SLS Detectors
Frequently Asked Questions

Anna Bergamaschi

August 22, 2017

Contents

Chapter 1

SLS Detectors Software

1.1 Which programs can I use to control my detector?

The complete software package is composed of several programs which can be installed (or locally compiled) depending on the needs:

- The **slsDetector shared and static libraries** which are necessary for all user interfaces.
  The class slsDetectorUsers can be used as API from your acquisition software (see separate documentation).
- The **command line interfaces (sls_detector_put, sls_detector_get, sls_detector_acquire, sls_detector_help)**, which are provided to communicate with the detectors using the command line and eventually to the data receiver
- The **data receiver (slsReceiver)**, which can be run on a different machine, receives the data from the detector and interfaces to the control software via TCP/IP for defining e.g. the file name, output path and return status and progress of the acquisition
- The **graphical user interface (slsDetectorGUI)** which provides a user friendly way of operating the detectors with online data preview
- The **calibration wizards (energyCalibrationWizard, angularCalibrationWizard)** to analyze the data and produce the energy or angular calibration files

1.2 How can I control many detectors in parallel or independently?

For most users the detector will be composed by a single module. Therefore all configurations of the detector will refere to that single entity.

However, for some experiments it is necessary to concatenate the data from several detector controllers, and sometimes (e.g. MYTHEN) each controller can control many modules. This should be transparent to the user since most parameters will be identical for all controllers (e.g. exposure time, energy threshold etc.), except for the configurations specific to the controller (e.g. hardware configuration).
In principle it is possible to combine controllers of different type (e.g. MYTHEN, GOTTHARD, EIGER) but the user should then evaluate if it really makes sense to control such different systems in parallel.

In other cases, several SLS detectors will independently acquire data during the same experiment. In this case it will be necessary to be able to seperately control them.

The detectors can be controlled in parallel from several PCs (clients). However it is important the the configurations match on all of the them such that no conflict arise. Eventually a detector can be locked to a specific control PC, still different users interfaces (command line, GUI) can be used in parallel.

A sketch of a possible complex detector configuration is shown in figure 1.1

For this reason and index is assigned to each detector. If a single detector is used, as in most cases, the index will be omitted and defaults to 0.
To control the other detectors the index cannot be omitted!

An index will also be assigned to each controller within a detector. However the user normally will not need to address single controllers, except for the most advanced settings which can be left to configuration files.

Finally each module within a controller has an internal index. However in general it is not required that the user is aware of the system architecture and, if needed (rarely), the modules can simply be addressed sequentially starting from controller 0.

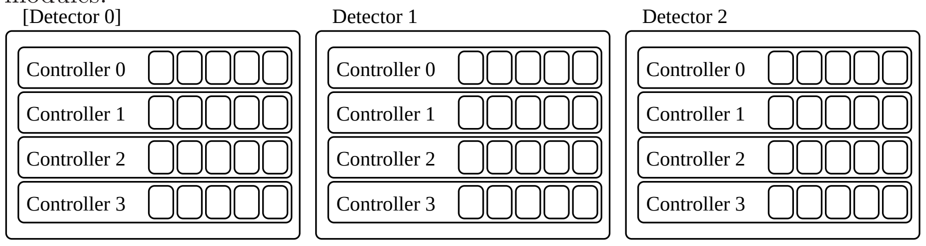
1.2.1 Examples

For MYTHEN, if one needs to control 6 modules, the system can either be composed by and MCS6 with 6 modules (1 detector, 1 controller, 6 modules), or by 6 MCS1 (1 detector, 6 controller, 1 module each). After apppropriate configuration of the system, the interface to the user will be the same for both systems.

For GOTTHARD, one module corresponds to one controller. A detector will have the smae number of controllers and modules.

For EIGER, one module consists in two controllers. Fo a multi-module

Figure 1.1: Scketch of a possible complex system architecture composed of several detector, each consisting in many controllers eventually controlling several modules.



system, the number of controllers will increase accordingly, but should be left to a configuration file.

You will need to configure more than one detector, only in case you want to operate several detectors independently.

1.3 How can I configure the data receiver?

For slower acquisitions, the detector will return the data to the control PC over TCP/IP (e.g. MYTHEN).

However, for faster frame rates (e.g. GOTTHARD, EIGER) the controllers will return the data to a data receiver i.e. a process specifically designed to receive the data from the controller over a GBit network and save them to disk. The data receiver can run on any machine (e.g. a file server) accessible by both the control PC and the detector controller, as sketched in figure 1.2. A data receiver process must be configured for each controller. Normally, to avoid performance loss it is better if different data receivers run on different machines.

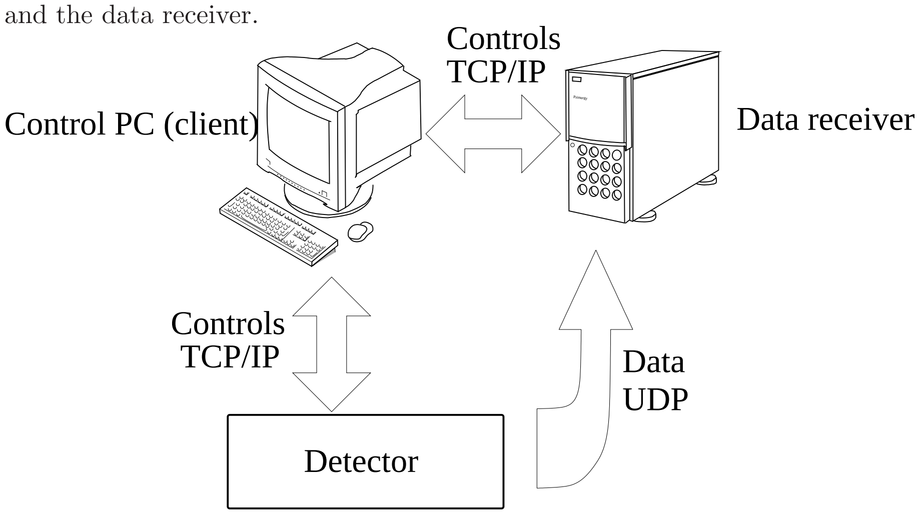
To setup the system, you should configure:

Client-Detector TCP/IP connection i.e. for each controller hostname or IP address (*hostname*) and communication port (*port*, use default).

Client-Receiver TCP/IP connection i.e. hostname or IP address of the data receiver (*rx_hostname*) and communication port (textitrx_tcpport, use default).

Detector-Receiver UDP connection i.e. for each controller IP address of the receiver network interface (*rx_udpip*) and communication port (*rx_udpport*) used for receiveing the data. By detfault the IP address of the TCP/IP receiver interface will be used also for the UDP conenction. Editing the UDP network interfaces and ports is useful if several controller are sending data to a single receiver (not reccomended to avoid performance loss).
A MAC (*detectormac*) and IP address (*detectorudpip*) should also be assigned to the controller network interface used for the UDP communi-

Figure 1.2: Scketch of the communication between the control PC, the detector and the data receiver.



cation, but the default values can normally be used unless firewalls are defined between the detectors and the receiver.

All these configurations are normally left to the configuration file and should not be changed dynamically by the user.

After starting the data receiver process and correctly configuring the client and the detector, this architecture should be completely transparent for the user, except that the output file path must be properly configured from the client for the data receiver machine (easiest is that the disk is mounted for both machines in the same location).
The client will take care of communicating with the data receiver and the detector. A feedback about the progress of the acquisition and a preview of the data being acquired can also be obtained by the client from the data receiver.

1.4 What are settings and calibration files for?

The analog characteristics of the detector have to be initialized in order to define the noise and the dynamic range which need to be used for the measurements. These parameters have a different meaning for analog or digital detectors, but in both cases some predefined voltage levels and current (we call them *settings*) must be laoded to the detector. Moreover, there are some parameters that are custom to single detectors or modules (e.g. the trimbits). All these settings are stored in some settings file, which are organized in a *settingsdir* with a definite architecture, where the software will look for the files to load to the detector whaen changing its settings.

