Jungfrau documentation Documentation

Release 0.1

M. Ramilli

Aug 16, 2024

Contents

Contents:

CHAPTER 1

General Introduction

This section contains a quick overlook on the general properties of the JUNGFRAU detector, in order to better understand its properties, the way to control it and its output, independently from the specific installation.

1.1 JUNGFRAU basics

1.1.1 The ASIC

JUNGFRAU is a hybrid pixel detector designed and produced at Paul Scherrer Insitut (PSI)[1], in Villigen (CH), consisting of pixelated ASICs bump-bonded to a semiconductor sensor (tipically 320 μm thick Si, but 450 μm is also available, and the use of high-Z materials like CdZnTe and GaAs is currently under study).

Its pixel pitch is of 75 μm and it features a charge integrating Dynamic Gain Switching (DGS) architecture, which grants it a dynamic range of the order 110 dB. Each pixel has an array of 16 memory cells that can store the analog signal before the readout.

The pixels are assembled in ASICs of 256×256 pixels, which are in turn assembled in an array of 4×2 chips, constituting a single JUNGFRAU module of 1024 columns and 512 rows, for a total of about 500 thousand pixels.

Each ASIC is divided in *supercolumns* of 64 columns $\times$ 256 rows, whose pixel output signal is multiplexed to an individual off-chip Analog to Digital Converter (ADC) for digitization; the whole module is therefore read out by a total of 32 different ADCs.

1.1.2 The readout board

The module read out and control is performed via the JUNGFRAU board (see Fig. 1.2), an electronic component also provided by PSI, which connects directly to the front end module. The slow control is performed through a server (jungfrauDetectorServer) running on a PC mounted on the board running a LINUX-based OS, and a receiver running on a local physical host. The jungfrauDetectorServer allows to control an FPGA (also mounted on the JUNGFRAU

[1] https://www.psi.ch/de/lxn/jungfrau

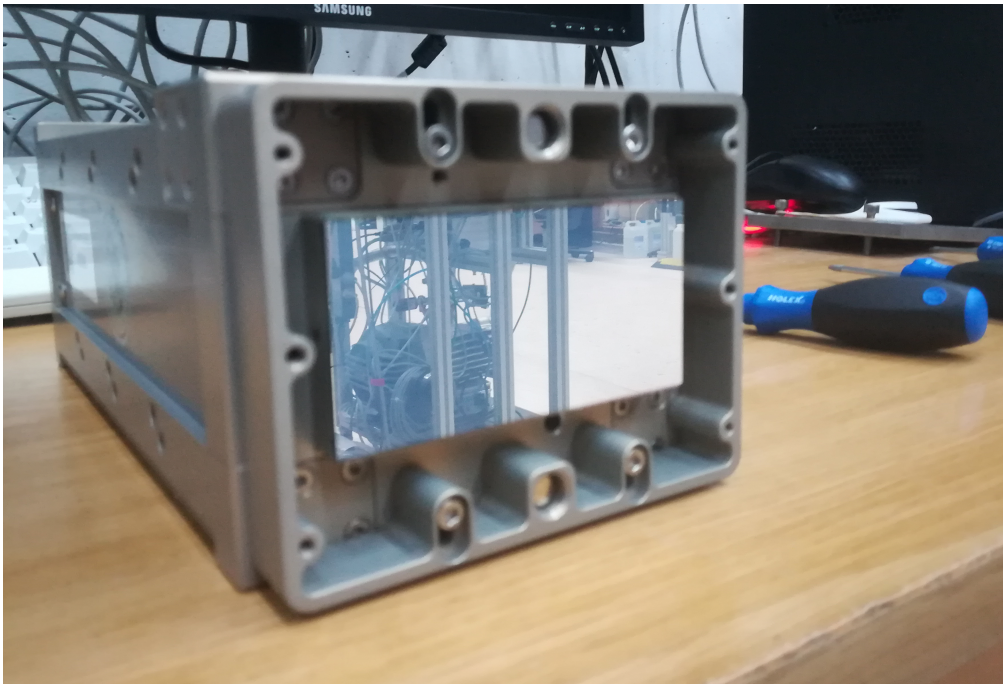

Fig. 1.1: A front end module with silicon sensor.

board), which in turn regulates the operation of the ASICs, by starting and stopping the acquisition, and changing certain parameters in between runs, e.g.:

- integration time;
- number of trains;
- operation mode (DGS or fixed gain);
- self-trigger or external trigger.

1.1.3 External and internal triggering

A module can run in *autotrigger* mode, with read out period adjustable by the user, or, as it will almost always be the case at the European XFEL, in *external trigger* mode. In this second case, each module must be triggered independently, with a TTL signal whose top-up part is at least 100 ns long. The external signal is converted and delivered to the JUNGFRAU board through an additional triggering board (see Fig. 1.3), with three LEMO ports: trigger in, trigger out, and one port without a reserved signal yet (see Fig. 1.4).

1.1.4 Cooling

The module temperature must be kept stable by external cooling: each FEM module dissipates about 20 W, and the amount increases to about 36 W including the readout electronics. It is therefore **strongly not recommended** the module **operation without a functioning cooling** system in place: the performances of the detector will certainly suffer (pedestal stability, noise, etc.), and damage to the components may occur. For normal operation, a temperature of the cooling fluid between 15 $^{\circ}C$ and 20 $^{\circ}C$ is suggested. It is also recommended not to vary the set temperature during operation.

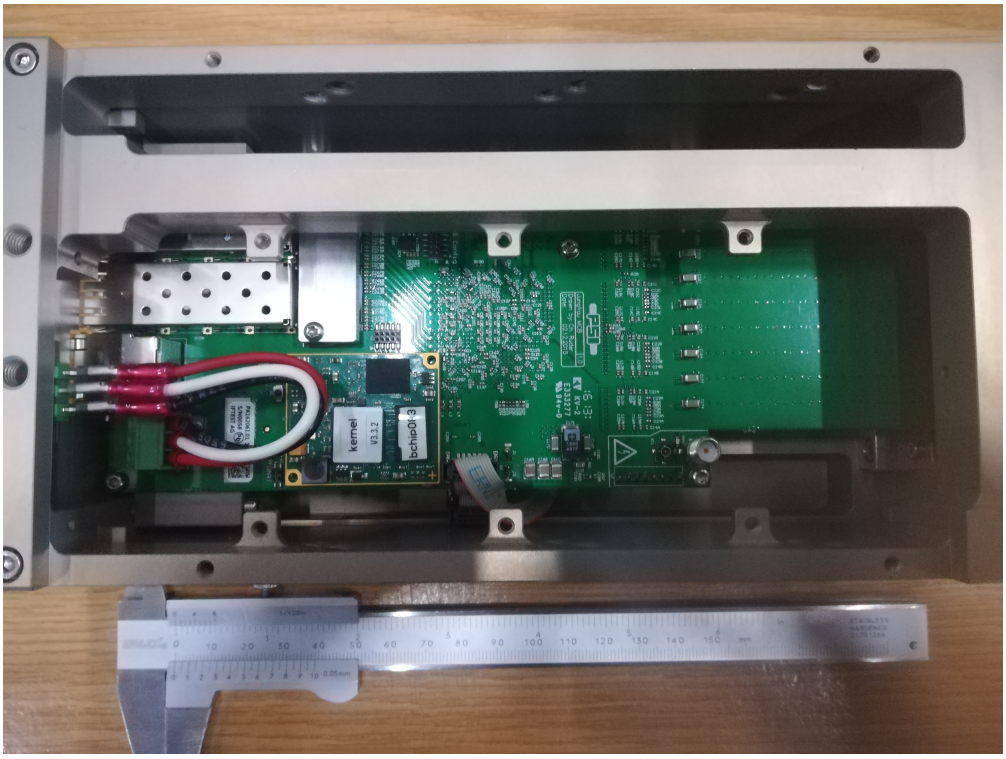

Fig. 1.2: A JUNGFRAU board; the rectangular piece of component on the bottom left, with a golden lining and two reference holes is the microcontroller on which the jungfrauDetectorServer runs.

1.2 Dynamic Gain Switching and Raw output

This section focuses on outlining the generalities of the DGS mechanism as implemented in the JUNGFRAU detector, to better understand the treatment of raw data needed in order to convert the ADC output in physical units.

1.2.1 Dynamic Gain Switching

In order to comply with the dynamic range requirements of a FEL, the pixel architecture of the JUNGFRAU detector is designed with a dynamic gain switching mechanism similar to the one implemented in the AGIPD detector: a pixel-wise threshold comparator switches in additional feedback capacitors in the pre-amplifier if the signal is above a certain value (set module-wise by the factory), thus increasing the feedback capacitance and reducing the gain. There are a total of three subsequent gain stages which can be independently triggered in each pixel, depending on the incoming signal:

- G0, which goes from single photon sensitivity up to a signal of about 20 - 25 10 keV photons;
- G1, which goes up to 500 - 600 10 keV photons;
- G2, which goes up to 8000 - 10000 10 keV photons.

Additionally, there is a fourth gain stage available:

- HG0 (High Gain Zero), which can be statically set for the whole module as first gain stage in place of G0; it has an amplification factor roughly five times higher than G0; this can grant better single photon resolution, at the price of a reduced resolution for multiple events, because the pixel will switch to G1 only after a signal equivalent to 4 - 5 10 keV photons.

