

My Project

Generated by Doxygen 1.8.5

Thu Sep 27 2018 18:31:05

Contents

1	Main Page	1
2	Class Documentation	2
2.1	slsDetectorUsers Class Reference	2
2.1.1	Detailed Description	6
2.1.2	Constructor & Destructor Documentation	6
2.1.3	Member Function Documentation	6
2.2	slsReceiverUsers Class Reference	29
2.2.1	Detailed Description	29
2.2.2	Constructor & Destructor Documentation	29
2.2.3	Member Function Documentation	30
2.2.4	Member Data Documentation	31
3	File Documentation	31
3.1	mainClient.cpp File Reference	31
3.1.1	Detailed Description	31
3.1.2	Function Documentation	32
3.2	mainReceiver.cpp File Reference	33
3.2.1	Detailed Description	33
3.2.2	Macro Definition Documentation	33
3.2.3	Function Documentation	34
3.2.4	Variable Documentation	35
3.3	/afs/psi.ch/project/sls_det_software/dhanya_softwareDevelopment/mySoft/slsDetectorPackage/sls-DetectorSoftware/slsDetector/slsDetectorUsers.h File Reference	36
3.4	/afs/psi.ch/project/sls_det_software/dhanya_softwareDevelopment/mySoft/slsDetectorPackage/sls-ReceiverSoftware/include/slsReceiverUsers.h File Reference	36
	Index	37

1 Main Page

API for SLS detectors data acquisition

Although the SLS detectors group develops several types of detectors (1/2D, counting/integrating etc.) it is common interest of the group to use a common platfor for data acquisition

The architecture of the acquisitions system is intended as follows:

- A socket server running on the detector (or more than one in some special cases)
- C++ classes common to all detectors for client-server communication. These can be supplied to users as libraries and embedded also in acquisition systems which are not developed by the SLS
- the possibility of using a Qt-based graphical user interface (with eventually root analisys capabilities)

- the possibility of running all commands from command line. In order to ensure a fast operation of this so called "text client" the detector parameters should not be re-initialized everytime. For this reason a shared memory block is allocated where the main detector flags and parameters are stored
- a Root library for data postprocessing and detector calibration (energy, angle).

[slsDetectorUsers](#) is a class to control the detector which should be instantiated by the users in their acquisition software (EPICS, spec etc.). A callback for displaying the data can be registered. More advanced configuration functions are not implemented and can be written in a configuration file tha can be read/written.

[slsReceiverUsers](#) is a class to receive the data for detectors with external data receiver (e.g. GOTTHARD). Callbacks can be registered to process the data or save them in specific formats.

detectorData is a structure containing the data and additional information which is used to return the data e.g. to the GUI for displaying them.

You can find examples of how this classes can be instatiated in [mainClient.cpp](#) and [mainReceiver.cpp](#)

Authors

Anna Bergamaschi, Dhanya Thattil

Version

3.0

Currently supported detectors

- MYTHEN
- GOTTHARD controls
- GOTTHARD data receiver
- EIGER
- JUNGFRAU

2 Class Documentation

2.1 slsDetectorUsers Class Reference

The [slsDetectorUsers](#) class is a minimal interface class which should be instantiated by the users in their acquisition software (EPICS, spec etc.). More advanced configuration functions are not implemented and can be written in a configuration or parameters file that can be read/written.

```
#include <slsDetectorUsers.h>
```

Public Member Functions

- [slsDetectorUsers](#) (int &ret, int id=0)
default constructor
- virtual [~slsDetectorUsers](#) ()
virtual destructor
- std::string [getDetectorDeveloper](#) ()
useful to define subset of working functions
- int [setOnline](#) (int const online=-1)
sets the onlineFlag

- int [setReceiverOnline](#) (int const online=-1)
sets the receivers onlineFlag
- void [startMeasurement](#) ()
start measurement and acquires
- int [stopMeasurement](#) ()
stop measurement
- int [getDetectorStatus](#) ()
get run status
- std::string [getFilePath](#) ()
returns the default output files path
- std::string [setFilePath](#) (std::string s)
sets the default output files path
- std::string [getFileName](#) ()
- std::string [setFileName](#) (std::string s)
sets the default output files path
- int [getFileIndex](#) ()
- int [setFileIndex](#) (int i)
sets the default output file index
- std::string [getFlatFieldCorrectionDir](#) ()
get flat field corrections file directory
- std::string [setFlatFieldCorrectionDir](#) (std::string dir)
set flat field corrections file directory
- std::string [getFlatFieldCorrectionFile](#) ()
get flat field corrections file name
- int [setFlatFieldCorrectionFile](#) (std::string fname="")
set flat field correction file
- int [enableFlatFieldCorrection](#) (int i=-1)
enable/disable flat field corrections (without changing file name)
- int [enableCountRateCorrection](#) (int i=-1)
enable/disable count rate corrections
- int [enablePixelMaskCorrection](#) (int i=-1)
enable/disable bad channel corrections
- int [enableAngularConversion](#) (int i=-1)
enable/disable angular conversion
- int [enableWriteToFile](#) (int i=-1)
- int [setPosition](#) (int nPos, double *pos)
set positions for the acquisition
- int [getPosition](#) (double *pos=NULL)
get positions for the acquisition
- int [setDetectorSize](#) (int x0=-1, int y0=-1, int nx=-1, int ny=-1)
sets the detector size
- int [getDetectorSize](#) (int &x0, int &y0, int &nx, int &ny)
gets detector size
- int [getMaximumDetectorSize](#) (int &nx, int &ny)
gets the maximum detector size
- int [setBitDepth](#) (int i=-1)
set/get dynamic range
- int [setSettings](#) (int isettings=-1)
set detector settings
- int [getThresholdEnergy](#) ()
get threshold energy

- int [setThresholdEnergy](#) (int e_eV)
set threshold energy
- int [setThresholdEnergy](#) (int e_ev, int tb, int isettings=-1, int id=-1)
set threshold energy with choice to load trimbits (eiger only)
- double [setExposureTime](#) (double t=-1, bool inseconds=false, int imod=-1)
set/get exposure time value
- double [setExposurePeriod](#) (double t=-1, bool inseconds=false, int imod=-1)
set/get exposure period
- double [setDelayAfterTrigger](#) (double t=-1, bool inseconds=false, int imod=-1)
set/get delay after trigger
- int64_t [setNumberOfGates](#) (int64_t t=-1, int imod=-1)
set/get number of gates
- int64_t [setNumberOfFrames](#) (int64_t t=-1, int imod=-1)
set/get number of frames i.e. number of exposure per trigger
- int64_t [setNumberOfCycles](#) (int64_t t=-1, int imod=-1)
set/get number of cycles i.e. number of triggers
- int [setTimingMode](#) (int pol=-1)
set/get the external communication mode
- int [readConfigurationFile](#) (std::string const fname)
Reads the configuration file – will contain all the informations needed for the configuration (e.g. for a PSI detector caldir, settingsdir, angconv, badchannels, hostname etc.)
- int [dumpDetectorSetup](#) (std::string const fname)
Reads the parameters from the detector and writes them to file.
- int [retrieveDetectorSetup](#) (std::string const fname)
Loads the detector setup from file.
- std::string [getDetectorType](#) ()
useful for data plotting etc.
- int [setReceiverMode](#) (int n=-1)
sets the mode by which gui requests data from receiver
- void [registerDataCallback](#) (int(*userCallback)(detectorData *d, int f, int s, void *), void *pArg)
register callback for accessing detector final data, also enables data streaming in client and receiver (if receiver exists)
- void [registerRawDataCallback](#) (int(*userCallback)(double *p, int n, void *), void *pArg)
register callback for accessing raw data - if the rawDataCallback is registered, no filewriting/postprocessing will be carried on automatically by the software - the raw data are deleted by the software
- virtual void [initDataset](#) (int refresh)
function to initialize a set of measurements (reset binning if angular conversion, reset summing otherwise) - can be overcome by the user's functions thanks to the virtual property
- virtual void [addFrame](#) (double *data, double pos, double i0, double t, std::string fname, double var)
adds frame to merging/summation - can be overcome by the user's functions thanks to the virtual property
- virtual void [finalizeDataset](#) (double *a, double *v, double *e, int &np)
finalizes the data set returning the array of angles, values and errors to be used as final data - can be overcome by the user's functions thanks to the virtual property
- int [enableDataStreamingFromReceiver](#) (int i=-1)
- int [enableDataStreamingToClient](#) (int i=-1)
- int [setReceiverDataStreamingOutPort](#) (int i=-1)
- int [setClientDataStreamingInPort](#) (int i=-1)
- std::string [setReceiverDataStreamingOutIP](#) (std::string ip="")
- std::string [setClientDataStreamingInIP](#) (std::string ip="")
- int64_t [getModuleFirmwareVersion](#) ()
- int64_t [getModuleSerialNumber](#) (int imod=-1)
- int64_t [getDetectorFirmwareVersion](#) ()

