HKL Reference Manual

0.2

Generated by Doxygen 1.3.9.1

Wed Nov 24 10:50:05 2004

Contents

Chapter 1

HKL Hierarchical Index

1.1 HKL Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

HKL Class Index

2.1 HKL Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

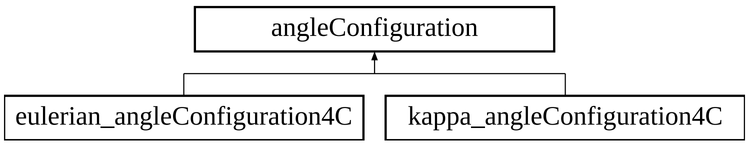
Chapter 3

HKL Class Documentation

3.1 angleConfiguration Class Reference

`#include <angleconfig.h>`

Inheritance diagram for angleConfiguration::



3.1.1 Detailed Description

The base class to represent the different kinds of configurations whatever the diffractometer type is.

Definition at line 10 of file angleconfig.h.

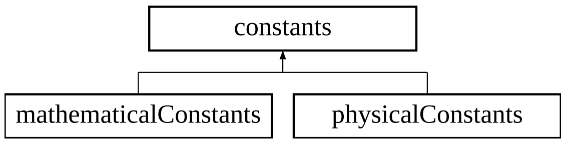
The documentation for this class was generated from the following file:

- D:/DS-sources/HKL/src/angleconfig.h

3.2 constants Class Reference

`#include <constants.h>`

Inheritance diagram for constants::



3.2.1 Detailed Description

This file contains all the constants we use in the HKL project. They are stored as static variables so we do not have to create instances of such objects.

Definition at line 11 of file constants.h.

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

- D:/DS-sources/HKL/src/constants.h

3.3 cristal Class Reference

```
#include <cristal.h>
```

Public Member Functions

- cristal (double alpha1, double alpha2, double alpha3, double beta1, double beta2, double beta3, double a1, double a2, double a3, double b1, double b2, double b3)
- cristal (double alpha1, double alpha2, double alpha3, double a1, double a2, double a3)
- cristal (const cristal &C)

  Copy constructor.

- void set (double alpha1, double alpha2, double alpha3, double a1, double a2, double a3)
- void set (const cristal &C)

  Copy a cristal.

- int check_cristal (const smatrix &B) const

Static Public Member Functions

- int test_cristals ()

3.3.1 Detailed Description

Class cristal to store direct and reciprocal lattice parameters and the matrix to move from the reciprocal lattice to the cristal cartesian system.

References :

William R. Busing and Henri A. Levy "Angle calculation for 3- and 4- Circle X-ray and Neutron Diffractometer" (1967) Acta Cryst., 22, 457-464.

A.J.C. Wilson "X-Ray Optics, The Diffraction of X-Rays By Finite and Imperfect Crystals" (1962) John Wiley & Sons Inc., 14-17.

Definition at line 21 of file cristal.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 cristal::cristal (double *alpha1*, double *alpha2*, double *alpha3*, double *beta1*, double *beta2*, double *beta3*, double *a1*, double *a2*, double *a3*, double *b1*, double *b2*, double *b3*)

Constructor to fill the class with data from both the direct and reciprocal lattice.

Definition at line 14 of file cristal.cpp.

3.3.2.2 cristal::cristal (double *alpha1*, double *alpha2*, double *alpha3*, double *a1*, double *a2*, double *a3*)

Constructor to fill the class with data from the direct lattice and compute the reciprocal parameters with computeReciprocalLattice(), then call computeB().

Definition at line 39 of file cristal.cpp.

3.3.3 Member Function Documentation

3.3.3.1 int cristal::check_cristal (const smatrix & *B*) const

Check if the matrices B are the same in both cristals.

Returns:
0 if everything is OK, -1 otherwise.

Definition at line 177 of file cristal.cpp.

Referenced by test_cristals().

3.3.3.2 void cristal::set (double *alpha1*, double *alpha2*, double *alpha3*, double *a1*, double *a2*, double *a3*)

Fill the class with data from the direct lattice and compute the reciprocal parameters with compute-ReciprocalLattice(), then call computeB().

Definition at line 119 of file cristal.cpp.

Referenced by cristal(), set(), diffractometer::setCrystal(), and test_cristals().