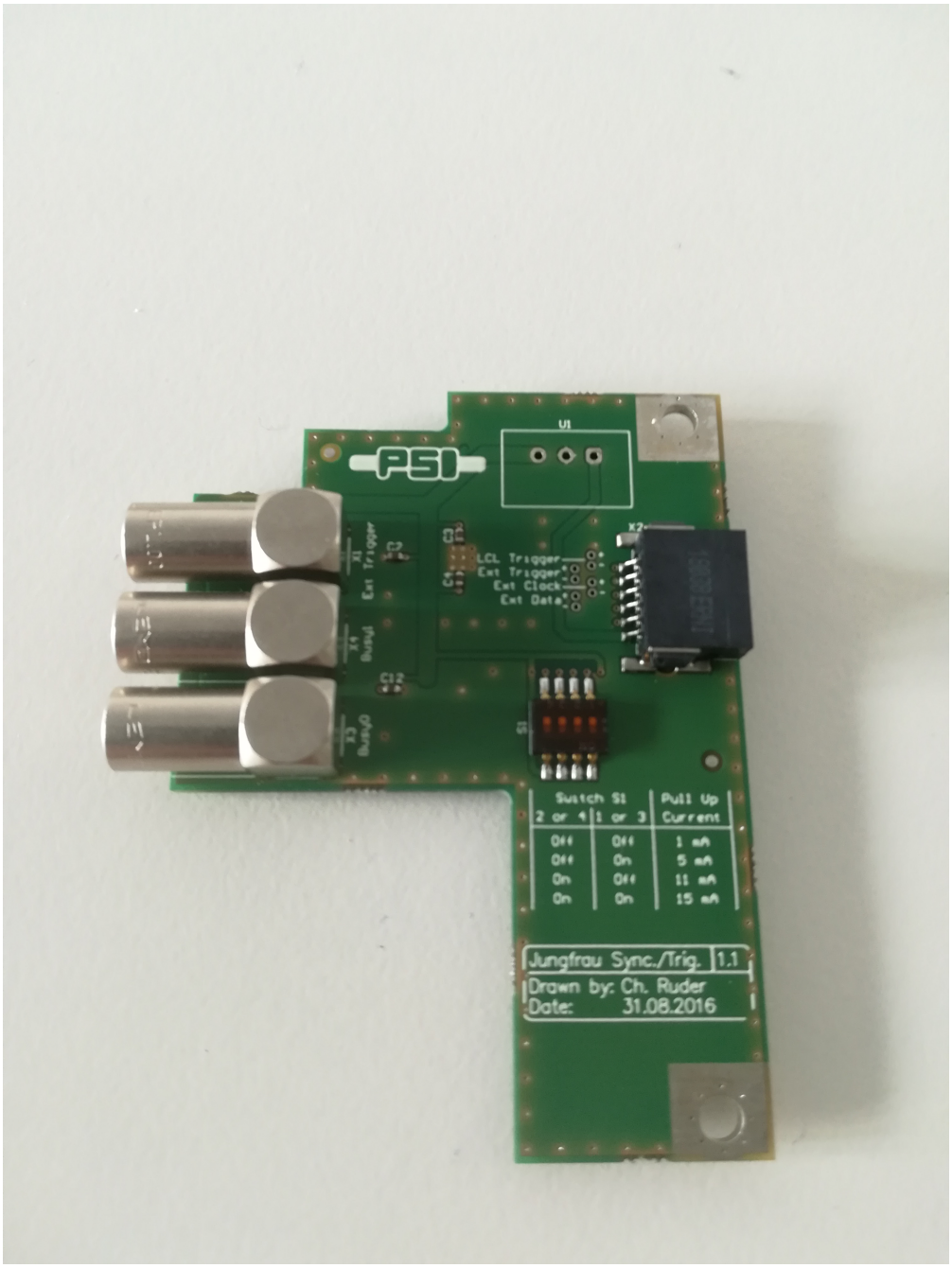

Fig. 1.3: A JUNGFRAU trigger board

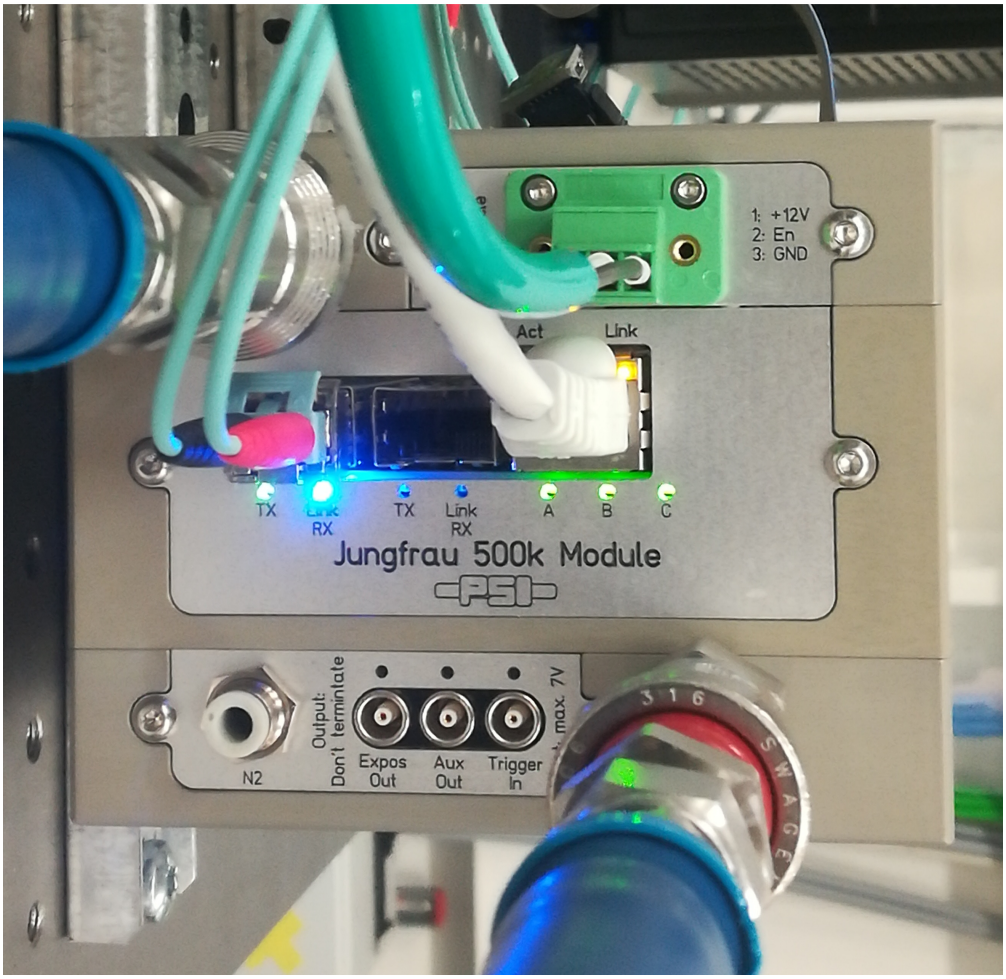

Fig. 1.4: All the ports and connectors on a JUNGFRAU board

The state of the DGS mechanism is recorded memory cell-wise and for every raw image a corresponding gain bit map is produced, which indicates whether each pixel was in G0, G1 or G2 at the end of the integration time.

1.2.2 Raw data output

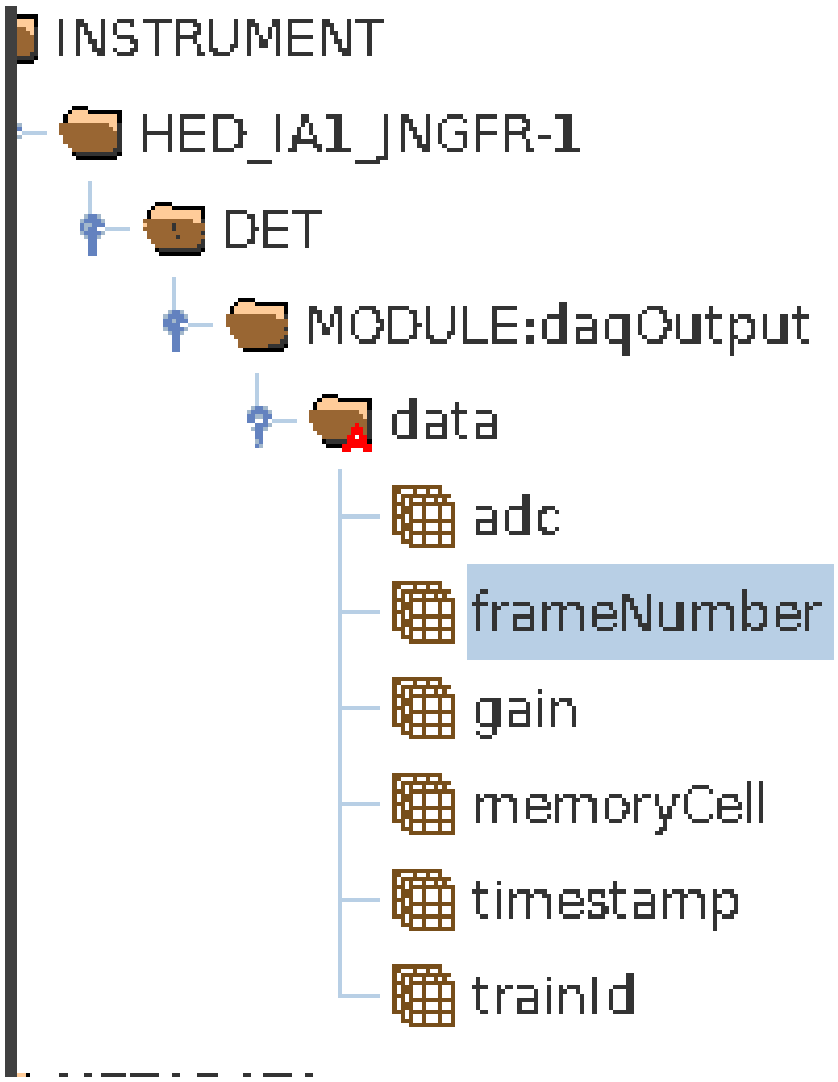

Fig. 1.5: The data structures that will be found in each hdf5 file, at the path corresponding to the RECEIVER device

Therefore, the data structures that will be found in each hdf5 file that contains the output of a JUNGFRAU module are shown in Fig. 1.5 and consist of:

- *adc* contains the raw ADC output and is an array of 16-bit integers, with the following shape: (trains, memorycells, rows, columns);
- *gain* contains the gain bit value (16-bit integer) registered by the threshold comparator at the end of the exposure time; it has the same dimensions of *adc*;
- *timestamp* is an array of 64-bit floats, as long as many trains in the file, indicating the time stamping of each one of them;
- *trainId* an array of 64-bit integers, with the train ID associated to each acquired train contained in the file;

- *frameNumber* is the output of the internal frame counter of each module; this count will reset after reboot;
- *memoryCell* an array 16-bit integers, ranging from 0 to 15, indicating the cell ID of the corresponding image.

Due to some technicality in the chip design, the polarity of the signal after the first gain switching is *inverted*; this means that:

- for G0, increasing input signal corresponds to increasing ADC output, and the values will range from ~ 2000 ADC units (average offset with 10 μs integration time) up to ~ 13000 ADC units (the average gain switching point);
- for G1 and G2, the opposite is true: for increasing input signal, the ADC output *decreases*, which means that the raw output values will range from a minimum of ~14000 ADC units (average offset value) down to 0 - 1000 ADC units for high signals.

An example can be seen in Fig. 1.6.

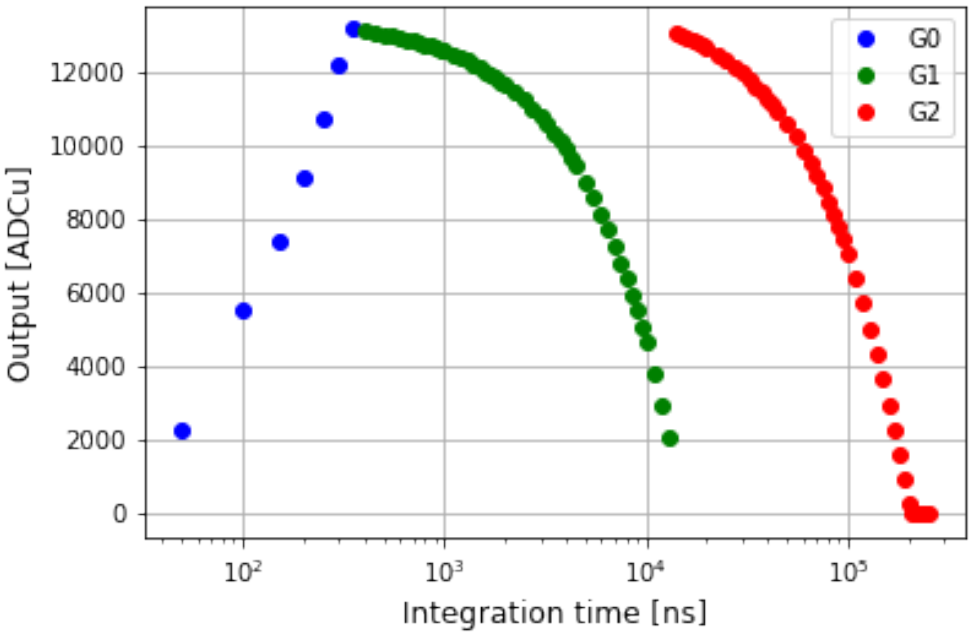

Fig. 1.6: Example of pixel switching gains during an internal current source scan, when the pixel gain switching mechanism is calibrated by providing a constant current and increasing the integration time

As a consequence, the offset subtracted output *without gain calibration correction* will return positive values for the pixels in G0, and negative values for the ones in G1 or G2. This feature of the detector must be taken in consideration while inspecting the corrected output data: if no gain correction has been applied, and substantial parts of the images are in the middle or low gain stage, the physical interpretation of the output may be difficult to subsume.

In order to convert the raw JUNGFRAU ADC output into physical unit, the following constants are needed:

- a memory cell-wise map of the gain calibration correction factors, for each gain stage; these maps exist for each module and provide these factors in ADC units/keV;
- a memory cell-wise map of the offset to subtract from the raw output, for each gain stage; the offset varies with the integration time, therefore such maps must be calculated from dark runs expressly taken every time the integration time has changed.

With both these pieces of information available, the procedure to convert the raw data into physical units is therefore:

1. for each image, the corresponding gain bit map is used to understand at which gain stage each pixel is;

2. on a memory cell by memory cell basis, the offset corresponding to the proper gain stage must be subtracted from the raw output;
3. on a memory cell by memory cell basis, the offset-subtracted ADC value must be converted into physical units, dividing it by the corresponding gain conversion factor.

The gain calibration constants are, for each module present at the EuXFEL, inserted in the calibration database, and so are the corresponding offsets. New offset values can however be easily calculated from dark runs and subsequently injected in the database; details of the procedure are explained in *Data Correction*.

