AARE

Generated by Doxygen 1.9.7

Chapter 1

aare

Data analysis library for PSI hybrid detectors

1.1 Folder structure

Folder	subfolder	Content
include/	aare/	top level header/s
core/	include/	public headers for core
	src/	source files and non public headers

1.2 file_io class diagram

1.3 Test the zmq socket with a detector simulator

1. Download and build the slsDetectorPackage

```
git clone https://github.com/slsdetectorgroup/slsDetectorPackage.git --branch=8.0.1 #or the desired branch
cd slsDetectorPackage
mkdir build && cd build
cmake .. -DSLS_USE_SIMULATOR=ON
make -j8 #or your number of cores
```

2. Launch the slsReceiver

```
bin/slsReceiver
```

3. Launch the virtual server

```
bin/jungfrauDetectorServer_virtual
```

4 Configure the detector simulator

```
#sample config file is in etc/ in the aare repo
sls_detector_put config etc/virtual_jf.config

#Now you can take images using sls_detector_acquire
sls_detector_acquire
```

5. Run the zmq example

```
examples/zmq_example

#Will print the headers fof the frames received
```

1.4 Test the zmq processing replaying data

To be implemented

1.5 generate documentation

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 aare Namespace Reference

Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Namespaces

- namespace File
- namespace Frame
- namespace logger
- namespace network_io
- namespace NumpyHelpers

Data Structures

- class CircularFifo
- class ClusterFinder
- class DType
- class File

  RAII File class for reading and writing image files in various formats wrapper on a FileInterface to abstract the underlying file format.
- struct FileConfig

  FileConfig structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.
- class FileInterface

  FileInterface class to define the interface for file operations.
- class Frame
- class NDArray
- class NDView
- class NumpyFile

  NumpyFile class to read and write numpy files.
- struct NumpyHeader
- class RawFile

  RawFile class to read .raw and .json files.
- struct RawFileConfig

- struct sls_detector_header
- class SubFile

  Class to read a subfile from a RawFile.
- struct xy
- struct ZmqFrame

  ZmqFrame structure wrapper class to contain a ZmqHeader and a Frame.
- struct ZmqHeader
- class ZmqSocket
- class ZmqSocketReceiver
- class ZmqSocketSender

Typedefs

- using dynamic_shape = std::vector< ssize_t >
- using DataTypeVariants = std::variant< uint16_t, uint32_t >
- template<ssize_t Ndim>
  using Shape = std::array< ssize_t, Ndim >
- using shape_t = std::vector< size_t >

Enumerations

- enum class DetectorType {
  Jungfrau , Eiger , Mythen3 , Moench ,
  ChipTestBoard }
- enum class TimingMode { Auto , Trigger }
- enum class endian { little = __ORDER_LITTLE_ENDIAN__ , big = __ORDER_BIG_ENDIAN__ , native = __BYTE_ORDER__ }

Functions

- template<class T >
  T StringTo (std::string sv)
- template<class T >
  std::string toString (T sv)
- template<> DetectorType StringTo (std::string)
- template<> std::string toString (DetectorType type)
- template<> TimingMode StringTo (std::string)
- template<typename T , ssize_t Ndim>
  void save (NDArray< T, Ndim > &img, std::string pathname)
- template<typename T , ssize_t Ndim>
  NDArray< T, Ndim > load (const std::string &pathname, std::array< ssize_t, Ndim > shape)
- template<ssize_t Ndim>
  Shape< Ndim > make_shape (const std::vector< size_t > &shape)
- template<ssize_t Dim = 0, typename Strides >
  ssize_t element_offset (const Strides &)
- template<ssize_t Dim = 0, typename Strides , typename... Ix>
  ssize_t element_offset (const Strides &strides, ssize_t i, Ix... index)
- template<ssize_t Ndim>
  std::array< ssize_t, Ndim > c_strides (const std::array< ssize_t, Ndim > &shape)
- template<ssize_t Ndim>
  std::array< ssize_t, Ndim > make_array (const std::vector< ssize_t > &vec)
- template<> std::string toString (DetectorType type)
- template<> DetectorType StringTo (std::string name)
- template<> TimingMode StringTo (std::string mode)

6.1.1 Detailed Description

Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

6.1.2 Typedef Documentation

6.1.2.1 DataTypeVariants

```
using aare::DataTypeVariants = typedef std::variant<uint16_t, uint32_t>
```

6.1.2.2 dynamic_shape

```
using aare::dynamic_shape = typedef std::vector<ssize_t>
```

6.1.2.3 Shape

```
template<ssize_t Ndim>
using aare::Shape = typedef std::array<ssize_t, Ndim>
```

6.1.2.4 shape_t

```
using aare::shape_t = typedef std::vector<size_t>
```

6.1.3 Enumeration Type Documentation

6.1.3.1 DetectorType

```
enum class aare::DetectorType  [strong]
```

Enumerator

Jungfrau	
Eiger	
Mythen3	
Moench	
ChipTestBoard	

6.1.3.2 endian

```
enum class aare::endian  [strong]
```

Enumerator

little	
big	
native	

6.1.3.3 TimingMode

```
enum class aare::TimingMode  [strong]
```

Enumerator

Auto	
Trigger	

6.1.4 Function Documentation

6.1.4.1 c_strides()

```
template<ssize_t Ndim>
std::array< ssize_t, Ndim > aare::c_strides (
            const std::array< ssize_t, Ndim > & shape )
```

6.1.4.2 element_offset() [1/2]

```
template<ssize_t Dim = 0, typename Strides >
ssize_t aare::element_offset (
            const Strides &  )
```

6.1.4.3 element_offset() [2/2]

```
template<ssize_t Dim = 0, typename Strides , typename...  Ix>
ssize_t aare::element_offset (
            const Strides & strides,
            ssize_t i,
            Ix...  index )
```

6.1.4.4 load()

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::load (
            const std::string & pathname,
            std::array< ssize_t, Ndim > shape )
```

6.1.4.5 make_array()

```
template<ssize_t Ndim>
std::array< ssize_t, Ndim > aare::make_array (
            const std::vector< ssize_t > & vec )
```

6.1.4.6 make_shape()

```
template<ssize_t Ndim>
Shape< Ndim > aare::make_shape (
            const std::vector< size_t > & shape )
```

6.1.4.7 save()

```
template<typename T , ssize_t Ndim>
void aare::save (
            NDArray< T, Ndim > & img,
            std::string pathname )
```

6.1.4.8 StringTo() [1/5]

```
template<>
TimingMode aare::StringTo (
            std::string mode )
```

6.1.4.9 StringTo() [2/5]

```
template<>
DetectorType aare::StringTo (
            std::string name )
```

6.1.4.10 StringTo() [3/5]

```
template<class T >
T aare::StringTo (
            std::string sv )
```

6.1.4.11 StringTo() [4/5]

```
template<>
DetectorType aare::StringTo (
            std::string name )
```

6.1.4.12 StringTo() [5/5]

```
template<>
TimingMode aare::StringTo (
             std::string name )
```

6.1.4.13 toString() [1/3]

```
template<>
std::string aare::toString (
             DetectorType type )
```

6.1.4.14 toString() [2/3]

```
template<>
std::string aare::toString (
             DetectorType type )
```

6.1.4.15 toString() [3/3]

```
template<class T >
std::string aare::toString (
             T sv )
```

6.2 aare.File Namespace Reference

Data Structures

- class File

6.3 aare.Frame Namespace Reference

Data Structures

- class Frame

6.4 aare::logger Namespace Reference

Namespaces

- namespace internal

Data Structures

- class Logger

Enumerations

- enum LOGGING_LEVEL { DEBUG = 0 , INFO = 1 , WARNING = 2 , ERROR = 3 }

Functions

- template<LOGGING_LEVEL level, typename... Strings>
  void log (const Strings... s)
- template<typename... Strings>
  void debug (const Strings... s)
- template<typename... Strings>
  void info (const Strings... s)
- template<typename... Strings>
  void warn (const Strings... s)
- template<typename... Strings>
  void error (const Strings... s)
- void set_streams (std::streambuf *out, std::streambuf *err)
- void set_streams (std::streambuf *out)
- void set_verbosity (LOGGING_LEVEL level)
- void set_output_file (std::string filename)
- Logger & get_logger_instance ()

6.4.1 Enumeration Type Documentation

6.4.1.1 LOGGING_LEVEL

```
enum aare::logger::LOGGING_LEVEL
```

Enumerator

DEBUG	
INFO	
WARNING	
ERROR	

6.4.2 Function Documentation

6.4.2.1 debug()

```
template<typename...  Strings>
void aare::logger::debug (
             const Strings...  s )
```

6.4.2.2 error()

```
template<typename...  Strings>
void aare::logger::error (
             const Strings...  s )
```

6.4.2.3 get_logger_instance()

```
Logger & aare::logger::get_logger_instance ( )
```

6.4.2.4 info()

```
template<typename...  Strings>
void aare::logger::info (
             const Strings...  s )
```

6.4.2.5 log()

```
template<LOGGING_LEVEL level, typename...  Strings>
void aare::logger::log (
             const Strings...  s )
```

6.4.2.6 set_output_file()

```
void aare::logger::set_output_file (
             std::string filename )
```

6.4.2.7 set_streams() [1/2]

```
void aare::logger::set_streams (
             std::streambuf * out )
```

6.4.2.8 set_streams() [2/2]

```
void aare::logger::set_streams (
             std::streambuf * out,
             std::streambuf * err )
```

6.4.2.9 set_verbosity()

```
void aare::logger::set_verbosity (
             LOGGING_LEVEL level )
```

6.4.2.10 warn()

```
template<typename...  Strings>
void aare::logger::warn (
             const Strings...  s )
```

6.5 aare::logger::internal Namespace Reference

Variables

- aare::logger::Logger logger_instance = aare::logger::Logger()

6.5.1 Variable Documentation

6.5.1.1 logger_instance

```
aare::logger::Logger aare::logger::internal::logger_instance = aare::logger::Logger()
```

6.6 aare::network_io Namespace Reference

Data Structures

- class NetworkError

    NetworkError exception class.

6.7 aare::NumpyHelpers Namespace Reference

Functions

- std::string parse_str (const std::string &in)
- std::string trim (const std::string &str)
- std::vector< std::string > parse_tuple (std::string in)
- bool parse_bool (const std::string &in)
- std::string get_value_from_map (const std::string &mapstr)
- std::unordered_map< std::string, std::string > parse_dict (std::string in, const std::vector< std::string > &keys)
- template<typename T , size_t N>
  bool in_array (T val, const std::array< T, N > &arr)
- bool is_digits (const std::string &str)
- aare::DType parse_descr (std::string typestring)
- size_t write_header (std::filesystem::path fname, const NumpyHeader &header)
- size_t write_header (std::ostream &out, const NumpyHeader &header)
- void write_magic (std::ostream &ostream, int version_major, int version_minor)
- template<typename T >
  std::string write_tuple (const std::vector< T > &v)
- std::string write_boolean (bool b)
- std::string write_header_dict (const std::string &descr, bool fortran_order, const shape_t &shape)

Variables

- const constexpr std::array< char, 6 > magic_str {'\x93', 'N', 'U', 'M', 'P', 'Y'}
- const uint8_t magic_string_length {6}

6.7.1 Function Documentation

6.7.1.1 get_value_from_map()

```
std::string aare::NumpyHelpers::get_value_from_map (
             const std::string & mapstr )
```

6.7.1.2 in_array()

```
template<typename T , size_t N>
bool aare::NumpyHelpers::in_array (
             T val,
             const std::array< T, N > & arr )
```

6.7.1.3 is_digits()

```
bool aare::NumpyHelpers::is_digits (
             const std::string & str )
```

6.7.1.4 parse_bool()

```
bool aare::NumpyHelpers::parse_bool (
             const std::string & in )
```

6.7.1.5 parse_descr()

```
aare::DType aare::NumpyHelpers::parse_descr (
             std::string typestring )
```

6.7.1.6 parse_dict()

```
std::unordered_map< std::string, std::string > aare::NumpyHelpers::parse_dict (
             std::string in,
             const std::vector< std::string > & keys )
```

6.7.1.7 parse_str()

```
std::string aare::NumpyHelpers::parse_str (
             const std::string & in )
```

6.7.1.8 parse_tuple()

```
std::vector< std::string > aare::NumpyHelpers::parse_tuple (
             std::string in )
```

6.7.1.9 trim()

```
std::string aare::NumpyHelpers::trim (
             const std::string & str )
```

Removes leading and trailing whitespaces

6.7.1.10 write_boolean()

```
std::string aare::NumpyHelpers::write_boolean (
             bool b ) [inline]
```

6.7.1.11 write_header() [1/2]

```
size_t aare::NumpyHelpers::write_header (
             std::filesystem::path fname,
             const NumpyHeader & header )
```

6.7.1.12 write_header() [2/2]

```
size_t aare::NumpyHelpers::write_header (
             std::ostream & out,
             const NumpyHeader & header )
```

6.7.1.13 write_header_dict()

```
std::string aare::NumpyHelpers::write_header_dict (
             const std::string & descr,
             bool fortran_order,
             const shape_t & shape ) [inline]
```

6.7.1.14 write_magic()

```
void aare::NumpyHelpers::write_magic (
             std::ostream & ostream,
             int version_major,
             int version_minor )
```

6.7.1.15 write_tuple()

```
template<typename T >
std::string aare::NumpyHelpers::write_tuple (
            const std::vector< T > & v )  [inline]
```

6.7.2 Variable Documentation

6.7.2.1 magic_str

```
const constexpr std::array<char, 6> aare::NumpyHelpers::magic_str {'\x93', 'N', 'U', 'M', 'P',
'Y'}  [constexpr]
```

6.7.2.2 magic_string_length

```
const uint8_t aare::NumpyHelpers::magic_string_length {6}
```

6.8 example Namespace Reference

Namespaces

- namespace read_frame

6.9 example.read_frame Namespace Reference

Variables

- root_dir = Path(os.environ.get(¨PROJECT_ROOT_DIR¨))
- data_path = str(root_dir / ¨data¨/¨jungfrau_single_master_0.json¨)
- file = File(data_path)
- frame = file.get_frame(0)
- arr = np.array(frame.get_array())

6.9.1 Variable Documentation

6.9.1.1 arr

```
example.read_frame.arr = np.array(frame.get_array())
```

6.9.1.2 data_path

```
example.read_frame.data_path = str(root_dir / ¨data¨/¨jungfrau_single_master_0.json¨)
```

6.9.1.3 file

```
example.read_frame.file = File(data_path)
```

6.9.1.4 frame

```
example.read_frame.frame = file.get_frame(0)
```

6.9.1.5 root_dir

```
example.read_frame.root_dir = Path(os.environ.get(¨PROJECT_ROOT_DIR¨))
```

6.10 folly Namespace Reference

Data Structures

- struct ProducerConsumerQueue

6.11 read_first_frame_number Namespace Reference

Variables

- header_dt
- frame_number = np.fromfile(f, dtype=header_dt, count=1)[¨Frame Number¨][0]

6.11.1 Variable Documentation

6.11.1.1 frame_number

```
read_first_frame_number.frame_number = np.fromfile(f, dtype=header_dt, count=1)[¨Frame Number¨][0]
```

6.11.1.2 header_dt

```
read_first_frame_number.header_dt
```

Initial value:

```
=  np.dtype(
    [
        ("Frame Number", "u8"),
        ("SubFrame Number/ExpLength", "u4"),
        ("Packet Number", "u4"),
        ("Bunch ID", "u8"),
        ("Timestamp", "u8"),
        ("Module Id", "u2"),
        ("Row", "u2"),
        ("Column", "u2"),
        ("Reserved", "u2"),
        ("Debug", "u4"),
        ("Round Robin Number", "u2"),
        ("Detector Type", "u1"),
        ("Header Version", "u1"),
        ("Packets caught mask", "8u8")
    ]
)
```

6.12 read_frame Namespace Reference

Variables

- header_dt
- int rows = 512
- int cols = 1024
- int frames = 10
- data = np.zeros((frames,rows,cols), dtype = np.uint16)
- header = np.zeros(frames, dtype = header_dt)
- str file_name = 'jungfrau_single_d0_f{}_0.raw'.format(file_id)
- f
- dtype
- count
- uint16

6.12.1 Variable Documentation

6.12.1.1 cols

```
int read_frame.cols = 1024
```

6.12.1.2 count

```
read_frame.count
```

6.12.1.3 data

```
read_frame.data = np.zeros((frames,rows,cols), dtype = np.uint16)
```

6.12.1.4 dtype

```
read_frame.dtype
```

6.12.1.5 f

```
read_frame.f
```

6.12.1.6 file_name

```
str read_frame.file_name = 'jungfrau_single_d0_f{}_0.raw'.format(file_id)
```

6.12.1.7 frames

```
int read_frame.frames = 10
```

6.12.1.8 header

```
read_frame.header = np.zeros(frames, dtype = header_dt)
```

6.12.1.9 header_dt

```
read_frame.header_dt
```

Initial value:

```
=  np.dtype(
    [
        ("Frame Number", "u8"),
        ("SubFrame Number/ExpLength", "u4"),
        ("Packet Number", "u4"),
        ("Bunch ID", "u8"),
        ("Timestamp", "u8"),
        ("Module Id", "u2"),
        ("Row", "u2"),
        ("Column", "u2"),
        ("Reserved", "u2"),
        ("Debug", "u4"),
        ("Round Robin Number", "u2"),
        ("Detector Type", "u1"),
        ("Header Version", "u1"),
        ("Packets caught mask", "8u8")
    ]
)
```

6.12.1.10 rows

```
int read_frame.rows = 512
```

6.12.1.11 uint16

```
read_frame.uint16
```

6.13 read_multiport Namespace Reference

Variables

- header_dt
- int frames = 1
- int parts = 2
- int frame_cols = 1024
- int frame_rows = 512
- int part_cols = 1024
- int part_rows = 256
- parts_data = np.zeros((frames,parts,part_rows,part_cols), dtype = np.uint16)
- data = np.zeros((frames,frame_rows,frame_cols), dtype = np.uint16)
- header = np.zeros((frames,parts), dtype = header_dt)
- str file_name = f'jungfrau_double_d{part}_f{frame}_{0}.raw'
- f
- dtype
- count
- uint16
- axis

6.13.1 Variable Documentation

6.13.1.1 axis

```
read_multiport.axis
```

6.13.1.2 count

```
read_multiport.count
```

6.13.1.3 data

```
read_multiport.data = np.zeros((frames,frame_rows,frame_cols), dtype = np.uint16)
```

6.13.1.4 dtype

```
read_multiport.dtype
```

6.13.1.5 f

```
read_multiport.f
```

6.13.1.6 file_name

```
str read_multiport.file_name = f'jungfrau_double_d{part}_f{frame}_{0}.raw'
```

6.13.1.7 frame_cols

```
int read_multiport.frame_cols = 1024
```

6.13.1.8 frame_rows

```
int read_multiport.frame_rows = 512
```

6.13.1.9 frames

```
int read_multiport.frames = 1
```

6.13.1.10 header

```
read_multiport.header = np.zeros((frames,parts), dtype = header_dt)
```

6.13.1.11 header_dt

```
read_multiport.header_dt
```

Initial value:

```
=  np.dtype(
    [
        ("Frame Number", "u8"),
        ("SubFrame Number/ExpLength", "u4"),
        ("Packet Number", "u4"),
        ("Bunch ID", "u8"),
        ("Timestamp", "u8"),
        ("Module Id", "u2"),
        ("Row", "u2"),
        ("Column", "u2"),
        ("Reserved", "u2"),
        ("Debug", "u4"),
        ("Round Robin Number", "u2"),
        ("Detector Type", "u1"),
        ("Header Version", "u1"),
        ("Packets caught mask", "8u8")
    ]
)
```

6.13.1.12 part_cols

```
int read_multiport.part_cols = 1024
```

6.13.1.13 part_rows

```
int read_multiport.part_rows = 256
```

6.13.1.14 parts

```
int read_multiport.parts = 2
```

6.13.1.15 parts_data

```
read_multiport.parts_data = np.zeros((frames,parts,part_rows,part_cols), dtype = np.uint16)
```

6.13.1.16 uint16

```
read_multiport.uint16
```

6.14 simdjson Namespace Reference

6.15 write_test_files Namespace Reference

Variables

- arr = np.arange(10, dtype = np.int32)
- arr2 = np.zeros((3,2,5), dtype = np.float64)

6.15.1 Variable Documentation

6.15.1.1 arr

```
write_test_files.arr = np.arange(10, dtype = np.int32)
```

6.15.1.2 arr2

```
write_test_files.arr2 = np.zeros((3,2,5), dtype = np.float64)
```

Chapter 7

Data Structure Documentation

7.1 aare::CircularFifo< ItemType > Class Template Reference

```
#include <CircularFifo.hpp>
```

Public Types

- using value_type = ItemType

Public Member Functions

- CircularFifo ()
- CircularFifo (uint32_t size)
- bool next ()
- ∼CircularFifo ()
- auto numFilledSlots () const noexcept
- auto numFreeSlots () const noexcept
- auto isFull () const noexcept
- ItemType pop_free ()
- bool try_pop_free (ItemType &v)
- ItemType pop_value (std::chrono::nanoseconds wait, std::atomic< bool > &stopped)
- ItemType pop_value ()
- ItemType ∗ frontPtr ()
- template<class... Args>
  void push_value (Args &&...recordArgs)
- template<class... Args>
  bool try_push_value (Args &&...recordArgs)
- template<class... Args>
  void push_free (Args &&...recordArgs)
- template<class... Args>
  bool try_push_free (Args &&...recordArgs)

Private Attributes

- uint32_t fifo_size
- folly::ProducerConsumerQueue< ItemType > free_slots
- folly::ProducerConsumerQueue< ItemType > filled_slots

7.1.1 Member Typedef Documentation

7.1.1.1 value_type

```
template<class ItemType >
using aare::CircularFifo< ItemType >::value_type = ItemType
```

7.1.2 Constructor & Destructor Documentation

7.1.2.1 CircularFifo() [1/2]

```
template<class ItemType >
aare::CircularFifo< ItemType >::CircularFifo ( )  [inline]
```

7.1.2.2 CircularFifo() [2/2]

```
template<class ItemType >
aare::CircularFifo< ItemType >::CircularFifo (
             uint32_t size )  [inline]
```

7.1.2.3 ∼CircularFifo()

```
template<class ItemType >
aare::CircularFifo< ItemType >::∼CircularFifo ( )  [inline]
```

7.1.3 Member Function Documentation

7.1.3.1 frontPtr()

```
template<class ItemType >
ItemType * aare::CircularFifo< ItemType >::frontPtr ( )  [inline]
```

7.1.3.2 isFull()

