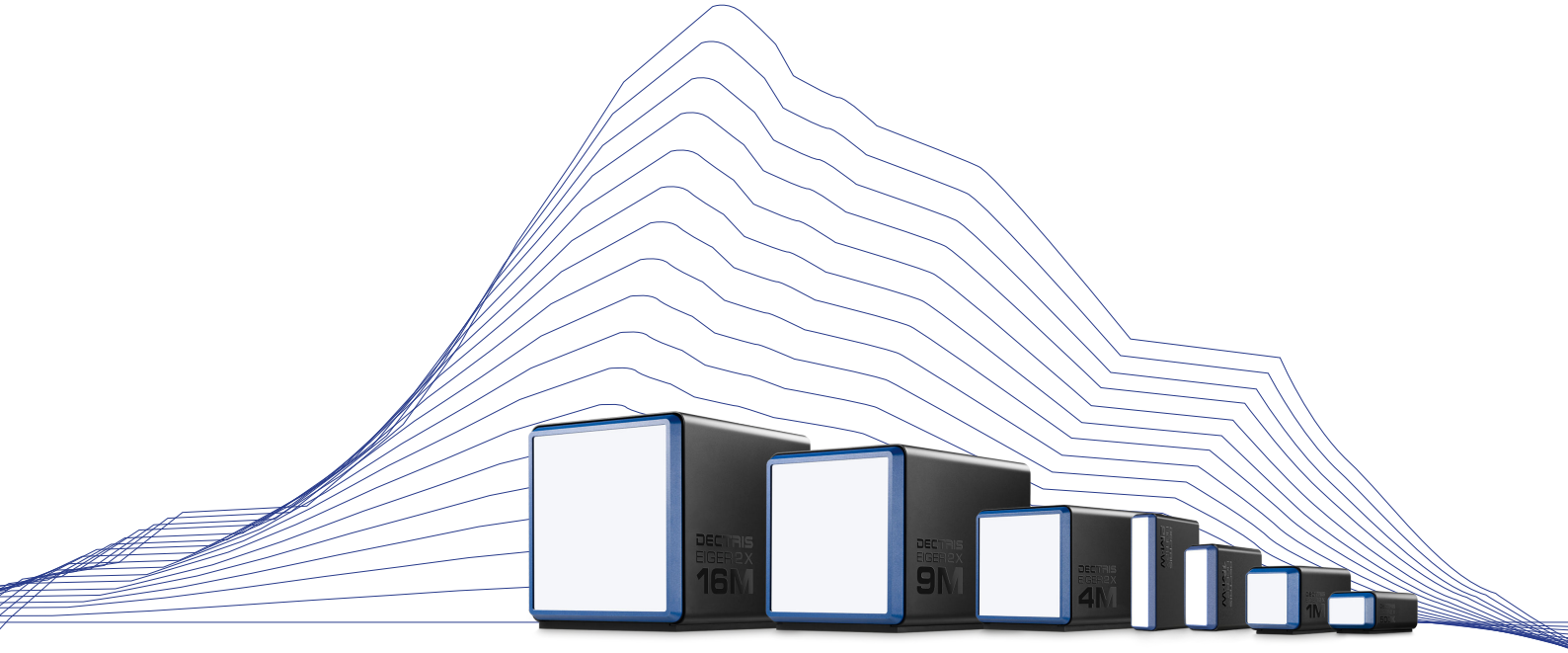
DECTRIS



User Manual

DECTRIS EIGER®2

Document Version 2.0.0

DECTRIS Ltd.

5405 Baden-Daettwil, Switzerland

www.dectris.com

TABLE OF CONTENTS

1. GENERAL INFORMATION

1.1. Contact and Support

Address	**DECTRIS Ltd** Taefernweg 1 5405 Baden-Daettwil Switzerland	**DECTRIS USA Inc.** 1500 Walnut Street, Suite 1630 Philadelphia, PA 19102 USA	**DECTRIS Japan K.K.** Nakagawa building 801 3-37 Higashi-Nobusue Himeji-shi Hyogo 670-0965 Japan
Phone	+41 56 500 21 02	+1 267 924 5179	+81 79 280 9585
Website	www.dectris.com		
Email	support@dectris.com		

1.2. Explanation of Symbols

DANGER
Danger blocks are used to indicate immediate danger or risk to personnel or equipment.

WARNING
Warning blocks are used to indicate danger or risk to personnel or equipment.

CAUTION
Caution blocks are used to indicate danger or risk to equipment.

NOTICE
Information blocks are used to highlight important information.

1.3. Warranty Information

Should your detector require warranty service, contact DECTRIS® for further information. Before shipping the system back, please contact DECTRIS® to receive the necessary transport and shipping information. Make sure that the original packaging is used when returning the system.

CAUTION
Do not ship the system back before you receive the necessary transport and shipping information.

CAUTION
Opening the detector or the power supply housing without explicit instructions from DECTRIS® will void the warranty.

1.4. Disclaimer

DECTRIS® has carefully compiled the contents of this manual according to the current state of knowledge. Damage and warranty claims arising from missing or incorrect data are excluded.

DECTRIS® bears no responsibility or liability for damage of any kind, also for indirect or consequential damage resulting from the use of this system.

DECTRIS® is the sole owner of all user rights related to the contents of the manual (in particular information, images or materials), unless otherwise indicated. Without the written permission of DECTRIS® it is prohibited to integrate the protected contents in this publication into other programs or other websites or to use them by any other means.

DECTRIS® reserves the right, at its own discretion and without liability or prior notice, to modify and/or discontinue this publication in whole or in part at any time, and is not obliged to update the contents of the manual.

2. SAFETY INSTRUCTIONS

CAUTION
Please read these safety instructions before operating the detector system.

- Before turning the power supply on, check the supply voltage against the label on the power supply. Using an improper main voltage will destroy the power supply and damage the detector.
- Power down the detector system before connecting or disconnecting any cable.
- Make sure the cables are connected and properly secured.
- Avoid pressure or tension on the cables.
- Operating the detector outside of the specified conditions could damage the system.
- The detector is not specified to withstand direct beam at a synchrotron. Such exposure will damage the exposed pixels.
- Place the protective cover on the detector when it is not in use to prevent the detector from accidental damage.
- Do not install additional software or change the operating system of the Detector Control Unit.
- Do not touch the front window of the detector or the sensor modules.

3. SYSTEM DESCRIPTION

3.1. Components

The DECTRIS EIGER®2 detector system consists of the following components:

- EIGER2 detector
- Power supply unit (PSU)[1]
- Detector control unit (DCU)
- Thermal stabilization unit[2]
- Accessories
- Documentation

3.2. Hybrid Photon Counting

3.2.1. Basic Functionality

DECTRIS® X-ray detectors provide direct detection of X-rays with optimized solid-state sensors and CMOS readout ASICs in hybrid pixel technology. Well-proven standard technologies are employed independently for both the sensor and the CMOS readout ASIC. The EIGER2 hybrid pixel detector is composed of a sensor, which is a two-dimensional array of photodiodes or photoconductors processed in a high-resistivity semiconductor, connected to an array of readout channels designed in advanced CMOS technology. The X-ray detectors operate in single-photon counting mode and provide outstanding data quality. They feature high dynamic range, zero dark signal and zero readout noise and hence achieve optimal signal-to-noise ratio at short readout time and high frame rates. Large-area detectors with dedicated active areas are built of multiple identical modules using a modular system concept.

Key advantages

- Direct detection of X-rays
- Single-photon counting
- Excellent signal-to-noise ratio and high dynamic range (zero dark signal, zero readout noise)
- Two energy discriminating thresholds
- Continuous readout with zero dead time and high frame rates
- Shutterless operation
- Modular system enabling multi-module detectors with a large active area
- Full factory calibration in the specified energy range

3.2.2. Continuous Readout

A hallmark feature of EIGER2 is its continuous readout that enables high frame rates at high duty cycles with only 100 ns deadtime between frames. Every pixel of the ASIC features two digital counters per energy discriminating threshold. After acquisition of a frame, the pixels switch from counting in one digital counter to the other. While one counter is being read out, data acquisition continues in the other counter.

[1]Some systems might be delivered without an external power supply unit. Please consult the Technical Specifications for more information.
[2]Also referred to as Chiller. Some systems might be delivered without a thermal stabilization unit. Please consult the Technical Specifications for more information.

3.2.3. Auto-Summation

Auto-summation is a further benefit of continuous readout in EIGER2 detectors. While a single frame is limited to the 16 or 8 bit of the digital counter, auto-summation extends the data depth up to 32 bit, or more than 4.2 billion counts per pixel, depending on the number of summed frames in an image.

At short exposure times and high frame rates, all counts are captured in the digital counter of a pixel and directly read out as an image. If long exposure times are requested, frames are still acquired at high rates on the pixel level, effectively avoiding any overflows. The detector system sums the frames to images on the fly, extending the bit depth of the data by the number of summed frames.

3.2.4. Instant Retrigger

EIGER2 X-ray detectors feature the DECTRIS INSTANT RETRIGGER® technology for improved high-rate counting performance. The Instant Retrigger capability results in non-paralyzable counting and allows for enhanced count-rate correction.

DECTRIS INSTANT RETRIGGER® with adjustable dead time is a photon counting imaging method that results in non-paralyzable counting and achieves an improved high-rate counting performance. In a conventional single-photon counting X-ray detector, the charge pulses generated by impinging photons are counted by digital circuits. Simultaneously generated pulses can pile up and result in photon counts being lost. At high photon fluxes, pulse pile-up significantly affects the observed count rate and can lead to complete paralyzation of the counting circuit. In the first generations of EIGER and PILATUS photon counting detectors, even though a count rate correction was applied to compensate for the counting loss at high count rates, the maximum usable count rate was still limited by counter paralyzation. In EIGER2 detectors, the Instant Retrigger reevaluates the pulse signal after a predetermined dead-time interval after each count and is able to retrigger the counting circuit should a pulse pile-up occur. The dead time interval is adjustable and is equivalent to the width of one single photon pulse. This results in non-paralyzable counting and allows for enhanced count-rate correction so as to achieve improved data quality at high count rates.