- int64_t [getDetectorSerialNumber](#) ()
- int64_t [getDetectorSoftwareVersion](#) ()
- int64_t [getThisSoftwareVersion](#) ()
- int [enableGapPixels](#) (int enable=-1)
- std::string [setReceiverFramesDiscardPolicy](#) (std::string f="get")
- int [setReceiverPartialFramesPadding](#) (int f=-1)
- int [setReceiverFramesPerFile](#) (int f=-1)
- int [sendSoftwareTrigger](#) ()
- double [getMeasuredPeriod](#) (bool inseconds=false, int imod=-1)
- double [getMeasuredSubFramePeriod](#) (bool inseconds=false, int imod=-1)
- void [registerAcquisitionFinishedCallback](#) (int(*func)(double, int, void *), void *pArg)
register callback for accessing detector final data
- void [registerGetPositionCallback](#) (double(*func)(void *), void *arg)
register callback for reading detector position
- void [registerConnectChannelsCallback](#) (int(*func)(void *), void *arg)
register callback for connecting to the epics channels
- void [registerDisconnectChannelsCallback](#) (int(*func)(void *), void *arg)
register callback to disconnect the epics channels
- void [registerGoToPositionCallback](#) (int(*func)(double, void *), void *arg)
register callback for moving the detector
- void [registerGoToPositionNoWaitCallback](#) (int(*func)(double, void *), void *arg)
register callback for moving the detector without waiting
- void [registerGetI0Callback](#) (double(*func)(int, void *), void *arg)
register callback reading to I0
- std::string [putCommand](#) (int nargs, char *args[], int pos=-1)
sets parameters in command interface <http://www.psi.ch/detectors/UsersSupportEN/sIs-DetectorClientHowTo.pdf>
- std::string [getCommand](#) (int nargs, char *args[], int pos=-1)
gets parameters in command interface <http://www.psi.ch/detectors/UsersSupportEN/sIs-DetectorClientHowTo.pdf>
- int [setClockDivider](#) (int value)
sets clock divider of detector
- int [setParallelMode](#) (int value)
sets parallel mode
- int [setOverflowMode](#) (int value)
show saturated for overflow in subframes in 32 bit mode (eiger only)
- int [setAllTrimbits](#) (int val, int id=-1)
sets all trimbits to value (only available for eiger)
- int [setDAC](#) (std::string dac, int val, int id=-1)
set dac value
- int [getADC](#) (std::string adc, int id=-1)
get adc value
- int [startReceiver](#) ()
start receiver listening mode
- int [stopReceiver](#) ()
stop receiver listening mode
- int [startAcquisition](#) ()
- int [stopAcquisition](#) ()
- int [setReceiverSilentMode](#) (int i)
- int [setHighVoltage](#) (int i)
- int [resetFramesCaughtInReceiver](#) ()
- int [setReceiverFifoDepth](#) (int i=-1)

- int [setFlowControl10G](#) (int i=-1)
- int [setTenGigabitEthernet](#) (int i=-1)
- int [getNMods](#) ()
- double [setSubFrameExposureTime](#) (double t=-1, bool inseconds=false, int imod=-1)
- double [setSubFrameExposureDeadTime](#) (double t=-1, bool inseconds=false, int imod=-1)
- int64_t [setNumberOfStorageCells](#) (int64_t t=-1, int imod=-1)
- int [setStorageCellStart](#) (int pos=-1)

Static Public Member Functions

- static std::string [runStatusType](#) (int s)
returns std::string from run status index
- static int [getDetectorSettings](#) (std::string s)
returns detector settings std::string from index
- static std::string [getDetectorSettings](#) (int s)
returns detector settings std::string from index
- static std::string [getTimingMode](#) (int f)
returns external communication mode std::string from index
- static int [getTimingMode](#) (std::string s)
returns external communication mode std::string from index

2.1.1 Detailed Description

The [slsDetectorUsers](#) class is a minimal interface class which should be instantiated by the users in their acquisition software (EPICS, spec etc.). More advanced configuration functions are not implemented and can be written in a configuration or parameters file that can be read/written.

Class for detector functionalities to embed the detector controls in the users custom interface e.g. EPICS, Lima etc.

Definition at line 83 of file [slsDetectorUsers.h](#).

2.1.2 Constructor & Destructor Documentation

2.1.2.1 [slsDetectorUsers::slsDetectorUsers \(int &ret, int id = 0 \)](#)

default constructor

Parameters

<i>ret</i>	address of return value. It will be set to 0 for success, else 1 for failure
<i>id</i>	multi detector id in creating multidetector object

2.1.2.2 [virtual slsDetectorUsers::~~slsDetectorUsers \(\)](#) [virtual]

virtual destructor

2.1.3 Member Function Documentation

2.1.3.1 [virtual void slsDetectorUsers::addFrame \(double * data, double pos, double i0, double t, std::string fname, double var \)](#) [virtual]

adds frame to merging/summation - can be overcome by the user's functions thanks to the virtual property

Parameters

<i>data</i>	pointer to the raw data
<i>pos</i>	encoder position
<i>i0</i>	beam monitor readout for intensity normalization (if 0 not performed)
<i>t</i>	exposure time in seconds, required only if rate corrections
<i>fname</i>	file name (unused since filewriting would be performed by the user)
<i>var</i>	optional parameter - unused.

2.1.3.2 int sIsDetectorUsers::dumpDetectorSetup (std::string const *fname*)

Reads the parameters from the detector and writes them to file.

Parameters

<i>fname</i>	file to write to
--------------	------------------

Returns

OK or FAIL

2.1.3.3 int sIsDetectorUsers::enableAngularConversion (int *i* = -1)

enable/disable angular conversion

Parameters

<i>i</i>	0 disables, 1 enables, -1 gets
----------	--------------------------------

Returns

0 if angular conversion disabled, 1 if enabled

2.1.3.4 int sIsDetectorUsers::enableCountRateCorrection (int *i* = -1)

enable/disable count rate corrections

Parameters

<i>i</i>	0 disables, 1 enables with default values, -1 gets
----------	--

Returns

0 if count corrections disabled, 1 if enabled

2.1.3.5 int sIsDetectorUsers::enableDataStreamingFromReceiver (int *i* = -1)

Enable or disable streaming data from receiver (creates transmitting sockets)

Parameters

<i>i</i>	0 to disable 1 to enable -1 to only get the value
----------	---

Returns

data streaming from receiver enable

2.1.3.6 int sIsDetectorUsers::enableDataStreamingToClient (int *i* = -1)

Enable data streaming to client (creates receiving sockets)

Parameters

<i>i</i>	0 to disable, 1 to enable, -1 to get the value
----------	--

Returns

data streaming to client enable

2.1.3.7 int sIsDetectorUsers::enableFlatFieldCorrection (int *i* = -1)

enable/disable flat field corrections (without changing file name)

Parameters

<i>i</i>	0 disables, 1 enables, -1 gets
----------	--------------------------------

Returns

0 if ff corrections disabled, 1 if enabled

2.1.3.8 int sIsDetectorUsers::enableGapPixels (int *enable* = -1)

Enable gap pixels, only for Eiger and for 8,16 and 32 bit mode. 4 bit mode gap pixels only in gui call back (register-DataCallback)

Parameters

<i>enable</i>	1 sets, 0 unsets, -1 gets
---------------	---------------------------

Returns

gap pixel enable or -1 for error

2.1.3.9 int sIsDetectorUsers::enablePixelMaskCorrection (int *i* = -1)

enable/disable bad channel corrections

Parameters

<i>i</i>	0 disables, 1 enables, -1 gets
----------	--------------------------------

Returns

0 if bad channels corrections disabled, 1 if enabled

2.1.3.10 int sIsDetectorUsers::enableWriteToFile (int *i* = -1)

Enable write file function included

2.1.3.11 virtual void sIsDetectorUsers::finalizeDataset (double * *a*, double * *v*, double * *e*, int & *np*) [virtual]

finalizes the data set returning the array of angles, values and errors to be used as final data - can be overcome by the user's functions thanks to the virtual property

