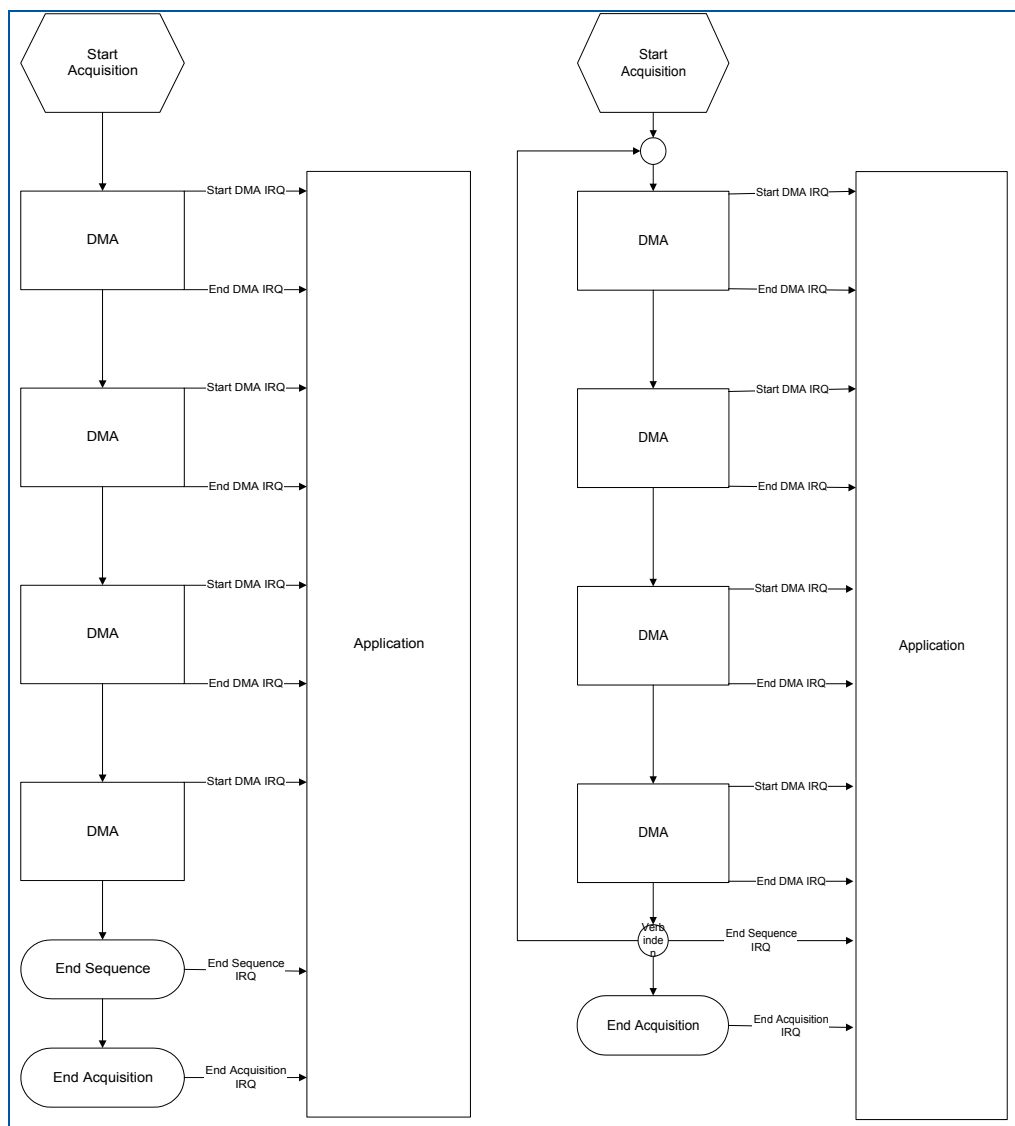


Figure 29: Interrupt sources at a) sequence acquisition mode (4 frames) b) continuous mode.

After the application started the acquisition one interrupt is caused by the beginning of DMA (start DMA interrupt). If the whole frame data is transferred the end of DMA interrupt is fired. Both interrupts show the same behavior in all acquisition modes. What happens if the end of DMA buffer is reached depends on the acquisition mode. In sequence or single shot acquisition mode an end of sequence interrupt occurs and after that an end of acquisition interrupt is caused. In continuous acquisition mode an end of sequence interrupt is caused and the data transfer continuous at start of DMA buffer. If interrupts are enabled the end of frame, the end of sequence and the end of acquisition interrupts are used. An cancel of acquisition sets the end of frame interrupt.

6.8 Polling mode

If no interrupts are enabled the acquisition is running in polling mode. This is a time consuming task because the application actively asks the acquisition driver for its status. The behavior of the driver for applications is the same as in interrupt mode. To choose polling mode makes sense if you have non solvable interrupt resource conflicts in your computer. But in the interrupt mode there is an active image synchronization. This means if there is a disturbance on the data line or at the detector itself the acquisition will be synchronized only in the interrupt mode.

6.9 Sorting

6.9.1 Sorting schemes overview

Depending on the sensor and detector type the data come in different orders from the detector. The XISL sort the data in an internal buffer with highly optimized routines written in machine code. If the sensor and detector type is unknown the XIS comes up with a detector type dialog at initialization, where the correct sorting has to be entered. The following detector types and sortings are supported:

RID 128	1
RID 256	2
RID 128-400	3
RID 1024-100	4
RID 512-400 A0	5
XRD 512-400 A1/A2	6
XRD 0840	6
XRD 512-400 E	7
XRD 1640 A	8
XRD 0820	8
XRD 1620 A	8
XRD 1680 A	9
XRD 1620/21 AM/AN	11
XRD 1620/40 AN CS	12

Table 39: Sorting schemes overview

6.9.2 Sorting RTM 128

The following picture demonstrates the data stream coming in from an RTM 128 sensor. The first acquired data value stems from row 3 and column 3 (start label). The next data word is coming in from row 3 and column 4. This is represented by an arrow. As a result of the upper scheme we get the following table:

data stream no.	sensor pixel (row, column)
1	(3,3)
2	(3,4)
3	(3,1)
4	(3,2)
5	(4,3)
6	(4,4)
7	(4,1)
8	(4,2)
9	(1,3)
10	(1,4)
11	(1,1)
12	(1,2)
13	(2,3)
14	(2,4)
15	(2,1)
16	(2,2)
17	(7,7)
...	...

Table 40: Sorting overview of the RTM 128

6.9.4 Sorting RID 128-400

The read out schematic of an RID 128-400 sensor is similar to that of an RTM 128 sensor except that rows come in the correct order. The following picture illustrates the read out cycles.

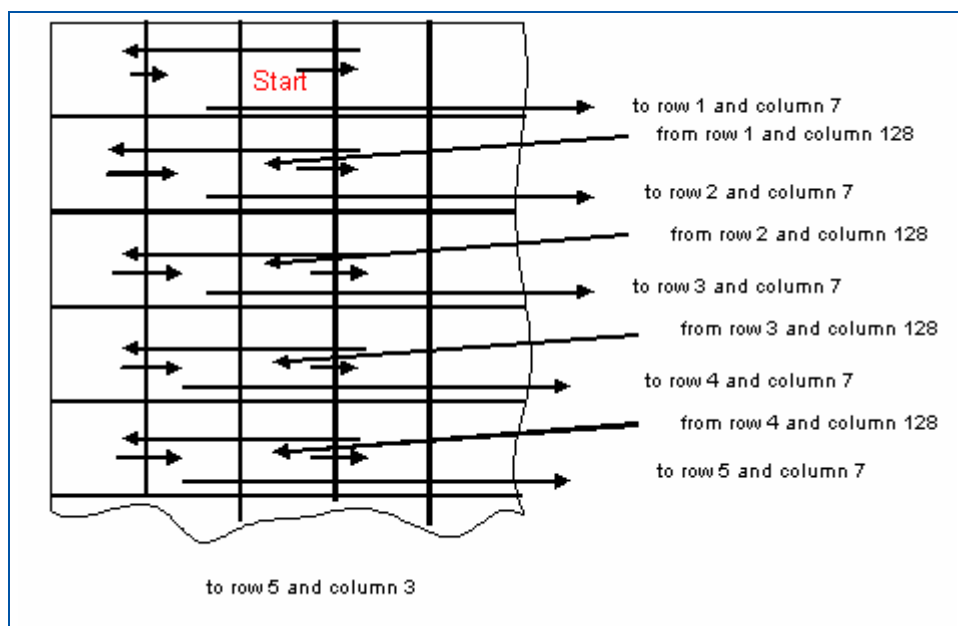


Figure 31: Sorting overview of the RID 128-400

This results in the following list:

data stream no.	sensor pixel (row, column)
1	(1,3)
2	(1,4)
3	(1,1)
4	(1,2)
5	(1,7)
6	(1,8)
7	(1,5)
8	(1,6)
9	(1,11)
...	...
125	(1,127)
126	(1,128)
127	(1,125)
128	(1,126)
129	(2,3)
130	(2,4)
131	(2,1)
132	(2,2)
...	...

Table 42: Sorting overview of the RID 128-400

6.9.5 RID 1024-100

The figure demonstrates the read out schematic of the RID 1024-100 sensor. The whole sensor is divided into four parts. Every part is read out by four "read out groups" (ROG). The upper ROG (number 1,3,5,7) read out the odd columns while the lower ROG (number 2,4,6,8) read out the even columns. At first ROG no. 1 is transferred to data stream, after that no. 2, after that no. 3 and so on. The upper ROG scan the pixel columns from right to left while the lower ones scan from left to right. The data stream is described in the Table 38:

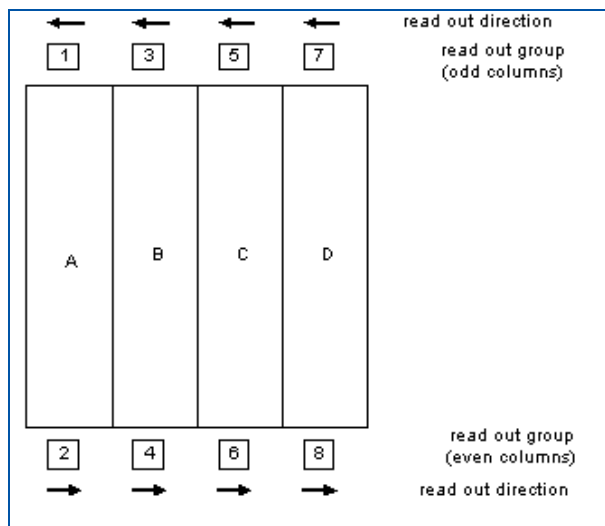


Figure 32: Sorting overview of the RID 1024-100

data stream no.	sensor pixel (row, column)	ROG no.
1	(1,255)	1
2	(1,2)	2
3	(1,511)	3
4	(1,258)	4
5	(1,767)	5
6	(1,514)	6
7	(1,1023)	7
8	(1,770)	8
9	(1,253)	1
10	(1,4)	2
11	(1,509)	3
12	(1,260)	4
13	(1,765)	5
14	(1,516)	6
15	(1,1021)	7
...
1015	(1,771)	7
1016	(1,1022)	8
1017	(1,1)	1
1018	(1,256)	2
1019	(1,257)	3
1020	(1,512)	4
1021	(1,513)	5
1022	(1,768)	6
1023	(1,769)	7
1024	(1,1024)	8
1025	(2,255)	1
1026	(2,2)	2
...

Table 43: Sorting overview of the RID 1024-100

6.9.6 Sorting RID 512-400 A0

The figure demonstrates the read out scheme of the sensor.

The whole sensor is divided into an upper and a lower part. Both parts are electrically separated. The data of every part are transferred by four "read out groups" (ROG). Every group scans the sensor columns from left to right. At first the upper groups are transferred and after that the lower ones. This results in the following Table 39:

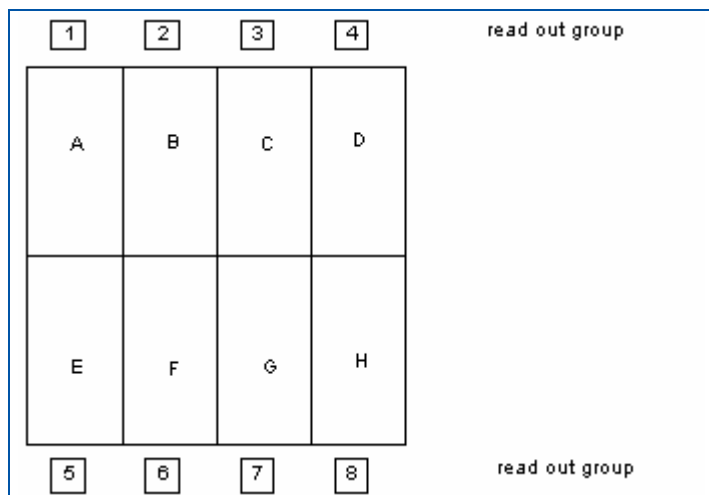


Figure 33: Sorting overview of the RID 512-400 A0

data stream no.	sensor pixel (row, column)	ROG no.
1	(1,1)	1
2	(1,129)	2
3	(1,257)	3
4	(1,385)	4
5	(257,1)	5
6	(257,129)	6
7	(257,257)	7
8	(257,385)	8
9	(1,2)	1
10	(1,130)	2
11	(1,258)	3
12	(1,386)	4
13	(257,2)	5
14	(257,130)	6
15	(257,258)	7
...
262131	(256,383)	3
262132	(256,511)	4
262133	(512,127)	5
262134	(512,255)	6
262135	(512,383)	7
262136	(512,511)	8
262137	(256,128)	1
262138	(256,256)	2
262139	(256,384)	3
262140	(256,512)	4
262141	(512,128)	5
262142	(512,256)	6
262143	(512,384)	7
262144	(512,512)	8

Table 44: Sorting overview of the RID 512-400 A0

6.9.7 Sorting XRD 512-400 A1/A2 // XRD 0840

The figure demonstrates the read out scheme of the sensor.

