



PerkinElmerTM
optoelectronics.

HIS

Reference Book

Heimann Imaging Software
Version 2.4



GENERAL.....	1
ABOUT THE PROGRAM HIS	1
GETTING STARTED - THE FIRST IMAGE	1
1. Introduction.....	1
2. General considerations	1
3. The first image	2
HOW TO PERFORM CORRECTIONS.....	2
MATHEMATICAL DESCRIPTION OF CORRECTIONS.....	3
Offset correction	3
Gain correction.....	3
Pixel correction.....	3
SORTING SCHEMES OVERVIEW	3
HOW TO ACQUIRE DATA FROM SEVERAL SENSORS PARALLEL	4
HOW TO USE FUNCTION KEYS.....	4
WHAT IS NEW IN HIS VERSION 2.4.....	4
HOW TO GET UPDATES.....	4
INSTALLATION.....	5
INSTALLATION OF HEIMANN IMAGING SOFTWARE	5
Hardware Installation for both types of frame grabber.....	5
Installation of the PcEye3 on Windows®NT or Windows®95	5
Installation of the Squirrel on Windows®XP, Windows®2000 or Windows®98(ME).....	6
Installation of the Squirrel on Windows®NT.....	6
TROUBLE SHOOTING.....	6
DUAL PROCESSOR BOARD COMPATIBILITY	7
HEIMANN IMAGING SOFTWARE	8
OVERVIEW OF THE MENU COMMANDS	8
File Menu Commands	8
Edit Menu Commands.....	8
Acquire Menu Commands.....	9
Camera Menu Commands.....	9
View Menu Commands.....	10
Window Menu Commands.....	10
Options Menu Commands.....	10
Help menu commands	10
FILE MENU	11
New command.....	11
Open command	11
Close command.....	12
Print command.....	12
Print Preview	12
Print Setup	13
Save command.....	13
Save As command.....	13
Save Correction files.....	13
Exit command.....	13
Import command.....	13
1, 2, 3, 4 command.....	13
EDIT MENU.....	14
Edit build sequence	14
Edit Math	15
Copy.....	15

Copy Frames	15
Average	15
Select by Value	15
Select All	16
Deselect All	16
Set Value	16
Create Pixel Map	16
ACQUIRE MENU	17
Single Shot	17
Sequence Command	17
Continuous Command	18
Get Offset Image	18
Get Gain/Offset Image	19
Get All Offset Images	19
Link Offset Correction	19
Link Gain Correction	19
Link Pixel Correction	19
Convert to Gain Image	20
End of Acquisition	20
CAMERA	21
Timings menu	21
Camera Mode	21
VIEW MENU	22
Status Bar	22
Toolbar	22
Acquisition Bar	22
LUT - Look-Up-Table	22
Player	22
View Palette	22
View Control Box	23
WINDOW MENU	24
New Window	24
Cascade	24
Tile Horizontal	24
Tile Vertical	24
Window Arrange Icons	24
1, 2, ... command	24
OPTIONS MENU	25
Acquisition	25
View	26
Options Active Sensor	27
Options Sensor	27
HELP	28
Contents	28
Index	28
Using Help	28
About	28
TOOLBAR	29
ACQUISITION TOOLBAR	29
STATUS BAR	30
TITLE BAR	30
Scroll bars	30
Size command (System menu)	31
Move command (Control menu)	31
Minimize command (application Control menu)	31

<i>Maximize command (System menu)</i>	31
<i>Next Window command (document Control menu)</i>	31
<i>Previous Window command (document Control menu)</i>	32
<i>Close command (Control menus)</i>	32
<i>Restore command (Control menu)</i>	32
<i>Switch to command (application Control menu)</i>	32
STANDARD DIALOGS	33
<i>File Selection Dialog</i>	33
<i>Overwrite data dialog</i>	34
<i>Choose Directory Dialog</i>	34
APPLICATION.....	35
MATHEMATICAL EXPRESSIONS.....	35
<i>Parsing expression:</i>	36
IMAGE CORRECTION	38
<i>Use of the Offset Correction</i>	38
<i>Use of the Gain/Offset Correction</i>	38
<i>Use and generation of the Pixel Correction</i>	39
<i>Correct already acquired images</i>	39
ACQUISITION CONTROL MODES	40
WARNING TABLE BY USING THE DETECTOR AND ITS STATUS	41
DETAILS FOR THE HARDWARE	42
READOUT SCHEMA.....	42
EXTERNAL TRIGGER	42
FILE FORMAT	44
DESCRIPTION OF HARDWARE HEADER	45
ROW TYPES	46
INTERRUPT SOURCES.....	47
POLLING MODE	48
SORTING	48
<i>Sorting schemes overview</i>	48
<i>Sorting RTM128</i>	49
<i>Sorting RID256</i>	49
<i>Sorting RID128-400</i>	50
<i>Sorting RID1024-100</i>	51
<i>Sorting RID512-400 A0</i>	52
<i>Sorting RID512-400 A1/A2</i>	53
<i>Sorting RID512-400 E</i>	54
<i>Sorting RID1640 A // RID 1620 A</i>	55
HEIMANN SOFTWARE LIBRARY	57
HEIMANN SOFTWARE LIBRARY OVERVIEW	57
HSL FUNCTIONS.....	58
<i>Acquisition_Descriptor</i>	58
<i>Acquisition_EnumSensors</i>	58
<i>Acquisition_GetNextSensor</i>	58
<i>Acquisition_Init</i>	60
<i>Acquisition_GetCommChannel</i>	61
<i>Acquisition_DefineDestBuffers</i>	61
<i>Acquisition_SetCallbacksAndMessages</i>	62
<i>Acquisition_Acquire_Image</i>	63
<i>Acquisition_Abort</i>	64
<i>Acquisition_AbortCurrentFrame</i>	64
<i>Acquisition_SetCameraMode</i>	64

<i>Acquisition_SetCorrData</i>	65
<i>Acquisition_Acquire_OffsetImage</i>	65
<i>Acquisition_Acquire_GainImage</i>	66
<i>Acquisition_CreatePixelMap</i>	66
<i>Acquisition_DoOffsetCorrection</i>	67
<i>Acquisition_DoGainCorrection</i>	67
<i>Acquisition_DoOffsetGainCorrection</i>	68
<i>Acquisition_DoPixelCorrection</i>	68
<i>Acquisition_IsAcquiringData</i>	69
<i>Acquisition_GetErrorCode</i>	69
<i>Acquisition_Close</i>	69
<i>Acquisition_CloseAll</i>	70
<i>Acquisition_SetReady</i>	70
<i>Acquisition_GetReady</i>	70
<i>Acquisition_GetConfiguration</i>	71
<i>Acquisition_GetIntTimes</i>	72
<i>Acquisition_SetAcqData</i>	72
<i>Acquisition_GetAcqData</i>	73
<i>Acquisition_GetActFrame</i>	73
<i>Acquisition_SetFrameSyncMode</i>	74
<i>Acquisition_SetFrameSync</i>	74
<i>AcquisitionSetTimerSync</i>	74
<i>CHwHeaderInfo</i>	75
<i>Acquisition_GetHwHeaderInfo</i>	76
<i>Acquisition_GetWinHandle</i>	76
DEMO APPLICATION.....	77
HSL ERROR CODES.....	86
<i>Frame Grabber error codes</i>	87

General

About the program HIS

The program HIS is capable to demonstrate the functions of the **RID (Radiation Image Detector)**. It can be used with any type of RID through the installed PCI 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 and **Gain/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 and Gain/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 HIS is embedded in the WINDOWS'9x/WINDOWS NT/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 HIS 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 gray levels for optimum presentation.

Getting Started - The first image

1. Introduction

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

2. General considerations

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

Each column for example 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 for example is arranged in groups of 128 readout channels. The groups can deviate in their gain that one can distinguish blocks of 128 channels in a bright image caused by this gain differences.

At least 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.

3. The first image

It is provided that 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 Mode. In both cases the system attempts to initialize the frame grabber board(s) and the connected cameras. The "Cancel" button starts the program without initialization. The initialization can take some time which depends of the number of frame grabber boards and cameras. If more than one camera is connected a dialog appears that contains a list of all recognized cameras and an active camera 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 a gray image is displayed the parameters of x-ray source and detector are in a good range. If a white or a black image is displayed the image can be perform 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

How to perform corrections

The **RID** works as an independent camera to acquire X-ray images. After starting the HIS.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).

Similarly to CCD based cameras, RID detectors need an Offset correction to take into account dark current of each pixel. In particular during the warm-up phase of the detector, the Offset correction should be repeated periodically.

Additionally, a Gain correction is necessary to homogenize differences in pixel sensitivities or to take into account of the X-ray beam illumination. The Gain correction should be proceeded under optimum illumination of the sensor (70-80 % of the full range) and it is automatically Offset corrected. The use of an Offset corrected Gain calibration eliminates offset dependency and therefore any stored Gain correction can be used for a specific frame time for longer time periods.

The Pixel correction allows a 'software repair' of defect 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 BADPIX.HIS. The delivery package includes the BADPIX.HIS file for the specific detector, the user can also generate its 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 of 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 HIS software sends out a warning if not all acquired images are accepted by the computer and displayed on the screen.

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.

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.

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.

Pixel correction

The pixel correction is done by the HSL 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 HIS 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.

Sorting schemes overview

Depending on the sensor and camera type the data come in different orders from the camera. The HSL 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 camera type is unknown the HIS comes up with a camera type dialog at initialization, where the correct sorting has to be entered. The following camera types and sortings are supported:

RID128	1
RID256	2
RID128-400	3
RID1024-100	4
RID512-400 A0	5
RID512-400 A1/A2	6
RID512-400 E	7
RID 1640 A	8
RID 1620 A	8

How to acquire data from several sensors parallel

This feature of the HIS can be used by plugging in two or more frame grabber boards and connecting them to suitable cameras. But it is not recommended to use different types of frame grabbers together. During the HIS 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.