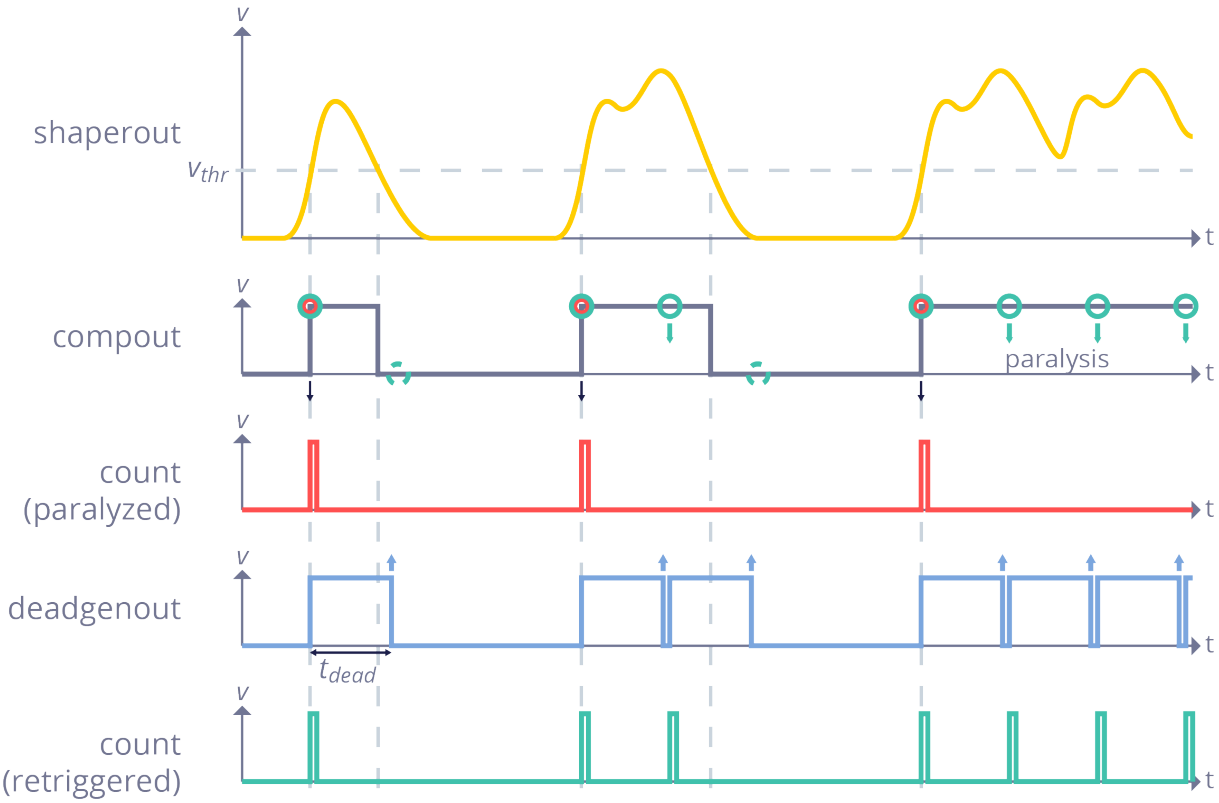


Figure 1. Signal waveforms illustrating the DECTRIS INSTANT RETRIGGER® technology

The Instant Retrigger principle is illustrated in Figure 1. The first diagram shows the signal pulses generated by impinging photons and the effective discriminator threshold level for single photon counting. The pulse signal includes a series of one single pulse, a pile-up of two pulses and a pile-up of multiple pulses. The second diagram shows the corresponding digital discriminator output signal that triggers the counting circuit. The third diagram shows the corresponding counts being registered by a conventional single photon counting X-ray detector, clearly illustrating that counts are lost in the case of pulse pile-up and that this can lead to paralyzation. The fourth diagram shows the respective dead-time generator output signal provided by a single photon counting X-ray detector with Instant Retrigger. Here, a predefined dead time interval is started whenever a count has been registered. The fifth diagram shows the corresponding counts being registered, including potential retriggering of the counting circuit after the dead-time interval after each count. This clearly illustrates that pulses are counted more accurately in the case of pile-up and that counting is non-paralyzable.

3.2.5. 8-bit Readout

NOTICE
The 8-bit readout mode is only available with DECTRIS EIGER®2 X and XE detectors.

The EIGER2 ASIC allows the option to read out only the first 8 bits of every counter instead of the full 16 bits. This enables a higher maximum frame rate at the cost of a lower dynamic range. The 8-bit mode is automatically enabled as soon as the frame rate exceeds the maximum frame rate in 16-bit mode, as indicated in the Technical Specifications document.

When using the 8-bit mode, it is important not to exceed the maximum flux per pixel, as there is no overflow protection and the counter will roll over if 255 counts are exceeded. This means the maximum flux per pixel is 255 × frame rate in 8-bit mode.

3.3. Software

3.3.1. SIMPLON API

The EIGER2 detector system is controlled via the SIMPLON API, which relies on a http/REST interface. Section 6.3 covers a selection of essential API parameters. The full API details are provided as a separate document: ”SIMPLON API Reference”.

The detector’s web interface (Section 5) gives access to fundamental settings and status parameters and also enables a first test to see if the detector system has been set up properly after installation and startup as described in Section 4.

3.3.2. HDF5 File Format

EIGER2 detectors deliver FileWriter images in the HDF5 format. This is a high-performance hierarchical data format designed to organize and store large datasets within a simple structure. More details on FileWriter and the other data interfaces can be found in Section 6.1.

Third Party HDF5 Libraries

The EIGER2 HDF5 data can be directly read with programs using the HDF5 library.

By default, the EIGER2 data is compressed using the BSLZ4 algorithm[3]. In order to decompress the data, the HDF5 plug-in filter[4] can be used. See https://github.com/nexusformat/HDF5-External-Filter-Plugins.

By setting the environment variable HDF5_PLUGIN_PATH to the path where the compiled plug-in filter can be found, the HDF5 library will decompress the data compressed with LZ4 by itself.

If you want to use proprietary software like Matlab, IDL or similar, make sure that the HDF5 library version used by this software is at least v1.8.11 in order for the plug-in mechanism to work.

[3]See: https://github.com/lz4/lz4 and https://github.com/kiyo-masui/bitshuffle
[4]See: https://support.hdfgroup.org/releases/hdf5/documentation/rfc/HDF5DynamicallyLoadedFilters.pdf

4. QUICK START GUIDE

4.1. Accessing the Detector Control Unit

The EIGER2 detector is controlled via the network interface of the detector control unit. Hence, the IP network address of the detector control unit has to be known to be able to connect to the API. Depending on the network structure, there are several ways of determining the IP network address, which are described below.

- See the Technical Specifications document for the default network port configuration of your detector control unit.
- The default network port configuration may be changed through the detector's web interface (see Section 5.2).

4.1.1. Using the Service Port

You can access the Detector Control Unit (DCU) by using the network address of the Service port. Plug the network cable into the Service port of the DCU which is pre-configured to the fixed IP address 169.254.254.1. See the Technical Specifications document for further details on the network port configurations of the DCU.

If you use, for example, a laptop to access the DCU directly for the initial configuration, you can use the following network settings on the laptop:

Table 1. User network settings

IP Address	169.254.254.100
Subnet Mask	255.255.0.0
Default Gateway	not required

With those settings, the web interface can be accessed through a web browser at the IP address 169.254.254.1. See Section 5 for more details on using the web interface.

4.1.2. Using DHCP

If there is a DHCP server available on the network, plug the network cable into a port of the Detector Control Unit pre-configured for DHCP. See the Technical Specifications document for the default network port configuration of the Detector Control Unit.

If you have access to the web interface via the static IP of the service port, you can determine the DHCP IP of the detector via the Network tab (see Section 5.2). Alternatively, you can connect a keyboard and a monitor to the Detector Control Unit and follow these steps:

- Once the Detector Control Unit is fully booted, a command line login prompt will appear.
- Type `recovery` and press return.
- If a password prompt appears, leave it empty and press return. The login name was most likely misspelled. Restart from the first point.
- Type `i` to select option `(i) show ip addresses`.
- The IP address will be displayed on the screen.
- If no IP is displayed, make sure that the DCU is properly connected to your network.

4.2. Detector Startup Procedure

NOTICE
Instructions on properly mounting and preparing the detector system can be found in the Technical Specifications document. Before operating the detector, read the complete documentation.

- Turn on the nitrogen or dry air flow at least 30 min before turning on the detector.
- Turn on the Thermal Stabilization Unit and set the operation temperature as specified in the Technical Specifications document. Please read the Thermal Stabilization Unit manual, as some models must be powered and additionally activated in order to operate properly.
- Connect all the required cables for power, data, and trigger signals (if applicable) to the detector and the detector control unit.
- Turn on the detector.
- Turn on the Detector Control Unit. Please allow 6 minutes for the BIOS test procedures and startup of software to complete.
- Access the web interface via the IP address of the detector control unit or use the SIMPLON API to initialize and control the detector (see the API Reference documentation).
- For detectors with a CdTe sensor, it is necessary to wait up to 30 minutes after initializing the detector in order to get the best data quality.

5. WEB INTERFACE

5.1. Overview

The EIGER2 web interface (Figure 2) provides simple access to basic functions and settings of the detector system for installation, debugging, and system updates. For productive operation of the detector, please refer to the SIMPLON API documentation.