The whole sensor is divided into an upper and a lower part. Both parts are electrically separated. The data of every part are transferred by four "read out groups" (ROG). The upper groups scan the sensor columns from left to right. The lower groups scan from right to left. At first the upper groups are transferred and after that the lower ones. The upper groups start their read out from the upper row. The lower ones start read out from the last row. This results in the following Table 40:

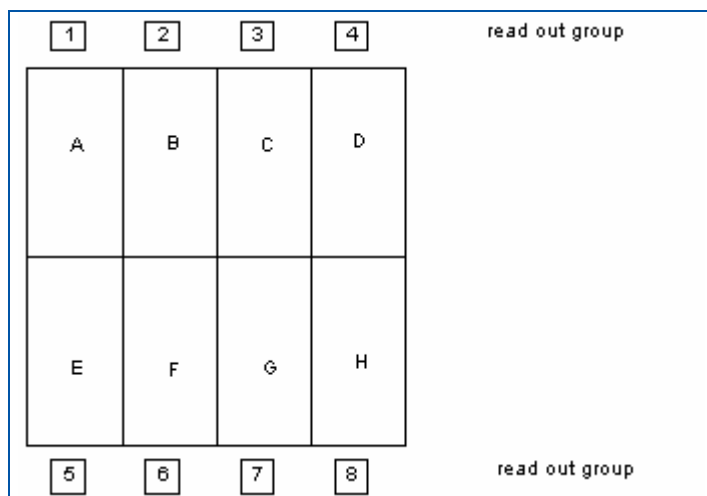


Figure 34: Sorting overview of the XRD 512-400 A1/A2

data stream no.	sensor pixel (row, column)	ROG no.
1	(1,1)	1
2	(1,129)	2
3	(1,257)	3
4	(1,385)	4
5	(512,128)	5
6	(512,256)	6
7	(512,384)	7
8	(512,512)	8
9	(1,2)	1
10	(1,130)	2
11	(1,258)	3
12	(1,386)	4
13	(512,127)	5
14	(512,255)	6
...
262129	(256,127)	1
262130	(256,255)	2
262131	(256,383)	3
262132	(256,511)	4
262133	(257,2)	5
262134	(257,130)	6
262135	(257,258)	7
262136	(257,386)	8
262137	(256,128)	1
262138	(256,256)	2
262139	(256,384)	3
262140	(256,512)	4
262141	(257,1)	5
262142	(257,129)	6
262143	(257,257)	7
262144	(257,385)	8

Table 45: Sorting overview of the XRD 512-400 A1/A2

6.9.8 Sorting XRD 512-400 E

The figure demonstrates the read out scheme of the sensor.

The whole sensor is divided into an upper and a lower part. Both parts are electrically separated. The data of every part are transferred by four "read out groups" (ROG). The upper groups scan the sensor columns from left to right. The lower groups scan from right to left. At first the lower groups are transferred and after that the upper ones. The upper groups start their read out from the upper row. The lower ones start read out from the last row. This results in the following list:

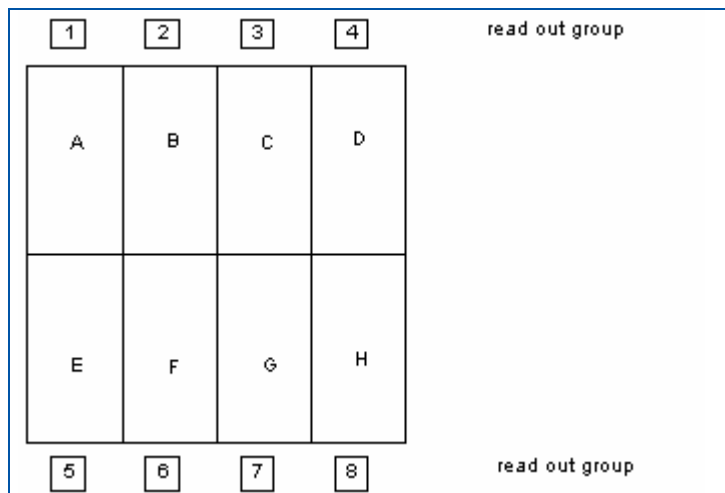


Figure 35: Sorting overview of the XRD 512-400 E

data stream no.	sensor pixel (row, column)	ROG no.
1	(512,128)	5
2	(512,256)	6
3	(512,384)	7
4	(512,512)	8
5	(1,1)	1
6	(1,129)	2
7	(1,257)	3
8	(1,385)	4
9	(512,127)	5
10	(512,255)	6
11	(512,283)	7
12	(512,511)	8
13	(1,2)	1
...
1022	(1,384)	3
1023	(1,512)	4
1024	(511,128)	5
1025	(511,256)	6
1026	(511,384)	7
1027	(511,512)	8
1028	(2,1)	1
1029	(2,129)	2
...
262137	(257,1)	5
262138	(257,129)	6
262139	(257,257)	7
262140	(257,385)	8
262141	(256,128)	1
262142	(256,256)	2
262143	(256,384)	3
262144	(256,512)	4

Table 46: Sorting overview of the XRD 512-400 E

6.9.9 Sorting XRD 1640 A // XRD 1620 AJ // XRD 0820

The figure demonstrates the read out scheme of the sensor. The sensors XRD 1620 AJ and XRD 1640 have a similar sorting.

The whole sensor is divided into an upper and a lower part. Both parts are electrically separated. The data of every part are transferred by eight "read out groups" (ROG). Each ROG has 128 channels for the XRD 1640/0820 and 256 for the XRD 1620 AJ. The upper groups scan the sensor columns from left to right. The lower groups scan from right to left. At first the upper groups are transferred and after that the lower ones. The upper groups start their read out from the upper row. The lower ones start read out from the last row. The following table displays the data stream for XRD 1640:

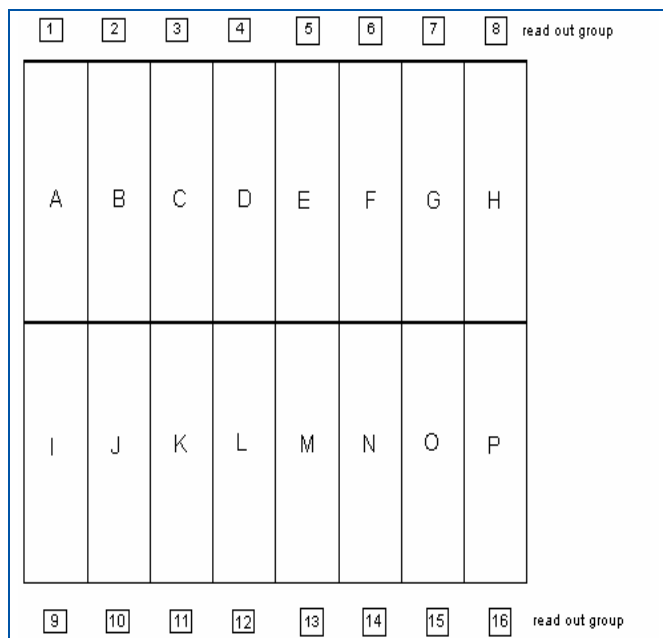


Figure 36: Sorting overview of the XRD 1640 A / 1620 AJ

data stream no.	sensor pixel (row, column)	XRD1620 AJ	ROG no.
1	(1,1)	(1,1)	1
2	(1,129)	(1,257)	2
3	(1,257)	(1513)	3
4	(1,385)	(1,769)	4
5	(1,513)	(1,1025)	5
6	(1,641)	(1,1281)	6
7	(1,769)	(1,1537)	7
8	(1,897)	(1,1793)	8
9	(1024,128)	(2024,256)	9
10	(1024,256)	10
11	(1024,384)		11
12	(1024,512)		12
13	(1024,640)		13
14	(1024,768)		14
15	(1024,896)		15
16	(1024,1024)		16
17	(1,2)		1
18	(1,130)		2
19	(1,258)		3
20	(1,386)		4
21	(1,514)		5
22	(1,642)		6
23	(1,770)		7
24	(1,898)		8
25	(1024,127)		9
26	(1024,255)		10
27	(1024,283)		11
28	(1024,511)		12
29	(1024,639)		13

30	(1024,767)		14
31	(1024,895)		15
32	(1024,1023)		16
33	(1,3)		1
34	(1,131)		2
35	(1,259)		3
36	(1,387)		4
...
1048540	(513,387)		12
1048541	(513,515)		13
1048542	(513,643)		14
1048543	(513,771)		15
1048544	(513,899)		16
1048545	(512,127)		1
1048546	(512,255)		2
1048547	(512,383)		3
1048548	(512,511)		4
1048549	(512,639)		5
1048550	(512,767)		6
1048551	(512,895)		7
1048552	(512,1023)		8
1048553	(513,2)		9
1048554	(513,130)		10
1048555	(513,258)		11
1048556	(513,386)		12
1048557	(513,514)		13
1048558	(513,642)		14
1048559	(513,770)		15
1048560	(513,898)		16
1048561	(512,128)		1
1048562	(512,256)		2
1048563	(512,384)		3
1048564	(512,512)		4
1048565	(512,640)		5
1048566	(512,768)		6
1048567	(512,896)		7
1048568	(512,1024)		8
1048569	(513,1)		9
1048570	(513,129)		10
1048571	(513,257)		11
1048572	(513,385)		12
1048573	(513,513)		13
1048574	(513,641)		14
1048575	(513,769)		15
1048576	(513,897)		16

Table 47: Sorting overview of the XRD 1640 A / 1620 AJ

6.9.10 Sorting XRD 1620/21 AM/AN

The figure demonstrates the read out scheme of the sensor.

The whole sensor is divided into an upper and a lower part. Both parts are electrically separated. The data of every part are transferred by 16 “read out groups” (ROG). Each ROG has 128 channels. The upper groups scan the sensor columns from left to right. The lower groups scan from right to left. At first the upper groups are transferred and after that the lower ones. The upper groups start their read out from the upper row. The lower ones start read out from the last row.

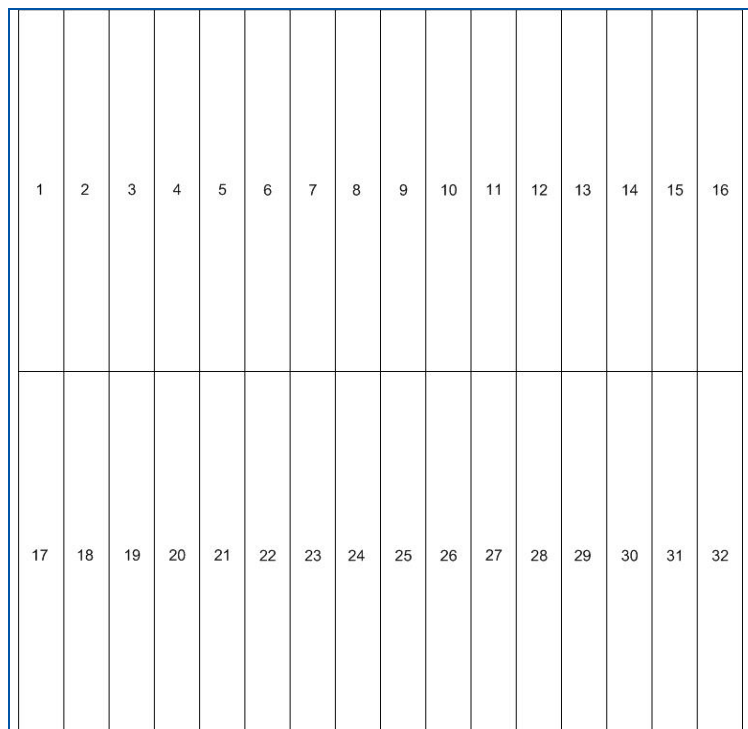


Figure 37: Sorting overview of the XRD 1620 AN / 1621 AN

The following table displays the data stream for XRD 1620 AM:

data stream no.	sensor pixel (row, column)	ROG no.
1	(1,1)	1
2	(1,129)	2
3	(1,257)	3
4	(1,385)	4
5	(1,513)	5
6	...	
15	(1,1793)	15
16	(1,1921)	16
17	(2048,128)	17
18	(2048,256)	18
19	(2048,384)	19
20	(2048,512)	20
...

Table 48: Sorting overview of the XRD 1620/21 AN//AM

6.9.11 Sorting XRD 1620/40 AN CS

The figure demonstrates the read out scheme of the sensor.

The data are transferred by 16 “read out groups” (ROG). Each ROG has 128 channels (64 for 1640). The groups scan from right to left. The rows start read out from the last row.