3.3.3.3 int cristal::test_cristals () `[static]`

Test 6 different cristals (cubic, orthorombic, hexagonal, triclinic) to make sure the computations are OK.

Returns:
0 if everything's fine, otherwise return the number of the cristal whose reciprocal lattice or matrix is wrong.

Definition at line 213 of file cristal.cpp.

References check_cristal(), and set().

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

- D:/DS-sources/HKL/src/cristal.h
- D:/DS-sources/HKL/src/cristal.cpp

3.4 diffractometer Class Reference

```
#include <diffractometer.h>
```

Public Member Functions

- virtual smatrix computeR ()=0
- virtual smatrix computeR (angleConfiguration ∗ac1)=0

  Return the matrix for the given configuration.

- virtual void setAngleConfiguration (angleConfiguration ∗ac1)=0
- virtual smatrix computeU (angleConfiguration ∗ac1, double h1, double k1, double l1, angleConfiguration ∗ac2, double h2, double k2, double l2)=0

  Compute the orientation matrix from two basic reflections.

- virtual smatrix computeU (reflection &r1, reflection &r2)=0

  Compute the orientation matrix from two basic reflections.

- virtual void setMode (mode::diffractometer_mode currentMode)=0

  Change the current computational mode.

- smatrix get_U () const

  Return the orientation matrix.

- smatrix get_UB () const

  Return the product of the orientation matrix by the crystal matrix.

- double getReflection_h (int) const

  Get h from the array of experimental reflections.

- double getReflection_k (int) const

  Get k from the array of experimental reflections.

- double getReflection_l (int) const

  Get l from the array of experimental reflections.

- void setCrystal (double alpha1, double alpha2, double alpha3, double a1, double a2, double a3)

  Change the crystal from the direct lattice parameters.

- void setCrystal (const cristal &C)
- void setWaveLength (double wl)

Protected Member Functions

- diffractometer (cristal currentCristal, source currentSource, reflection &reflection1, reflection &reflection2)
- diffractometer (cristal currentCristal, source currentSource)
- diffractometer ()

Protected Attributes

- smatrix m_U

  The orientation matrix.

- smatrix m_UB

  Product $U * B$ where B defines the crystal matrix.

- mode * m_currentMode

  The mode describes the way we use the diffractometer.

- source m_currentSource

  The light source and its wave length.

- cristal m_currentCristal

  The crystal direct and reciprocal parameters and B.

- reflection * m_reflectionList

  The array to store up to 100 experiment results.

- const int m_sizeOfArray

  Size of the reflection array.

- int m_numberOfInsertedElements

  The number of reflections inserted into m_reflectionList.

- angleConfiguration * m_currentConfiguration

  The current diffractometer angle configuration.

3.4.1 Detailed Description

The abstract base class to define all different kinds of diffractometers.

Definition at line 17 of file diffractometer.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 diffractometer::diffractometer (cristal *currentCristal*, source *currentSource*, reflection & *reflection1*, reflection & *reflection2*) `[protected]`

Commun constructor

- protected to make sure this class is abstract.

Definition at line 22 of file diffractometer.cpp.

References reflection::get_h(), reflection::get_k(), reflection::get_l(), reflection::getAngleConfiguration(), reflection::getRelevance(), m_currentConfiguration, and m_reflectionList.

3.4.2.2 diffractometer::diffractometer (cristal *currentCristal*, source *currentSource*) `[protected]`

Constructor designed for testing purposes

- protected to make sure this class is abstract.

Definition at line 69 of file diffractometer.cpp.

References m_reflectionList.

3.4.2.3 diffractometer::diffractometer () `[protected]`

Empty constructor

- protected to make sure this class is abstract.

Definition at line 59 of file diffractometer.cpp.

References m_reflectionList, m_UB, and reflection::set().

3.4.3 Member Function Documentation

3.4.3.1 virtual smatrix diffractometer::computeR () `[pure virtual]`

Return the matrix describing a complex rotation involving all the diffractometer circles. Return the matrix for the current configuration.

3.4.3.2 virtual void diffractometer::setAngleConfiguration (angleConfiguration ∗ *ac1*) `[pure virtual]`

Set the angle configuration and compute the corresponding rotation matrices according to the chosen rotation axes.