In addition to that, in a single photon counting detector the threshold is

set as a voltage level for the comparator, but for the user it is useful to have a direct conversion to the energy level. For this, after a proper calibration of the detector (see specific documentation) calibration file are generated in order to convert threshold in volts to keV. Also in this case the directory *caldir* where the calibration files are stored must be defined ad organized with a proper architecture, suche that the software can find the calibration coefficients for settings the threshold.
Normally *settingsdir* and *caldir* can be the same, but have been left separate for flexibility.

The *settingsdir* and *caldir* should be properly configured for your detector either in a configuration file (for use with text clients, GUI or API) or dynamically (works only for the text clients).

In the following, the architecture of the *settingsdir* and *caldir* is described for the different detectors.

1.4.1 MYTHEN

For mythen, an example of *settingsdir* and *caldir* is given in the software package by the directory `trimdir`. Since these directories are customized by producing trimbit files and calibration for each detector, make sure not to overwrite yours every time you upgrade the software.

settingsdir should contain three subdirectories `standard`, `fast` and `highgain` containing respectively the trimfiles `standard.trim`, `fast.trim` and `highgain.trim` which contain the correct voltage settings for the detector although all the individual channel thresholds set to 0. The original files contained in the package should be used, infact in case of error the detector would not recognize the correct settings.
The default trimbit files for each file will be stored in the directory according to the settings with the name `noise.snxxx` where `xxx` is the module serial number.

caldir should contain three subdirectories `standard`, `fast` and `highgain` containing respectively the trimfiles `standard.cal`, `fast.cal` and `highgain.cal` which contain an average calibration of the modules for the diffrent settings. However this can different from the correct one for each individual module even of several kev and therefore it is very important to perform an energy calibration on a module basis.
The default calibration files for each file will be stored in the directory according to the settings with the name `calibration.snxxx` where `xxx` is the module serial number.

1.4.2 GOTTHARD

A *settingsdir* should be configured, as the directory `settings` in this software package.
It must contain the subdirectories `dynamicgain`, `gain1`, `gain2`, `gain3`, `highgain`,

`lowgain`, `mediumgain`, and `veryhighgain` in order to properly configure the GOTTHARD detector using the various gain settings.

1.5 How should a configuration file look like?

A configuration file is a list of command necessary to properly configure your detector systems, with default valuee for some parameters and other settings that the users should normally not change dinamically. For this reason most of the commands present in the configuration file cannot be modified when using the API.
The syntax of the configuration file is exactly the same as in the comman line interface, therefore you can refere to that documentation to edit the files.
The configuration files look different for the different detector types. Examples of configuration files can be found in the examples directory.

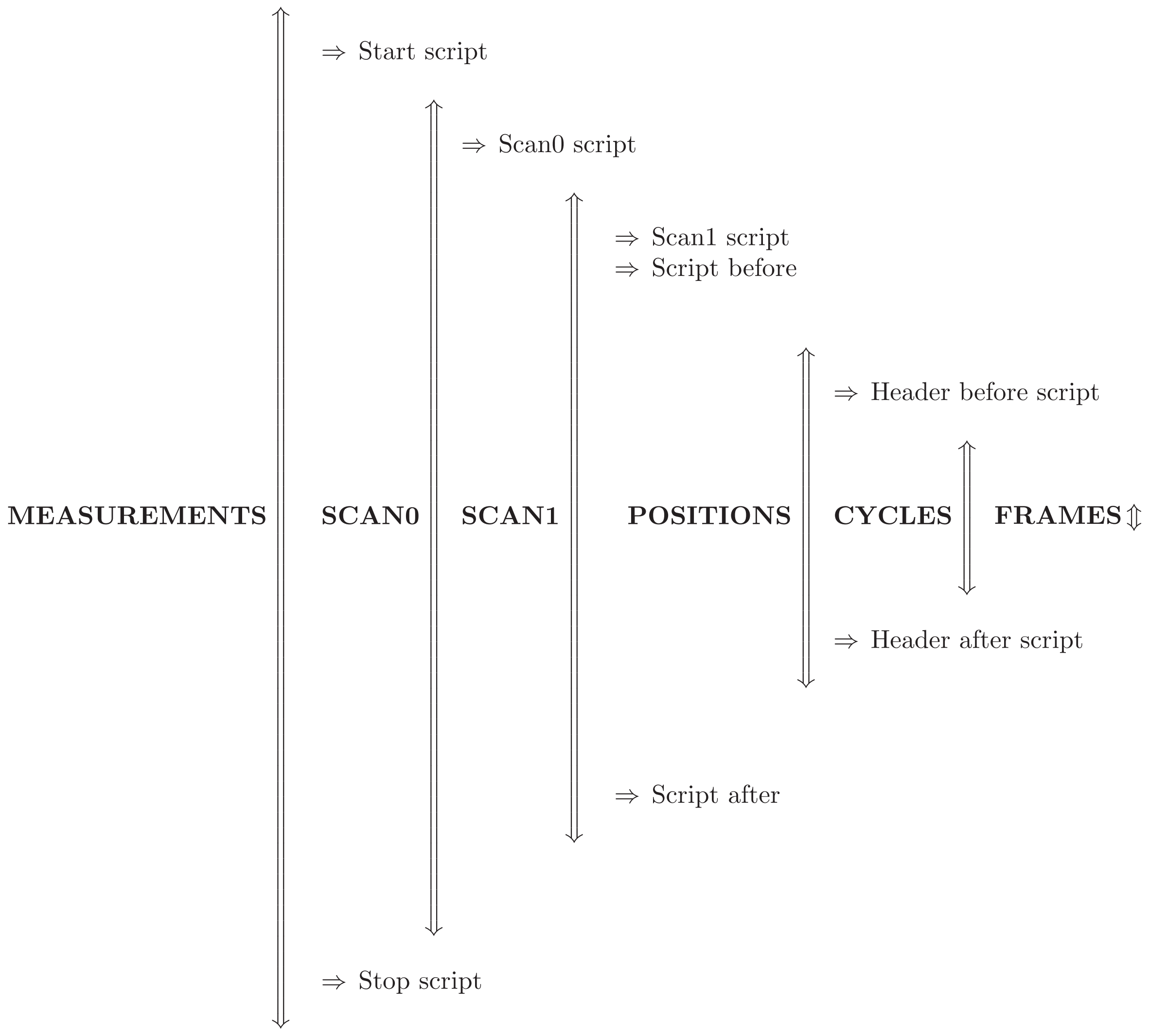
1.6 What is the meaning of the file name?

The final file name will be:
outdir/prefix`[_d`*d*`][_S`*v0*`][_s`*v1*`][_p`*p*`][_f`*f*`]_`*i*`.`*ext*
where:
outdir is the output directory path;
prefix is the chosen prefix for the file name;
d is the detector index, in case of data receiver and more than one detector;
v0 is the scan0 variable with the desired precision, if scan0 is enabled;
v1 is the scan1 variable with the desired precision, if scan1 is enabled;
p is the position index, if different positions are configured;
f is the frame index of the first frame stored in the file, if many frames and cycles are configured;
i is the file index;
ext is the file extension e.g. *.raw* for MYTHEN raw data, *.dat* for MYTHEN processed data.

1.7 Which is the sequence of the acquisition flow?

The software gives the possibility to setup several loops, actions and scan utilities which are then handled during the acquisition. The software will also take care to generate the file names and increment the indexes accordingly.

Figure 1.7 shows in which sequence the various scripts and loops are executed when calling the acquire command. The loops are drawn using the $\Updownarrow$ symbol, while the scripts using the $\Rightarrow$.