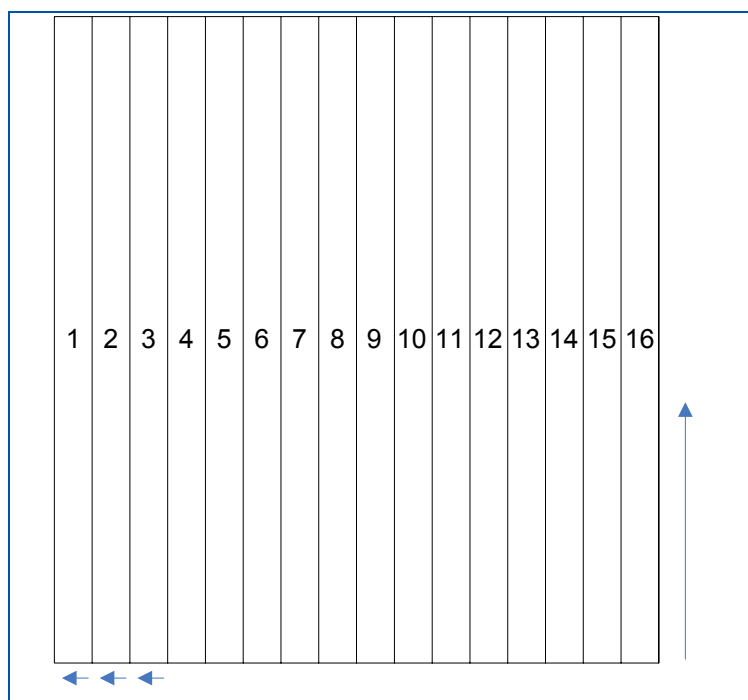


Figure 38: Sorting overview of the XRD 1620/40 AN CS

The following table displays the data stream for XRD 1620 AN CS:

data stream no.	sensor pixel (row, column) 1620	sensor pixel (row, column) 1640	ROG no.
1	(2048,128)	(1024,64)	1
2	(2048,256)	(1024,128)	2
3	(2048,384)	(1024,192)	3
...
16	(2048,2048)	(1024,1024)	16
17	(2048,127)	(1024,63)	1
18	(2048,255)	(1024,127)	2
19	(2048,384)	(1024,191)	3
...
32	(2048,2047)	(1024,1023)	16
...

Table 49: sorting overview of the XRD 1620/40 AN CS

7 X-Ray Imaging Software Library

7.1 X-Ray Imaging Library Overview

The X-Ray Imaging Software Library provides the basic functionality to acquire and correct XRD/RID data. To get an impression of the tasks you have to implement to acquire images by help of the XISL look at XISL demo. This demo is also a part of the XIS setup.

Acquisition_Descriptor	Basic structure for data acquisition used by XISL.
Acquisition_EnumSensors	Enumerates all connected sensors
Acquisition_GetNextSensor	Iterates through all recognized sensors.
Acquisition_Init	Initializes driver and frame grabber.
Acquisition_Acquire_Image	Acquires images from the detector.
Acquisition_Acquire_GainImage	Acquires a gain correction image.
Acquisition_Acquire_OffsetImage	Acquires an offset correction image.
Acquisition_DoOffsetCorrection	Performs an offset correction on an acquired image.
Acquisition_DoGainCorrection	Performs a gain correction on an acquired image.
Acquisition_DoOffsetGainCorrection	Performs offset and gain correction on an acquired image.
Acquisition_DoPixelCorrection	Performs a pixel correction on an acquired image.
Acquisition_DefineDestBuffers	Definition of the destination buffers for image capturing.
Acquisition_CreatePixelMap	Creates a list for a mean correction of defective pixels.
Acquisition_GetReady	Informs the user if image drawing is ready.
Acquisition_SetReady	Informs the XISL if image drawing is ready.
Acquisition_SetCameraMode	Allows setting of detector frame time.
Acquisition_IsAcquiringData	Checks if sensor is about to acquire data.
Acquisition_GetErrorCode	Returns extended information if an error occurred.
Acquisition_GetConfiguration	Retrieves the current configuration setting of the XISL.
Acquisition_GetIntTimes	Routine to detect the actual frame time automatically.
Acquisition_GetWinHandle	Returns the handle of the current acquisition window.
Acquisition_SetAcqData	Sets a 32 bit value that can be extracted by Acquisition_GetAcqData during acquisition time.
Acquisition_GetAcqData	Extracts a 32 bit value at acquisition time that was setted by Acquisition_SetAcqData.
Acquisition_GetActFrame	Retrieves the current acquisition frames.
Acquisition_GetHwHeaderInfo	Returns the contents of the detector's hardware header in a info structure.
Acquisition_Abort	Aborts a running acquisition.
Acquisition_AbortCurrentFrame	Aborts the transmission of the current frame and immediately starts a new transfer.
Acquisition_Close	Closes driver and hardware.
Acquisition_CreateGainMap	Create a List of Median values for the Gain-Sequence correction
Acquisition_Acquire_Image_Ex	Acquires images from the detector.
Acquisition_SetCorrData_Ex	This function switches the correction buffers during a running acquisition.
Acquisition_GetCorrData_Ex	This function retrieves the correction buffers during a running acquisition
Acquisition_DoOffsetGainCorrection_Ex	Performs offset and gain-sequence correction on an acquired image.
Acquisition_SetCameraGain	Set Detector Gain (not available for all detectors)
Acquisition_Acquire_Image_PreloadCorr	Acquire Image w/o setting correction data
Acquisition_Acquire_GainImage_PreloadCorr	Acquire Gain Image w/o setting correction data
Acquisition_Acquire_OffsetImage_PreloadCorr	Acquire Offset Image w/o setting correction data
Acquisition_Acquire_GainImage_EX_ROI	Acquire Gain Image with defined Region of interest
Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr	Acquire Gain Image (ROI) w/o setting correction data
Acquisition_SetCameraBinningMode	Set binning Mode (not available for all detectors)
Acquisition_GetCameraBinningMode	Retrieve actual binning Mode.
Acquisition_SetCameraTriggerMode	Set trigger Mode (not available for all detectors)
Acquisition_GetCameraTriggerMode	Retrieve actual trigger Mode.
Acquisition_ResetFrameCnt	Reset detector Frame counter (not for all detectors)
Acquisition_GetHwHeaderInfoEx	Acquire Header and Extended Header if available
Acquisition_GetLatestFrameHeader	Retrieve Last Frame Header and Extended Header if available
Acquisition_SetFrameSyncTimeMode	Set Parameters for 'Data Delivered on Demand' triggermode

Table 50: Overview of the X-Ray Imaging Software Library

7.2 XISL Functions

7.2.1 Acquisition Descriptor

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 pointer to the acquisition descriptor. A valid **HACQDESC** pointer is returned by `Acquisition_Init` and all allocated resources are freed by `Acquisition_Close`. After a call of `Acquisition_Close` the acquisition descriptor pointer is invalid.

7.2.2 Acquisition_EnumSensors

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.

```
UINT WINAPI Acquisition_EnumSensors(
    UINT *pdwNumSensors,
    BOOL bEnableIRQ,
    BOOL bInitAlways
);
```

PdwNumSensor

Address of a 4 byte integer that receives the number of recognized sensors.

bEnableIRQ

If you want to run the acquisition in polling mode set this parameter to zero. If you want to enable hardware interrupts set the parameter to one.

bInitAlways

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.

Return Value

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

7.2.3 Acquisition_GetNextSensor

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

```
UINT WINAPI Acquisition_GetNextSensor(
    ACQDESCPOS *Pos,
    HACQDESC *hAcqDesc
);
```

Pos

Pointer to an unsigned 4 byte integer that receives information's that are needed for subsequent calls of this function. To receive the acquisition descriptor (**HACQDESC**) for the first recognized sensor set `Pos` to `NULL`.

hAcqDesc

Handle of a structure that contains all needed parameters for acquisition (**HACQDESC**). If you call `Acquisition_Init` the first time set `hAcqDesc` to `NULL`, in subsequent calls use the former returned value.

Return Value

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

7.2.4 Acquisition_Init

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).

UINT WINAPI Acquisition_Init(

HACQDESC *phAcqDesc,
DWORD dwBoardType,
int nChannelNr,
BOOL bEnableIRQ,
UINT Rows,
UINT Columns,
UINT dwSortFlags,
BOOL bSelfInit,
BOOL bInitAlways

);

phAcqDesc

Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call Acquisition_Init the first time set hAcqDesc to NULL, in subsequent calls use the former returned value.

dwBoardType

This parameter defines on which communication device the sensor is located. Only one type of frame grabber can be used at the same time.

dwBoardType can have the following values:

symbolic name	numeric value	meaning
HIS_BOARD_TYPE_ELTEC	1	The communication to the sensor is provided by the XRD-FG frame grabber.
HIS_BOARD_TYPE_ELTEC_XRD_FGX	8	The communication to the sensor is provided by the XRD-FGX frame grabber.

Table 51: Supported Frame Grabber device and their channel type

dwChannelNr

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 board selector is set to three, set *dwChannelNr* equal to 3.

bEnableIRQ

If you want to run the acquisition in polling mode set this parameter to zero. If you want to enable hardware interrupts set the parameter to one.

Rows, Columns

Number of sensor columns, and rows.

dwSortFlags

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:

HIS_SORT_NOSORT (0x0)	no sorting
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

Table 52: Sorting Flags

The sorting is done automatically by XISL during acquisition. The sorting routines are written in machine code and are therefore very fast.

bSelfInit

If a detector with an unknown PROM-ID is connected to the frame grabber, the XISL normally comes up with a dialog to enter the detector parameters. If *bSelfInit* is set to zero this dialog is suppressed (see *lpfnEndFrameCallback*) and the configuration parameters supplied by *Rows*, *Columns*, *dwSortFlags* are used.

bInitAlways

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.

Return Value

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

7.2.5 Acquisition_GetCommChannel

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

UINT WINAPI Acquisition_GetCommChannel(

HACQDESC hAcqDesc,

UINT *pdwChannelType,

int *pnChannelNr

);

hAcqDesc

Pointer to HACQDESC.

pdwChannelType

Address of a 4 byte integer that receives an id of the currently open communication device. Currently supported devices and their corresponding id's are:

Symbolic name	id	Description
HIS_BOARD_TYPE_NOONE	0x0	no device (not valid)
HIS_BOARD_TYPE_ELTEC	0x1	XRD-FG Frame Grabber
HIS_BOARD_TYPE_ELTEC_XRD_FGX	0x 8	The communication to the sensor is provided by the XRD-FGX frame grabber.

Table 53: Supported Frame Grabber device and their channel type

pnChannelNr

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.

7.2.6 Acquisition_DefineDestBuffers

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.

UINT WINAPI Acquisition_DefineDestBuffers(

HACQDESC hAcqDesc,

unsigned short *pProcessedData,

UINT nFrames,

UINT nRows,

UINT nColumns

);

hAcqDesc

Pointer to HACQDESC.

pProcessedData

Pointer to the destination buffer.

nFrames

Number of frames of the destination buffer. It must be greater than zero.

nRows, nColumns

Number of rows and columns of the destination buffer. If these numbers are not suitable to the sensor the function return with an error code. If you need extended information call Acquisition_GetErrorCode.

Return Value

Zero if function is successful, otherwise with a greater value.

7.2.7 Acquisition_SetCallbacksAndMessages

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.

```
UINT WINAPI Acquisition_SetCallbacksAndMessages(
    HACQDESC hAcqDesc,
    HWND hWnd,
    UINT dwErrorMsg,
    UINT dwLoosingFramesMsg,
    void (CALLBACK *lpfnEndFrameCallback)(HACQDESC),
    void (CALLBACK *lpfnEndAcqCallback)(HACQDESC)
);
```

hAcqDesc

Handle of a structure that contains all needed parameters for acquisition (HACQDESC). If you call **Acquisition_Init** the first time set **hAcqDesc** to **NULL**, in subsequent calls use the former returned value.

hWnd

If the XISL recognizes an end of DMA transfer and it is ready with sorting, it checks if the application called **Acquisition_SetReady** after redrawing. If the application did not call the function, an user defined message (**dwLoosingFramesMsg**) is posted to **hWnd** for further handling. If an error occurred during acquisition also a user defined message (**dwErrorMsg**) is posted to **hWnd**.

dwErrorMsg

Defines a user message that is posted to **hWnd** if an error occurs during acquisition.

dwLoosingFramesMsg

Defines a user message that is posted to **hWnd** if **Acquisition_SetReady** wasn't called by the application at the end of sorting.

lpfnEndFrameCallback

Defines a function pointer that is called after the XISL did the sorting. The prototype for the function is given by:

```
void CALLBACK OnEndFrameCallback(HACQDESC hAcqDesc);
```

In this routine you can do corrections, on-line image processing and redrawing of your data images. Be careful with sending messages from this callback to your application. **lpfnEndFrameCallback** and **lpfnEndAcqCallback** are called from a separate thread which is dissimilar to the applications main thread. That should cause problems if you send messages to your main thread via **SendMessage**. If this causes problems use **PostMessage** instead. If this parameter is set to **NULL** it is ignored.

lpfnEndAcqCallback

Defines a function pointer that is called after the XISL did the sorting. 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.

7.2.8 Acquisition_Acquire_Image

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

UINT WINAPI Acquisition_Acquire_Image(

HACQDESC *hAcqDesc*,
UINT *dwFrames*,
UINT *dwSkipFrames*,
UINT *dwOpt*,
 unsigned short **pwOffsetData*,
DWORD **pdwGainData*,
DWORD **pdwPixelData*

);

hAcqDesc

Pointer to acquisition descriptor structure

dwFrames

Number of frames to acquire is one of the sequence options is set for *dwOpt*. If the continuous option is set this value gives the number of frames in a ring buffer that is used for continuous data acquisition.

dwSkipFrames

Number of frames to skip before a frames is copied into the acquisition buffer.

dwOpt

Options for sequence acquisition: Valid values are

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

Table 54: Supported Acquisition Sequence Options

pwOffsetData

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 don't want to perform an offset correction set this parameter to NULL.

pdwGainData

Pointer that contains gain data. (see *Acquisition_Acquire_GainImage*). If you don't want to perform a gain correction set this parameter to NULL.

pdwPixelData

Pointer to a buffer that contains pixel correction data. *pdwPixelData* points to a linear array of data. Its size is given through $((\text{number of wrong pixels}) * 10 + 1) * \text{sizeof(int)}$. The first entry in a group of nine contains the offset of the pixel from the base pointer of the data array. The other eight entries equal the offset of the correction pixels from the base pointer. If you want to use less than eight pixels for correction, then set the remaining entries to -1. The value of the pixel is replaced by the mean value of the correction pixels. If you don't want to perform a pixel

correction set this parameter to NULL. An easier way to create a pixel map is the use of the XISL function `Acquisition_CreatePixelMap`.