```
template<class ItemType >
auto aare::CircularFifo< ItemType >::isFull ( ) const  [inline], [noexcept]
```

7.1.3.3 next()

```
template<class ItemType >
bool aare::CircularFifo< ItemType >::next ( )  [inline]
```

7.1.3.4 numFilledSlots()

```
template<class ItemType >
auto aare::CircularFifo< ItemType >::numFilledSlots ( ) const  [inline], [noexcept]
```

7.1.3.5 numFreeSlots()

```
template<class ItemType >
auto aare::CircularFifo< ItemType >::numFreeSlots ( ) const  [inline], [noexcept]
```

7.1.3.6 pop_free()

```
template<class ItemType >
ItemType aare::CircularFifo< ItemType >::pop_free ( )  [inline]
```

7.1.3.7 pop_value() [1/2]

```
template<class ItemType >
ItemType aare::CircularFifo< ItemType >::pop_value ( )  [inline]
```

7.1.3.8 pop_value() [2/2]

```
template<class ItemType >
ItemType aare::CircularFifo< ItemType >::pop_value (
             std::chrono::nanoseconds wait,
             std::atomic< bool > & stopped )  [inline]
```

7.1.3.9 push_free()

```
template<class ItemType >
template<class...  Args>
void aare::CircularFifo< ItemType >::push_free (
             Args &&...  recordArgs )  [inline]
```

7.1.3.10 push_value()

```
template<class ItemType >
template<class...  Args>
void aare::CircularFifo< ItemType >::push_value (
             Args &&...  recordArgs )  [inline]
```

7.1.3.11 try_pop_free()

```
template<class ItemType >
bool aare::CircularFifo< ItemType >::try_pop_free (
             ItemType & v )  [inline]
```

7.1.3.12 try_push_free()

```
template<class ItemType >
template<class... Args>
bool aare::CircularFifo< ItemType >::try_push_free (
             Args &&... recordArgs ) [inline]
```

7.1.3.13 try_push_value()

```
template<class ItemType >
template<class... Args>
bool aare::CircularFifo< ItemType >::try_push_value (
             Args &&... recordArgs ) [inline]
```

7.1.4 Field Documentation

7.1.4.1 fifo_size

```
template<class ItemType >
uint32_t aare::CircularFifo< ItemType >::fifo_size [private]
```

7.1.4.2 filled_slots

```
template<class ItemType >
folly::ProducerConsumerQueue<ItemType> aare::CircularFifo< ItemType >::filled_slots [private]
```

7.1.4.3 free_slots

```
template<class ItemType >
folly::ProducerConsumerQueue<ItemType> aare::CircularFifo< ItemType >::free_slots [private]
```

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

- core/include/aare/core/CircularFifo.hpp

7.2 aare::ClusterFinder< T > Class Template Reference

```
#include <VariableSizeClusterFinder.hpp>
```

Data Structures

- struct Hit

Public Member Functions

- ClusterFinder (image_shape shape, T threshold)
- NDArray< int, 2 > labeled ()
- void set_noiseMap (NDView< T, 2 > noise_map)
- void set_peripheralThresholdFactor (int factor)
- void find_clusters (NDView< T, 2 > img)
- void find_clusters_X (NDView< T, 2 > img)
- void rec_FillHit (int clusterIndex, int i, int j)
- void single_pass (NDView< T, 2 > img)
- void first_pass ()
- void second_pass ()
- void store_clusters ()
- std::vector< Hit > steal_hits ()
- void clear_hits ()
- void print_connections ()
- size_t total_clusters () const

Private Member Functions

- int check_neighbours (int i, int j)
- void add_link (int from, int to)

Private Attributes

- const std::array< ssize_t, 2 > shape_
- NDView< T, 2 > original_
- NDArray< int, 2 > labeled_
- NDArray< int, 2 > peripheral_labeled_
- NDArray< bool, 2 > binary_
- T threshold_
- NDView< T, 2 > noiseMap
- bool use_noise_map = false
- int peripheralThresholdFactor_ = 5
- int current_label
- const std::array< int, 4 > di {{0, -1, -1, -1}}
- const std::array< int, 4 > dj {{-1, -1, 0, 1}}
- const std::array< int, 8 > di_ {{0, 0, -1, 1, -1, 1, -1, 1}}
- const std::array< int, 8 > dj_ {{-1, 1, 0, 0, 1, -1, -1, 1}}
- std::map< int, int > child
- std::unordered_map< int, Hit > h_size
- std::vector< Hit > hits

7.2.1 Constructor & Destructor Documentation

7.2.1.1 ClusterFinder()

```
template<typename T >
aare::ClusterFinder< T >::ClusterFinder (
            image_shape shape,
            T threshold )  [inline]
```

7.2.2 Member Function Documentation

7.2.2.1 add_link()

```
template<typename T >
void aare::ClusterFinder< T >::add_link (
             int from,
             int to )  [inline], [private]
```

7.2.2.2 check_neighbours()

```
template<typename T >
int aare::ClusterFinder< T >::check_neighbours (
             int i,
             int j )  [private]
```

7.2.2.3 clear_hits()

```
template<typename T >
void aare::ClusterFinder< T >::clear_hits ( )  [inline]
```

7.2.2.4 find_clusters()

```
template<typename T >
void aare::ClusterFinder< T >::find_clusters (
             NDView< T, 2 > img )
```

7.2.2.5 find_clusters_X()

```
template<typename T >
void aare::ClusterFinder< T >::find_clusters_X (
             NDView< T, 2 > img )
```

7.2.2.6 first_pass()

```
template<typename T >
void aare::ClusterFinder< T >::first_pass
```

7.2.2.7 labeled()

```
template<typename T >
NDArray< int, 2 > aare::ClusterFinder< T >::labeled ( )  [inline]
```

7.2.2.8 print_connections()

```
template<typename T >
void aare::ClusterFinder< T >::print_connections ( )  [inline]
```

7.2.2.9 rec_FillHit()

```
template<typename T >
void aare::ClusterFinder< T >::rec_FillHit (
            int clusterIndex,
            int i,
            int j )
```

7.2.2.10 second_pass()

```
template<typename T >
void aare::ClusterFinder< T >::second_pass
```

7.2.2.11 set_noiseMap()

```
template<typename T >
void aare::ClusterFinder< T >::set_noiseMap (
            NDView< T, 2 > noise_map )  [inline]
```

7.2.2.12 set_peripheralThresholdFactor()

```
template<typename T >
void aare::ClusterFinder< T >::set_peripheralThresholdFactor (
            int factor )  [inline]
```

7.2.2.13 single_pass()

```
template<typename T >
void aare::ClusterFinder< T >::single_pass (
            NDView< T, 2 > img )
```

7.2.2.14 steal_hits()

```
template<typename T >
std::vector< Hit > aare::ClusterFinder< T >::steal_hits ( )  [inline]
```

7.2.2.15 store_clusters()

```
template<typename T >
void aare::ClusterFinder< T >::store_clusters
```

7.2.2.16 total_clusters()

```
template<typename T >
size_t aare::ClusterFinder< T >::total_clusters ( ) const  [inline]
```

7.2.3 Field Documentation

7.2.3.1 binary_

```
template<typename T >
NDArray<bool, 2> aare::ClusterFinder< T >::binary_  [private]
```

7.2.3.2 child

```
template<typename T >
std::map<int, int> aare::ClusterFinder< T >::child  [private]
```

7.2.3.3 current_label

```
template<typename T >
int aare::ClusterFinder< T >::current_label  [private]
```

7.2.3.4 di

```
template<typename T >
const std::array<int, 4> aare::ClusterFinder< T >::di {{0, -1, -1, -1}}  [private]
```

7.2.3.5 di_

```
template<typename T >
const std::array<int, 8> aare::ClusterFinder< T >::di_ {{0, 0, -1, 1, -1, 1, -1, 1}}  [private]
```

7.2.3.6 dj

```
template<typename T >
const std::array<int, 4> aare::ClusterFinder< T >::dj {{-1, -1, 0, 1}}  [private]
```

7.2.3.7 dj_

```
template<typename T >
const std::array<int, 8> aare::ClusterFinder< T >::dj_ {{-1, 1, 0, 0, 1, -1, -1, 1}}  [private]
```

7.2.3.8 h_size

```
template<typename T >
std::unordered_map<int, Hit> aare::ClusterFinder< T >::h_size  [private]
```

7.2.3.9 hits

```
template<typename T >
std::vector<Hit> aare::ClusterFinder< T >::hits  [private]
```

7.2.3.10 labeled_

```
template<typename T >
NDArray<int, 2> aare::ClusterFinder< T >::labeled_  [private]
```

7.2.3.11 noiseMap

```
template<typename T >
NDView<T, 2> aare::ClusterFinder< T >::noiseMap  [private]
```

7.2.3.12 original_

```
template<typename T >
NDView<T, 2> aare::ClusterFinder< T >::original_  [private]
```

7.2.3.13 peripheral_labeled_

```
template<typename T >
NDArray<int, 2> aare::ClusterFinder< T >::peripheral_labeled_  [private]
```

7.2.3.14 peripheralThresholdFactor_

```
template<typename T >
int aare::ClusterFinder< T >::peripheralThresholdFactor_ = 5  [private]
```

7.2.3.15 shape_

```
template<typename T >
const std::array<ssize_t, 2> aare::ClusterFinder< T >::shape_  [private]
```

7.2.3.16 threshold_

```
template<typename T >
T aare::ClusterFinder< T >::threshold_  [private]
```

7.2.3.17 use_noise_map

```
template<typename T >
bool aare::ClusterFinder< T >::use_noise_map = false  [private]
```

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

- core/include/aare/core/VariableSizeClusterFinder.hpp

7.3 aare::DType Class Reference

```
#include <DType.hpp>
```

Public Types

- enum TypeIndex {
  INT8 , UINT8 , INT16 , UINT16 ,
  INT32 , UINT32 , INT64 , UINT64 ,
  FLOAT , DOUBLE , ERROR }

Public Member Functions

- uint8_t bitdepth () const
- DType (const std::type_info &t)
- DType (std::string_view sv)
- DType (DType::TypeIndex ti)
- bool operator== (const DType &other) const noexcept
- bool operator!= (const DType &other) const noexcept
- bool operator== (const std::type_info &t) const
- bool operator!= (const std::type_info &t) const
- std::string str () const

Private Attributes

- TypeIndex m_type {TypeIndex::ERROR}

7.3.1 Member Enumeration Documentation

7.3.1.1 TypeIndex

```
enum aare::DType::TypeIndex
```

Enumerator

INT8	
UINT8	
INT16	
UINT16	
INT32	
UINT32	
INT64	
UINT64	

7.3.2 Constructor & Destructor Documentation

7.3.2.1 DType() [1/3]

```
aare::DType::DType (
            const std::type_info & t )  [explicit]
```

7.3.2.2 DType() [2/3]

```
aare::DType::DType (
            std::string_view sv )  [explicit]
```

7.3.2.3 DType() [3/3]

```
aare::DType::DType (
            DType::TypeIndex ti )
```

7.3.3 Member Function Documentation

7.3.3.1 bitdepth()

```
uint8_t aare::DType::bitdepth ( ) const
```

7.3.3.2 operator"!=() [1/2]

```
bool aare::DType::operator!= (
            const DType & other ) const  [noexcept]
```

7.3.3.3 operator"!=() [2/2]

```
bool aare::DType::operator!= (
            const std::type_info & t ) const
```

7.3.3.4 operator==() [1/2]

```
bool aare::DType::operator== (
            const DType & other ) const  [noexcept]
```

7.3.3.5 operator==() [2/2]

```
bool aare::DType::operator== (
            const std::type_info & t ) const
```

7.3.3.6 str()

```
std::string aare::DType::str ( ) const
```

7.3.4 Field Documentation

7.3.4.1 m_type

```
TypeIndex aare::DType::m_type {TypeIndex::ERROR}  [private]
```

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

- core/include/aare/core/DType.hpp
- core/src/DType.cpp

7.4 aare::File Class Reference

RAII File class for reading and writing image files in various formats wrapper on a FileInterface to abstract the underlying file format.

```
#include <File.hpp>
```

Public Member Functions

- File (std::filesystem::path fname, std::string mode, FileConfig cfg={})

  Construct a new File object.
- void write (Frame &frame)
- Frame read ()
- Frame iread (size_t frame_number)
- std::vector< Frame > read (size_t n_frames)
- void read_into (std::byte ∗image_buf)
- void read_into (std::byte ∗image_buf, size_t n_frames)
- size_t frame_number (size_t frame_index)
- size_t bytes_per_frame ()
- size_t pixels ()
- void seek (size_t frame_number)
- size_t tell () const
- size_t total_frames () const
- ssize_t rows () const
- ssize_t cols () const
- ssize_t bitdepth () const
- File (File &&other)

  Move constructor.
- ∼File ()

  destructor: will only delete the FileInterface object

Private Attributes

- FileInterface ∗ file_impl

7.4.1 Detailed Description

RAII File class for reading and writing image files in various formats wrapper on a FileInterface to abstract the underlying file format.

Note

documentation for each function is in the FileInterface class

7.4.2 Constructor & Destructor Documentation

7.4.2.1 File() [1/2]

```
aare::File::File (
            std::filesystem::path fname,
            std::string mode,
            FileConfig cfg = {} )
```

Construct a new File object.

Parameters

fname	path to the file
mode	file mode (r, w, a)
cfg	file configuration

Exceptions

std::runtime_error	if the file cannot be opened
std::invalid_argument	if the file mode is not supported

7.4.2.2 File() [2/2]

```
aare::File::File (
            File && other )
```

Move constructor.

Parameters

other	File object to move from

7.4.2.3 ∼File()

```
aare::File::∼File ( )
```

destructor: will only delete the FileInterface object

7.4.3 Member Function Documentation

7.4.3.1 bitdepth()

```
ssize_t aare::File::bitdepth ( ) const
```

7.4.3.2 bytes_per_frame()

```
size_t aare::File::bytes_per_frame ( )
```

7.4.3.3 cols()

```
ssize_t aare::File::cols ( ) const
```

7.4.3.4 frame_number()

```
size_t aare::File::frame_number (
             size_t frame_index )
```

7.4.3.5 iread()

```
Frame aare::File::iread (
             size_t frame_number )
```

7.4.3.6 pixels()

```
size_t aare::File::pixels ( )
```

7.4.3.7 read() [1/2]

```
Frame aare::File::read ( )
```

7.4.3.8 read() [2/2]

```
std::vector< Frame > aare::File::read (
             size_t n_frames )
```

7.4.3.9 read_into() [1/2]

```
void aare::File::read_into (
             std::byte * image_buf )
```

7.4.3.10 read_into() [2/2]

```
void aare::File::read_into (
             std::byte * image_buf,
             size_t n_frames )
```

7.4.3.11 rows()

```
ssize_t aare::File::rows ( ) const
```

7.4.3.12 seek()

```
void aare::File::seek (
             size_t frame_number )
```

7.4.3.13 tell()

```
size_t aare::File::tell ( ) const
```

7.4.3.14 total_frames()

```
size_t aare::File::total_frames ( ) const
```

7.4.3.15 write()

```
void aare::File::write (
             Frame & frame )
```

7.4.4 Field Documentation

7.4.4.1 file_impl

```
FileInterface* aare::File::file_impl  [private]
```

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

- file_io/include/aare/file_io/File.hpp
- file_io/src/File.cpp

7.5 aare.File.File Class Reference

Public Member Functions

- __init__ (self, path)
- Any __getattribute__ (self, str __name)

Data Fields

- path

Protected Attributes

- _file

7.5.1 Detailed Description

```
File class. uses proxy pattern to wrap around the pybinding class
abstracts the python binding class that is requires type and detector information
(e.g. _FileHandler_Jungfrau_16)
```

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__()

```
aare.File.File.__init__ (
              self,
              path )
```

```
opens the master file and checks the dynamic range and detector
```

7.5.3 Member Function Documentation

7.5.3.1 __getattribute__()

```
Any aare.File.File.__getattribute__ (
              self,
             str __name )
```

```
Proxy pattern to call the methods of the _file
```

7.5.4 Field Documentation

7.5.4.1 _file

```
aare.File.File._file  [protected]
```

7.5.4.2 path

```
aare.File.File.path
```

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

- python/aare/File.py

7.6 aare::FileConfig Struct Reference

FileConfig structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.

```
#include <FileInterface.hpp>
```

Public Member Functions

- bool operator== (const FileConfig &other) const
- bool operator!= (const FileConfig &other) const

Data Fields

- aare::DType dtype = aare::DType(typeid(uint16_t))
- uint64_t rows
- uint64_t cols
- xy geometry {1, 1}

7.6.1 Detailed Description

FileConfig structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.

7.6.2 Member Function Documentation

7.6.2.1 operator"!=()

```
bool aare::FileConfig::operator!= (
             const FileConfig & other ) const  [inline]
```

7.6.2.2 operator==()

```
bool aare::FileConfig::operator== (
             const FileConfig & other ) const  [inline]
```

7.6.3 Field Documentation

7.6.3.1 cols

```
uint64_t aare::FileConfig::cols
```

7.6.3.2 dtype

```
aare::DType aare::FileConfig::dtype = aare::DType(typeid(uint16_t))
```

7.6.3.3 geometry

```
xy aare::FileConfig::geometry {1, 1}
```

7.6.3.4 rows

```
uint64_t aare::FileConfig::rows
```

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

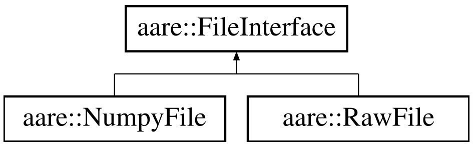
- file_io/include/aare/file_io/FileInterface.hpp

7.7 aare::FileInterface Class Reference

FileInterface class to define the interface for file operations.

```
#include <FileInterface.hpp>
```

Inheritance diagram for aare::FileInterface:



Public Member Functions

- virtual void write (Frame &frame)=0
    write a frame to the file
- virtual Frame read ()=0
    write a vector of frames to the file
- virtual std::vector< Frame > read (size_t n_frames)=0
    read n_frames from the file at the current position
- virtual void read_into (std::byte ∗image_buf)=0
    read one frame from the file at the current position and store it in the provided buffer
- virtual void read_into (std::byte ∗image_buf, size_t n_frames)=0
    read n_frames from the file at the current position and store them in the provided buffer
- virtual size_t frame_number (size_t frame_index)=0
    get the frame number at the given frame index
- virtual size_t bytes_per_frame ()=0
    get the size of one frame in bytes
- virtual size_t pixels ()=0
    get the number of pixels in one frame
- virtual void seek (size_t frame_number)=0
    seek to the given frame number
- virtual size_t tell ()=0
    get the current position of the file pointer

- virtual size_t total_frames () const =0

  get the total number of frames in the file
- virtual ssize_t rows () const =0

  get the number of rows in the file
- virtual ssize_t cols () const =0

  get the number of columns in the file
- virtual ssize_t bitdepth () const =0

  get the bitdepth of the file
- Frame iread (size_t frame_number)

  read one frame from the file at the given frame number
- std::vector< Frame > iread (size_t frame_number, size_t n_frames)

  read n_frames from the file starting at the given frame number
- virtual ~FileInterface ()

Data Fields

- std::string m_mode
- std::filesystem::path m_fname
- std::filesystem::path m_base_path
- std::string m_base_name
- std::string m_ext
- int m_findex
- size_t m_total_frames {}
- size_t max_frames_per_file {}
- std::string version
- DetectorType m_type
- ssize_t m_rows {}
- ssize_t m_cols {}
- ssize_t m_bitdepth {}
- size_t current_frame {}

7.7.1 Detailed Description

FileInterface class to define the interface for file operations.

Note

parent class for NumpyFile and RawFile

all functions are pure virtual and must be implemented by the derived classes

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ~FileInterface()

```
virtual aare::FileInterface::~FileInterface ( )  [inline], [virtual]
```

7.7.3 Member Function Documentation

7.7.3.1 bitdepth()

```
virtual ssize_t aare::FileInterface::bitdepth ( ) const  [pure virtual]
```

get the bitdepth of the file

Returns

bitdepth of the file

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.2 bytes_per_frame()

```
virtual size_t aare::FileInterface::bytes_per_frame ( )  [pure virtual]
```

get the size of one frame in bytes

Returns

size of one frame

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.3 cols()

```
virtual ssize_t aare::FileInterface::cols ( ) const  [pure virtual]
```

get the number of columns in the file

Returns

number of columns in the file

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.4 frame_number()

```
virtual size_t aare::FileInterface::frame_number (
             size_t frame_index )  [pure virtual]
```

get the frame number at the given frame index

Parameters

frame_index	index of the frame

Returns

frame number

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.5 iread() [1/2]

```
Frame aare::FileInterface::iread (
            size_t frame_number )  [inline]
```

read one frame from the file at the given frame number

Parameters

frame_number	frame number to read

Returns

frame

7.7.3.6 iread() [2/2]

```
std::vector< Frame > aare::FileInterface::iread (
            size_t frame_number,
            size_t n_frames )  [inline]
```

read n_frames from the file starting at the given frame number

Parameters

frame_number	frame number to start reading from
n_frames	number of frames to read

Returns

vector of frames

7.7.3.7 pixels()

```
virtual size_t aare::FileInterface::pixels ( )  [pure virtual]
```

get the number of pixels in one frame

Returns

number of pixels in one frame

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.8 read() [1/2]

```
virtual Frame aare::FileInterface::read ( )  [pure virtual]
```

write a vector of frames to the file

Parameters

frames	vector of frames to write

Returns

void

read one frame from the file at the current position

Returns

Frame

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.9 read() [2/2]

```
virtual std::vector< Frame > aare::FileInterface::read (
             size_t n_frames )  [pure virtual]
```

read n_frames from the file at the current position

Parameters

n_frames	number of frames to read

Returns

vector of frames

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.10 read_into() [1/2]

```
virtual void aare::FileInterface::read_into (
             std::byte * image_buf )  [pure virtual]
```

read one frame from the file at the current position and store it in the provided buffer

Parameters

image_buf	buffer to store the frame

Returns

void

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.11 read_into() [2/2]

```
virtual void aare::FileInterface::read_into (
            std::byte * image_buf,
            size_t n_frames ) [pure virtual]
```

read n_frames from the file at the current position and store them in the provided buffer

Parameters

image_buf	buffer to store the frames
n_frames	number of frames to read

Returns

void

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.12 rows()

```
virtual ssize_t aare::FileInterface::rows ( ) const [pure virtual]
```

get the number of rows in the file

Returns

number of rows in the file

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.13 seek()

```
virtual void aare::FileInterface::seek (
            size_t frame_number ) [pure virtual]
```

seek to the given frame number

Parameters

frame_number	frame number to seek to

Returns

void

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.14 tell()