CHAPTER 2

Network setup

This section will illustrate the generalities on how to configure the network between the control machine, the JUNGFRAU module(s) and the receiver server.

2.1 Basic components

The control and operation of JUNGFRAU modules is performed through the use of a software package developed at Paul Scherrer Institut, called SLS Detector Software[1].

The basic architecture of this software is common to all the detectors developed at PSI and is depicted in Fig. 2.1 for a GOTTHARD module; to summarize, it consists of three elements:

1. a client, running the SLS Detector Software, from which commands are launched;
2. a server running on the detector module, which receives the commands from the clients and consequently controls the ASICs (jungfrauDetectorServer);
3. a receiver, which collects the data sent out of the module in form of UDP packets.

This software is then wrapped in Karabo, and can be used via the standard Karabo GUI interface:

- a CONTROL device will allow to launch SLS Detector Software commands;
- a RECEIVER device will collect the data packets.

In principle, the client and the receiver can be running on two separate machines, but in practice, at EuXFEL, the corresponding devices will be running on the same server.

2.2 Device configuration

In order for the network to function properly, there are three ports that need to be configured as part of the same subnet of the client:

[1] https://slsdetectorgroup.github.io/devdoc/

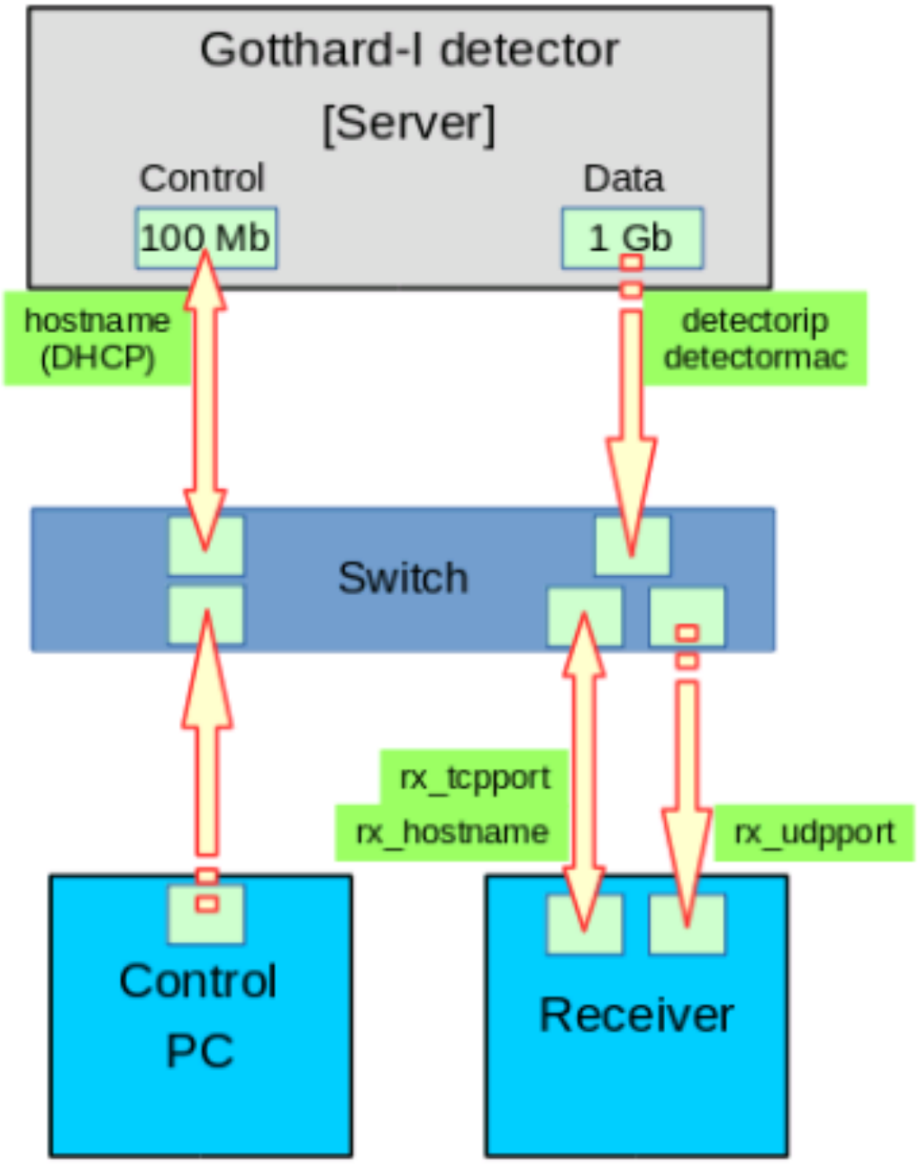

Fig. 2.1: Schema of the architecture of the SLS Detector Software Client-Server-Receiver network. The picture is taken from the "Quick guide to the SLS Detectors Package" by PSI Detector Group

1. a 100 Mb port for control on the JUNGFRAU module, where the commands issued by the client will be sent, to be received by the jungfrauDetectorServer;
2. a 10 Gb port for the data output on the JUNGFRAU module (indicated as 1 Gb on the picture);
3. the port on the receiving server, to which the jungfrauDetectorServer will send the UDP packets to be collected by the RECEIVER device.

Fig. 2.2: Detail of a JUNGFRAU CONTROL device

To configure these parameters, one must access the CONTROL device before instantiation.

- *Detector Hostname* configures the control port on the JUNGFRAU module; at EuXFEL this is usually accomplished by inserting here the alias corresponding to the microcontroller mounted on the JUNGFRAU board; this alias is usually assigned by ITDM at the moment of registration of the module MAC address;
- *RX UDP/IP* configure the IP address of the port on the physical server to which the RECEIVER device must listen to; usually this IP addess is assigned by ITDM;
- *Detector UDP/IP* configures the IP address of the data out port on the JUNGFRAU module; it can be any address, but it **must** be chosen so that it is in the same network of the other two.

Note: all these fields in the CONTROL device are actually lists; to control more modules simultaneously, it suffices to insert all the parameters for every module in each list; one must pay attention that the parameters for a particular module are inserted always at the same position in each list (see e.g. Fig. 2.2).

Fig. 2.3: Detail of a JUNGFRAU RECEIVER device, showing the *RX TCP Port* field

Additionally, TCP ports on the RECEIVER and UDP ports on the CONTROL need to be configured.

The RECEIVER device has the default TCP port of 1954 and this value can be safely used when just one module is used; when more modules are in use, each RECEIVER must be configured to a different TCP port:

- go to each RECEIVER device before instantiation and set *RX TCP Port* field to the desired value (e.g. 1954 for the first RECEIVER, 2954 for the second and so on, see Fig. 2.3);
- go the CONTROL device and fill in the *RX TCP Port* list with the values assigned to each RECEIVER (see e.g. Fig. 2.2).

Configure the UDP port value is useful only if there are more than one RECEIVER device listening to the same IP address; if this is not the case (and it normally isn't at EuXFEL), the default value for *RX UDP/IP Port* of 50001 can be used.

Finally, the name of the server which hosts the RECEIVER must be configured:

- on the uninstantiated CONTROL device, set *RX Hostname* to the name of the server machine where the RECEIVER device is running, e.g. exflconXXX.

2.3 Software and firmware compatibility

The detector group at PSI routinely releases new versions of the SLS Detector Software. Before updating the client installation, one has to make sure that the new version is compatible with the firmware installed on the boards. In case of doubt, please consult the PSI Software Releases page.

In order to check the version of the firmware, there are two possibilities:

- **from command line**: from a shell of the machine where the client is running (after configuration, see *Device configuration*), type: *sls_detector_get detectorversion*; you should get a number formatted as hexadecimal like: *0x171113*; this number is the date of release of the firmware, which identifies it (in the example corresponds to 13.11.2017);
- **from the CONTROL device**: there is a field in the CONTROL deice that checks the firmware version (see Fig. 2.4); there it can be seen the release date of the firmware, formatted as hexadecimal.

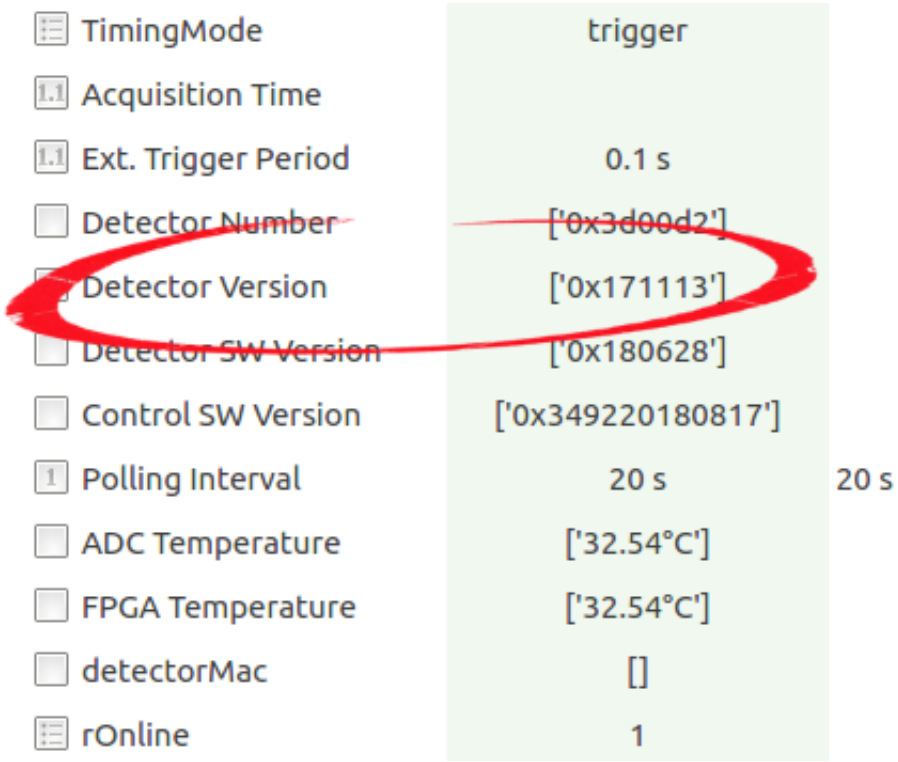

Fig. 2.4: Detail of a JUNGFRAU CONTROL, highlighting the *Detector Version* field

CHAPTER 3

Operation

This section highlights the balic principle of operation of a JUNGFRAU module.

3.1 Turning on and off

3.1.1 Cooling

Each module of JUNGFRAU needs cooling to dissipate the heat generated by the FEM and the JUNGFRAU readout board, for a total of about 36 W. Without cooling in place, the modules will heat up, operation parameters like the sensor current will not be stable, increasing the difficulty of the measurements. Moreover, there is the chance of damaging the electronics with an inefficient heat dissipation. Therefore, **before powering up the modules, cooling system must be turned on, and its correct functioning verified**.

For modules in air, a set temperature above the dew point is suggested, in order to avoid possible condensation on sensitive electronic components: any temperature between 15 $°C$ and 20 $°C$ is suitable.

For the reasons explained above, **operation without proper cooling is strongly not recommended**.

3.1.2 Power

A module is powered by applying a Low Voltage value of +12 V through the dedicated connector on the JUNGFRAU board. This will turn on the board and the ASICs in the FEM. An average JUNGFRAU module draws a current in the range of 2.5 - 3.5 A, but during the power up it can briefly reach higher values: therefore, a current limitation of 5 A per module (if there are more modules in series) is suggested.

After a few seconds from the power up, the current should stabilize and the module is on. To control its operation from Karabo GUI, two devices are needed:

1. a RECEIVER per module;
2. a CONTROL device.

These two devices must be turned on *in this order* (after the module is powered): first all the RECEIVERS, then the CONTROL.