If you prefere to handle the acquisition from your acquisition enviroment, simply leave al scripts and scans disabled and call the acquition from your acquisition enviroment.
Only the frames and cycles loops are defined in firmware and guarantee a precise timing of the acquisition which cannot replaced by any other method (you can synchronize to your beamline by hardware connection of the IO signals as described in 1.8).

Hereafter a description of the meaning of the various loops:

Measurement loop executes offline several times the entire sequence of the acquisition. At the end of each measurement the *file index* is incremented.

Scan 0 loop is a high level scan loop which can be used e.g to loop on an enviroment variable (temperature, humidity...) or even to change sample. The list of steps or range of the *scan0 variable* must be set as scan0steps or scan0range. For small steps of the scan variable, avoid overwriting of the files specifying all the necessary digits in the filename by properly setting the precision with scan0prec.

Scan 1 loop is a low level scan loop which can be used e.g to loop on an enviroment variable (temperature, humidity...) or to move the sample in case of radiation damage.
The list of steps or range of the *scan1 variable* must be set as scan1steps or scan1range. For small steps of the scan variable, avoid overwriting of the files specifying all the necessary digits in the filename by properly setting the precision with scan1prec.

Position loop The detector is moved in the angular positions specified by the positions command.
The command for moving the detector should be defined as described in 1.10.
All data acquired during a position loop will be merged together, unless the number of positions is set to 0. In this case single frames will be converted to angle without merging.
Avoid using the position loop together with many frames/cycles.

Cycles loop is executed in real time and defines e.g. the number of triggers that will be accepted. The total number of images will be given by frames times cycles.

Frames loop is executed in real time and defines e.g. the images acquired per trigger. The total number of images will be given by frames times cycles.

Executing a script simply consists in a system call with the arguments specified below. The various scripts are executed only if they are enabled and different than *none*.
The scripts must be executable and the capability of parsing the arguments passed by the acquition program is left to the user writing the scripts. some example scripts writte in awk can be found in the examples directory.
Hereafter a short description of how the scripts are called and with which options:

Start script is executed at the very beginning of the measurement and can be used e.g. to initialize all the devices needed for the acquisition or open the beamline valves. The script is executed as:
script nrun=i par=p
where i is the *file index* and p is the *start script parameter*.

Scan0 script There are a few predefined scan modes i.e. *threshold* changing the detector threshold in DAC units, *energy* chaning the calibrated

detector threshold in eV, *trimbits* chaning the trimbits of the detector (advanced: do not use) and *position* changing the detector position (if the motor movement is correctly setup as described in 1.10). Otherwise the scan0script is executed as:
script nrun=i fn=fn var=v par=p
where i is the *file index*, fn is the *file name*, v is the value of the *scan0 variable* at the present step of the scan0 loop and p is the *scan 0 script parameter*.

Scan1 script There are a few predefined scan modes i.e. *threshold* changing the detector threshold in DAC units, *energy* chaning the calibrated detector threshold in eV, *trimbits* chaning the trimbits of the detector (advanced: do not use) and *position* changing the detector position (if the motor movement is correctly setup as described in 1.10). Otherwise the scan1script is executed as:
script nrun=i fn=fn var=v par=p
where i is the *file index*, fn is the *file name*, v is the value of the *scan1 variable* at the present step of the scan1 loop and p is the *scan 1 script parameter*.

Script before is called just before the beginning of the data taking and can be used e.g. to open the shutter. The script is executed as:
script nrun=i fn=fn par=p sv0=v0 sv1=v1 p0=p0 p1=p1
where i is the *file index*, fn is the *file name*, p is the *script before parameter*, v0 and v1 are the values of the *scan0 and scan1 variables* at the present step of the scan loops and p0 and p1 are the *scan0 and scan1 script parameters*.

Header before script is called before every step of the data taking (i.e. for each position, but at the beginning of the frames train if several acquisition have been programmed in real time) and can e.g. be used to dump the exact settings of the detector and beamline to reproduce or analyze the data offline. The script is executed as:
script nrun=i fn=fn par=p
where i is the *file index*, fn is the *file name*, and p is the *header before parameter*.

Header after script is called after every step of the data taking (i.e. for each position, but at the end of the frames train if several acquisition have been programmed in real time) and can e.g. be used to dump the exact settings of the detector and beamline to reproduce or analyze the data offline. The script is executed as:
script nrun=i fn=fn par=p
where i is the *file index*, fn is the *file name*, and p is the *header after parameter*.

Script after is called just after the end of the data taking and can be used e.g. to close the shutter. The script is executed as:

script nrun=i fn=fn par=p sv0=v0 sv1=v1 p0=p0 p1=p1
where i is the *file index*, fn is the *file name*, p is the *script after parameter*, v0 and v1 are the values of the *scan0 and scan1 variables* at the present step of the scan loops and p0 and p1 are the *scan0 and scan1 script parameters*.

Stop script is executed at the very end of the measurement and can be used e.g. to switch off all devices. The script is executed as:
script nrun=i par=p
where i si the *file index* and p is the *stop script parameter*.

1.8 How can I synchronize my detector with the experiment?

The timing of the detector is always defined by an active detection time followed by a dead time during which the detector is read out. This read out time as a fixed duration depending on the detector type and its configuration (e.g. dynamic range) which limits the maximum frame rate achievable.
In the following is a list of the main parameters involved in the acquisition timing:

Exposure time is the time during which the detector is detecting X-rays for each image (ignored is the timing mode is *gating*).

Period is the period of the images acquired. If it is shorter than the exposure time plus readout time, it will be ignored.

Delay after trigger can be set as a delay between the trigger signal and the start of the detection time.

Number of gates is used only in *gating* mode and is the number of times that the gate is toggled before the detector is read out. Useful for stroboscopic measurements with gate period shorter than the minim acquisition period of the detector, otherwise can be left to 1.

Number of frames is the number of images to be acquired per cycle. Frames and cycles have the same meaning except in trigger mode, when frames means the number of images per trigger. The total number of images is frames time cycles.

Number of cycles is the number of times that the frames are acquired. Frames and cycles have the same meaning except in trigger mode, when cycles means the number of triggers that will be accepted. The total number of images is frames time cycles.

Number of probes is used in stoboscopic measurements when the period is longer than the minimum acquisition period, but shorter than the frame

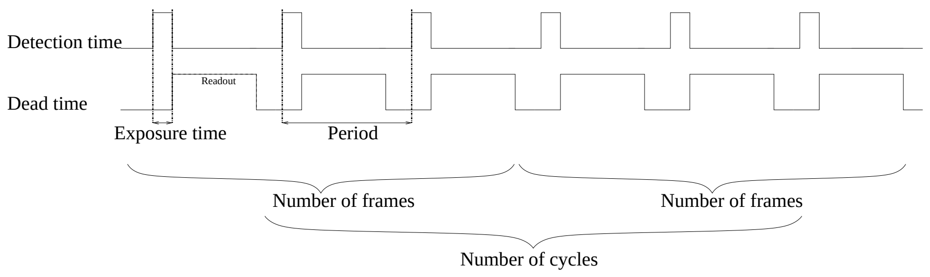


Figure 1.3: Auto timing: the detection time is defined by the exposure time and the period by period (if longer than exposure time plus readout time). The total number of images is frames (in the example 3) times cycles (in the example 2), and in this case there is no difference between the acquisition of the two.

rate.
In this case the data can be summed in firmware.
Currently it is implemented for Mythen only. If probes is set to 0, works normallyreturning an image for each readout, otherwise set number of cycles to 1. The maximum number of probes that can be set is 3. The detector will return a number of image equal to the number of probes, where all frames are going to be accumulated. The total number of readouts is number of frames time probes and for probes=1 the detector will return one image where all frames have been summed, for probes=2 two images where every second frame has been summed (each image accumulates the number of frames), for probes=3 three images where every third image has been summed (each image accumulates the number of frames).
The returned images will always have 32 bit dynamic range, while the dynamic range if the detector defines the bit depth of the counters in rder to limit the readout time, if necessary.
The probes counter waorks also in trigger and gating modes.