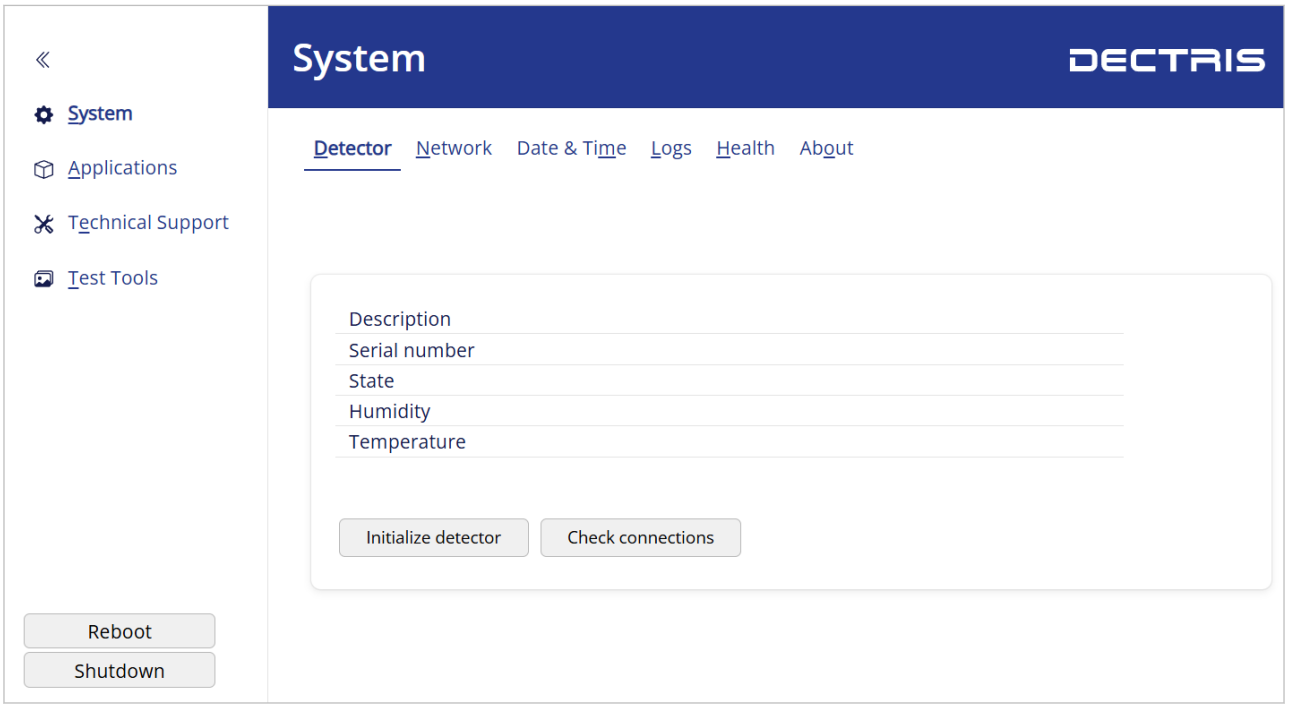


Figure 2. EIGER2 home page.

Table 2 summarizes the functions available through the left panel of the web system interface.

Table 2. Menu items of the EIGER2 web interface.

Menu Item	Content
System	View the system information and access the Detector Control Unit settings (see Section 5.2)
Applications	Manage the detector and calibration applications. This tab provides functions to start and stop the detector software, change software versions and upgrade the detector software to a new version.
Technical Support	Create a bug report that can be downloaded and sent to support@dectris.com. Note: The bug report is not sent automatically.

Menu Item	Content
Test Tools	Acquire images with these tools for **basic** detector configuration, data acquisition and image previewing via the web interface. Available from software version 2026.1. See Section 6.2.1 for more details. **NOTICE** These are out-of-the-box tools to help acquire your first images, with a limited scope. They are not a full performance solution for data acquisition, image processing and analysis. See Section 6.3 for more in-depth functionality. **CAUTION** Actions on this interface may affect active acquisitions or impact other users concurrently operating the detector.

5.2. System Settings and Administration

To access the EIGER2 system settings, click on the corresponding menu item on the homepage. The System tab shows the detector information and allows to configure the network settings of the Detector Control Unit.

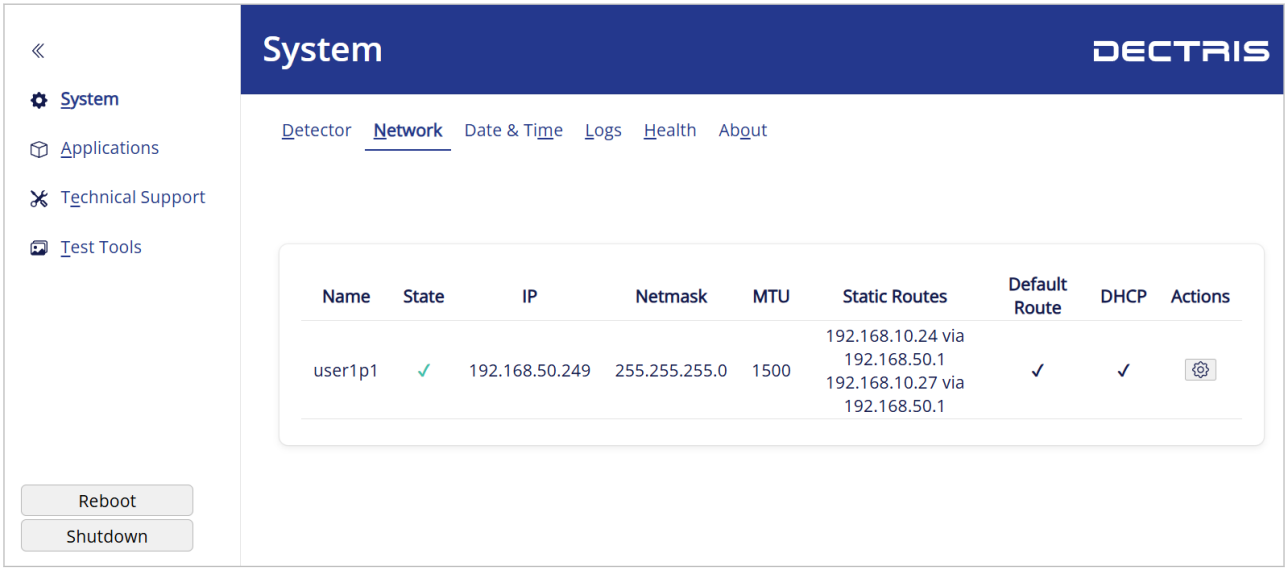


Figure 3. Screenshot of the System settings showing the network configuration page.

Table 3. Description of the System menu Items

Tab	Description
Detector	See the detector status and initialize the detector. The detector status can only be read out when the detector is initialized and the detector application is running. This tab also provides a way to quickly check the quality of the detector connection using the ”Check connections” button. Typical values are above 0.15 mW for the TX power and above 0.3 mW for the RX power.

Tab	Description
Network	Configure the user accessible network interfaces.
Date & Time	Configure date and time on the Detector Control Unit. Provides the possibility to add an NTP server.
Logs	Select and view detector logs, including error, warning and debugging information.
Health	View system health information, such as temperature and humidity history, as well as board errors.
About	Display system details.

6. GENERAL USAGE OF THE DETECTOR SYSTEM

6.1. Detector Control and Output

The EIGER2 detector system is controlled through the SIMPLON API, an interface to the detector that is based on the http protocol. The API documentation supplied with the system describes this interface in detail and allows for easy integration of the detector control into the user instrumentation independent of the operating system or programming language being used. Please refer to the SIMPLON API Reference document for details.

The data recorded by the detector can be accessed in different ways. Images can be stored by the FileWriter on the Detector Control Unit as HDF5 files, which include metadata in a NeXus-compatible format. Buffered files have to be regularly fetched and subsequently deleted on the Detector Control Unit as the buffer space is limited[5]. Support for reading out multiple thresholds has been introduced in FileWriter2. See the SIMPLON API Reference for further details.

Data can also be fetched through the Stream interface, which has a low latency and offers utmost flexibility. The metadata is transferred as part of the header. Streamed data is not buffered and will be lost if not fetched or incompletely fetched. With the introduction of Stream2 it is possible to retrieve the images from multiple thresholds simultaneously at full frame rates.

Lastly, the Monitor interface provides images as raw tiff files with minimal metadata. This is a low performance and low bandwidth interface, which allows to retrieve the latest recorded image for monitoring purposes during acquisitions. The Monitor offers a relatively small buffer. In case of buffer overflow, the oldest images will be overwritten. The Monitor also allows to retrieve the images from multiple thresholds.

6.2. Recording an Image or an Image Series

CAUTION
Data might have to be fetched concurrently to a running image series. The lifespan of the data on the Detector Control Unit is dependent on the configuration of your system as well as the interface used for collecting data. Data not fetched within this lifespan is permanently lost.

To record images or image series, the following steps need to be performed through the SIMPLON API (see the API Reference for details).

1. Make sure the detector has been set up according to the steps described in Section 4.
2. Initialize the detector if this has not yet been done. The detector does not need to be reinitialized unless the detector entered an error status.
3. Set the detector parameters for data acquisition and specify and configure the desired output interface (FileWriter and/or Stream interface). A list of essential configuration parameters can be found in Section 6.3.1.
4. Arm the detector.
5. Record the image or image series.
   • Send trigger(s) to record the image or image series as previously configured.
   • Fetch data through the data interface(s).

[5]Buffer space varies dependent on the configuration of your system, buffer overflow will cause loss of data. See API Reference for further details.

6. Disarm the detector (to ensure files are finalized and closed).
7. Repeat from step 3 for further data acquisition with different settings, or from step 4 for identical settings.

6.2.1. Basic Image Acquisition via the Web Interface

The EIGER2 web interface has built-in tools to acquire and display your first images in a simple way (available from software version 2026.1).

After accessing the web interface (Section 5.1), click the **Test Tools** on the left panel.