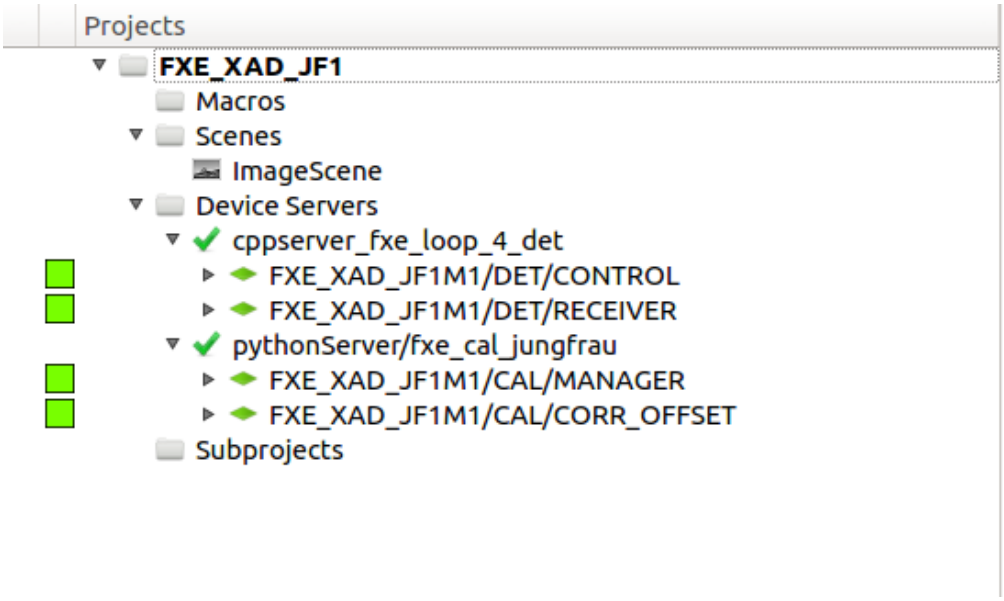

Fig. 3.1: Detail of the JUNGFRAU Project at FXE; the RECEIVER and CONTROL devices can be seen, both active

High voltage to the sensor can be supplied in two ways:

- if the JUNGFRAU board of the module in use has an internal voltage divider board (see Fig. 3.2), HV can be applied via CONTROL; it will not applied until the CONTROL device is first instantiated and in that case the default value of the *HighVoltage* parameter (see *Basic operation settings: CONTROL device*) will be applied;
- otherwise, HV must be applied externally by a power supply.

In both cases, HV can range between 90 V and 200 V. There is a slight dependence of charge sharing performances from HV value (the higher the value, the lower the charge sharing), therefore a consistent biasing throughout operation is suggested. For normal operation, a HV value of 180 V is suggested.

To turn off the module, it suffices to turn off the LV. In case the HV is externally applied, it is recommended to switch off the HV first.

3.2 Control configuration

The JUNGFRAU CONTROL device communicates with jungfrauDetectorServer running on each module, and therefore controls the module(s) operation, by setting the acquisition parameters, and giving start and eventually stop to the acquisition. Through this device the data acquisition can be started (by clicking on the START button) and stopped (by clicking on the STOP button); the CONTROL START command has of course no impact on the DAQ and its devices, therefore if data needs to be saved, the run has to be started separately from the RUN CONTROL scene.

3.2.1 Basic operation settings: CONTROL device

Important configuration parameters that are present on the CONTROL device are listed below.

- *Exposure time*: expressed in seconds, sets the integration time for the individual memory cell; an exposure time shorter than 2 μs **is not recommended**.

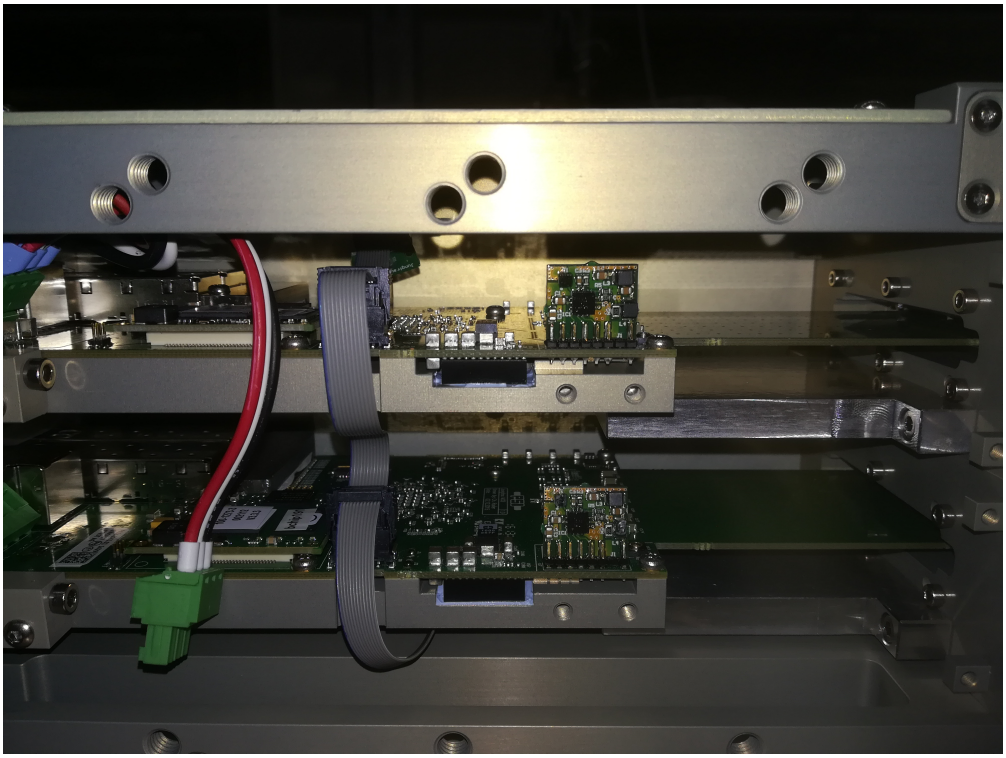

Fig. 3.2: A detail of a JUNGFRAU board in which the voltage divider is present (the small vertical element of PCB); if this is not present, HV must be externally supplied.

- ***Settings*: this property sets the feedback capacitor of the CDS:**
  - *gain0*: default value;
  - *highgain0*: smaller feedback capacitor for higher amplification in G0 stage;
- ***Gain Mode*: this entry sets the operation mode for the modules:**
  - *dynamic*: dynamic gain switching, standard operation mode;
  - *forceswitchg1*: used to measure offsets for G1;
  - *forceswitchg2*: same as above, but for G2;
  - *fixg1*: fixes the feedback capacitance of the pre-amplifier to the one corresponding to G1; no DGS;
  - *fixg2*: same as above, for G2;

  the last two settings should be used only in experimental conditions when no gain switching is foreseen, and should not be used to estimate the offset for G1 and G2 in *dynamicgain* mode, since the presence of the DGS mechanism has some impact on its value.
- *HighVoltage*: if the apposite voltage divider is mounted on the JUNGFRAU Board (shown in Fig. 3.2), the HV to the sensor can be applied via a command to the jungfrauDetectorServer and this command sets it; a recommended operational value is 180 V, but anything between 90 V and 200 V can be used, depending from experimental requirements.
- *Delay after trigger*: sets the delay for the beginning of the exposure time after the trigger input; expressed in seconds.
- *Additional Storage Cells*: sets the *additional* number of memory cells to be used during acquisition; it accepts an integer whose value is between 0 (zero, i.e. *single-cell* operation) and 15 (fifteen, i.e. using all the memory

cells).

- *Storage Cell Start*: sets the first memory cell to be filled (the remaining are filled in a round-robin fashion); default value is 15, which is the default storage cell in *single cell* operation.

Note: *Exposure time* value and readout rate (whether in *autotrigger* or *external trigger* mode) should be compatible quantities; e.g. one should not set 2 ms exposure time, if the device is running in *autotrigger* with an *Exposure period* of 0.001 (i.e. 1 kHz).

3.2.2 Basic operation settings: RECEIVER device

The RECEIVER devices have some operational parameters that need to be configured, especially when moving from *single cell* operation to *burst mode* operation (i.e. with more than one memory cell per train) or viceversa.

- *framesPerTrain*: this parameter configures the schema of the data for the DAQ, by setting how many images per train have to be expected; this number has to be equivalent to number of memory cells that are going to be used (e.g. 1 for *single cell* operation, equal to 5 if only 5 memory cells are used, or 16 if one wants to use the whole memory cell array and so on);
- *Storage Cell Start*: this parameter needs to have the same value as its equivalent on the CONTROL device.

3.2.3 External trigger and autotrigger

The JUNGFRAU can be operated in two ways: *external trigger* and *autotrigger* mode.

To set the detector to operate in *external trigger* mode, the following parameters of the JUNGFRAU CONTROL must be set:

- *Trigger mode*: trigger;
- *extSig0*: communicates the shape of the trigger signal; *trigger-in-rising-edge* is the viable option for EuXFEL trigger;
- *Number of Triggers*: this sets how many trigger inputs the detector must accept before declaring the measurement terminated; in normal operation, this is set to the desired number of trains to acquire;
- *Number of frames*: this sets the number of acquisitions per trigger; it is normally set to 1 in external trigger mode;
- *Exposure period*: this dictates the acquisition period at the end of which the ASIC is read out; it is irrelevant if *Number of frames* = 1;

Conversely, to operate the module in *autotrigger* mode, the CONTROL device must be set:

- *Trigger mode*: auto;
- *Number of Triggers*: 1;
- *Number of frames*: set to the desired number of acquisitions;
- *Exposure period*: this regulate the period between *frames* at the end of which the ASICs are read out and it must be expressed in seconds; we suggest not to exceed ~100 readouts per second, otherwise there may be packet loss.

To have an estimate of the total output, the following formula can be used:

$$N_{images} = N_{triggers} \times N_{frames} \times N_{memcells}$$

In *single cell* operation $N_{memcells} = 1$ (obviously); in *burst mode* operation the number of memory cells used is configurable and ranges from 1 to 16.

Note: it is strongly suggested to terminate an acquisition properly, i.e. by pressing STOP or waiting that the set number of images is acquired. Improper termination of an acquisition will likely lead the devices in ERROR state.

3.2.4 Single cell and 'Burst' operation modes

The detector can be configured to acquire one image per acquisition cycle (i.e. before reading out the image), or to use more than one of the sixteen storage cells: these two operation modes are referred to as *single-cell* and *burst* operation mode, respectively. Since the readout time of one single image is of about 819 μs, *burst* mode is necessary to acquire more than one image per EuXFEL train.

To properly configure the detector, some properties have to be changed both in the CONTROL and RECEIVER devices.

Table 3.1: Change operation mode

Device	Property	Single	Burst
CONTROL	Additional Storage Cells	0	15
RECEIVER	Frames per Train	1	16
RECEIVER	Burst Mode	False	True

It is recommended *NOT* to change the property *Storage Cell Start* and instead leave it to the default value of 15. *NOTE:* it is important that the DAQ is set in IGNORE while the detectors are being reconfigured, since the schema is updated every time the *Apply Configuration* is called, and it is done via reading the receiver *Frames per Train* property: forgetting to do so will eventually result in data being saved in arrays of the wrong shape.

It is also important to note that, for details linked to the firmware, the delay between the trigger signal and the first exposure gate opening slightly increases when switching to *burst* operation mode, and it recommended to always control the synchronization between pulses and gates on the oscilloscope.

In *burst* mode the CONTROL device property *Exposure Timeout* can be used to *further* increase the dead time in between acquisition gates with respect to the factory minimum of 2.1 μs. Default value for this property is 25 ns (cannot be reduced): any value larger than that can be entered (in nanoseconds) when the CONTROL device is unistantiated.

3.2.5 Temperature Control

There is the possibility to set a Temperature Control based on the temperature readout of the sensor on the Jungfrau board: if a certain temperature threshold is passed, the firmware will power off the ASICs of the FEM. The possibility to set up this safety check is implemented in the Karabo CONTROL device through five properties:

- *Temperature threshold*: is the threshold that needs to be passed to trigger the *Temperature Event*; default is 65 $^{\circ}C$, but it can be set at will and a different threshold can be set, if needed, for each module controlled by the same CONTROL device; if only one entry is left, it will be applied to each module;
- *Temperature Control*: when set to '1' (default is '0') enables the temperature control;
- *Temperature Events (Modules)*: a list of results of the temperature control, for each module; when it changes to '1', it means that that particular module triggered a *Temperature Event* (i.e. it passed the set threshold);
- *Temperature Event (Summary)*: a single boolean summarizing the status of the whole detector, by doing the OR of all *Temperature Event* output of each module;
- *Reset Temperature Event*: a button that resets all the *Temperature Events* to 0 (zero); of course, if the problem that caused the event persists, it will be triggered again, so it is recommended to click the Reset button only after investigation.