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 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.
F9	BAD PIXEL	Enable/disable Pixel correction.
<ESC>	STOP	Stop Acquisition

What is new in HIS version 2.4

- The new camera family RID 16xx is integrated.
- The new frame grabber Squirrel is supported
- HIS/HSL are running under Windows 2000 and Windows XP with the new FG Squirrel
- Some datatypes of the HSL (former CSL) have changed during development of Version 2.2.
- A new Acquisition toolbar is integrated

How to get Updates

The server address is 165.88.199.17, the user name is "opto_guest" and the password is "opto". After login change to the directory "opto/output/HISxx" whereby "xx" is standing for the version number (e.g. "23"). Please be on lookout for readme-files containing the newest information for these setup-files and the HIS-generation.

The folder "Full_Inst" contains the full installation. Download these files to a temporary folder. Start the "HIS_Setup.exe" and follow the instructions of the setup.

The folder "Update" contains update files for a current version of HIS, at least HIS 2.2. Download the update files in a temporary folder and start the setup file.

Important hint: It isn't possible to update an older version of HIS (e.g. Version 2.1) by executing a patch of a newer version. In this case uninstall the older version before starting the new installation.

If you don't have any ftp services available please contact our customer support service.

Installation

Installation of Heimann Imaging Software

The software supports the PcEye3 frame grabber family of the German company "Eltec" and the Squirrel frame grabber developed by PerkinElmer Optoelectronics.

The PcEye3 works with the operation systems Windows®NT and Windows®95. The Squirrel works with the operation systems Windows®XP, Windows®2000, Windows®NT and Windows®98(ME). It is not recommended to use both types of frame grabber together, but each grabber can be used multiple.

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 for both types of frame grabbers 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, but beware only one type of frame grabber can be used in one computer system.

Hardware Installation for both types of frame grabber

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

Installation of the PcEye3 on Windows®NT or Windows®95

1. Plug in the HIS Installation CD-ROM
2. If the Setup does not start automatically start the START.EXE in the root directory of the CD
3. Select the HIS SETUP in the appearing menu
4. The SETUP program will lead you through the installation process
5. Select the PcEye3 as your favorite frame grabber
6. If you use WINDOWS 95 make sure that DirectX is installed.
A new version of DirectX is in the folder DirectX on the Installation CD.
7. Restart the computer system
8. The HIS is now ready to start
9. If the initialization of the frame grabber and the camera is successful a corresponding message appears in the status bar.

Installation of the Squirrel on Windows®XP, Windows®2000 or Windows®98(ME)

1. After LOG ON the Hardware Wizard notice the new frame grabber as an multimedia device
2. Plug in the HIS Installation CD-ROM
3. Follow the hardware wizard to install the Squirrel as a new device
4. After installation of the Squirrel and HSL drivers by the Wizard start the HIS 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 HIS SETUP program will lead you through the installation process
6. Restart the computer system
7. The HIS is now ready to start
8. If the initialization of the frame grabber and the camera is successful a corresponding message appears in the status bar.

Installation of the Squirrel on Windows®NT

1. Plug in the HIS Installation CD-ROM
2. If the Setup does not start automatically start the START.EXE in the root directory of the CD
3. Select the HIS SETUP in the appearing menu
4. The SETUP program will lead you through the installation process
5. Select the Squirrel as your favorite frame grabber
6. Restart the computer system The HIS is now ready to start
7. If the initialization of the frame grabber and the camera is successful a corresponding message appears in the status bar.

Trouble Shooting

Setup:

Color 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.

HIS: Initialization:

HIS Error 2 // Eltec Error -4: The Eltec-board cannot be found

- a) IRQ-Problem
 - WINDOWS 95: Change the IRQ-Settings in the Settings **Control Panel/System**
 - WINDOWS NT: Change the IRQ-Settings the **BIOS**Use an other PCI-Slot without IRQ-Sharing or remove not absolutely necessary PCI-boards
- b) A false driver for the PCI frame grabber is loaded (WINDOWS 9x / PcEye3)
To correct this problem, choose "Settings/Control Panel" in the Windows "Start" menu and double click the icon "System". Select the property page "Device Manager". Look for an unknown PCI SCSI HOST controller. Delete it by pressing the "Remove" button or if none is displayed nothing is to do instead. Now restart Windows. At start up the operating system recognizes a new device and prompts to load suitable drivers automatically. Always confirm the current default selections until

system boot is done. After that call the "Device Manager" again. Double click the device "Unknown SCSI Host device". Select the appearing property page "General". Uncheck the all check boxes in the group "Device Usage" and start the system again. Now all currently loaded device drivers are disabled and the HIS can load its own drivers now. There isn't any need to install the HIS again.

c) The ELTEC board is not plugged correctly (PcEye3)

If this is ok it might be that you have trouble with your PCI bus. Just plug the board into another computer and look if Windows recognizes the board as unknown device. Don't do any installations in this step. If the board is recognized by another system change your main board. If the other system doesn't recognize the frame grabber too then contact the vendor.

d) The ELTEC board is not plugged correctly (Squirrel)

If this is ok it might be that you have trouble with your PCI bus. Restart the computer and view for an multimedia board in startup table of installed boards. If not plug the board into another computer and look this bios recognizes the board as multimedia device. If the board is recognized by another system change your main board. If the other system doesn't recognize the frame grabber too then contact the vendor.

HIS Error 2 // Eltec Error –303

IRQ-Problem

Change the IRQ-Settings in the System Settings Control Panel (Win9x) or in the BIOS (WIN NT).

Use an other PCI-Slot without IRQ-Sharing or remove not absolutely necessary PCI-boards.

Use the polling mode.

HIS Error 2 // Eltec Error –41: Virtual device driver not present

No PcEye driver is loaded. Start PcEye.sys in the "Settings/Control Panel/ Devices" again.

If this false, check if exists a correct PcEye.sys in the system32/device folder of WINDOWS NT.

HIS 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 camera setup by the Option/Acquisition dialog

Write down the header information and contact your vendor

Dual Processor Board Compatibility

	PcEye3	Squirrel	Chip
ASUS CUV4X-D	Yes*	Yes	VIA Apollo Pro 133A
ASUS CUR-DLS	No	Yes*	Server Works LE 3.0
ASUS P2B-DS	Yes	Yes	
ASUS P2L-DS	Yes	Yes	Intel 440LX
ASUS A7M266-D	NO	Yes	AMD 762
Tyan S205 Tiger 200	Yes	Yes	VIA Apollo Pro 133A
Tyan S2567 Thunder HE	No	Yes*	Server Works HE

* newest BIOS and Windows mainboard drivers are needed

Heimann Imaging Software

Overview of the Menu Commands

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 HIS.

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.
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.

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 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)
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.

Camera Menu Commands

The Camera menu offers the following commands:

Mode	Selects the camera mode. The camera is able to operate the following modes: <ul style="list-style-type: none"> • free running, • triggered by external sources • triggered by an customized internal timing
Timings	Selects the currently active frame time for camera operation.

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).

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.

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 than one sensor is connected to the system.
Sensor	Gives information and allows to set some options for the active sensor

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.

File menu

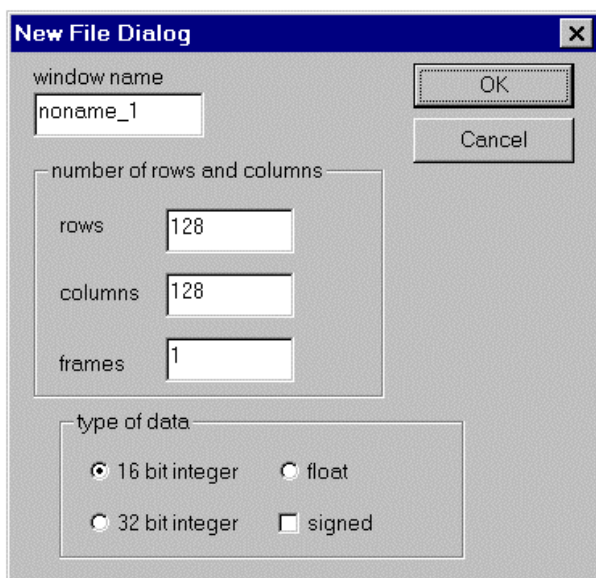
New command

This command creates a new document in HIS. 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



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 HIS supports different data types for data acquisition and evaluation. The suitable type could be specified here. All operations done by the HIS 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.

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

Close command

The Close command closes all windows containing the active document. HIS 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, HIS 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:



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.

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.


Print Setup

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

Save command

The Save command saves the active document to its current name and directory. Saving a document for the first time, HIS 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

Save As command

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

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

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.

Exit command

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

Shortcuts

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



Keys: ALT+F4

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.

