My Project

Generated by Doxygen 1.8.5

Fri Sep 28 2018 11:35:06

Contents

1 Main Page

API for SLS detectors data acquisition

Although the SLS detectors group delvelops 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 dislaying 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 setPositions (int nPos, double *pos)
  set positions for the acquisition
- int getPositions (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 calbback 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 initalize 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 calbback for accessing detector final data
- void registerGetPositionCallback (double(*func)(void *), void *arg)

  register calbback 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 calbback reading to I0
- std::string putCommand (int narg, char *args[], int pos=-1)

  sets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/sls-DetectorClientHowTo.pdf
- std::string getCommand (int narg, char *args[], int pos=-1)

  gets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/sls-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

ret	address of return value. It will be set to 0 for success, else 1 for failure
id	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

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

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

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

Parameters

fname	file to write to

Returns

OK or FAIL

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

enable/disable angular conversion

Parameters

i	0 disables, 1 enables, -1 gets

Returns

0 if angular conversion disabled, 1 if enabled

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

enable/disable count rate corrections

Parameters

i	0 disables, 1 enables with default values, -1 gets

Returns

0 if count corrections disabled, 1 if enabled

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

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

Parameters

i	0 to disable 1 to enable -1 to only get the value

Returns

data streaming from receiver enable

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

Enable data streaming to client (creates receiving sockets)

Parameters

i	0 to disable, 1 to enable, -1 to get the value

Returns

data streaming to client enable

2.1.3.7 int slsDetectorUsers::enableFlatFieldCorrection (int *i* = −1)

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

Parameters

i	0 disables, 1 enables, -1 gets

Returns

0 if ff corrections disabled, 1 if enabled

2.1.3.8 int slsDetectorUsers::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 (registerDataCallback)

Parameters

enable	1 sets, 0 unsets, -1 gets

Returns

gap pixel enable or -1 for error

2.1.3.9 int slsDetectorUsers::enablePixelMaskCorrection (int *i* = −1)

enable/disable bad channel corrections

Parameters

i	0 disables, 1 enables, -1 gets

Returns

0 if bad channels corrections disabled, 1 if enabled

2.1.3.10 int slsDetectorUsers::enableWriteToFile (int *i* = −1)

Enable write file function included

2.1.3.11 virtual void slsDetectorUsers::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

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

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

get adc value

Parameters

adc	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
id	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 slsDetectorUsers::getCommand (int *narg*, char * *args[]*, int *pos* = -1)

gets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/slsDetectorClientHowTo.pdf

Parameters

narg	value to be set
args	value to be set
pos	position of detector in multislsdetector list

Returns

answer std::string

2.1.3.14 std::string slsDetectorUsers::getDetectorDeveloper ( )

useful to define subset of working functions

Returns

"PSI" or "Dectris"

2.1.3.15 int64_t slsDetectorUsers::getDetectorFirmwareVersion ( )

get get Detector Firmware Version

Returns

id

2.1.3.16 int64_t slsDetectorUsers::getDetectorSerialNumber ( )

get get Detector Serial Number

Returns

id

2.1.3.17 static int slsDetectorUsers::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 slsDetectorUsers.h.

2.1.3.18 static std::string slsDetectorUsers::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 slsDetectorUsers.h.

2.1.3.19 int slsDetectorUsers::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 horiziontal
ny	number of channels in vertical

Returns

OK/FAIL

2.1.3.20 int64_t slsDetectorUsers::getDetectorSoftwareVersion ( )

get get Detector Software Version

Returns

id

2.1.3.21 int slsDetectorUsers::getDetectorStatus ( )

get run status

Returns

status mask

2.1.3.22 std::string slsDetectorUsers::getDetectorType ( )

useful for data plotting etc.

Returns

Mythen, Eiger, Gotthard etc.

2.1.3.23 int slsDetectorUsers::getFileIndex ( )

Returns

the default output file index

2.1.3.24 std::string slsDetectorUsers::getFileName ( )

Returns

the default output files root name

2.1.3.25 std::string slsDetectorUsers::getFilePath ( )

returns the default output files path

2.1.3.26 std::string slsDetectorUsers::getFlatFieldCorrectionDir ( )

get flat field corrections file directory

Returns

flat field correction file directory

2.1.3.27 std::string slsDetectorUsers::getFlatFieldCorrectionFile ( )

get flat field corrections file name

Returns

flat field correction file name

2.1.3.28 int slsDetectorUsers::getMaximumDetectorSize (int & *nx,* int & *ny*)

gets the maximum detector size

Parameters

nx	number of channels in horiziontal
ny	number of channels in vertical

Returns

OK/FAIL

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

get measured period between previous two frames(EIGER only)