Of course, the status of the Karabo device properties will be refreshed on a rate depending from the Polling Interval property (default is 20 s).

Once the control is activated, the temperature checks are performed on the firmware level. This is done because (almost) all the software communications between detector and client are done, for each module, on an individual TCP port, which also means that during acquisition this port is constantly busy (so no real time updates are possible): therefore the Karabo device status *will not update* during acquisition, but the *Temperature Control will work nonetheless*. If the event has been triggered during an acquisition, it will become apparent on the RAW image preview, which will show just the ADC baseline output (similar to what is shown in figure Fig. 5.4). Stopping the acquisition will then lead the Karabo CONTROL device to update its status, and the condition of the *Temperature Control* will become apparent.

CHAPTER 4

Data Correction

In this section, the procedure to update the offset calibration for the JUNGFRAU detector will be described, starting from the procedure to collect useful dark runs.

4.1 Dark runs acquisition

In order to fully characterize the offsets for the JUNGFRAU detector operating with **dynamic gain switching**, dark runs for all the three gain stages need to be acquired, in three separate runs, i.e. one run in *Gain Mode*: *dynamic*, one in *forceswitchg1* and third in *forceswitchg2*. Each run must contain at least 500 trains of real data (i.e. non empty trains due to the DAQ running before the modules are sending out images). In order to maintain consistency with the constant map shape in the database, in the case of **fixed gain** operation three separate runs need to be acquired as well, i.e. in *dynamic* (as there is no *fixg0* available), in *fixg1* and *fixg2*. For the single cell operation these runs may be taken manually; however, since there is a JungfrauDarkChar middlelayer device deployed at each scientific instrument, the usage of this device is strongly encouraged, in order to minimize human error.

4.1.1 JungfrauDarkChar middlelayer

A middlelayer device with Class ID JungfrauDarkChar is deployed at each scientific instrument to automatize the dark run acquisition procedure for the JUNGFRAU detector. The currently deployed version works on a single detector, therefore a different instance is needed for each detector in operation, e.g. in the case of the scientific instrument FXE, there exist one instance for JF1M and a different instance for the JF500K. A version that can operate on more detectors simultaneously is currently under development.

Here a list of its relevant properties:

- *Control device ids*: list of the CONTROL devices it controls (typically one per instance);
- *Receiver device ids*: list of the RECEIVER devices it controls: these should be all the RECEIVERs referring to the one CONTROL in the previous property;
- *DAQ controller id*: self explanatory;
- *Run controller id*: self explanatory;

- *Number of trains*: number of train for each step of the process; default is 500;

The *Run* button starts the procedure; the detector should not be sending data while starting this, and the DAQ should be in MONITORING. The *Cancel* button halts the procedure in case it is needed, but clean reset somehow does not work and device shutdown and reinstantiation is needed in case of not proper termination.

The device will automatically recognize the detector operation mode from the settings in the CONTROL device by reading the properties *Gain Mode*, *Exposure Time*, *Additional Storage Cells* and *Storage Cell Start* (i.e. whether in *burst* or *single cell*, dynamic gain switching or fixed gain) and take dark runs accordingly, i.e. **only** for the current operation mode. It is therefore the responsibility of the operator to correctly configure the detector.

For example: if the CONTROL device is configured with Gain Mode: *fixg2* and Additional Storage Cells: 15, the middlelayer will take dark runs for 'Burst Mode - Fixed Gain' operation only.

While taking runs, the device will also attempt to set the proper Run Type; it is therefore important that all the device properties named *Experiment <operation> - <Gain Mode>* are configured with Run Type name effectively present in the current proposal, otherwise the middlelayer will fail (additional empty spaces are relevant, unfortunately).

NOTE: as dynamic gain switching operation in *burst* mode is not supported, it is important to make sure that the detector is configured in *Gain Mode*: *fixg1* or *fixg2* before using the middlelayer, when operating it with 16 memory

cells.

Below the procedure to manually acquire darks is described. As explained before, it is recommended to use it only if problems with the JungfrauDarkChar middlelayer appear.

4.1.2 Single cell operation

1. Select the desired integration time in seconds (e.g. insert 1e-5 for 10 μs);
2. assuming that the module is running in external trigger mode, select the number of trains, e.g. by setting *Number of Trains* = 1000 and *Number of frames* = 1;
3. select the desired gain stage:
   - for the G0 darks, use the *Gain Mode*: *dynamic* and *Setting*: *gain0*,
   - for the HG0 darks use the *Gain Mode*: *dynamic* and *Setting*: *highgain0*;
   - for the G1 or G2 darks use *forceswitchg1* or *forceswitchg2* gain modes, respectively, when operating the detector in dynamic gain switching;
   - the *Gain Mode*: *fixg1* or *fixg2* should be used for G1 and G2 if the detector is being operated in fixed gain;
4. take the run normally;
5. repeat for all the gain stages.

4.1.3 Burst mode operation

The current procedure to acquire dark runs in *burst* mode with *dynamic gain switching* operation is convoluted and it is not easily performed manually, so it won't be described here. Additionally, this operation mode **is currently not supported**, therefore darks in this configuration should not be taken.

As for *burst* mode, *fixed gain* operation:

1. Select the desired integration time in seconds (e.g. insert 1e-5 for 10 μs);
2. assuming that the module is running in external trigger mode, select the number of trains, e.g. by setting *Number of Trains* = 1000 and *Number of frames* = 1;
3. select the desired gain stage:

- for the G0 darks, use the *Gain Mode*: *dynamic* and *Setting*: *gain0*,
- for the HG0 darks use the *Gain Mode*: *dynamic* and *Setting*: *highgain0*;
- for the G1 or G2 darks use *fixg1* or *fixg2* gain modes, respectively;

4. take the run normally;
5. repeat for all the gain stages.

4.1.4 Remarks

A few final comments:

- the offset value depends from the integration time: if this parameter has been changed, it is necessary to repeat the dark runs acquisition;
- dark runs should be obviously repeated if the performances of the module changes, e.g. due to radiation damage;
- the stability of the offset depends from the thermal stability of the module: it is therefore suggested to let the module run for 10 - 15 min after power up and wait that it has thermalized, before taking darks and/or data.

4.2 Offsets injection

Once the dark runs for all the three gain stages have been acquired, it is possible to calculate new offset constants and inject them into the database by using a Jupyter notebook developed by the CAL team. This notebook accepts both dark runs for *single cell* mode and for *burst mode* (all the three of them must have been acquired in the same operation mode, though), and the Maxwell jobs running it can be launched directly from the myMdC page of your proposal. The procedure can be summarized in the following steps:

1. migrate the three runs (G0, G1 and G2) to the Maxwell cluster;
2. go to the *Calibration Constants* tab in the myMdC page of your proposal;
3. under **Detector** select the desired JUNGFRAU detector;
4. Under **Run Number(s)** input the run numbers for G0, G1 and G2 (in this order);
5. click the *Request* button.

The notebook will require a few minutes (around 5 min) to terminate (for *single-cell*: darks in *burst* may require longer). After that, the newly calculated offset are injected into the database. A report is issued after the jobs are completed, and it can be downloaded from the same page.

In order to update also the online corrected preview, click on 'Load Most Recent Constants' from the calng MANAGER device.

As an example, typical results are also displayed below: on the left, the offset map for each gain stage, for 10 μs integration time.

4.3 Gain correction

There exist notebooks that use the offset and gain calibration constants present in the database, to convert the raw ADC output in physical units, how it has been outlined in subsection *Raw data output*. The recommended procedure is described in the section Calibration using the Metadata Catalogue Interface.

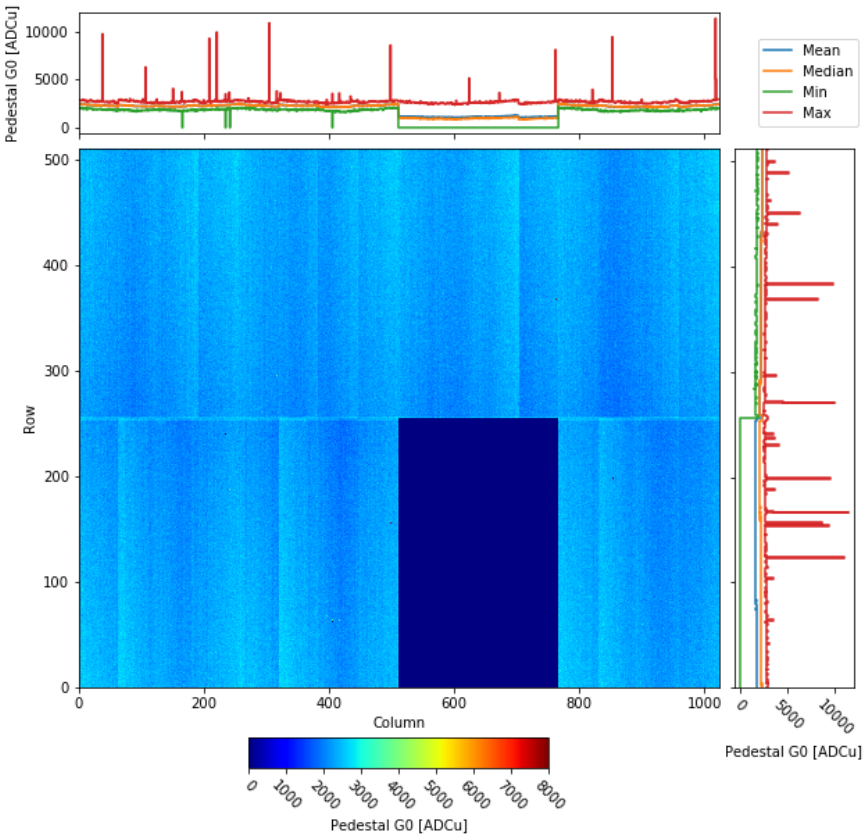

Fig. 4.1: Example of an offset map for G0; please note how the offset value lies around 2000 ADC units

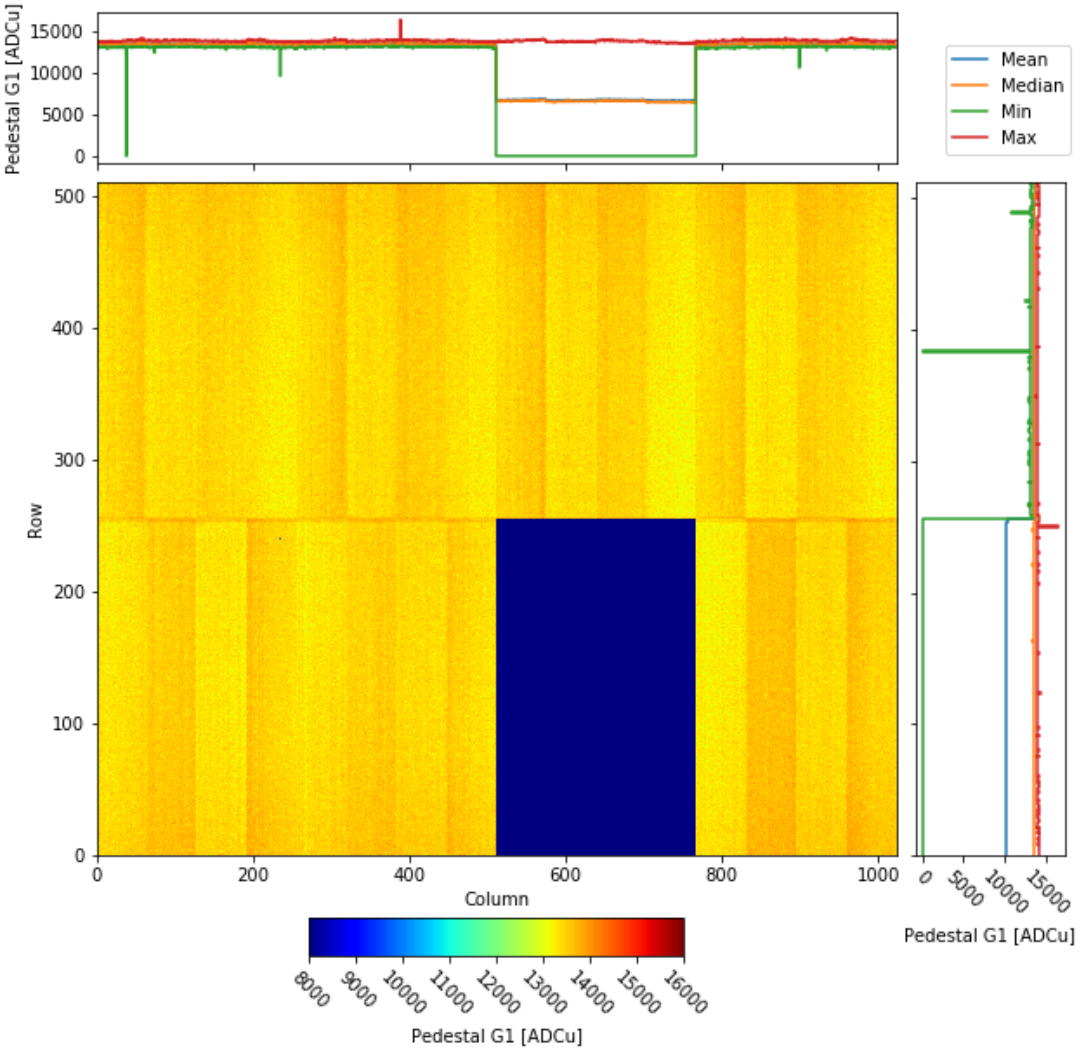

Fig. 4.2: Example of an offset map for G1; due to the signal inversion, the offset value is arounf 13000 ADC units