1, 2, 3, 4 command

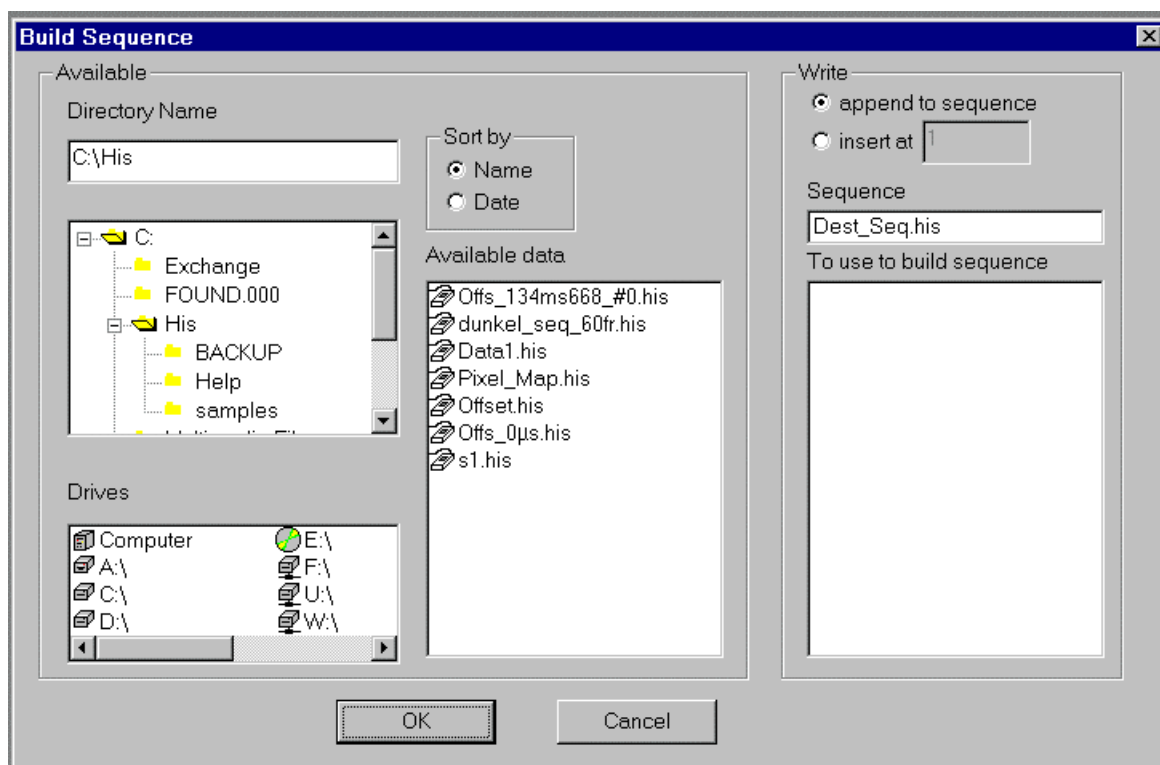
Use the numbers and filenames listed at the bottom of the File menu to open the last four documents you closed. Choose the number that corresponds with the document you want to open.

Edit menu

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



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 appended to or inserted into the sequence. Drag and drop files from the "available data" list box.

Edit Math

The Edit Math menu entry opens the “enter expression” dialog box. The build 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 describe in the chapter “Mathematical expressions”.

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

Copy Frames

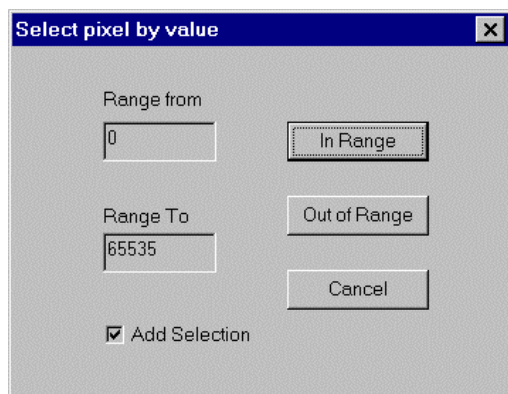
This command copies a number of frames into a new document. In the appearing dialog the range of frames can be specified.

Average

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

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.



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.

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.

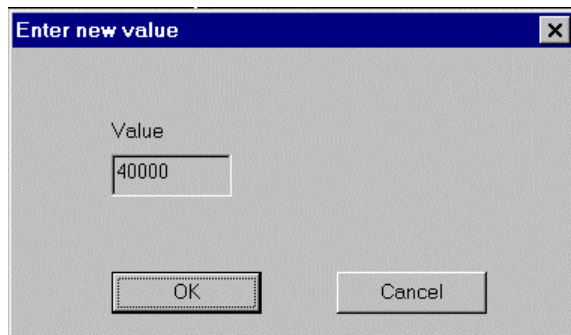
Deselect All

This command deselects all pixels of the active document.

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:



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

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.

Acquire menu

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.

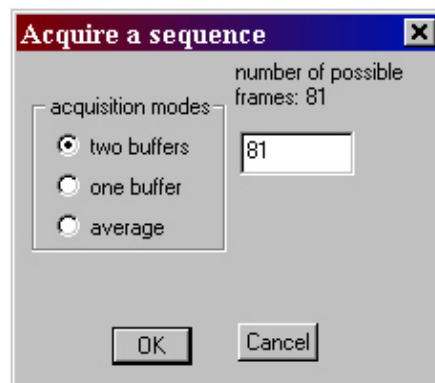
Toolbar: 

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 alternative an average image of the sequence can be build. Details for the settings are displayed in the following table.



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.

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

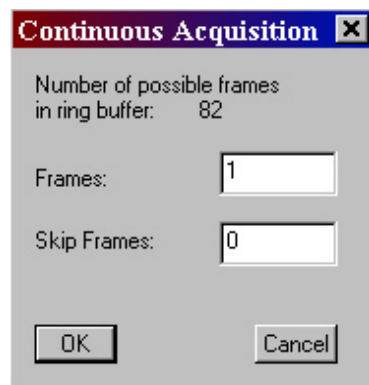
Toolbar: 
Keys: CTRL+SHIFT+S

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.

Shortcuts

Toolbar: 
Keys: CTRL+SHIFT+C

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

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

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.

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.

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.

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.

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.

End of Acquisition

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

Shortcut: **Esc**

Camera

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 RID 512-400 A:

(TIMING 0)	200 ms	(Shortest available frame time of the detector.)
(TIMING 1)	400 ms	
(TIMING 2)	800 ms	
(TIMING 3)	1600 ms	
(TIMING 4)	3200 ms	
(TIMING 5)	6400 ms	
(TIMING 6)	12800 ms	
(TIMING 7)	25600 ms	

Camera Mode

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

- The free running mode means that the camera sends out continuously frames according to the selected frame time. This is the default mode.
- The external triggered mode means that the camera 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 camera 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.

View menu

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.

Toolbar

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

For details see the chapter Toolbar

Acquisition Bar

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

For details see the chapter Acquisition Toolbar

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.

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.

Toolbar: 

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.

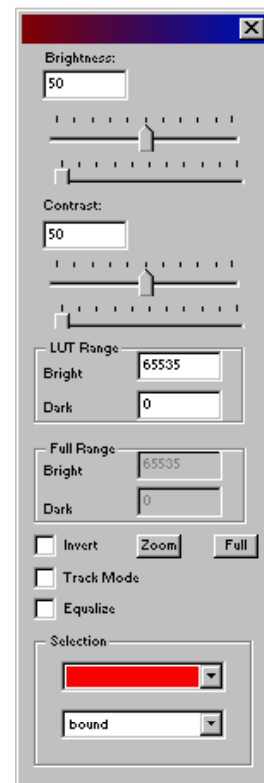
View Control Box

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.



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.

Window menu

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.

Cascade

This command arranges multiple opened windows in an overlapped fashion.

Tile Horizontal

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

Tile Vertical

This command arranges multiple opened windows side by side.

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.

1, 2, ... command

HIS 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.

Options menu

Acquisition

This command restarts the sensor initialization and allows to select the initialization type (automatic or manual). After chose of this menu entry a dialog comes up, which asks to initialize the sensor automatically or by manually The selection of automatic initialization leads to a dialog asking for the mode in which the frame grabber will run. The choose "yes" runs the frame grabber in Interrupt Mode and "No" runs the frame grabber in Polling Mode. The "Cancel" button starts the HIS without initialization of the camera. If the "Open always" radio button is checked the HIS 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 HSL can't free all resources in one process that were allocated by another process.

For the automatically initialization the HIS scans the PCI bus for plugged in frame grabbers and tries to detect connected cameras. If it recognizes more than one camera 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 HIS 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 expected parameters to initialize the sensors connected to this channels:

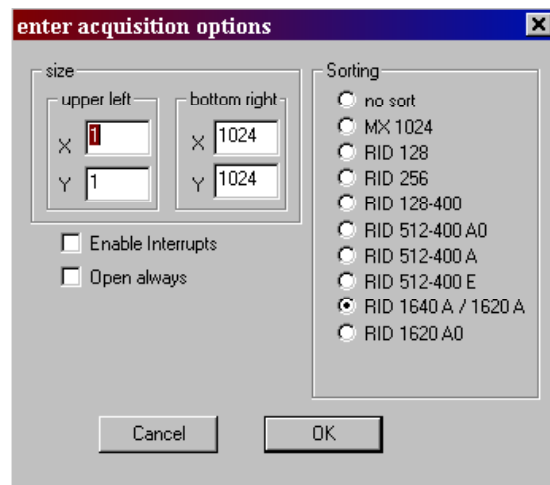
Size of the detector

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

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
- RID 512-400
- RID 1640 A / 1620 A



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 camera is running in polling mode. It is recommended to use the interrupted mode for an frame synchronization.