```
virtual size_t aare::FileInterface::tell ( )  [pure virtual]
```

get the current position of the file pointer

Returns

current position of the file pointer

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.15 total_frames()

```
virtual size_t aare::FileInterface::total_frames ( ) const  [pure virtual]
```

get the total number of frames in the file

Returns

total number of frames in the file

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.3.16 write()

```
virtual void aare::FileInterface::write (
            Frame & frame )  [pure virtual]
```

write a frame to the file

Parameters

frame	frame to write

Returns

void

Exceptions

std::runtime_error	if the function is not implemented

Implemented in aare::NumpyFile, and aare::RawFile.

7.7.4 Field Documentation

7.7.4.1 current_frame

```
size_t aare::FileInterface::current_frame {}
```

7.7.4.2 m_base_name

```
std::string aare::FileInterface::m_base_name
```

7.7.4.3 m_base_path

```
std::filesystem::path aare::FileInterface::m_base_path
```

7.7.4.4 m_bitdepth

```
ssize_t aare::FileInterface::m_bitdepth {}
```

7.7.4.5 m_cols

```
ssize_t aare::FileInterface::m_cols {}
```

7.7.4.6 m_ext

```
std::string aare::FileInterface::m_ext
```

7.7.4.7 m_findex

```
int aare::FileInterface::m_findex
```

7.7.4.8 m_fname

```
std::filesystem::path aare::FileInterface::m_fname
```

7.7.4.9 m_mode

```
std::string aare::FileInterface::m_mode
```

7.7.4.10 m_rows

```
ssize_t aare::FileInterface::m_rows {}
```

7.7.4.11 m_total_frames

```
size_t aare::FileInterface::m_total_frames {}
```

7.7.4.12 m_type

```
DetectorType aare::FileInterface::m_type
```

7.7.4.13 max_frames_per_file

```
size_t aare::FileInterface::max_frames_per_file {}
```

7.7.4.14 version

```
std::string aare::FileInterface::version
```

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

- file_io/include/aare/file_io/FileInterface.hpp

7.8 aare::Frame Class Reference

```
#include <Frame.hpp>
```

Public Member Functions

- Frame (ssize_t rows, ssize_t cols, ssize_t m_bitdepth)
- Frame (std::byte ∗fp, ssize_t rows, ssize_t cols, ssize_t m_bitdepth)
- std::byte ∗ get (int row, int col)
- template<typename T >
  void set (int row, int col, T data)
- ssize_t rows () const
- ssize_t cols () const
- ssize_t bitdepth () const
- ssize_t size () const
- std::byte ∗ data () const
- Frame & operator= (Frame &other)
- Frame (Frame &&other)
- Frame (const Frame &other)
- template<typename T >
  NDView< T > view ()
- template<typename T >
  NDArray< T > image ()
- ∼Frame ()

Private Attributes

- ssize_t m_rows
- ssize_t m_cols
- ssize_t m_bitdepth
- std::byte ∗ m_data

7.8.1 Constructor & Destructor Documentation

7.8.1.1 Frame() [1/4]

```
aare::Frame::Frame (
            ssize_t rows,
            ssize_t cols,
            ssize_t m_bitdepth )
```

7.8.1.2 Frame() [2/4]

```
aare::Frame::Frame (
            std::byte ∗ fp,
            ssize_t rows,
            ssize_t cols,
            ssize_t m_bitdepth )
```

7.8.1.3 Frame() [3/4]

```
aare::Frame::Frame (
            Frame && other )  [inline]
```

7.8.1.4 Frame() [4/4]

```
aare::Frame::Frame (
            const Frame & other )  [inline]
```

7.8.1.5 ∼Frame()

```
aare::Frame::∼Frame ( )  [inline]
```

7.8.2 Member Function Documentation

7.8.2.1 bitdepth()

```
ssize_t aare::Frame::bitdepth ( ) const  [inline]
```

7.8.2.2 cols()

```
ssize_t aare::Frame::cols ( ) const  [inline]
```

7.8.2.3 data()

```
std::byte * aare::Frame::data ( ) const  [inline]
```

7.8.2.4 get()

```
std::byte * aare::Frame::get (
             int row,
             int col )
```

7.8.2.5 image()

```
template<typename T >
NDArray< T > aare::Frame::image ( )  [inline]
```

7.8.2.6 operator=()

```
Frame & aare::Frame::operator= (
             Frame & other )  [inline]
```

7.8.2.7 rows()

```
ssize_t aare::Frame::rows ( ) const  [inline]
```

7.8.2.8 set()

```
template<typename T >
void aare::Frame::set (
             int row,
             int col,
             T data )  [inline]
```

7.8.2.9 size()

```
ssize_t aare::Frame::size ( ) const  [inline]
```

7.8.2.10 view()

```
template<typename T >
NDView< T > aare::Frame::view ( )  [inline]
```

7.8.3 Field Documentation

7.8.3.1 m_bitdepth

```
ssize_t aare::Frame::m_bitdepth  [private]
```

7.8.3.2 m_cols

```
ssize_t aare::Frame::m_cols  [private]
```

7.8.3.3 m_data

```
std::byte* aare::Frame::m_data  [private]
```

7.8.3.4 m_rows

```
ssize_t aare::Frame::m_rows  [private]
```

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

- core/include/aare/core/Frame.hpp
- core/src/Frame.cpp

7.9 aare.Frame.Frame Class Reference

Public Member Functions

- __init__ (self, frameImpl)
- Any __getattribute__ (self, str __name)

Protected Attributes

- _frameImpl

7.9.1 Detailed Description

```
Frame class. uses proxy pattern to wrap around the pybinding class
the intention behind it is to only use one class for frames in python (not Frame_8, Frame_16, etc)
```

7.9.2 Constructor & Destructor Documentation

7.9.2.1 __init__()

```
aare.Frame.Frame.__init__ (
              self,
              frameImpl )
```

7.9.3 Member Function Documentation

7.9.3.1 __getattribute__()

```
Any aare.Frame.Frame.__getattribute__ (
              self,
             str __name )
```

```
Proxy pattern to call the methods of the frameImpl
```

7.9.4 Field Documentation

7.9.4.1 _frameImpl

```
aare.Frame.Frame._frameImpl  [protected]
```

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

- python/aare/Frame.py

7.10 aare::ClusterFinder< T >::Hit Struct Reference

```
#include <VariableSizeClusterFinder.hpp>
```

Data Fields

- int16_t size {}
- int16_t row {}
- int16_t col {}
- uint16_t reserved {}
- T energy {}
- T max {}
- int16_t rows [MAX_CLUSTER_SIZE] = {0}
- int16_t cols [MAX_CLUSTER_SIZE] = {0}
- double enes [MAX_CLUSTER_SIZE] = {0}

7.10.1 Field Documentation

7.10.1.1 col

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::col {}
```

7.10.1.2 cols

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::cols[MAX_CLUSTER_SIZE] = {0}
```

7.10.1.3 energy

```
template<typename T >
T aare::ClusterFinder< T >::Hit::energy {}
```

7.10.1.4 enes

```
template<typename T >
double aare::ClusterFinder< T >::Hit::enes[MAX_CLUSTER_SIZE] = {0}
```

7.10.1.5 max

```
template<typename T >
T aare::ClusterFinder< T >::Hit::max {}
```

7.10.1.6 reserved

```
template<typename T >
uint16_t aare::ClusterFinder< T >::Hit::reserved {}
```

7.10.1.7 row

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::row {}
```

7.10.1.8 rows

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::rows[MAX_CLUSTER_SIZE] = {0}
```

7.10.1.9 size

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::size {}
```

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

- core/include/aare/core/VariableSizeClusterFinder.hpp

7.11 aare::logger::Logger Class Reference

```
#include <logger.hpp>
```

Public Member Functions

- void set_output_file (std::string filename)
- void set_streams (std::streambuf ∗out, std::streambuf ∗err)
- void set_streams (std::streambuf ∗out)
- void set_verbosity (LOGGING_LEVEL level)
- Logger ()
- ∼Logger ()
- template<LOGGING_LEVEL level, typename... Strings>
  void log (const Strings... s)
- template<typename... Strings>
  void debug (const Strings... s)
- template<typename... Strings>
  void info (const Strings... s)
- template<typename... Strings>
  void warn (const Strings... s)
- template<typename... Strings>
  void error (const Strings... s)

Private Member Functions

- template<LOGGING_LEVEL level>
  void log_ ()
- template<LOGGING_LEVEL level, typename First , typename... Strings>
  void log_ (First arg, const Strings... s)

Private Attributes

- std::streambuf ∗ standard_buf = std::cout.rdbuf()
- std::streambuf ∗ error_buf = std::cerr.rdbuf()
- std::ostream ∗ standard_output
- std::ostream ∗ error_output
- LOGGING_LEVEL VERBOSITY_LEVEL = LOGGING_LEVEL::INFO
- std::ofstream out_file

7.11.1 Constructor & Destructor Documentation

7.11.1.1 Logger()

```
aare::logger::Logger::Logger ( )   [inline]
```

7.11.1.2 ∼Logger()

```
aare::logger::Logger::∼Logger ( )   [inline]
```

7.11.2 Member Function Documentation

7.11.2.1 debug()

```
template<typename...  Strings>
void aare::logger::Logger::debug (
             const Strings...  s )  [inline]
```

7.11.2.2 error()

```
template<typename...  Strings>
void aare::logger::Logger::error (
             const Strings...  s )  [inline]
```

7.11.2.3 info()

```
template<typename...  Strings>
void aare::logger::Logger::info (
             const Strings...  s )  [inline]
```

7.11.2.4 log()

```
template<LOGGING_LEVEL level, typename...  Strings>
void aare::logger::Logger::log (
             const Strings...  s )  [inline]
```

7.11.2.5 log_() [1/2]

```
template<LOGGING_LEVEL level>
void aare::logger::Logger::log_ ( )  [inline], [private]
```

7.11.2.6 log_() [2/2]

```
template<LOGGING_LEVEL level, typename First , typename...  Strings>
void aare::logger::Logger::log_ (
             First arg,
             const Strings...  s )  [inline], [private]
```

7.11.2.7 set_output_file()

```
void aare::logger::Logger::set_output_file (
             std::string filename )  [inline]
```

7.11.2.8 set_streams() [1/2]

```
void aare::logger::Logger::set_streams (
            std::streambuf * out )  [inline]
```

7.11.2.9 set_streams() [2/2]

```
void aare::logger::Logger::set_streams (
            std::streambuf * out,
            std::streambuf * err )  [inline]
```

7.11.2.10 set_verbosity()

```
void aare::logger::Logger::set_verbosity (
            LOGGING_LEVEL level )  [inline]
```

7.11.2.11 warn()

```
template<typename... Strings>
void aare::logger::Logger::warn (
            const Strings... s )  [inline]
```

7.11.3 Field Documentation

7.11.3.1 error_buf

```
std::streambuf* aare::logger::Logger::error_buf = std::cerr.rdbuf()  [private]
```

7.11.3.2 error_output

```
std::ostream* aare::logger::Logger::error_output  [private]
```

7.11.3.3 out_file

```
std::ofstream aare::logger::Logger::out_file  [private]
```

7.11.3.4 standard_buf

```
std::streambuf* aare::logger::Logger::standard_buf = std::cout.rdbuf()  [private]
```

7.11.3.5 standard_output

```
std::ostream* aare::logger::Logger::standard_output  [private]
```

7.11.3.6 VERBOSITY_LEVEL

```
LOGGING_LEVEL aare::logger::Logger::VERBOSITY_LEVEL = LOGGING_LEVEL::INFO  [private]
```

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

- utils/include/aare/utils/logger.hpp

7.12 MoveOnlyInt Struct Reference

Public Member Functions

- MoveOnlyInt ()=default
- MoveOnlyInt (int i)
- MoveOnlyInt (const MoveOnlyInt &)=delete
- MoveOnlyInt & operator= (const MoveOnlyInt &)=delete
- MoveOnlyInt (MoveOnlyInt &&other)
- MoveOnlyInt & operator= (MoveOnlyInt &&other)
- bool operator== (int other) const

Data Fields

- int value {}

7.12.1 Constructor & Destructor Documentation

7.12.1.1 MoveOnlyInt() [1/4]

```
MoveOnlyInt::MoveOnlyInt ( )  [default]
```

7.12.1.2 MoveOnlyInt() [2/4]

```
MoveOnlyInt::MoveOnlyInt (
              int i )  [inline]
```

7.12.1.3 MoveOnlyInt() [3/4]

```
MoveOnlyInt::MoveOnlyInt (
              const MoveOnlyInt &  )  [delete]
```

7.12.1.4 MoveOnlyInt() [4/4]

```
MoveOnlyInt::MoveOnlyInt (
              MoveOnlyInt && other )  [inline]
```

7.12.2 Member Function Documentation

7.12.2.1 operator=() [1/2]

```
MoveOnlyInt & MoveOnlyInt::operator= (
             const MoveOnlyInt &  )  [delete]
```

7.12.2.2 operator=() [2/2]

```
MoveOnlyInt & MoveOnlyInt::operator= (
             MoveOnlyInt && other )  [inline]
```

7.12.2.3 operator==()

```
bool MoveOnlyInt::operator== (
             int other ) const  [inline]
```

7.12.3 Field Documentation

7.12.3.1 value

```
int MoveOnlyInt::value {}
```

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

- core/test/CircularFifo.test.cpp

7.13 aare::NDArray< T, Ndim > Class Template Reference

```
#include <NDArray.hpp>
```

Public Types

- using value_type = T

Public Member Functions

- NDArray ()
- NDArray (std::array< ssize_t, Ndim > shape)
- NDArray (std::array< ssize_t, Ndim > shape, T value)
- NDArray (NDView< T, Ndim > span)
- NDArray (NDArray &&other)
- NDArray (const NDArray &other)
- ∼NDArray ()
- auto begin ()
- auto end ()
- NDArray & operator= (NDArray &&other)
- NDArray & operator= (const NDArray &other)
- NDArray operator+ (const NDArray &other)
- NDArray & operator+= (const NDArray &other)
- NDArray operator- (const NDArray &other)
- NDArray & operator-= (const NDArray &other)
- NDArray operator∗ (const NDArray &other)
- NDArray & operator∗= (const NDArray &other)
- NDArray operator/ (const NDArray &other)
- template<typename V >
  NDArray & operator/= (const NDArray< V, Ndim > &other)
- NDArray< bool, Ndim > operator> (const NDArray &other)
- bool operator== (const NDArray &other) const
- bool operator!= (const NDArray &other) const
- NDArray & operator= (const T &)
- NDArray & operator+= (const T &)
- NDArray operator+ (const T &)
- NDArray & operator-= (const T &)
- NDArray operator- (const T &)
- NDArray & operator∗= (const T &)
- NDArray operator∗ (const T &)
- NDArray & operator/= (const T &)
- NDArray operator/ (const T &)
- NDArray & operator&= (const T &)
- void sqrt ()
- NDArray & operator++ ()
- template<typename... Ix>
  std::enable_if< sizeof...(Ix)==Ndim, T & >::type operator() (Ix... index)
- template<typename... Ix>
  std::enable_if< sizeof...(Ix)==Ndim, T & >::type operator() (Ix... index) const
- template<typename... Ix>
  std::enable_if< sizeof...(Ix)==Ndim, T >::type value (Ix... index)
- T & operator() (int i)
- const T & operator() (int i) const
- T ∗ data ()
- std::byte ∗ buffer ()
- ssize_t size () const
- size_t total_bytes () const
- std::array< ssize_t, Ndim > shape () const noexcept
- ssize_t shape (ssize_t i) const noexcept
- std::array< ssize_t, Ndim > strides () const noexcept
- std::array< ssize_t, Ndim > byte_strides () const noexcept
- NDView< T, Ndim > span () const
- void Print ()
- void Print_all ()
- void Print_some ()
- void reset ()

Private Attributes

- std::array< ssize_t, Ndim > shape_
- std::array< ssize_t, Ndim > strides_
- ssize_t size_
- T * data_

7.13.1 Member Typedef Documentation

7.13.1.1 value_type

```
template<typename T , ssize_t Ndim = 2>
using aare::NDArray< T, Ndim >::value_type = T
```

7.13.2 Constructor & Destructor Documentation

7.13.2.1 NDArray() [1/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray ( )  [inline]
```

7.13.2.2 NDArray() [2/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
             std::array< ssize_t, Ndim > shape )  [inline], [explicit]
```

7.13.2.3 NDArray() [3/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
             std::array< ssize_t, Ndim > shape,
             T value )  [inline]
```

7.13.2.4 NDArray() [4/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
             NDView< T, Ndim > span )  [inline]
```

7.13.2.5 NDArray() [5/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
             NDArray< T, Ndim > && other )  [inline]
```

7.13.2.6 NDArray() [6/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
             const NDArray< T, Ndim > & other )  [inline]
```

7.13.2.7 ∼NDArray()

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::∼NDArray ( )  [inline]
```

7.13.3 Member Function Documentation

7.13.3.1 begin()

```
template<typename T , ssize_t Ndim = 2>
auto aare::NDArray< T, Ndim >::begin ( )  [inline]
```

7.13.3.2 buffer()

```
template<typename T , ssize_t Ndim = 2>
std::byte * aare::NDArray< T, Ndim >::buffer ( )  [inline]
```

7.13.3.3 byte_strides()

```
template<typename T , ssize_t Ndim = 2>
std::array< ssize_t, Ndim > aare::NDArray< T, Ndim >::byte_strides ( ) const  [inline], [noexcept]
```

7.13.3.4 data()

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDArray< T, Ndim >::data ( )  [inline]
```

7.13.3.5 end()

```
template<typename T , ssize_t Ndim = 2>
auto aare::NDArray< T, Ndim >::end ( )  [inline]
```

7.13.3.6 operator"!=()

```
template<typename T , ssize_t Ndim>
bool aare::NDArray< T, Ndim >::operator!= (
             const NDArray< T, Ndim > & other ) const
```

7.13.3.7 operator&=()

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator&= (
            const T & mask )
```

7.13.3.8 operator()() [1/4]

```
template<typename T , ssize_t Ndim = 2>
T & aare::NDArray< T, Ndim >::operator() (
            int i )  [inline]
```

7.13.3.9 operator()() [2/4]

```
template<typename T , ssize_t Ndim = 2>
const T & aare::NDArray< T, Ndim >::operator() (
            int i ) const  [inline]
```

7.13.3.10 operator()() [3/4]

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDArray< T, Ndim >::operator() (
            Ix...  index )  [inline]
```

7.13.3.11 operator()() [4/4]

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDArray< T, Ndim >::operator() (
            Ix...  index ) const  [inline]
```

7.13.3.12 operator*() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator* (
            const NDArray< T, Ndim > & other )
```

7.13.3.13 operator*() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator* (
            const T & value )
```

7.13.3.14 operator*=() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator*= (
            const NDArray< T, Ndim > & other )
```

7.13.3.15 operator*=() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator*= (
            const T & value )
```

7.13.3.16 operator+() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator+ (
            const NDArray< T, Ndim > & other )
```

7.13.3.17 operator+() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator+ (
            const T & value )
```

7.13.3.18 operator++()

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator++
```

7.13.3.19 operator+=() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator+= (
            const NDArray< T, Ndim > & other )
```

7.13.3.20 operator+=() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator+= (
            const T & value )
```

7.13.3.21 operator-() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator- (
            const NDArray< T, Ndim > & other )
```

7.13.3.22 operator-() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator- (
             const T & value )
```

7.13.3.23 operator-=() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator-= (
             const NDArray< T, Ndim > & other )
```

7.13.3.24 operator-=() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator-= (
             const T & value )
```

7.13.3.25 operator/() [1/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator/ (
             const NDArray< T, Ndim > & other )
```

7.13.3.26 operator/() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator/ (
             const T & value )
```

7.13.3.27 operator/=() [1/2]

```
template<typename T , ssize_t Ndim = 2>
template<typename V >
NDArray & aare::NDArray< T, Ndim >::operator/= (
             const NDArray< V, Ndim > & other )  [inline]
```

7.13.3.28 operator/=() [2/2]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator/= (
             const T & value )
```

7.13.3.29 operator=() [1/3]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator= (
             const NDArray< T, Ndim > & other )
```

7.13.3.30 operator=() [2/3]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator= (
             const T & value )
```

7.13.3.31 operator=() [3/3]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator= (
             NDArray< T, Ndim > && other )
```

7.13.3.32 operator==()

```
template<typename T , ssize_t Ndim>
bool aare::NDArray< T, Ndim >::operator== (
             const NDArray< T, Ndim > & other ) const
```

7.13.3.33 operator>()

```
template<typename T , ssize_t Ndim>
NDArray< bool, Ndim > aare::NDArray< T, Ndim >::operator> (
             const NDArray< T, Ndim > & other )
```

7.13.3.34 Print()

```
template<typename T , ssize_t Ndim>
void aare::NDArray< T, Ndim >::Print
```

7.13.3.35 Print_all()

```
template<typename T , ssize_t Ndim>
void aare::NDArray< T, Ndim >::Print_all
```

7.13.3.36 Print_some()

```
template<typename T , ssize_t Ndim>
void aare::NDArray< T, Ndim >::Print_some
```

7.13.3.37 reset()

```
template<typename T , ssize_t Ndim = 2>
void aare::NDArray< T, Ndim >::reset ( )  [inline]
```

7.13.3.38 shape() [1/2]

```
template<typename T , ssize_t Ndim = 2>
std::array< ssize_t, Ndim > aare::NDArray< T, Ndim >::shape ( ) const  [inline], [noexcept]
```

7.13.3.39 shape() [2/2]

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDArray< T, Ndim >::shape (
             ssize_t i ) const  [inline], [noexcept]
```

7.13.3.40 size()

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDArray< T, Ndim >::size ( ) const  [inline]
```

7.13.3.41 span()

```
template<typename T , ssize_t Ndim = 2>
NDView< T, Ndim > aare::NDArray< T, Ndim >::span ( ) const  [inline]
```

7.13.3.42 sqrt()

```
template<typename T , ssize_t Ndim = 2>
void aare::NDArray< T, Ndim >::sqrt ( )  [inline]
```

7.13.3.43 strides()

```
template<typename T , ssize_t Ndim = 2>
std::array< ssize_t, Ndim > aare::NDArray< T, Ndim >::strides ( ) const  [inline], [noexcept]
```

7.13.3.44 total_bytes()

```
template<typename T , ssize_t Ndim = 2>
size_t aare::NDArray< T, Ndim >::total_bytes ( ) const  [inline]
```

7.13.3.45 value()