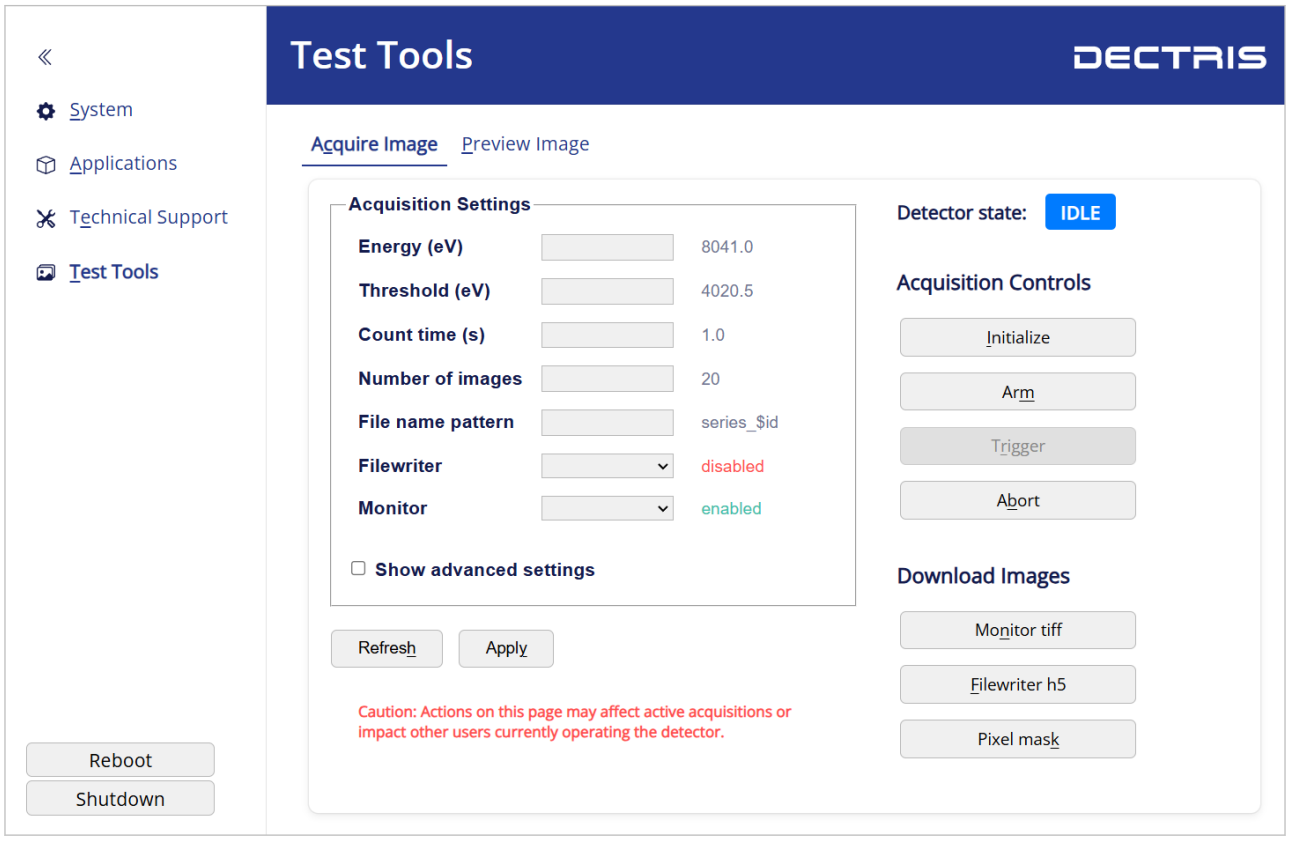


Figure 4. Basic detector configuration via the Web Interface

The **Acquire Image** tab allows you to set the most essential configuration parameters for basic image acquisition (see Section 6.3.1 for details). Make sure to enable the FileWriter and/or the Monitor interfaces to record and download the images.

CAUTION
The *Advanced Settings* menu should only be used in very specific cases and applications. Changing the advanced parameters will impact the data quality and should only be done with full understanding of the outcome. Make sure to read the SIMPLON API Reference manual for details.

When changing parameters, first click the Apply button and then Arm to save the settings on the detector.

The Trigger button will start the image series via the internal software trigger (see INTS mode in Section 8.2). The detector will disarm automatically once the series is finished.

The Abort button can be used to immediately stop an ongoing acquisition, but note that the last running image will be lost when aborted.

The images can be downloaded in TIFF format via the Monitor interface, or HDF5 through the FileWriter.

The Pixel Mask can also be downloaded as a TIFF file. See Section 9 for more details.

NOTICE
These are out-of-the-box tools to help acquire your first images, with a limited scope. They are not a full performance solution for data acquisition, image processing and analysis. See Section 6.3 for more in-depth functionality.

CAUTION
Actions on this interface may affect active acquisitions or impact other users concurrently operating the detector.

6.2.2. Displaying an Image via the Web Interface

TIFF images can be viewed via the Web Interface under the **Preview Image** tab of the Test Tools (available from software version 2026.1).

This feature allows to load images from the Monitor Interface or upload TIFF images from your local storage. Monitor images can also be played automatically.

The tool provides basic display settings and allows to save the current image as PNG or TIFF.

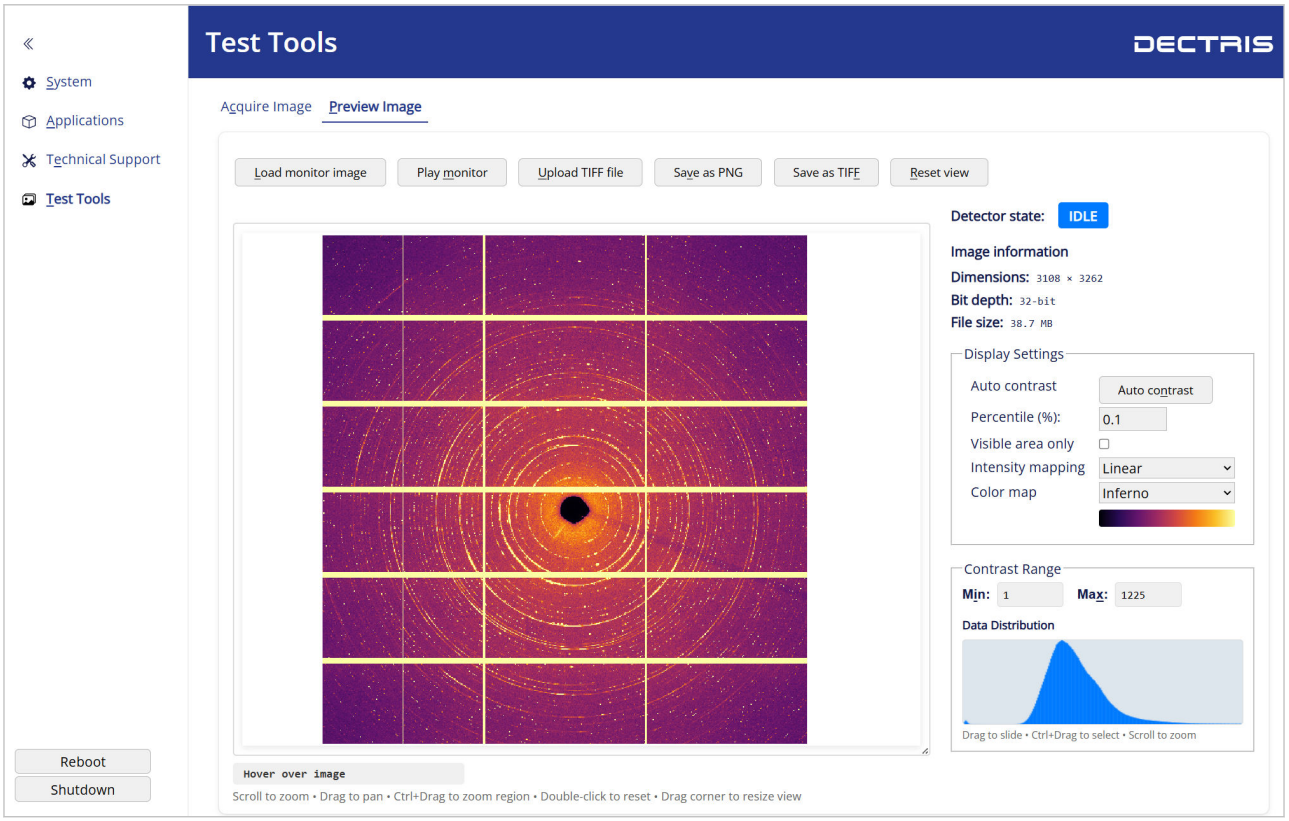


Figure 5. Viewing an example TIFF image via the Web Interface

6.3. Control of the Detector from a Specific Environment

Integrating the detector into a specific environment requires understanding of the necessary detector functions. The API reference will list all possible commands and features, but it does not give an explanation of the required functionality. Section 6.3.1 and Section 6.3.2 cover a selection of essential and situational parameters respectively.

6.3.1. Main Configuration Parameters

The parameters described in this section allow control of the detector and data acquisition. Data will be acquired, however further configuration of the interface for data retrieval might be necessary depending on your set up. For starting a data acquisition, only the following parameters need to be adjusted:

- detector | config | count_time
- detector | config | frame_time
- detector | config | photon_energy

The parameter count_time refers to the actual time the detector counts photons, while frame_time is the interval between acquisitions of subsequent frames (i.e. period) and defines the frame rate.

The detector configuration parameter photon_energy has to be set to the X-ray energy used for the experiment. There is a convenience function for setting the photon_energy, called element.

- detector | config | element

The parameter *element* accepts the chemical symbols for elements, e.g., ”Cu”, ”Mo”, as argument and sets photon_energy to the $K_{\alpha 1}$ emission line for that element.

- detector | config | ntrigger
- detector | config | nimages

The ntrigger parameter configures the number of expected triggers for one acquisition series. Setting values greater than 1 for ntrigger allows several trigger commands or external trigger pulses per arm/disarm sequence. This mode allows recording several series of nimages with the same parameters.

The parameter nimages configures the number of images that will be acquired for one trigger. In internally and externally triggered series (”ints” and ”exts”, respectively), the detector considers a trigger as the start of a series of n images. For example a single image is considered as a series of images containing 1 image. Once the detector has been armed, a series can be started by issuing a trigger command or triggering the detector using an electronic pulse on the external trigger input. In ”inte” and ”exte” trigger modes the parameter nimages should be set to 1, as every trigger will always trigger a single image. To switch between the trigger modes (see Section 8) one can use the configuration parameter trigger_mode.

- detector | config | trigger_mode

NOTICE
Please note that the acquired data can be retrieved via different interfaces. For details, please see the API Reference.

With the filewriter mode set to ”enabled”, the acquired data is written to HDF5 files. The FileWriter has the following important configuration parameters:

- filewriter | config | mode
- filewriter | config | name_pattern
- filewriter | config | nimages_per_file

- filewriter | config | compression_enabled

The filewriter name_pattern sets the name for the HDF5 files. When using the pattern "$id", it will be replaced with a sequence identification number and can be used to differentiate between subsequent series. The sequence identification number is reset after initializing the detector. The parameter nimages_per_file sets the number of images stored per data file. A value of 1000 (default) means that for every 1000th image, a data file is created. If for example, 1800 images are expected to be recorded, the arm, trigger, disarm sequence means that a master file is created in the data directory after arming the detector. The trigger starts the image series and after 1000 recorded images one data container is made available on the buffer of the Detector Control Unit. No further files will be made available until the series is finished either by completing the nth image (nimages) of the nth trigger (ntrigger) or by ending the series using the detector command disarm. As soon as either criteria is met, the second data container, with 800 images in this example, is closed and made available for fetching.

6.3.2. Additional Configuration Parameters

NOTICE
The following parameters are for special conditions and should be set with care and with understanding of the consequences. Changing these parameters to non-default values can have a substantial negative impact on data quality!

- detector | config | threshold/n/energy
- detector | config | threshold/n/mode
- detector | config | threshold/difference/mode

The EIGER2 detectors provide two independent thresholds and it is possible to read out either one threshold, all of them at once, or the difference between thresholds 1 and 2. Reading out more than one threshold at a time is possible with the Stream2, FileWriter2 or Monitor interfaces. The thresholds can be enabled or disabled using the threshold/n/mode configuration parameter, where *n* is the threshold number. The difference mode is enabled with the threshold/difference/mode parameter. This requires thresholds 1 and 2 to be enabled beforehand.