Parameters

<i>a</i>	pointer to the array of angles - can be null if no angular coversion is required
<i>v</i>	pointer to the array of values
<i>e</i>	pointer to the array of errors
<i>np</i>	reference returning the number of points

2.1.3.12 int sIsDetectorUsers::getADC (std::string *adc*, int *id* = -1)

get adc value

Parameters

<i>adc</i>	adc as std::string. can be temp_fpga, temp_fpgaext, temp_10ge, temp_dcdc, temp_sodl, temp_sodr, temp_fpgafl, temp_fpgafr. others not supported
<i>id</i>	module index (-1 for all)

Returns

adc value in millidegree Celsius or -1 (if id=-1 & adc value is different for all modules) or -9999 if adc std::string does not match

2.1.3.13 std::string sIsDetectorUsers::getCommand (int *narg*, char * *args*[], int *pos* = -1)

gets parameters in command interface <http://www.psi.ch/detectors/UsersSupportEN/sIs-DetectorClientHowTo.pdf>

Parameters

<i>narg</i>	value to be set
<i>args</i>	value to be set
<i>pos</i>	position of detector in multisIsdetector list

Returns

answer std::string

2.1.3.14 std::string sIsDetectorUsers::getDetectorDeveloper ()

useful to define subset of working functions

Returns

"PSI" or "Dectris"

2.1.3.15 int64_t sIsDetectorUsers::getDetectorFirmwareVersion ()

get get Detector Firmware Version

Returns

id

2.1.3.16 int64_t sIsDetectorUsers::getDetectorSerialNumber ()

get get Detector Serial Number

Returns

id

2.1.3.17 static int sIsDetectorUsers::getDetectorSettings (std::string *s*) [inline],[static]

returns detector settings std::string from index

Parameters

s	can be standard, fast, highgain, dynamicgain, lowgain, mediumgain, veryhighgain
----------	---

Returns

setting index (-1 unknown std::string)

Definition at line 845 of file sIsDetectorUsers.h.

2.1.3.18 `static std::string sIsDetectorUsers::getDetectorSettings (int s)` `[inline], [static]`

returns detector settings std::string from index

Parameters

s	settings index
----------	----------------

Returns

standard, fast, highgain, dynamicgain, lowgain, mediumgain, veryhighgain, undefined when wrong index

Definition at line 859 of file sIsDetectorUsers.h.

2.1.3.19 `int sIsDetectorUsers::getDetectorSize (int & x0, int & y0, int & nx, int & ny)`

gets detector size

Parameters

x0	horizontal position origin in channel number
y0	vertical position origin in channel number
nx	number of channels in horizontal
ny	number of channels in vertical

Returns

OK/FAIL

2.1.3.20 `int64_t sIsDetectorUsers::getDetectorSoftwareVersion ()`

get get Detector Software Version

Returns

id

2.1.3.21 `int sIsDetectorUsers::getDetectorStatus ()`

get run status

Returns

status mask

2.1.3.22 `std::string sIsDetectorUsers::getDetectorType ()`

useful for data plotting etc.

Returns

Mythen, Eiger, Gotthard etc.

2.1.3.23 `int sIsDetectorUsers::getFileIndex ()`**Returns**

the default output file index

2.1.3.24 `std::string sIsDetectorUsers::getFileName ()`**Returns**

the default output files root name

2.1.3.25 `std::string sIsDetectorUsers::getFilePath ()`

returns the default output files path

2.1.3.26 `std::string sIsDetectorUsers::getFlatFieldCorrectionDir ()`

get flat field corrections file directory

Returns

flat field correction file directory

2.1.3.27 `std::string sIsDetectorUsers::getFlatFieldCorrectionFile ()`

get flat field corrections file name

Returns

flat field correction file name

2.1.3.28 `int sIsDetectorUsers::getMaximumDetectorSize (int & nx, int & ny)`

gets the maximum detector size

Parameters

<i>nx</i>	number of channels in horizontal
<i>ny</i>	number of channels in vertical

Returns

OK/FAIL

2.1.3.29 `double sIsDetectorUsers::getMeasuredPeriod (bool inseconds = false, int imod = -1)`

get measured period between previous two frames(EIGER only)

Parameters

<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

measured period

2.1.3.30 `double sIsDetectorUsers::getMeasuredSubFramePeriod (bool inseconds = false, int imod = -1)`

get measured sub period between previous two sub frames in 32 bit mode (EIGER only)

Parameters

<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

measured sub period

2.1.3.31 `int64_t sIsDetectorUsers::getModuleFirmwareVersion ()`

get get Module Firmware Version

Returns

id

2.1.3.32 `int64_t sIsDetectorUsers::getModuleSerialNumber (int imod = -1)`

get get Module Serial Number

Parameters

<i>imod</i>	module number
-------------	---------------

Returns

id

2.1.3.33 `int sIsDetectorUsers::getNMods ()`

returns total number of detector modules

Returns

the total number of detector modules

2.1.3.34 `int sIsDetectorUsers::getPositions (double * pos = NULL)`

get positions for the acquisition

Parameters

<i>pos</i>	array which will contain the encoder positions
------------	--

Returns

number of positions

2.1.3.35 `int64_t sIsDetectorUsers::getThisSoftwareVersion ()`

get this Software Version

Returns

id

2.1.3.36 `int sIsDetectorUsers::getThresholdEnergy ()`

get threshold energy

Returns

current threshold value for imod in ev (-1 failed)

2.1.3.37 `static std::string sIsDetectorUsers::getTimingMode (int f)` `[inline], [static]`

returns external communication mode std::string from index

Parameters

<i>f</i>	index for communication mode
----------	------------------------------

Returns

auto, trigger, ro_trigger, gating, triggered_gating, unknown when wrong mode

Definition at line 879 of file sIsDetectorUsers.h.

2.1.3.38 `static int sIsDetectorUsers::getTimingMode (std::string s)` `[inline], [static]`

returns external communication mode std::string from index

Parameters

<i>s</i>	index for communication mode
----------	------------------------------

Returns

auto, trigger, ro_trigger, gating, triggered_gating, unknown when wrong mode

Definition at line 896 of file sIsDetectorUsers.h.

2.1.3.39 `virtual void sIsDetectorUsers::initDataset (int refresh)` `[virtual]`

function to initialize a set of measurements (reset binning if angular conversion, reset summing otherwise) - can be overcome by the user's functions thanks to the virtual property

Parameters

<i>refresh</i>	if 1, all parameters like fcoefficients, badchannels, ratecorrections etc. are reset (should be called at least once with this option), if 0 simply reset merging/ summation
----------------	--

2.1.3.40 `std::string sIsDetectorUsers::putCommand (int narg, char * args[], int pos = -1)`

sets parameters in command interface <http://www.psi.ch/detectors/UsersSupportEN/sIs-DetectorClientHowTo.pdf>

Parameters

<i>narg</i>	value to be set
<i>args</i>	value to be set
<i>pos</i>	position of detector in multisIsdetector list

Returns

answer std::string

2.1.3.41 `int sIsDetectorUsers::readConfigurationFile (std::string const fname)`

Reads the configuration file – will contain all the informations needed for the configuration (e.g. for a PSI detector caldir, settingsdir, angconv, badchannels, hostname etc.)

