Gotthard-II Documentation Documentation

Release 0.0

M. Ramilli

Aug 16, 2024

Contents

Contents:

CHAPTER 1

Network setup

This section will illustrate the generalities on how to configure the network between the control machine, the Gotthard-II (GH2) module(s) and the receiver server.

1.1 Basic components

The control and operation of Gotthard-II 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 `label_sls_arch` 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 (gotthard2DetectorServer);
3. one or two receiver devices, collecting the data sent out of the module in form of UDP packets: the Gotthard-II 50 μm employs only one receiver, while the 25 μm version has two.

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.

1.2 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/

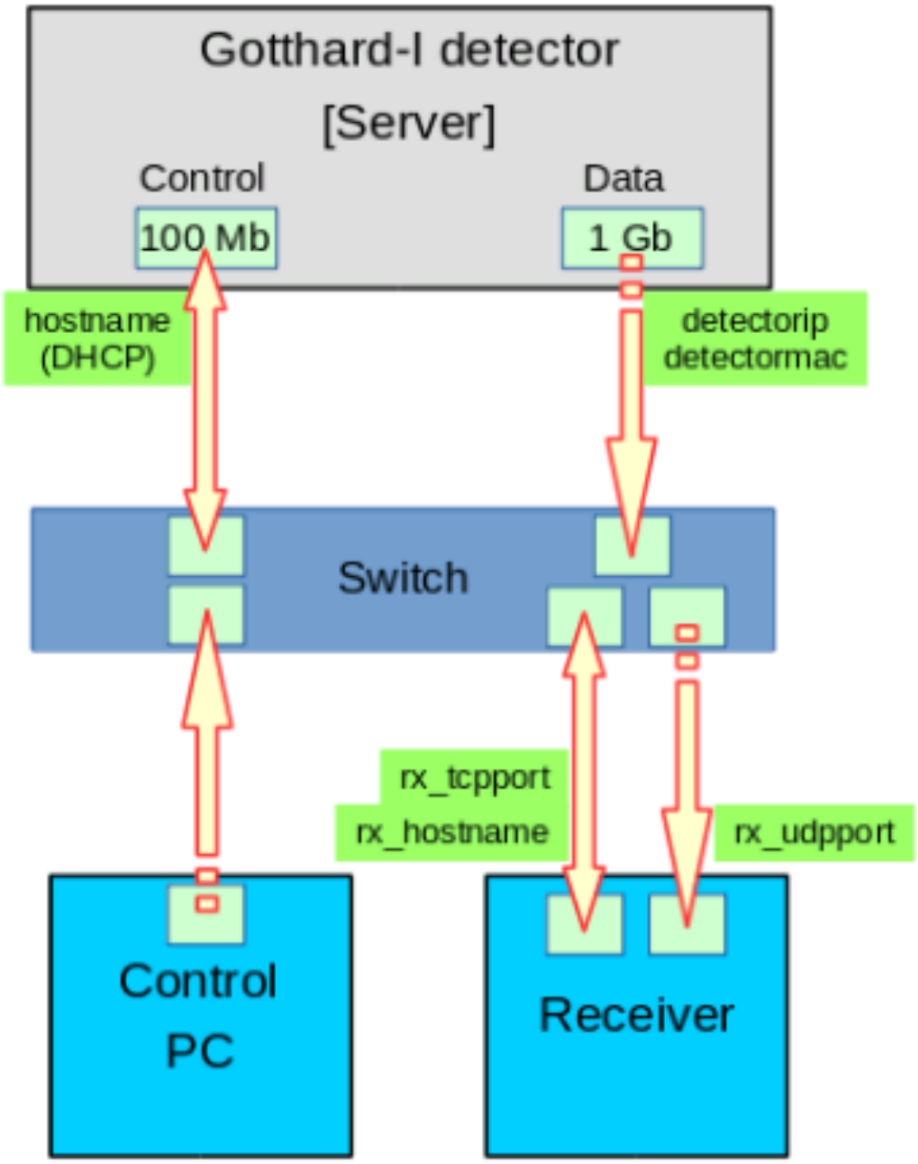

Fig. 1: Schema of the architecture of the SLS Detector Software Client-Server-Receiver network. The picture is taken from the "Quick guide to the SLS Detectors Package" by PSI Detector Group.