When setting the photon_energy, the lower threshold (n=1) is set automatically to a default value of 50 % of the photon_energy. The value of threshold/1/energy should only be changed in cases where the suppression of fluorescence is a necessity in the experiment. The value of threshold/1/energy should be kept within 50% to 80% of the incoming photon_energy. The API incorporates no sanity check on the threshold/n/energy.

Corrections are enabled by default, but can be turned off using the following detector configuration parameters. In the vast majority of experiments data quality benefits from the data corrections. Therefore, disabling either correction will likely result in inferior data quality.

- detector | config | countrate_correction_applied
- detector | config | flatfield_correction_applied
- detector | config | pixel_mask_applied

Further parameters, represented by the following selection, allow to enrich the meta-data of the image (series) with experimental data.

- beam_center_x
- beam_center_y
- detector_distance
- detector_orientation

- detector_translation
- wavelength (see Section 6.3.3 for dependency with photon_energy)

Further parameters and their function are described in the API Reference document.

6.3.3. Interdependency of Configuration Parameters

Interdependency of Calibration Parameters
The following calibration parameters have an implied or direct dependency. Changing either of the parameters might influence other parameters in the list.

- detector | config | photon_energy

Changing photon_energy sets element to an empty string and sets wavelength to its corresponding value. The threshold_energy is set to half of photon_energy, which is the optimal threshold energy in most cases. The threshold/2/energy is set to 1.3 x photon_energy, which can be used to suppress higher energy photons and cosmic radiation. In order to use values different from the defaults, the threshold energies should be explicitly set after setting photon_energy. The flatfield is recalculated whenever a calibration relevant parameter is changed.

- detector | config | element

Setting the element is equivalent to setting photon_energy to the energy of the $K_{\alpha 1}$ emission line of the element. Hence, photon_energy, wavelength and all parameters that depend on photon_energy are changed accordingly.

- detector | config | wavelength

Setting the wavelength is equivalent to setting photon_energy to the equivalent energy of the wavelength. Hence, photon_energy, element and all parameters that depend on photon_energy are changed accordingly.

- detector | config | threshold/n/energy

Changing the threshold energy causes the flatfield for this threshold to be recalculated.

- detector | config | flatfield

The flatfield applied for a given photon_energy and threshold is a result of the detector calibration. During the factory calibration a multitude of flatfields at different settings have been recorded to ensure optimal data quality of the flatfield for all common settings.

Interdependency of Timing Parameters
The following parameters are essential for exposure timing. Changing these values might influence other values, as described below.

- detector | config | frame_time

If frame_time conflicts with the current count_time, then count_time is set to the difference of frame_time and detector_readout_time.

- detector | config | count_time

If *count_time* conflicts with frame_time, then frame_time is set to the sum of count_time and detector_readout_time. To acquire images with a certain frame rate and best possible duty cycle, a simple procedure is to first set count_time to the inverse of the frame rate and subsequently frame_time to the inverse of the frame rate.

6.3.4. Examples

cURL
cURL, an abbreviation for Client for URLs, is a robust command-line tool designed for making HTTP requests. It is compatible with most operating systems and serves as an effective means of interacting with the SIMPLON API. Below, several example cURL prompts are provided to illustrate how to read and write detector settings, as well as interface files stored on the system. The examples assume the IP address of the system to be 169.254.254.1.

Commands can be transmitted using a PUT request.

```
# Reboot the DCU
curl -X PUT http://169.254.254.1/system/api/1.8.0/command/reboot

# Initialize the detector
curl -X PUT http://169.254.254.1/detector/api/1.8.0/command/initialize

# Arm the detector for acquisition
curl -X PUT http://169.254.254.1/detector/api/1.8.0/command/arm

# Send a software trigger signal
curl -X PUT http://169.254.254.1/detector/api/1.8.0/command/trigger
```

Configuration parameters are modifiable through PUT requests with an additional payload and retrievable via GET requests in the SIMPLON API. As the API primarily uses JSON as the default encoding, it is essential that the payload of a PUT request be in the form of a JSON-encoded string.

```
# Set parameter of configuration key e.g. setting the count_time to 10 seconds
curl -X PUT -d "{\"value\":10}" -H "Content-Type: application/json"
http://169.254.254.1/detector/api/1.8.0/config/count_time

# Read configuration key of e.g. count_time
curl -X GET http://169.254.254.1/detector/api/1.8.0/config/count_time
```

The FileWriter interface creates HDF5 files compatible with applications like NOVENA® or ALBULA. It must be enabled explicitly and a custom filename can be specified. Once an acquisition is completed with the FileWriter interface enabled, the buffered HDF5 files become accessible on the system. Users can then conveniently download or delete these files as needed.

```
# Enable the filewriter interface
curl -X PUT -d "{\"value\":\"enabled\"}" -H "Content-Type: application/json"
http://169.254.254.1/filewriter/api/1.8.0/config/mode

# Change the filename to 'example_file'
curl -X PUT -d "{\"value\":\"example_file\"}" -H "Content-Type: application/json"
http://169.254.254.1/filewriter/api/1.8.0/config/name_pattern

# Request a list of all filewriter files on the system
curl -X GET http://169.254.254.1/filewriter/api/1.8.0/status/files

# Download file 'example_file_master.h5'
curl -X GET http://169.254.254.1/data/example_file_master.h5
--output example_file_master.h5

# Delete filewriter file 'example_file_master.h5'
curl -X DELETE http://169.254.254.1/data/example_file_master.h5
```

Python3

Python offers a convenient and versatile approach for operating the detector through the SIMPLON API. Below is an example Python script that uses the DEigerClient[6] class to operate the detector through the SIMPLON API. The example assumes the IP address of the system to be 169.254.254.1.

```
from DEigerClient import DEigerClient

# Create an instance of the client
client = DEigerClient('169.254.254.1')

# Initialize the detector (only needed once)
client.sendDetectorCommand('initialize')

# Enables the filewriter interface and change the name_pattern to 'example_file_$id'
client.setFileWriterConfig('mode', 'enabled')
client.setFileWriterConfig('name_pattern', 'example_file_$id')

# Set detector configuration parameters (10 images with 0.1 s exposure time each)
client.setDetectorConfig('nimages', 10)
client.setDetectorConfig('count_time', 0.1)
client.setDetectorConfig('frame_time', 0.1)

# Arm the detector, trigger the acquisition and disarm the detector
client.sendDetectorCommand('arm')
client.sendDetectorCommand('trigger')
client.sendDetectorCommand('disarm')

# Download the files from the system to the current folder
client.fileWriterSave('example_file_master.h5', '.')
client.fileWriterSave('example_file_data_000001.h5', '.')

# Delete filewriter file 'example_file_master.h5'
client.fileWriterFiles('example_file_master.h5', 'DELETE')
client.fileWriterFiles('example_file_data_000001.h5', 'DELETE')

# Delete all filewriter files
client.sendFileWriterCommand('clear')
```

[6]You can request the DEigerClient python script from DECTRIS Support (support@dectris.com).

7. REGION OF INTEREST (ROI)

7.1. 4M ROI mode

The Region Of Interest (ROI) feature enables the user to read out a reduced area of the EIGER2 X/XE 9M or 16M detectors at higher frame rates. The ROI mode is also available on the EIGER2 S 9M or 16M without the increase in frame rate. Refer to the corresponding Technical Specifications document for the ROI frame rate specifications.

NOTICE
Please consult the API reference for further details concerning the usage of the region of interest detector configuration parameter (roi_mode).

NOTICE
Please consult the Technical Specifications for further details about the ROI capability of your detector.

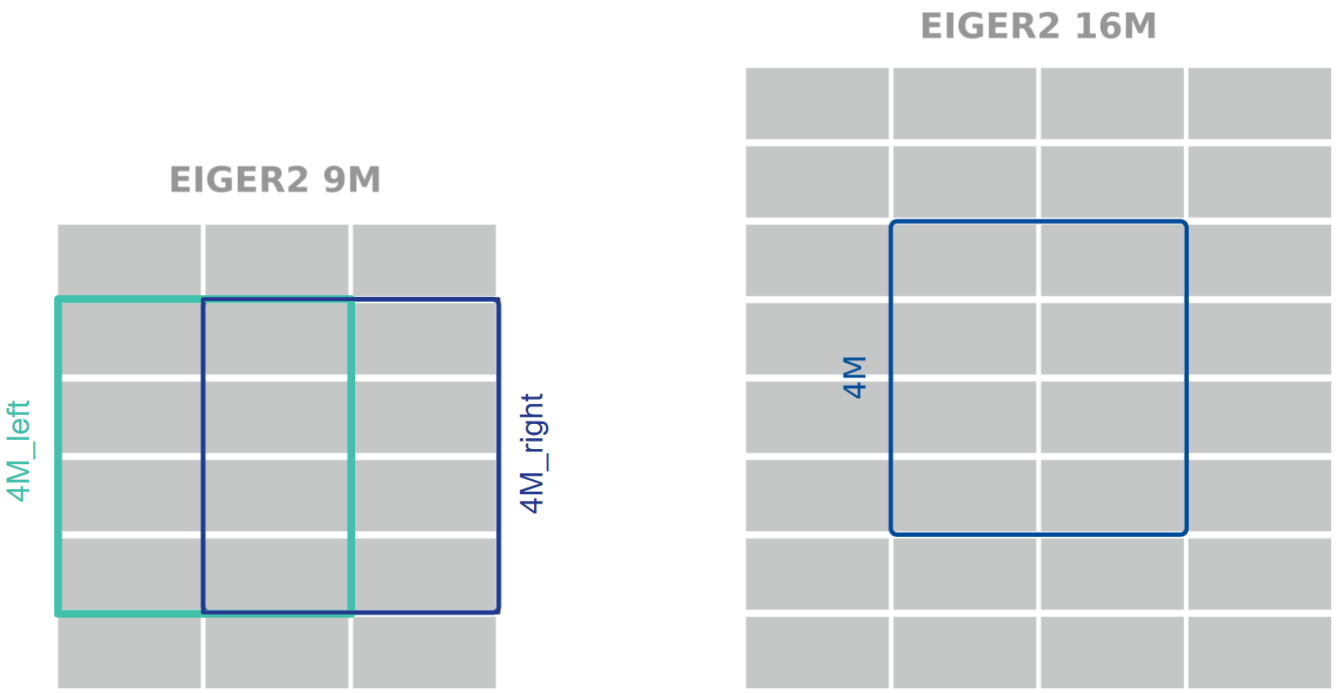


Figure 6. The different ROI modes for EIGER2 as seen from the front of the detector.