Parameters

<i>fname</i>	file name
--------------	-----------

Returns

OK or FAIL

2.1.3.42 void sIsDetectorUsers::registerAcquisitionFinishedCallback (int(*) (double, int, void *) *func*, void * *pArg*)

register callback for accessing detector final data

Parameters

<i>func</i>	function to be called at the end of the acquisition. gets detector status and progress index as arguments
<i>pArg</i>	argument

2.1.3.43 void sIsDetectorUsers::registerConnectChannelsCallback (int(*) (void *) *func*, void * *arg*)

register callback for connecting to the epics channels

Parameters

<i>func</i>	function for connecting to the epics channels
<i>arg</i>	argument

2.1.3.44 void sIsDetectorUsers::registerDataCallback (int(*) (detectorData *d, int f, int s, void *) *userCallback*, void * *pArg*)

register callback for accessing detector final data, also enables data streaming in client and receiver (if receiver exists)

Parameters

<i>userCallback</i>	function for plotting/analyzing the data. Its arguments are the data structure d and the frame number f, s is for subframe number for eiger for 32 bit mode
<i>pArg</i>	argument

2.1.3.45 void sIsDetectorUsers::registerDisconnectChannelsCallback (int(*) (void *) *func*, void * *arg*)

register callback to disconnect the epics channels

Parameters

<i>func</i>	function to disconnect the epics channels
<i>arg</i>	argument

2.1.3.46 void sIsDetectorUsers::registerGetI0Callback (double(*) (int, void *) *func*, void * *arg*)

register callback reading to I0

Parameters

<i>func</i>	function for reading the I0 (called with parameter 0 before the acquisition, 1 after and the return value used as I0)
<i>arg</i>	argument

2.1.3.47 void sIsDetectorUsers::registerGetPositionCallback (double(*) (void *) *func*, void * *arg*)

register callback for reading detector position

Parameters

<i>func</i>	function for reading the detector position
<i>arg</i>	argument

2.1.3.48 `void sIsDetectorUsers::registerGoToPositionCallback (int(*) (double, void *) func, void * arg)`

register callback for moving the detector

Parameters

<i>func</i>	function for moving the detector
<i>arg</i>	argument

2.1.3.49 `void sIsDetectorUsers::registerGoToPositionNoWaitCallback (int(*) (double, void *) func, void * arg)`

register callback for moving the detector without waiting

Parameters

<i>func</i>	function for moving the detector
<i>arg</i>	argument

2.1.3.50 `void sIsDetectorUsers::registerRawDataCallback (int(*) (double *p, int n, void *) userCallback, void * pArg)`

register callback for accessing raw data - if the `rawDataCallback` is registered, no filewriting/postprocessing will be carried on automatically by the software - the raw data are deleted by the software

Parameters

<i>userCallback</i>	function for postprocessing and saving the data - p is the pointer to the data, n is the number of channels
<i>pArg</i>	argument

2.1.3.51 `int sIsDetectorUsers::resetFramesCaughtInReceiver ()`

reset frames caught in receiver should be called before [startReceiver\(\)](#)

Returns

OK or FAIL

2.1.3.52 `int sIsDetectorUsers::retrieveDetectorSetup (std::string const fName)`

Loads the detector setup from file.

Parameters

<i>fName</i>	file to read from
--------------	-------------------

Returns

OK or FAIL

2.1.3.53 `static std::string sIsDetectorUsers::runStatusType (int s)` `[inline], [static]`

returns `std::string` from run status index

Parameters

<i>s</i>	run status index
----------	------------------

Returns

std::string error, waiting, running, data, finished or unknown when wrong index

Definition at line 827 of file sIsDetectorUsers.h.

2.1.3.54 int sIsDetectorUsers::sendSoftwareTrigger ()

Sends a software internal trigger (EIGER only)

Returns

0 for success, 1 for fail

2.1.3.55 int sIsDetectorUsers::setAllTrimbits (int *val*, int *id* = -1)

sets all trimbits to value (only available for eiger)

Parameters

<i>val</i>	value to be set (-1 gets)
<i>id</i>	module index (-1 for all)

Returns

value set

2.1.3.56 int sIsDetectorUsers::setBitDepth (int *i* = -1)

set/get dynamic range

Parameters

<i>i</i>	dynamic range (-1 get)
----------	------------------------

Returns

current dynamic range

2.1.3.57 std::string sIsDetectorUsers::setClientDataStreamingInIP (std::string *ip* = " ")

(for expert users) Set/Get client streaming in ZMQ IP By default, it is the IP of receiver hostname

Parameters

<i>ip</i>	sets, empty std::string gets
-----------	------------------------------

Returns

client streaming in ZMQ IP

2.1.3.58 int sIsDetectorUsers::setClientDataStreamingInPort (int *i* = -1)

(for expert users) Set/Get client streaming in ZMQ port For multi modules, it calculates (increments), sets the ports and restarts the sockets

Parameters

<i>i</i>	sets, -1 gets
----------	---------------

Returns

client streaming in ZMQ port (if multiple, of first client socket)

2.1.3.59 `int sIsDetectorUsers::setClockDivider (int value)`

sets clock divider of detector

Parameters

<i>value</i>	value to be set (-1 gets)
--------------	---------------------------

Returns

speed of detector

2.1.3.60 `int sIsDetectorUsers::setDAC (std::string dac, int val, int id = -1)`

set dac value

Parameters

<i>dac</i>	dac as std::string. can be vcmp_ll, vcmp_lr, vcmp_rl, vcmp_rr, vthreshold, vrf, vrs, vtr, vcall, vcp. others not supported
<i>val</i>	value to be set (-1 gets)
<i>id</i>	module index (-1 for all)

Returns

dac value or -1 (if id=-1 & dac value is different for all modules) or -9999 if dac std::string does not match

2.1.3.61 `double sIsDetectorUsers::setDelayAfterTrigger (double t = -1, bool inseconds = false, int imod = -1)`

set/get delay after trigger

Parameters

<i>t</i>	time in ns (-1 gets)
<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

timer set value in ns, or s if specified

2.1.3.62 `int sIsDetectorUsers::setDetectorSize (int x0 = -1, int y0 = -1, int nx = -1, int ny = -1)`

sets the detector size

Parameters

<i>x0</i>	horizontal position origin in channel number (-1 unchanged)
-----------	---

<i>y0</i>	vertical position origin in channel number (-1 unchanged)
<i>nx</i>	number of channels in horizontal (-1 unchanged)
<i>ny</i>	number of channels in vertical (-1 unchanged)

Returns

OK/FAIL

2.1.3.63 `double sIsDetectorUsers::setExposurePeriod (double t = -1, bool inseconds = false, int imod = -1)`

set/get exposure period

Parameters

<i>t</i>	time in ns (-1 gets)
<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

timer set value in ns, or s if specified

2.1.3.64 `double sIsDetectorUsers::setExposureTime (double t = -1, bool inseconds = false, int imod = -1)`

set/get exposure time value

Parameters

<i>t</i>	time in sn (-1 gets)
<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

timer set value in ns, or s if specified

2.1.3.65 `int sIsDetectorUsers::setFileIndex (int i)`

sets the default output file index

Parameters

<i>i</i>	file index
----------	------------

Returns

the default output file index

2.1.3.66 `std::string sIsDetectorUsers::setFileName (std::string s)`

sets the default output files path

Parameters

<i>s</i>	file name
----------	-----------

Returns

the default output files root name

2.1.3.67 `std::string slsDetectorUsers::setFilePath (std::string s)`

sets the default output files path

Parameters

<i>s</i>	file path
----------	-----------

Returns

file path

2.1.3.68 `std::string sIsDetectorUsers::setFlatFieldCorrectionDir (std::string dir)`

set flat field corrections file directory

Parameters

<i>dir</i>	flat field correction file directory
------------	--------------------------------------

Returns

flat field correction file directory

2.1.3.69 `int sIsDetectorUsers::setFlatFieldCorrectionFile (std::string fname = " ")`

set flat field correction file

Parameters

<i>fname</i>	name of the flat field file (or "" if disable)
--------------	--

Returns

0 if disable (or file could not be read), >0 otherwise

2.1.3.70 `int sIsDetectorUsers::setFlowControl10G (int i = -1)`

set flow control for 10Gbe (Eiger only)

Parameters

<i>i</i>	1 sets, 0 unsets (-1 gets)
----------	----------------------------

Returns

flow control enable for 10 Gbe

2.1.3.71 `int sIsDetectorUsers::setHighVoltage (int i)`

set high voltage

Parameters

<i>i</i>	> 0 sets, 0 unsets, (-1 gets)
----------	-------------------------------

Returns

high voltage

2.1.3.72 `int64_t sIsDetectorUsers::setNumberOfCycles (int64_t t = -1, int imod = -1)`

set/get number of cycles i.e. number of triggers

Parameters

<i>t</i>	number of frames (-1 gets)
<i>imod</i>	module number (-1 for all)

Returns

number of frames

2.1.3.73 `int64_t sIsDetectorUsers::setNumberOfFrames (int64_t t = -1, int imod = -1)`

set/get number of frames i.e. number of exposure per trigger

Parameters

<i>t</i>	number of frames (-1 gets)
<i>imod</i>	module number (-1 for all)

Returns

number of frames

2.1.3.74 `int64_t sIsDetectorUsers::setNumberOfGates (int64_t t = -1, int imod = -1)`

set/get number of gates

Parameters

<i>t</i>	number of gates (-1 gets)
<i>imod</i>	module number (-1 for all)

Returns

number of gates

2.1.3.75 `int64_t sIsDetectorUsers::setNumberOfStorageCells (int64_t t = -1, int imod = -1)`

set/get number of additional storage cells (Jungfrau)