Parameters

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

Returns

measured period

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

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

Parameters

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

Returns

measured sub period

2.1.3.31 int64_t slsDetectorUsers::getModuleFirmwareVersion ( )

get get Module Firmware Version

Returns

id

2.1.3.32 int64_t slsDetectorUsers::getModuleSerialNumber (int *imod* = -1)

get get Module Serial Number

Parameters

imod	module number

Returns

id

2.1.3.33 int slsDetectorUsers::getNMods ( )

returns total number of detector modules

Returns

the total number of detector modules

2.1.3.34 int slsDetectorUsers::getPositions (double * *pos* = NULL)

get positions for the acquisition

Parameters

pos	array which will contain the encoder positions

Returns

number of positions

2.1.3.35 int64_t slsDetectorUsers::getThisSoftwareVersion ( )

get this Software Version

Returns

id

2.1.3.36 int slsDetectorUsers::getThresholdEnergy ( )

get threshold energy

Returns

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

2.1.3.37 static std::string slsDetectorUsers::getTimingMode (int *f*) [inline], [static]

returns external communication mode std::string from index

Parameters

f	index for communication mode

Returns

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

Definition at line 879 of file slsDetectorUsers.h.

2.1.3.38 static int slsDetectorUsers::getTimingMode (std::string *s*) [inline], [static]

returns external communication mode std::string from index

Parameters

s	index for communication mode

Returns

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

Definition at line 896 of file slsDetectorUsers.h.

2.1.3.39 virtual void slsDetectorUsers::initDataset (int *refresh*) [virtual]

function to initalize 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

refresh	if 1, all parameters like ffcoefficients, badchannels, ratecorrections etc. are reset (should be called at least onece with this option), if 0 simply reset merging/ summation

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

sets parameters in command interface http://www.psi.ch/detectors/UsersSupportEN/slsDetectorClientHowTo.pdf

Parameters

narg	value to be set
args	value to be set
pos	position of detector in multislsdetector list

Returns

answer std::string

2.1.3.41 int slsDetectorUsers::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

fname	file name

Returns

OK or FAIL

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

register calbback for accessing detector final data

Parameters

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

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

register callback for connecting to the epics channels

Parameters

func	function for connecting to the epics channels
arg	argument

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

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

Parameters

userCallback	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
pArg	argument

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

register callback to disconnect the epics channels

Parameters

func	function to disconnect the epics channels
arg	argument

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

register calbback reading to I0

Parameters

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

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

register calbback for reading detector position

Parameters

func	function for reading the detector position
arg	argument

2.1.3.48 void slsDetectorUsers::registerGoToPositionCallback (int(∗)(double, void ∗) *func,* void ∗ *arg*)

register callback for moving the detector

Parameters

func	function for moving the detector
arg	argument

2.1.3.49 void slsDetectorUsers::registerGoToPositionNoWaitCallback (int(∗)(double, void ∗) *func,* void ∗ *arg*)

register callback for moving the detector without waiting

Parameters

func	function for moving the detector
arg	argument

2.1.3.50 void slsDetectorUsers::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

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

2.1.3.51 int slsDetectorUsers::resetFramesCaughtInReceiver ( )

reset frames caught in receiver should be called before startReceiver()

Returns

OK or FAIL

2.1.3.52 int slsDetectorUsers::retrieveDetectorSetup (std::string const *fname*)

Loads the detector setup from file.

Parameters

fname	file to read from

Returns

OK or FAIL

2.1.3.53 static std::string slsDetectorUsers::runStatusType (int *s*) [inline], [static]

returns std::string from run status index

Parameters

s	run status index

Returns

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

Definition at line 827 of file slsDetectorUsers.h.

2.1.3.54 int slsDetectorUsers::sendSoftwareTrigger ( )

Sends a software internal trigger (EIGER only)

Returns

0 for success, 1 for fail

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

sets all trimbits to value (only available for eiger)

Parameters

val	value to be set (-1 gets)
id	module index (-1 for all)

Returns

value set

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

set/get dynamic range

Parameters

i	dynamic range (-1 get)

Returns

current dynamic range

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

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

Parameters

ip	sets, empty std::string gets

Returns

client streaming in ZMQ IP

2.1.3.58 int slsDetectorUsers::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	sets, -1 gets

Returns

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

2.1.3.59 int slsDetectorUsers::setClockDivider (int *value*)

sets clock divider of detector

Parameters

value	value to be set (-1 gets)