Return values

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

7.2.9 `Acquisition_Acquire_Image_PreloadCorr`

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

UINT WINAPI RETURN `Acquisition_Acquire_Image_PreloadCorr` (

HACQDESC hAcqDesc,

UINT nFrames,

UINT dwSkipFrms,

UINT dwOpt

);

Please find the parameter description above. (`Acquisition_Acquire_Image(..)`)

7.2.10 `Acquisition_Abort`

This routine aborts a currently running acquisition.

UINT WINAPI `Acquisition_Abort`(

HACQDESC hAcqDesc

);

hAcqDesc

Pointer to acquisition descriptor structure

Return values

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

7.2.11 `Acquisition_AbortCurrentFrame`

This routine aborts the transfer of the current frame from the detector to the frame grabber.

The detector will be reset then and a new frame transfer will be started immediately.

The detector should run in the same mode as the image correction files have been

obtained. Therefore it is not recommended to use this function during a measurement.

UINT WINAPI `Acquisition_AbortCurrentFrame`(

HACQDESC hAcqDesc

);

hAcqDesc

Pointer to acquisition descriptor structure

Return values

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

7.2.12 Acquisition_SetCameraMode

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

UINT WINAPI Acquisition_SetCameraMode(

HACQDESC *hAcqDesc*,

UINT *dwMode*

);

hAcqDesc

Pointer to acquisition descriptor structure

dwMode

Must be a number between 0 and 7. The corresponding frame time depends on the used PROM. (see frame times).

Return values

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

7.2.13 Acquisition_SetCorrData

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`.

UINT WINAPI Acquisition_SetCorrData(

HACQDESC *hAcqDesc*,

WORD **pOffsetData*,

DWORD **pGainData*,

DWORD **pPixelCorrList*

);

hAcqDesc

Pointer to acquisition descriptor structure

pOffsetData

Point to offset data (see `Acquisition_Acquire_OffsetImage`).

pGainData

Pointer that contains gain data (see `Acquisition_Acquire_GainImage`).

pPixelCorrList

Pointer to a pixel correction list (see `Acquisition_DoPixelCorrection`).

Return values

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

7.2.14 Acquisition_Acquire_OffsetImage

This function acquires *nFrames*, adds them in a 32 bit buffer and after acquisition the data values are divided by *nFrames* (averaging). The last acquired data at frame end time are available via *pOffsetData*. At the end of the acquisition time the averaged data are also accessible via *pOffsetData*.

The routine returns immediately. If you want to be informed about frame end or acquisition end then define in *Acquisition_Init* the suitable Callback functions and post from there a corresponding message to your application.

UINT WINAPI Acquisition_Acquire_OffsetImage(

HACQDESC hAcqDesc,
unsigned short *pOffsetData,
UINT nRows,
UINT nCols,
UINT nFrames

);

hAcqDesc

Pointer to acquisition descriptor structure.

pOffsetData

Pointer to a acquisition buffer for offset data.

nFrames

Number of frames to acquire.

nRows, nCols

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.

Return values

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

7.2.15 Acquisition_Acquire_OffsetImage_PreloadCorr

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

UINT WINAPI Acquisition_Acquire_OffsetImage_PreloadCorr (

HACQDESC hAcqDesc,
unsigned short *pwOffsetData,
UINT nRows,
UINT nColumns,
UINT nFrames,
UINT dwOpt);

Please find the parameter description above.

(*Acquisition_Acquire_OffsetImage(..)*)

UINT dwOpt must be 0

);

7.2.16 Acquisition_Acquire_GainImage

This function acquires *nFrames* which are all offset corrected by data stored in *pOffsetData*. After that the gain data are added in a 32 bit buffer and after acquisition the data values are divided by *nFrames* (averaging). After averaging the data are further processed for subsequent gain correction of image data. The last acquired data at frame end time are available via *pGainData*. At the end of the acquisition time the gain data are also accessible via *pGainData*.

The offset data are necessary to derive a valid gain image.

The routine returns immediately. If you want to be informed about frame end or acquisition end then define in *Acquisition_Init* the suitable Callback functions and post from there a corresponding message to your application.

UINT WINAPI Acquisition_Acquire_GainImage(

HACQDESC hAcqDesc,

WORD *pOffsetData,

DWORD *pGainData,

UINT nRows,

UINT nCols,

UINT nFrames

);

hAcqDesc

Pointer to acquisition descriptor structure.

pOffsetData

Pointer that contains offset data. (see *Acquisition_Acquire_OffsetImage*). It is recommended to acquire the Offset shortly before calling

Acquisition_Acquire_GainImage.

pGainData

Pointer to buffer that receives the gain data.

nFrames

Number of frames to acquire.

nRows, nCols

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.

Return values

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

7.2.17 Acquisition_Acquire_GainImage_PreloadCorr

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.

UINT WINAPI Acquisition_Acquire_GainImage_PreloadCorr(

HACQDESC hAcqDesc,

DWORD *pGainData,

UINT nRows,

UINT nCols,

UINT nFrames

);

Please find the parameter description above. (*Acquisition_Acquire_GainImage(...)*)

7.2.18 Acquisition_CreatePixelMap

This function specifies the size of or creates a pixel correction map (depending on pCorrList) that one can use in Acquisition_Acquire_Image or Acquisition_DoPixelCorrection.

UINT WINAPI Acquisition_CreatePixelMap(

```

    WORD *pData,
    int nDataRows,
    int nDataColumns,
    int *pCorrList,
    int *pnCorrListSize

```

```
);
```

pData

Pointer to a data source buffer. Defective pixels are marked by setting their value to -1 (0xFFFF). All other pixel values are recognized as good ones.

nDataRows

Number of rows of the data source.

nDataColumns

Number of columns of the data source.

pCorrList

Pointer to an array (pixel map buffer) that receives the pixel correction data. If pCorrList is set to NULL then only the required size of the pixel map buffer is returned in pnCorrListSize.

pnCorrListSize

Pointer to an integer that receives the required size of the pixel map buffer if pCorrList is set to NULL otherwise it contains the size of the pixel buffer at function call time.

7.2.19 Acquisition_DoOffsetCorrection

This function performs an offset correction for the data defined in Acquisition_DefinedDestBuffers. A suitable place to call this function is the end of frame callback function defined by Acquisition_Init.

UINT WINAPI Acquisition_DoOffsetCorrection(

```

    WORD *pSource,
    WORD *pDest,
    WORD *pOffsetData,
    int nCount

```

```
);
```

pSource

Pointer to data source buffer.

pDest

Pointer to data destination buffer. This parameter can be equal to pSource.

pOffsetData

Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). These data must be actual. It is recommended to acquire them shortly before calling Acquisition_DoOffsetCorrection.

nCount

Number of pixels to correct.

Return values

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

7.2.20 Acquisition_DoGainCorrection

This function performs a gain correction for the data defined in Acquisition_DefinedDestBuffers. A suitable place to call this function is the end of frame callback function defined by Acquisition_Init.

```
UINT WINAPI Acquisition_DoGainCorrection(  
HACQDESC hAcqDesc,  
DWORD *pGainData  
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

pGainData

Pointer that contains gain data. (see Acquisition_Acquire_GainImage).

Return values

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

7.2.21 Acquisition_DoOffsetGainCorrection

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

```
UINT WINAPI Acquisition_DoGainCorrection(  
WORD *pSource,  
WORD *pDest,  
WORD *pOffsetData,  
DWORD *pGainData,  
int nCount  
);
```

pSource

Pointer to data source buffer.

pDest

Pointer to data destination buffer. This parameter can be equal to pSource.

pOffsetData

Point to offset data. (see Acquisition_Acquire_OffsetImage).

pGainData

Pointer that contains gain data. (see Acquisition_Acquire_GainImage).

nCount

Number of data entries to correct.

Return values

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

7.2.22 Acquisition_DoPixelCorrection

This function performs a pixel correction for the data defined in Acquisition_DefineDestBuffers. A suitable place to call this function is the end of frame callback function defined by Acquisition_Init.

```
UINT WINAPI Acquisition_DoPixelCorrection(
    WORD *pData,
    int *pCorrList
);
```

***pData**

Pointer to data.

pCorrList

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

Return values

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

7.2.23 Acquisition_IsAcquiringData

This function tests if XISL is about to acquire data.

```
UINT WINAPI Acquisition_IsAcquiringData(
    HACQDESC hAcqDesc
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

Return values

If an acquisition is running a one is returned, otherwise zero.

7.2.24 Acquisition_GetErrorCode

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

```
UINT WINAPI Acquisition_GetErrorCode(
    HACQDESC hAcqDesc,
    DWORD *dwHISError,
    DWORD *dwBoardError
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

dwHISError

Retrieves an error code regarding the XISL itself.

dwBoardError

Retrieves an error code regarding the acquisition board. Please consult the corresponding documentation of your data acquisition board.

Return values

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

7.2.25 Acquisition_Close

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

```
UINT WINAPI Acquisition_Close(  

    HACQDESC hAcqDesc  

);
```

hAcqDesc

Pointer to acquisition descriptor structure.

Return values

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

7.2.26 Acquisition_CloseAll

This function closes shuts down all frame grabbers and serial interfaces currently allocated by the XISL. All acquisition descriptor structures returned by other functions are invalid after this function call.

```
UINT WINAPI Acquisition_CloseAll();
```

Return values

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

7.2.27 Acquisition_SetReady

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.

```
UINT WINAPI Acquisition_SetReady(
```

```
    HACQDESC hAcqDesc,
```

```
    BOOL bFlag
```

```
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

bFlag

Boolean value. Set to zero to signal XISL set redrawing isn't ready, otherwise set to one.

Return values

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

7.2.28 Acquisition_GetReady

Retrieves the draw status of the XISL.

```
UINT WINAPI Acquisition_GetReady(
```

```
    HACQDESC hAcqDesc
```

```
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

Return values

The function returns the flag set by `Acquisition_SetReady`.

7.2.29 Acquisition_GetConfiguration

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.

UINT WINAPI Acquisition_GetConfiguration(

```

    HACQDESC hAcqDesc,
    UINT *dwFrames,
    UINT *dwRows,
    UINT *dwColumns,
    UINT *dwDataType,
    UINT *dwSortFlags,
    BOOL *bIRQEnabled,
    DWORD *dwAcqType,
    DWORD *dwSystemID,
    DWORD *dwSyncMode,
    DWORD      *dwHwAccess

```

);

hAcqDesc

Pointer to acquisition descriptor structure.

dwFrames

Number of frames of acquisition buffer.

dwRows, dwColumns

Number of rows and columns of the sensor.

dwDataType

Type of data of acquisition buffer (should always be two = unsigned short).

dwSortFlags

Type of sorting. This value depends on detector and used sensor (see sorting schemes).

bIRQEnabled

Retrieves a flag that indicates if interrupts are enabled (see Acquisition_Init, hardware interrupts) or if the hardware is running in polling mode. In interrupt mode this parameter is equal to one and zero in the other case.

dwAcqType

Only for internal use.

dwSystemID

PROM identification number of the used detector. This number is only important if the detector operates objectionably and you need any support from PerkinElmer.

dwSyncMode

This parameter can receive the following values:

HIS_SYNCMODE_FREE_RUNNING	The sensor is operating in free running mode.
HIS_SYNCMODE_EXTERNAL_TRIGGER	The sensor is operating in triggered mode. Frames are only send if an external trigger signal is applied to the detector.
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).

Table 55: Description of the SyncMode Options

dwHwAccess

This parameter receives the hardware access parameter, that is programmed into the detector.

If you have to use this parameter (that depends from your contract with PerkinElmer) please contact PerkinElmer for the possible values.

Return values

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

7.2.30 Acquisition_GetIntTimes

This function retrieves the current integration times.

```
UINT WINAPI Acquisition_GetIntTimes(
    HACQDESC hAcqDesc,
    double *pdblIntTime,
    int *nIntTimes
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

***pdblIntTime**

Pointer to an array of 8 byte floating point numbers. This array must contain at least 8 entries.

nIntTimes

This parameter contains the number of maximum entries in the array of 8 byte floating point numbers pointed to by **pdblIntTime*. After return of the function this variable provides the number of available integration times.

Return values

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

Example:

```
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]);
}
}
```

7.2.31 Acquisition_SetAcqData

This routine sets a 32 bit integer 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*.

```
UINT WINAPI Acquisition_SetAcqData(
    HACQDESC hAcqDesc
    DWORD dwData
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

dwData

Data to be setted. To put through more than one parameters define a structure with the required parameters and cast *dwData* to the pointer.

Return values

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

7.2.32 Acquisition_GetAcqData

This routine returns a 32 bit integer that can be set by Acquisition_SetAcqData. These two functions are useful to avoid global variables to put through parameters to the end of frame and end of acquisition callback functions setted by Acquisition_Init.

```
UINT WINAPI Acquisition_GetAcqData(  

    HACQDESC hAcqDesc  

    DWORD *dwData  

);
```

hAcqDesc

Pointer to acquisition descriptor structure.

dwData

Data to be received. To put through more than one parameters define a structure with the required parameters and cast the pointer to dwData.

Return values

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

7.2.33 Acquisition_GetActFrame

This function retrieves the current acquisition frames.