Parameters

<i>t</i>	number of additional storage cells. Default is 0. (-1 gets)
<i>imod</i>	module number (-1 for all)

Returns

number of additional storage cells

2.1.3.76 `int sIsDetectorUsers::setOnline (int const online = -1)`

sets the onlineFlag

Parameters

<i>online</i>	can be: -1 returns whether the detector is in online (1) or offline (0) state; 0 detector in offline state; 1 detector in online state
---------------	--

Returns

0 (offline) or 1 (online)

2.1.3.77 int sIsDetectorUsers::setOverflowMode (int *value*)

show saturated for overflow in subframes in 32 bit mode (eiger only)

Parameters

<i>value</i>	0 for do not show saturatd, 1 for show saturated (-1 gets)
--------------	--

Returns

overflow mode enable in 32 bit mode

2.1.3.78 int sIsDetectorUsers::setParallelMode (int *value*)

sets parallel mode

Parameters

<i>value</i>	0 for non parallel, 1 for parallel, 2 for safe mode (-1 gets)
--------------	---

Returns

gets parallel mode

2.1.3.79 int sIsDetectorUsers::setPositions (int *nPos*, double * *pos*)

set positions for the acquisition

Parameters

<i>nPos</i>	number of positions
<i>pos</i>	array with the encoder positions

Returns

number of positions

2.1.3.80 std::string sIsDetectorUsers::setReceiverDataStreamingOutIP (std::string *ip* = " ")

(for expert users) Set/Get receiver streaming out ZMQ IP By default, it is the IP of receiver hostname

Parameters

<i>ip</i>	sets, empty std::string gets
-----------	------------------------------

Returns

receiver streaming out ZMQ IP

2.1.3.81 int sIsDetectorUsers::setReceiverDataStreamingOutPort (int *i* = -1)

(for expert users) Set/Get receiver streaming out ZMQ port For multi modules, it calculates (increments), sets the ports and restarts the sockets

Parameters

<i>i</i>	sets, -1 gets
----------	---------------

Returns

receiver streaming out ZMQ port (if multiple, of first receiver socket)

2.1.3.82 int sIsDetectorUsers::setReceiverFifoDepth (int *i* = -1)

set receiver fifo depth

Parameters

<i>i</i>	number of images in fifo depth (-1 gets)
----------	--

Returns

receiver fifo depth

2.1.3.83 `std::string sIsDetectorUsers::setReceiverFramesDiscardPolicy (std::string f = "get ")`

Sets the frames discard policy in receiver frame discard policy options:

Parameters

<i>f</i>	nodiscard (default),discardempty, discardpartial (fastest), get to get the value
----------	--

Returns

f nodiscard (default),discardempty, discardpartial (fastest)

2.1.3.84 `int sIsDetectorUsers::setReceiverFramesPerFile (int f = -1)`

Sets the frames per file in receiver

Parameters

<i>f</i>	frames per file, 0 is infinite ie. every frame in same file (-1 gets)
----------	---

Returns

frames per file

2.1.3.85 `int sIsDetectorUsers::setReceiverMode (int n = -1)`

sets the mode by which gui requests data from receiver

Parameters

<i>n</i>	is 0 for random requests for fast acquisitions and greater than 0 for nth read requests
----------	---

Returns

the mode set in the receiver

2.1.3.86 `int sIsDetectorUsers::setReceiverOnline (int const online = -1)`

sets the receivers onlineFlag

Parameters

<i>online</i>	can be: -1 returns wether the receiver is in online (1) or offline (0) state; 0 receiver in offline state; 1 receiver in online state
---------------	---

Returns

0 (offline) or 1 (online)

2.1.3.87 `int sIsDetectorUsers::setReceiverPartialFramesPadding (int f = -1)`

Sets the frame padding in receiver

Parameters

<i>f</i>	0 does not partial frames, 1 pads partial frames (-1 gets)
----------	--

Returns

partial frames padding enable

2.1.3.88 `int sIsDetectorUsers::setReceiverSilentMode (int i)`

set receiver in silent mode

Parameters

<i>i</i>	1 sets, 0 unsets (-1 gets)
----------	----------------------------

Returns

silent mode of receiver

2.1.3.89 `int sIsDetectorUsers::setSettings (int isettings = -1)`

set detector settings

Parameters

<i>isettings</i>	settings index (-1 gets)
------------------	--------------------------

Returns

current settings

2.1.3.90 `int sIsDetectorUsers::setStorageCellStart (int pos = -1)`

Set storage cell that stores first acquisition of the series (Jungfrau)

Parameters

<i>pos</i>	storage cell index. Value can be 0 to 15. Default is 15. (-1 gets)
------------	--

Returns

the storage cell that stores the first acquisition of the series

2.1.3.91 `double sIsDetectorUsers::setSubFrameExposureDeadTime (double t = -1, bool inseconds = false, int imod = -1)`

Set sub frame dead time (only for Eiger) Very advanced feature. Meant to be a constant in config file by an expert for each individual module

Parameters

<i>t</i>	sub frame dead time (-1 gets)
<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

sub frame dead time in ns, or s if specified

2.1.3.92 `double sIsDetectorUsers::setSubFrameExposureTime (double t = -1, bool inseconds = false, int imod = -1)`

Set sub frame exposure time (only for Eiger)

Parameters

<i>t</i>	sub frame exposure time (-1 gets)
<i>inseconds</i>	true if the value is in s, else ns
<i>imod</i>	module number (-1 for all)

Returns

sub frame exposure time in ns, or s if specified

2.1.3.93 int sIsDetectorUsers::setTenGigabitEthernet (int *i* = -1)

enable/disable 10GbE (Eiger only)

Parameters

<i>i</i>	1 sets, 0 unsets (-1 gets)
----------	----------------------------

Returns

10GbE enable

2.1.3.94 int sIsDetectorUsers::setThresholdEnergy (int *e_eV*)

set threshold energy

Parameters

<i>e_eV</i>	threshold in eV
-------------	-----------------

Returns

current threshold value for imod in ev (-1 failed)

2.1.3.95 int sIsDetectorUsers::setThresholdEnergy (int *e_ev*, int *tb*, int *isettings* = -1, int *id* = -1)

set threshold energy with choice to load trimbits (eiger only)

Parameters

<i>e_ev</i>	threshold in ev
<i>tb</i>	1 loads trimbits, 0 does not load trimbits
<i>isettings</i>	settings index (-1 uses current setting)
<i>id</i>	module index (-1 for all)

Returns

current threshold value in ev (-1 failed)

2.1.3.96 int sIsDetectorUsers::setTimingMode (int *pol* = -1)

set/get the external communication mode

Parameters

<i>pol</i>	value to be set
------------	-----------------

See Also

[getTimingMode](#)

Returns

current external communication mode

2.1.3.97 int sIsDetectorUsers::startAcquisition ()

start detector real time acquisition in non blocking mode does not include scans, scripts, incrementing file index, starting/stopping receiver, resetting frames caught in receiver

Returns

OK if all detectors are properly started, FAIL otherwise

2.1.3.98 void sIsDetectorUsers::startMeasurement ()

start measurement and acquires

Returns

OK/FAIL

2.1.3.99 int sIsDetectorUsers::startReceiver ()

start receiver listening mode

Returns

returns OK or FAIL

2.1.3.100 int sIsDetectorUsers::stopAcquisition ()

stop detector real time acquisition

Returns

OK if all detectors are properly started, FAIL otherwise

2.1.3.101 int sIsDetectorUsers::stopMeasurement ()

stop measurement

Returns

OK/FAIL

2.1.3.102 int sIsDetectorUsers::stopReceiver ()

stop receiver listening mode

Returns

returns OK or FAIL

The documentation for this class was generated from the following file:

- /afs/psi.ch/project/sls_det_software/dhanya_softwareDevelopment/mySoft/sIsDetectorPackage/sIsDetector-Software/sIsDetector/[sIsDetectorUsers.h](#)

2.2 sIsReceiverUsers Class Reference

Class for implementing the SLS data receiver in the users application. Callbacks can be defined for processing and/or saving data.

```
#include <sIsReceiverUsers.h>
```

Public Member Functions

- [sIsReceiverUsers](#) (int argc, char *argv[], int &success)
- [~sIsReceiverUsers](#) ()
- int [start](#) ()
- void [stop](#) ()
- int64_t [getReceiverVersion](#) ()
- void [registerCallBackStartAcquisition](#) (int(*func)(char *filepath, char *filename, uint64_t fileindex, uint32_t datasize, void *), void *arg)
register callback for starting the acquisition
- void [registerCallBackAcquisitionFinished](#) (void(*func)(uint64_t nf, void *), void *arg)
register callback for end of acquisition
- void [registerCallBackRawDataReady](#) (void(*func)(char *header, char *datapointer, uint32_t datasize, void *), void *arg)
register callback to be called when data are available (to process and/or save the data).
- void [registerCallBackRawDataModifyReady](#) (void(*func)(char *header, char *datapointer, uint32_t &rev-Datasize, void *), void *arg)
register callback to be called when data are available (to process and/or save the data).