Open always

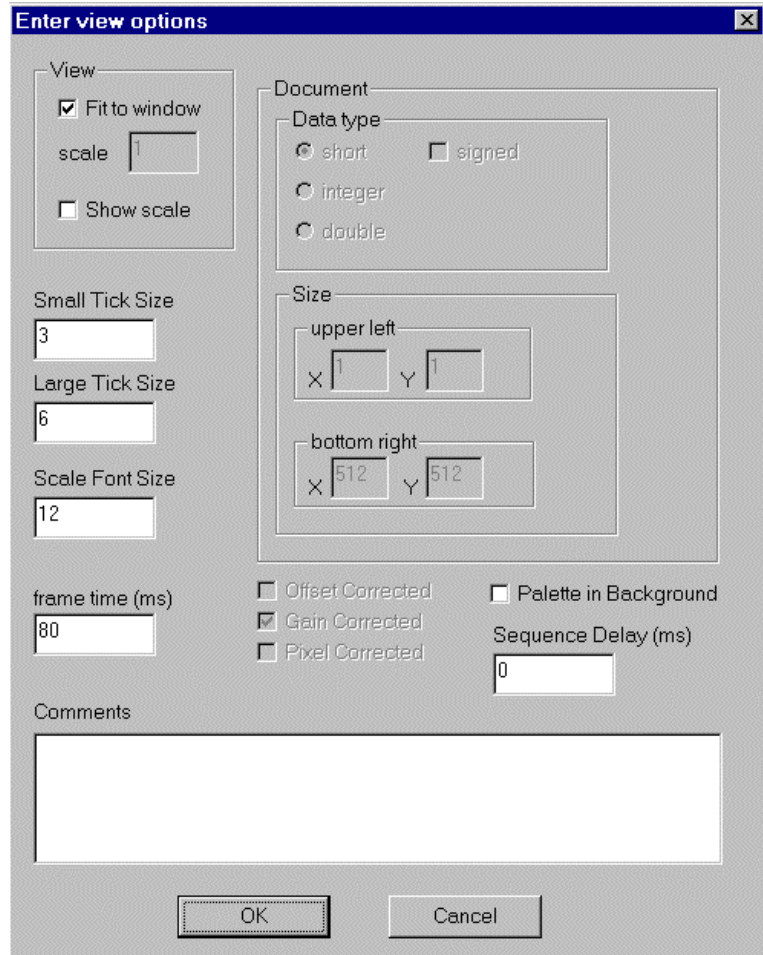
This option opens the requested communication channel regardless if it has been already captured by another process running on the system. It isn't recommended to use this option except for debugging because the HSL can't free all resources in one process that were allocated by another process.

If you initialized more than one sensor the HIS asks now a default camera to that all the further instructions correspond.

View

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

Toolbar: 



View - Fit to window
View - Show Scaling

Defines the scale of the image window.
To be switched ON for scaling the edges of the image

- Small Tick 2 (default)
- Large Tick 4 (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.

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:



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

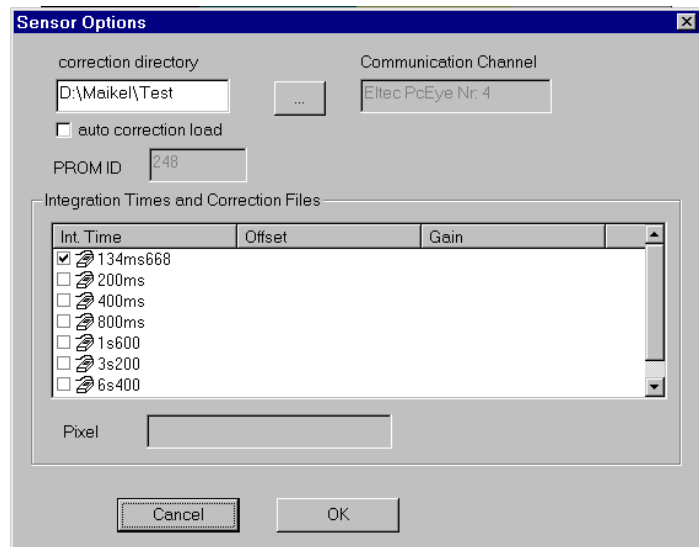
In the correction directory the correction files are stored by the Save correction files command.

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.

The PROM ID is the identifier of the actual used camera PROM. This information is important to get support at hardware problems concerning the camera 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.



Help

Contents

Displays the contents of HIS help file.

Index

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

Using Help

This command displays instructions about using Help.

About

Displays the version number and copyright notice of HIS.

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 HIS.

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. HIS 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 HIS 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

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 HIS.

The Toolbar can be hide or displayed by the command **Toolbar** in the View menu (ALT, V, T).

Click	Application
-------	-------------



Starts the continuous Acquisition.

Shortcut: CTR+SHIFT+C



Opens the Sequence Dialog



Starts a Single Shot



Opens the Player Dialog.

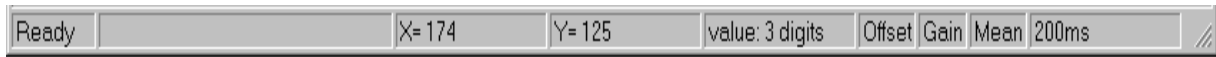


Opens the View Dialog



Opens the Active Sensor Dialog

Status Bar



The status bar is displayed at the bottom of the HIS 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 8:	Marker for Pixel correction ON/OFF.
Box 9:	Selected frame time.

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

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.

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.

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

Minimize command (application Control menu)

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

Shortcut

Mouse: Click the minimize icon  on the title bar.

Keys: ALT+F9

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.

Next Window command (document Control menu)

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

Shortcut

Keys: CTRL+F6

Previous Window command (document Control menu)

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

Shortcut

Keys: SHIFT+CTRL+F6

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

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.

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.

Shortcut

Keys: CTRL+ESC

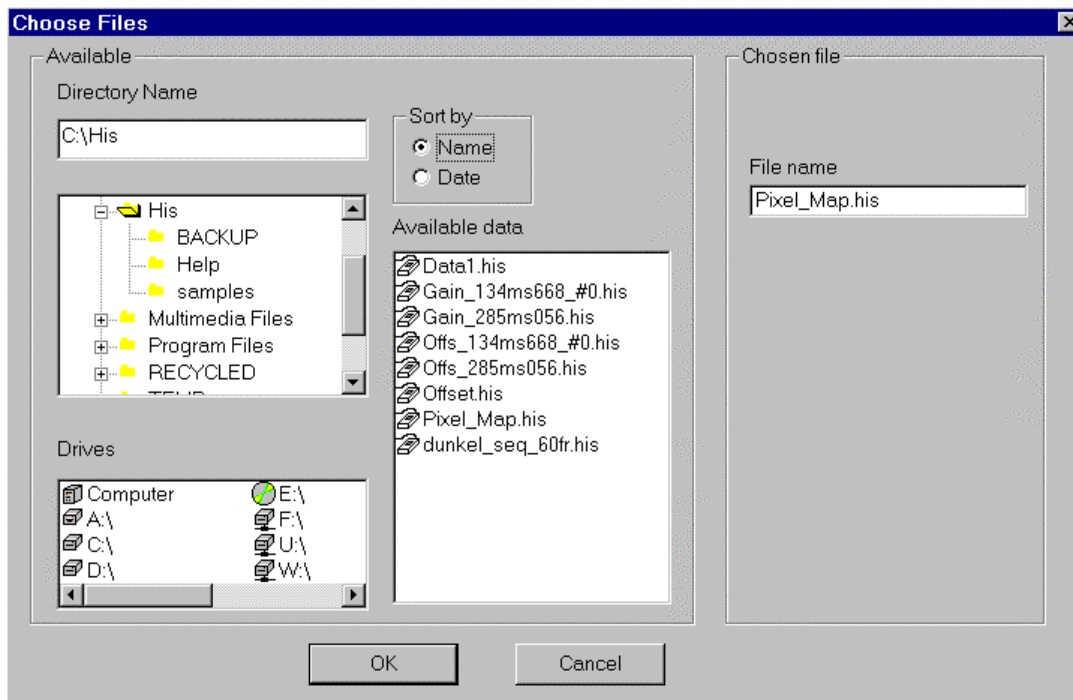
Dialog Box Options of the Switch command

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.

Standard Dialogs

File Selection Dialog

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



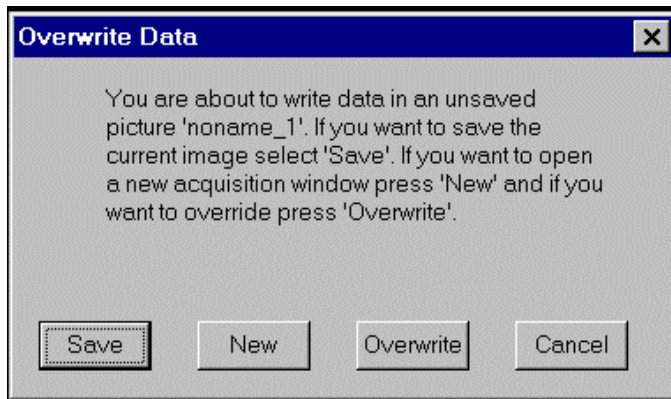
The available group box shows you all available drives, directories and files. The currently loaded data's are listed below the drive "computer".

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

Simply drag the selected file from the "available data" box to the "correction file name" edit box and drop it there. The name of the image will be inserted. The process will be executed by pressing the "Ok" button or the "enter" key

Overwrite data dialog

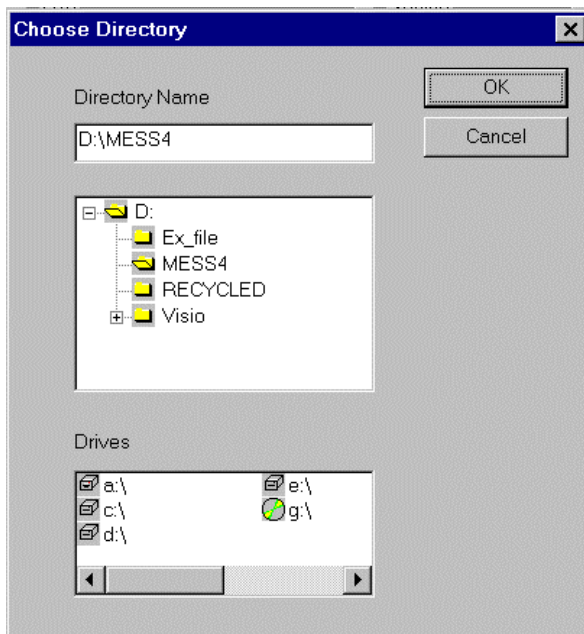
This dialog appears if new data has to be acquired into an unsaved window.



- Save** Saves the current loaded data and overwrites the window buffer with the new ones.
- New** Creates a new window. A dialog to select the suitable document type appears .
- Overwrite** Overwrites the existing data without any saving.
- Cancel** Aborts the acquisition.

Choose Directory Dialog

This dialog appears if a directory has to be selected.