3.4.3.3 void diffractometer::setCrystal (const cristal & *C*)

Change the crystal where the reciprocal lattice and matrix have already been computed.

Definition at line 154 of file diffractometer.cpp.

References m_currentCristal, and cristal::set().

3.4.3.4 void diffractometer::setWaveLength (double *wl*)

Change the light source wave length as it is something usual in an experiment.

Definition at line 159 of file diffractometer.cpp.

References m_currentSource, and source::setWaveLength().

3.4.4 Member Data Documentation

3.4.4.1 smatrix diffractometer::m_U `[protected]`

The orientation matrix.

This orthogonal matrix relates the crystal cartesian system to the phi-axis system. It is computed from at least two relevant reflections.

Definition at line 91 of file diffractometer.h.

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

- D:/DS-sources/HKL/src/diffractometer.h
- D:/DS-sources/HKL/src/diffractometer.cpp

3.5 Error Class Reference

The HKL exception abstraction base class.

`#include <HKLException.h>`

Public Member Functions

- Error (void)
- Error (const char ∗reason, const char ∗desc, const char ∗origin, int severity=ERR)
- Error (const std::string &reason, const std::string &desc, const std::string &origin, int severity=ERR)
- Error (const Error &src)
- virtual ∼Error (void)
- Error & operator= (const Error &_src)

Public Attributes

- std::string reason
- std::string desc
- std::string origin
- int severity

3.5.1 Detailed Description

The HKL exception abstraction base class.

Definition at line 40 of file HKLException.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Error::Error (void)

Initialization.

Definition at line 26 of file HKLException.cpp.

3.5.2.2 Error::Error (const char ∗ *reason*, const char ∗ *desc*, const char ∗ *origin*, int *severity* = ERR)

Initialization.

Definition at line 38 of file HKLException.cpp.

3.5.2.3 Error::Error (const std::string & *reason*, const std::string & *desc*, const std::string & *origin*, int *severity* = ERR)

Initialization.

Definition at line 53 of file HKLException.cpp.

3.5.2.4 Error::Error (const Error & *src*)

Copy constructor.

Definition at line 68 of file HKLException.cpp.

3.5.2.5 Error::~Error (void) `[virtual]`

Error details: code

Definition at line 80 of file HKLException.cpp.

3.5.3 Member Function Documentation

3.5.3.1 Error & Error::operator= (const Error & *_src*)

operator=

Definition at line 88 of file HKLException.cpp.

References desc, origin, reason, and severity.

3.5.4 Member Data Documentation

3.5.4.1 std::string Error::desc

Error details: description

Definition at line 89 of file HKLException.h.

Referenced by operator=().

3.5.4.2 std::string Error::origin

Error details: origin

Definition at line 94 of file HKLException.h.

Referenced by operator=().

3.5.4.3 std::string Error::reason

Error details: reason

Definition at line 84 of file HKLException.h.

Referenced by operator=().

3.5.4.4 int Error::severity

Error details: severity

Definition at line 99 of file HKLException.h.

Referenced by operator=().

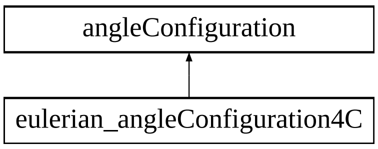
The documentation for this class was generated from the following files:

- D:/DS-sources/HKL/src/HKLException.h
- D:/DS-sources/HKL/src/HKLException.cpp

3.6 eulerian_angleConfiguration4C Class Reference

`#include <angleconfig.h>`

Inheritance diagram for eulerian_angleConfiguration4C::



Public Member Functions

- eulerian_angleConfiguration4C ()

  Empty constructor which sets everything to zero.

- eulerian_angleConfiguration4C (double, double, double, double)

  Constructor with an already made configuration.

- eulerian_angleConfiguration4C (double o, double c, double p, double t, double oi, double os, double ci, double cs, double pi, double ps, double ti, double ts)
- angleConfiguration ∗ makeCopy () const

  This redefined function builds a copy of the class.

- void printOnScreen ()

  Print the angle values in radians.

- void printDegreesOnScreen ()

  Print the angle values in degrees.

- void printStaticOnScreen ()

  Print only static fields.

Protected Attributes

- double m_omega

  The first angle in an eulerian 4-circle diffractometer.

- double m_chi

  The second angle in an eulerian 4-circle diffractometer.