Public Attributes

- sIsReceiver * [receiver](#)

2.2.1 Detailed Description

Class for implementing the SLS data receiver in the users application. Callbacks can be defined for processing and/or saving data.

[sIsReceiverUsers](#) is a class that can be instantiated in the users software to receive the data from the detectors. Callbacks can be defined for processing and/or saving data

Definition at line 15 of file sIsReceiverUsers.h.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 sIsReceiverUsers::sIsReceiverUsers (int argc, char * argv[], int & success)

Constructor reads config file, creates socket, assigns function table

Parameters

<i>argc</i>	from command line
<i>argv</i>	from command line
<i>success</i>	socket creation was successfull

2.2.2.2 sIsReceiverUsers::~~sIsReceiverUsers ()

Destructor

2.2.3 Member Function Documentation

2.2.3.1 int64_t sIsReceiverUsers::getReceiverVersion ()

get get Receiver Version

Returns

id

2.2.3.2 void sIsReceiverUsers::registerCallBackAcquisitionFinished (void(*)(uint64_t nf, void *) func, void * arg)

register callback for end of acquisition

Parameters

<i>func</i>	end of acquisition callback. Argument nf is total frames caught
<i>arg</i>	argument

Returns

nothing

2.2.3.3 void sIsReceiverUsers::registerCallBackRawDataModifyReady (void(*)(char *header, char *datapointer, uint32_t &revDatasize, void *) func, void * arg)

register callback to be called when data are available (to process and/or save the data).

Parameters

<i>func</i>	raw data ready callback. arguments are sIs_receiver_header, dataPointer, revDatasize is the reference of data size in bytes. Can be modified to the new size to be written/streamed. (only smaller value).
<i>arg</i>	argument

Returns

nothing

2.2.3.4 void sIsReceiverUsers::registerCallBackRawDataReady (void(*)(char *header, char *datapointer, uint32_t datasize, void *) func, void * arg)

register callback to be called when data are available (to process and/or save the data).

Parameters

<i>func</i>	raw data ready callback. arguments are sIs_receiver_header, dataPointer, dataSize
<i>arg</i>	argument

Returns

nothing

2.2.3.5 void sIsReceiverUsers::registerCallBackStartAcquisition (int(*)(char *filepath, char *filename, uint64_t fileindex, uint32_t datasize, void *) func, void * arg)

register callback for starting the acquisition

Parameters

<i>func</i>	callback to be called when starting the acquisition. Its arguments are filepath, filename, fileindex, datasize
<i>arg</i>	argument

Returns

value is insignificant at the moment, we write depending on file write enable, users get data to write depending on call backs registered

2.2.3.6 int slsReceiverUsers::start ()

starts listening on the TCP port for client communication

Returns

0 for success or 1 for FAIL in creating TCP server

2.2.3.7 void slsReceiverUsers::stop ()

stops listening to the TCP & UDP port and exit receiver program

2.2.4 Member Data Documentation**2.2.4.1 slsReceiver* slsReceiverUsers::receiver**

Definition at line 86 of file slsReceiverUsers.h.

The documentation for this class was generated from the following file:

- /afs/psi.ch/project/sls_det_software/dhanya_softwareDevelopment/mySoft/slsDetectorPackage/slsReceiver-Software/include/[slsReceiverUsers.h](#)

3 File Documentation

3.1 mainClient.cpp File Reference

```
#include "slsDetectorUsers.h"
#include "detectorData.h"
#include <iostream>
#include <cstdlib>
```

Functions

- int [dataCallback](#) (detectorData *pData, int iframe, int isubframe, void *pArg)
- int [main](#) (int argc, char **argv)

3.1.1 Detailed Description

This file is an example of how to implement the [slsDetectorUsers](#) class You can compile it linking it to the slsDetector library

```
g++ mainClient.cpp -L lib -lslsDetector -L/usr/lib64/ -L lib2 -lzmq -pthread -lrt -lm -lstdc++
```


where,

lib is the location of libSlsDetector.so

lib2 is the location of the libzmq.a. [libzmq.a is required only when using data call backs and enabling data streaming from receiver to client. It is linked in manual/manual-api from slsReceiverSoftware/include]

Definition in file [mainClient.cpp](#).

3.1.2 Function Documentation

3.1.2.1 int dataCallback (detectorData * *pData*, int *iframe*, int *isubframe*, void * *pArg*)

Data Call back function defined

Parameters

<i>pData</i>	pointer to data structure received from the call back
<i>iframe</i>	frame number of data passed
<i>isubframe</i>	sub frame number of data passed (only valid for EIGER in 32 bit mode)
<i>pArg</i>	pointer to object

Returns

integer that is currently ignored

Definition at line 32 of file mainClient.cpp.

3.1.2.2 int main (int *argc*, char ** *argv*)

Example of a main program using the [slsDetectorUsers](#) class

- Arguments are optional
 - argv[1] : Configuration File
 - argv[2] : Measurement Setup File
 - argv[3] : Detector Id (default is zero)
- if specified, set ID from argv[3]
- [slsDetectorUsers](#) Object is instantiated with appropriate ID
- if specified, load configuration file (necessary at least the first time it is called to properly configure advanced settings in the shared memory)
- set detector in shared memory online (in case no config file was used)
- set receiver in shared memory online (in case no config file was used)
- registering data callback
- ensuring detector status is idle before starting acquisition. exiting if not idle
- if provided, load detector settings
- start measurement
- returning when acquisition is finished or data are available
- delete [slsDetectorUsers](#) object

Definition at line 49 of file mainClient.cpp.

3.2 mainReceiver.cpp File Reference

```
#include "sls_receiver_defs.h"
#include "slsReceiverUsers.h"
#include <iostream>
#include <string.h>
#include <signal.h>
#include <cstdlib>
#include <sys/types.h>
#include <sys/wait.h>
#include <string>
#include <unistd.h>
#include <errno.h>
#include <syscall.h>
```

Macros

- `#define PRINT_IN_COLOR(c, f,...) printf ("\033[%dm" f RESET, 30 + c+1, ##__VA_ARGS__)`

Functions

- void `sigInterruptHandler` (int p)
- void `printHelp` ()
- int `StartAcq` (char *filepath, char *filename, uint64_t fileindex, uint32_t datasize, void *p)
- void `AcquisitionFinished` (uint64_t frames, void *p)
- void `GetData` (char *metadata, char *datapointer, uint32_t datasize, void *p)
- void `GetData` (char *metadata, char *datapointer, uint32_t &revDatasize, void *p)
- int `main` (int argc, char *argv[])

Variables

- bool `keeprunning`

3.2.1 Detailed Description

This file is an example of how to implement the `slsReceiverUsers` class You can compile it linking it to the `slsReceiver` library

```
g++ mainReceiver.cpp -L lib -lSlsReceiver -L/usr/lib64/ -L lib2 -lzmq -pthread -lrt -lm -lstdc++
```

where,

`lib` is the location of `ISlsReceiver.so`

`lib2` is the location of the `libzmq.a`. [`libzmq.a` is required only when using data call backs and enabling data streaming from receiver to client. It is linked in `manual/manual-api` from `slsReceiverSoftware/include`]

Definition in file `mainReceiver.cpp`.

3.2.2 Macro Definition Documentation

3.2.2.1 `#define PRINT_IN_COLOR(c, f, ...) printf ("\033[%dm" f RESET, 30 + c+1, ##__VA_ARGS__)`

Define Colors to print data call back in different colors for different receivers

Definition at line 38 of file `mainReceiver.cpp`.

3.2.3 Function Documentation

3.2.3.1 void AcquisitionFinished (uint64_t frames, void * p)

Acquisition Finished Call back

Parameters

<i>frames</i>	Number of frames caught
<i>p</i>	pointer to object

Definition at line 85 of file mainReceiver.cpp.