1. a 100 Mb port for control on the GH2 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 GH2 module (indicated as 1 Gb on the picture);
3. the port on the receiving server, to which the gotthard2DetectorServer will send the UDP packets to be collected by the RECEIVER device.

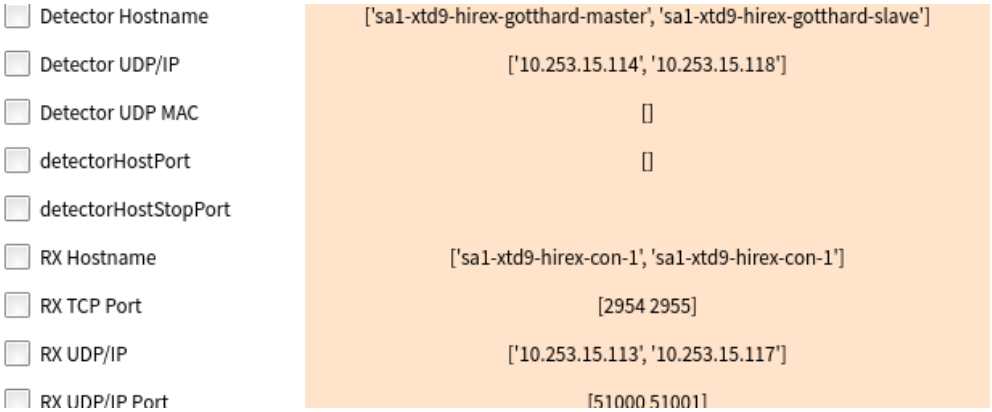

Fig. 2: Detail of a GH2 25 μm CONTROL device

This procedure has to be done twice for the GH2 25 μm , because the FEM is read out by two boards. To configure these parameters, one must access the CONTROL device before instantiation.

- *Detector Hostname* configures the control port on the GH2 module; at EuXFEL this is usually accomplished by inserting here the alias corresponding to the the GH2 Readout 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 GH2 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. `label_cont_conf`).

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 `label_rec_conf`);
- go the CONTROL device and fill in the *RX TCP Port* list with the values assigned to each RECEIVER (see e.g. `label_cont_conf`).

Configuring the UDP port value is useful only if there is 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.