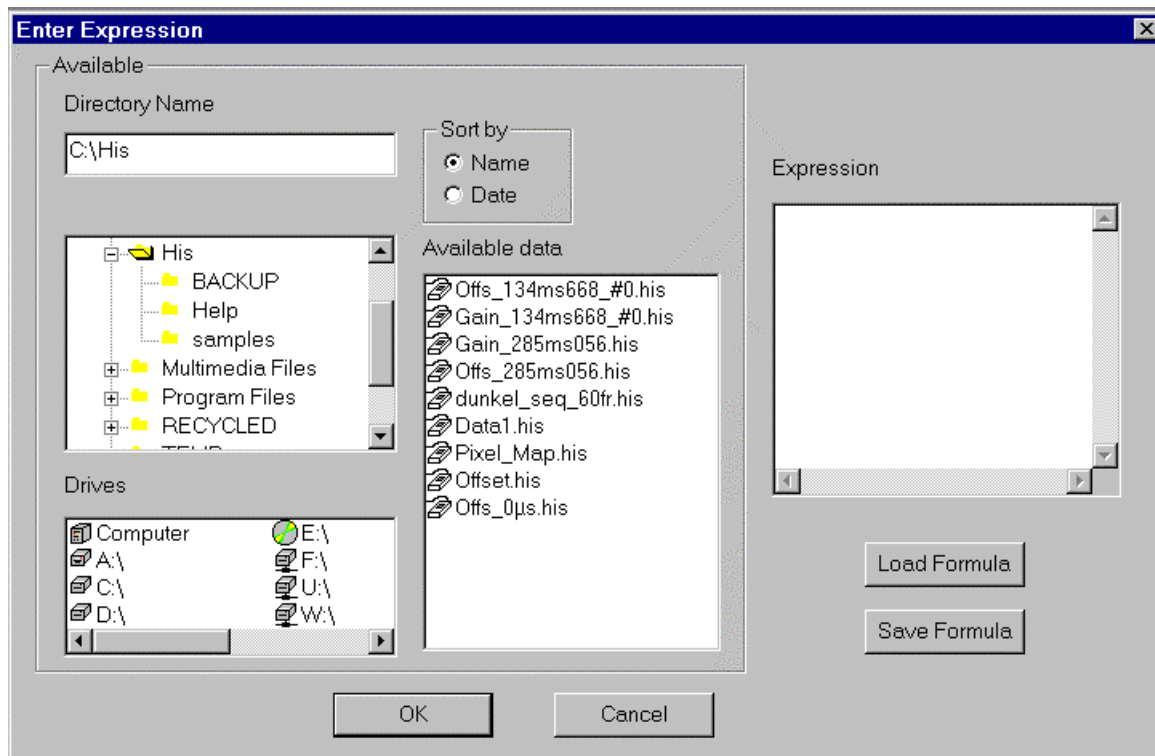


- directory name** This edit line specifies the selected directory. It's connected to a directory tree and all current available folders are listed.
- drives** All current available drives are listed here. If a drive is selected the **directory name** and its directory tree is actualized.

Application

Mathematical Expressions

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



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 linebreaks can be set by pressing the "enter" key while holding the "Control" key.

Note: The loaded images have to be connected with the drive "computer".

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

+ 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 = 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".

The expression *Dest = 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.

The expression $data1 = data1 + 8000$ is allowed.

(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

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 than 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 than one frame, the result is a sequence of frames containing the summed columns of frames.

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

Image Correction

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 perform:

1. Select the desired frame time (see Timings menu).
2. Switch off the X-ray source so that the detector only transfers its "dark image".
3. Start the Get Offset Image. / Start All Offset Images.
4. 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.
5. Control the new acquired image by using the Options/View command and/or Brightness, Contrast or LUT range.
6. 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. In automatic routines, the detector allows the full S/N ratio of more than 80 dB using updated Offset corrections.

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.

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 perform:

1. Select the desired frame time (see Timings menu).
2. Create an Offset correction image, if none is available.
3. Switch on the X-ray source and control in the continuous mode the brightness of the acquired image. The detector's acquired intensity should be between 70-80 % of its maximum signal or in the region of further interests.
4. Start the 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.

Use and generation of the Pixel Correction

The Pixel Correction of images is recommended to eliminate the influence of defective pixels of the detector in acquired images.

To get an Pixel correction file the following steps have to perform:

1. Select the desired frame time (see Timings menu).
2. Link correction files.
3. Switch on the X-ray source and control in the continuous mode 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 windows should show an homogenous corrected image. Intensity deviations are a sign of not full working pixels.
7. Change the x-ray source 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 BADPIX.HIS.

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

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 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.

Acquisition Control Modes

Three different acquisition control modes are available. The **Free Running** mode is the default mode which means that the camera sends out continuously frames according to the selected frame time. The **External Trigger** mode means that the camera 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 camera 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 fastest free running mode of the camera and 5 seconds in steps of 1 μ s can be selected and these pulses are send 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 **Camera**. 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 PcEye3 and Squirrel as to the camera (see: External Trigger).

The trigger modes are recommended if an 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 artifacts which are not correctable. The trigger mode realize 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 camera or with the frame grabber board
2. Power on the camera
3. Startup HIS
4. Select an initialization of the camera (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 camera 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 camera 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 camera is set to the integration time 400ms and the trigger pulse is send every 200ms, the camera nevertheless runs with 400ms. But if the pulse appear every 450ms the camera 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.

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 HIS 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 HIS commands.

Ready

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

Details for the Hardware

Readout schema

Free Running

The first timing of each camera is the fastest readout time. This means that the detector needs a minimum of 134.668 ms (RID512-400 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 RID512-400 AL1 is 400 ms, this means the first 134.668 ms is a readout of the camera and the following 265.332 ms is a delay and the detector is only integrating radiation.

Note:

If a pulsed x-ray source is used it is recommended to exposure 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 artifacts 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.

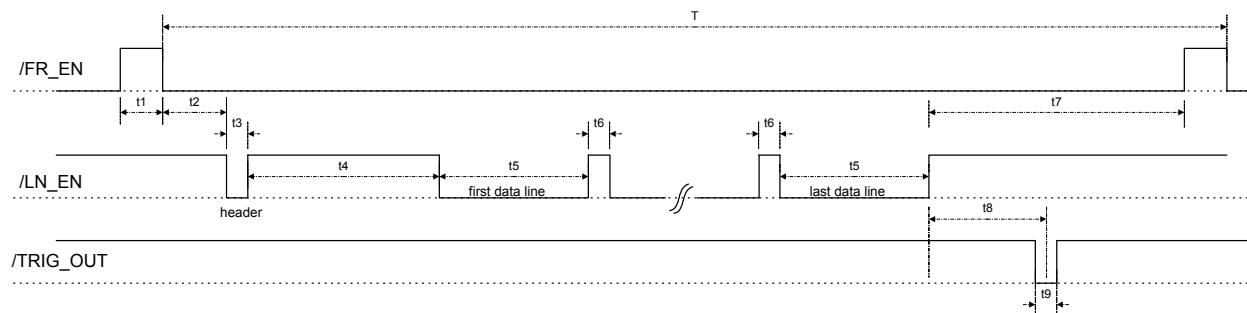


Figure 1: General timing diagram in continuous mode

External Trigger

Triggering the camera is the attempt to synchronise the camera to other devices e.g. x-ray sources having their specific schemes of radiating x-ray pulses. The current mode is triggering the camera on a frame by frame base which means that the camera 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 camera can be triggered by a new pulse.

In order to trigger the camera a 20µs wide low active trigger pulse (TTL-signal) 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 camera device or connected to the sub-click located on the rear side of the frame grabber (/TRIG_IN converted to RS422 signal as /FR_SYNC). Trigger pulses are accepted from both sources. Prior to this the camera has to be set per command into the external trigger mode. The waveform of the trigger pulse as shown in fig. 2 describes the triggering mode on a frame by frame base. The period of the trigger waveform determines the integration time.

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.

Internal Trigger

The internal trigger mode works similar to the external trigger mode and is also triggering the camera on a frame by frame base which means that the camera 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 camera 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 camera and 5 seconds in steps of 1µs. The frame grabber sends the RS422 signal /FR_SYNC over the HIIB to the camera. 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.

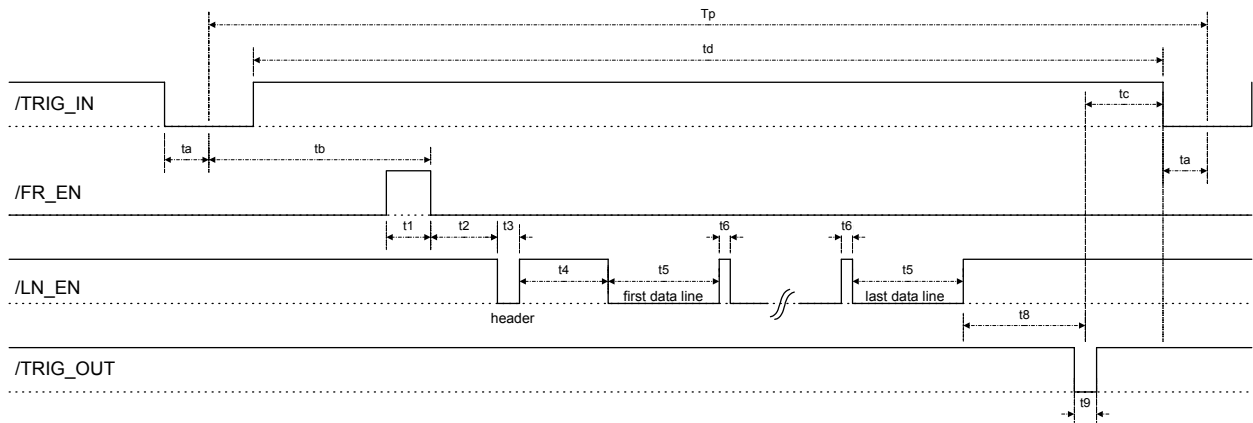
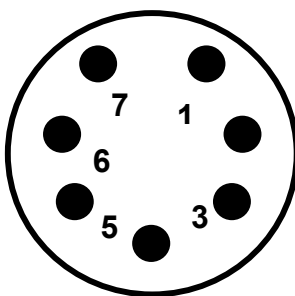


Figure 2: General timing diagram in triggered mode

External trigger inputs/outputs

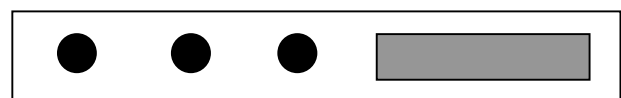
a.) Detector I / O



Front view

- 1 – /TRIG IN
- 2 – /TRIG OUT (Begin of frame)
- 3 – NC
- 4 – GND
- 5 – NC
- 6 – 5 V
- 7 – NC

b.) Frame Grabber I / O



Front view

- 1 – /TRIG IN
- 2 – /FR_SYNC (Squirrel)
- 2 – /LN_ENB (PcEye3)
- 3 – /FR_EN (Begin of frame)

The /TRIG IN ,/TRIG OUT, /LN_EN and /FR_EN signals are low active /TTL signals.