1.9 How can several controllers be synchronized?

If you are not performing time resolved measurements, you will probably not need any synchronization of the controllers: they will be started sequentially by the software and their acquisition will have a jitter of a few ms.
In the case you need a precise synchronization, on the other hand, hardware connection is required between the controllers through the external IO signals. The external signals used for this synchronization should be configured as *sync* with the *extsig* command.
In this case a *master* controller should be defined for the acquisition which will the send the synchronization signal to the other controllers, while the other controllers will use them as inputs.

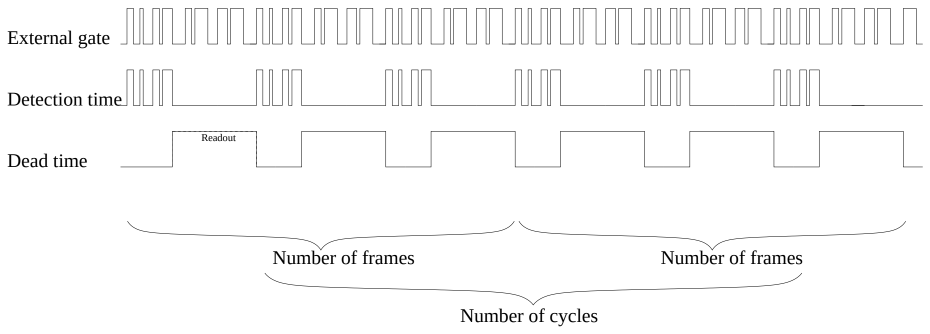


Figure 1.4: Gating mode: the detector acquires for a number of gates define by the user (in this case 4) before being read out, independently on the timing of the gates. The detector remains insensitive during the readout time and then starts being active again. External gates given during the readout time are ignored. The total number of images is frames (in the example 3) times cycles (in the example 2), and in this case there is no difference between the acquisition of the two. The polarity of the external gate signal can be defined by the user through the *external signal flag* (in the example active high).

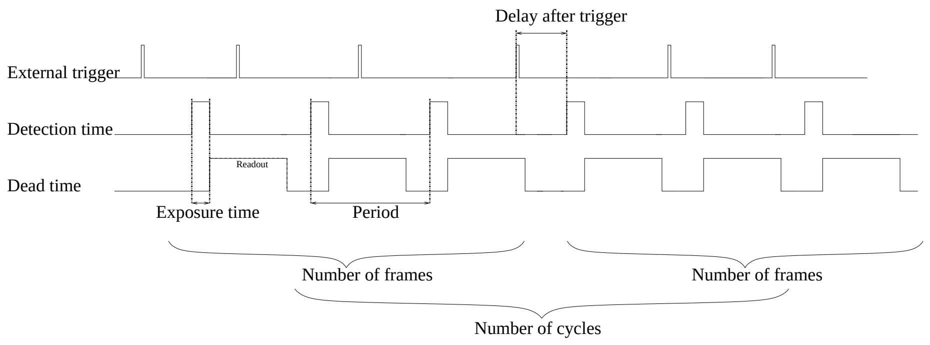


Figure 1.5: Trigger mode: the external trigger signal defines the start of the beginning of the acquisition, which starts after the delay set by the user. For each trigger, the number of frames is acquired (in the example 3) and all trigger signals ignored. The number of trigger accepted is given by the number of cycles (in the example 2). The polarity of the external trigger signal can be defined by the user through the *external signal flag* (in the example rising edge).

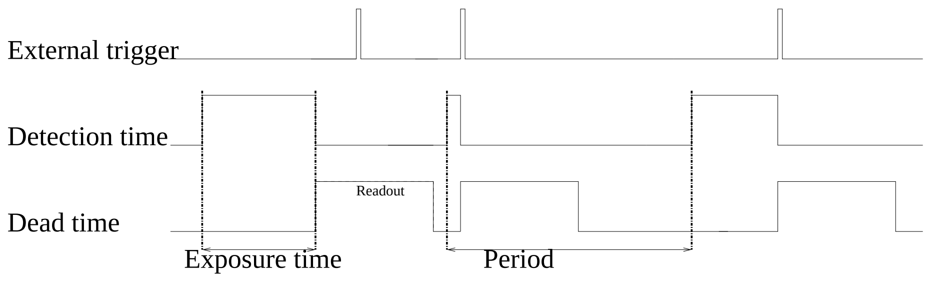


Figure 1.6: Read Out Trigger mode: the external trigger signal defines the beginning of the readout. The exposure time works as a time out for the waiting time for the trigger signal. The number of trigger accepted is given by the number of cycles (in the example 3) and it does not make sense to program more than one frame. The polarity of the external trigger signal can be defined by the user through the *external signal flag* (in the example rising edge).

The type of synchronization can be*gating* or *trigger* depending if the synchronization signal will gate the slave detectors or trigegr the beginning of the acquisition. There are no particular reasons to chose one or the other method, except if the user finds out that one is more stable than the other.
Normally the configuration of the synchronization is configured inside the configuration file and should not be changed dynamically by the user.

After the configuration, the synchronization of the controllers will be completely transparent for the user, who will simply have to setup the timing parameters of the detector as a whole.

1.10 How can the detector movement and position and I0 readout be customized for my beamline?

The easiest way to allow the software to perform all the necessary normalization and angular conversion steps, is enable it to move your detector and read the encoder position and the value of the ionization chamber.
These functions are defines as callbacks and can be redifined by registering your own functions. This is normally a good method if you use the API or are willing to write your main client program.
Otherwise the simpleast way is to edit the file
`slsDetectorSoftware/usersFunctions/usersFunctions.cpp`
where the default functions performing these actions are implemented and modify them to interface with your beamline hardware. The functions are written in C and are very simple to implement for anyone with some programming knowl-

edge.
A simple high-level solution in case you need to maintain the software for several beamlines and don't want to recompile for all of them is to call external scripts.

1.11 In which data format are written the data?

For MYTHEN the data are writen in ASCII fomat, one file per frame, in columns, either channel number - counts for the *.raw* files or angle (or channel number)-counts-error for the *.dat* files.

For the other detectors the files are written in binary format, and must be decoded depending on the detector.

1.11.1 GOTTHARD

Each file contains 100 frames.

Normal mode Each frame is split into 2 packets of 1286 bytes each, where actual data is 1280 bytes each. Both the packets (incl header and footer) are written one after the other into the file.

Representation of each packet:

- The first 4 bytes represents a number from which, the frame number and packet number can be derived. If the number was 108601, increment it by 1 to get 108602.
  Then this $(108602\&0xFFFFFFFE) >> 1 = 54301$ is the frame number and $(108602\&0x1) = 0$ is the packet number.
  0 is the packet on the left and 1 is the packet on the right.
  On a side note, when you use the data call back, we also give you the derived frame number as an argument.
- Data of 1280 bytes. 16 bits per pixel.
- 2 bytes of insignificant footer.

Short Frame Mode One Frame has only one packet of 518 bytes, where actual data is 512 bytes.

- first 4 bytes is the frame number. There is no packet number or increment required herecompared to the normal mode.
- Data of 512 bytes.
- 2 bytes of insignificant footer.

1.11.2 EIGER

1.11.3 JUNGFRAU

Chapter 2

Single photon counting detectors

2.1 Which detector settings should I choose?

The choice of the operation settings is very important in order to obtain good quality data.
Normally slower settings will reduce the electronics noise and therefore it is possible to work at lower energies, but will saturate for high photon fluxes.
On the other hand, faster settings will allow to work with higher photon intensities without pileup, but not to access lower energies because of an higher electronics noise.
Therefore it is extremely important to chose adequate settings for the detector depending on the X-ray energy and expected maximum count rate. In the following is a description of the energy and intensity range coverd by the different settings for each detector.