Returns

speed of detector

2.1.3.60 int slsDetectorUsers::setDAC (std::string *dac,* int *val,* int *id =* -1)

set dac value

Parameters

dac	dac as std::string. can be vcmp_ll, vcmp_lr, vcmp_rl, vcmp_rr, vthreshold, vrf, vrs, vtr, vcall, vcp. others not supported
val	value to be set (-1 gets)
id	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 slsDetectorUsers::setDelayAfterTrigger (double *t =* -1, bool *inseconds =* false, int *imod =* -1)

set/get delay after trigger

Parameters

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

Returns

timer set value in ns, or s if specified

2.1.3.62 int slsDetectorUsers::setDetectorSize (int *x0 =* -1, int *y0 =* -1, int *nx =* -1, int *ny =* -1)

sets the detector size

Parameters

x0	horizontal position origin in channel number (-1 unchanged)

y0	vertical position origin in channel number (-1 unchanged)
nx	number of channels in horiziontal (-1 unchanged)
ny	number of channels in vertical (-1 unchanged)

Returns

OK/FAIL

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

set/get exposure period

Parameters

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

Returns

timer set value in ns, or s if specified

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

set/get exposure time value

Parameters

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

Returns

timer set value in ns, or s if specified

2.1.3.65 int slsDetectorUsers::setFileIndex (int *i*)

sets the default output file index

Parameters

i	file index

Returns

the default output file index

2.1.3.66 std::string slsDetectorUsers::setFileName (std::string *s*)

sets the default output files path

Parameters

s	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

s	file path

Returns

file path

2.1.3.68 std::string slsDetectorUsers::setFlatFieldCorrectionDir (std::string *dir*)

set flat field corrections file directory

Parameters

dir	flat field correction file directory

Returns

flat field correction file directory

2.1.3.69 int slsDetectorUsers::setFlatFieldCorrectionFile (std::string *fname* = " ")

set flat field correction file

Parameters

fname	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 slsDetectorUsers::setFlowControl10G (int *i* = -1)

set flow control for 10Gbe (Eiger only)

Parameters

i	1 sets, 0 unsets (-1 gets)

Returns

flow control enable for 10 Gbe

2.1.3.71 int slsDetectorUsers::setHighVoltage (int *i*)

set high voltage

Parameters

i	$>$ 0 sets, 0 unsets, (-1 gets)

Returns

high voltage

2.1.3.72 int64_t slsDetectorUsers::setNumberOfCycles (int64_t *t* = -1, int *imod* = -1)

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

Parameters

t	number of frames (-1 gets)
imod	module number (-1 for all)

Returns

number of frames

2.1.3.73 int64_t slsDetectorUsers::setNumberOfFrames (int64_t *t* = −1, int *imod* = −1)

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

Parameters

t	number of frames (-1 gets)
imod	module number (-1 for all)

Returns

number of frames

2.1.3.74 int64_t slsDetectorUsers::setNumberOfGates (int64_t *t* = −1, int *imod* = −1)

set/get number of gates

Parameters

t	number of gates (-1 gets)
imod	module number (-1 for all)

Returns

number of gates

2.1.3.75 int64_t slsDetectorUsers::setNumberOfStorageCells (int64_t *t* = −1, int *imod* = −1)

set/get number of additional storage cells (Jungfrau)

Parameters

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

Returns

number of additional storage cells

2.1.3.76 int slsDetectorUsers::setOnline (int const *online* = −1)

sets the onlineFlag

Parameters

online	can be: -1 returns wether 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 slsDetectorUsers::setOverflowMode (int *value*)

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

Parameters

value	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 slsDetectorUsers::setParallelMode (int *value*)

sets parallel mode

Parameters

value	0 for non parallel, 1 for parallel, 2 for safe mode (-1 gets)

Returns

gets parallel mode

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

set positions for the acquisition

Parameters

nPos	number of positions
pos	array with the encoder positions

Returns

number of positions

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

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

Parameters

ip	sets, empty std::string gets

Returns

receiver streaming out ZMQ IP

2.1.3.81 int slsDetectorUsers::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	sets, -1 gets

Returns

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

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

set receiver fifo depth

Parameters

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

Returns

receiver fifo depth

2.1.3.83 std::string slsDetectorUsers::setReceiverFramesDiscardPolicy (std::string *f* = "get")

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

Parameters

f	nodiscard (default),discardempty, discardpartial (fastest), get to get the value

Returns

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

2.1.3.84 int slsDetectorUsers::setReceiverFramesPerFile (int *f* = -1)

Sets the frames per file in receiver

Parameters

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

Returns

frames per file

2.1.3.85 int slsDetectorUsers::setReceiverMode (int *n* = -1)

sets the mode by which gui requests data from receiver

Parameters

n	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 slsDetectorUsers::setReceiverOnline (int const *online* = -1)

sets the receivers onlineFlag

Parameters

online	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 slsDetectorUsers::setReceiverPartialFramesPadding (int *f* = -1)