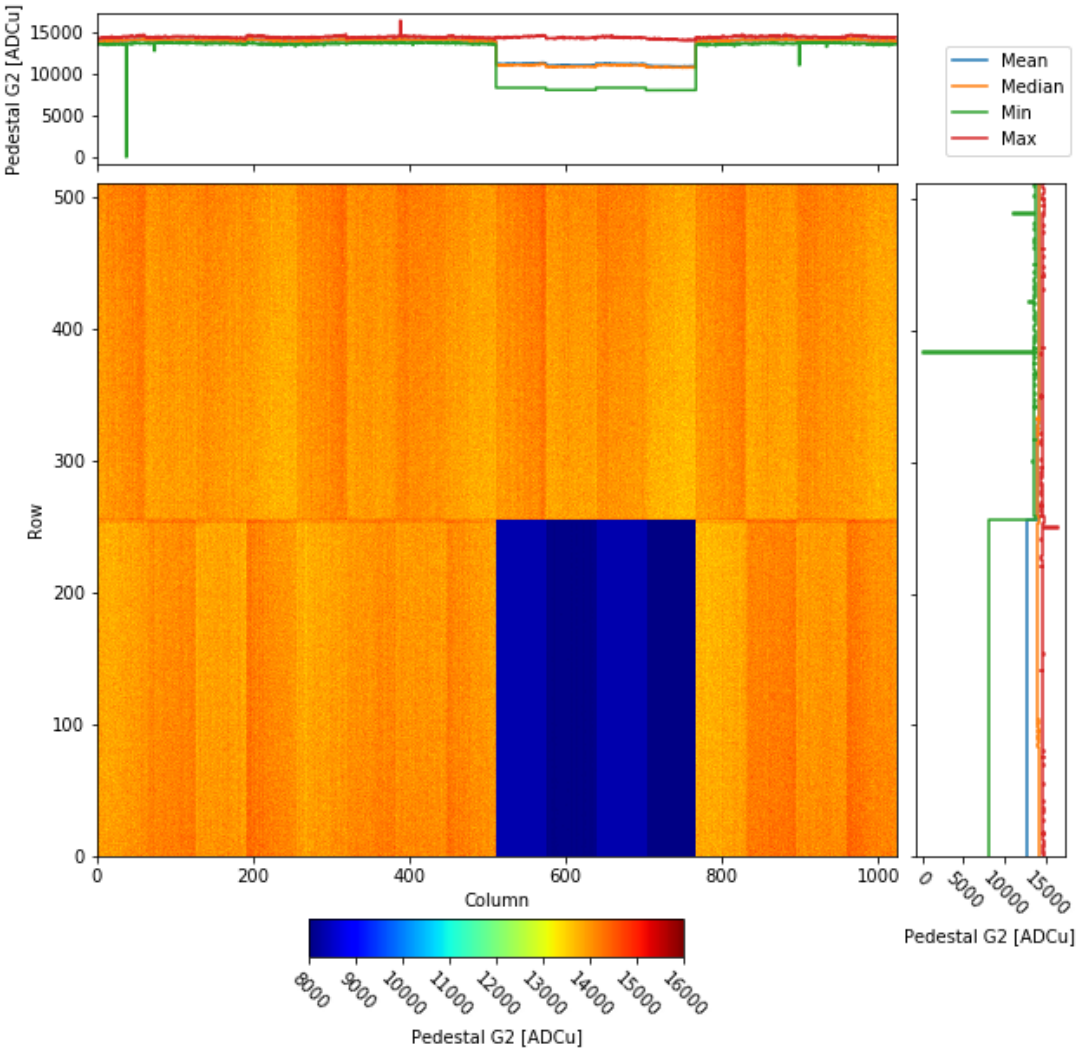

Fig. 4.3: Example of an offset map for G2; similar to the case of G1, due to signal inversion the offset value is again at the high end of the ADC range

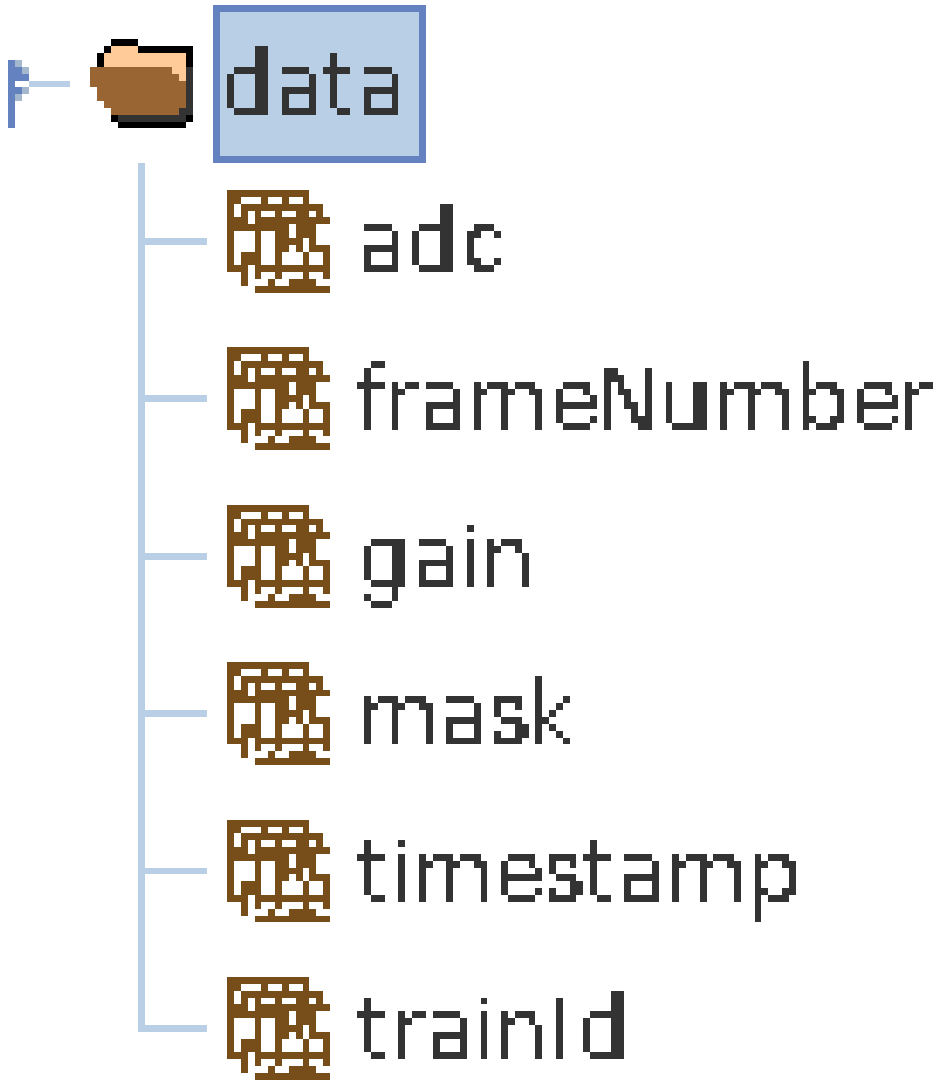

Fig. 4.4: The data structures that will be found in each hdf5 file, after calibration correction.

The notebook will create hdf5 files containing the corrected data using the currently available constants. In Fig. 4.4, the data structures that can be found in the hdf5 file containing the corrected data can be seen; they are similar to the data structures in the raw file, but with some differences:

- *adc* has the same shape of the data structure with the same name contained in the raw file (see *Raw data output*), but it now contains the corrected data, in keV units;
- *mask* is a new data structure and contains the map of the masked out pixels;

the remaining data structures are left unchanged.

CHAPTER 5

Troubleshooting

5.1 How to use Command Line Interface

The Karabo devices working with detectors developed at Paul Scherrer Institut make use of the control software developed there, called SLS Detector Software[1]. This software packages allowes to operate the detectors from Command Line Interface (CLI). For all the detectors installed at EuXFEL, it is always possible to use the CLI in parallel to the Karabo devices or in their place if the CONTROL device has issues. in order to do so, however, xctrl access to the Host running the CONTROL device is needed.

In order to run multiple CONTROL device instances on the same Host, every time a CONTROL device is instantiated, it reserves a different segment of the shared memory, and identifies it with a randomly generated DETECTOR_ID; to use the CLI in parallel to Karabo it is therefore necessary to know the last DETECTOR_ID generated. To do so one should look in the /dev/shm folder in the Host:

1. ssh as xctrl to the Host and run:

   ```
   source karabo/activate
   ```

2. go into the /dev/shm folder and run:

   ```
   ls -lrth
   ```

   You should see a list of entries like the ones in figure :numref:'label_dev_shm'. We are interested in the last entries of this list and in the multiple digit number before the word '_module': this is the DETECTOR_ID that is needed ('657808900' in the example). The second integer number is the MODULE_ID, in case of multi-module detectors.

At this point, the syntax to send a command is

```
sls_detector_put DETECTOR_ID-MODULE_ID: <command>
```

For example, if we want to stop the acquisition of the detector in figure Fig. 5.1 we will run:

[1] https://slsdetectorgroup.github.io/devdoc/

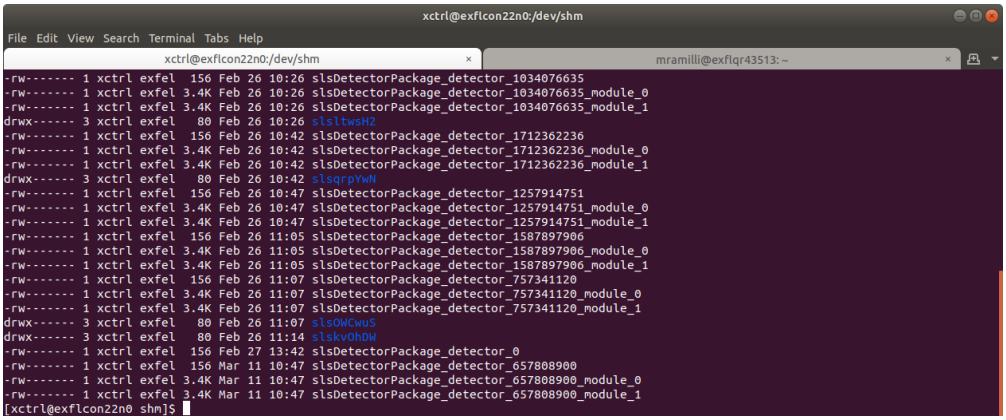

Fig. 5.1: Screenshot of the list of entries in the /dev/shm folder in a Host running slsDetector instances

```
sls_detector_put 657808900-0: stop
```

and for the slave:

```
sls_detector_put 657808900-1: stop
```

Similarly, to retrieve a parameter value:

```
sls_detector_get DETECTOR_ID-MODULE_ID: <parameter>
```

Continuing the above example, if we want to retrieve the number of frames set:

```
sls_detector_get 657808900-0: frames
```

Note: the CONTROL device clears the shared memory when it is shut down, so the above mentioned procedure is necessary only to work *in parallel* to the CONTROL (maybe to set commands not exposed in Karabo) or when the CONTROL device is in ERROR or not responsive, but *before shutting it down*.