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T >::type aare::NDArray< T, Ndim >::value (
             Ix... index ) [inline]
```

7.13.4 Field Documentation

7.13.4.1 data_

```
template<typename T , ssize_t Ndim = 2>
T* aare::NDArray< T, Ndim >::data_ [private]
```

7.13.4.2 shape_

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDArray< T, Ndim >::shape_ [private]
```

7.13.4.3 size_

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDArray< T, Ndim >::size_ [private]
```

7.13.4.4 strides_

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDArray< T, Ndim >::strides_ [private]
```

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

- core/include/aare/core/NDArray.hpp

7.14 aare::NDView< T, Ndim > Class Template Reference

```
#include <NDView.hpp>
```

Public Member Functions

- NDView ()
- NDView (T ∗buffer, std::array< ssize_t, Ndim > shape)
- NDView (T ∗buffer, const std::vector< ssize_t > &shape)
- template<typename... Ix>
  std::enable_if< sizeof...(Ix)==Ndim, T & >::type operator() (Ix... index)
- template<typename... Ix>
  std::enable_if< sizeof...(Ix)==Ndim, T & >::type operator() (Ix... index) const
- ssize_t size () const
- NDView (const NDView &)=default
- NDView (NDView &&)=default
- T ∗ begin ()
- T ∗ end ()
- T & operator() (ssize_t i)
- T & operator[ ] (ssize_t i)
- bool operator== (const NDView &other) const
- NDView & operator+= (const T val)
- NDView & operator-= (const T val)
- NDView & operator∗= (const T val)
- NDView & operator/= (const T val)
- NDView & operator/= (const NDView &other)
- NDView & operator= (const T val)
- NDView & operator= (const NDView &other)
- auto & shape ()
- auto shape (ssize_t i) const
- T ∗ data ()

Private Member Functions

- template<class BinaryOperation >
  NDView & elemenwise (T val, BinaryOperation op)
- template<class BinaryOperation >
  NDView & elemenwise (const NDView &other, BinaryOperation op)

Private Attributes

- T ∗ buffer_ {nullptr}
- std::array< ssize_t, Ndim > strides_ {}
- std::array< ssize_t, Ndim > shape_ {}
- ssize_t size_ {}

7.14.1 Constructor & Destructor Documentation

7.14.1.1 NDView() [1/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView ( )  [inline]
```

7.14.1.2 NDView() [2/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
             T * buffer,
             std::array< ssize_t, Ndim > shape )  [inline]
```

7.14.1.3 NDView() [3/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
             T * buffer,
             const std::vector< ssize_t > & shape )  [inline]
```

7.14.1.4 NDView() [4/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
             const NDView< T, Ndim > &  )  [default]
```

7.14.1.5 NDView() [5/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
             NDView< T, Ndim > &&  )  [default]
```

7.14.2 Member Function Documentation

7.14.2.1 begin()

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDView< T, Ndim >::begin ( )  [inline]
```

7.14.2.2 data()

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDView< T, Ndim >::data ( )  [inline]
```

7.14.2.3 elemenwise() [1/2]

```
template<typename T , ssize_t Ndim = 2>
template<class BinaryOperation >
NDView & aare::NDView< T, Ndim >::elemenwise (
             const NDView< T, Ndim > & other,
             BinaryOperation op )  [inline], [private]
```

7.14.2.4 elemenwise() [2/2]

```
template<typename T , ssize_t Ndim = 2>
template<class BinaryOperation >
NDView & aare::NDView< T, Ndim >::elemenwise (
             T val,
             BinaryOperation op )  [inline], [private]
```

7.14.2.5 end()

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDView< T, Ndim >::end ( )  [inline]
```

7.14.2.6 operator()() [1/3]

```
template<typename T , ssize_t Ndim = 2>
template<typename...  Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDView< T, Ndim >::operator() (
             Ix...  index )  [inline]
```

7.14.2.7 operator()() [2/3]

```
template<typename T , ssize_t Ndim = 2>
template<typename...  Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDView< T, Ndim >::operator() (
             Ix...  index ) const  [inline]
```

7.14.2.8 operator()() [3/3]

```
template<typename T , ssize_t Ndim = 2>
T & aare::NDView< T, Ndim >::operator() (
             ssize_t i )  [inline]
```

7.14.2.9 operator*=()

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator*= (
             const T val )  [inline]
```

7.14.2.10 operator+=()

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator+= (
             const T val )  [inline]
```

7.14.2.11 operator-=()

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator-= (
             const T val )  [inline]
```

7.14.2.12 operator/=() [1/2]

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator/= (
             const NDView< T, Ndim > & other )  [inline]
```

7.14.2.13 operator/=() [2/2]

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator/= (
             const T val )  [inline]
```

7.14.2.14 operator=() [1/2]

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator= (
             const NDView< T, Ndim > & other )  [inline]
```

7.14.2.15 operator=() [2/2]

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator= (
             const T val )  [inline]
```

7.14.2.16 operator==()

```
template<typename T , ssize_t Ndim = 2>
bool aare::NDView< T, Ndim >::operator== (
             const NDView< T, Ndim > & other ) const  [inline]
```

7.14.2.17 operator[]()

```
template<typename T , ssize_t Ndim = 2>
T & aare::NDView< T, Ndim >::operator[ ] (
             ssize_t i )  [inline]
```

7.14.2.18 shape() [1/2]

```
template<typename T , ssize_t Ndim = 2>
auto & aare::NDView< T, Ndim >::shape ( )  [inline]
```

7.14.2.19 shape() [2/2]

```
template<typename T , ssize_t Ndim = 2>
auto aare::NDView< T, Ndim >::shape (
             ssize_t i ) const  [inline]
```

7.14.2.20 size()

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDView< T, Ndim >::size ( ) const  [inline]
```

7.14.3 Field Documentation

7.14.3.1 buffer_

```
template<typename T , ssize_t Ndim = 2>
T* aare::NDView< T, Ndim >::buffer_ {nullptr}  [private]
```

7.14.3.2 shape_

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDView< T, Ndim >::shape_ {}  [private]
```

7.14.3.3 size_

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDView< T, Ndim >::size_ {}  [private]
```

7.14.3.4 strides_

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDView< T, Ndim >::strides_ {}  [private]
```

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

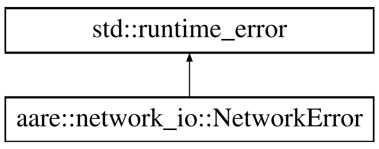
- core/include/aare/core/NDView.hpp

7.15 aare::network_io::NetworkError Class Reference

NetworkError exception class.

```
#include <defs.hpp>
```

Inheritance diagram for aare::network_io::NetworkError:



Public Member Functions

- NetworkError (const char ∗msg)
- NetworkError (const std::string msg)
- virtual const char ∗ what () const noexcept override

Private Attributes

- const char ∗ m_msg

7.15.1 Detailed Description

NetworkError exception class.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 NetworkError() [1/2]

```
aare::network_io::NetworkError::NetworkError (
             const char * msg )  [inline]
```

7.15.2.2 NetworkError() [2/2]

```
aare::network_io::NetworkError::NetworkError (
             const std::string msg )  [inline]
```

7.15.3 Member Function Documentation

7.15.3.1 what()

```
virtual const char * aare::network_io::NetworkError::what ( ) const  [inline], [override],
[virtual], [noexcept]
```

7.15.4 Field Documentation

7.15.4.1 m_msg

```
const char* aare::network_io::NetworkError::m_msg  [private]
```

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

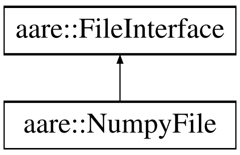
- network_io/include/aare/network_io/defs.hpp

7.16 aare::NumpyFile Class Reference

NumpyFile class to read and write numpy files.

```
#include <NumpyFile.hpp>
```

Inheritance diagram for aare::NumpyFile:



Public Member Functions

- NumpyFile (const std::filesystem::path &fname, const std::string &mode="r", FileConfig cfg={})

  NumpyFile constructor.
- void write (Frame &frame) override

  write a frame to the file
- Frame read () override

  write a vector of frames to the file
- std::vector< Frame > read (size_t n_frames) override

  read n_frames from the file at the current position
- void read_into (std::byte ∗image_buf) override

  read one frame from the file at the current position and store it in the provided buffer
- void read_into (std::byte ∗image_buf, size_t n_frames) override

  read n_frames from the file at the current position and store them in the provided buffer
- size_t frame_number (size_t frame_index) override

  get the frame number at the given frame index
- size_t bytes_per_frame () override

  get the size of one frame in bytes
- size_t pixels () override

  get the number of pixels in one frame
- void seek (size_t frame_number) override

  seek to the given frame number
- size_t tell () override

  get the current position of the file pointer
- size_t total_frames () const override

  get the total number of frames in the file
- ssize_t rows () const override

  get the number of rows in the file
- ssize_t cols () const override

  get the number of columns in the file
- ssize_t bitdepth () const override

  get the bitdepth of the file
- DType dtype () const

  get the data type of the numpy file
- std::vector< size_t > shape () const

  get the shape of the numpy file

- template<typename T , size_t NDim>
  NDArray< T, NDim > load ()

  load the numpy file into an NDArray
- ∼NumpyFile ()
- Frame iread (size_t frame_number)

  read one frame from the file at the given frame number
- std::vector< Frame > iread (size_t frame_number, size_t n_frames)

  read n_frames from the file starting at the given frame number

Data Fields

- std::string m_mode
- std::filesystem::path m_fname
- std::filesystem::path m_base_path
- std::string m_base_name
- std::string m_ext
- int m_findex
- size_t m_total_frames {}
- size_t max_frames_per_file {}
- std::string version
- DetectorType m_type
- ssize_t m_rows {}
- ssize_t m_cols {}
- ssize_t m_bitdepth {}

Private Member Functions

- void load_metadata ()
- void get_frame_into (size_t, std::byte ∗)
- Frame get_frame (size_t frame_number)

Private Attributes

- FILE ∗ fp = nullptr
- size_t initial_header_len = 0
- size_t current_frame {}
- uint32_t header_len {}
- uint8_t header_len_size {}
- size_t header_size {}
- NumpyHeader m_header
- uint8_t major_ver_ {}
- uint8_t minor_ver_ {}

7.16.1 Detailed Description

NumpyFile class to read and write numpy files.

Note

derived from FileInterface

implements all the pure virtual functions from FileInterface

documentation for the functions can also be found in the FileInterface class

7.16.2 Constructor & Destructor Documentation

7.16.2.1 NumpyFile()

```
aare::NumpyFile::NumpyFile (
            const std::filesystem::path & fname,
            const std::string & mode = "r",
            FileConfig cfg = {} )
```

NumpyFile constructor.

Parameters

fname	path to the numpy file
mode	file mode (r, w)
cfg	file configuration

7.16.2.2 ∼NumpyFile()

```
aare::NumpyFile::∼NumpyFile ( )
```

7.16.3 Member Function Documentation

7.16.3.1 bitdepth()

```
ssize_t aare::NumpyFile::bitdepth ( ) const [inline], [override], [virtual]
```

get the bitdepth of the file

Returns

bitdepth of the file

Implements aare::FileInterface.

7.16.3.2 bytes_per_frame()

```
size_t aare::NumpyFile::bytes_per_frame ( ) [override], [virtual]
```

get the size of one frame in bytes

Returns

size of one frame

Implements aare::FileInterface.

7.16.3.3 cols()

```
ssize_t aare::NumpyFile::cols ( ) const  [inline], [override], [virtual]
```

get the number of columns in the file

Returns

number of columns in the file

Implements aare::FileInterface.

7.16.3.4 dtype()

```
DType aare::NumpyFile::dtype ( ) const  [inline]
```

get the data type of the numpy file

Returns

DType

7.16.3.5 frame_number()

```
size_t aare::NumpyFile::frame_number (
            size_t frame_index )  [inline], [override], [virtual]
```

get the frame number at the given frame index

Parameters

frame_index	index of the frame

Returns

frame number

Implements aare::FileInterface.

7.16.3.6 get_frame()

```
Frame aare::NumpyFile::get_frame (
            size_t frame_number )  [private]
```

7.16.3.7 get_frame_into()

```
void aare::NumpyFile::get_frame_into (
            size_t frame_number,
            std::byte * image_buf )  [private]
```

7.16.3.8 iread() [1/2]

```
Frame aare::FileInterface::iread (
            size_t frame_number )  [inline], [inherited]
```

read one frame from the file at the given frame number

Parameters

frame_number	frame number to read

Returns

frame

7.16.3.9 iread() [2/2]

```
std::vector< Frame > aare::FileInterface::iread (
            size_t frame_number,
            size_t n_frames )  [inline], [inherited]
```

read n_frames from the file starting at the given frame number

Parameters

frame_number	frame number to start reading from
n_frames	number of frames to read

Returns

vector of frames

7.16.3.10 load()

```
template<typename T , size_t NDim>
NDArray< T, NDim > aare::NumpyFile::load ( )  [inline]
```

load the numpy file into an NDArray

Template Parameters

T	data type of the NDArray
NDim	number of dimensions of the NDArray

Returns

NDArray<T, NDim>

7.16.3.11 load_metadata()

```
void aare::NumpyFile::load_metadata ( )  [private]
```

7.16.3.12 pixels()

```
size_t aare::NumpyFile::pixels ( )  [override], [virtual]
```

get the number of pixels in one frame

Returns

number of pixels in one frame

Implements aare::FileInterface.

7.16.3.13 read() [1/2]

```
Frame aare::NumpyFile::read ( )  [inline], [override], [virtual]
```

write a vector of frames to the file

Parameters

frames	vector of frames to write

Returns

void

read one frame from the file at the current position

Returns

Frame

Implements aare::FileInterface.

7.16.3.14 read() [2/2]

```
std::vector< Frame > aare::NumpyFile::read (
             size_t n_frames )  [override], [virtual]
```

read n_frames from the file at the current position

Parameters

n_frames	number of frames to read

Returns

vector of frames

Implements aare::FileInterface.

7.16.3.15 read_into() [1/2]

```
void aare::NumpyFile::read_into (
            std::byte * image_buf )  [inline], [override], [virtual]
```

read one frame from the file at the current position and store it in the provided buffer

Parameters

image_buf	buffer to store the frame

Returns

void

Implements aare::FileInterface.

7.16.3.16 read_into() [2/2]

```
void aare::NumpyFile::read_into (
            std::byte * image_buf,
            size_t n_frames )  [override], [virtual]
```

read n_frames from the file at the current position and store them in the provided buffer

Parameters

image_buf	buffer to store the frames
n_frames	number of frames to read

Returns

void

Implements aare::FileInterface.

7.16.3.17 rows()

```
ssize_t aare::NumpyFile::rows ( ) const  [inline], [override], [virtual]
```

get the number of rows in the file

Returns

number of rows in the file

Implements aare::FileInterface.

7.16.3.18 seek()

```
void aare::NumpyFile::seek (
            size_t frame_number )  [inline], [override], [virtual]
```

seek to the given frame number

Parameters

frame_number	frame number to seek to

Returns

void

Implements aare::FileInterface.

7.16.3.19 shape()

```
std::vector< size_t > aare::NumpyFile::shape ( ) const  [inline]
```

get the shape of the numpy file

Returns

vector of type size_t

7.16.3.20 tell()

```
size_t aare::NumpyFile::tell ( )  [inline], [override], [virtual]
```

get the current position of the file pointer

Returns

current position of the file pointer

Implements aare::FileInterface.

7.16.3.21 total_frames()

```
size_t aare::NumpyFile::total_frames ( ) const  [inline], [override], [virtual]
```

get the total number of frames in the file

Returns

total number of frames in the file

Implements aare::FileInterface.

7.16.3.22 write()

```
void aare::NumpyFile::write (
            Frame & frame )  [override], [virtual]
```

write a frame to the file

Parameters

frame	frame to write

Returns

void

Exceptions

std::runtime_error	if the function is not implemented

Implements aare::FileInterface.

7.16.4 Field Documentation

7.16.4.1 current_frame

```
size_t aare::NumpyFile::current_frame {}  [private]
```

7.16.4.2 fp

```
FILE* aare::NumpyFile::fp = nullptr  [private]
```

7.16.4.3 header_len

```
uint32_t aare::NumpyFile::header_len {}  [private]
```

7.16.4.4 header_len_size

```
uint8_t aare::NumpyFile::header_len_size {}  [private]
```

7.16.4.5 header_size

```
size_t aare::NumpyFile::header_size {}  [private]
```

7.16.4.6 initial_header_len

```
size_t aare::NumpyFile::initial_header_len = 0  [private]
```

7.16.4.7 m_base_name

```
std::string aare::FileInterface::m_base_name  [inherited]
```

7.16.4.8 m_base_path

```
std::filesystem::path aare::FileInterface::m_base_path  [inherited]
```

7.16.4.9 m_bitdepth

```
ssize_t aare::FileInterface::m_bitdepth {}  [inherited]
```

7.16.4.10 m_cols

```
ssize_t aare::FileInterface::m_cols {}  [inherited]
```

7.16.4.11 m_ext

```
std::string aare::FileInterface::m_ext  [inherited]
```

7.16.4.12 m_findex

```
int aare::FileInterface::m_findex  [inherited]
```

7.16.4.13 m_fname

```
std::filesystem::path aare::FileInterface::m_fname  [inherited]
```

7.16.4.14 m_header

```
NumpyHeader aare::NumpyFile::m_header  [private]
```

7.16.4.15 m_mode

```
std::string aare::FileInterface::m_mode  [inherited]
```

7.16.4.16 m_rows

```
ssize_t aare::FileInterface::m_rows {}  [inherited]
```

7.16.4.17 m_total_frames

```
size_t aare::FileInterface::m_total_frames {}  [inherited]
```

7.16.4.18 m_type

```
DetectorType aare::FileInterface::m_type  [inherited]
```

7.16.4.19 major_ver_

```
uint8_t aare::NumpyFile::major_ver_ {}  [private]
```

7.16.4.20 max_frames_per_file

```
size_t aare::FileInterface::max_frames_per_file {}  [inherited]
```

7.16.4.21 minor_ver_

```
uint8_t aare::NumpyFile::minor_ver_ {}  [private]
```

7.16.4.22 version

```
std::string aare::FileInterface::version  [inherited]
```

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

- file_io/include/aare/file_io/NumpyFile.hpp
- file_io/src/NumpyFile.cpp

7.17 aare::NumpyHeader Struct Reference

```
#include <NumpyHelpers.hpp>
```

Public Member Functions

- std::string to_string () const

Data Fields

- DType dtype {aare::DType::ERROR}
- bool fortran_order {false}
- shape_t shape {}

7.17.1 Member Function Documentation

7.17.1.1 to_string()

```
std::string aare::NumpyHeader::to_string ( ) const
```

7.17.2 Field Documentation

7.17.2.1 dtype

```
DType aare::NumpyHeader::dtype {aare::DType::ERROR}
```

7.17.2.2 fortran_order

```
bool aare::NumpyHeader::fortran_order {false}
```

7.17.2.3 shape

```
shape_t aare::NumpyHeader::shape {}
```

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

- file_io/include/aare/file_io/NumpyHelpers.hpp
- file_io/src/NumpyHelpers.cpp

7.18 folly::ProducerConsumerQueue< T > Struct Template Reference

```
#include <ProducerConsumerQueue.hpp>
```

Public Types

- typedef T value_type

Public Member Functions

- ProducerConsumerQueue (const ProducerConsumerQueue &)=delete
- ProducerConsumerQueue & operator= (const ProducerConsumerQueue &)=delete
- ProducerConsumerQueue (uint32_t size)
- ∼ProducerConsumerQueue ()
- template<class... Args>
  bool write (Args &&...recordArgs)
- bool read (T &record)
- T ∗ frontPtr ()
- void popFront ()
- bool isEmpty () const
- bool isFull () const
- size_t sizeGuess () const
- size_t capacity () const

Private Types

- using AtomicIndex = std::atomic< unsigned int >

Private Attributes

- char pad0_ [hardware_destructive_interference_size]
- const uint32_t size_
- T ∗const records_
- AtomicIndex readIndex_
- AtomicIndex writeIndex_
- char pad1_ [hardware_destructive_interference_size - sizeof(AtomicIndex)]

7.18.1 Member Typedef Documentation

7.18.1.1 AtomicIndex

```
template<class T >
using folly::ProducerConsumerQueue< T >::AtomicIndex = std::atomic<unsigned int>  [private]
```

7.18.1.2 value_type

```
template<class T >
typedef T folly::ProducerConsumerQueue< T >::value_type
```

7.18.2 Constructor & Destructor Documentation

7.18.2.1 ProducerConsumerQueue() [1/2]

```
template<class T >
folly::ProducerConsumerQueue< T >::ProducerConsumerQueue (
             const ProducerConsumerQueue< T > &  )  [delete]
```

7.18.2.2 ProducerConsumerQueue() [2/2]

```
template<class T >
folly::ProducerConsumerQueue< T >::ProducerConsumerQueue (
             uint32_t size )  [inline], [explicit]
```

7.18.2.3 ∼ProducerConsumerQueue()

```
template<class T >
folly::ProducerConsumerQueue< T >::∼ProducerConsumerQueue ( )  [inline]
```

7.18.3 Member Function Documentation

7.18.3.1 capacity()

```
template<class T >
size_t folly::ProducerConsumerQueue< T >::capacity ( ) const  [inline]
```

7.18.3.2 frontPtr()

```
template<class T >
T * folly::ProducerConsumerQueue< T >::frontPtr ( )  [inline]
```

7.18.3.3 isEmpty()

```
template<class T >
bool folly::ProducerConsumerQueue< T >::isEmpty ( ) const  [inline]
```

7.18.3.4 isFull()

```
template<class T >
bool folly::ProducerConsumerQueue< T >::isFull ( ) const  [inline]
```

7.18.3.5 operator=()

```
template<class T >
ProducerConsumerQueue & folly::ProducerConsumerQueue< T >::operator= (
             const ProducerConsumerQueue< T > &  )  [delete]
```

7.18.3.6 popFront()

```
template<class T >
void folly::ProducerConsumerQueue< T >::popFront ( )  [inline]
```

7.18.3.7 read()

```
template<class T >
bool folly::ProducerConsumerQueue< T >::read (
             T & record )  [inline]
```

7.18.3.8 sizeGuess()

```
template<class T >
size_t folly::ProducerConsumerQueue< T >::sizeGuess ( ) const  [inline]
```

7.18.3.9 write()

```
template<class T >
template<class...  Args>
bool folly::ProducerConsumerQueue< T >::write (
             Args &&...  recordArgs )  [inline]
```

7.18.4 Field Documentation

7.18.4.1 pad0_

```
template<class T >
char folly::ProducerConsumerQueue< T >::pad0_[hardware_destructive_interference_size]  [private]
```

7.18.4.2 pad1_

```
template<class T >
char folly::ProducerConsumerQueue< T >::pad1_[hardware_destructive_interference_size - sizeof(AtomicIndex)]
[private]
```

7.18.4.3 readIndex_

```
template<class T >
AtomicIndex folly::ProducerConsumerQueue< T >::readIndex_  [private]
```

7.18.4.4 records_

```
template<class T >
T* const folly::ProducerConsumerQueue< T >::records_  [private]
```

7.18.4.5 size_

```
template<class T >
const uint32_t folly::ProducerConsumerQueue< T >::size_  [private]
```

7.18.4.6 writeIndex_

```
template<class T >
AtomicIndex folly::ProducerConsumerQueue< T >::writeIndex_  [private]
```

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

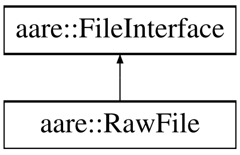
- core/include/aare/core/ProducerConsumerQueue.hpp

7.19 aare::RawFile Class Reference

RawFile class to read .raw and .json files.