3.2.3.2 void GetData (char * metadata, char * datapointer, uint32_t datasize, void * p)

Get Receiver Data Call back Prints in different colors(for each receiver process) the different headers for each image call back.

Parameters

<i>metadata</i>	sls_receiver_header metadata
<i>datapointer</i>	pointer to data
<i>datasize</i>	data size in bytes.
<i>p</i>	pointer to object

Definition at line 98 of file mainReceiver.cpp.

3.2.3.3 void GetData (char * metadata, char * datapointer, uint32_t & revDatasize, void * p)

Get Receiver Data Call back (modified) Prints in different colors(for each receiver process) the different headers for each image call back.

Parameters

<i>metadata</i>	sls_receiver_header metadata
<i>datapointer</i>	pointer to data
<i>datasize</i>	data size in bytes.
<i>revDatasize</i>	new data size in bytes after the callback. This will be the size written/streamed. (only smaller value is allowed).
<i>p</i>	pointer to object

Definition at line 132 of file mainReceiver.cpp.

3.2.3.4 int main (int argc, char * argv[])

Example of main program using the [slsReceiverUsers](#) class

- Defines in file for:
 - Default Number of receivers is 1
 - Default Start TCP port is 1954
- set default values
- get number of receivers and start tcp port from command line arguments
- Catch signal SIGINT to close files and call destructors properly
 - Ignore SIG_PIPE, prevents global signal handler, handle locally, instead of a server crashing due to client crash when writing, it just gives error
- loop over number of receivers
- fork process to create child process

- if fork failed, raise SIGINT and properly destroy all child processes
- if child process
- create `slsReceiverUsers` object with appropriate arguments

- register callbacks. remember to set file write enable to 0 (using the client)

if we should not write files and you will write data using the callbacks

- Call back for start acquisition
- Call back for acquisition finished
- start tcp server thread
- as long as keeprunning is true (changes with Ctrl+C)
- interrupt caught, delete `slsReceiverUsers` object and exit
- Parent process ignores SIGINT (exits only when all child process exits)
- Print Ready and Instructions how to exit
- Parent process waits for all child processes to exit

Definition at line 167 of file mainReceiver.cpp.

3.2.3.5 void printHelp ()

prints usage of this example program

Definition at line 55 of file mainReceiver.cpp.

3.2.3.6 void sigInterruptHandler (int p)

Control+C Interrupt Handler Sets the variable keeprunning to false, to let all the processes know to exit properly

Definition at line 48 of file mainReceiver.cpp.

3.2.3.7 int StartAcq (char * filepath, char * filename, uint64_t fileindex, uint32_t datasize, void * p)

Start Acquisition Call back slsReceiver writes data if file write enabled. Users get data to write using call back if registerCallBackRawDataReady is registered.

Parameters

<i>filepath</i>	file path
<i>filename</i>	file name
<i>fileindex</i>	file index
<i>datasize</i>	data size in bytes
<i>p</i>	pointer to object

Returns

ignored

Definition at line 72 of file mainReceiver.cpp.

3.2.4 Variable Documentation

3.2.4.1 bool keeprunning

Variable is true to continue running, set to false upon interrupt

Definition at line 42 of file mainReceiver.cpp.

3.3 /afs/psi.ch/project/sls_det_software/dhanya_softwareDevelopment/mySoft/slsDetectorPackage/sls-DetectorSoftware/slsDetector/slsDetectorUsers.h File Reference

```
#include <stdint.h>
#include <string>
```

Classes

- class [slsDetectorUsers](#)

The [slsDetectorUsers](#) class is a minimal interface class which should be instantiated by the users in their acquisition software (EPICS, spec etc.). More advanced configuration functions are not implemented and can be written in a configuration or parameters file that can be read/written.

3.4 /afs/psi.ch/project/sls_det_software/dhanya_softwareDevelopment/mySoft/slsDetectorPackage/sls-ReceiverSoftware/include/slsReceiverUsers.h File Reference

```
#include <stdio.h>
#include <stdint.h>
```

Classes

- class [slsReceiverUsers](#)

Class for implementing the SLS data receiver in the users application. Callbacks can be defined for processing and/or saving data.

Index

- ~slsDetectorUsers
 - slsDetectorUsers, [6](#)
- ~slsReceiverUsers
 - slsReceiverUsers, [29](#)
- /afs/psi.ch/project/sls_det_software/dhanya_software-Development/mySoft/slsDetectorPackage/sls-ReceiverSoftware/include/slsReceiverUsers.-h, [36](#)
- AcquisitionFinished
 - mainReceiver.cpp, [34](#)
- addFrame
 - slsDetectorUsers, [6](#)
- dataCallback
 - mainClient.cpp, [32](#)
- dumpDetectorSetup
 - slsDetectorUsers, [7](#)
- enableAngularConversion
 - slsDetectorUsers, [7](#)
- enableCountRateCorrection
 - slsDetectorUsers, [7](#)
- enableDataStreamingFromReceiver
 - slsDetectorUsers, [7](#)
- enableDataStreamingToClient
 - slsDetectorUsers, [7](#)
- enableFlatFieldCorrection
 - slsDetectorUsers, [8](#)
- enableGapPixels
 - slsDetectorUsers, [8](#)
- enablePixelMaskCorrection
 - slsDetectorUsers, [8](#)
- enableWriteToFile
 - slsDetectorUsers, [8](#)
- finalizeDataset
 - slsDetectorUsers, [8](#)
- getADC
 - slsDetectorUsers, [9](#)
- getCommand
 - slsDetectorUsers, [9](#)
- GetData
 - mainReceiver.cpp, [34](#)
- getDetectorDeveloper
 - slsDetectorUsers, [9](#)
- getDetectorFirmwareVersion
 - slsDetectorUsers, [9](#)
- getDetectorSerialNumber
 - slsDetectorUsers, [9](#)
- getDetectorSettings
 - slsDetectorUsers, [9](#), [10](#)
- getDetectorSize
 - slsDetectorUsers, [10](#)
- getDetectorSoftwareVersion
 - slsDetectorUsers, [10](#)
- getDetectorStatus
 - slsDetectorUsers, [10](#)
- getDetectorType
 - slsDetectorUsers, [10](#)
- getFileIndex
 - slsDetectorUsers, [10](#)
- getFileName
 - slsDetectorUsers, [11](#)
- getFilePath
 - slsDetectorUsers, [11](#)
- getFlatFieldCorrectionDir
 - slsDetectorUsers, [11](#)
- getFlatFieldCorrectionFile
 - slsDetectorUsers, [11](#)
- getMaximumDetectorSize
 - slsDetectorUsers, [11](#)
- getMeasuredPeriod
 - slsDetectorUsers, [11](#)
- getMeasuredSubFramePeriod
 - slsDetectorUsers, [11](#)
- getModuleFirmwareVersion
 - slsDetectorUsers, [12](#)
- getModuleSerialNumber
 - slsDetectorUsers, [12](#)
- getNMods
 - slsDetectorUsers, [12](#)
- getPositions
 - slsDetectorUsers, [12](#)
- getReceiverVersion
 - slsReceiverUsers, [30](#)
- getThisSoftwareVersion
 - slsDetectorUsers, [12](#)
- getThresholdEnergy
 - slsDetectorUsers, [12](#)
- getTimingMode
 - slsDetectorUsers, [13](#)
- initDataset
 - slsDetectorUsers, [13](#)
- keeprunning
 - mainReceiver.cpp, [35](#)
- main
 - mainClient.cpp, [32](#)
 - mainReceiver.cpp, [34](#)
- mainClient.cpp, [31](#)
 - dataCallback, [32](#)
 - main, [32](#)
- mainReceiver.cpp, [33](#)
 - AcquisitionFinished, [34](#)
 - GetData, [34](#)
 - keeprunning, [35](#)
 - main, [34](#)
 - PRINT_IN_COLOR, [33](#)