The ROI mode is disabled by default and the full active area is read out (Figure 6, grey areas). Changing the ROI mode to another value (eg. ”4M”, ”4M_left”, ”4M_right”, Figure 6, blue / green frames) will cause the detector to only read out the selected modules.

7.2. Lines-ROI

The Lines-ROI® functionality is available in EIGER2 X and XE detectors with 1 module in height (500K, 1M-W, 2M-W). This feature allows to read out a user-defined area of the sensor equivalent to a selected number of lines (pixel rows) counted along the y axis, extending over the full horizontal length of the detector. This can be used to increase the acquisition frame rate with a reduced area selection.

In order to activate this modality, the roi_mode should be set to ”lines” via the API. The ROI area is configured via the parameter roi_y_size, which defines the total number of pixel rows to be read out and should be a multiple of two. The Lines-ROI is centered around the horizontal axis of the detector.

NOTICE
Please consult the API reference for further details concerning the usage of the Lines-ROI® configuration parameters (roi_mode and roi_y_size).

NOTICE
Please consult the Technical Specifications document for further details about the Lines-ROI® capability of your detector.

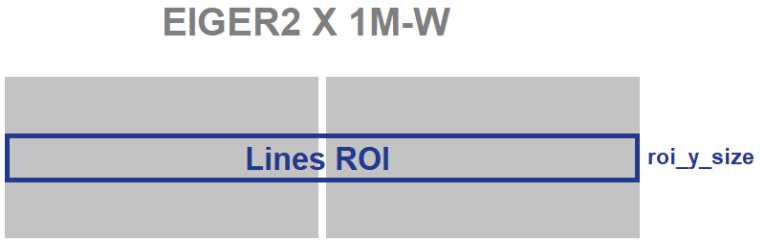


Figure 7. Lines-ROI area selection example for EIGER2 X 1M-W as seen from the front of the detector.

8. TRIGGER MODES

8.1. Introduction

NOTICE
Depending on the type of EIGER2 system, the valid ranges for the trigger parameter differ. Please consult the Technical Specifications for your system. The values presented in the examples below should work on every EIGER2 system. If the settings or the external trigger/enable pulses applied are out of specification, acquisitions will not be performed and the measurement obtained with the detector might be incomplete. All values used in the example are for demonstration purposes only and should be adapted to meet the requirements of your application.

In order to record an image or a series of images, the EIGER2 detector has to be initialized, configured, armed, and triggered or gated. The steps necessary to record an image series are comprehensively described in Section 6.2. The detector can be triggered through software (internal trigger) or by an externally applied trigger signal (external trigger). Various different trigger and gating modes are available and described in the following sections.

8.2. INTS - Internal (Software) Triggering

An exposure (series) can be triggered by using a software trigger. This is the default mode of operation.

Example detector configuration for internally triggered exposure series:

detector | config | trigger_mode	{"value": "ints"}
detector | config | nimages	{"value": 10}
detector | config | ntrigger	{"value": 3}
detector | config | frame_time	{"value": 1}
detector | config | count_time	{"value": 0.7}

The detector starts the first exposure after the trigger command has been received and processed[7]. All subsequent frames are triggered according to the configuration of the frame_time and count_time parameters. The detector records nimages frames per trigger and stays armed until ntrigger are received. Figure 8 depicts an internally triggered series defined by frame_time, count_time and nimages.

[7]As the trigger command is sent over a TCP/IP connection the exact latency of the start of the exposure is hard to predict.

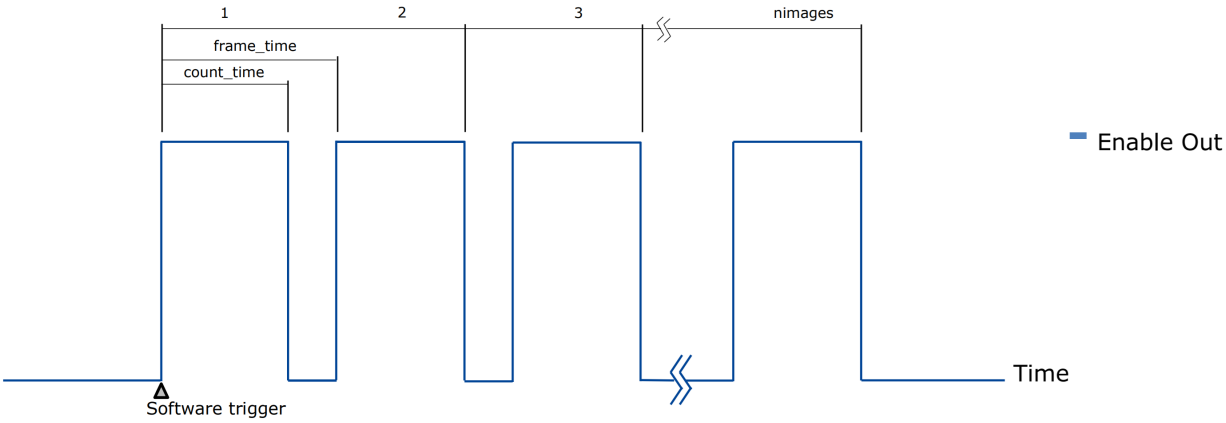


Figure 8. INTS - Series of exposures, defined by frame_time, count_time and nimages, triggered by a software trigger.

8.3. INTE – Internal (Software) Enable

In the trigger_mode 'inte' a single image or series of images with varying exposure times can be started by issuing a number ntrigger of trigger commands. Unlike the trigger_mode 'ints', the 'inte' trigger commands take an additional argument containing the count_time for the subsequent frame. In all enable modes the detector configuration parameter nimages is implied to be 1. The number of frames in a series therefore is solely based on the value of the parameter ntrigger.

NOTICE
The configured count_time and frame_time should be close to the count time and frame time of the shortest expected exposure in the configured series. The set count_time will be used to calculate internal auto-summation configuration values (see Section 3.2.3). In most situations a reasonable estimate of these values is sufficient.

Example detector configuration for an internally enabled exposure series:

detector | config | trigger_mode	{"value": "inte"}
detector | config | nimages	{"value": 1}
detector | config | ntrigger	{"value": 3}
detector | config | frame_time	{"value": 1.0}
detector | config | count_time	{"value": 0.7}

The detector starts the first exposure after a trigger command has been received and processed[8]. All subsequent frames are triggered by individual trigger commands with an additional argument containing the count_time of the triggered frame. The detector stays armed until ntrigger are issued or the detector is disarmed. Figure 9 depicts an internally enabled exposure series defined by count_time (payload of the trigger command) and ntrigger. Table 4 summarizes the commands issued to record the same series.

[8]As the trigger command is sent over a TCP/IP connection the exact latency of the start of the exposure is hard to predict.

Table 4. Command sequence for an internally enabled (inte) series.

Method	Parameter	Payload
PUT	detector | command | arm	
PUT	detector | command | trigger	{"value": 0.7}
PUT	detector | command | trigger	{"value": 2.1}
...		
PUT	detector | command | trigger	{"value": 0.7}

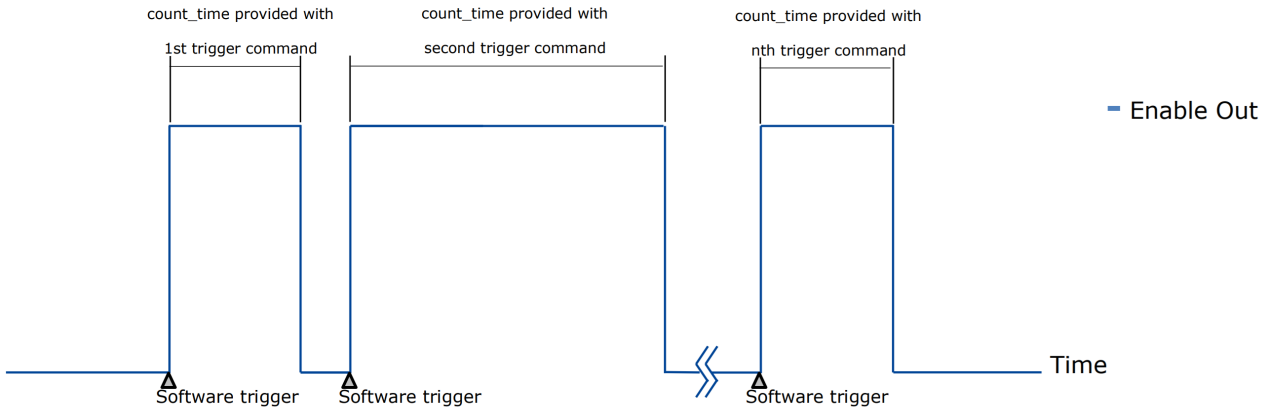


Figure 9. INTE - Series of exposures, defined by count_time (payload of the trigger command) and ntrigger, triggered by a software trigger.

8.4. EXTS - Externally Triggered Exposure Series

CAUTION
Consult the Technical Specifications for details about the required electrical characteristics of the trigger signal.

The EIGER2 detector systems also support external triggering. In the trigger_mode 'exts', nimages are recorded per trigger until ntrigger are received. Both count_time as well as frame_time are defined by the configuration. Example settings for an externally triggered exposure series are shown here:

detector | config | trigger_mode	{"value": "exts"}
detector | config | nimages	{"value": 10}
detector | config | ntrigger	{"value": 1}
detector | config | frame_time	{"value": 1.0}
detector | config | count_time	{"value": 0.7}

After the detector has been initialized, configured, and armed the acquisition can be triggered by a single external trigger pulse. The detector starts exposing after the (electrical) trigger signal has been issued. All subsequent frames are internally triggered according to the information previously configured by the frame_time and count_time parameters. The detector records nimages frames and stays armed until ntrigger are received. Figure 10 depicts an externally triggered series defined by frame_time, count_time and nimages.