```
#include <RawFile.hpp>
```

Inheritance diagram for aare::RawFile:



Public Member Functions

- RawFile (const std::filesystem::path &fname, const std::string &mode="r", const FileConfig &cfg={})

  RawFile constructor.
- void write (Frame &frame) override

  write function is not implemented for RawFile
- Frame read () override

  write a vector of frames to the file
- std::vector< Frame > read (size_t n_frames) override

  read n_frames from the file at the current position
- void read_into (std::byte ∗image_buf) override

  read one frame from the file at the current position and store it in the provided buffer
- void read_into (std::byte ∗image_buf, size_t n_frames) override

  read n_frames from the file at the current position and store them in the provided buffer
- size_t frame_number (size_t frame_index) override

  get the frame number at the given frame index
- size_t bytes_per_frame () override

  get the number of bytess per frame
- size_t pixels () override

  get the number of pixels in the frame
- void seek (size_t frame_number) override

  seek to the given frame number
- size_t tell () override

  get the current position of the file pointer
- void set_config (int row, int col)

  set the module gap row and column
- void find_number_of_subfiles ()

  get the number of subfiles for the RawFile
- std::filesystem::path master_fname ()

  get the master file name path for the RawFile
- std::filesystem::path data_fname (int mod_id, int file_id)

  get the data file name path for the RawFile with the given module id and file id
- ∼RawFile ()

  destructor: will delete the subfiles
- size_t total_frames () const override

  get the total number of frames in the file
- ssize_t rows () const override

  get the number of rows in the file
- ssize_t cols () const override

  get the number of columns in the file
- ssize_t bitdepth () const override

  get the bitdepth of the file
- Frame iread (size_t frame_number)

  read one frame from the file at the given frame number
- std::vector< Frame > iread (size_t frame_number, size_t n_frames)

  read n_frames from the file starting at the given frame number

Static Public Member Functions

- static bool is_master_file (std::filesystem::path fpath)

  check if the file is a master file

Data Fields

- std::string m_mode
- std::filesystem::path m_fname
- std::filesystem::path m_base_path
- std::string m_base_name
- std::string m_ext
- int m_findex
- size_t m_total_frames {}
- size_t max_frames_per_file {}
- std::string version
- DetectorType m_type
- ssize_t m_rows {}
- ssize_t m_cols {}
- ssize_t m_bitdepth {}
- size_t current_frame {}

Private Member Functions

- void get_frame_into (size_t frame_number, std::byte ∗image_buf)

  read the frame at the given frame number into the image buffer
- Frame get_frame (size_t frame_number)

  get the frame at the given frame number
- void parse_fname ()

  parse the file name to get the extension, base name and index
- void parse_metadata ()

  parse the metadata from the file
- void parse_raw_metadata ()

  parse the metadata of a .raw file
- void parse_json_metadata ()

  parse the metadata of a .json file
- void find_geometry ()

  finds the geometry of the file
- sls_detector_header read_header (const std::filesystem::path &fname)

  read the header of the file
- void open_subfiles ()

  open the subfiles

Private Attributes

- size_t n_subfiles
- size_t n_subfile_parts
- std::vector< std::vector< SubFile ∗ > > subfiles
- int subfile_rows
- int subfile_cols
- xy geometry
- std::vector< xy > positions
- RawFileConfig cfg {0, 0}
- TimingMode timing_mode
- bool quad {false}

7.19.1 Detailed Description

RawFile class to read .raw and .json files.

Note

derived from FileInterface

documentation can also be found in the FileInterface class

7.19.2 Constructor & Destructor Documentation

7.19.2.1 RawFile()

```
aare::RawFile::RawFile (
            const std::filesystem::path & fname,
            const std::string & mode = "r",
            const FileConfig & cfg = {} )
```

RawFile constructor.

Parameters

fname	path to the file
mode	file mode (r, w)
cfg	file configuration

7.19.2.2 ∼RawFile()

```
aare::RawFile::~RawFile ( )
```

destructor: will delete the subfiles

7.19.3 Member Function Documentation

7.19.3.1 bitdepth()

```
ssize_t aare::RawFile::bitdepth ( ) const  [inline], [override], [virtual]
```

get the bitdepth of the file

Returns

bitdepth of the file

Implements aare::FileInterface.

7.19.3.2 bytes_per_frame()

```
size_t aare::RawFile::bytes_per_frame ( )  [inline], [override], [virtual]
```

get the number of bytess per frame

Returns

size of one frame in bytes

Implements aare::FileInterface.

7.19.3.3 cols()

```
ssize_t aare::RawFile::cols ( ) const  [inline], [override], [virtual]
```

get the number of columns in the file

Returns

number of columns in the file

Implements aare::FileInterface.

7.19.3.4 data_fname()

```
std::filesystem::path aare::RawFile::data_fname (
             int mod_id,
             int file_id )  [inline]
```

get the data file name path for the RawFile with the given module id and file id

Parameters

mod← _id	module id
file_id	file id

Returns

path to the data file

7.19.3.5 find_geometry()

```
void aare::RawFile::find_geometry ( )  [private]
```

finds the geometry of the file

7.19.3.6 find_number_of_subfiles()

```
void aare::RawFile::find_number_of_subfiles ( )
```

get the number of subfiles for the RawFile

Returns

number of subfiles

7.19.3.7 frame_number()

```
size_t aare::RawFile::frame_number (
             size_t frame_index )  [override], [virtual]
```

get the frame number at the given frame index

Parameters

frame_index	index of the frame

Returns

frame number

Implements aare::FileInterface.

7.19.3.8 get_frame()

```
Frame aare::RawFile::get_frame (
             size_t frame_number )  [private]
```

get the frame at the given frame number

Parameters

frame_number	frame number to read

Returns

Frame

7.19.3.9 get_frame_into()

```
void aare::RawFile::get_frame_into (
             size_t frame_number,
             std::byte * image_buf )  [private]
```

read the frame at the given frame number into the image buffer

Parameters

frame_number	frame number to read
image_buf	buffer to store the frame

7.19.3.10 iread() `[1/2]`

```
Frame aare::FileInterface::iread (
            size_t frame_number )  [inline], [inherited]
```

read one frame from the file at the given frame number

Parameters

frame_number	frame number to read

Returns

frame

7.19.3.11 iread() `[2/2]`

```
std::vector< Frame > aare::FileInterface::iread (
            size_t frame_number,
            size_t n_frames )  [inline], [inherited]
```

read n_frames from the file starting at the given frame number

Parameters

frame_number	frame number to start reading from
n_frames	number of frames to read

Returns

vector of frames

7.19.3.12 is_master_file()

```
bool aare::RawFile::is_master_file (
            std::filesystem::path fpath )  [static]
```

check if the file is a master file

Parameters

fpath	path to the file

7.19.3.13 master_fname()

```
std::filesystem::path aare::RawFile::master_fname ( )  [inline]
```

get the master file name path for the RawFile

Returns

path to the master file

7.19.3.14 open_subfiles()

```
void aare::RawFile::open_subfiles ( )  [private]
```

open the subfiles

7.19.3.15 parse_fname()

```
void aare::RawFile::parse_fname ( )  [private]
```

parse the file name to get the extension, base name and index

7.19.3.16 parse_json_metadata()

```
void aare::RawFile::parse_json_metadata ( )  [private]
```

parse the metadata of a .json file

7.19.3.17 parse_metadata()

```
void aare::RawFile::parse_metadata ( )  [private]
```

parse the metadata from the file

7.19.3.18 parse_raw_metadata()

```
void aare::RawFile::parse_raw_metadata ( )  [private]
```

parse the metadata of a .raw file

7.19.3.19 pixels()

```
size_t aare::RawFile::pixels ( )  [inline], [override], [virtual]
```

get the number of pixels in the frame

Returns

number of pixels

Implements aare::FileInterface.

7.19.3.20 read() [1/2]

```
Frame aare::RawFile::read ( )  [inline], [override], [virtual]
```

write a vector of frames to the file

Parameters

frames	vector of frames to write

Returns

void

read one frame from the file at the current position

Returns

Frame

Implements aare::FileInterface.

7.19.3.21 read() [2/2]

```
std::vector< Frame > aare::RawFile::read (
            size_t n_frames ) [override], [virtual]
```

read n_frames from the file at the current position

Parameters

n_frames	number of frames to read

Returns

vector of frames

Implements aare::FileInterface.

7.19.3.22 read_header()

```
sls_detector_header aare::RawFile::read_header (
            const std::filesystem::path & fname ) [private]
```

read the header of the file

Parameters

fname	path to the data subfile

Returns

sls_detector_header

7.19.3.23 read_into() [1/2]

```
void aare::RawFile::read_into (
             std::byte * image_buf )  [inline], [override], [virtual]
```

read one frame from the file at the current position and store it in the provided buffer

Parameters

image_buf	buffer to store the frame

Returns

void

Implements aare::FileInterface.

7.19.3.24 read_into() [2/2]

```
void aare::RawFile::read_into (
             std::byte * image_buf,
             size_t n_frames )  [override], [virtual]
```

read n_frames from the file at the current position and store them in the provided buffer

Parameters

image_buf	buffer to store the frames
n_frames	number of frames to read

Returns

void

Implements aare::FileInterface.

7.19.3.25 rows()

```
ssize_t aare::RawFile::rows ( ) const  [inline], [override], [virtual]
```

get the number of rows in the file

Returns

number of rows in the file

Implements aare::FileInterface.

7.19.3.26 seek()

```
void aare::RawFile::seek (
             size_t frame_number )  [inline], [override], [virtual]
```

seek to the given frame number

Parameters

frame_number	frame number to seek to

Returns

void

Implements aare::FileInterface.

7.19.3.27 set_config()

```
void aare::RawFile::set_config (
             int row,
             int col )  [inline]
```

set the module gap row and column

Parameters

row	gap between rows
col	gap between columns

7.19.3.28 tell()

```
size_t aare::RawFile::tell ( )  [inline], [override], [virtual]
```

get the current position of the file pointer

Returns

current position of the file pointer

Implements aare::FileInterface.

7.19.3.29 total_frames()

```
size_t aare::RawFile::total_frames ( ) const  [inline], [override], [virtual]
```

get the total number of frames in the file

Returns

total number of frames in the file

Implements aare::FileInterface.

7.19.3.30 write()

```
void aare::RawFile::write (
             Frame & frame )  [inline], [override], [virtual]
```

write function is not implemented for RawFile

Parameters

frame	frame to write

Implements aare::FileInterface.

7.19.4 Field Documentation

7.19.4.1 cfg

```
RawFileConfig aare::RawFile::cfg {0, 0}  [private]
```

7.19.4.2 current_frame

```
size_t aare::FileInterface::current_frame {}  [inherited]
```

7.19.4.3 geometry

```
xy aare::RawFile::geometry  [private]
```

7.19.4.4 m_base_name

```
std::string aare::FileInterface::m_base_name  [inherited]
```

7.19.4.5 m_base_path

```
std::filesystem::path aare::FileInterface::m_base_path  [inherited]
```

7.19.4.6 m_bitdepth

```
ssize_t aare::FileInterface::m_bitdepth {}  [inherited]
```

7.19.4.7 m_cols

```
ssize_t aare::FileInterface::m_cols {}  [inherited]
```

7.19.4.8 m_ext

```
std::string aare::FileInterface::m_ext  [inherited]
```

7.19.4.9 m_findex

```
int aare::FileInterface::m_findex  [inherited]
```

7.19.4.10 m_fname

```
std::filesystem::path aare::FileInterface::m_fname  [inherited]
```

7.19.4.11 m_mode

```
std::string aare::FileInterface::m_mode  [inherited]
```

7.19.4.12 m_rows

```
ssize_t aare::FileInterface::m_rows {}  [inherited]
```

7.19.4.13 m_total_frames

```
size_t aare::FileInterface::m_total_frames {}  [inherited]
```

7.19.4.14 m_type

```
DetectorType aare::FileInterface::m_type  [inherited]
```

7.19.4.15 max_frames_per_file

```
size_t aare::FileInterface::max_frames_per_file {}  [inherited]
```

7.19.4.16 n_subfile_parts

```
size_t aare::RawFile::n_subfile_parts  [private]
```

7.19.4.17 n_subfiles

```
size_t aare::RawFile::n_subfiles  [private]
```

7.19.4.18 positions

```
std::vector<xy> aare::RawFile::positions  [private]
```

7.19.4.19 quad

```
bool aare::RawFile::quad {false}  [private]
```

7.19.4.20 subfile_cols

```
int aare::RawFile::subfile_cols  [private]
```

7.19.4.21 subfile_rows

```
int aare::RawFile::subfile_rows  [private]
```

7.19.4.22 subfiles

```
std::vector<std::vector<SubFile *> > aare::RawFile::subfiles  [private]
```

7.19.4.23 timing_mode

```
TimingMode aare::RawFile::timing_mode  [private]
```

7.19.4.24 version

```
std::string aare::FileInterface::version  [inherited]
```

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

- file_io/include/aare/file_io/RawFile.hpp
- file_io/src/RawFile.cpp

7.20 aare::RawFileConfig Struct Reference

```
#include <defs.hpp>
```

Public Member Functions

- bool operator== (const RawFileConfig &other) const

Data Fields

- int module_gap_row {}
- int module_gap_col {}

7.20.1 Member Function Documentation

7.20.1.1 operator==()

```
bool aare::RawFileConfig::operator== (
             const RawFileConfig & other ) const  [inline]
```

7.20.2 Field Documentation

7.20.2.1 module_gap_col

```
int aare::RawFileConfig::module_gap_col {}
```

7.20.2.2 module_gap_row

```
int aare::RawFileConfig::module_gap_row {}
```

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

- core/include/aare/core/defs.hpp

7.21 aare::sls_detector_header Struct Reference

```
#include <defs.hpp>
```

Data Fields

- uint64_t frameNumber
- uint32_t expLength
- uint32_t packetNumber
- uint64_t bunchId
- uint64_t timestamp
- uint16_t modId
- uint16_t row
- uint16_t column
- uint16_t reserved
- uint32_t debug
- uint16_t roundRNumber
- uint8_t detType
- uint8_t version
- uint8_t packetMask [64]

7.21.1 Field Documentation

7.21.1.1 bunchId

```
uint64_t aare::sls_detector_header::bunchId
```

7.21.1.2 column

```
uint16_t aare::sls_detector_header::column
```

7.21.1.3 debug

```
uint32_t aare::sls_detector_header::debug
```

7.21.1.4 detType

```
uint8_t aare::sls_detector_header::detType
```

7.21.1.5 expLength

```
uint32_t aare::sls_detector_header::expLength
```

7.21.1.6 frameNumber

```
uint64_t aare::sls_detector_header::frameNumber
```

7.21.1.7 modId

```
uint16_t aare::sls_detector_header::modId
```

7.21.1.8 packetMask

```
uint8_t aare::sls_detector_header::packetMask[64]
```

7.21.1.9 packetNumber

```
uint32_t aare::sls_detector_header::packetNumber
```

7.21.1.10 reserved

```
uint16_t aare::sls_detector_header::reserved
```

7.21.1.11 roundRNumber

```
uint16_t aare::sls_detector_header::roundRNumber
```

7.21.1.12 row

```
uint16_t aare::sls_detector_header::row
```

7.21.1.13 timestamp

```
uint64_t aare::sls_detector_header::timestamp
```

7.21.1.14 version

```
uint8_t aare::sls_detector_header::version
```

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

- core/include/aare/core/defs.hpp

7.22 aare::SubFile Class Reference

Class to read a subfile from a RawFile.

```
#include <SubFile.hpp>
```

Public Member Functions

- SubFile (std::filesystem::path fname, DetectorType detector, ssize_t rows, ssize_t cols, uint16_t bitdepth)

  SubFile constructor.
- size_t read_impl_normal (std::byte ∗buffer)

  read the subfile into a buffer
- template<typename DataType >
  size_t read_impl_flip (std::byte ∗buffer)

  read the subfile into a buffer with the bytes flipped
- template<typename DataType >
  size_t read_impl_reorder (std::byte ∗buffer)

  read the subfile into a buffer with the bytes reordered
- size_t get_part (std::byte ∗buffer, int frame_number)

  read the subfile into a buffer with the bytes reordered and flipped
- size_t frame_number (int frame_index)
- size_t bytes_per_part ()
- size_t pixels_per_part ()

Protected Types

- using pfunc = size_t(SubFile::∗)(std::byte ∗)

  type of the read_impl function pointer

Protected Attributes

- pfunc read_impl = nullptr
- std::map< std::pair< DetectorType, int >, pfunc > read_impl_map

    map to store the read_impl functions for different detectors
- FILE ∗ fp = nullptr
- ssize_t m_bitdepth
- std::filesystem::path m_fname
- ssize_t m_rows {}
- ssize_t m_cols {}
- ssize_t n_frames {}
- int m_sub_file_index_ {}

7.22.1 Detailed Description

Class to read a subfile from a RawFile.

7.22.2 Member Typedef Documentation

7.22.2.1 pfunc

```
using aare::SubFile::pfunc = size_t (SubFile::*)(std::byte *)  [protected]
```

type of the read_impl function pointer

Parameters

buffer	pointer to the buffer to read the data into

Returns

number of bytes read

7.22.3 Constructor & Destructor Documentation

7.22.3.1 SubFile()

```
aare::SubFile::SubFile (
            std::filesystem::path fname,
            DetectorType detector,
            ssize_t rows,
            ssize_t cols,
            uint16_t bitdepth )
```

SubFile constructor.

Parameters

fname	path to the subfile

Parameters

detector	detector type
rows	number of rows in the subfile
cols	number of columns in the subfile
bitdepth	bitdepth of the subfile

Exceptions

std::invalid_argument	if the detector,type pair is not supported

7.22.4 Member Function Documentation

7.22.4.1 bytes_per_part()

```
size_t aare::SubFile::bytes_per_part ( )  [inline]
```

7.22.4.2 frame_number()

```
size_t aare::SubFile::frame_number (
             int frame_index )
```

7.22.4.3 get_part()

```
size_t aare::SubFile::get_part (
             std::byte * buffer,
             int frame_number )
```

read the subfile into a buffer with the bytes reordered and flipped

Parameters

buffer	pointer to the buffer to read the data into
frame_number	frame number to read

Returns

number of bytes read

7.22.4.4 pixels_per_part()

```
size_t aare::SubFile::pixels_per_part ( )  [inline]
```

7.22.4.5 read_impl_flip()

```
template<typename DataType >
size_t aare::SubFile::read_impl_flip (
            std::byte * buffer )
```

read the subfile into a buffer with the bytes flipped

Parameters

buffer	pointer to the buffer to read the data into

Returns

number of bytes read

7.22.4.6 read_impl_normal()

```
size_t aare::SubFile::read_impl_normal (
            std::byte * buffer )
```

read the subfile into a buffer

Parameters

buffer	pointer to the buffer to read the data into

Returns

number of bytes read

7.22.4.7 read_impl_reorder()

```
template<typename DataType >
size_t aare::SubFile::read_impl_reorder (
            std::byte * buffer )
```

read the subfile into a buffer with the bytes reordered

Parameters

buffer	pointer to the buffer to read the data into

Returns

number of bytes read

7.22.5 Field Documentation

7.22.5.1 fp

```
FILE* aare::SubFile::fp = nullptr  [protected]
```

7.22.5.2 m_bitdepth

```
ssize_t aare::SubFile::m_bitdepth  [protected]
```

7.22.5.3 m_cols

```
ssize_t aare::SubFile::m_cols {}  [protected]
```

7.22.5.4 m_fname

```
std::filesystem::path aare::SubFile::m_fname  [protected]
```

7.22.5.5 m_rows

```
ssize_t aare::SubFile::m_rows {}  [protected]
```

7.22.5.6 m_sub_file_index_

```
int aare::SubFile::m_sub_file_index_ {}  [protected]
```

7.22.5.7 n_frames

```
ssize_t aare::SubFile::n_frames {}  [protected]
```

7.22.5.8 read_impl

```
pfunc aare::SubFile::read_impl = nullptr  [protected]
```

7.22.5.9 read_impl_map

```
std::map<std::pair<DetectorType, int>, pfunc> aare::SubFile::read_impl_map  [protected]
```

Initial value:

```
= {
        {{DetectorType::Moench, 16}, &SubFile::read_impl_reorder<uint16_t>},
        {{DetectorType::Jungfrau, 16}, &SubFile::read_impl_normal},
        {{DetectorType::ChipTestBoard, 16}, &SubFile::read_impl_normal},
        {{DetectorType::Mythen3, 32}, &SubFile::read_impl_normal},
        {{DetectorType::Eiger, 32}, &SubFile::read_impl_normal},
        {{DetectorType::Eiger, 16}, &SubFile::read_impl_normal}

    }
```

map to store the read_impl functions for different detectors

Note

the key is a pair of DetectorType and bitdepth

the value is a pointer to the read_impl function specific for the detector

the read_impl function will be set to the appropriate function in the constructor

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

- file_io/include/aare/file_io/SubFile.hpp
- file_io/src/SubFile.cpp

7.23 aare::xy Struct Reference

```
#include <defs.hpp>
```

Public Member Functions

- bool operator== (const xy &other) const
- bool operator!= (const xy &other) const

Data Fields

- int row
- int col

7.23.1 Member Function Documentation

7.23.1.1 operator"!=()

```
bool aare::xy::operator!= (
            const xy & other ) const  [inline]
```

7.23.1.2 operator==()

```
bool aare::xy::operator== (
             const xy & other ) const  [inline]
```

7.23.2 Field Documentation

7.23.2.1 col

```
int aare::xy::col
```

7.23.2.2 row

```
int aare::xy::row
```

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

- core/include/aare/core/defs.hpp

7.24 aare::ZmqFrame Struct Reference

ZmqFrame structure wrapper class to contain a ZmqHeader and a Frame.