```
UINT WINAPI Acquisition_GetActFrame(  

    HACQDESC hAcqDesc,  

    DWORD *dwActAcqFrame,  

    DWORD *dwActBuffFrame  

);
```

hAcqDesc

Pointer to acquisition descriptor structure.

dwActAcqFrame

Actual frame of acquisition buffer. The acquisition buffer is allocated internally by the frame grabber driver and is not accessible externally.

dwActBuffFrame

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_DefineDestBuffers.

Return values

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

7.2.34 Acquisition_SetFrameSyncMode

This function sets the synchronization mode of the frame grabber. (see 6.1Readout schema)

UINT WINAPI Acquisition_SetFrameSyncMode(

HACQDESC *hAcqDesc*,

DWORD *dwMode*

);

hAcqDesc

Pointer to acquisition descriptor structure.

dwMode

This parameter can have the values:

HIS_SYNCMODE_FREE_RUNNING
HIS_SYNCMODE_EXTERNAL_TRIGGER
HIS_SYNCMODE_INTERNAL_TIMER
HIS_SYNCMODE_SOFT_TRIGGER

Table 56: Trigger option Flags

Return values

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

7.2.35 Acquisition_SetFrameSync

This function supplies a synchronization signal to the detector every time this function is called.

UINT WINAPI Acquisition_SetFrameSync(

HACQDESC *hAcqDesc*

);

hAcqDesc

Pointer to acquisition descriptor structure.

Before calling this function you have to set the frame grabber to a suitable synchronization mode by a call of `Acquisition_SetFrameSyncMode`.

Return values

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

7.2.36 Acquisition_SetTimerSync

This function supplies a synchronization signal to the detector every time *dwCycleTime* is over.

UINT WINAPI Acquisition_SetTimerSync(

HACQDESC *hAcqDesc*,

DWORD **dwCycleTime*

);

hAcqDesc

Pointer to acquisition descriptor structure.

dwCycleTime

Pointer to a 32 bit integer that provides the required cycle time in μ s. After returning of the function this parameter contains the realized cycle time.

Before calling this function you have to set the frame grabber to a suitable synchronization mode by a call of `Acquisition_SetFrameSyncMode`.

Some frame grabbers can realize synchronization cycles only in discreet steps.

That's why the function returns the realized cycle time in *dwCycleTime*.

Return values

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

7.2.37 Acquisition_SetFrameSyncTimeMode

This function sets the synchronization mode of the detector to triggered mode and sets the delay time for the “Data Delivered on Demand with[out] clearance scan “ triggered mode with defined integration time.

UINT WINAPI Acquisition_SetFrameSyncTimeMode(

```
HACQDESC hAcqDesc,
unsigned int uiMode,
unsigned int dwDelayTime
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

uiMode

Trigger Mode must be 0 .. 7

dwDelayTime

Additional delay time in milliseconds (can be 0ms up to (4048 ms-intTime)

Return values

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

Example:

```
// 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);
```

7.2.38 CHwHeaderInfo

Structure that is used to retrieve the contents of detector's hardware header by Acquisition_GetHwHeader.

```
typedef struct
{
    DWORD    dwPROMID;
    DWORD    dwHeaderID;
    BOOL     bAddRow;
    BOOL     bPwrSave;
    DWORD    dwNrRows;
    DWORD    dwNrColumns;
    DWORD    dwZoomULRow;
    DWORD    dwZoomULColumn;
    DWORD    dwZoomBRRow;
    DWORD    dwZoomBRColumn;
    DWORD    dwFrmNrRows;
    DWORD    dwFrmRowType;
    DWORD    dwFrmFillRowIntervals;
    DWORD    dwNrOfFillingRows;
    DWORD    dwDataType;
    DWORD    dwDataSorting;
    DWORD    dwTiming;
    DWORD    dwAcqMode;
    DWORD    dwGain;
    DWORD    dwOffset;
    BOOL     bSyncMode;
    DWORD    dwBias;
    DWORD    dwLeakRows;
} CHwHeaderInfo;
```

DWORD	dwPROMID;	identifies the detector's PROM set
DWORD	dwHeaderID;	identification number of this header type. Header types: 10 all Detectors except AM/AN 11 XRD 16x0 AM 12 13 XRD 16x0 AN, 14 XRD 1621 AN
BOOL	bAddRow;	indicates if an additional row is transferred
BOOL	bPwrSave;	indicates if detector is in power safe mode
DWORD	dwNrRows;	number of sensor rows
DWORD	dwNrColumns;	number of sensor columns
DWORD	dwZoomULRow;	row of the upper left edge of zoom region
DWORD	dwZoomULColumn;	column of the upper left edge of zoom region
DWORD	dwZoomBRRow;	row of bottom right edge of zoom region
DWORD	dwZoomBRColumn;	column of bottom right edge of zoom region
DWORD	dwFrmNrRows;	Number of rows that are used to synthesize the frame scheme of the detector. It results from the number of sensor rows plus the number of rows in which the sensor only integrates charge but doesn't transfer data to the frame grabber plus the number of filling rows.
DWORD	dwFrmRowType;	see Row Types
DWORD	dwFrmFillRowIntervals;	Intervals of 10 nanoseconds to synthesize a frame (see description of hardware header (Header ID <11 otherwise used for int time detection in some header versions)
DWORD	dwNrOfFillingRows	Number of rows of the above mentioned row time.
DWORD	dwDataType;	used for int time detection in some header versions
DWORD	dwDataSorting;	see sorting
DWORD	dwTiming;	selected integration time
DWORD	dwAcqMode;	fixed mode (0), sync mode (1) with fixed frame regime (Header ID <11)
DWORD	dwGain;	1 for Header ID < 11 otherwise Gain mode
DWORD	dwOffset;	
BOOL	bSyncMode	1 if the detector operates in triggered mode else 0.
DWORD	dwBias;	10 V * SensorBias / 255 (Header ID <11)
DWORD	DwLeakRows	Number of rows without driven gates

Table 57: Description of the ChwHeaderInfo structure

BYTE is an unsigned character, USHORT an unsigned 16 bit integer. If HeaderID is zero the whole hardware header is invalid.

For Header-ID 10 the sensor frame time Tint can be derived by the following formula:

$$Tint = (LSBNumberRows + MSBNumberRows * 65536) * RowTime + CycleTime * NumberOfEmptyCycles$$

To get the value for RowTime see Row Types. CycleTime equals 15.625 nanoseconds. During acquisition time a direct access to the hardware header is possible by the XISL function Acquisition_GetHwHeader(..).

For Header-ID 11 and 12 the frame time is

$$Tint = dwFrmNrRows * dwFrmFillRowIntervals / 64$$

For Header-ID 13 the frame time is

$$Tint = dwFrmNrRows * (1 / dwDataType) * 32 * (dwFrmFillRowIntervals / 64)$$

For Header-ID 14 (extended header values are used) the frame time is

$$Tint = dwFrmNrRows * (1 / (wClock / 64)) * 32.0 * (wRowTime / 64);$$

7.2.39 Acquisition_GetHwHeaderInfo

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

UINT WINAPI Acquisition_GetHwHeaderInfo(

HACQDESC hAcqDesc,

CHwHeaderInfo *pInfo

);

hAcqDesc

Pointer to acquisition descriptor structure.

pInfo

Structure of type CHwHeaderInfo that contains the contents of the detector's hardware header necessary for self configuration features.

Return values

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

7.2.40 Acquisition_GetWinHandle

This function retrieves the current acquisition window handle.

UINT WINAPI Acquisition_GetWinHandle(

HACQDESC hAcqDesc,

HWND *hWnd

);

hAcqDesc

Pointer to acquisition descriptor structure.

hWnd

Retrieves the window handle defined in Acquisition_Init.

Return values

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

7.2.41 Acquisition_CreateGainMap

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

UINT WINAPI Acquisition_CreateGainMap(

WORD *pGainData,
WORD *pGainAVG,
int nCount,
int nFrame

);

pGainData

pointer to a sequence of offset corrected data. The images in the array must be sorted by median.

pGainAVG

pointer to a *nFrame* sized array of type word

nCount

Number of pixels to per image.

NFrame

Number of Frames in pGainData

Return values

If the function is successful it returns true otherwise false.

7.2.42 Acquisition_Acquire_Image_Ex

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

UINT WINAPI 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 *pdwPxICorrList

);

hAcqDesc

Pointer to acquisition descriptor structure

dwFrames

Number of frames to acquire is one of the sequence options is set for *dwOpt*. If the continuous option is set this value gives the number of frames in a ring buffer that is used for continuous data acquisition.

dwSkipFrames

Number of frames to skip before a frames is copied into the acquisition buffer.

dwOpt

Options for sequence acquisition: Valid values are

HIS_SEQ_TWO_BUFFERS	0x1	Storage of the sequence into two buffers. Secure image acquisition by separated data transfer and later performed image correction.
HIS_SEQ_ONE_BUFFER	0x2	Storage of the sequence into one buffer. Direct acquisition and linked correction into one buffer.
HIS_SEQ_AVERAGE	0x4	All acquired single images are directly added into one buffer

		and after acquisition divided by the number of frames, including linked correction files.
HIS_SEQ_DEST_ONE_FRAME	0x8	Sequence of frames using the same image buffer
HIS_SEQ_COLLATE	0x10	Skip frames after acquiring frames in a ring buffer
HIS_SEQ_CONTINUOUS	0x100	Continuous acquisition Frames are continuously acquired into a ring buffer of <i>dwFrames</i>

Table 58 : Description of the SyncMode Options

pwOffsetData

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 don't want to perform an offset correction set this parameter to NULL.

dwGainFrames,

number of frames in pwGainData

pwGainData

pointer to a sequence of offset corrected data. The images in the array must be sorted by median.

pwGainAvgData

pointer to the median-list created by Acquisition_CreateGainMap

pdwGainData

Pointer that contains gain data. (see Acquisition_Acquire_GainImage). If you don't want to perform a gain correction set this parameter to NULL.

pdwPixelData

Pointer to a buffer that contains pixel correction data. *pdwPixelData* points to a linear array of data. Its size is given through $((\text{number of wrong pixels}) * 10 + 1) * \text{sizeof}(\text{int})$. The first entry in a group of nine contains the offset of the pixel from the base pointer of the data array. The other eight entries equal the offset of the correction pixels from the base pointer. If you want to use less than eight pixels for correction, then set the remaining entries to -1. The value of the pixel is replaced by the mean value of the correction pixels. If you don't want to perform a pixel correction set this parameter to NULL. An easier way to create a pixel map is the use of the XISL function Acquisition_CreatePixelMap.

Return values

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

7.2.43 Acquisition_SetCorrData_Ex

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.

UINT WINAPI Acquisition_SetCorrData_Ex(

HACQDESC hAcqDesc,
unsigned short *pwOffsetData,
unsigned short *pwGainData,
unsigned short *pwGainAvgData,
UINT nGainFrames,
DWORD *pdwGainData,
DWORD *pdwPxICorrList

);

hAcqDesc

Pointer to acquisition descriptor structure

pwOffsetData

Pointer to offset data (see Acquisition_Acquire_OffsetImage).

pwGainData

Pointer to a sequence of offset corrected data. The images in the array must be sorted by median.

pwGainAvgData

Pointer to the median-list created by **Acquisition_CreateGainMap**

nGainFrames

number of frames in pwGainData

pdwGainData

Pointer that contains gain data (see Acquisition_Acquire_GainImage).

pdwPxICorrList)

Pointer to a pixel correction list (see Acquisition_DoPixelCorrection)

Return values

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

7.2.44 Acquisition_GetCorrData_Ex

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

UINT WINAPI Acquisition_GetCorrData_Ex(

HACQDESC hAcqDesc,

unsigned short **ppwOffsetData,

unsigned short **ppwGainData,

unsigned short **ppwGainAvgData,

UINT **nGainFrames,

DWORD **ppdwGainData,

DWORD **ppdwPxICorrList

);

hAcqDesc

Pointer to acquisition descriptor structure

ppwOffsetData

Pointer to offset data (see Acquisition_Acquire_OffsetImage).

ppwGainData

Pointer to a sequence of offset corrected data. The images in the array must be sorted by median.

ppwGainAvgData

Pointer to the median-list created by **Acquisition_CreateGainMap**

nGainFrames

UINT *pointer retrieving the number of frames in pwGainData

ppdwGainData

Pointer that contains gain data (see Acquisition_Acquire_GainImage).

ppdwPxICorrList)

Pointer to a pixel correction list (see Acquisition_DoPixelCorrection)

Return values

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

7.2.45 Acquisition_DoOffsetGainCorrection_Ex

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

UINT WINAPI Acquisition_DoOffsetGainCorrection_Ex(

WORD *pSource,

WORD *pDest,

WORD *pOffsetData,

WORD *pGainData,

WORD *pGainAVG,


```

        int nCount,
        int nFrame
    )

pSource
    Pointer to data source buffer

pDest
    Pointer to data destination buffer. This parameter can be equal to pSource.

pOffsetData
    Pointer to offset data. (see Acquisition_Acquire_OffsetImage).

pGainData
    Pointer to a sequence of offset corrected data. The images in the array must be
    sorted by median.

pGainAVG
    Pointer to the median-list created by Acquisition_CreateGainMap

nCount
    Number of data entries to correct. (rows*cols)

nFrame
    number of frames in pGainData

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

```

7.2.46 Acquisition_SetCameraGain

This function can be used to set the gain factor of the detector.*

Acquisition_SetCameraGain(

HACQDESC *hAcqDesc*,

WORD *wMode*

);

hAcqDesc

Pointer to acquisition descriptor structure

wmode

Gain factor to set.

For the AM-Type the values of all capacities are added. All bitwise combinations are valid. For example : 3 => 1.3pF.

For the AN-Type the Value in the table is set.

	Detector 16x0 AM
0	0.1pF (allways on)
1	0.3pF
2	0.9pF
4	4.7pF
8	10pF
	Detector 16x0 AN
0	0.25pF (only 1621)
1	0.5pF
2	1 pF
3	2 pF
4	4 pF
5	8 pF

Table 59: Detector gain values

Return values

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

*only available with the new 1640AM/AN and the 1620 AM/AN and full access mode

7.2.47 Acquisition_Acquire_GainImage_EX_ROI

This function is similar to the Acquisition_Acquire_GainImage(..)-function.