2.1.1 MYTHEN

Normally the user can follow these rules:

1. If the X-ray energy is lower than 8 keV the *High gain* setting should be used. Since it is a slow mode of operation it is necessary to take care that the maximum count rate is lower than 100 kcounts/s for all channels (use filters to reduce the beam intensisty).

2. For energies higher than 8 keV, the *Standard* setting is normally fine if the count rate can be kept lower than 300 kcounts/s for all channels (use filters to reduce the beam intensisty).

3. In case a larger count rate is required in order to keep the acquisition time shorter, the *Fast* setting must be selected. However the maximum count rate should never exceed 1 Mcounts/s for all channels.

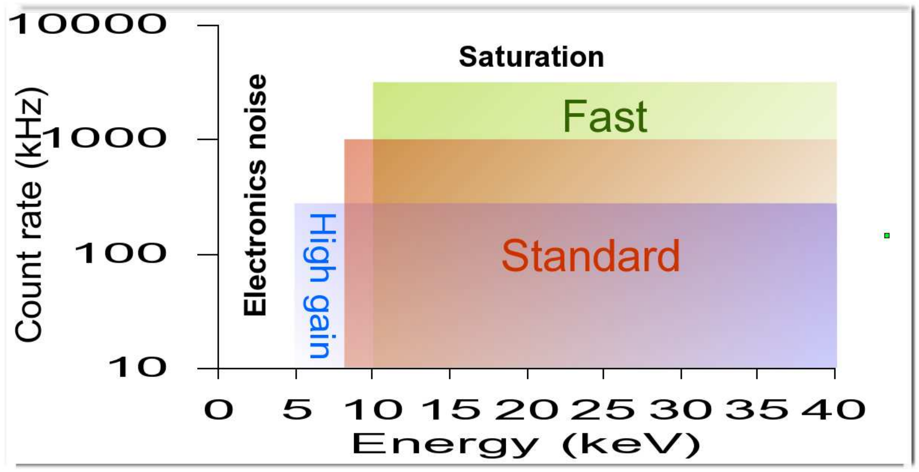


Figure 2.1: Plot indicating the reccomended choice of detector settings as a function of the X-ray energy and maximum count rate per channel..

2.2 How do I chose the comparator threshold?

Once selected the settings, the threshold should be selected. Figure 2.2 shows the number of counts as a function of the threshold value in the ideal case of monoenergetix X-rays of energy E_0=10 keV. For thresholds larger than the X-ray energy the detector should always count 0 and for lower thresholds it should always count all the photons. However the curve is smoothed around E_0 because of the electronic noise (ENC) and is not perfectly flat for lower energies because the photons absorbed in the region between two strips distribute their energy between them and it is not flully collected by a single channel (charge sharing). In order to count once al X-rays the threshold should be set at half of the X-ray energy $E_t = E_0/2$: if the threshold would be higher some photons would not be counted, leading to a loss of efficiency, while if it would be lower some photons would be counted twice leading to a loss of spatial resolution.

Since the detector threshold can't be precisely set at the same value for all channels but there will always be some spread of the order of 200 eV (threshold dispersion) there will always be some fluctuations on the number of counts between channels, which however should be corrected by the flat field correction.

The choice of the threshold should also depend from considerations regarding the emission of fluorescent radiation from the sample.
Figure 2.3 shows how the curve of the counts would look like for monochromatic X-rays of energy E_0 in presence of radiation of energy E_f emitted by the sample. The curve would show a second step at E_f.

Since the fluorecence emission is not present in the flat field data, the difference of counts between the channels due to the fluorescent radiation cannot be

corrected and the threshold E_t should be set at an energy larger than E_f. This also helps to cut down the background.
The difference of counts between the channels will be particularly large if the threshold is set in some "steep" part of the curve i.e. close to E_f or to E_0 (but in this case it would be corrected by the flat field, at cost of loss of efficiency). Because of the presence of the electronic noise, E_t should be at least 3 keV larger than E_f.

Here is a short list of rules to select the appropriate working threshold in order of importance (and eventually modify the X-ray energy):

1. List the fluorescent emission lines E_f that you expect from your sample.

2. If there is no fluorescent emission ($E_f < E_0$) $E_t = E_0/2$

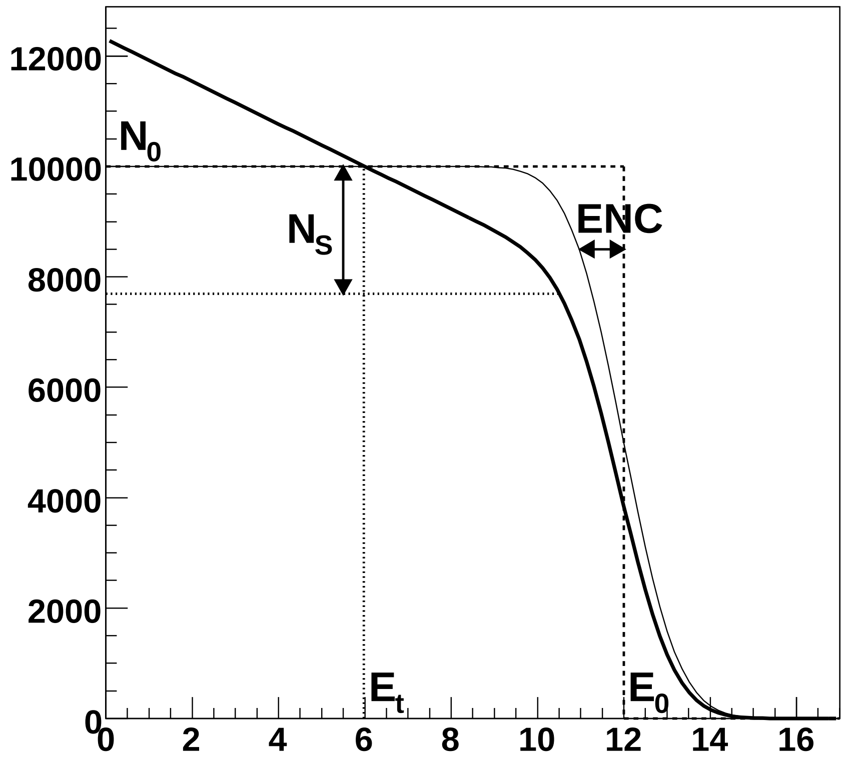


Figure 2.2: Number of counts as a function of the threshold detected in an ideal case.