```
#include <defs.hpp>
```

Data Fields

- ZmqHeader header
- Frame frame

7.24.1 Detailed Description

ZmqFrame structure wrapper class to contain a ZmqHeader and a Frame.

7.24.2 Field Documentation

7.24.2.1 frame

```
Frame aare::ZmqFrame::frame
```

7.24.2.2 header

```
ZmqHeader aare::ZmqFrame::header
```

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

- network_io/include/aare/network_io/defs.hpp

7.25 aare::ZmqHeader Struct Reference

```
#include <ZmqHeader.hpp>
```

Public Member Functions

- std::string to_string () const
- void from_string (std::string &s)
- bool operator== (const ZmqHeader &other) const

Data Fields

- bool data {true}
- uint32_t jsonversion {0}
- uint32_t dynamicRange {0}
- uint64_t fileIndex {0}
- uint32_t ndetx {0}
- uint32_t ndety {0}
- uint32_t npixelsx {0}
- uint32_t npixelsy {0}
- uint32_t imageSize {0}
- uint64_t acqIndex {0}
- uint64_t frameIndex {0}
- double progress {0}
- std::string fname
- uint64_t frameNumber {0}
- uint32_t expLength {0}
- uint32_t packetNumber {0}
- uint64_t detSpec1 {0}
- uint64_t timestamp {0}
- uint16_t modId {0}
- uint16_t row {0}
- uint16_t column {0}
- uint16_t detSpec2 {0}
- uint32_t detSpec3 {0}
- uint16_t detSpec4 {0}
- uint8_t detType {0}
- uint8_t version {0}
- int flipRows {0}
- uint32_t quad {0}
- bool completeImage {false}
- std::map< std::string, std::string > addJsonHeader
- std::array< int, 4 > rx_roi {}

7.25.1 Detailed Description

zmq header structure (from slsDetectorPackage)

7.25.2 Member Function Documentation

7.25.2.1 from_string()

```
void aare::ZmqHeader::from_string (
            std::string & s )
```

7.25.2.2 operator==()

```
bool aare::ZmqHeader::operator== (
            const ZmqHeader & other ) const
```

7.25.2.3 to_string()

```
std::string aare::ZmqHeader::to_string ( ) const
```

serialize struct to json string

7.25.3 Field Documentation

7.25.3.1 acqIndex

```
uint64_t aare::ZmqHeader::acqIndex {0}
```

frame number from detector

7.25.3.2 addJsonHeader

```
std::map<std::string, std::string> aare::ZmqHeader::addJsonHeader
```

additional json header

7.25.3.3 column

```
uint16_t aare::ZmqHeader::column {0}
```

7.25.3.4 completeImage

```
bool aare::ZmqHeader::completeImage {false}
```

true if complete image, else missing packets

7.25.3.5 data

```
bool aare::ZmqHeader::data {true}
```

true if incoming data, false if end of acquisition

7.25.3.6 detSpec1

```
uint64_t aare::ZmqHeader::detSpec1 {0}
```

7.25.3.7 detSpec2

```
uint16_t aare::ZmqHeader::detSpec2 {0}
```

7.25.3.8 detSpec3

```
uint32_t aare::ZmqHeader::detSpec3 {0}
```

7.25.3.9 detSpec4

```
uint16_t aare::ZmqHeader::detSpec4 {0}
```

7.25.3.10 detType

```
uint8_t aare::ZmqHeader::detType {0}
```

7.25.3.11 dynamicRange

```
uint32_t aare::ZmqHeader::dynamicRange {0}
```

7.25.3.12 expLength

```
uint32_t aare::ZmqHeader::expLength {0}
```

7.25.3.13 fileIndex

```
uint64_t aare::ZmqHeader::fileIndex {0}
```

7.25.3.14 flipRows

```
int aare::ZmqHeader::flipRows {0}
```

if rows of image should be flipped

7.25.3.15 fname

```
std::string aare::ZmqHeader::fname
```

file name prefix

7.25.3.16 frameIndex

```
uint64_t aare::ZmqHeader::frameIndex {0}
```

frame index (starting at 0 for each acquisition)

7.25.3.17 frameNumber

```
uint64_t aare::ZmqHeader::frameNumber {0}
```

header from detector

7.25.3.18 imageSize

```
uint32_t aare::ZmqHeader::imageSize {0}
```

number of bytes for an image in this socket

7.25.3.19 jsonversion

```
uint32_t aare::ZmqHeader::jsonversion {0}
```

7.25.3.20 modId

```
uint16_t aare::ZmqHeader::modId {0}
```

7.25.3.21 ndetx

```
uint32_t aare::ZmqHeader::ndetx {0}
```

number of detectors/port in x axis

7.25.3.22 ndety

```
uint32_t aare::ZmqHeader::ndety {0}
```

number of detectors/port in y axis

7.25.3.23 npixelsx

```
uint32_t aare::ZmqHeader::npixelsx {0}
```

number of pixels/channels in x axis for this zmq socket

7.25.3.24 npixelsy

```
uint32_t aare::ZmqHeader::npixelsy {0}
```

number of pixels/channels in y axis for this zmq socket

7.25.3.25 packetNumber

```
uint32_t aare::ZmqHeader::packetNumber {0}
```

7.25.3.26 progress

```
double aare::ZmqHeader::progress {0}
```

progress in percentage

7.25.3.27 quad

```
uint32_t aare::ZmqHeader::quad {0}
```

quad type (eiger hardware specific)

7.25.3.28 row

```
uint16_t aare::ZmqHeader::row {0}
```

7.25.3.29 rx_roi

```
std::array<int, 4> aare::ZmqHeader::rx_roi {}
```

(xmin, xmax, ymin, ymax) roi only in files written

7.25.3.30 timestamp

```
uint64_t aare::ZmqHeader::timestamp {0}
```

7.25.3.31 version

```
uint8_t aare::ZmqHeader::version {0}
```

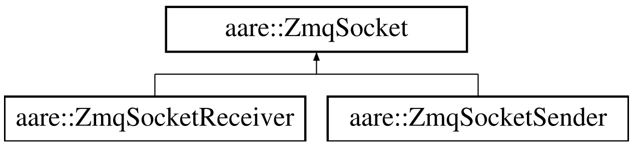
The documentation for this struct was generated from the following files:

- network_io/include/aare/network_io/ZmqHeader.hpp
- network_io/src/ZmqHeader.cpp

7.26 aare::ZmqSocket Class Reference

```
#include <ZmqSocket.hpp>
```

Inheritance diagram for aare::ZmqSocket:



Public Member Functions

- ZmqSocket ()=default
- ∼ZmqSocket ()
- ZmqSocket (const ZmqSocket &)=delete
- ZmqSocket operator= (const ZmqSocket &)=delete
- ZmqSocket (ZmqSocket &&)=delete
- void disconnect ()
- void set_zmq_hwm (int hwm)
- void set_timeout_ms (int n)
- void set_potential_frame_size (size_t size)

Protected Attributes

- void ∗ m_context {nullptr}
- void ∗ m_socket {nullptr}
- std::string m_endpoint
- int m_zmq_hwm {1000}
- int m_timeout_ms {1000}
- size_t m_potential_frame_size {1024 ∗ 1024}
- char ∗ m_header_buffer = new char[m_max_header_size]

Static Protected Attributes

- static constexpr size_t m_max_header_size = 1024

7.26.1 Constructor & Destructor Documentation

7.26.1.1 ZmqSocket() [1/3]

```
aare::ZmqSocket::ZmqSocket ( ) [default]
```

7.26.1.2 ∼ZmqSocket()

```
aare::ZmqSocket::∼ZmqSocket ( )
```

7.26.1.3 ZmqSocket() [2/3]

```
aare::ZmqSocket::ZmqSocket (
             const ZmqSocket & ) [delete]
```

7.26.1.4 ZmqSocket() [3/3]

```
aare::ZmqSocket::ZmqSocket (
             ZmqSocket && ) [delete]
```

7.26.2 Member Function Documentation

7.26.2.1 disconnect()

```
void aare::ZmqSocket::disconnect ( )
```

7.26.2.2 operator=()

```
ZmqSocket aare::ZmqSocket::operator= (
             const ZmqSocket & ) [delete]
```

7.26.2.3 set_potential_frame_size()

```
void aare::ZmqSocket::set_potential_frame_size (
             size_t size )
```

7.26.2.4 set_timeout_ms()

```
void aare::ZmqSocket::set_timeout_ms (
             int n )
```

7.26.2.5 set_zmq_hwm()

```
void aare::ZmqSocket::set_zmq_hwm (
             int hwm )
```

7.26.3 Field Documentation

7.26.3.1 m_context

```
void* aare::ZmqSocket::m_context {nullptr}  [protected]
```

7.26.3.2 m_endpoint

```
std::string aare::ZmqSocket::m_endpoint  [protected]
```

7.26.3.3 m_header_buffer

```
char* aare::ZmqSocket::m_header_buffer = new char[m_max_header_size]  [protected]
```

7.26.3.4 m_max_header_size

```
constexpr size_t aare::ZmqSocket::m_max_header_size = 1024  [static], [constexpr], [protected]
```

7.26.3.5 m_potential_frame_size

```
size_t aare::ZmqSocket::m_potential_frame_size {1024 * 1024}  [protected]
```

7.26.3.6 m_socket

```
void* aare::ZmqSocket::m_socket {nullptr}  [protected]
```

7.26.3.7 m_timeout_ms

```
int aare::ZmqSocket::m_timeout_ms {1000}  [protected]
```

7.26.3.8 m_zmq_hwm

```
int aare::ZmqSocket::m_zmq_hwm {1000}  [protected]
```

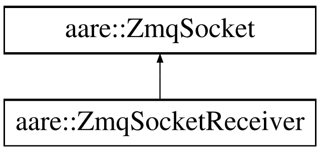
The documentation for this class was generated from the following files:

- network_io/include/aare/network_io/ZmqSocket.hpp
- network_io/src/ZmqSocket.cpp

7.27 aare::ZmqSocketReceiver Class Reference

```
#include <ZmqSocketReceiver.hpp>
```

Inheritance diagram for aare::ZmqSocketReceiver:



Public Member Functions

- ZmqSocketReceiver (const std::string &endpoint)

  Construct a new ZmqSocketReceiver object.
- void connect ()

  Connect to the given endpoint subscribe to a Zmq published.
- std::vector< ZmqFrame > receive_n ()
- void disconnect ()
- void set_zmq_hwm (int hwm)
- void set_timeout_ms (int n)
- void set_potential_frame_size (size_t size)

Protected Attributes

- void ∗ m_context {nullptr}
- void ∗ m_socket {nullptr}
- std::string m_endpoint
- int m_zmq_hwm {1000}
- int m_timeout_ms {1000}
- size_t m_potential_frame_size {1024 ∗ 1024}
- char ∗ m_header_buffer = new char[m_max_header_size]

Static Protected Attributes

- static constexpr size_t m_max_header_size = 1024

Private Member Functions

- int receive_data (std::byte ∗data, size_t size)

  receive data following a ZmqHeader
- ZmqFrame receive_zmqframe ()
- ZmqHeader receive_header ()

  receive a ZmqHeader

7.27.1 Constructor & Destructor Documentation

7.27.1.1 ZmqSocketReceiver()

```
aare::ZmqSocketReceiver::ZmqSocketReceiver (
             const std::string & endpoint )
```

Construct a new ZmqSocketReceiver object.

7.27.2 Member Function Documentation

7.27.2.1 connect()

```
void aare::ZmqSocketReceiver::connect ( )
```

Connect to the given endpoint subscribe to a Zmq published.

7.27.2.2 disconnect()

```
void aare::ZmqSocket::disconnect ( )  [inherited]
```

7.27.2.3 receive_data()

```
int aare::ZmqSocketReceiver::receive_data (
             std::byte * data,
             size_t size )  [private]
```

receive data following a ZmqHeader

Parameters

data	pointer to data
size	size of data

Returns

ZmqHeader

7.27.2.4 receive_header()

```
ZmqHeader aare::ZmqSocketReceiver::receive_header ( )  [private]
```

receive a ZmqHeader

Returns

ZmqHeader

7.27.2.5 receive_n()

```
std::vector< ZmqFrame > aare::ZmqSocketReceiver::receive_n ( )
```

7.27.2.6 receive_zmqframe()

```
ZmqFrame aare::ZmqSocketReceiver::receive_zmqframe ( )  [private]
```

7.27.2.7 set_potential_frame_size()

```
void aare::ZmqSocket::set_potential_frame_size (
             size_t size )  [inherited]
```

7.27.2.8 set_timeout_ms()

```
void aare::ZmqSocket::set_timeout_ms (
             int n )  [inherited]
```

7.27.2.9 set_zmq_hwm()

```
void aare::ZmqSocket::set_zmq_hwm (
             int hwm )  [inherited]
```

7.27.3 Field Documentation

7.27.3.1 m_context

```
void* aare::ZmqSocket::m_context {nullptr}  [protected], [inherited]
```

7.27.3.2 m_endpoint

```
std::string aare::ZmqSocket::m_endpoint  [protected], [inherited]
```

7.27.3.3 m_header_buffer

```
char* aare::ZmqSocket::m_header_buffer = new char[m_max_header_size]  [protected], [inherited]
```

7.27.3.4 m_max_header_size

```
constexpr size_t aare::ZmqSocket::m_max_header_size = 1024  [static], [constexpr], [protected],
[inherited]
```

7.27.3.5 m_potential_frame_size

```
size_t aare::ZmqSocket::m_potential_frame_size {1024 * 1024}  [protected], [inherited]
```

7.27.3.6 m_socket

```
void* aare::ZmqSocket::m_socket {nullptr}  [protected], [inherited]
```

7.27.3.7 m_timeout_ms

```
int aare::ZmqSocket::m_timeout_ms {1000}  [protected], [inherited]
```

7.27.3.8 m_zmq_hwm

```
int aare::ZmqSocket::m_zmq_hwm {1000}  [protected], [inherited]
```

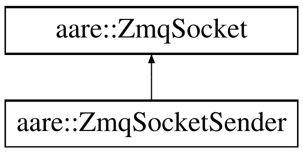
The documentation for this class was generated from the following files:

- network_io/include/aare/network_io/ZmqSocketReceiver.hpp
- network_io/src/ZmqSocketReceiver.cpp

7.28 aare::ZmqSocketSender Class Reference

```
#include <ZmqSocketSender.hpp>
```

Inheritance diagram for aare::ZmqSocketSender:



Public Member Functions

- ZmqSocketSender (const std::string &endpoint)
- void bind ()
- size_t send (const ZmqHeader &header, const std::byte *data, size_t size)
- size_t send (const ZmqFrame &zmq_frame)
- size_t send (const std::vector< ZmqFrame > &zmq_frames)
- void disconnect ()
- void set_zmq_hwm (int hwm)
- void set_timeout_ms (int n)
- void set_potential_frame_size (size_t size)

Protected Attributes

- void ∗ m_context {nullptr}
- void ∗ m_socket {nullptr}
- std::string m_endpoint
- int m_zmq_hwm {1000}
- int m_timeout_ms {1000}
- size_t m_potential_frame_size {1024 ∗ 1024}
- char ∗ m_header_buffer = new char[m_max_header_size]

Static Protected Attributes

- static constexpr size_t m_max_header_size = 1024

7.28.1 Constructor & Destructor Documentation

7.28.1.1 ZmqSocketSender()

```
aare::ZmqSocketSender::ZmqSocketSender (
             const std::string & endpoint )
```

Constructor

Parameters

endpoint	ZMQ endpoint

7.28.2 Member Function Documentation

7.28.2.1 bind()

```
void aare::ZmqSocketSender::bind ( )
```

bind to the given port

7.28.2.2 disconnect()

```
void aare::ZmqSocket::disconnect ( )  [inherited]
```

7.28.2.3 send() [1/3]

```
size_t aare::ZmqSocketSender::send (
             const std::vector< ZmqFrame > & zmq_frames )
```

Send a vector of headers and frames

Parameters

zmq_frames	vector of ZmqFrame

Returns

number of bytes sent

7.28.2.4 send() [2/3]

```
size_t aare::ZmqSocketSender::send (
            const ZmqFrame & zmq_frame )
```

Send a frame with a header

Parameters

ZmqFrame	that contains a header and a frame

Returns

number of bytes sent

7.28.2.5 send() [3/3]

```
size_t aare::ZmqSocketSender::send (
            const ZmqHeader & header,
            const std::byte * data,
            size_t size )
```

send a header and data

Parameters

header	
data	pointer to data
size	size of data

Returns

number of bytes sent

7.28.2.6 set_potential_frame_size()

```
void aare::ZmqSocket::set_potential_frame_size (
            size_t size )  [inherited]
```

7.28.2.7 set_timeout_ms()

```
void aare::ZmqSocket::set_timeout_ms (
             int n )  [inherited]
```

7.28.2.8 set_zmq_hwm()

```
void aare::ZmqSocket::set_zmq_hwm (
             int hwm )  [inherited]
```

7.28.3 Field Documentation

7.28.3.1 m_context

```
void* aare::ZmqSocket::m_context {nullptr}  [protected], [inherited]
```

7.28.3.2 m_endpoint

```
std::string aare::ZmqSocket::m_endpoint  [protected], [inherited]
```

7.28.3.3 m_header_buffer

```
char* aare::ZmqSocket::m_header_buffer = new char[m_max_header_size]  [protected], [inherited]
```

7.28.3.4 m_max_header_size

```
constexpr size_t aare::ZmqSocket::m_max_header_size = 1024  [static], [constexpr], [protected],
[inherited]
```

7.28.3.5 m_potential_frame_size

```
size_t aare::ZmqSocket::m_potential_frame_size {1024 * 1024}  [protected], [inherited]
```

7.28.3.6 m_socket

```
void* aare::ZmqSocket::m_socket {nullptr}  [protected], [inherited]
```

7.28.3.7 m_timeout_ms

```
int aare::ZmqSocket::m_timeout_ms {1000}  [protected], [inherited]
```

7.28.3.8 m_zmq_hwm

```
int aare::ZmqSocket::m_zmq_hwm {1000}  [protected], [inherited]
```

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

- network_io/include/aare/network_io/ZmqSocketSender.hpp
- network_io/src/ZmqSocketSender.cpp

Chapter 8

File Documentation

8.1 core/include/aare/core/CircularFifo.hpp File Reference

```
#include <chrono>
#include <fmt/color.h>
#include <fmt/format.h>
#include <memory>
#include <thread>
#include "aare/core/ProducerConsumerQueue.hpp"
```

Data Structures

- class aare::CircularFifo< ItemType >

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.2 CircularFifo.hpp

Go to the documentation of this file.

```
#pragma once

#include <chrono>
#include <fmt/color.h>
#include <fmt/format.h>
#include <memory>
#include <thread>

#include "aare/core/ProducerConsumerQueue.hpp"

namespace aare {

template <class ItemType> class CircularFifo {
    uint32_t fifo_size;
    folly::ProducerConsumerQueue<ItemType> free_slots;
    folly::ProducerConsumerQueue<ItemType> filled_slots;

```

```
  public:
    CircularFifo() : CircularFifo(100){};
    CircularFifo(uint32_t size) : fifo_size(size), free_slots(size + 1), filled_slots(size + 1) {

        // TODO! how do we deal with alignment for writing? alignas???
        // Do we give the user a chance to provide memory locations?
        // Templated allocator?
        for (size_t i = 0; i < fifo_size; ++i) {
            free_slots.write(ItemType{});
        }
    }

    bool next() {
        // TODO! avoid default constructing ItemType
        ItemType it;
        if (!filled_slots.read(it))
            return false;
        if (!free_slots.write(std::move(it)))
            return false;
        return true;
    }

    ~CircularFifo() {}

    using value_type = ItemType;

    auto numFilledSlots() const noexcept { return filled_slots.sizeGuess(); }
    auto numFreeSlots() const noexcept { return free_slots.sizeGuess(); }
    auto isFull() const noexcept { return filled_slots.isFull(); }

    ItemType pop_free() {
        ItemType v;
        while (!free_slots.read(v))
            ;
        return std::move(v);
        // return v;
    }

    bool try_pop_free(ItemType &v) { return free_slots.read(v); }

    ItemType pop_value(std::chrono::nanoseconds wait, std::atomic<bool> &stopped) {
        ItemType v;
        while (!filled_slots.read(v) && !stopped) {
            std::this_thread::sleep_for(wait);
        }
        return std::move(v);
    }

    ItemType pop_value() {
        ItemType v;
        while (!filled_slots.read(v))
            ;
        return std::move(v);
    }

    ItemType *frontPtr() { return filled_slots.frontPtr(); }

    // TODO! Add function to move item from filled to free to be used
    // with the frontPtr function

    template <class... Args> void push_value(Args &&...recordArgs) {
        while (!filled_slots.write(std::forward<Args>(recordArgs)...))
            ;
    }

    template <class... Args> bool try_push_value(Args &&...recordArgs) {
        return filled_slots.write(std::forward<Args>(recordArgs)...);
    }

    template <class... Args> void push_free(Args &&...recordArgs) {
        while (!free_slots.write(std::forward<Args>(recordArgs)...))
            ;
    }

    template <class... Args> bool try_push_free(Args &&...recordArgs) {
        return free_slots.write(std::forward<Args>(recordArgs)...);
    }
};

} // namespace aare
```

8.3 core/include/aare/core/defs.hpp File Reference

```
#include <array>
#include <stdexcept>
#include <cstdint>
#include <string>
#include <string_view>
#include <variant>
#include <vector>
```

Data Structures

- struct aare::sls_detector_header
- struct aare::xy
- struct aare::RawFileConfig

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Typedefs

- using aare::dynamic_shape = std::vector< ssize_t >
- using aare::DataTypeVariants = std::variant< uint16_t, uint32_t >

Enumerations

- enum class aare::DetectorType {
  aare::Jungfrau , aare::Eiger , aare::Mythen3 , aare::Moench ,
  aare::ChipTestBoard }
- enum class aare::TimingMode { aare::Auto , aare::Trigger }

Functions

- template<class T >
  T aare::StringTo (std::string sv)
- template<class T >
  std::string aare::toString (T sv)
- template<> DetectorType aare::StringTo (std::string)
- template<> std::string aare::toString (DetectorType type)
- template<> TimingMode aare::StringTo (std::string)

8.4 defs.hpp

Go to the documentation of this file.