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 camera 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
524388	byte	524288 *n	+100 byte		

Table 1: 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	// self explanatory
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 2: 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

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

rows = BRY-ULY+1
columns = BRX-ULX+1

Description of Hardware Header

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

BYTE HeaderID	identification number of this header type (10)
USHORT PROM ID	identification number of the PROM set of the camera, used to identify the camera type
BYTE NumberOfEmptyLines	Number of lines of empty cycles.
USHORT Rows	number of sensor rows - 1
USHORT Columns	number of sensor columns - 1
USHORT ZoomULY	zooming upper row - 1
USHORT ZoomBRY	zooming bottom row - 1
USHORT ZoomULX	zooming left column - 1
USHORT ZoomBRX	zooming right column - 1
USHORT LSBNumberRows	least significant bytes of empty rows
BYTE MSBNumberRows	most significant bytes of empty rows
BYTE RowType	describes the different row driving schemes
USHORT NumberEmptyCycles	number of empty cycles of 15.625 Nanoseconds
BYTE Sorting	required sorting (see sorting overview)
BYTE DataType	type of data (0 for USHORT)
BYTE CameraMode	fixed mode (0), sync mode (1) with fixed frame regime
BYTE Timing	selected integration time (preliminary)
USHORT Gain	only used for the RISL camera family otherwise (0x7FFF)
USHORT Offset	only used for the RISL camera family otherwise (0x7FFF)
BYTE SensorBias	$10 \text{ V} * \text{SensorBias} / 255$
BYTE Reserved	for later use

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

The sensor frame time Tint can be derived by the following formula:

$$\text{Tint} = (\text{LSBNumberRows} + \text{MSBNumberRows} * 65536) * \text{RowTime} + \text{CycleTime} * \text{NumberOfEmptyCycles}$$

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 HSL function Acquisition_GetHwHeader.

Row types

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

At gate on time the sensor TFT is switched on and the charge transfer from the pixel diode to the read out electronics takes place. At the gate off time the TFT is switched off. All incoming signals are collected by the diode until the next switch on in the next frame. The rising and falling time of the switch pulse is 15 microseconds. The analog digital conversion is started at the "Start ADC" time and is finished at "end ADC" time. The row time describes the time needed to drive the row including electronic reset time and sampling times of the analog electronic by the ADC.

There are two row driving schemes in general. One drives the row at once without any interruption. This gives the best dynamic performance. The other switches the TFT in the first row interval and the ADC takes place in the second row time. That means that analog switching and analog to digital conversion of two neighboring rows is done at once. That causes a faster frame regime but lowers the dynamic performance of the whole systems.

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	gate on time	gate off time	start ADC	end ADC	two row sch.	row time
0	2	180	0	256	yes	300
1	2	180	0	256	yes	300
2	2	180	0	256	yes	312.5
3	2	240	0	256	yes	390.625
4	20	350	500	756	no	781.25
5	2	180	0	256	yes	310
6	2	240	0	256	yes	387.5
7	20	160	256	772	no	781.25
8	820	1150	1300	1556	no	2500

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.

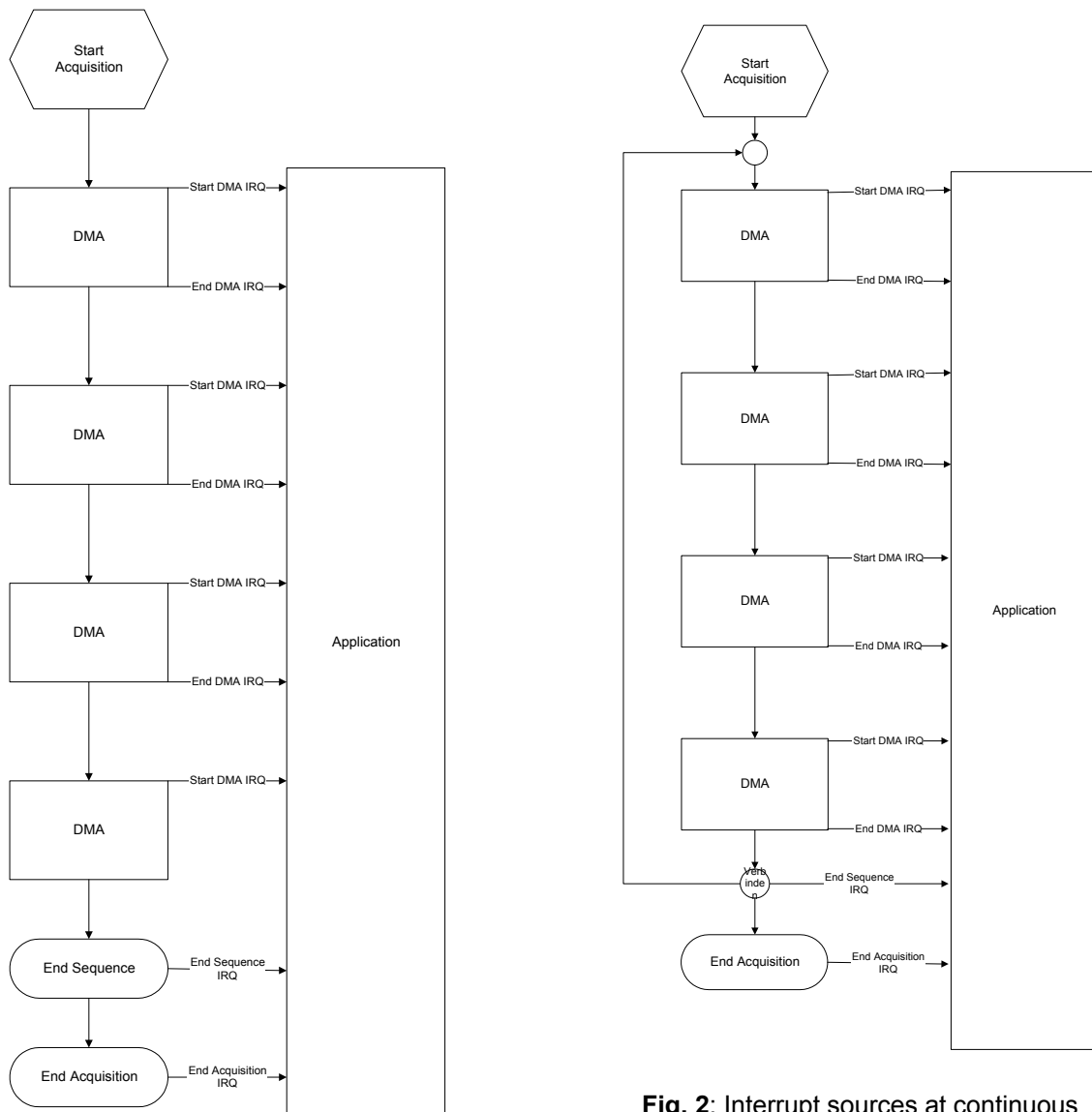


Fig. 2: Interrupt sources at continuous acquisition mode.

Fig 1: Interrupt sources at sequence acquisition mode (4 frames).

After the application started the acquisition one interrupt is caused by the begin of DMA (start DMA interrupt). If the whole frame data are 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 the acquisition is canceled an end of acquisition interrupt is fired (see also Acquisition_Init, Option/Acquisition). If interrupts are enabled the end of frame, end of sequence and the end of acquisition interrupts are used.

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 an disturbance on the data line or at the camera itself the acquisition will be synchronized only in the interrupt mode.

Sorting

Sorting schemes overview

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

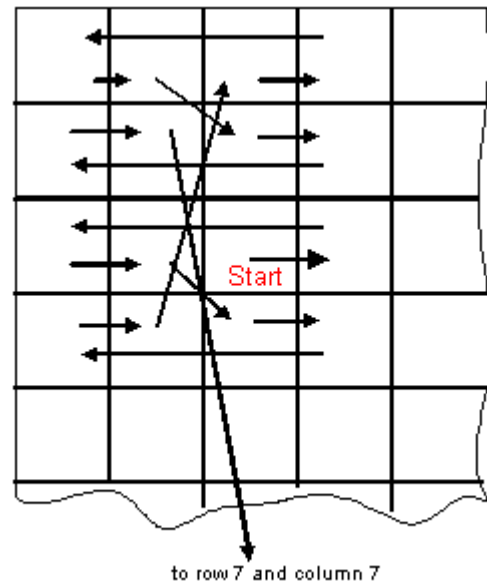
RID128	1
RID256	2
RID128-400	3
RID1024-100	4
RID512-400 A0	5
RID512-400 A1/A2	6
RID512-400 E	7
RID 1640 A	8
RID 1620 A	8

Sorting RTM128

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)
...	...