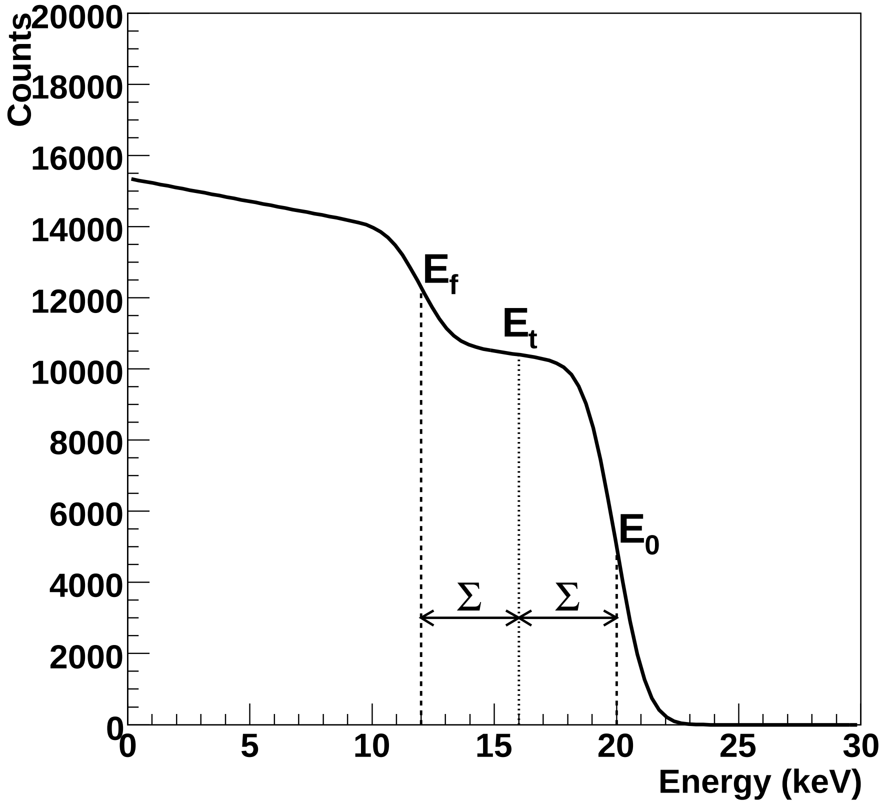


Figure 2.3: Number of counts as a function of the threshold detected in presence of fluorescent radiation

3. If there is fluorescent emission

   (a) $E_t > E_f + 3$ keV

   (b) $E_t < E_0 - 3$ keV

   If the range where both requirements are satisfied is large, try to increase the distance of E_t from E_f up to 5 keV and then set E_t as close as possible to the ideal value $E_t = E_0/2$

4. If it is not possible to satisfy the previous minimal requirements:

   (a) If you need high quality data and you can sacrifice detector efficiency (a lot!) $E_t > E_f + 3$ keV

(b) If you need fast measurments and you can sacrifice detector uniformity (difficult to say how much) and increase the background $E_t < E_f - 3$ keV. Remember that E_t is klimited by the electronic noise $E_t > 4$ keV (3 keV for *High gain* settings).

(c) Consider to change E_0 to values lower than E_f or at least 6-8 keV larger than E_f

2.3 How does the flat field correction work?

2.3.1 Why isn't my flat-field flat?

The main reasons of a non flat flat-field can be:

- The scattering from the glass rod is not uniform over the angular range. In this case you should take the flat field dynamically i.e. scanning the detector in front of the cylinder with the small window, as we do at the SLS. In this case when you shift the detector, the shape of the illumination remains in the same angular position (and shifts in channel number). Of course it depends a lot on the energy and on the geometry of the flat field acquisition.

- The entrance window for the X-rays is deformed (we also have this problem at the SLS). In this case when you move the detector the "mountain" moves with it in angle (And remains still in channel number). However this should correct without problems with the flat field correction, even in case of fluorescent emission. Should appear at all energies.

- Differences of efficiency between the modules i.e. mainly bad energy calibration. You normally see really steps at the transition between modules. Sometimes you have some groups of strips withing a module that are not properly trimmed and look as smallish peaks or valleys in the flat field. When you move the detector, these steps or peaks move in angle and remain still in channel number. These differences can slightly change as a function of the energy (probably more evident at lower energies) but should normally always be there for the same settings. These differences get much worse in presence of fluorescent emission, but normally correct properly with flat field correction.

2.3.2 Dynamic acquisition of the flat field

In case it is not possible to uniformely illuminate the detector due to its large dimensions, one of the solutions is to scan it in front of an illuminated are with a uniform speed such that the integrated number of counts during the exposure time is the same for all channels.

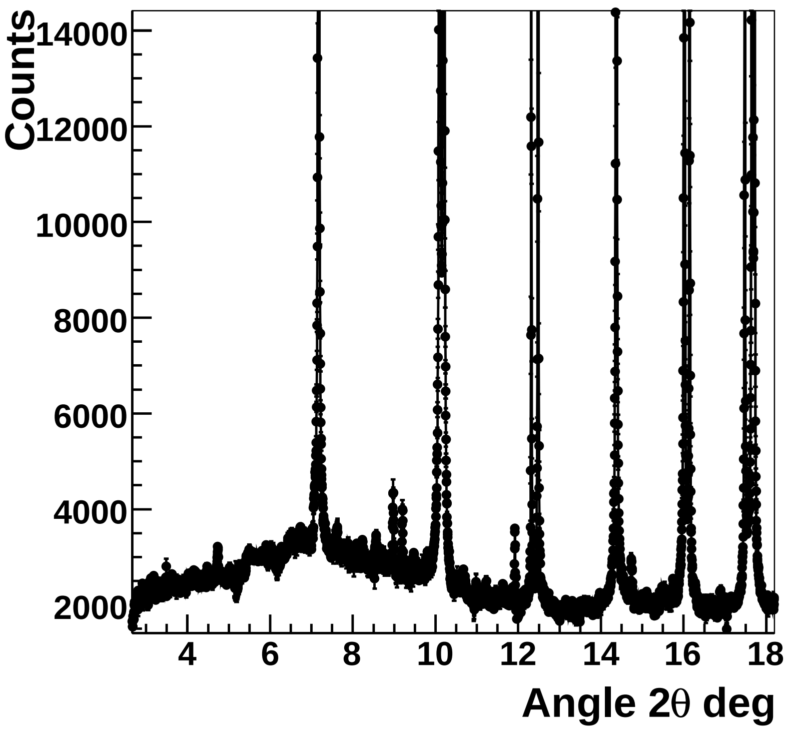


Figure 2.4: Example of data from a sample emitting fluorescent light and detector threshold set at a value close to the emission line. The background data cannot be properly flat field corrected.

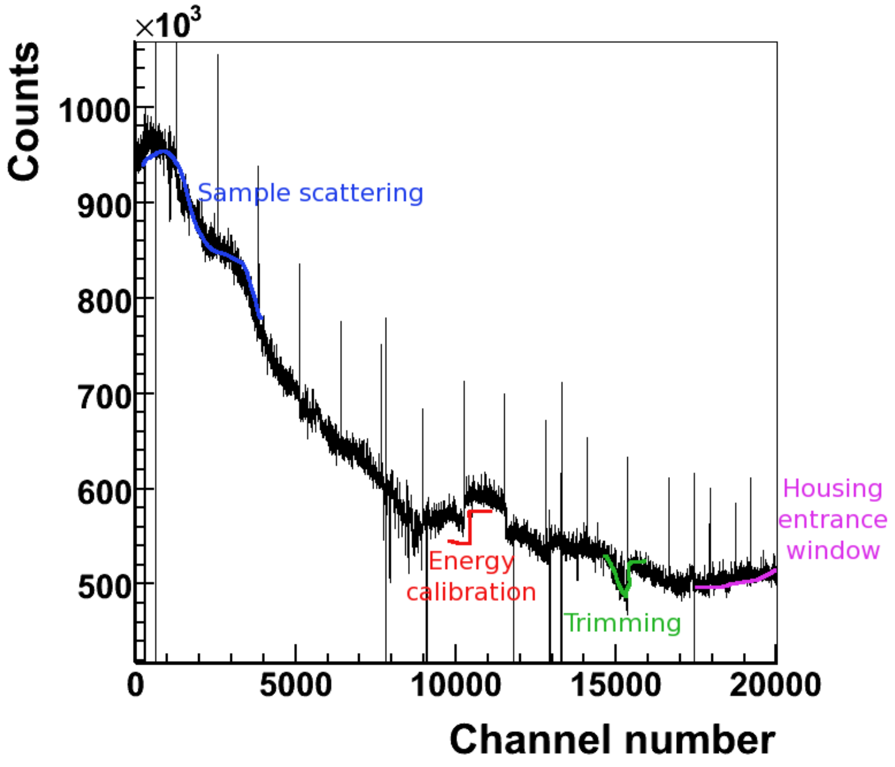


Figure 2.5: Example of a very bad flat field data set with highlights of some of the reasons which can cause the non-flat behavior for the MYTHEN detector. Similar effects can be visible also in 2D.