- double m_phi

  The third angle in an eulerian 4-circle diffractometer.

- double m_2theta

  The detector angle in an eulerian 4-circle diffractometer.

Static Protected Attributes

- double m_omegaInf = 0.

  The first angle lower bound.

- double m_omegaSup = 0.

  The first angle upper bound.

- double m_chiInf = 0.

  The second angle lower bound.

- double m_chiSup = 0.

  The second angle upper bound.

- double m_phiInf = 0.

  The third angle lower bound.

- double m_phiSup = 0.

  The third angle upper bound.

- double m_2thetaInf = 0.

  The detector angle lower bound.

- double m_2thetaSup = 0.

  The detector angle upper bound.

3.6.1 Detailed Description

The 4C eulerian diffractometer is defined by a set of 3 angles omega, chi and phi to move the cristal and also a fourth angle to move the detector 2theta. Angles are in radians.

Definition at line 26 of file angleconfig.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 eulerian_angleConfiguration4C::eulerian_angleConfiguration4C (double *o*, double *c*, double *p*, double *t*, double *oi*, double *os*, double *ci*, double *cs*, double *pi*, double *ps*, double *ti*, double *ts*)

Constructor with an already made Configuration and the angle intervals.

Definition at line 56 of file eulerian_angleconfiguration4C.cpp.

References m_2theta, m_2thetaInf, m_2thetaSup, m_chi, m_chiInf, m_chiSup, m_omega, m_omegaInf, m_omegaSup, m_phi, m_phiInf, and m_phiSup.

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

- D:/DS-sources/HKL/src/angleconfig.h
- D:/DS-sources/HKL/src/eulerian_angleconfiguration4C.cpp

3.7 HKLException Class Reference

The HKL exception abstraction base class.

`#include <HKLException.h>`

Public Member Functions

- HKLException (void)
- HKLException (const char ∗reason, const char ∗desc, const char ∗origin, int severity=ERR)
- HKLException (const std::string &reason, const std::string &desc, const std::string &origin, int severity=ERR)
- HKLException (const Error &error)
- HKLException (const HKLException &src)
- HKLException & operator= (const HKLException &_src)
- virtual ∼HKLException (void)
- void push_error (const char ∗reason, const char ∗desc, const char ∗origin, int severity=ERR)
- void push_error (const std::string &reason, const std::string &desc, const std::string &origin, int severity=ERR)
- void push_error (const Error &error)

Public Attributes

- ErrorList errors

3.7.1 Detailed Description

The HKL exception abstraction base class.

Definition at line 115 of file HKLException.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 HKLException::HKLException (void)

Initialization.

Definition at line 106 of file HKLException.cpp.

References push_error().

3.7.2.2 HKLException::HKLException (const char ∗ *reason*, const char ∗ *desc*, const char ∗ *origin*, int *severity* = ERR)

Initialization.

Definition at line 115 of file HKLException.cpp.

References push_error().

3.7.2.3 HKLException::HKLException (const std::string & *reason*, const std::string & *desc*, const std::string & *origin*, int *severity* = `ERR`)

Initialization.

Definition at line 127 of file HKLException.cpp.

References push_error().

3.7.2.4 HKLException::HKLException (const Error & *error*)

Initialization.

3.7.2.5 HKLException::HKLException (const HKLException & *src*)

Copy constructor.

Definition at line 139 of file HKLException.cpp.

References errors, and push_error().

3.7.2.6 HKLException::~HKLException (void) `[virtual]`

Release resources.

Definition at line 169 of file HKLException.cpp.

References errors.

3.7.3 Member Function Documentation

3.7.3.1 HKLException & HKLException::operator= (const HKLException & *_src*)

operator=

Definition at line 150 of file HKLException.cpp.

References errors, and push_error().

3.7.3.2 void HKLException::push_error (const Error & *error*)

Push the specified error into the errors list.

Definition at line 200 of file HKLException.cpp.

References errors.

3.7.3.3 void HKLException::push_error (const std::string & *reason*, const std::string & *desc*, const std::string & *origin*, int *severity* = `ERR`)

Push the specified error into the errors list.

Definition at line 189 of file HKLException.cpp.

References errors.

3.7.3.4 void HKLException::push_error (const char * *reason*, const char * *desc*, const char * *origin*, int *severity* = ERR)

Push the specified error into the errors list.