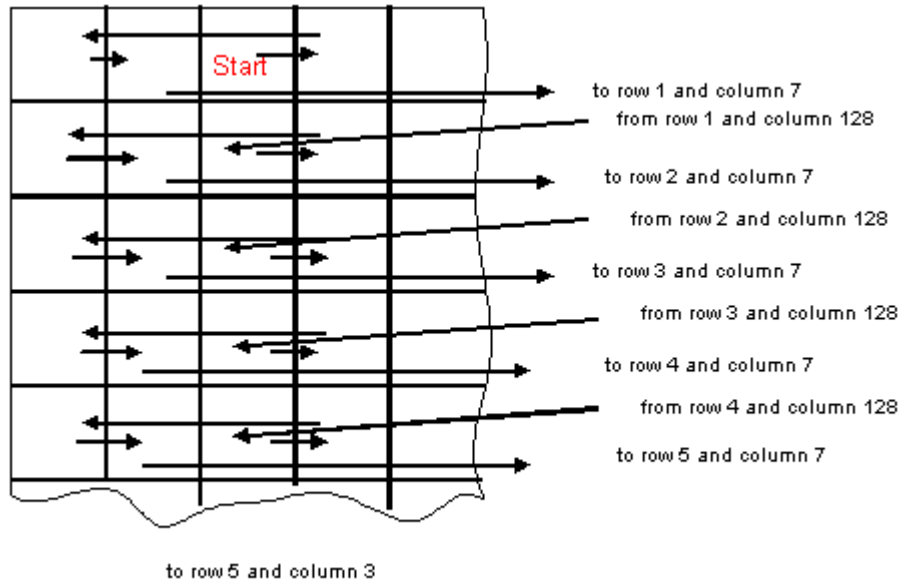
Sorting RID256

An RID 256 sensor is read out by two analog to digital converters simultaneously. The first reads out the first half and the other transfers the second half of columns. This results in the following data stream.

data stream no.	sensor pixel (row, column)
1	(1,1)
2	(1,129)
3	(1,2)
4	(1,130)
...	...
255	(1,128)
256	(1,256)
257	(2,1)
258	(2,129)
...	...

Sorting RID128-400

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.



This results in the following list:

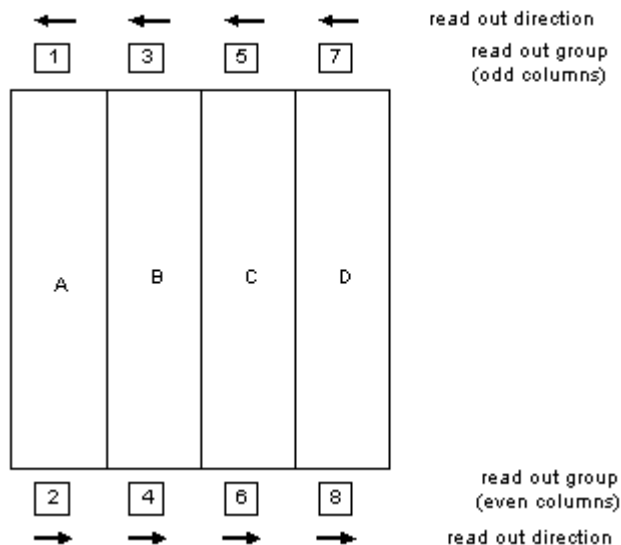
data stream no.	sensor pixel (row, column)
1	(1,3)
2	(1,4)
3	(1,1)
4	(1,2)
5	(1,7)
6	(1,8)
7	(1,5)
8	(1,6)
9	(1,11)
...	...
125	(1,127)
126	(1,128)
127	(1,125)
128	(1,126)
129	(2,3)
130	(2,4)
131	(2,1)
132	(2,2)
...	...

Sorting RID1024-100

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.



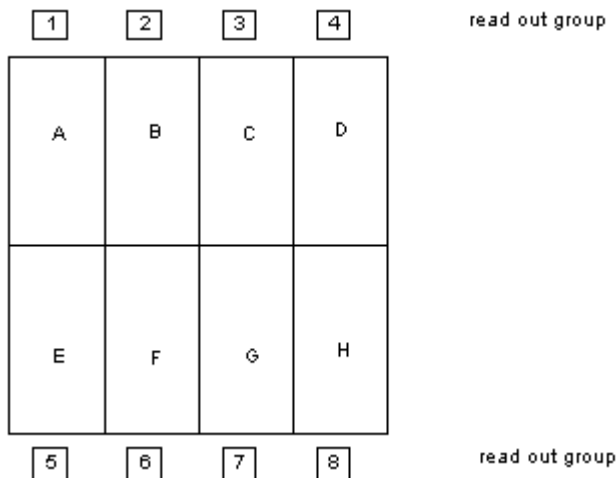
The data stream is described in the following list:

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
16	(1,772)	8
17	(1,251)	1
18	(1,6)	2
19	(1,507)	3
20	(1,262)	4
...
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
...

Sorting RID512-400 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 list:

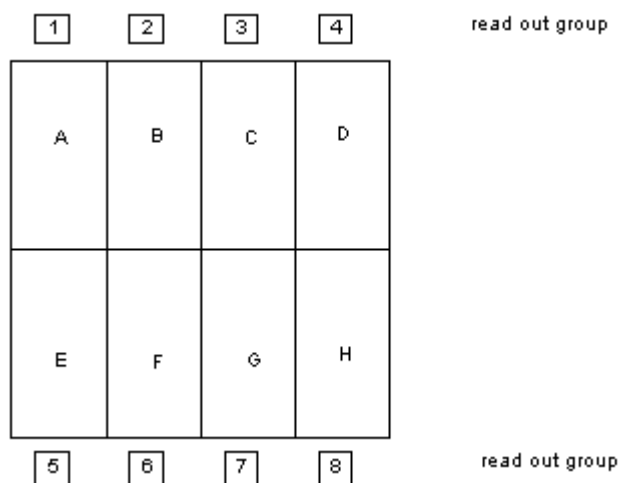


data stream no.	sensor pixel (row, column)	ROG no.
1	(1,1)	1
2	(1,129)	2
3	(1,257)	3
4	(1,385)	4
5	(257,1)	5
6	(257,129)	6
7	(257,257)	7
8	(257,385)	8
9	(1,2)	1
10	(1,130)	2
11	(1,258)	3
12	(1,386)	4
13	(257,2)	5
14	(257,130)	6
15	(257,258)	7
...
262131	(256,383)	3
262132	(256,511)	4
262133	(512,127)	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

Sorting RID512-400 A1/A2

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 list:

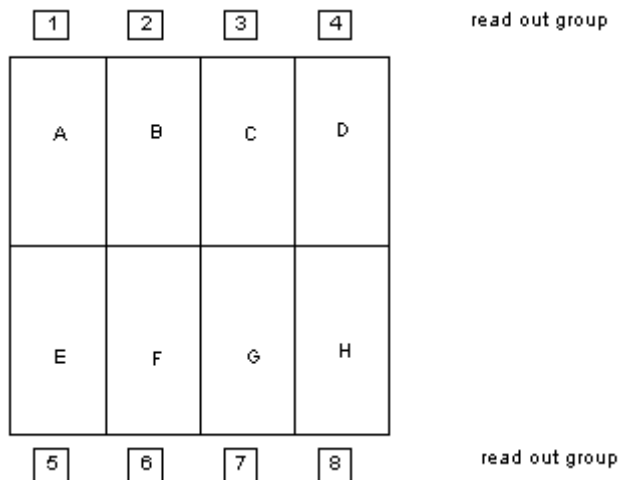


data stream no.	sensor pixel (row, column)	ROG no.
1	(1,1)	1
2	(1,129)	2
3	(1,257)	3
4	(1,385)	4
5	(512,128)	5
6	(512,256)	6
7	(512,384)	7
8	(512,512)	8
9	(1,2)	1
10	(1,130)	2
11	(1,258)	3
12	(1,386)	4
13	(512,127)	5
14	(512,255)	6
...
262128	(257,387)	8
262129	(256,127)	1
262130	(256,255)	2
262131	(256,383)	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

Sorting RID512-400 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:



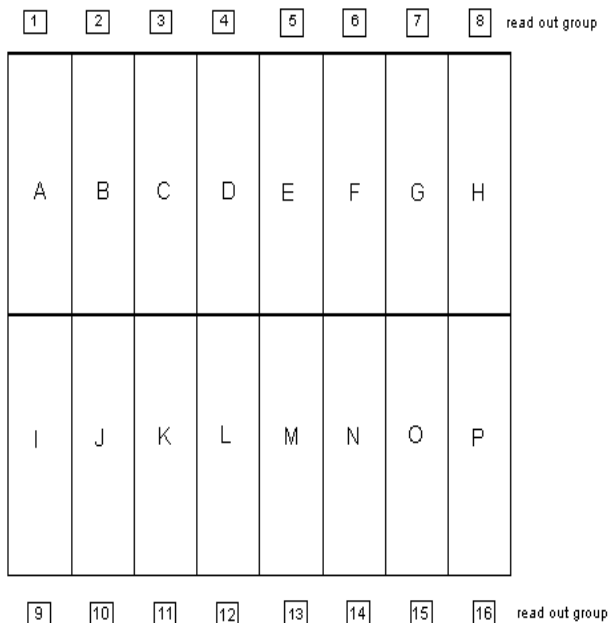
data stream no.	sensor pixel (row, column)	ROG no.
1	(512,128)	5
2	(512,256)	6
3	(512,384)	7
4	(512,512)	8
5	(1,1)	1
6	(1,129)	2
7	(1,257)	3
8	(1,385)	4
9	(512,127)	5
10	(512,255)	6
11	(512,283)	7
12	(512,511)	8
13	(1,2)	1
14	(1,130)	2
15	(1,258)	3
16	(1,386)	4
17	(512,126)	5
...
1022	(1,384)	3
1023	(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

Sorting RID1640 A // RID 1620 A

The figure demonstrates the read out scheme of the sensor. The sensors RID 1620 and RID 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 RID 1640 and 256 for the RID 1620. 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 list displays the data stream for RID 1640:



data stream no.	sensor pixel (row, column)	RID1620	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

Heimann Software Library

Heimann Software Library Overview

The Heimann Software Library provides the basic functionality to acquire and correct RID data. To get an impression of the tasks you have to implement to acquire images by help of the HSL look at HSL demo. This demo is also a part of the HIS setup.