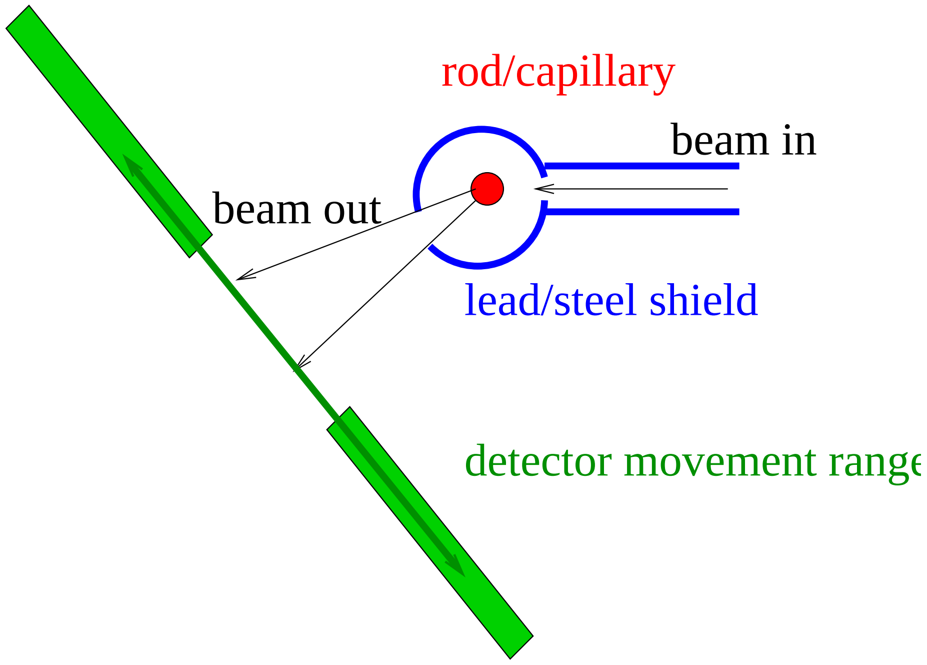


Figure 2.6: Sketch of the experimental setup for a dynamic acquisition of the flat field.

To do that, at the SLS we have optimized the dynamic acquisition of the flat fiel with the MYTHEN detector using a setup similar to the one sketched in figure 2.6. It is important that the scanning range of the detector is chose such that the detector is not illuminated both at the beginning and at the end of the acquisition. Moreover the movement of the detector should be as uniform as possible. To avoid this kind of systematic errors we normally sum two flat field images taken in the two opposite directions of translation.

Also take care that your sample does not emit fluorescent light at the chosen energy (e.g. a glass rod works at all energies, but heavier materials can be chosen to increase the efficiency at higher energies taking care that the fluorescence emission is negligible).

2.4 What happens when I trim the detector?

General remarks about trimming.

2.4.1 MYTHEN

Trimming with noise

The first step in the trimming procedure is to trim with noise (this is often sufficient). This has to be done for all the settings which are foreseen to be used (highgain, standard and fast).
The procedure for the noise trimming is as follows:

1. In the *Initialization tab* click on the settings for which you want to trim (e.g. standard)

2. In the *Initialization tab* click on the *advanced* radio button to make the trimming accessible.

3. In the *Acquisition tab* set the acquisition time to 100 ms, the repetion to 1 and the delay between frames to 0.

4. For noise trimming usually the default parameters $Vthreshold = 7$, $Counts = 500$, $Resolution = 4$ work.
   However, to verify the threshold setting it is best to make a threshold scan. To do this go to the *Data* tab, in the Data display section select the 2D color and type advanced option. In the *Acquisition* tab select your data directory. Set the number of positions to 0. Select Scan, Type threshold. Typical values for the range are 500 to 900 with a step size of 10. Then click on the start button to perform the threshold scan. After the threhold scan has finished an image similar to the one in 2.7 should be shown. Depending on the system the number of modules may vary. If the plot is similar to the one in 2.10 the noise trim files did already exist and have been loaded when selecting the settings. In this case you don't need to trim with noise again.
   Set the parameter Vthreshold in the *Trimming* box (*Initialization tab*) 10-30 DAC units below the onset of the noise for the module with the lowest threshold offset. Since the modules have differences in the offset and gain the onset of the noise varies.
   You can usually leave the remaining parameters unchanged (Counts/pixel=500; Resolution=4).

5. Select the directory where the noise trim files should be written and the filename, to wich will be attached the extension given by the module serial number (.snxxx). If you want the trimfiles to be loaded authomatically when the global settings are selected, select the default directory specified in the config file (or in the "trimbits/beamline" directory for the older software versions). Click on *Trim* to start the noise trimming process. After the trimming has finished look at the plot and the distribution of the trim bits. The distribution should be around 32±5 and should look gaussian. An example distribution is shown in figure 2.8 and an example plot in 2.9. If the distribution is too much off center change the counts/pixel, if it is

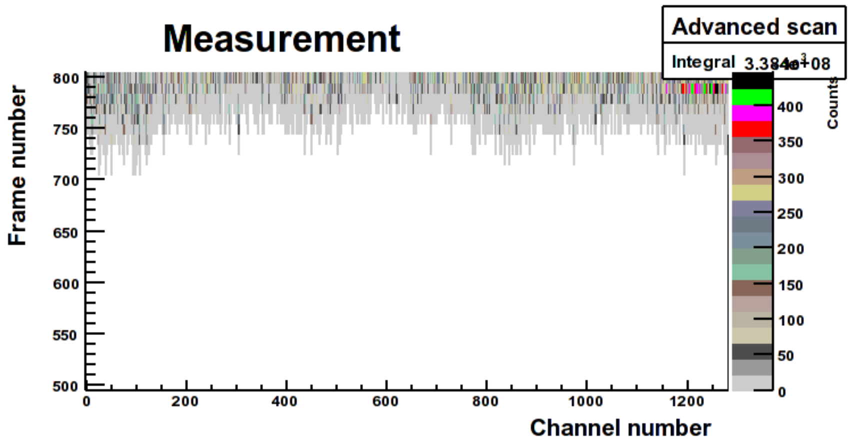


Figure 2.7: The untrimmed threshold scan.

too narrow reduce the resolution (set it to 3), if it is too wide increase it (set it to 5). Make sure not too many channels have a trim value of 0 or 63.

6. Execute the treshold scan again to verify the trimming was done properly. A plot similar tho the one in figure 2.10 should appear.

Improve the trimming using X-rays

The improvement of the trimming acquired with noise is not essential: at 12 keV an untrimmed module has a threshold dispersion which is about 1.4 keV and is already reduced to 200 eV at 12 keV by the noise trimming. At lower energies the noise trimming will be more effective, while the threshold dispesion will be still larger at higher energies. The trimming improvement reduces the threshold dispersion to 140 eV at 12 keV and is expected to be almost constant at all energies. For this reason it is suggested to perform the trimming improvement only when a small threshold dispersion is really important (e.g. to avoid flat field corrections or in presence of fluorescent lines close to the threshold value) and it will probably be not worthy at lower energies (i.e. threshold lower than 6 keV and X-ray energy lower than 12 keV). The procedure for the trimming improvement is as follows:

1. Select the settings of the detector and load the noise trimming file
2. Set the threshold at half of the X-ray energy (better if the detector has already been calibrated in energy like explained in 2.5)
3. Illuminate the detector with a flat field. This is very important to obtain a good trimming.