```
#pragma once

#include <array>
#include <stdexcept>

#include <cstdint>
#include <string>
#include <string_view>
#include <variant>
#include <vector>

namespace aare {

struct sls_detector_header {
    uint64_t frameNumber;
    uint32_t expLength;
    uint32_t packetNumber;
    uint64_t bunchId;
    uint64_t timestamp;
    uint16_t modId;
    uint16_t row;
    uint16_t column;
    uint16_t reserved;
    uint32_t debug;
    uint16_t roundRNumber;
    uint8_t detType;
    uint8_t version;
    uint8_t packetMask[64];
};

struct xy {
    int row;
    int col;
    bool operator==(const xy &other) const { return row == other.row && col == other.col; }
    bool operator!=(const xy &other) const { return !(*this == other); }
};

// using image_shape = std::array<ssize_t, 2>;
using dynamic_shape = std::vector<ssize_t>;

enum class DetectorType { Jungfrau, Eiger, Mythen3, Moench, ChipTestBoard };

enum class TimingMode { Auto, Trigger };

template <class T> T StringTo(std::string sv) { return T(sv); }

template <class T> std::string toString(T sv) { return T(sv); }

template <> DetectorType StringTo(std::string);
template <> std::string toString(DetectorType type);

template <> TimingMode StringTo(std::string);

using DataTypeVariants = std::variant<uint16_t, uint32_t>;

struct RawFileConfig {
    int module_gap_row{};
    int module_gap_col{};

    bool operator==(const RawFileConfig &other) const {
        if (module_gap_col != other.module_gap_col)
            return false;
        if (module_gap_row != other.module_gap_row)
            return false;
        return true;
    }
};

} // namespace aare
```

8.5 network_io/include/aare/network_io/defs.hpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include <stdexcept>
#include <string>
```

Data Structures

- struct aare::ZmqFrame

  ZmqFrame structure wrapper class to contain a ZmqHeader and a Frame.
- class aare::network_io::NetworkError

  NetworkError exception class.

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare::network_io

8.6 defs.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"

#include <stdexcept>
#include <string>

namespace aare {
struct ZmqFrame {
    ZmqHeader header;
    Frame frame;
};

namespace network_io {
class NetworkError : public std::runtime_error {
  private:
    const char *m_msg;

  public:
    NetworkError(const char *msg) : std::runtime_error(msg), m_msg(msg) {}
    NetworkError(const std::string msg) : std::runtime_error(msg) { m_msg = strdup(msg.c_str()); }
    virtual const char *what() const noexcept override { return m_msg; }
};

} // namespace network_io

} // namespace aare
```

8.7 core/include/aare/core/DType.hpp File Reference

```
#include <cstdint>
#include <string>
#include <typeinfo>
```

Data Structures

- class aare::DType

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Enumerations

- enum class aare::endian { aare::little = __ORDER_LITTLE_ENDIAN__ , aare::big = __ORDER_BIG_↩ ENDIAN__ , aare::native = __BYTE_ORDER__ }

8.8 DType.hpp

Go to the documentation of this file.

```
#pragma once
#include <cstdint>
#include <string>
#include <typeinfo>

namespace aare {

enum class endian {
#ifdef _WIN32
    little = 0,
    big = 1,
    native = little
#else
    little = __ORDER_LITTLE_ENDIAN__,
    big = __ORDER_BIG_ENDIAN__,
    native = __BYTE_ORDER__
#endif
};

class DType {
    // TODO! support for non native endianess?
    static_assert(sizeof(long) == sizeof(int64_t), "long should be 64bits");

  public:
    enum TypeIndex { INT8, UINT8, INT16, UINT16, INT32, UINT32, INT64, UINT64, FLOAT, DOUBLE, ERROR };

    uint8_t bitdepth() const;

    explicit DType(const std::type_info &t);
    explicit DType(std::string_view sv);

    // not explicit to allow conversions form enum to DType
    DType(DType::TypeIndex ti);

    bool operator==(const DType &other) const noexcept;
    bool operator!=(const DType &other) const noexcept;
    bool operator==(const std::type_info &t) const;
    bool operator!=(const std::type_info &t) const;

    // bool operator==(DType::TypeIndex ti) const;
    // bool operator!=(DType::TypeIndex ti) const;
    std::string str() const;

  private:
    TypeIndex m_type{TypeIndex::ERROR};
};

} // namespace aare
```

8.9 core/include/aare/core/Frame.hpp File Reference

```
#include "aare/core/NDArray.hpp"
#include "aare/core/defs.hpp"
#include <cstddef>
```

```
#include <cstdint>
#include <memory>
#include <sys/types.h>
#include <vector>
```

Data Structures

- class aare::Frame

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.10 Frame.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/NDArray.hpp"
#include "aare/core/defs.hpp"
#include <cstddef>
#include <cstdint>
#include <memory>
#include <sys/types.h>
#include <vector>

namespace aare {

class Frame {
    ssize_t m_rows;
    ssize_t m_cols;
    ssize_t m_bitdepth;
    std::byte *m_data;

  public:
    Frame(ssize_t rows, ssize_t cols, ssize_t m_bitdepth);
    Frame(std::byte *fp, ssize_t rows, ssize_t cols, ssize_t m_bitdepth);
    std::byte *get(int row, int col);

    // TODO! can we, or even want to remove the template?
    template <typename T> void set(int row, int col, T data) {
        assert(sizeof(T) == m_bitdepth / 8);
        if (row < 0 || row >= m_rows || col < 0 || col >= m_cols) {
            throw std::out_of_range("Invalid row or column index");
        }
        std::memcpy(m_data + (row * m_cols + col) * (m_bitdepth / 8), &data, m_bitdepth / 8);
    }

    ssize_t rows() const { return m_rows; }
    ssize_t cols() const { return m_cols; }
    ssize_t bitdepth() const { return m_bitdepth; }
    ssize_t size() const { return m_rows * m_cols * m_bitdepth / 8; }
    std::byte *data() const { return m_data; }

    Frame &operator=(Frame &other) {
        m_rows = other.rows();
        m_cols = other.cols();
        m_bitdepth = other.bitdepth();
        m_data = new std::byte[m_rows * m_cols * m_bitdepth / 8];
        std::memcpy(m_data, other.m_data, m_rows * m_cols * m_bitdepth / 8);
        return *this;
    }
    // add move constructor
    Frame(Frame &&other) {
        m_rows = other.rows();
        m_cols = other.cols();
        m_bitdepth = other.bitdepth();
        m_data = other.m_data;
        other.m_data = nullptr;
        other.m_rows = other.m_cols = other.m_bitdepth = 0;
    }
    // copy constructor
    Frame(const Frame &other) {
        m_rows = other.rows();
        m_cols = other.cols();
        m_bitdepth = other.bitdepth();
        m_data = new std::byte[m_rows * m_cols * m_bitdepth / 8];
        std::memcpy(m_data, other.m_data, m_rows * m_cols * m_bitdepth / 8);
    }

    template <typename T> NDView<T> view() {
        std::vector<ssize_t> shape = {m_rows, m_cols};
        T *data = reinterpret_cast<T *>(m_data);
        return NDView<T>(data, shape);
    }

    template <typename T> NDArray<T> image() { return NDArray<T>(this->view<T>()); }

    ~Frame() { delete[] m_data; }
};

} // namespace aare
```

8.11 core/include/aare/core/NDArray.hpp File Reference

```
#include "aare/core/NDView.hpp"
#include <algorithm>
#include <array>
#include <cmath>
#include <fmt/format.h>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <numeric>
```

Data Structures

- class aare::NDArray< T, Ndim >

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Functions

- template<typename T , ssize_t Ndim>
  void aare::save (NDArray< T, Ndim > &img, std::string pathname)
- template<typename T , ssize_t Ndim>
  NDArray< T, Ndim > aare::load (const std::string &pathname, std::array< ssize_t, Ndim > shape)

8.12 NDArray.hpp

Go to the documentation of this file.

```
#pragma once
/*
Container holding image data, or a time series of image data in contigious
memory.


TODO! Add expression templates for operators

*/
#include "aare/core/NDView.hpp"

#include <algorithm>
#include <array>
#include <cmath>
#include <fmt/format.h>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <numeric>

namespace aare {

template <typename T, ssize_t Ndim = 2> class NDArray {
  public:
    NDArray() : shape_(), strides_(c_strides<Ndim>(shape_)), size_(0), data_(nullptr){};

    explicit NDArray(std::array<ssize_t, Ndim> shape)
        : shape_(shape), strides_(c_strides<Ndim>(shape_)),
          size_(std::accumulate(shape_.begin(), shape_.end(), 1, std::multiplies<ssize_t>())),
data_(new T[size_]){};

    NDArray(std::array<ssize_t, Ndim> shape, T value) : NDArray(shape) { this->operator=(value); }

    /* When constructing from a NDView we need to copy the data since
    NDArray expect to own its data, and span is just a view*/
    NDArray(NDView<T, Ndim> span) : NDArray(span.shape()) {
        std::copy(span.begin(), span.end(), begin());
        // fmt::print("NDArray(NDView<T, Ndim> span)\n");
    }

    // Move constructor
    NDArray(NDArray &&other)
        : shape_(other.shape_), strides_(c_strides<Ndim>(shape_)), size_(other.size_), data_(nullptr)
{
        data_ = other.data_;
        other.reset();
        // fmt::print("NDArray(NDArray &&other)\n");
    }

    // Copy constructor
    NDArray(const NDArray &other)
        : shape_(other.shape_), strides_(c_strides<Ndim>(shape_)), size_(other.size_), data_(new
T[size_]) {
        std::copy(other.data_, other.data_ + size_, data_);
        // fmt::print("NDArray(const NDArray &other)\n");
    }

    ~NDArray() { delete[] data_; }

    auto begin() { return data_; }
    auto end() { return data_ + size_; }

    using value_type = T;

    NDArray &operator=(NDArray &&other);      // Move assign
    NDArray &operator=(const NDArray &other); // Copy assign

    NDArray operator+(const NDArray &other);
    NDArray &operator+=(const NDArray &other);
    NDArray operator-(const NDArray &other);
    NDArray &operator-=(const NDArray &other);
    NDArray operator*(const NDArray &other);
    NDArray &operator*=(const NDArray &other);
    NDArray operator/(const NDArray &other);
    // NDArray& operator/=(const NDArray& other);
    template <typename V> NDArray &operator/=(const NDArray<V, Ndim> &other) {
        // check shape
        if (shape_ == other.shape()) {
            for (int i = 0; i < size_; ++i) {
                data_[i] /= other(i);
            }
            return *this;
        } else {
            throw(std::runtime_error("Shape of NDArray must match"));
        }
    }

    NDArray<bool, Ndim> operator>(const NDArray &other);

    bool operator==(const NDArray &other) const;
    bool operator!=(const NDArray &other) const;

    NDArray &operator=(const T &);
    NDArray &operator+=(const T &);
    NDArray operator+(const T &);
    NDArray &operator-=(const T &);
    NDArray operator-(const T &);
    NDArray &operator*=(const T &);
    NDArray operator*(const T &);
    NDArray &operator/=(const T &);
    NDArray operator/(const T &);

    NDArray &operator&=(const T &);

    void sqrt() {
        for (int i = 0; i < size_; ++i) {
            data_[i] = std::sqrt(data_[i]);
        }
    }

    NDArray &operator++(); // pre inc

    template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &>::type operator()(Ix... index) {
        return data_[element_offset(strides_, index...)];
    }

    template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &>::type operator()(Ix... index) const {
        return data_[element_offset(strides_, index...)];
    }

    template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T>::type value(Ix... index) {
        return data_[element_offset(strides_, index...)];
    }

    T &operator()(int i) { return data_[i]; }
    const T &operator()(int i) const { return data_[i]; }

    T *data() { return data_; }
    std::byte *buffer() { return reinterpret_cast<std::byte *>(data_); }
    ssize_t size() const { return size_; }
    size_t total_bytes() const { return size_ * sizeof(T); }
    std::array<ssize_t, Ndim> shape() const noexcept { return shape_; }
    ssize_t shape(ssize_t i) const noexcept { return shape_[i]; }
    std::array<ssize_t, Ndim> strides() const noexcept { return strides_; }
    std::array<ssize_t, Ndim> byte_strides() const noexcept {
        auto byte_strides = strides_;
        for (auto &val : byte_strides)
            val *= sizeof(T);
        return byte_strides;
        // return strides_;
    }

    NDView<T, Ndim> span() const { return NDView<T, Ndim>{data_, shape_}; }

    void Print();
    void Print_all();
    void Print_some();

    void reset() {
        data_ = nullptr;
        size_ = 0;
        std::fill(shape_.begin(), shape_.end(), 0);
        std::fill(strides_.begin(), strides_.end(), 0);
    }

 private:
    std::array<ssize_t, Ndim> shape_;
    std::array<ssize_t, Ndim> strides_;
    ssize_t size_;
    T *data_;
};

// Move assign
template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator=(NDArray<T, Ndim> &&other) {
    if (this != &other) {
        delete[] data_;
        data_ = other.data_;
        shape_ = other.shape_;
        size_ = other.size_;
        strides_ = other.strides_;
        other.reset();
    }
    return *this;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator+(const NDArray &other)
{
    NDArray result(*this);
    result += other;
    return result;
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator+=(const
NDArray<T, Ndim> &other) {
    // check shape
    if (shape_ == other.shape_) {
        for (int i = 0; i < size_; ++i) {
            data_[i] += other.data_[i];
        }
        return *this;
    } else {
        throw(std::runtime_error("Shape of ImageDatas must match"));
    }
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator-(const NDArray &other)
{
    NDArray result{*this};
    result -= other;
    return result;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator-=(const
NDArray<T, Ndim> &other) {
    // check shape
    if (shape_ == other.shape_) {
        for (int i = 0; i < size_; ++i) {
            data_[i] -= other.data_[i];
        }
        return *this;
    } else {
        throw(std::runtime_error("Shape of ImageDatas must match"));
    }
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator*(const NDArray &other)
{
    NDArray result = *this;
    result *= other;
    return result;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator*=(const
NDArray<T, Ndim> &other) {
    // check shape
    if (shape_ == other.shape_) {
        for (int i = 0; i < size_; ++i) {
            data_[i] *= other.data_[i];
        }
        return *this;
    } else {
        throw(std::runtime_error("Shape of ImageDatas must match"));
    }
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator/(const NDArray &other)
{
    NDArray result = *this;
    result /= other;
    return result;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator&=(const T &mask) {
    for (auto it = begin(); it != end(); ++it)
        *it &= mask;
    return *this;
}

// template <typename T, ssize_t Ndim>
// NDArray<T, Ndim>& NDArray<T, Ndim>::operator/=(const NDArray<T, Ndim>&
// other)
// {
//     //check shape
//     if (shape_ == other.shape_) {
```

```
//         for (int i = 0; i < size_; ++i) {
//             data_[i] /= other.data_[i];
//         }
//         return *this;
//     } else {
//         throw(std::runtime_error("Shape of ImageDatas must match"));
//     }
// }

template <typename T, ssize_t Ndim> NDArray<bool, Ndim> NDArray<T, Ndim>::operator>(const NDArray
&other) {
    if (shape_ == other.shape_) {
        NDArray<bool> result{shape_};
        for (int i = 0; i < size_; ++i) {
            result(i) = (data_[i] > other.data_[i]);
        }
        return result;
    } else {
        throw(std::runtime_error("Shape of ImageDatas must match"));
    }
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator=(const
NDArray<T, Ndim> &other) {
    if (this != &other) {
        delete[] data_;
        shape_ = other.shape_;
        strides_ = other.strides_;
        size_ = other.size_;
        data_ = new T[size_];
        std::copy(other.data_, other.data_ + size_, data_);
    }
    return *this;
}

template <typename T, ssize_t Ndim> bool NDArray<T, Ndim>::operator==(const NDArray<T, Ndim> &other)
const {
    if (shape_ != other.shape_)
        return false;

    for (int i = 0; i != size_; ++i)
        if (data_[i] != other.data_[i])
            return false;

    return true;
}

template <typename T, ssize_t Ndim> bool NDArray<T, Ndim>::operator!=(const NDArray<T, Ndim> &other)
const {
    return !((*this) == other);
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator++() {
    for (int i = 0; i < size_; ++i)
        data_[i] += 1;
    return *this;
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator=(const T &value) {
    std::fill_n(data_, size_, value);
    return *this;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator+=(const T &value) {
    for (int i = 0; i < size_; ++i)
        data_[i] += value;
    return *this;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator+(const T &value) {
    NDArray result = *this;
    result += value;
    return result;
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator-=(const T &value) {
    for (int i = 0; i < size_; ++i)
        data_[i] -= value;
    return *this;
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator-(const T &value) {
    NDArray result = *this;
    result -= value;
    return result;
}

template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator/=(const T &value) {
    for (int i = 0; i < size_; ++i)
        data_[i] /= value;
    return *this;
```

```
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator/(const T &value) {
    NDArray result = *this;
    result /= value;
    return result;
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator*=(const T &value) {
    for (int i = 0; i < size_; ++i)
        data_[i] *= value;
    return *this;
}
template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator*(const T &value) {
    NDArray result = *this;
    result *= value;
    return result;
}
template <typename T, ssize_t Ndim> void NDArray<T, Ndim>::Print() {
    if (shape_[0] < 20 && shape_[1] < 20)
        Print_all();
    else
        Print_some();
}
template <typename T, ssize_t Ndim> void NDArray<T, Ndim>::Print_all() {
    for (auto row = 0; row < shape_[0]; ++row) {
        for (auto col = 0; col < shape_[1]; ++col) {
            std::cout « std::setw(3);
            std::cout « (*this)(row, col) « " ";
        }
        std::cout « "\n";
    }
}
template <typename T, ssize_t Ndim> void NDArray<T, Ndim>::Print_some() {
    for (auto row = 0; row < 5; ++row) {
        for (auto col = 0; col < 5; ++col) {
            std::cout « std::setw(7);
            std::cout « (*this)(row, col) « " ";
        }
        std::cout « "\n";
    }
}

template <typename T, ssize_t Ndim> void save(NDArray<T, Ndim> &img, std::string pathname) {
    std::ofstream f;
    f.open(pathname, std::ios::binary);
    f.write(img.buffer(), img.size() * sizeof(T));
    f.close();
}

template <typename T, ssize_t Ndim>
NDArray<T, Ndim> load(const std::string &pathname, std::array<ssize_t, Ndim> shape) {
    NDArray<T, Ndim> img{shape};
    std::ifstream f;
    f.open(pathname, std::ios::binary);
    f.read(img.buffer(), img.size() * sizeof(T));
    f.close();
    return img;
}

} // namespace aare
```

8.13 core/include/aare/core/NDView.hpp File Reference

```
#include <algorithm>
#include <array>
#include <cassert>
#include <cstdint>
#include <numeric>
#include <stdexcept>
#include <vector>
```

Data Structures

- class aare::NDView< T, Ndim >

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Typedefs

- template<ssize_t Ndim>
  using aare::Shape = std::array< ssize_t, Ndim >

Functions

- template<ssize_t Ndim>
  Shape< Ndim > aare::make_shape (const std::vector< size_t > &shape)
- template<ssize_t Dim = 0, typename Strides >
  ssize_t aare::element_offset (const Strides &)
- template<ssize_t Dim = 0, typename Strides , typename... Ix>
  ssize_t aare::element_offset (const Strides &strides, ssize_t i, Ix... index)
- template<ssize_t Ndim>
  std::array< ssize_t, Ndim > aare::c_strides (const std::array< ssize_t, Ndim > &shape)
- template<ssize_t Ndim>
  std::array< ssize_t, Ndim > aare::make_array (const std::vector< ssize_t > &vec)

8.14 NDView.hpp

Go to the documentation of this file.

```
#pragma once
#include <algorithm>
#include <array>
#include <cassert>
#include <cstdint>
#include <numeric>
#include <stdexcept>
#include <vector>

namespace aare {

template <ssize_t Ndim> using Shape = std::array<ssize_t, Ndim>;

// TODO! fix mismatch between signed and unsigned
template <ssize_t Ndim> Shape<Ndim> make_shape(const std::vector<size_t> &shape) {
    if (shape.size() != Ndim)
        throw std::runtime_error("Shape size mismatch");
    Shape<Ndim> arr;
    std::copy_n(shape.begin(), Ndim, arr.begin());
    return arr;
}

template <ssize_t Dim = 0, typename Strides> ssize_t element_offset(const Strides &) { return 0; }

template <ssize_t Dim = 0, typename Strides, typename... Ix>
ssize_t element_offset(const Strides &strides, ssize_t i, Ix... index) {
    return i * strides[Dim] + element_offset<Dim + 1>(strides, index...);
}

template <ssize_t Ndim> std::array<ssize_t, Ndim> c_strides(const std::array<ssize_t, Ndim> &shape) {
    std::array<ssize_t, Ndim> strides;
    std::fill(strides.begin(), strides.end(), 1);
    for (ssize_t i = Ndim - 1; i > 0; --i) {
        strides[i - 1] = strides[i] * shape[i];
    }
    return strides;
}

template <ssize_t Ndim> std::array<ssize_t, Ndim> make_array(const std::vector<ssize_t> &vec) {
    assert(vec.size() == Ndim);
    std::array<ssize_t, Ndim> arr;
    std::copy_n(vec.begin(), Ndim, arr.begin());
    return arr;
}

template <typename T, ssize_t Ndim = 2> class NDView {
  public:
    NDView(){};

    NDView(T *buffer, std::array<ssize_t, Ndim> shape) {
        buffer_ = buffer;
        strides_ = c_strides<Ndim>(shape);
        shape_ = shape;
        size_ = std::accumulate(std::begin(shape), std::end(shape), 1, std::multiplies<ssize_t>());
    }

    NDView(T *buffer, const std::vector<ssize_t> &shape) {
        buffer_ = buffer;
        strides_ = c_strides<Ndim>(make_array<Ndim>(shape));
        shape_ = make_array<Ndim>(shape);
        size_ = std::accumulate(std::begin(shape), std::end(shape), 1, std::multiplies<ssize_t>());
    }

    template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &>::type
operator()(Ix... index) {
        return buffer_[element_offset(strides_, index...)];
    }

    template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &>::type
operator()(Ix... index) const {
        return buffer_[element_offset(strides_, index...)];
    }

    ssize_t size() const { return size_; }

    NDView(const NDView &) = default;
    NDView(NDView &&) = default;

    T *begin() { return buffer_; }
    T *end() { return buffer_ + size_; }
    T &operator()(ssize_t i) { return buffer_[i]; }
    T &operator[](ssize_t i) { return buffer_[i]; }

    bool operator==(const NDView &other) const {
        if (size_ != other.size_)
            return false;
        for (ssize_t i = 0; i != size_; ++i) {
            if (buffer_[i] != other.buffer_[i])
                return false;
        }
        return true;
    }

    NDView &operator+=(const T val) { return elemenwise(val, std::plus<T>()); }
    NDView &operator-=(const T val) { return elemenwise(val, std::minus<T>()); }
    NDView &operator*=(const T val) { return elemenwise(val, std::multiplies<T>()); }
    NDView &operator/=(const T val) { return elemenwise(val, std::divides<T>()); }

    NDView &operator/=(const NDView &other) { return elemenwise(other, std::divides<T>()); }

    NDView &operator=(const T val) {
        for (auto it = begin(); it != end(); ++it)
            *it = val;
        return *this;
    }

    NDView &operator=(const NDView &other) {
        shape_ = other.shape_;
        strides_ = other.strides_;
        size_ = other.size_;
        buffer_ = other.buffer_;
        return *this;
    }
    auto &shape() { return shape_; }
    auto shape(ssize_t i) const { return shape_[i]; }

    T *data() { return buffer_; }

  private:
    T *buffer_{nullptr};
    std::array<ssize_t, Ndim> strides_{};
    std::array<ssize_t, Ndim> shape_{};
    ssize_t size_{};

    template <class BinaryOperation> NDView &elemenwise(T val, BinaryOperation op) {
        for (ssize_t i = 0; i != size_; ++i) {
            buffer_[i] = op(buffer_[i], val);
        }
        return *this;
    }
    template <class BinaryOperation> NDView &elemenwise(const NDView &other, BinaryOperation op) {
        for (ssize_t i = 0; i != size_; ++i) {
            buffer_[i] = op(buffer_[i], other.buffer_[i]);
        }
        return *this;
    }
};

template class NDView<uint16_t, 2>;

} // namespace aare
```