5.2 How to update firmware and server

In order to upgrade the firmware version the CLI needs to be used. The slsDetectorSoftware client version has to be the same version as the one of the server running on the detector in need of an upgrade.

For server versions above v6.1.1 and for detector running a version of firmware and software mutually compatible, a description is given on PSI FW_Upgrade page. The description assumes however expert users.

A more detailed description is given here:

1. Once the client has been initialized, clean the shared memory with:

   ```
   sls_detector_get free
   ```

2. Set up the network:

   ```
   sls_detector_put hostname MODULE-HOSTNAME
   ```

3. Check the “Hardware version” (i.e. if the module is using a Jungfrau MCB v1.0 or v2.0) with:

```
sls_detector_get versions
```

This command should return a list of all the software and hardware parameters of the module under consideration. Check the "Hardware version" item of the list. It should either be "1.0" or "2.0". This is relevant, because different versions of the board need different versions of the firmware. Select the FW version according to the compatibility table on the SW_Releases page of PSI.

4. In order to update both server version and firmware, run:

```
sls_detector_put update jungfrauDetectorServervxxx xxx.pof
```

Otherwise, to just update the firmware, run:

```
sls_detector_put programfpga xxx.pof
```

where jungfrauDetectorServervxxx and xxx.pof indicated respectively the server file and the .pof file containing the firmware upgrade instructions, which should be located in the directory where the command is being launched. The procedure will last a few minutes, and progress will be communicated on the command line output.

5.2.1 Older client versions or incompatible SW and FW versions

In this case, a lot of the steps hidden in the `update` command must be done explicitly. The following instructions assume that the module has automatic server restart.

1. Copy the new the jungfrauDetectorServervxxx to the module:
   a. In the directory where the server is located, run:

```
python3 -m http.server
```

   b. telnet on the JUNGFRAU module Blackfin:

```
telnet MODULE-HOSTNAME
```

   c. From the module command line, copy the server:

```
wget http://HTTP-SERVER-HOST-NAME:8000/jungfrauDetectorServervxxx
```

2. From the Blackfin command line, set up the new server to run in update mode:
   a. Give the server run privileges:

```
chmod 775 jungfrauDetectorServervxxx
```

   b. Create a symbolic link:

```
ln -sf jungfrauDetectorServervxxx jungfrauDetectorServer
```

   c. Remove the automatic respawn of the server. Open the inittab file with, e.g.

```
vi /etc/inittab
```

   Afterwards, comment out the line `ttyS0::respawn:/jungfrauDetectorServer`

   d. Reboot the Blackfin chip by running the `reboot` command. This should kick you out of the module.
3. After reboot, telnet again on the detector Blackfin and start the server in update mode:

```
./jungfrauDetectorServer --update
```

4. From the client PC, clean the shared memory:

```
sls_detector_get free
```

5. Set up the network:

```
sls_detector_put hostname MODULE-HOSTNAME
```

6. Update the firmware:

```
sls_detector_put programfpga xxx.pof
```

7. Telnet again on the Blackfin and remove the comment on the `/etc/inittab` file that prevented automatic respawn; Afterwards `reboot` again the Blackfin.
8. After restart, remove the update mode:

```
sls_detector_put updatemode 0
```

5.3 Emergency shutdown

First of all, do not panic. Generally speaking, it is very difficult to permanently damage a JUNGFRAU module; examples of these kind of extreme cases may be:

- mechanical damage to the sensor (scratching the sensor, touching the wire bonds, etc.);
- elevate doses delivered to the sensor and/or the electronics (e.g. direct unattenuated XFEL beam);
- apply a voltage value outside specifications (e.g. low voltage higher than +12 V, or with inverted polarity);
- apply high voltage with air pressure around Paschen's Law minimum (for air, it around 0.01 mbar);
- prolonged operation without cooling in place can also be damaging.

In case of an emergency, when it can be necessary to put the detector in a safe state (e.g. a vacuum leak), one can simply di the following:

1. **power down the detector**, by powering off the LV; since the HV is generally delivered to the sensor via a voltage divider mounted on the JUNGFRAU board, this will also bring the HV down; if possible use your standard power off procedure, otherwise, *exceptionally*, it is also possible to power it off by simply removing the power plug from the power supply: it is in fact *not recommended* to do it by removing the green connector that is plugged directly into the JUNGFRAU board (see Fig. 1.4); if necessary, **disconnect the power supply** to prevent an uncontrolled power up without cooling;
2. **turn off the cooling**;
3. **mount back the cover plates** to protect the sensor.

When the emergency is recovered and the situation is back to normal operation, it should be possible to bring the module back to operation following the normal power up procedure.

Following, the most common problems encountered so far with the JUNGFRAU operation will be listed, and possible solutions presented. Anyhow, a few sanity checks should be performed as preliminary, to rule out simple mistakes:

1. check power;
2. check connectors, in particular the network (i.e. fiber and RJ45);

3. **check that the module is responding:**
    - ping it as described in *Control and operation*;
    - if necessary telnet on it to check that jungfrauDetectorServer is up and running.

5.4 Raw preview

5.4.1 No image on the preview and the RECEIVERS are not updating

If the Frame Rate In field of the RECEIVER is stuck to zero, it is probably a problem of network configuration.

1. Check that the module is up and running (ping it); if not sure, reboot it as explained below in *Control and operation*;
2. check that the fiber is connected and that there's signal (LEDs below the interface are both shining blue);
3. check that the IP addresses are correctly configured;
4. if all the above points are satisfied, probably ITDM or Controls on OCD must be called; possible further misconfigurations are:
    - the RECEIVER interfaces on the host PC must be configured so that MTU = 9000;
    - the UDP Socket buffer size must be = 2000 * 1024 * 1024 = 2097152000;
    - if both conditions are met, this is a problem for experts

5.4.2 I can see the RECEIVERS updating but no image on the preview

If the frame rate of the receiver updates, it should be receiving data; if no image is seen on the preview, try opening the corresponding RECEIVER device and setting 'Online display enable' as 'True' (see Fig. 5.2)

5.4.3 There are striped artifacts in the image

It has been reported the presence of striped artifacts in the online preview. The shape of the stripe indicates different problems.

1. The stripes are horizontal, i.e. they manifest on the raw image as bands of uniform value equal to zero *across* all the columns (see Fig. 5.3); this effect indicates packet loss: some of the UDP packets sent out by the JUNGFRAU module(s) are not collected by the RECEIVER device; this is probably due to some misconfiguration of the network and Controls should be called.
2. The stripes are vertical, i.e. they manifest themselves *along* the columns, may or may not stop at half module, and they are overlaid on the signal. First, stop the acquisition and check if your module triggered a *Temperature Event* (see *Temperature Control* for details). If this is not the case, check the raw data of your dark

    runs; if the artifacts are in there as well and their structure looks similar to what shown in

    Fig. 5.4, the problem is with the FPGA in the JUNGFRAU board. Power cycling the module has been reported as a solution to this issue.

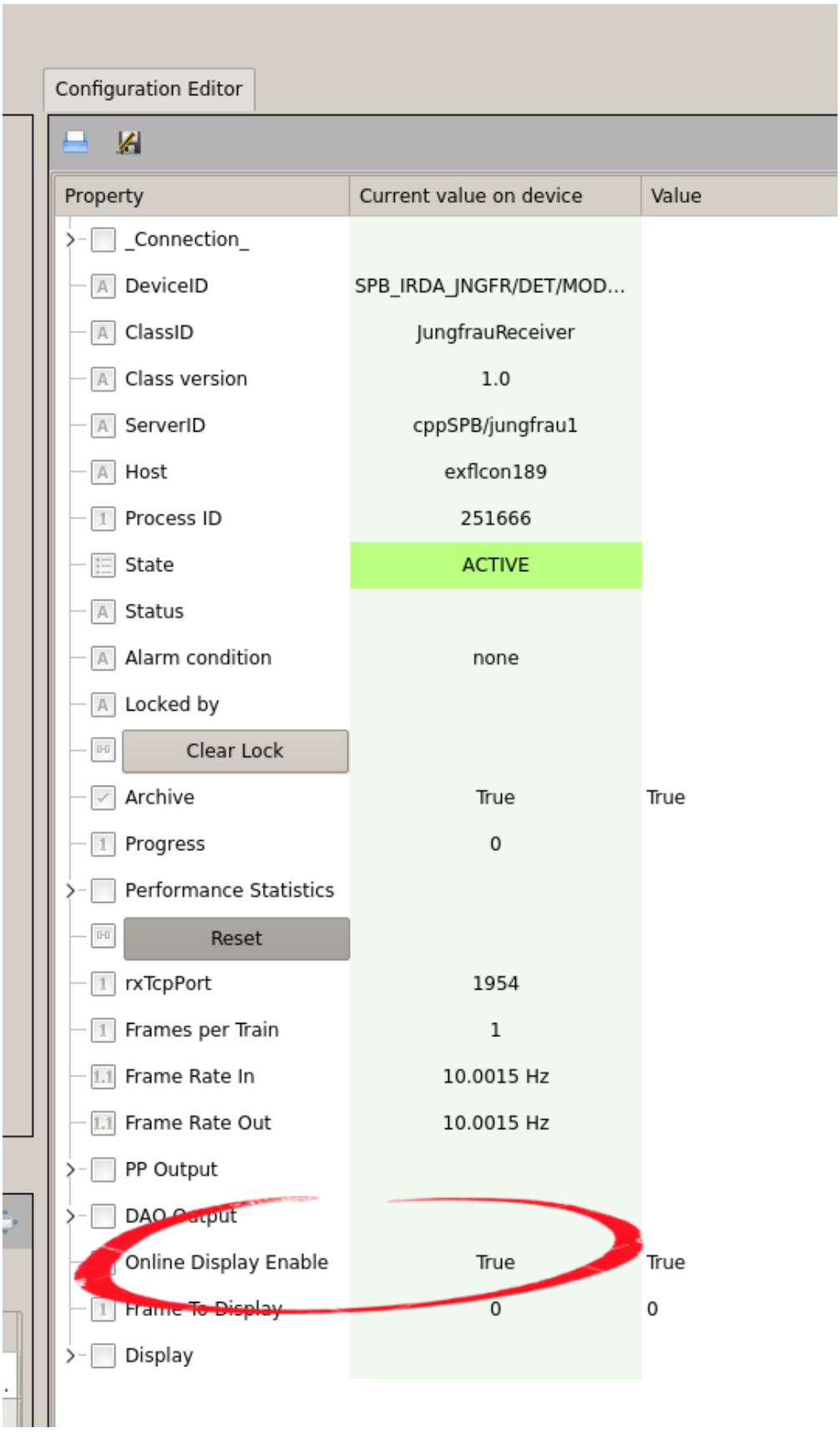

Fig. 5.2: The setting to be marked as 'true' to allow online display

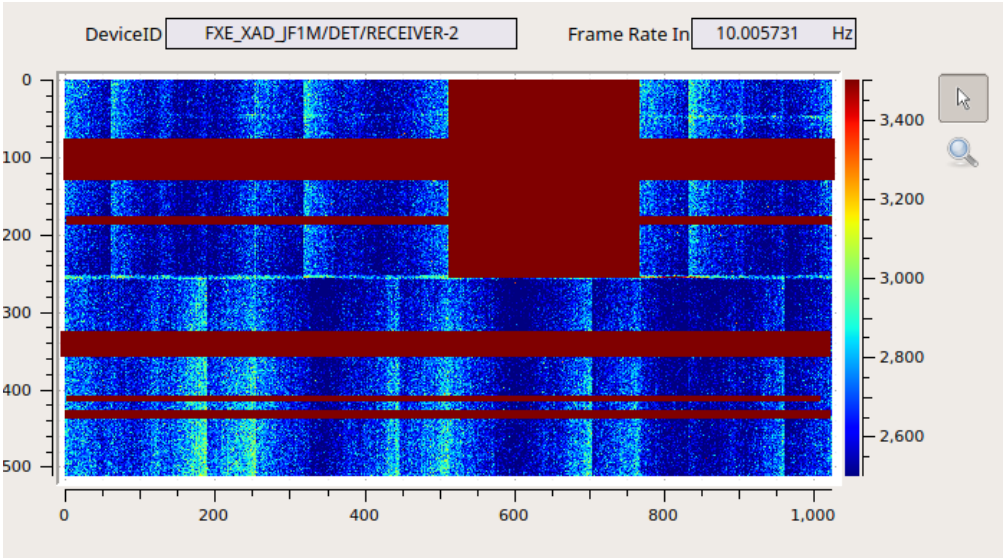

Fig. 5.3: Example of raw output indicating incomplete frames, i.e. with UDP packet loss