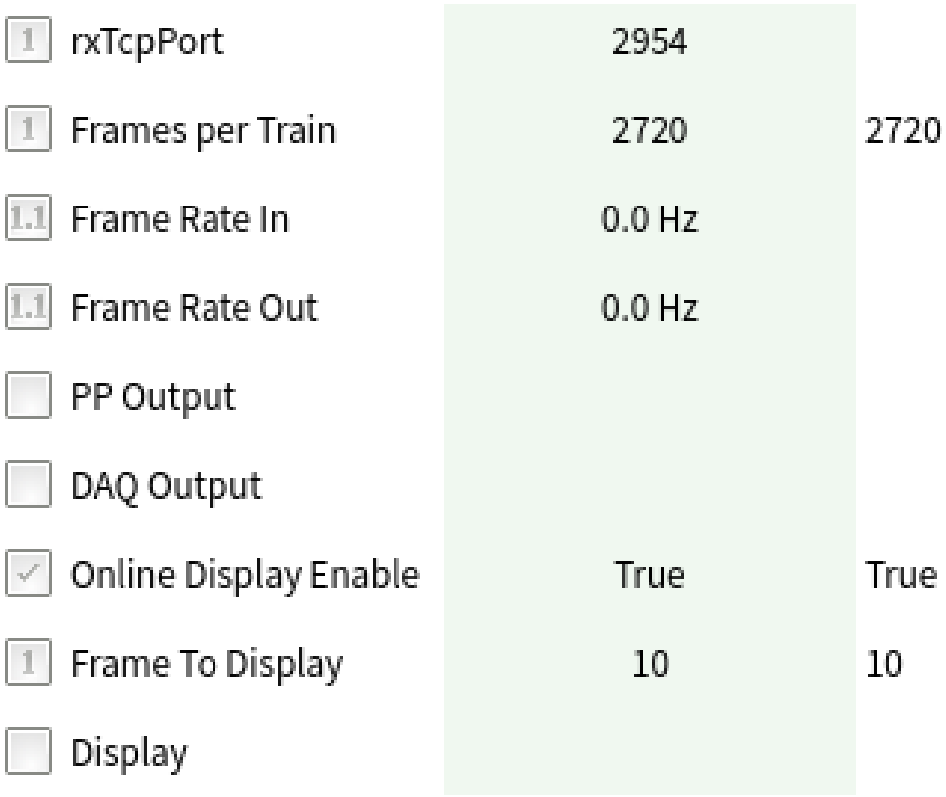

Fig. 3: Detail of a GH2 RECEIVER device, showing the *RX TCP Port* field

1.3 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 versions
```

  you should get the list of all the versions, including the software and firmware one, this last one may be formatted as a hexadecimal like: *0x171113*; this number is the date of release of the firmware, which identifies it (in the example corresponds to 13.11.2017); to use the CLI one has to follow the same procedure described in **'Stopping the acquisition from command line'_**;

- **from the CONTROL device**: there is a field in the CONTROL device that checks the firmware version (see `label_cont_detv`); the release date of the firmware can be seen there, formatted as hexadecimal.

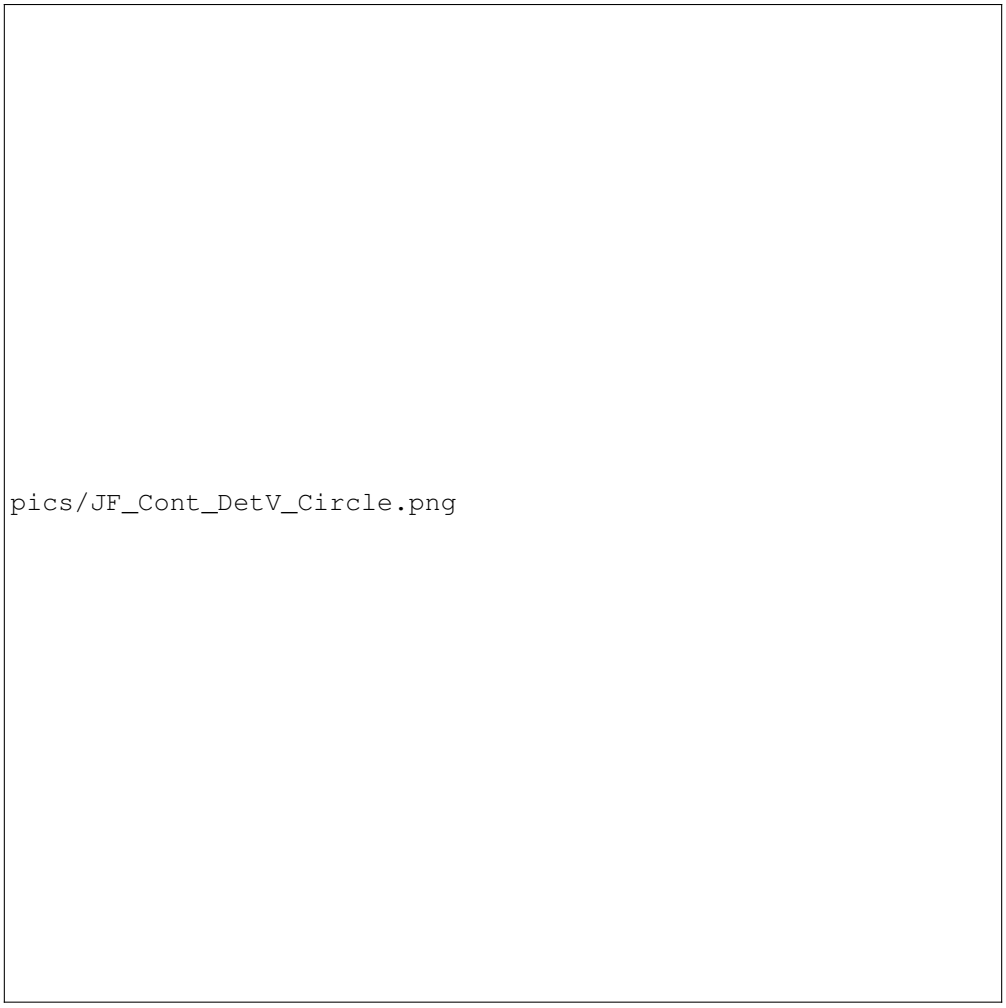

Fig. 4: Detail of a JUNGFRAU CONTROL, highlighting the *Detector Version* field

CHAPTER 2

Operation

This section highlights the basic principle of operation of a Gotthard-II (GH2) module.

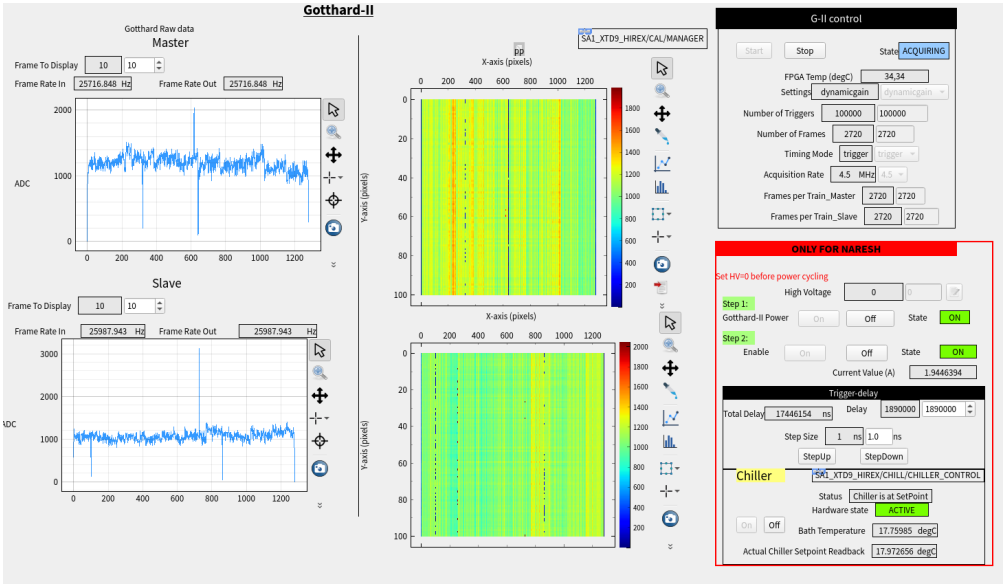

Fig. 1: Temporary scene controlling the operation of the GH2 25 um installed in the SASE1 HIREX spectrometer.

2.1 Turning on and off

2.1.1 Cooling

Each module of GH2 needs cooling to dissipate the heat generated by the FEM and the GH2 readout board(s), for a total of about 12 W (for the 50 μm model) or 24 W (for the 25 μm one). 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, the 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 dangerous for the module**.

2.1.2 Power

In order to be fully operational, the GH2 detector needs:

1. a +12 V LV signal, to power the readout and control electronics and the FEM ASICs;
2. an *enable* signal whose value must be at least +1.5 V and not higher than +12 V; this signal turns the electronics on;
3. a +200 V HV signal to bias the sensor; this can be provided either by an external power supply, or with the HV module mounted on the Main Board. In this second case, the HV is set via the *High Voltage* property of the CONTROL device.

When powering a GH2 module it is important that the correct power procedure is being followed, otherwise the safety of the detector may be compromised.

Power Up

1. apply the +12 V LV;
2. only after, apply the *enable* signal: the LV needs to be already on the pin, to allow the electronics to power on in its correct state; after the *enable* signal is given, the detector current will start to increase: when it reaches ~1 A for the 50 μm , or roughly double for 25 μm , it is a sign that the power up is completed;
3. **only at this point** the HV can be applied; it is however strongly suggested to first push some data from the detector, to make sure that the ASICs powered up correctly.

Power down