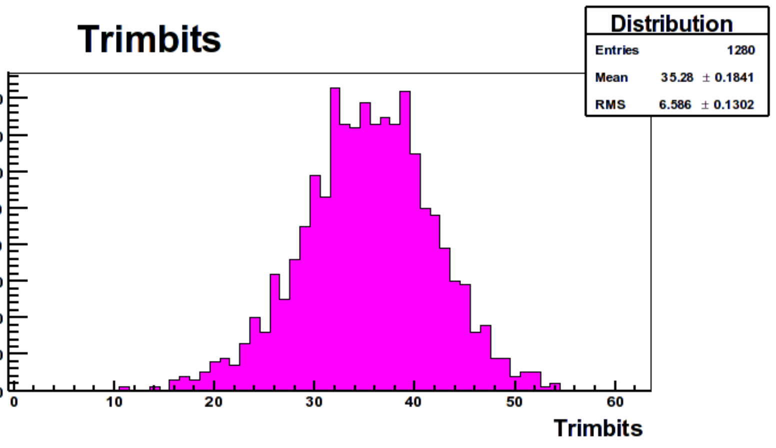


Figure 2.8: The distribution of the trimbits.

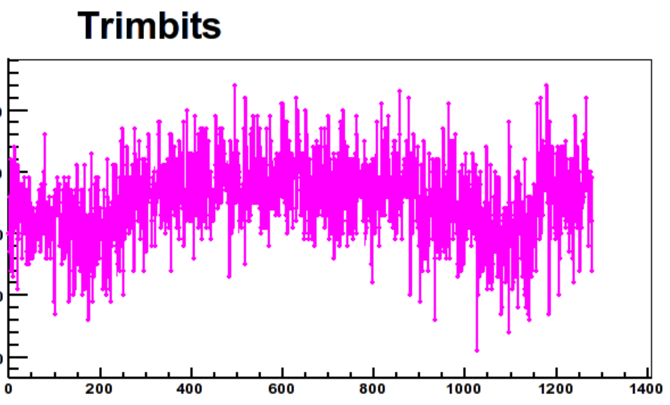


Figure 2.9: The trimbits for all the channels.

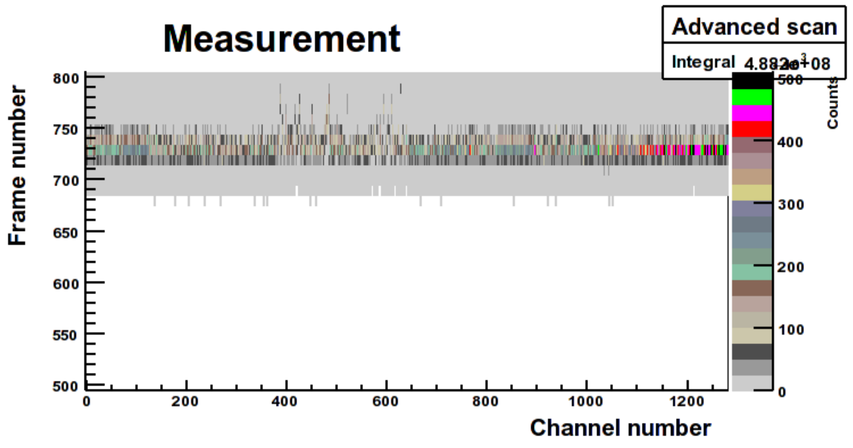


Figure 2.10: The trimmed threshold scan.

4. Select the *acquisition time* in the *acquisition tab* so that there are at least 1000 counts/strip per frame (the more counts, the better trimming). Set the repetions to 1 and the delay between frames to 0.

5. Go to expert mode by clicking on *advanced* in the *initialization tab*, *settings* box

6. In the trimming box select the directory where the noise trim files should be written and the filename, to wich will be attached the extension given by the module serial number (.snxxx).

7. Select the *improve* method *S*tart the trimming

If the trimming is correctly performed and the illumination is flat enough, the same trimming can be used every time you will measure at this same energy. The authomatic loading of energy-specific trim files is not yet implemented.

2.5 In what consists the energy calibration of the detector?

General remarks about DAC to energy conversion

2.5.1 MYTHEN

Since the conversion between the threshold DAC units and energy depends on the gain and offset of the channels the energy calibration has to be done for all settings (high gain, standard and fast). For each setting follow this procedure:

- Select the setting in the *Initialization* tab.
- Enter in expert mode by clicking the *Advanced* radiobutton in the *Global settings* box in the *Initialization* tab.
- If the trimfiles are in the correct location and with the correct name, they should be loaded by default every time you select the corresponding settings in the *global settings* box in the *initialization* tab [1]. If the trim files do not yet exist generate them as explained in section 2.4.1.
- Execute a threshold scan of the detector with at least three different energies. The more monochromatic are the X-rays, the better the calibration will be (i.e. scattered X-rays are better than the fluorescent emission). The scan should range from where all modules count 0 (estimate 850-20·energy(keV) DAcu) and where all modules start having a lot of noise (usually 800 DACu) with a step of 1 or 2 DACu. The acquisition time should be chosen so that there are at least 1000 counts per strip on the plateau.
- Open the file root/CalAllModules.C for editing. Change the value of the following global variables according to your needs:
  - *nmod* is the number of modules of your system.
  - *nscan* is the number of different threshold scans you acquired.
  - *en* is the array with the energies at which you acquired the scans, in keV.
  - *een* is the array with the errors on the energies at which you acquired the scans, in keV. It is usually small, but can be some hundreds eV in case of dirty fluorescent samples.
  - *fn* is the array containing the location and root file name of your data.
  - *run* is the array containing the run index of your data.
  - *startscan* is the array containing the threshold value at which you started the scans.
  - *stopscan* is the array containing the threshold value at which you finished the scans.
  - *stepscan* is the array containing the threshold step of the scans.
  - *ave* is the array containing the average number of counts per strip on the plateau (it must not be too precise).
  - *sn* is the array containing the list of the serial number of the modules to be calibrated. It is important that the list is in the right order, so that the optput calibration files have the extension .snxxx corresponding to the right module.

[1] The default name of the calibrated trimfiles is trimbits/beamline/*settings*/noise.snxxx where *settings* is the chosen settings. You can change it in src/qDetector.h and then recompile the acquisition program as described in **??**.

  - *of* is the location and root file name of the calibration file. The directory should already exist and the extension .snxxx will be attached to the output file.

- Launch root, which you should have already installed on your linux PC
- Execute the following commands in order to load the macros needed for the calibration:

  ```
  root$ .L root/NewMythenMacros.C++
  root$ .L root/CalAllModules.C++
  ```

  You should get a lot of warnings, but no errors.
- Execute the following command in order to run the calibration:

  ```
  root$ EnCalModules()
  root$
  ```

  Reading and analyzing the data takes some time, but, after a while, a canvas should open where the plots of the median of the counts of every module as a function of the threshold should be shown for each energy, fitted with a modified *erf* function in order to find the inflextion point. The last plot of the canvas should represent the inflexion points as a function of the energies, and by fitting it with a straight line it is possible to calculate the offset and gain for each module i.e. calibrate it as a function of the energy. Please check that this automated fitting procedure succeeds. In case you see many fitting errors you should try to check wether the variable you edited in root/CalAllModules.C are all correct or try to edit the fitting procedures in the two root macro files (sorry!).
- Copy the calibration file you obtained to calibration/*settings*.snxxx [2] By doing this the correct threshold for each module will be calculated every time you change the *threhsold energy* in the *global settings* box in the *initialization* tab, you have loaded some default settings and you are not in expert mode.

2.6 Why should I change the dynamic range of the counters?

2.7 When should I enable rate correction

2.7.1 How can I choose the dead time?

[2]The default name of the calibration file calibration/*settings*.snxxx where *settings* is the chosen settings. You can change it in src/qDetector.h and then recompile the acquisition program.