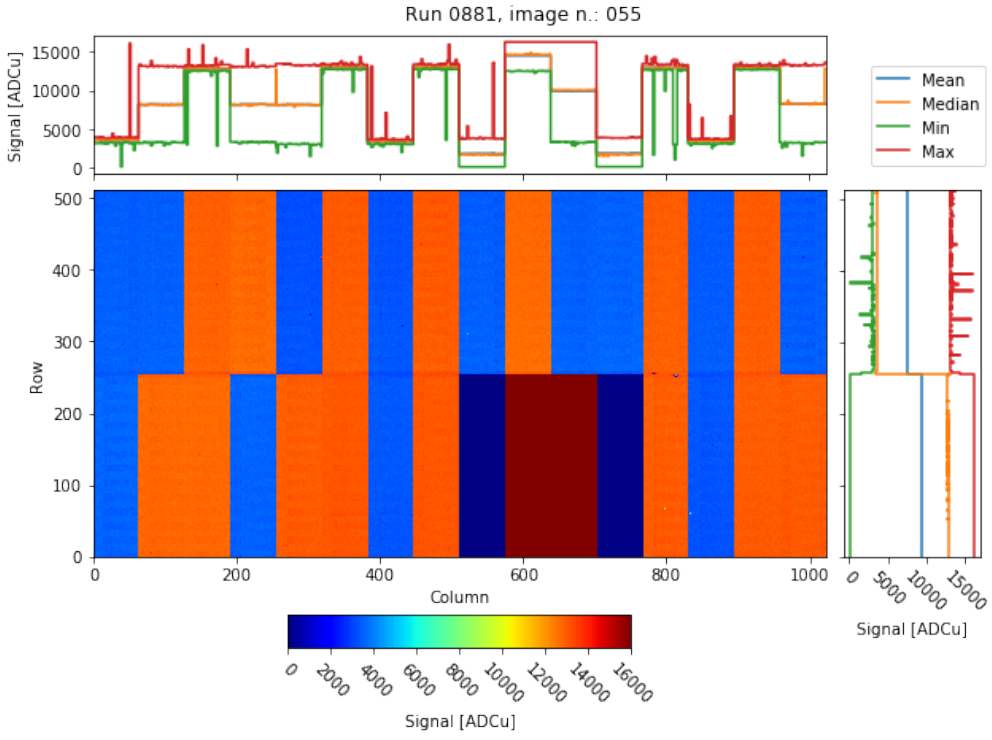

Fig. 5.4: Raw output when the FPGA on the JUNGFRAU board is not behaving correctly

5.4.4 Large parts of the detector have a baseline too high

The RAW image output of the detector display large non-uniformities, in which the baseline of the detector is too high of several hundreds if not thousands ADC units, which look like large spots on the detector surface, similar to what is shown in Fig. 5.5. In this case, it is very likely that the detector is not properly cooled, and the non-uniformities are simply the effect of the sensor generating higher leakege current due to inefficient heat dissipation: power down the detector and check the cooling; if it off or not working properly (e.g. set to a temperature too elevated) bring it back to normal, let it run for a few minutes and then power back the detector.

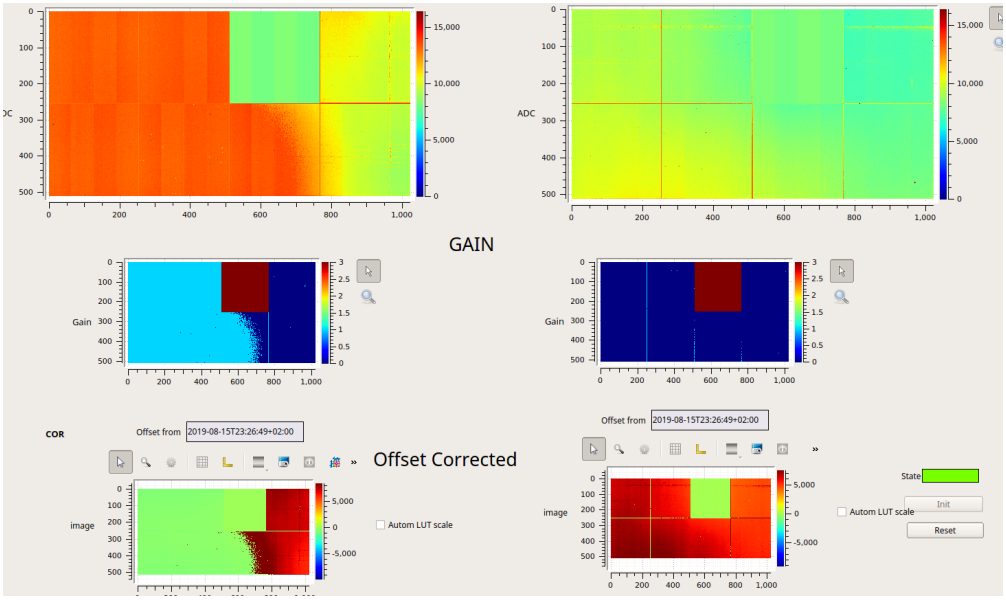

Fig. 5.5: Example of overheated JUNGFRAU modules: probably due to a combination of long exposure time and cooling failure, the leakage current alone is enough to bring some pixels to G1.

5.5 Control and operation

5.5.1 My CONTROL or RECEIVER device is in ERROR

There is an order in which these devices must be instantiated. To recover, try:

1. shut down all the devices (or even the server device, for good measure);
2. instantiate *all* the RECEIVER devices first;
3. once the RECEIVERS are up, instantiate the CONTROL device.

I have instantiated the devices in the correct order, but devices ar still in ERROR

At this point, it is probably necessary to restart the jungfrauDetectorServer running on each module. Shut down all the JUNGFRAU devices and proceed with the restart of the jungfrauDetectorServers.

Suggested method:

1. a list of the microcontroller aliases can be found in the detectorHostName list in the CONTROL device (see Fig. 5.7); connect via telnet the microcontroller on each JUNGFRAU module using that aliases, e.g.: from command line type:

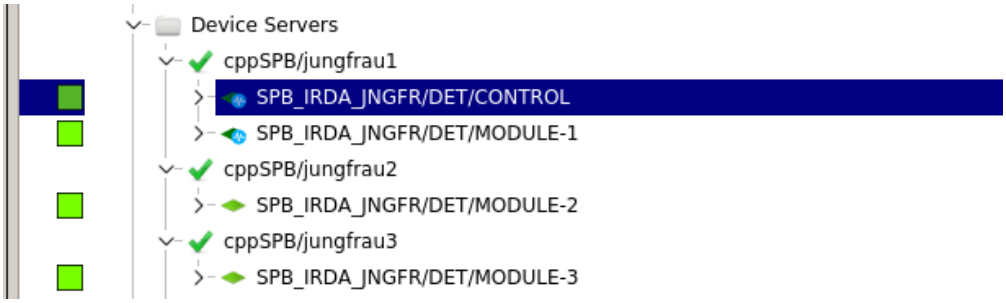

Fig. 5.6: The device servers for the operation of three modules at SPB: after shutting them down, the three RECEIVER devices (here named 'MODULE') must be re-instantiated, before instantiating the CONTROL device

```
telnet spb-irda-jngfr-det-control-1
```

(see Fig. 5.8);

2. once on the module, from the command line, launch the *reboot* command (see Fig. 5.9); this will reboot the embedded LINUX OS on the microcontroller and the jungfrauDetectorServer will automatically respawn;
3. repeat this for each module.

Fig. 5.7: The list detectorHostname contains the aliases of the microcontrollers on each JUNGFRAU board

Fig. 5.8: Example of telnet use to connect to a microcontroller

Not suggested method:

it is possible to obtain the same result by simply power cycling the modules; however, **frequent power cycles** are obviously **not recommended**. It is strongly suggested to use this option only if the individual reboot is not possible.

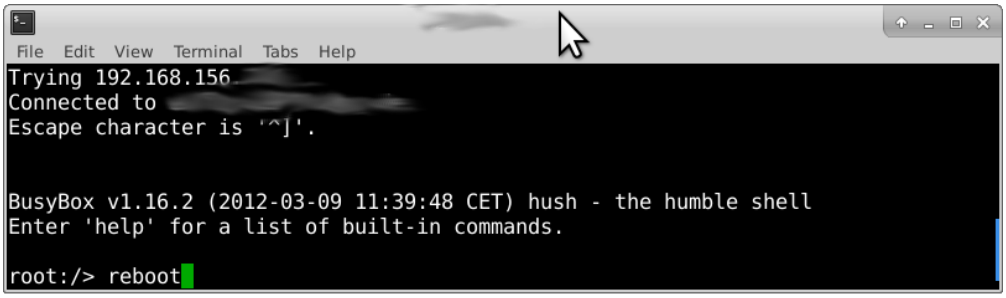

Fig. 5.9: After connection to the microcontroller, launch the reboot

After the reboot of the jungfrauDetectorServers (you can ping the module microcontrollers to check if they are up again, see Fig. 5.10), re-instantiate the devices in the correct order.

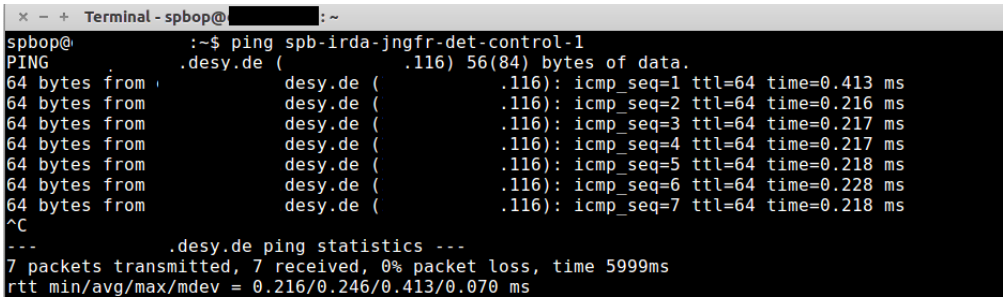

Fig. 5.10: Ping the module to see if the reboot has been completed

Despite reboot of the modules and correct instantiation of the devices, they are still in ERROR

If this is the case, it may be necessary a reboot of the physical host where the server devices are running; if you have the permits to do so (i.e. to ssh to that machine and start a reboot), and feel confident in doing it, go ahead and do it; after it is done, reboot the controllers on the JUNGFRAU modules as explained above and then re-instantiate the JUNGFRAU devices in the correct order. Otherwise, Controls OCD needs to be called.

5.5.2 External trigger acquisition does not work

This is under the assumption that it has been verified that acquisition in autotrigger mode is instead working.

Check if LED near the LEMO connector (see Fig. 1.4) is flashing:

- if it's not flashing:
  1. check with a scope that input signal is actually a valid trigger: TTL, positive, at least 100 ns;
  2. access the module and check that the flat cable is correctly connected both to the trigger board and the JUNGFRAU board;
  3. if both conditions are satisfied, try to replace the trigger board or leave it to the experts;
- if the LED is flashing and still acquisition does not work, try a reboot and restart as explained in *My CONTROL or RECEIVER device is in ERROR*; if this does not work either, it is a problem for experts.