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

UINT WINAPI Acquisition_Acquire_GainImage_EX_ROI (

HACQDESC hAcqDesc,

WORD *pOffsetData,

DWORD *pGainData,

UINT nRows,

UINT nCols,

UINT nFrames,

UINT dwOpt,

UINT uiULX, **UINT** uiULY, **UINT** uiBRX, **UINT** uiBRY,

UINT uiMode);

hAcqDesc

Pointer to acquisition descriptor structure.

pOffsetData

Pointer that contains offset data. (see Acquisition_Acquire_OffsetImage). It is recommended to acquire the Offset shortly before calling

Acquisition_Acquire_GainImage.

pGainData

Pointer to buffer that receives the gain data.

nFrames

Number of frames to acquire.

nRows, nCols

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.

dwOpt

must be 0

uiULX, uiULY, uiBRX, uiBRY,

UpperLeftX, UpperLeftY, BottomRightX, BottomRightY

define a rect which is used to calculate the Median for the pixel-gain calculation.

uiMode

0 - normal Gain whole image used for median determination.

1 - Median for pixel-gain calculation from ROI (defined by uiULX...)

2 - Median for pixel-gain calculation from ROI each pixel-gain outside ROI will be set to 1

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

Return values

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

7.2.48 Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr

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.

UINT WINAPI Acquisition_Acquire_GainImage_Ex_ROI_PreloadCorr(

```

    HACQDESC hAcqDesc,
    DWORD *pGainData,
    UINT nRows,
    UINT nCols,
    UINT nFrames,
    UINT dwOpt,UINT uiULX,UINT uiULY,UINT uiBRX,UINT uiBRY,UINT uiMode

```

);

Please find the parameter description above.

7.2.49 CHwHeaderInfoEx

Structure that is used to retrieve the extended detector hardware header by Acquisition_GetHwHeaderInfoEx(..) or GetLatestFrameHeader(..). This function is only available for detectors with HeaderID 14 or above. (1621, 0820)

typedef struct{

```

    WORD wHeaderID;
    WORD wPROMID;
    WORD wResolutionX;
    WORD wResolutionY;
    WORD wNrRows;
    WORD wNrColumns;
    WORD wZoomULRow;
    WORD wZoomULColumn;
    WORD wZoomBRRow;
    WORD wZoomBRColumn;
    WORD wFrmNrRows;
    WORD wFrmRowType;
    WORD wRowTime;
    WORD wClock;
    WORD wDataSorting;
    WORD wTiming;
    WORD wGain;
    WORD wLeakRows;
    WORD wAccess;
    WORD wBias;
    WORD wUgComp;
    WORD wCameratype;
    WORD wFrameCnt;
    WORD wBinningMode;
    WORD wReallntime_milliSec;
    WORD wReallntime_microSec;
    WORD wStatus;

```

} CHwHeaderInfoEx;

Description of structure entries (All entries are 16 bit Values)

WORD	wHeaderID	identifies the used header version (>=14)		
WORD	wPROMID	identifies the detector's PROM set		
WORD	wResolutionX	detectors pixel resolution in x direktion		
WORD	wResolutionY	detectors pixel resolution in y direktion		
WORD	wNrRows	number of sensor rows		
WORD	wNrColumns	number of sensor columns		
WORD	wZoomULRow	row of the upper left edge of zoom region		
WORD	wZoomULColumn	column of the upper left edge of zoom region		
WORD	wZoomBRRow	row of bottom right edge of zoom region		
WORD	wZoomBRColumn	column of bottom right edge of zoom region		
WORD	wFrmNrRows	Number of rows that are used to synthesize the frame scheme of the detector. It results from the number of sensor rows plus the number of rows in which the sensor only integrates charge but doesn't transfer data to the frame grabber.		
WORD	wFrmRowType	Identifies the implemented Row scheme		
WORD	wRowTime	detector row time in 32MhZ Ticks (6bit shifted to the left)		
WORD	wClock	detector row time in Mhz (6bit shifted to the left)		
WORD	wDataSorting	see sorting		
WORD	wTiming;	selected integration time		
WORD	wGain;	Selected detector gain (see table 6.2)		
WORD	wLeakRows;	Number of rows without driven gates		
WORD	wAccess;	Access mode		
WORD	wBias;	selected detector Bias mode		
WORD	wUgComp;	selected detector compensation		
WORD	wCameratype;	Detector type (1 support Binning, 2 supports Binning and special TriggerModes)		
WORD	wFrameCnt;	Internal Frame counter of the detector		
WORD	wBinningMode;	selected detector binning mode: <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; padding: 2px;">Cameratype 1: BitNr set 0 active - default no binning 1 active - 2x2 Binning</td> <td style="width: 50%; padding: 2px;">Cameratype 2: BitNr set 0 active – no binning (default) 1 active 2x2 Binning 8 active Averaged binning 9 active accumulated binning</td> </tr> </table>	Cameratype 1: BitNr set 0 active - default no binning 1 active - 2x2 Binning	Cameratype 2: BitNr set 0 active – no binning (default) 1 active 2x2 Binning 8 active Averaged binning 9 active accumulated binning
Cameratype 1: BitNr set 0 active - default no binning 1 active - 2x2 Binning	Cameratype 2: BitNr set 0 active – no binning (default) 1 active 2x2 Binning 8 active Averaged binning 9 active accumulated binning			
WORD	wRealInttime_milliSec;	measured integration time of actual frame (millisec)		
WORD	wRealInttime_microSec;	measured integration time of actual frame (millisec)		
WORD	wStatus;	detector status word: Bit 0: 0 - OK 1 – Trigger lost Bit 1-3 Triggermode : Value 0: Data Delivered on Demand 1: Data Delivered on Demand with clearance scan 2: Linewise (Start/Stop) 3: Framewise (default)		

Table 60: Description of the CHwHeaderInfoEx structure

7.2.50 Acquisition_GetHwHeaderInfoEx

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. (only available in xisl versions>3-2-0-9)

UINT WINAPI Acquisition_GetHwHeaderInfoEx (

HACQDESC *hAcqDesc*,
CHwHeaderInfo *pInfo ,
CHwHeaderInfoEx *pInfoEx

);

hAcqDesc

Pointer to acquisition descriptor structure.

*pInfo

Pointer to Structure of type CHwHeaderInfo to retrieve the detector's hardware header.

*pInfoEx

Pointer to Structure of type CHwHeaderInfoEx to retrieve the detector's hardware header when available.

Can be NULL.

Return values

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

7.2.51 Acquisition_GetLatestFrameHeader

Use this function to retrieve the last acuiered 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. (only available in xisl versions>3-2-0-9)

UINT WINAPI Acquisition_GetLatestFrameHeader (

HACQDESC *hAcqDesc*,
CHwHeaderInfo *pInfo ,
CHwHeaderInfoEx *pInfoEx

);

hAcqDesc

Pointer to acquisition descriptor structure.

*pInfo

Pointer to Structure of type CHwHeaderInfo to retrieve the detector's hardware header.

*pInfoEx

Pointer to Structure of type CHwHeaderInfoEx to retrieve the detector's hardware header when available.

Can be NULL.

Return values

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

7.2.52 Acquisition_ResetFrameCnt

This function is used to set internal frame counter to zero.

Note: If this function is used during active acquisition the detector Readout time may be affected.

(Only XISL version > 3-2-0-9, HeaderId 14 and Cameratype 1 and 2)

UINT WINAPI Acquisition_ResetFrameCnt (

HACQDESC *hAcqDesc*,

);

hAcqDesc

Pointer to acquisition descriptor structure.

Return values

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

7.2.53 Acquisition_SetCameraBinningMode

Use this function to set the detectors binning mode. (xisl version>3-2-0-9,HeaderId 14 and Cameratype 1 or 2)

UINT WINAPI Acquisition_SetCameraBinningMode (

HACQDESC *hAcqDesc*,

WORD *wMode*

);

hAcqDesc

Pointer to acquisition descriptor structure.

wMode

Binning Mode to be set (Bitwise)

Cameratype 1:

0 active : no binning (default)

1 active: 2x2 binning

Cameratype 2:

as in Cameratype 1 with additional

8 active for AVG Mode

9 active for Sum Mode

Return values

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

7.2.54 Acquisition_GetCameraBinningMode

Use this function to retrieve the detectors binning mode.

(XIS version > 3-2-0-9, HeaderId 14 and Cameratype 1 or 2)

UINT WINAPI Acquisition_GetCameraBinningMode (

HACQDESC *hAcqDesc*,

WORD **wMode*

);

hAcqDesc

Pointer to acquisition descriptor structure.

**wMode*

pointer to value to retrieve the actual binning mode.

Return values

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

7.2.55 Acquisition_SetCameraTriggerMode

Use this function to set the detectors trigger mode. (xisl version>3-2-0-21,HeaderId 14 and Cameratyp 2)

UINT WINAPI Acquisition_SetCameraTriggerMode (

HACQDESC *hAcqDesc*,

WORD *wMode*

);

hAcqDesc

Pointer to acquisition descriptor structure.

wMode

Trigger Mode to be set

0: Data Delivered on Demand

1: Data Delivered on Demand without clearance scan

2: Linewise (Start/Stop)

3: Framewise (default)

Return values

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

Remark:

In section `Acquisition_SetFrameSyncTimeMode(..)` is a code snippet as an example how to use the 'Data delivered on Demand' mode.

7.2.56 Acquisition_GetCameraTriggerMode

Use this function to retrieve the detectors trigger mode.

(XIS version > 3-2-0-21, HeaderId 14 and Cameratyp 2)

UINT WINAPI Acquisition_GetCameraTriggerMode (

HACQDESC *hAcqDesc*,

WORD **wMode*

);

hAcqDesc

Pointer to acquisition descriptor structure.

**wMode*

pointer to value to retrieve the actual trigger mode.

Return values

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

7.3 Demo application

The XISL demonstration application is a simple console application running under Windows OS. It demonstrates how to acquire data continuously, how to acquire a sequence, how to create correction files and how to acquire corrected images.

Files:

The demonstration program was build with MS Visual C++ 6.0. The name of the project file is "XISL_Demo.dsp". If you want to use your own development environment insert XISL.lib in your project.

The header file declaring the required function prototypes is named "Acq.h". The main application program source is called "main.c".

Implementation:

The first task to do is to initialize the frame grabber and the detector.

Initialization:

The XISL is able to recognize all sensors connected to the system automatically.

The following code fragment shows the corresponding function call:

```
//board initialization and self configuration
    //if you want interrupt support set bEnableIRQ to one
    //before calling this function
    if ((nRet = Acquisition_EnumSensors(&dwNumSensors, bEnableIRQ,
FALSE))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
```

In dwNumSensors the number of recognized sensors is returned.

For Polling Mode set *bEnableIRQ* to FALSE.

The last parameter (*bInitAlways*) is only recommended in debug versions of your applications. 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.

This function is used to initialize the sensor by its internal PROM settings. With help of the next code fragment we iterate through all recognized sensors, set further parameters (sensor size, sorting scheme, system id, interrupt settings and so on) and extract information from every sensor.

To start the loop Pos must be initialized by zero.

```
do
{
    int nChannelNr;
    UINT nChannelType;
    if ((nRet = Acquisition_GetNextSensor(&Pos, &hAcqDesc))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
    //ask for communication device type and its number
    if ((nRet=Acquisition_GetCommChannel(hAcqDesc, &nChannelType,
&nChannelNr))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
}
```



```

        //ask for data organization of all sensors
if ((nRet=Acquisition_GetConfiguration(hAcqDesc, &dwFrames, &dwRows, &dwColumns,
&dwDataType, &dwSortFlags,
&dwIRQFlags, &dwAcqType, &dwSystemID, &dwSyncMode,
&dwHwAccess))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
    // now set callbacks and messages for every sensor
if ((nRet=Acquisition_SetCallbacksAndMessages(hAcqDesc, NULL, 0,
0, OnEndFrameCallback, OnEndAcqCallback))!=HIS_ALL_OK)
    {
        //error handling
        goto Exit;
    }
} while (Pos!=0);

```

For further description see `Acquisition_GetNextSensor`, `Acquisition_GetCommChannel`, `Acquisition_SetCallbacksAndMessages` and `Acquisition_GetConfiguration`.

For the further steps we select the last recognized sensor to demonstrate the remaining tasks.