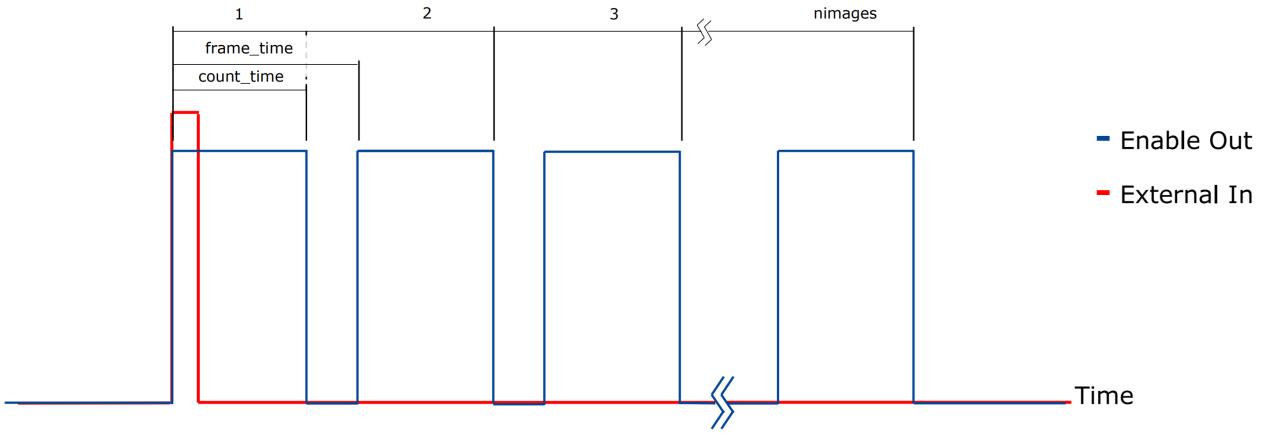


Figure 10. EXTS - Exposure series defined by frame_time, count_time and nimages, triggered by a single external trigger pulse. Note that the periods are not drawn true to scale.

8.5. EXTE - Externally Enabled Exposure Series

CAUTION
Consult the Technical Specifications for details about the required electrical characteristics of the trigger signal.

The EIGER2 detector systems also support external enabling. In the external enable mode 'exte' a series of ntrigger frames can be recorded. The count time as well as the period of individual frames of a series are defined by the duration of the high state of the external enable signal. In all enable modes the detector configuration parameter nimages is implied to be 1. The number of frames in a series therefore is solely based on the value of the parameter ntrigger.

NOTICE
The configured count_time and frame_time should be close to the count time and frame time of the shortest expected exposure in the configured series. The set count_time will be used to calculate internal auto-summation configuration values (see Section 3.2.3). In most situations, a reasonable estimate of these values is sufficient.

Example detector configuration for externally enabled exposure series:

detector | config | trigger_mode	{"value": "exte"}
detector | config | nimages	{"value": 1}
detector | config | ntrigger	{"value": 10}
detector | config | frame_time	{"value": 1.0}

detector | config | count_time	{"value": 0.7}

After arming the detector, the acquisition can be enabled by an external signal. The value ntrigger defines how often this can be repeated. The detector starts exposing the first image after the rising edge and stops after the falling edge of the external trigger signal. In the same manner, all subsequent frames are externally enabled. The count time and period are therefore solely determined by the external enable signal and the limitations of your detector system. The detector records as many frames as valid (according to the specifications) enable pulses are received until the value set for ntrigger is reached. Figure 11 illustrates a externally enabled series.

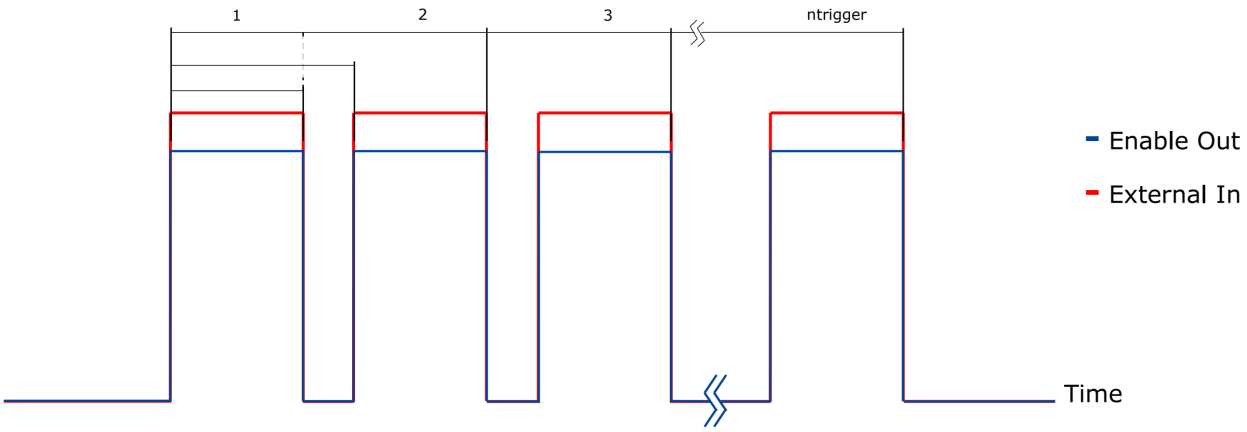


Figure 11. EXTE - Exposures defined by external enable

8.6. EIES - Externally Interrupted Exposure Series

NOTICE
The externally interrupted trigger series is not available on the EIGER2 R 250K and EIGER2 R 500K detector systems.

In the externally interrupted exposure series ("eies" mode), frames will be started and stopped via the external trigger. The first trigger will start the first frame. Subsequent triggers will stop the running exposure and immediately start the next exposure. The length of the trigger pulse itself is inconsequential. The total number of frames to be acquired is defined by ntrigger. However, in this mode, it is necessary to send an additional trigger (ntrigger + 1) to stop the last frame and end the series. Setting the count_time and frame_time as close to the expected real value is recommended to ensure the detector count rate correction works optimally. This mode is useful when the exact length of an exposure is unknown and no gating signal is available.

It is also possible to skip triggers if not all triggers should start a new frame. This can be done by configuring the parameter ntriggers_skipped. If set to 0, every trigger starts a new frame. If set to 1, every second trigger starts a new frame (skipping 1). If set to 2, every third trigger starts a new frame, and so on. The final number of images acquired will still be equal to the value of ntrigger.

NOTICE
To use this mode, the parameter nimages has to be set to 1. The number of frames to be acquired will be defined by ntrigger. This configuration must be set before enabling the "eies" mode, otherwise an error will be returned and the mode cannot be enabled.

Example configuration for an externally interrupted exposure series:

detector | config | nimages	{"value": 1}
detector | config | ntrigger	{"value": 4}
detector | config | trigger_mode	{"value": "eies"}
detector | config | frame_time	{"value": 1.0}
detector | config | count_time	{"value": 0.99}
detector | config | ntriggers_skipped	{"value": 0}

The configuration above will generate 4 images, with a total of 5 external triggers needed (ntrigger + 1). The additional trigger is required to end the last frame of the series.

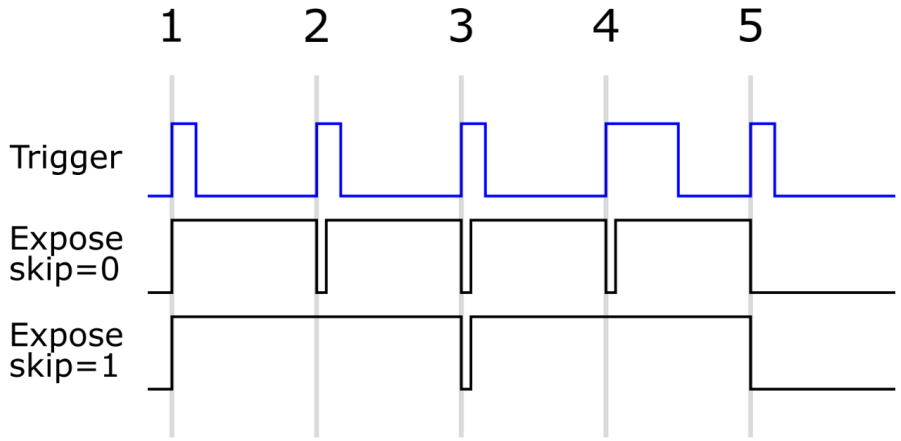


Figure 12. EIES - Schematic illustrating the "eies" mode and the usage of the ntriggers_skipped parameter.

8.7. EXTG - Externally Gated Exposure

NOTICE
The trigger mode "extg" is only available with EIGER2 X and XE detectors.

CAUTION
Consult the Technical Specifications document for details about the required electrical characteristics of the gate signal.

External gating allows for measurements down to a few tens of nanoseconds. In this mode, the detector counts only when the external input signal is high. This mode allows multiple short exposures to be summed in one image. This functionality is useful to capture data from rapidly repeating events with a high temporal resolution, such as in pump-probe experiments.

In this documentation, a "gated window" describes the operation, time, and behavior of the detector between the transition from low to high and back to low of the signal on the external trigger input. Typical times for "gate windows" are a few tens of ns to a few 100 ns. The delay and jitter of the external gating is well below 100 ns.

The EIGER2 application specific integrated circuit (ASIC) offers two 16 bit counters per threshold. To allow maximum flexibility, two different gating modes (”extg_mode”) are provided: ”single” and ”double”.

The ”double” mode allows to acquire two images with two different delay times to be recorded during the same acquisition by alternating the two counters. For example, this can be used in pump-probe experiments to acquire an image of the ground state and one of the excited state for each pump-probe cycle in the same acquisition. This mode is explained in more detail below.

The ”single” mode connects the two 16 bit counters to form one 32 bit counter. This makes it possible to take gated acquisitions with a 32 bit dynamic range at the cost of a longer dead time. This mode is useful if short gating times are needed. All other trigger modes use a feature called auto-summation to achieve the 32 bit dynamic range while keeping a duty cycle above 99.9% (see Section 3.2.3). This mode is explained in more detail below.

General Usage

To use the external gating feature, the trigger_mode ”extg” needs to be set. If this mode is set, the behavior of the detector is controlled by a signal applied to the trigger input. In this mode, the enable out signal only mirrors the trigger input signal and does not give feedback on the internal status of the detector.