An incorrect power down is extremely dangerous for the detector: the reason is that the filtering capacitors between the sensor and the ASICs have a long discharge time. If all the power is removed simultaneously, the ASICs will power down faster than the sensor and an uncontrolled voltage at the input node is dangerous for the pixel safety. Therefore the following procedure must be followed:

1. remove the HV; if using the built-in HV module, set *High Voltage* to zero, then wait for 10 - 15 seconds;
2. remove the enable; at this point the firmware will start the power down of the electronics, starting from the ASICs, and it will be completed in few seconds (current will rapidly drop to a value close to zero);
3. removing the LV is not necessary, it is however suggested to do so to put the detector in safe condition, when it will not be used for long periods of time (e.g. it is not necessary to remove it while power cycling).

Power supplies

There are currently three ways to provide LV and enable to a GH2 device at EuXFEL:

1. via power chord; a specially adapted power chord with a power bank that transform the socket current in +12 V DC with a CC limit of at least 3 A can be used. In this case, since LV and *enable* would be supplied simultaneously at the respective pins, the *enable* must be excluded by flipping the switch in the back of the module in the *OFF* position: only afterwards, the power chord can be connected to the socket; a LED in the back will start to pulse green light at roughly 1 Hz; after that, the switch can be moved to the *ON* position, sending the enable signal to the electronics. The usage of this method is recommended only as a temporary solution;
2. via MPODs; two MPOD channels can be used to provide LV and *enable*; a dedicated version of the powerProcedures middlelayer exists (class ID Gotthard2Power) which follows the power procedures and performs checks on the state of the *High Voltage* key;
3. via an EuXFEL custom made power circuit coupled to LV PLC module; the module has a built-in continuity group useful in case of sudden power cuts, and the circuit will take care of automatically removing the *enable* in such occurrence, thus triggering the emergency power down procedure implemented in the detector firmware. However, the Karabo implementation of the control devices for such power supply has not been centralized so it varies from instrument to instrument (an example is shown in `label_hirex`); morover, the procedure and the safety checks are so far left to the operator.

2.2 Control configuration

The Gotthard2 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.

2.2.1 Basic operation settings: CONTROL device

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

- *Settings*: this entry sets the operation mode for the modules:
  - *dynamicgain*: standard dynamic gain switching operation;
  - *fixedg1*: fixed gain operation with medium gain feedback capacitance;
  - *fixedg2*: fixed gain operation with low gain feedback capacitance;
- *Number of Frames*: number of images acquired per train; it can be set to any value between 1 and 2720; this value has to match the *Frames per Train* value in the RECEIVER device;
- *Number of Triggers*: number of trains that the detector will acquire before ending the acquisition;
- *Reverse Slave Read-out Mode*: should be set to false for the 50 μm version and True for the 25 μm one;

There are additional parameters that can be configured, but whose operation is at the moment not supported:

- *Optimize for single photon*: (default: False) a boolean that if True, configures the detector with settings optimized for single photon resolution; dedicated gain calibration needs to be performed in order to properly correct raw data acquired in this mode;
- *Acquisition Rate*: (default: 4.5 MHz), changes the frame rate of the detector by changing the clock divider (and therefore also changing the exposure time). The only other available option is 1.1 MHz, although calibration constants are not fully available and results are in arbitrary units.

Finally, from the CONTROL device there are some parameters accessible, but they should be left to their default value:

- *Additional Exposure Time*: (default: 0) further increases the detector exposure time from the minimum of ~110 ns; useful only in CW mode;
- *Additional Exposure Period*: (default: 0)) further increases the detector exposure period from the minimum of ~222 ns; useful only in CW mode;
- *Timing Mode*: (default: 'trigger') changes the operation from external trigger (setting = 'trigger') to auto trigger (setting = 'auto');
- *Timing Source*: (default: external) changes the source of the acquisition trigger from external to internal and vice versa;
- *Burst mode*: (default: 'burst_internal'), changes the operation mode of the detector to CW or burst with external timing.

2.2.2 Basic operation settings: RECEIVER device

The RECEIVER devices have some operational parameters that need to be configured, especially when changing the number of images acquired per train:

- *Frames Per Train*: 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 the value of *Number of Frames* in the CONTROL device;
- *Online Display Enable*: has to be set to True to allow display of raw data directly from the RECEIVER.

CHAPTER 3

Troubleshooting

3.1 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 `label_dev_shm` we will run:

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

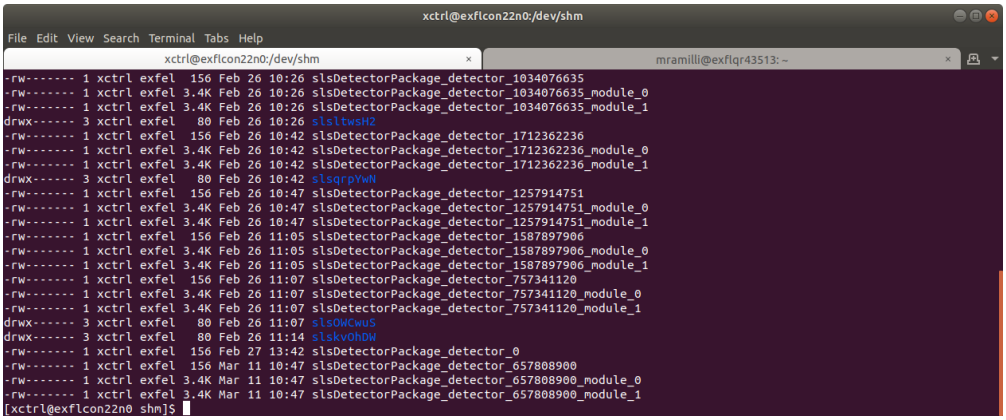

Fig. 1: Screenshot of the list of entries in the /dev/shm folder in a Host running slsDetector instances