Sequence Acquisition

Now we've got the sensor size and can acquire for instance a sequence of frames. At first we have to allocate an acquisition buffer.

```

//acquire 10 frames
dwFrames = 10;
//allocate acquisition buffer
pAcqBuffer = malloc(dwFrames*dwRows*dwColumns*sizeof(short));
if (!pAcqBuffer)
    {
        //error handling
        return 0;
    }

```

After that we have to inform the XISL of the address of this buffer and its numbers of rows and columns.

```

//route acquisition buffer to XISL
if ((nRet=Acquisition_DefineDestBuffers(hAcqDesc, pAcqBuffer,
dwFrames, dwRows, dwColumns))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }

```

At next I create a scheduler event here that is used to block the main thread from execution after starting the acquisition. In a normal windowed application you should skip this step and post a message to your acquisition window instead of setting a scheduler event in the end of acquisition callback function.

```

//create end of acquisition event
hevEndAcq = CreateEvent(NULL, FALSE, FALSE, NULL);
if (!hevEndAcq)
    {
        //error handling
        return 0;
    }

```

Now we force the XISL to acquire the 10 images.

```

if ((nRet=Acquisition_Acquire_Image(hAcqDesc, dwFrames, 0,
    HIS_SEQ_ONE_BUFFER, NULL, NULL, NULL))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }

```

Now we have to prevent the application from calling other XISL functions during data acquisition except the Acquisition_Abort, Acquisition_DoOffsetCorrection, Acquisition_DoGainCorrection, Acquisition_DoOffsetGainCorrection and Acquisition_DoPixelCorrection, functions. In this console application I block the main thread from execution until the end of acquisition event is signaled by the end of acquisition callback. In a windowed application you cannot use this approach because the message handling of your program would block either. You are responsible by yourself to prevent the above mentioned additional XISL functions from execution (for instance gray corresponding menu items).

```

//wait for end of acquisition event
    WaitForSingleObject(hevEndAcq, INFINITE);
After finishing the acquisition I free the allocated memory.
    free(pAcqBuffer);

```

Continuous Acquisition

Acquiring data continuously is as easy as in the code fragment above. At first we allocate memory for one frame.

```

dwFrames=1;
    pAcqBuffer = malloc(dwFrames*dwRows*dwColumns*sizeof(short));
    if (!pAcqBuffer)
    {
        //error handling
        return 0;
    }

```

After that we have to inform the XISL again about the new buffer address.

```

//route acquisition buffer to XISL
    if ((nRet=Acquisition_DefineDestBuffers(hAcqDesc, pAcqBuffer,
        dwFrames, dwRows, dwColumns))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }

```

Some flags are defined to decide between the different modes of acquisition. Before starting the acquisition an internal variable is set by:

```

//set acquisition data to use it in callback
Acquisition_SetAcqData(hAcqDesc, ACQ_CONT);

```

This variable I extract in the end of frame callback function by:

```

Acquisition_GetAcqData(hAcqDesc, &dwAcqData);

```

But you are free to define for instance a global variable to indicate the different acquisition modes in the callback functions. The above mentioned approach has the advantage of a encapsulation similar to that used in object orientated programming.

Depending from the status flags I decide how to read out the acquisition buffer:

```

if (dwAcqData & ACQ_CONT)
    {

```

```

        sprintf(strBuffer, "acq buffer frame: %d, dest frame %d, row: %d, col: %d,
value: %d\n",
                dwActFrame, dwSecFrame, dwRow, dwCol,
pAcqBuffer[dwColumns*dwRow+dwCol]);
    } else if (dwAcqData & ACQ_OFFSET)
    {
        sprintf(strBuffer, "offset buffer frame: %d, dest frame %d, row: %d, col:
%d, value: %d\n",
                dwActFrame, dwSecFrame, dwRow, dwCol,
pOffsetBuffer[dwColumns*dwRow+dwCol]);
    } else if (dwAcqData & ACQ_GAIN)
    {
        sprintf(strBuffer, "gain buffer frame: %d, dest frame %d, row: %d, col: %d,
value: %d\n",
                dwActFrame, dwSecFrame, dwRow, dwCol,
pGainBuffer[dwColumns*dwRow+dwCol]);
    } else
    {
        sprintf(strBuffer, "acq buffer frame: %d, dest frame %d, row: %d, col: %d,
value: %d\n",
                dwActFrame, dwSecFrame, dwRow, dwCol,
pAcqBuffer[dwRows*dwColumns*(dwSecFrame-1) +dwColumns*dwRow+dwCol]);
    }
    WriteConsole(hOutput, strBuffer, strlen(strBuffer), &dwCharsWritten, NULL);

```

This decision must be made, because the data format and the size of the acquisition buffers depend on the currently running acquisition.

Now we can force the XISL to acquire images until we call `Acquisition_Abort`.

```

//continuous acquisition
    if ((nRet=Acquisition_Acquire_Image(hAcqDesc, dwFrames, 0,
HIS_SEQ_CONTINUOUS, NULL, NULL, NULL))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }

```

In the demonstration the main thread polls now the keyboard until a key is pressed. After that it fires the abort request.

Polling:

```

FlushConsoleInputBuffer(hInput);
do
{
    ReadConsoleInput(hInput, &ir, 1, &dwRead);
}
while(ir.EventType!=KEY_EVENT);
Request and waiting for end of acquisition signal:
Acquisition_Abort(hAcqDesc);
WaitForSingleObject(hevEndAcq, INFINITE);

```

In a windowed application you should prevent all XISL calls until your application got a respond from the end of acquisition callback to the call of `Acquisition_Abort`.

Changing frame time

The detector's frame time can be changed by `Acquisition_SetCameraMode`. The corresponding code fragment shouldn't cause any difficulty:

```
//select another frame time
    printf("\nSet detector mode to 4!\n");
    if ((nRet=Acquisition_SetCameraMode(hAcqDesc, 4))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
}
```

Now another acquisition can be started by the above described procedures.

Note: You can't call this function during a running acquisition.

Changing Detector Mode

The detector's running mode be changed by `Acquisition_SetFrameSyncMode` and the corresponding frame time can be set by `Acquisition_SetTimerSync`. The following code shows this:

```
// Setting of the readout time (detector specific)
Acquisition_SetcameraMode(hAcqDesc, 1)

// change of the syncmode from "freerunning" to "internal trigger" mode
Acquisition_SetFrameSyncMode(hAcqDesc,HIS_SYNCMODE_INTERNAL_TIME
R);

// setting of the integration time in  $\mu$ s (integration time = read out time plus delay time)
time=1000*1000; // time = 1s
Acquisition_SetTimerSync(hAcqDesc,&time);
```

Now another acquisition can be started by the above described procedures.

Note: You can't call this function during a running acquisition.

Acquiring Offset Images

The sensor needs an offset correction to work properly. For this purpose the XISL provides a special function `Acquisition_Acquire_OffsetImage`.

First we have to allocate a buffer for the offset data:

```
//allocate memory for offset data
pOffsetBuffer = malloc(dwRows*dwColumns*sizeof(short));
```

After that we can acquire the frames.

```
//acquire 13 dark frames and average them
if ((nRet=Acquisition_Acquire_OffsetImage(hAcqDesc, pOffsetBuffer, dwRows,
dwColumns, 13))!=HIS_ALL_OK)
{
    //error handler
    return 0;
}
```

The XISL automatically averages them.

The main thread of the demonstration program waits now for the signaled end of acquisition event.

```
//wait for end of acquisition event
WaitForSingleObject(hevEndAcq, INFINITE);
```

In a windowed application you should post a message to you acquisition windows in your end of acquisition callback function.

Acquiring Offset Corrected Data

Because we have offset images available now we can acquire offset corrected images. After allocating the acquisition buffer and informing the XISL about its address we start a continuous acquisition until a key is pressed.

```
//continuous acquisition
    if ((nRet=Acquisition_Acquire_Image(hAcqDesc, dwFrames, 0,
HIS_SEQ_CONTINUOUS, pOffsetBuffer, NULL, NULL))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
    FlushConsoleInputBuffer(hInput);
    do
    {
        ReadConsoleInput(hInput, &ir, 1, &dwRead);
    } while(ir.EventType!=KEY_EVENT);
    Acquisition_Abort(hAcqDesc);
    WaitForSingleObject(hevEndAcq, INFINITE);
```

The offset correction is done here during acquisition. But you are of course free to perform this task after acquisition time in any other program module.

Acquisition_DoOffsetCorrection is optimized for time critical tasks. That's why it is recommended to use it in the callback functions. The offset correction can also be performed by the following code fragment:

```
unsigned short * pEndOffsetPtr = pOffsetBuffer + dwRows * dwColumns;
unsigned short * pEndDataPtr = pAcqBuffer + dwRows * dwColumns;
while(--pEndDataPtr>=pAcqBuffer)
{
    pEndOffsetPtr--;
    * pEndDataPtr -= *pEndOffsetPtr
}
}
```

Acquiring Gain/Offset Data

After we acquired offset images we are able to acquire gain images either. It isn't possible to acquire valid gain images without offset images. First we have to allocate the buffer for the gain image. The XISL uses a 32 bit data format for gain images. That's why the allocation step is a little bit different from that above.

```
pGainBuffer = malloc(dwRows*dwColumns*sizeof(DWORD));
```

Now we have to illuminate the sensor and can acquire the gain images:

```
//acquire 17 bright frames and average them
    if ((nRet=Acquisition_Acquire_GainImage(hAcqDesc, pOffsetBuffer, pGainBuffer,
dwRows, dwColumns, 17))!=HIS_ALL_OK)
    {
        //error handler
        return 0;
    }
}
```

Again we wait for the end of acquisition event signaled (Please refer to the remarks for acquisition of offset images):

```
//wait for end of acquisition event
    WaitForSingleObject(hevEndAcq, INFINITE);
```

Acquiring Offset/Gain corrected data

After allocating the acquisition buffer and informing the XISL about the new address we can start the acquisition. At first we set flags that we extract in the callback functions to inform about continuous acquisition (similar to offset correction):

```
//set acquisition data to use it in callback
    Acquisition_SetAcqData(hAcqDesc, ACQ_CONT);
```

Now we start acquisition:

```
//continuous acquisition
    if ((nRet=Acquisition_Acquire_Image(hAcqDesc, dwFrames, 0,
HIS_SEQ_CONTINUOUS, pOffsetBuffer, pGainBuffer, NULL))!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }
```

You can perform the offset/gain correction by the following code fragment:

```
unsigned short * pEndOffsetPtr = pOffsetBuffer + dwRows * dwColumns;
unsigned short * pEndDataPtr = pAcqBuffer + dwRows * dwColumns;
DWORD *pEndGainPtr = pGainBuffer + dwRows * dwColumns;
DWORD dwValue;
while(--pEndDataPtr>=pAcqBuffer)
{
    pEndOffsetPtr--;
    pEndGainPtr--;
    * pEndDataPtr -= *pEndOffsetPtr
    dwValue = *pEndDataPtr * (*pEndGainPtr);
    dwValue /= 4096;
    *pEndDataPtr = dwValue;
}
```

or by a call of `Acquisition_DoOffsetGainCorrection`.

If you supplied valid pointers to `Acquisition_Acquire_Image` for *pOffsetData* and *pGainData* the correction is automatically executed in the XISL. There is no need to call the upper code fragment.

Like for acquisition of offset corrected images we now poll the keyboard for input and wait for end of all mathematical tasks done by XISL.

```
FlushConsoleInputBuffer(hInput);
do
{
    ReadConsoleInput(hInput, &ir, 1, &dwRead);
} while(ir.EventType!=KEY_EVENT);
Acquisition_Abort(hAcqDesc);
WaitForSingleObject(hevEndAcq, INFINITE);
```

Multiple-Gain Correction

```
// build an use of the multi gain correction
// acquire offset corrected bright images for each ROI
hevEndAcq = CreateEvent(NULL, FALSE, FALSE, NULL);
Acquisition_SetCameraMode(hAcqDesc,0);
Acquisition_SetFrameSyncMode(hAcqDesc,HIS_SYNCMODE_FREE_RUNNING)
;

// 2 point multi-gain first acquire 2 offsetcorrected bright images. one for each
intensity
// and build a sequence of the data
dwFrames=2;
pGainSeqBuffer = malloc(dwFrames*dwRows*dwColumns*sizeof(short));
// buffer for offsetcorrected bright image
pBrightocBuffer=malloc(dwRows*dwColumns*sizeof(short));
Acquisition_SetAcqData(hAcqDesc, ACQ_Brightoc);
if ((nRet=Acquisition_DefineDestBuffers(hAcqDesc, pBrightocBuffer,
1, dwRows, dwColumns))!=HIS_ALL_OK)
```

```

{
    //error handling
    return 0;
}
// acquire offsetcorrected-avg bright image with low intensity
if((Acquisition_Acquire_Image(hAcqDesc,15,0,
    HIS_SEQ_AVERAGE, NULL, NULL, NULL))!=HIS_ALL_OK)
{
    //error handling
    return 0;
}
//wait for end of acquisition event
nRet = WaitForSingleObject(hevEndAcq, INFINITE);
// copy acquired data to sequence buffer
memcpy(pGainSeqBuffer,pBrightocBuffer,dwRows*dwColumns*sizeof(short));

// acquire offsetcorrected-avg bright image with higher intensity
if((Acquisition_Acquire_Image(hAcqDesc,15,0,
    HIS_SEQ_AVERAGE, NULL, NULL, NULL))!=HIS_ALL_OK)
{
    //error handling
    return 0;
}
//wait for end of acquisition event
nRet = WaitForSingleObject(hevEndAcq, INFINITE);
// copy acquired data to sequence buffer
memcpy(pGainSeqBuffer+dwRows*dwColumns,pBrightocBuffer,dwRows*dwColumns*sizeof(short));
// build buffer for median data to be used with the multi-gain correction
pGainSeqMedBuffer= malloc(2*sizeof(short));
//create gain map
Acquisition_CreateGainMap(pGainSeqBuffer,pGainSeqMedBuffer,dwRows*dwColumns,2);
// now acquire image with offset and multigain correction
dwFrames=1;
if (pAcqBuffer)
    free (pAcqBuffer);
pAcqBuffer = malloc(dwRows*dwColumns*sizeof(short));
//route acquisition buffer to XISL
if ((nRet=Acquisition_DefineDestBuffers(hAcqDesc, pAcqBuffer, dwFrames, dwRows,
    dwColumns))!=HIS_ALL_OK)
{
    //error handling
    return 0;
}
Acquisition_SetAcqData(hAcqDesc, ACQ_SNAP);

if((Acquisition_Acquire_Image_Ex(hAcqDesc,30,0,
    HIS_SEQ_AVERAGE,pOffsetBuffer,2,pGainSeqBuffer,pGainSeqMedBuffer,NULL,
    NULL))!=
    HIS_ALL_OK)
{
    //error handling
    return 0;
}

```

```
//wait for end of acquisition event
nRet = WaitForSingleObject(hevEndAcq, INFINITE);
```

Pixel Corrections

It is also possible to replace the values of defective pixels by the averaged values of good neighboring ones. For this purpose we have to define a list containing the defective pixels and the pixels used to correct the defective ones. The list must have a size of $(\text{Number_Of_Defects} * 9 + 1) * \text{sizeof}(\text{DWORD})$.