Definition at line 178 of file HKLException.cpp.

References errors.

Referenced by HKLException(), and operator=().

3.7.4 Member Data Documentation

3.7.4.1 ErrorList HKLException::errors

The errors list

Definition at line 185 of file HKLException.h.

Referenced by HKLException(), operator=(), push_error(), and ~HKLException().

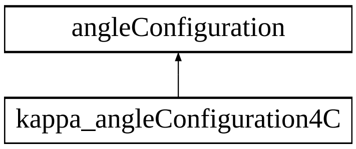
The documentation for this class was generated from the following files:

- D:/DS-sources/HKL/src/HKLException.h
- D:/DS-sources/HKL/src/HKLException.cpp

3.8 kappa_angleConfiguration4C Class Reference

`#include <angleconfig.h>`

Inheritance diagram for kappa_angleConfiguration4C::



Public Member Functions

- kappa_angleConfiguration4C ()

  Empty constructor which sets everything to zero.

- kappa_angleConfiguration4C (double, double, double, double)

  Constructor with an already made Configuration.

- kappa_angleConfiguration4C (double o, double c, double p, double t, double oi, double os, double ci, double cs, double pi, double ps, double ti, double ts)
- angleConfiguration ∗ makeCopy () const

  This redefined function builds a copy of the class.

- void printStaticOnScreen ()

  Print only static fields.

Protected Attributes

- double m_omega

  The four angles.

Static Protected Attributes

- double m_omegaInf = 0.

  The intervals associated to the angles.

3.8.1 Detailed Description

A space position in a 4C Kappa diffractometer is also defined by a set of of three angles omega, kappa and phi but the geometry axes are different. The fourth angle to move the detector is 2theta. Angles are in radians.

Definition at line 112 of file angleconfig.h.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 kappa_angleConfiguration4C::kappa_angleConfiguration4C (double *o*, double *c*, double *p*, double *t*, double *oi*, double *os*, double *ci*, double *cs*, double *pi*, double *ps*, double *ti*, double *ts*)

Constructor with an already made configuration and the angle intervals.

Definition at line 54 of file kappa_angleconfiguration4C.cpp.

References m_omega, and m_omegaInf.

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

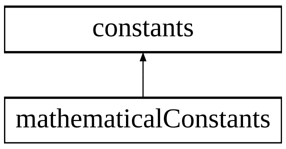
- D:/DS-sources/HKL/src/angleconfig.h
- D:/DS-sources/HKL/src/kappa_angleconfiguration4C.cpp

3.9 mathematicalConstants Class Reference

Store all the basic mathematical constants we need.

`#include <constants.h>`

Inheritance diagram for mathematicalConstants::



Static Protected Attributes

- double m_PI = 3.14159265358979323846

  The usual value of pi 3.14159265358979323846.

- double m_EPSILON_0 = 1.e-6

  The first precision factor.

- double m_EPSILON_1 = 1.e-10

  The second precision factor.

- double m_convertAnglesToDegrees = 57.2957795130823208

  To convert an angle in degrees (180 / PI).

- double m_convertAnglesToRadians = 0.0174532925199433

  To convert an angle in radians (PI / 180).

3.9.1 Detailed Description

Store all the basic mathematical constants we need.

Definition at line 40 of file constants.h.

3.9.2 Member Data Documentation

3.9.2.1 double mathematicalConstants::m_convertAnglesToDegrees = 57.2957795130823208 `[static, protected]`

To convert an angle in degrees (180 / PI).

All the computations are performed in radians, however if we want to have them in degrees (to print them out for example) we just need to multiply them by its value.

Definition at line 21 of file constants.cpp.

3.9.2.2 double mathematicalConstants::m_EPSILON_0 = 1.e-6 `[static, protected]`

The first precision factor.

This precision factor is used to test if two angles are the same.

Definition at line 11 of file constants.cpp.

3.9.2.3 double mathematicalConstants::m_EPSILON_1 = 1.e-10 `[static, protected]`

The second precision factor.

This precision factor is used to test if a double precision number is null.

Definition at line 15 of file constants.cpp.