```
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*.

3.2 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 of the same version as the server running on the detector in need of an upgrade.

For server versions above v6.1.1 and for a 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. In order to update both server version and firmware, run:

```
sls_detector_put update gotthard2DetectorServervxxx xxx.rbf
```

Otherwise, to just update the firmware, run:

```
sls_detector_put programfpga xxx.rbf
```

where gotthard2DetectorServervxxx and xxx.rbf indicate respectively the server file and the .rbf 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.

If the detector is a 25 μm version, the update must be performed for both half-modules.

3.2.1 Older client versions or incompatible SW and FW versions

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

1. Copy the the gotthard2DetectorServervxxx to the module, via:

```
scp gotthard2DetectorServervxxx root@MODULE-HOSTNAME:~/
```

2. log in into the module via:

```
ssh root@MODULE-HOSTNAME
```

Subsequently, from there:

a. Give the server run privileges:

```
chmod 775 gotthard2DetectorServervxxx
```

b. Create a symbolic link:

```
ln -sf gotthard2DetectorServervxxx gotthard2DetectorServer
```

c. Create in the root directory a file named 'update.txt'; this will allow the server to start in update mode at the following start, ignoring the checksums that verify software and firmware compatibility.

3. Power cycle the detector.
4. Now, from the client host, clean the shared memory:

```
sls_detector_get free
```

5. After restart, set up the network:

```
sls_detector_put hostname MODULE-HOSTNAME
```

6. For safety reasons, power off the chips:

```
sls_detector_put powerchip 0
```

7. Update the firmware:

```
sls_detector_put programfpga xxx.rbf
```

8. Remove the update mode:

```
sls_detector_put updatemode 0
```

3.3 Karabo CONTROL device down or stuck

If the Karabo CONTROL device goes down or it is somehow not responsive while the detector is in ACQUIRE state, shutting it down and trying to instantiate it again will result in a failure to connect, because the TCP port where all the instructions from server to RECEIVER are sent to is busy with data transfer. To properly restart the device, it is therefore necessary to stop the acquisition before instantiating the CONTROL device again.

There are three possible methods, which are listed below.

3.3.1 Killing the gotthard2DetectorServer

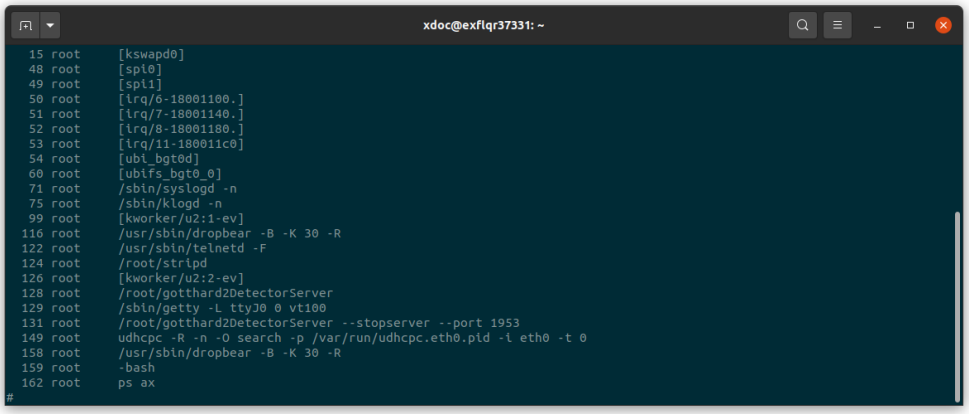

Fig. 2: Screenshot of the list of processes running on a healty Gotthard 2 detector

This method is usable by anyone who has access to the control network. It consists of killing the server running on the detector, in this way interrupting any operation currently ongoing. To do this, after shutting down the CONTROL device, follow these steps:

1. In the Karabo CONTROL device under the key 'Detector Hostname' the alias(es) of the readout board(s) are listed: one name for a 50 μm device, two names for a 25 μm model, e.g. 'hostname1', 'hostname2';
2. ssh onto each one of these devices with:

   ```
   ssh root@hostname1
   ```

3. run:

   ```
   killall gotthard2DetectorServer
   ```

This will kill all the gotthard2DetectorServer processes, but since they are configured for automatic respawn, they should restart immediately. Run:

```
ps ax
```

to check if the processes are running again; in `label_ps_ax` an example of a healty detector module is shown. After this has been done on all the relevant boards, the CONTROL device can be instantiated again.

3.3.2 Stopping the acquisition from command line

This is the cleanest and safest method, but it is the most elaborate and it requires xctrl access to the host where the Karabo CONTROL device was running. You can see its name in the 'Host' key in the device. **Before shutting down the CONTROL** device:

1. connect to the Host and find the DETECTOR_ID as explained in *How to use Command Line Interface*;
2. initialize the local Karabo environment if not done already;
3. from command line run:

```
sls_detector_put DETECTOR_ID-0: stop
sls_detector_put DETECTOR_ID-0: rx_stop
sls_detector_put DETECTOR_ID-0: clearbusy
sls_detector_put DETECTOR_ID-0: highvoltage 0
```

   Repeat this for the slave module by replacing '-0' with '-1' in the commands above, if there are two modules, i.e. if the detector is a 25 μm model.

Afterwards, the CONTROL device can be shut down and instantiated again.

3.3.3 Last resort

If the two above mentioned methods fail, the last resort option is to power cycle the detector by removing the *enable*. The firmware currently installed *should* power down the device in a safe way (provided the LV will be up long enough, i.e. 15 to 30 s), that is removing the HV first and only afterwards powering down the ASICs and the electronics. The safety of this procedure however, has not been thoroughly tested yet, and it is recommended to use it as a *last resort* and not as regular practice.

3.4 Emergency shutdown

In order to put the detector into a safe mode, it is useful to power it down. In this case, it is important that the power down procedure is performed correctly: **removing the LV before or while also powering down the sensor is dangerous for the ASIC and should never be performed**. As stated in **'Power'_** section the correct power down procedure is:

1. remove the HV; if using the built-in HV module, set *High Voltage* to zero, then wait for 10 - 15 seconds;
2. remove the enable; at this point the firmware will start the power down of the electronics, starting from the ASICs, and it will we completed in few seconds (current will rapidly drop to a value close to zero);
3. removing the LV is not necessary, it is however suggested to do so to put the detector in safe condition, when it will not be used for long periods of time (e.g. it is not necessary to remove it while power cycling).

After the detector is correctly powered down, it may be useful to put back the protective cover on it. It is also safe to disconnect the detector from the LV if there is the possibility of an uncontrolled delivery of LV and *enable*.

3.4.1 Power glitches

Power glitches that compromise the stability of the LV supply, can trigger an unsafe power down, by removing LV while the HV is still on. To avoid this, the firmware is equipped with a *safe power down procedure*, which, in case the *enable* signal is suddenly removed, powers down HV, ASICs and electronics in this order. For this it is important that the LV (and the LV only, not the *enable*) is connected to a UPS system that will allow at least 30 s of supply in case of emergency:

- for the custom power supply built internally at EuXFEL this is already the case;
- for the MPODs this is foreseen, but not yet implemented;
- for the power chord this is of course not implemented at all.

In any case, it is important to make sure that when power is back the detector does not get powered in an uncontrolled manner. To do this:

1. after the safe power down, shut down the CONTROL and RECEIVER devices;
2. check that the HV is set to zero; if power is provided through the built-in power module, make sure, when the Karabo CONTROL device is available again, that its property *High Voltage* is set equal to zero;
3. check that LV and *enable* signal are not accidentally delivered in an uncontrolled way:
    - while using the power chord, switch the button on the OFF position;
    - while using Karabo-controlled power supplies, when the Karabo devices are available again, make sure that the LV and *enable* channels are OFF;
4. instantiate the CONTROL and RECEIVER device(s), CONTROL device should go into UNKNOWN state;
5. when ready, follow the power up procedure detailed in **'Power'_**.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search