For the best results, the count rate correction and retrigger should be turned off in gating mode:

detector | config | countrate_correction_applied	{value: ”false”}
detector | config | counting_mode	{value: ”normal”}

Gating Mode "double"

In gating mode ”double”, the EIGER2 system will write the counts detected during the first gated window into counter A and the value of the second gated window into counter B (see Figure 13). The next gated window will be added again to counter A and so on. The minimal time between two gate signals is 100 ns. After the number of gate windows reaches the configured number of exposures per image nexpi for both counters A and B, a readout will happen (e.g. if nexpi = 4, the readout will happen after 8 gated windows have passed, meaning 4 gated windows will have been summed up in each counter). The readout takes less than 2 ms after which the exposure of the next image can be started. This will be repeated until the set number of nimages is reached, where every measurement produces 2 images (nimages = 6 would correspond to 3 measurements). Due to this, the configured number of images nimages has to be an even number.

Until the actual readout happens, the input line will continue to alternate the counters for every window received (typically derived from a continuous overall timing clock, such as e.g. the synchrotron ring clock). This way, the synchronization of the gates is maintained and the counters A and B will always be associated with the same gate delay in every readout. Note that in gating mode the detector does not auto-disarm and will continue to send an EXT OUT signal even after the readout has happened.

In this mode, it is very important to avoid missed trigger signals. Check the quality of the trigger signal and avoid any ringing of the signal or other unwanted interference. Refer to the Technical Specifications document for details on the trigger input signal level and impedance.

The images of the two counters can be read out using the Stream2, FileWriter2 or the Monitor interfaces. The counter corresponding to each image can be figured out using the image_id, passed as Stream2 header or tiff metadata. Counter A will always have an even image_id, while counter B will always be uneven.

This feature can be used for example in a pump-and-probe experiment to capture the state before and after the pump simultaneously in a single measurement.

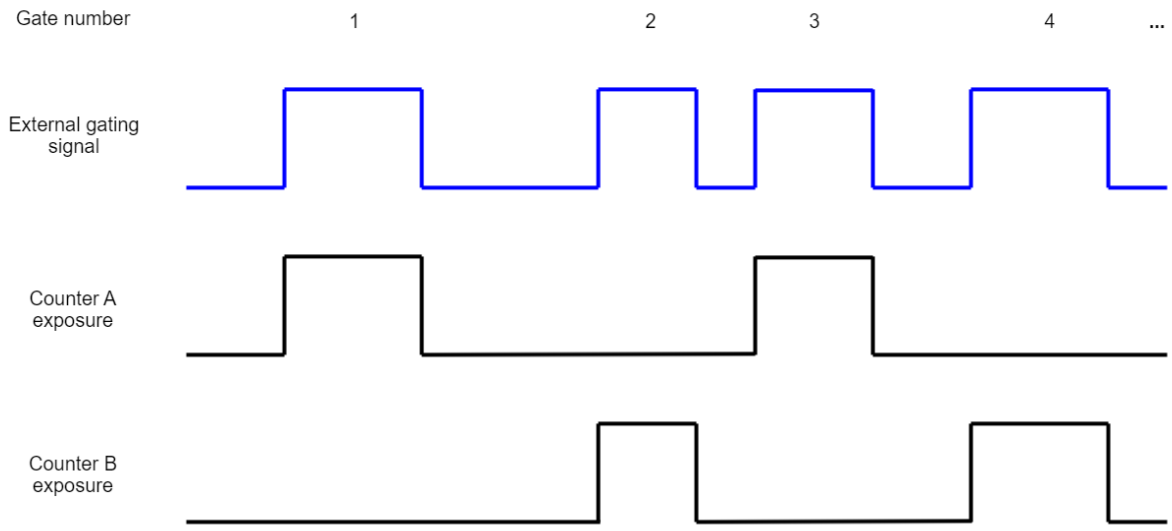


Figure 13. EXTG - Scheme illustrating the gating mode ”double”.

WARNING
In the EXTG mode ”double”, nimages has to be an even number and ntrigger has to be set to 1. If these parameters are not configured correctly, an error will be returned and the mode cannot be enabled.

Table 5. Example detector configuration for externally gated exposure series "double":

detector | config | ntrigger	{”value”: 1}
detector | config | nimages	{”value”: 10}
detector | config | trigger_mode	{”value”: ”extg”}
detector | config | extg_mode	{value: ”double”}
detector | config | nexpi	{”value”: 10000}
detector | config | countrate_correction_applied	{value: ”false”}
detector | config | counting_mode	{value: ”normal”}

Gating Mode "single"

For gating measurements using gating mode ”single”, the two 16-bit counters of each threshold will be connected to form a single 32 bit counter. In ”single” mode the system adds up the signals in the combined counter. The advantage is a very large dynamic range, as each threshold gets a single 32 bit combined counter. Photons arriving during each gated window will be integrated in the combined counter. After the desired number of gated windows is reached, as defined in nexpi, a readout will happen and produce a 32 bit image for each threshold enabled. The readout will take less than 2 ms. This will be repeated until the configured number of images nimages has been reached.

Please note, that it is not necessary to use the mode ”single” gating mode to achieve a 32-bit dynamic range. The other trigger modes offer auto-summation, also achieving 32 bit depth, and which are generally recommended unless you need very short gating windows and minimal jitter.

WARNING
In the extg_mode ”single”, ntrigger has to be set to 1. If ntrigger is not configured correctly, an error will be returned and the mode cannot be enabled.

Table 6. Example detector configuration for externally gated exposure series "single":

detector | config | ntrigger	{value: 1}
detector | config | nimages	{value: 20}
detector | config | trigger_mode	{value: ”extg”}
detector | config | extg_mode	{value: ”single”}
detector | config | nexpi	{value: 123456}
detector | config | countrate_correction_applied	{value: ”false”}
detector | config | counting_mode	{value: ”normal”}

When changing back to other trigger modes, consider enabling the retrigger and count rate correction again by setting the counting_mode to ”retrigger” and countrate_correction_applied to ”true”. The maximum count rate is only achievable with retrigger enabled.

9. PIXEL MASK

9.1. Applying the pixel mask

The detector configuration parameter pixel_mask_applied enables (True) or disables (False) applying the pixel mask on the acquired data. Every threshold has its own pixel mask. These masks may differ from one another. If true (default), pixels which have any bit set in the pixel_mask are flagged with ($2^{\text{image bit depth}}-1$). Please consult the API Reference for details on the detector configuration parameter pixel_mask.

9.2. Updating the pixel mask

9.2.1. Overview

Updating the pixel mask of a EIGER2 detector involves four basic steps:

1. Retrieving the current pixel mask from the detector system via the SIMPLON API.
2. Manipulating the pixel mask to add or update pixels.
3. Uploading the updated pixel mask to the detector system via the SIMPLON API.
4. Persistently storing the updated pixel mask on the detector system by sending the detector command "arm".

9.2.2. Retrieving the current mask from the detector system

The pixel mask can be retrieved from the detector system by a GET request on the detector configuration parameter threshold/n/pixel_mask where *n* is the threshold number. The data of the pixel mask is retrieved either as tiff or in JSON serialization by choosing application/tiff or application/json in the get request accordingly.

9.2.3. Manipulating the pixel mask

NOTICE
For details about the pixel values in the pixel mask and their meaning, consult the API Reference.

TIFF

If the pixel mask is retrieved and stored as tiff, the uint32 data in the tiff file can be manipulated with ALBULA API.

JSON

If the pixel mask is retrieved as JSON, the HTTP reply has to be parsed correctly into an array. Please see the example below and the API Reference for details. The values in this array can then be manipulated to reflect the required updates of the pixel mask. After updating the array, it has to be serialized again in JSON according to the specifications in the API Reference.

9.2.4. Uploading and storing the pixel mask

The pixel mask is uploaded by sending a PUT request on the detector configuration parameter pixel_mask with the new mask as data. After sending the detector command “arm”, the updated pixel mask is permanently stored on the detector system.

9.2.5. Python example

The following Python code using common libraries provides a simple example for updating the pixel mask:

```
import json
import numpy
import requests
from base64 import b64encode, b64decode

# Simple class to get/set a pixel mask or flatfield
class DMaskClient:
    def __init__(self, ip , port=80):
        self._ip = ip
        self._port = port

    def mask(self, mask):
        url = f'http://{self._ip}:{self._port}/detector/api/1.8.0/config/{mask}'
        darray = requests.get(url).json()['value']
        return numpy.frombuffer(b64decode(darray['data']),
            dtype=numpy.dtype(str(darray['type']))).reshape(darray['shape'])

    def setMask(self, ndarray, mask):
        url = f'http://{self._ip}:{self._port}/detector/api/1.8.0/config/{mask}'
        data_json = json.dumps({'value': {
                                '__darray__': (1,0,0),
                                'type': ndarray.dtype.str,
                                'shape': ndarray.shape,
                                'filters': ['base64'],
                                'data': b64encode(ndarray.data).decode('ascii') }})
        requests.put(url, data_json)

if __name__ == '__main__':
    # Create instance of the mask client
    maskClient = DMaskClient('169.254.254.1')

    # Get the pixel mask
    pixelMask = maskClient.mask('pixel_mask')

    # Copy the mask to writeable buffer, necessary for numpy>=1.16.0
    pixelMask = numpy.copy(pixelMask)

    # Set a new dead pixel [y,x]
    pixelMask[123, 234] = 2

    # Set a new noisy pixel [y,x]
    pixelMask[234, 123] = 8

    # Upload the updated pixel mask
    maskClient.setMask(pixelMask, 'pixel_mask')

    # Ensure that you 'arm' the detector to save the new pixel masks persistently
```

TRADEMARKS AND PATENTS

Registered Trademarks ®

- "DECTRIS": AU, AUS, CH, CN, DE, FR, IT, JP, KR, UK, USA
- "DETECTING THE FUTURE": AUS, CH, CN, EU, JP, KR, UK, USA
- "DECTRIS INSTANT RETRIGGER": AUS, CH, CN, EU, JP, KR, UK, USA
- "DECTRIS EIGER": AUS, CH, CN, EU, JP, KR, UK, USA
- "DECTRIS PILATUS": AUS, CH, CN, EU, JP, KR, USA
- "DECTRIS MYTHEN": AUS, CH, CN, EU, JP, KR, UK, USA
- "DECTRIS SELUN": AUS, CH, CN, EU, JP, KR, UK, USA
- "DECTRIS POLLUX": AUS, CH, CN, EU, JP, KR, UK
- "Lines-ROI": CH, EU, JP

Patents

DECTRIS products are protected by the following patents:

- PILATUS3: EP2734861B1, JP6264615B2, US9081103B2; EP1581971B1; EP1920465B1
- EIGER2: EP2734861B1, JP6264615B2, US9081103B2
- PILATUS4: EP2734861B1, JP6264615B2, US9081103B2