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

- D:/DS-sources/HKL/src/constants.h
- D:/DS-sources/HKL/src/constants.cpp

3.10 mode Class Reference

`#include <mode.h>`

Public Member Functions

- virtual angleConfiguration ∗ computeAngles_Rafin (double h, double k, double l, const smatrix &UB, double lambda) const =0

    Designed for testing implementing Rafin algorithm.

3.10.1 Detailed Description

This file defines the mode telling how to use the diffractometer.

abstract class mode | abstract class eulerian_mode ——— | | class eulerian_bissectorMode4C & eulerian_-bissectorMode4C

abstract class mode | abstract class kappa_mode ———— | | class kappa_bissectorMode4C & kappa_-bissectorMode4C

Definition at line 24 of file mode.h.

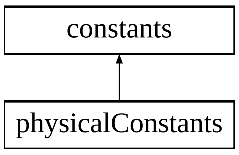
The documentation for this class was generated from the following files:

- D:/DS-sources/HKL/src/mode.h
- D:/DS-sources/HKL/src/eulerian_bissectormode4C.cpp

3.11 physicalConstants Class Reference

`#include <constants.h>`

Inheritance diagram for physicalConstants::



Static Protected Attributes

- double m_tau = 1.

3.11.1 Detailed Description

Store all the basic physical constants we need to define conventions.

Definition at line 20 of file constants.h.

3.11.2 Member Data Documentation

3.11.2.1 double physicalConstants::m_tau = 1. `[static, protected]`

We have to deal with two different conventions, if (a1,a2,a3) is the direct lattice and (b1,b2,b3) the reciprocal one, a1 $*$ b1 = 1 or a1 $*$ b1 = 2PI so we introduce tau = 1 or 2PI according to the user's choice.

Definition at line 3 of file constants.cpp.

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

- D:/DS-sources/HKL/src/constants.h
- D:/DS-sources/HKL/src/constants.cpp

3.12 reflection Class Reference

```
#include <reflection.h>
```

Public Types

- enum relevance

Public Member Functions

- reflection (angleConfiguration ∗this_angleConfiguration, double h, double k, double l, relevance this_relevance)
- double computeAngle (double h2, double k2, double l2) const
- void set (angleConfiguration ∗this_angleConfiguration, double h, double k, double l, relevance this_-relevance)

Static Public Member Functions

- double test_computeAngle ()

  Designed to test computeAngle().

3.12.1 Detailed Description

The class reflection defines a configuration where a diffraction occurs. It is defined by a set of angles, the 3 integers associated to the reciprocal lattice and its relevance to make sure we only take into account significant reflections.

Definition at line 12 of file reflection.h.

3.12.2 Member Enumeration Documentation

3.12.2.1 enum reflection::relevance

The enumeration "relevance" to make sure we only take into account significant reflections.

Definition at line 17 of file reflection.h.

3.12.3 Constructor & Destructor Documentation

3.12.3.1 reflection::reflection (angleConfiguration ∗ *this_angleConfiguration*, double *h*, double *k*, double *l*, relevance *this_relevance*)

Make a copy of the angle configuration to make sure we don’t share it in memory.

Definition at line 19 of file reflection.cpp.

References angleConfiguration::makeCopy().

3.12.4 Member Function Documentation

3.12.4.1 double reflection::computeAngle (double *h2*, double *k2*, double *l2*) const

Compute the angle between two reflections to get an idea about their level of relevance (return the absolute value). As an example it can detect if (m_h, m_k, m_l) and (h2, k2, l2) are parallel.

Definition at line 47 of file reflection.cpp.

Referenced by test_computeAngle().

3.12.4.2 void reflection::set (angleConfiguration ∗ *this_angleConfiguration*, double *h*, double *k*, double *l*, relevance *this_relevance*)

Make a copy of the angle configuration to make sure we don't share it in memory.

Definition at line 32 of file reflection.cpp.

References angleConfiguration::makeCopy().

Referenced by diffractometer::diffractometer().

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

- D:/DS-sources/HKL/src/reflection.h
- D:/DS-sources/HKL/src/reflection.cpp

3.13 source Class Reference

The class source to define a light ray and its main characteristics.

```
#include <source.h>
```

3.13.1 Detailed Description

The class source to define a light ray and its main characteristics.

Definition at line 6 of file source.h.

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

- D:/DS-sources/HKL/src/source.h
- D:/DS-sources/HKL/src/source.cpp

Index