Acquisition_Descriptor	Basic structure for data acquisition used by HSL.
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 HSL if image drawing is ready.
Acquisition_SetCameraMode	Allows setting of camera 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 HSL.
Acquisition_GetIntTimes	Detects the available times in microseconds.
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 camera'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.

HSL Functions

Acquisition Descriptor

AcquisitionDesc defines a data structure that is used by all functions of HSL. It contains all required parameters for the acquisition. Access to the data fields is only possible via the HSL 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.

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 HSL demonstration.

UINT WINAPI Acquisition_EnumSensors(

```
    UINT *pdwNumSensors,
    BOOL bEnableIRQ,
    BOOL bInitAlways
);
```

pdwNumSensors

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 HSL 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.

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 HSL 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.

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 PcEye3 or Squirrel frame grabber.
HIS_BOARD_TYPE_RS232	3	The communication to the sensor is provided by a serial interface (RS232).

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. If the sensor is connected to the serial interface on port COM2 set *dwChannelNr* equal to 2.

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 camera. *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
HID_SORT_QUAD_TILE (0x5)	RID512-400 A0
HIS_SORT_QUAD_TILE_INVERSE (0x6)	RID512-400 A1/A2
HIS_SORT_QUAD_TILE_INVERSE_SCRAMBLE (0x7)	RID 512-400 E
HIS_SORT_OCT_TILE_INVERSE (0x8)	RID 1640 A and RID 1620 A

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

bSelfInit

If a camera with an unknown PROM-ID is connected to the frame grabber, the HSL normally comes up with a dialog to enter the camera 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 HSL 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.

Acquisition_GetCommChannel

This function returns the type of the communication device that is used to transfer data from the camera 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	PcEye3 or Squirrel Frame Grabber
HIS_BOARD_TYPE_RS232	0x3	serial interface based on RS232

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.

Acquisition_DefineDestBuffers

This function defines the pointers of the destination buffer for Acquisition_Acquire_Image and Acquisition_Acquire_Continuous. 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.

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

dwErrorMsg

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

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 HSL 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 HSL 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.

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>

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_Continuous**. 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 $((\text{number of wrong pixels}) * 10 + 1) * \text{sizeof}(\text{int})$. 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 HSL function *Acquisition_CreatePixelMap*.

Return values

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

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`.

Acquisition_AbortCurrentFrame

This routine aborts the transfer of the current frame from the camera to the frame grabber. The camera will be reset then and a new frame transfer will be started immediately.

```
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`.

Acquisition_SetCameraMode

This function sets the acquisition mode of the camera. Currently eight fixed frame times of the camera 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`.

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*.

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*.

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*.

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.

Acquisition_DoOffsetCorrection

This function performs an offset 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_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.

Acquisition_DoGainCorrection

This function performs a gain 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_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.

Acquisition_DoOffsetGainCorrection

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_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.

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.

Acquisition_IsAcquiringData

This function tests if HSL 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.

Acquisition_GetErrorCode

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

Acquisition_Close

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

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

Parameters:

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`.

Acquisition_CloseAll

This function closes shuts down all frame grabbers and serial interfaces currently allocated by the HSL. 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.

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 HSL 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.

Acquisition_GetReady

Retrieves the draw status of the HSL.

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

hAcqDesc

Pointer to acquisition descriptor structure.

Return values

The function returns the flag set by Acquisition_SetReady.

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 HSL.

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 camera 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 camera. This number is only important if the camera operates objectionably and you need any support from EG&G.

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 camera.
HIS_SYNCMODE_INTERNAL_TIMER	The synchronization signal can be generated by the internal timer of the frame grabber (see Acquisition_SetTimerSync)
HIS_SYNCMODE_SOFT_TRIGGER	The synchronization signal can be generated by software (see Acquisition_SetFrameSync).

dwHwAccess

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

If you have to use this parameter (that depends from your contract with EG&G) please contact EG&G 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.

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]);
}
```

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`.

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.

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.

Acquisition_SetFrameSyncMode

This function sets the synchronization mode of the frame grabber.

```
UINT WINAPI Acquisition_SetFrameSyncMode(
    HACQDESC hAcqDesc,
    DWORD dwMode
);
```

hAcqDesc

Pointer to acquisition descriptor structure.

dwMode

This parameter can have the following values:

HIS_SYNCMODE_FREE_RUNNING	The sensor is operating in free running mode.
HIS_SYNCMODE_INTERNAL_TIMER	The synchronization signal can be generated by the internal timer of the frame grabber (see Acquisition_SetTimerSync)
HIS_SYNCMODE_EXTERNAL_TRIGGER	The frame grabber enables external trigger mode. The signal must be supplied at the trigger input of the frame grabber or the camera.
HIS_SYNCMODE_SOFT_TRIGGER	The synchronization signal can be generated by software (see Acquisition_SetFrameSync).

Return values

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

Acquisition_SetFrameSync

This function supplies a synchronization signal to the camera 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.

AcquisitionSetTimerSync

This function supplies a synchronization signal to the camera 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 ms. 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`.

CHwHeaderInfo

Structure that is used to retrieve the contents of camera'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;
```

Description of structure entries (All entries are 32 bit integers.)

DWORD	<i>dwPROMID;</i>	identifies the camera's PROM set
DWORD	<i>dwHeaderID;</i>	identifies the used header version (version zero)
BOOL	<i>bAddRow;</i>	indicates if an additional row is transferred
BOOL	<i>bPwrSave;</i>	indicates if camera is in power safe mode
DWORD	<i>dwNrRows;</i>	number of sensor rows
DWORD	<i>dwNrColumns;</i>	number of sensor columns
DWORD	<i>dwZoomULRow;</i>	row of the upper left edge of zoom region
DWORD	<i>dwZoomULColumn;</i>	column of the upper left edge of zoom region
DWORD	<i>dwZoomBRRow;</i>	row of bottom right edge of zoom region
DWORD	<i>dwZoomBRColumn;</i>	column of bottom right edge of zoom region

DWORD	<i>dwFrmNrRows;</i>	Number of rows that are used to synthesize the frame scheme of the camera. 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	<i>dwFrmRowType;</i>	see Row Types
DWORD	<i>dwFrmFillRowIntervals;</i>	Intervals of 10 nanoseconds to synthesize a frame (see description of hardware header
DWORD	<i>dwNrOfFillingRows</i>	Number of rows of the above mentioned row time.
DWORD	<i>dwDataType;</i>	normally zero (unsigned 16 bit integer)
DWORD	<i>dwDataSorting;</i>	see sorting
DWORD	<i>dwTiming;</i>	selected integration time (preliminary)
DWORD	<i>dwAcqMode;</i>	fixed mode (0), sync mode (1) with fixed frame regime
DWORD	<i>dwGain;</i>	only used for the RISL camera family otherwise (0x7FFF)
DWORD	<i>dwOffset;</i>	only used for the RISL camera family otherwise (0x7FFF)
BOOL	<i>bSyncMode</i>	1 if the camera operates in triggered mode else 0.
DWORD	<i>dwBias;</i>	10 V * SensorBias / 255
DWORD	<i>DwLeakRows</i>	Number of rows without driven gates

Acquisition_GetHwHeaderInfo

This function returns the contents of the camera'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 camera'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.

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.

Demo application

The HSL 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 "HSL_Demo.dsp". If you want to use your own development environment insert HIS_Acq.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 camera.

Initialization:

The HSL 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 (*blnitAlways*) is only recommended in debug versions of your applications. If this parameter is TRUE the HSL 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 HSL of the address of this buffer and its numbers of rows and columns.

```

//route acquisition buffer to HSL
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 HSL 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 HSL 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 HSL 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 HSL again about the new buffer address.

```
//route acquisition buffer to HSL
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 HSL 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 HSL calls until your application got a respond from the end of acquisition callback to the call of Acquisition_Abort.

Changing frame time

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

```

//select another frame time
printf("\nSet camera 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.

Acquiring Offset Images

The sensor needs an offset correction to work properly. For this purpose the HSL 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 HSL 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 HSL 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 HSL 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 HSL 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 HSL. 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 HSL.

```
FlushConsoleInputBuffer(hInput);
do
{
```

```

        ReadConsoleInput(hInput, &ir, 1, &dwRead);
    } while(ir.EventType!=KEY_EVENT);
    Acquisition_Abort(hAcqDesc);
    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 (Number_Of_Defects*9+1)*sizeof(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 HSL:

```
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 HSL

The following code fragment closes the HSL 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;
}
```

HSL error codes

The following table lists all error codes of the HSL 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_NOCAMERA	Got a time out for acquisition. May be no camera 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 camera 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

Frame Grabber error codes

The following table lists error codes of the HSL/FrameGrabber and gives a short description of them. The error codes are similar for booth types of frame grabber (PcEye3 and Squirrel).

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 camera 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_HWNOTOPENED	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_NOPCEYEFOUND	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_HWNOTINIT	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_WRONGCAMERASELECT	An invalid camera input number (0..3) is given.
-12	EL_E_ACQWINDOWTOOBIG	The acquisition window is too big for the camera 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 camera is not connected or powered down.
-24	EL_E_INVALIDPARAMETER	Invalid function parameter supplied.
-25	EL_E_INVALIDPOINTER	Invalid pointer supplied.
-64	EL_E_WRONGCAMERA	unknown camera
-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.