Sets the frame padding in receiver

Parameters

f	0 does not partial frames, 1 pads partial frames (-1 gets)

Returns

partial frames padding enable

2.1.3.88 int slsDetectorUsers::setReceiverSilentMode (int *i*)

set receiver in silent mode

Parameters

i	1 sets, 0 unsets (-1 gets)

Returns

silent mode of receiver

2.1.3.89 int slsDetectorUsers::setSettings (int *isettings* = `-1`)

set detector settings

Parameters

isettings	settings index (-1 gets)

Returns

current settings

2.1.3.90 int slsDetectorUsers::setStoragecellStart (int *pos* = `-1`)

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

Parameters

pos	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 slsDetectorUsers::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

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

Returns

sub frame dead time in ns, or s if specified

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

Set sub frame exposure time (only for Eiger)

Parameters

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

Returns

sub frame exposure time in ns, or s if specified

2.1.3.93 int slsDetectorUsers::setTenGigabitEthernet (int *i* = −1)

enable/disable 10GbE (Eiger only)

Parameters

i	1 sets, 0 unsets (-1 gets)

Returns

10GbE enable

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

set threshold energy

Parameters

e_eV	threshold in eV

Returns

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

2.1.3.95 int slsDetectorUsers::setThresholdEnergy (int *e_ev*, int *tb*, int *isettings* = −1, int *id* = −1)

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

Parameters

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

Returns

current threshold value in ev (-1 failed)

2.1.3.96 int slsDetectorUsers::setTimingMode (int *pol* = −1)

set/get the external communication mode

Parameters

pol	value to be set

See Also

getTimingMode

Returns

current external communication mode

2.1.3.97 int slsDetectorUsers::startAcquisition ( )

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

Returns

OK if all detectors are properly started, FAIL otherwise

2.1.3.98 void slsDetectorUsers::startMeasurement ( )

start measurement and acquires

Returns

OK/FAIL

2.1.3.99 int slsDetectorUsers::startReceiver ( )

start receiver listening mode

Returns

returns OK or FAIL

2.1.3.100 int slsDetectorUsers::stopAcquisition ( )

stop detector real time acquisition

Returns

OK if all detectors are properly started, FAIL otherwise

2.1.3.101 int slsDetectorUsers::stopMeasurement ( )

stop measurement

Returns

OK/FAIL

2.1.3.102 int slsDetectorUsers::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/slsDetectorPackage/slsDetector-Software/slsDetector/slsDetectorUsers.h

2.2 slsReceiverUsers Class Reference

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

`#include <slsReceiverUsers.h>`

Public Member Functions

- slsReceiverUsers (int argc, char ∗argv[], int &success)
- ∼slsReceiverUsers ()
- 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 calbback 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 &revDatasize, void ∗), void ∗arg)

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

Public Attributes

- slsReceiver ∗ 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.

slsReceiverUsers 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 slsReceiverUsers.h.

2.2.2 Constructor & Destructor Documentation

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

Constructor reads config file, creates socket, assigns function table

Parameters

argc	from command line
argv	from command line
success	socket creation was successfull

2.2.2.2 slsReceiverUsers::∼slsReceiverUsers ( )

Destructor

2.2.3 Member Function Documentation

2.2.3.1 int64_t slsReceiverUsers::getReceiverVersion ( )

get get Receiver Version

Returns

id

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

register callback for end of acquisition

Parameters

func	end of acquisition callback. Argument nf is total frames caught
arg	argument

Returns

nothing

2.2.3.3 void slsReceiverUsers::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

func	raw data ready callback. arguments are sls_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).
arg	argument

Returns

nothing

2.2.3.4 void slsReceiverUsers::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

func	raw data ready callback. arguments are sls_receiver_header, dataPointer, dataSize
arg	argument

Returns

nothing

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

register calbback for starting the acquisition

Parameters

func	callback to be called when starting the acquisition. Its arguments are filepath, filename, fileindex, datasize
arg	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 comminication

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/slsReceiverSoftware/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

pData	pointer to data structure received from the call back
iframe	frame number of data passed
isubframe	sub frame number of data passed (only valid for EIGER in 32 bit mode)
pArg	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 avilable
- 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 lSlsReceiver.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 recievers

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

frames	Number of frames caught
p	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

metadata	sls_receiver_header metadata
datapointer	pointer to data
datasize	data size in bytes.
p	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

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

filepath	file path
filename	file name
fileindex	file index
datasize	data size in bytes
p	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/slsDetectorSoftware/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/slsReceiverSoftware/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