printHelp, 35
 sigInterruptHandler, 35
 StartAcq, 35
 PRINT_IN_COLOR
 mainReceiver.cpp, 33
 printHelp
 mainReceiver.cpp, 35
 putCommand
 slsDetectorUsers, 13
 readConfigurationFile
 slsDetectorUsers, 13
 receiver
 slsReceiverUsers, 31
 registerAcquisitionFinishedCallback
 slsDetectorUsers, 15
 registerCallBackAcquisitionFinished
 slsReceiverUsers, 30
 registerCallBackRawDataModifyReady
 slsReceiverUsers, 30
 registerCallBackRawDataReady
 slsReceiverUsers, 30
 registerCallBackStartAcquisition
 slsReceiverUsers, 30
 registerConnectChannelsCallback
 slsDetectorUsers, 15
 registerDataCallback
 slsDetectorUsers, 15
 registerDisconnectChannelsCallback
 slsDetectorUsers, 15
 registerGetIIOCallback
 slsDetectorUsers, 15
 registerGetPositionCallback
 slsDetectorUsers, 15
 registerGoToPositionCallback
 slsDetectorUsers, 16
 registerGoToPositionNoWaitCallback
 slsDetectorUsers, 16
 registerRawDataCallback
 slsDetectorUsers, 16
 resetFramesCaughtInReceiver
 slsDetectorUsers, 16
 retrieveDetectorSetup
 slsDetectorUsers, 16
 runStatusType
 slsDetectorUsers, 16
 sendSoftwareTrigger
 slsDetectorUsers, 17
 setAllTrimbits
 slsDetectorUsers, 17
 setBitDepth
 slsDetectorUsers, 17
 setClientDataStreamingInIP
 slsDetectorUsers, 17
 setClientDataStreamingInPort
 slsDetectorUsers, 17
 setClockDivider
 slsDetectorUsers, 18
 setDAC
 slsDetectorUsers, 18
 setDelayAfterTrigger
 slsDetectorUsers, 18
 setDetectorSize
 slsDetectorUsers, 18
 setExposurePeriod
 slsDetectorUsers, 19
 setExposureTime
 slsDetectorUsers, 19
 setFileIndex
 slsDetectorUsers, 19
 setFileName
 slsDetectorUsers, 19
 setFilePath
 slsDetectorUsers, 19
 setFlatFieldCorrectionDir
 slsDetectorUsers, 21
 setFlatFieldCorrectionFile
 slsDetectorUsers, 21
 setFlowControl10G
 slsDetectorUsers, 21
 setHighVoltage
 slsDetectorUsers, 21
 setNumberOfCycles
 slsDetectorUsers, 21
 setNumberOfFrames
 slsDetectorUsers, 22
 setNumberOfGates
 slsDetectorUsers, 22
 setNumberOfStorageCells
 slsDetectorUsers, 22
 setOnline
 slsDetectorUsers, 22
 setOverflowMode
 slsDetectorUsers, 22
 setParallelMode
 slsDetectorUsers, 24
 setPositions
 slsDetectorUsers, 24
 setReceiverDataStreamingOutIP
 slsDetectorUsers, 24
 setReceiverDataStreamingOutPort
 slsDetectorUsers, 24
 setReceiverFifoDepth
 slsDetectorUsers, 24
 setReceiverFramesDiscardPolicy
 slsDetectorUsers, 25
 setReceiverFramesPerFile
 slsDetectorUsers, 25
 setReceiverMode
 slsDetectorUsers, 25
 setReceiverOnline
 slsDetectorUsers, 25
 setReceiverPartialFramesPadding
 slsDetectorUsers, 25
 setReceiverSilentMode

- slsDetectorUsers, 26
- setSettings
 - slsDetectorUsers, 26
- setStoragecellStart
 - slsDetectorUsers, 26
- setSubFrameExposureDeadTime
 - slsDetectorUsers, 26
- setSubFrameExposureTime
 - slsDetectorUsers, 26
- setTenGigabitEthernet
 - slsDetectorUsers, 27
- setThresholdEnergy
 - slsDetectorUsers, 27
- setTimingMode
 - slsDetectorUsers, 27
- sigInterruptHandler
 - mainReceiver.cpp, 35
- slsDetectorUsers, 2
 - ~slsDetectorUsers, 6
 - addFrame, 6
 - dumpDetectorSetup, 7
 - enableAngularConversion, 7
 - enableCountRateCorrection, 7
 - enableDataStreamingFromReceiver, 7
 - enableDataStreamingToClient, 7
 - enableFlatFieldCorrection, 8
 - enableGapPixels, 8
 - enablePixelMaskCorrection, 8
 - enableWriteToFile, 8
 - finalizeDataset, 8
 - getADC, 9
 - getCommand, 9
 - getDetectorDeveloper, 9
 - getDetectorFirmwareVersion, 9
 - getDetectorSerialNumber, 9
 - getDetectorSettings, 9, 10
 - getDetectorSize, 10
 - getDetectorSoftwareVersion, 10
 - getDetectorStatus, 10
 - getDetectorType, 10
 - getFileIndex, 10
 - getFileName, 11
 - getFilePath, 11
 - getFlatFieldCorrectionDir, 11
 - getFlatFieldCorrectionFile, 11
 - getMaximumDetectorSize, 11
 - getMeasuredPeriod, 11
 - getMeasuredSubFramePeriod, 11
 - getModuleFirmwareVersion, 12
 - getModuleSerialNumber, 12
 - getNMods, 12
 - getPositions, 12
 - getThisSoftwareVersion, 12
 - getThresholdEnergy, 12
 - getTimingMode, 13
 - initDataset, 13
 - putCommand, 13
 - readConfigurationFile, 13
 - registerAcquisitionFinishedCallback, 15
 - registerConnectChannelsCallback, 15
 - registerDataCallback, 15
 - registerDisconnectChannelsCallback, 15
 - registerGetI0Callback, 15
 - registerGetPositionCallback, 15
 - registerGoToPositionCallback, 16
 - registerGoToPositionNoWaitCallback, 16
 - registerRawDataCallback, 16
 - resetFramesCaughtInReceiver, 16
 - retrieveDetectorSetup, 16
 - runStatusType, 16
 - sendSoftwareTrigger, 17
 - setAllTrimbits, 17
 - setBitDepth, 17
 - setClientDataStreamingInIP, 17
 - setClientDataStreamingInPort, 17
 - setClockDivider, 18
 - setDAC, 18
 - setDelayAfterTrigger, 18
 - setDetectorSize, 18
 - setExposurePeriod, 19
 - setExposureTime, 19
 - setFileIndex, 19
 - setFileName, 19
 - setFilePath, 19
 - setFlatFieldCorrectionDir, 21
 - setFlatFieldCorrectionFile, 21
 - setFlowControl10G, 21
 - setHighVoltage, 21
 - setNumberOfCycles, 21
 - setNumberOfFrames, 22
 - setNumberOfGates, 22
 - setNumberOfStorageCells, 22
 - setOnline, 22
 - setOverflowMode, 22
 - setParallelMode, 24
 - setPositions, 24
 - setReceiverDataStreamingOutIP, 24
 - setReceiverDataStreamingOutPort, 24
 - setReceiverFifoDepth, 24
 - setReceiverFramesDiscardPolicy, 25
 - setReceiverFramesPerFile, 25
 - setReceiverMode, 25
 - setReceiverOnline, 25
 - setReceiverPartialFramesPadding, 25
 - setReceiverSilentMode, 26
 - setSettings, 26
 - setStoragecellStart, 26
 - setSubFrameExposureDeadTime, 26
 - setSubFrameExposureTime, 26
 - setTenGigabitEthernet, 27
 - setThresholdEnergy, 27
 - setTimingMode, 27
 - slsDetectorUsers, 6
 - slsDetectorUsers, 6
 - startAcquisition, 28
 - startMeasurement, 28

- startReceiver, [28](#)
- stopAcquisition, [28](#)
- stopMeasurement, [28](#)
- stopReceiver, [28](#)
- slsReceiverUsers, [29](#)
 - ~slsReceiverUsers, [29](#)
 - getReceiverVersion, [30](#)
 - receiver, [31](#)
 - registerCallBackAcquisitionFinished, [30](#)
 - registerCallBackRawDataModifyReady, [30](#)
 - registerCallBackRawDataReady, [30](#)
 - registerCallBackStartAcquisition, [30](#)
 - slsReceiverUsers, [29](#)
 - slsReceiverUsers, [29](#)
 - start, [31](#)
 - stop, [31](#)
- start
 - slsReceiverUsers, [31](#)
- StartAcq
 - mainReceiver.cpp, [35](#)
- startAcquisition
 - slsDetectorUsers, [28](#)
- startMeasurement
 - slsDetectorUsers, [28](#)
- startReceiver
 - slsDetectorUsers, [28](#)
- stop
 - slsReceiverUsers, [31](#)
- stopAcquisition
 - slsDetectorUsers, [28](#)
- stopMeasurement
 - slsDetectorUsers, [28](#)
- stopReceiver
 - slsDetectorUsers, [28](#)