At first the offset of the defective pixel to the first pixel in the data array in bytes has to be written in the list, on the next address the offset of the first good pixel to the first pixel in the array and so on. If there aren't any more good pixel used for correction than write -1 to the address to mark the end of good pixel list.

On the next address start with the next defective pixel and so on until you have entered all defects you want to correct. To mark the end of defects list set the contents of the next address to -1 . The following code fragment will illuminate that approach:

```
//now we want to perform a pixel correction of pixel
//(100, 22) and pixel (34, 56) by its eight neighbors
pPixelBuffer = (DWORD *) malloc((10*2+1)*sizeof(DWORD));
pPixelPtr = pPixelBuffer;
*pPixelPtr = (100*dwRows+22)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (99*dwRows+21)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (99*dwRows+22)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (99*dwRows+23)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (100*dwRows+21)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (100*dwRows+23)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (101*dwRows+21)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (101*dwRows+22)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (101*dwRows+23)*sizeof(short);
//end of list of correction pixels for pixel
100, 22)
*pPixelPtr = (34*dwRows+56)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (33*dwRows+55)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (33*dwRows+56)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (33*dwRows+57)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (34*dwRows+55)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (34*dwRows+57)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (35*dwRows+55)*sizeof(short);
pPixelPtr++;
*pPixelPtr = (35*dwRows+56)*sizeof(short);
pPixelPtr++;
```



```

        *pPixelPtr = (35*dwRows+57)*sizeof(short);
                                                // end of list of correction pixels for pixel
(34, 56)
        pPixelPtr++;
        *pPixelPtr = -1; //indicates end of list of pixels to correct

```

In the end of frame callback we can do some processing. The pixel correction is automatically executed in the XISL:

```

if (dwAcqData & ACQ_CORR)
{
    ....
}

```

Now we start continuous acquisition and poll for a key event to abort acquisition:

```

if ((nRet=Acquisition_Acquire_Image(hAcqDesc, dwFrames, 0, HIS_SEQ_CONTINUOUS,
NULL, NULL, pPixelBuffer))!=HIS_ALL_OK)
{
    //error handling
    return 0;
}
FlushConsoleInputBuffer(hInput);
do
{
    ReadConsoleInput(hInput, &ir, 1, &dwRead);
} while(ir.EventType!=KEY_EVENT);

Acquisition_Abort(hAcqDesc);
WaitForSingleObject(hevEndAcq, INFINITE);

```

Switching on and off corrections during acquisition

The following code fragment acquires offset gain and pixel corrected images at first for 1 second.

```

if ((nRet=Acquisition_Acquire_Image(hAcqDesc, dwFrames, 0, HIS_SEQ_CONTINUOUS,
pOffsetBuffer, pGainBuffer, pPixelBuffer))!=HIS_ALL_OK)
{
    //error handling
    return 0;
}
Sleep(1000);

```

Now it switches off all corrections:

```

if ((nRet=Acquisition_SetCorrData(hAcqDesc, NULL, NULL, NULL))!=HIS_ALL_OK)
{
    //error handling
    return 0;
}

```

Now we acquire continuously uncorrected data for 1 second:

```

Sleep(1000);

```

And now we switch on only offset corrections and acquire 1 second:

```

if ((nRet=Acquisition_SetCorrData(hAcqDesc, pOffsetBuffer, NULL, NULL))!=HIS_ALL_OK)

```

```

    {
        //error handling
        return 0;
    }
    Sleep(1000);

```

Now we finish acquisition:

```

    Acquisition_Abort(hAcqDesc);
    WaitForSingleObject(hevEndAcq, INFINITE);

```

Close the XISL

The following code fragment closes the XISL and does all clean up.

```

//close acquisition and clean up
    //free event object
    CloseHandle(hevEndAcq);
    hevEndAcq = NULL;
    free(pAcqBuffer);
    free(pOffsetBuffer);
    free(pGainBuffer);
    free(pPixelBuffer);

    if ((nRet=Acquisition_CloseAll())!=HIS_ALL_OK)
    {
        //error handling
        return 0;
    }

```

7.4 XISL error codes

The following table lists all error codes of the XISL and gives a short description of them. The symbolic names of the errors are defined in Acq.h”.

value	symbolic name	meaning
0	HIS_ALL_OK	No error
1	HIS_ERROR_MEMORY	Memory couldn't allocated.
2	HIS_ERROR_BOARDINIT	Unable to initialize board.
3	HIS_ERROR_NODETECTOR	Got a time out for acquisition. May be no detector present.
4	HIS_ERROR_CORRBUFFER_INCOMPATIBLE	Your correction files didn't have a proper size.
5	HIS_ERROR_ACQ_ALREADY_RUNNING	Unable to initialize board or allocate DMA buffer because a acquisition is running.
6	HIS_ERROR_TIMEOUT	Got a time out from hardware.
7	HIS_ERROR_INVALIDACQDESC	Acquisition descriptor invalid
8	HIS_ERROR_VXDNOTFOUND	Unable to find VxD.
9	HIS_ERROR_VXDNOTOPEN	Unable to open VxD.
10	HIS_ERROR_VXDUNKNOWNERROR	Unknown error during VxD loading.
11	HIS_ERROR_VXDGETDMAADR	VxD Error: GetDmaAddr failed.
12	HIS_ERROR_ACQABORT	An unexpected acquisition abort occurred.
13	HIS_ERROR_ACQUISITION	An error occurred during data acquisition.
14	HIS_ERROR_VXD_REGISTER_IRQ	Unable to register interrupt.
15	HIS_ERROR_VXD_REGISTER_STATADR	Register status address failed.
16	HIS_ERROR_GETOSVERSION	Getting version of operating system failed.
17	HIS_ERROR_SETFRMSYNC	Can't set frame sync.
18	HIS_ERROR_SETFRMSYNCMODE	Can't set frame sync mode.
19	HIS_ERROR_SETTIMERSYNC	Can't set timer sync.
20	HIS_ERROR_INVALID_FUNC_CALL	Function was called by another thread than Acquisition_Init.
21	HIS_ERROR_ABORTCURRFRAME	Aborting current frame failed
22	HIS_ERROR_GETHWHEADERINFO	Getting hardware header failed
23	HIS_ERROR_HWHEADER_INF	Hardware header is invalid
24	HIS_ERROR_SETLINETRIG_MODE	Setting line trigger mode failed
25	HIS_ERROR_WRITE_DATA	Writing data failed
26	HIS_ERROR_READ_DATA	Reading data failed
27	HIS_ERROR_SETBAUDRATE	Setting baud rate failed
28	HIS_ERROR_NODESC_AVAILABLE	No acquisition descriptor available
29	HIS_ERROR_BUFFERSPACE_NOT_SUFF	Buffer space not sufficient
30	HIS_ERROR_SETCAMERAMODE	Setting detector mode failed
31	HIS_ERROR_FRAME_INV	Frame invalid
32	HIS_ERROR_SLOW_SYSTEM	System to slow
33	HIS_ERROR_GET_NUM_BOARDS	Error during getting number of boards
34	HIS_ERROR_ALREADY_OPEN_BY_ANOTHER_PROCESS	Communication channel already opened by another process
35	HIS_ERROR_CREATE_MEMORYMAPPING	Error creating memory mapped file
36	HIS_ERROR_VXD_REGISTER_DMA_ADDRESS	Error registering DMA address
37	HIS_ERROR_VXD_REGISTER_STAT_ADDR	Error registering static address
38	HIS_ERROR_VXD_UNMASK_IRQ	Unable to unmask interrupt
39	HIS_ERROR_LOADDRIVER	Unable to load driver
40	HIS_ERROR_FUNC_NOTIMPL	Function is not implemented
41	HIS_ERROR_MEMORY_MAPPING	Unable to map memory
42	HIS_ERROR_CREATE_MUTEX	Mutex couldn't created
43	HIS_ERROR_ACQ	Error during acquisition
44	HIS_ERROR_DESC_NOT_LOCAL	Acquisition descriptor is not local
45	HIS_ERROR_INVALID_PARAM	Invalid Parameter
46	HIS_ERROR_ABORT	Error during abort acquisition
47	HIS_ERROR_WRONGBOARDSELECT	The wrong board is selected
48	HIS_ERROR_WRONG_DETECTOR_MODE	Change of Detector Mode during Acquisition

Table 61: Description of the SyncMode Options

7.4.1 Frame Grabber error codes

The following table lists error codes of the XISL/FrameGrabber and gives a short description of them.

value	symbolic name	meaning
Warnings:		
4	EL_W_WRONGREVISIONCRC	Wrong CRC in hardware revision EEPROM
3	EL_W_ACQWINDOWTOOBIG	Acquisition window too big for the detector selected; will be fit automatically
2	EL_W_INLUTINDEXTOOBIG	The requested entry of the input look-up table does not exist. Valid are 0..255
1	EL_W_HWALREADYOPENED	The hardware has been opened without subsequent close. This may indicate that another task uses the DLL already or the DLL was not closed properly by an aborted task which can be tolerated.
0	EL_UNKNOWNERROR	Unexpected Error
Errors:		
-1	EL_E_WRONGBOARDSELECT	Board select parameter in function el_OpenHW invalid.
-2	EL_E_HWNOPENED	Hardware has not been opened - call el_OpenHW prior to the offending call.
-3	EL_E_BIOSNOTCORRECT	PCI Bios may not be present. The BIOS call Find PCI Bios did not return correct values. This call verifies no hardware access yet, it checks only that the BIOS can handle PCI functions and that it complies with PCI rev. 2.0.
-4	EL_E_NOPCEYEFIND	PC_EYE board could not be found on PCI. This indicates that the PCI Bios of the computer is not capable of finding the PC_EYE board. PC_EYE boards can be identified by the driver in a unique way (Find PCI device and software-readable signature string).
-5	EL_E_PCEYESYSTEMMEMORY	The driver allocates n MB at startup time of Windows (n set in SYSTEM.INI); this may have failed at Windows start and is detected only now. The computer may not have enough memory installed.
-6	EL_E_FRAMEBUFALLOC	Memory for the frame buffer in the requested size could not be allocated. Closing other applications may help.
-7	EL_E_CONTEXTNOTINIT	The driver-internal context structure must be initialized first by calling el_InitContext.
-8	EL_E_HWNOPENED	Call el_InitHW before using other functions.
-9	EL_E_PITCHTOOSMALL	The acquisition pitch is smaller than the horizontal image size.
-10	EL_E_MEMORYALLOC	Internal memory allocation failed. Closing other applications may help.
-11	EL_E_WRONGDETECTORSELECT	An invalid detector input number (0..3) is given.
-12	EL_E_ACQWINDOWTOOBIG	The acquisition window is too big for the detector selected. May indicate not enough memory. See description of el_NewMemBuffer.
-13	EL_E_WRONGBOARDID	No board with this ID is open.
-21	EL_E_UNKNOWNACQMODE	A number for a non-existing acquisition mode is given.
-22	EL_E_FUNCNOTAVAILABLE	Not implemented yet or wrong function code.
-23	EL_E_ACQTIMEOUT	The driver waits a certain time (about 5 frame times) for the acquisition to finish. This error occurs when the detector is not connected or powered down.
-24	EL_E_INVALIDPARAMETER	Invalid function parameter supplied.
-25	EL_E_INVALIDPOINTER	Invalid pointer supplied.
-64	EL_E_WRONGDETECTOR	unknown detector
-300	EL_E_IRQNOTIMPLEMENTED	IRQs are not implemented on this platform.
-301	EL_E_IRQISENABLED	IRQ is enabled.
-302	EL_E_IRQNOTENABLED	IRQ is not enabled.
-303	EL_E_IRQNOTAVAILABLE	IRQs are not available, check correct driver load order.
-304	EL_E_IRQINVALIDEVENT	Invalid IRQ-event type, use one of 'EL_IRQ_...!'
-305	EL_E_IRQINVALIDBOOST	Invalid priority boost.
-306	EL_E_IRQOPENEVENT	Error opening event.
-307	EL_E_IRQINTERNALERROR	Internal error setting up IRQs.

Table 62: Description of the Frame Grabber Error Codes

Worldwide Headquarters**PerkinElmer Optoelectronics**

44370 Christy Street
Fremont, CA 94538-3180
Telephone: +1 510-979-6500
Toll free: (North America) +1 800-775-OPTO (6786)
Fax: +1 510-687-1140
Email: opto@perkinelmer.com
www.optoelectronics.perkinelmer.com

European Headquarters**PerkinElmer Optoelectronics**

Wenzel-Jaksch-Str. 31
65199 Wiesbaden, Germany
Telephone: (+49) 611-492-247
Fax: (+49) 611-492-170
Email: opto.Europe@perkinelmer.com

Asia Headquarters**PerkinElmer Optoelectronics**

47 Ayer Rajah Crescent #06-12
Singapore 139947
Telephone: (+65) 6775-2022
Fax: (+65) 6775-1008
Email: opto.Asia@perkinelmer.com



For a complete listing of our global offices, visit www.optoelectronics.perkinelmer.com

©2006 PerkinElmer, Inc. All rights reserved. The PerkinElmer logo and design are registered trademarks of PerkinElmer Inc., or its subsidiaries, in the United States and other countries. All other trademarks not owned by PerkinElmer, Inc. or its subsidiaries that are depicted herein are the property of their respective owners. PerkinElmer reserves the right to change this document at any time without notice and disclaims liability for editorial, pictorial or typographical errors.
600122_03 USM0807