8.15 core/include/aare/core/ProducerConsumerQueue.hpp File Reference

```
#include <atomic>
#include <cassert>
#include <cstdlib>
#include <memory>
#include <stdexcept>
#include <type_traits>
#include <utility>
```

Data Structures

- struct folly::ProducerConsumerQueue< T >

Namespaces

- namespace folly

Variables

- constexpr std::size_t hardware_destructive_interference_size = 128

8.15.1 Variable Documentation

8.15.1.1 hardware_destructive_interference_size

```
constexpr std::size_t hardware_destructive_interference_size = 128  [constexpr]
```

8.16 ProducerConsumerQueue.hpp

Go to the documentation of this file.

```
/*
 * Copyright (c) Meta Platforms, Inc. and affiliates.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

// @author Bo Hu (bhu@fb.com)
// @author Jordan DeLong (delong.j@fb.com)

// Changes made by PSD Detector Group:
// Copied: Line 34 constexpr std::size_t hardware_destructive_interference_size = 128; from
folly/lang/Align.h
// Changed extension to .hpp

#pragma once

#include <atomic>
#include <cassert>
#include <cstdlib>
#include <memory>
#include <stdexcept>
#include <type_traits>
#include <utility>

constexpr std::size_t hardware_destructive_interference_size = 128;
namespace folly {

/*
 * ProducerConsumerQueue is a one producer and one consumer queue
 * without locks.
 */
template <class T> struct ProducerConsumerQueue {
    typedef T value_type;

    ProducerConsumerQueue(const ProducerConsumerQueue &) = delete;
    ProducerConsumerQueue &operator=(const ProducerConsumerQueue &) = delete;

    // size must be >= 2.
    //
    // Also, note that the number of usable slots in the queue at any
    // given time is actually (size-1), so if you start with an empty queue,
    // isFull() will return true after size-1 insertions.
    explicit ProducerConsumerQueue(uint32_t size)
        : size_(size), records_(static_cast<T *>(std::malloc(sizeof(T) * size))), readIndex_(0),
writeIndex_(0) {
        assert(size >= 2);
        if (!records_) {
            throw std::bad_alloc();
        }
    }

    ~ProducerConsumerQueue() {
        // We need to destruct anything that may still exist in our queue.
        // (No real synchronization needed at destructor time: only one
        // thread can be doing this.)
        if (!std::is_trivially_destructible<T>::value) {
            size_t readIndex = readIndex_;
            size_t endIndex = writeIndex_;
            while (readIndex != endIndex) {
                records_[readIndex].~T();
                if (++readIndex == size_) {
                    readIndex = 0;
                }
            }
        }

        std::free(records_);
    }

    template <class... Args> bool write(Args &&...recordArgs) {
        auto const currentWrite = writeIndex_.load(std::memory_order_relaxed);
        auto nextRecord = currentWrite + 1;
        if (nextRecord == size_) {
            nextRecord = 0;
        }
        if (nextRecord != readIndex_.load(std::memory_order_acquire)) {
            new (&records_[currentWrite]) T(std::forward<Args>(recordArgs)...);
            writeIndex_.store(nextRecord, std::memory_order_release);
            return true;
        }

        // queue is full
        return false;
    }

    // move (or copy) the value at the front of the queue to given variable
    bool read(T &record) {
        auto const currentRead = readIndex_.load(std::memory_order_relaxed);
        if (currentRead == writeIndex_.load(std::memory_order_acquire)) {
            // queue is empty
            return false;
        }

        auto nextRecord = currentRead + 1;
        if (nextRecord == size_) {
            nextRecord = 0;
        }
        record = std::move(records_[currentRead]);
        records_[currentRead].~T();
        readIndex_.store(nextRecord, std::memory_order_release);
        return true;
    }

    // pointer to the value at the front of the queue (for use in-place) or
    // nullptr if empty.
    T *frontPtr() {
        auto const currentRead = readIndex_.load(std::memory_order_relaxed);
        if (currentRead == writeIndex_.load(std::memory_order_acquire)) {
            // queue is empty
            return nullptr;
        }
        return &records_[currentRead];
    }

    // queue must not be empty
    void popFront() {
        auto const currentRead = readIndex_.load(std::memory_order_relaxed);
        assert(currentRead != writeIndex_.load(std::memory_order_acquire));

        auto nextRecord = currentRead + 1;
        if (nextRecord == size_) {
            nextRecord = 0;
        }
        records_[currentRead].~T();
        readIndex_.store(nextRecord, std::memory_order_release);
    }

    bool isEmpty() const {
        return readIndex_.load(std::memory_order_acquire) ==
writeIndex_.load(std::memory_order_acquire);
    }

    bool isFull() const {
        auto nextRecord = writeIndex_.load(std::memory_order_acquire) + 1;
        if (nextRecord == size_) {
            nextRecord = 0;
        }
        if (nextRecord != readIndex_.load(std::memory_order_acquire)) {
            return false;
        }
        // queue is full
        return true;
    }

    // * If called by consumer, then true size may be more (because producer may
    //   be adding items concurrently).
    // * If called by producer, then true size may be less (because consumer may
    //   be removing items concurrently).
    // * It is undefined to call this from any other thread.
    size_t sizeGuess() const {
        int ret = writeIndex_.load(std::memory_order_acquire) -
readIndex_.load(std::memory_order_acquire);
        if (ret < 0) {
            ret += size_;
        }
        return ret;
    }

    // maximum number of items in the queue.
    size_t capacity() const { return size_ - 1; }

  private:
    using AtomicIndex = std::atomic<unsigned int>;

    char pad0_[hardware_destructive_interference_size];
    const uint32_t size_;
    T *const records_;

    alignas(hardware_destructive_interference_size) AtomicIndex readIndex_;
    alignas(hardware_destructive_interference_size) AtomicIndex writeIndex_;

    char pad1_[hardware_destructive_interference_size - sizeof(AtomicIndex)];
};

} // namespace folly
```

8.17 core/include/aare/core/VariableSizeClusterFinder.hpp File Reference

```
#include <algorithm>
#include <map>
#include <unordered_map>
#include <vector>
#include "aare/core/NDArray.hpp"
```

Data Structures

- class aare::ClusterFinder< T >
- struct aare::ClusterFinder< T >::Hit

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Variables

- const int MAX_CLUSTER_SIZE = 200

8.17.1 Variable Documentation

8.17.1.1 MAX_CLUSTER_SIZE

```
const int MAX_CLUSTER_SIZE = 200
```

8.18 VariableSizeClusterFinder.hpp

Go to the documentation of this file.

```cpp
#pragma once

#include <algorithm>
#include <map>
#include <unordered_map>
#include <vector>

#include "aare/core/NDArray.hpp"

const int MAX_CLUSTER_SIZE = 200;
namespace aare {

template <typename T> class ClusterFinder {
  public:
    struct Hit {
        int16_t size{};
        int16_t row{};
        int16_t col{};
        uint16_t reserved{}; // for alignment
        T energy{};
        T max{};

        // std::vector<int16_t> rows{};
        // std::vector<int16_t> cols{};
        int16_t rows[MAX_CLUSTER_SIZE] = {0};
        int16_t cols[MAX_CLUSTER_SIZE] = {0};
        double enes[MAX_CLUSTER_SIZE] = {0};
    };

  private:
    const std::array<ssize_t, 2> shape_;
    NDView<T, 2> original_;
    NDArray<int, 2> labeled_;
    NDArray<int, 2> peripheral_labeled_;
    NDArray<bool, 2> binary_; // over threshold flag
    T threshold_;
    NDView<T, 2> noiseMap;
    bool use_noise_map = false;
    int peripheralThresholdFactor_ = 5;
    int current_label;
    const std::array<int, 4> di{{0, -1, -1, -1}};              // row ### 8-neighbour by scaning from left to right
    const std::array<int, 4> dj{{-1, -1, 0, 1}};               // col ### 8-neighbour by scaning from top to bottom
    const std::array<int, 8> di_{{0, 0, -1, 1, -1, 1, -1, 1}}; // row
    const std::array<int, 8> dj_{{-1, 1, 0, 0, 1, -1, -1, 1}}; // col
    std::map<int, int> child;                                  // heirachy: key: child; val: parent
    std::unordered_map<int, Hit> h_size;
    std::vector<Hit> hits;
    // std::vector<std::vector<int16_t» row
    int check_neighbours(int i, int j);

  public:
    ClusterFinder(image_shape shape, T threshold)
        : shape_(shape), labeled_(shape, 0), peripheral_labeled_(shape, 0), binary_(shape), threshold_(threshold) {
        hits.reserve(2000);
    }

    NDArray<int, 2> labeled() { return labeled_; }

    void set_noiseMap(NDView<T, 2> noise_map) {
        noiseMap = noise_map;
        use_noise_map = true;
    }
    void set_peripheralThresholdFactor(int factor) { peripheralThresholdFactor_ = factor; }
    void find_clusters(NDView<T, 2> img);
    void find_clusters_X(NDView<T, 2> img);
    void rec_FillHit(int clusterIndex, int i, int j);
    void single_pass(NDView<T, 2> img);
    void first_pass();
    void second_pass();
    void store_clusters();

    std::vector<Hit> steal_hits() {
        std::vector<Hit> tmp;
        std::swap(tmp, hits);
        return tmp;
    };
    void clear_hits() { hits.clear(); };

    void print_connections() {
        fmt::print("Connections:\n");
        for (auto it = child.begin(); it != child.end(); ++it) {
            fmt::print("{} -> {}\n", it->first, it->second);
        }
    }
    size_t total_clusters() const {
        // TODO! fix for stealing
        return hits.size();
    }

  private:
    void add_link(int from, int to) {
        // we want to add key from -> value to
        // fmt::print("add_link({},{})\n", from, to);
        auto it = child.find(from);
        if (it == child.end()) {
            child[from] = to;
        } else {
            // found need to disambiguate
            if (it->second == to)
                return;
            else {
                if (it->second > to) {
                    // child[from] = to;
                    auto old = it->second;
                    it->second = to;
                    add_link(old, to);
                } else {
                    // found value is smaller than what we want to link
                    add_link(to, it->second);
                }
            }
        }
    }
};
template <typename T> int ClusterFinder<T>::check_neighbours(int i, int j) {
    std::vector<int> neighbour_labels;

    for (int k = 0; k < 4; ++k) {
        const auto row = i + di[k];
        const auto col = j + dj[k];
        if (row >= 0 && col >= 0 && row < shape_[0] && col < shape_[1]) {
            auto tmp = labeled_.value(i + di[k], j + dj[k]);
            if (tmp != 0)
                neighbour_labels.push_back(tmp);
        }
    }

    if (neighbour_labels.size() == 0) {
        return 0;
    } else {

        // need to sort and add to union field
        std::sort(neighbour_labels.rbegin(), neighbour_labels.rend());
        auto first = neighbour_labels.begin();
        auto last = std::unique(first, neighbour_labels.end());
        if (last - first == 1)
            return *neighbour_labels.begin();

        for (auto current = first; current != last - 1; ++current) {
            auto next = current + 1;
            add_link(*current, *next);
        }
        return neighbour_labels.back(); // already sorted
    }
}

template <typename T> void ClusterFinder<T>::find_clusters(NDView<T, 2> img) {
    original_ = img;
    labeled_ = 0;
    peripheral_labeled_ = 0;
    current_label = 0;
    child.clear();
    first_pass();
    // print_connections();
    second_pass();
    store_clusters();
}

template <typename T> void ClusterFinder<T>::find_clusters_X(NDView<T, 2> img) {
    original_ = img;
    int clusterIndex = 0;
    for (int i = 0; i < shape_[0]; ++i) {
        for (int j = 0; j < shape_[1]; ++j) {
            if (use_noise_map)
                threshold_ = 5 * noiseMap(i, j);
            if (original_(i, j) > threshold_) {
                // printf("========== Cluster index: %d\n", clusterIndex);
                rec_FillHit(clusterIndex, i, j);
                clusterIndex++;
            }
        }
    }
    for (const auto &h : h_size)
        hits.push_back(h.second);
    h_size.clear();
}

template <typename T> void ClusterFinder<T>::rec_FillHit(int clusterIndex, int i, int j) {
    // printf("original_(%d, %d)=%f\n", i, j, original_(i,j));
    // printf("h_size[%d].size=%d\n", clusterIndex, h_size[clusterIndex].size);
    if (h_size[clusterIndex].size < MAX_CLUSTER_SIZE) {
        h_size[clusterIndex].rows[h_size[clusterIndex].size] = i;
        h_size[clusterIndex].cols[h_size[clusterIndex].size] = j;
        h_size[clusterIndex].enes[h_size[clusterIndex].size] = original_(i, j);
    }
    h_size[clusterIndex].size += 1;
    h_size[clusterIndex].energy += original_(i, j);
    if (h_size[clusterIndex].max < original_(i, j)) {
        h_size[clusterIndex].row = i;
        h_size[clusterIndex].col = j;
        h_size[clusterIndex].max = original_(i, j);
    }
    original_(i, j) = 0;

    for (int k = 0; k < 8; ++k) { // 8 for 8-neighbour
        const auto row = i + di_[k];
        const auto col = j + dj_[k];
        if (row >= 0 && col >= 0 && row < shape_[0] && col < shape_[1]) {
            if (use_noise_map)
                threshold_ = peripheralThresholdFactor_ * noiseMap(row, col);
            if (original_(row, col) > threshold_) {
                rec_FillHit(clusterIndex, row, col);
            } else {
                // if (h_size[clusterIndex].size < MAX_CLUSTER_SIZE){
                //     h_size[clusterIndex].size += 1;
                //     h_size[clusterIndex].rows[h_size[clusterIndex].size] = row;
                //     h_size[clusterIndex].cols[h_size[clusterIndex].size] = col;
                //     h_size[clusterIndex].enes[h_size[clusterIndex].size] = original_(row, col);
                // }// ? weather to include peripheral pixels
                original_(row, col) = 0; // remove peripheral pixels, to avoid potential influence for
pedestal updating
            }
        }
    }
}

template <typename T> void ClusterFinder<T>::single_pass(NDView<T, 2> img) {
    original_ = img;
    labeled_ = 0;
    current_label = 0;
    child.clear();
    first_pass();
    // print_connections();
    // second_pass();
    // store_clusters();
}

template <typename T> void ClusterFinder<T>::first_pass() {

    for (int i = 0; i < original_.size(); ++i) {
        if (use_noise_map)
            threshold_ = 5 * noiseMap(i);
        binary_(i) = (original_(i) > threshold_);
    }

    for (int i = 0; i < shape_[0]; ++i) {
        for (int j = 0; j < shape_[1]; ++j) {

            // do we have someting to process?
            if (binary_(i, j)) {
                auto tmp = check_neighbours(i, j);
                if (tmp != 0) {
                    labeled_(i, j) = tmp;
                } else {
                    labeled_(i, j) = ++current_label;
                }
            }
        }
    }
}

template <typename T> void ClusterFinder<T>::second_pass() {

    for (ssize_t i = 0; i != labeled_.size(); ++i) {
        auto current_label = labeled_(i);
        if (current_label != 0) {
            auto it = child.find(current_label);
            while (it != child.end()) {
                current_label = it->second;
                it = child.find(current_label);
                // do this once before doing the second pass?
                // all values point to the final one...
            }
            labeled_(i) = current_label;
        }
    }
}

template <typename T> void ClusterFinder<T>::store_clusters() {

    // Accumulate hit information in a map
    // Do we always have monotonic increasing
    // labels? Then vector?
    // here the translation is label -> Hit
    std::unordered_map<int, Hit> h_size;
    for (int i = 0; i < shape_[0]; ++i) {
        for (int j = 0; j < shape_[1]; ++j) {
            if (labeled_(i, j) != 0 or false
                // (i-1 >= 0 and labeled_(i-1, j) != 0) or // another circle of peripheral pixels
                // (j-1 >= 0 and labeled_(i, j-1) != 0) or
                // (i+1 < shape_[0] and labeled_(i+1, j) != 0) or
                // (j+1 < shape_[1] and labeled_(i, j+1) != 0)
            ) {
                Hit &record = h_size[labeled_(i, j)];
                if (record.size < MAX_CLUSTER_SIZE) {
                    record.rows[record.size] = i;
                    record.cols[record.size] = j;
                    record.enes[record.size] = original_(i, j);
                } else {
                    continue;
                }
                record.size += 1;
                record.energy += original_(i, j);

                if (record.max < original_(i, j)) {
                    record.row = i;
                    record.col = j;
                    record.max = original_(i, j);
                }
            }
        }
    }

    for (const auto &h : h_size)
        hits.push_back(h.second);
}

} // namespace aare
```

8.19 core/src/defs.cpp File Reference

```
#include "aare/core/defs.hpp"
```

Namespaces

- namespace aare

    Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Functions

- template<> std::string aare::toString (DetectorType type)
- template<> DetectorType aare::StringTo (std::string name)
- template<> TimingMode aare::StringTo (std::string mode)

8.20 core/src/DType.cpp File Reference

```
#include "aare/core/DType.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/format.h>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.21 core/src/Frame.cpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/utils/logger.hpp"
#include <cassert>
#include <iostream>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.22 core/test/CircularFifo.test.cpp File Reference

```
#include <catch2/catch_all.hpp>
#include "aare/core/CircularFifo.hpp"
```

Data Structures

- struct MoveOnlyInt

Functions

- TEST_CASE ("CircularFifo can be default constructed")
- TEST_CASE ("Newly constructed fifo has the right size")
- TEST_CASE ("Can fit size number of objects")
- TEST_CASE ("Push move only type")
- TEST_CASE ("Push pop")
- TEST_CASE ("Pop free and then push")
- TEST_CASE ("Skip the first value")
- TEST_CASE ("Use in place and move to free")

8.22.1 Function Documentation

8.22.1.1 TEST_CASE() [1/8]

```
TEST_CASE (
             ¨Can fit size number of objects¨  )
```

8.22.1.2 TEST_CASE() [2/8]

```
TEST_CASE (
             ¨CircularFifo can be default constructed¨  )
```

8.22.1.3 TEST_CASE() [3/8]

```
TEST_CASE (
             ¨Newly constructed fifo has the right size¨  )
```

8.22.1.4 TEST_CASE() [4/8]

```
TEST_CASE (
             ¨Pop free and then push¨  )
```

8.22.1.5 TEST_CASE() [5/8]

```
TEST_CASE (
             ¨Push move only type¨  )
```

8.22.1.6 TEST_CASE() [6/8]

```
TEST_CASE (
             ¨Push pop¨  )
```

8.22.1.7 TEST_CASE() [7/8]

```
TEST_CASE (
             ¨Skip the first value¨  )
```

8.22.1.8 TEST_CASE() [8/8]

```
TEST_CASE (
             ¨Use in place and move to free¨  )
```

8.23 core/test/defs.test.cpp File Reference

```
#include "aare/core/defs.hpp"
#include <catch2/catch_test_macros.hpp>
#include <string>
```

Functions

- TEST_CASE ("Enum to string conversion")

8.23.1 Function Documentation

8.23.1.1 TEST_CASE()

```
TEST_CASE (
            "Enum to string conversion" )
```

8.24 core/test/DType.test.cpp File Reference

```
#include "aare/core/DType.hpp"
#include <catch2/catch_test_macros.hpp>
```

Functions

- TEST_CASE ("Construct from typeid")
- TEST_CASE ("Construct from string")
- TEST_CASE ("Construct from string with endianess")
- TEST_CASE ("Convert to string")

8.24.1 Function Documentation

8.24.1.1 TEST_CASE() [1/4]

```
TEST_CASE (
            "Construct from string with endianess" )
```

8.24.1.2 TEST_CASE() [2/4]

```
TEST_CASE (
            "Construct from string" )
```

8.24.1.3 TEST_CASE() [3/4]

```
TEST_CASE (
            ¨Construct from typeid¨  )
```

8.24.1.4 TEST_CASE() [4/4]

```
TEST_CASE (
            ¨Convert to string¨  )
```

8.25 core/test/Frame.test.cpp File Reference

```
#include ¨aare/core/Frame.hpp¨
#include <catch2/catch_test_macros.hpp>
```

Functions

- TEST_CASE (¨Construct a frame¨)
- TEST_CASE (¨Set a value in a 8 bit frame¨)
- TEST_CASE (¨Set a value in a 64 bit frame¨)
- TEST_CASE (¨Move construct a frame¨)

8.25.1 Function Documentation

8.25.1.1 TEST_CASE() [1/4]

```
TEST_CASE (
            ¨Construct a frame¨  )
```

8.25.1.2 TEST_CASE() [2/4]

```
TEST_CASE (
            ¨Move construct a frame¨  )
```

8.25.1.3 TEST_CASE() [3/4]

```
TEST_CASE (
            ¨Set a value in a 64 bit frame¨  )
```

8.25.1.4 TEST_CASE() [4/4]

```
TEST_CASE (
            ¨Set a value in a 8 bit frame¨  )
```

8.26 core/test/NDArray.test.cpp File Reference

```
#include ¨aare/core/NDArray.hpp¨
#include <array>
#include <catch2/catch_test_macros.hpp>
```

Functions

- TEST_CASE (¨Initial size is zero if no size is specified¨)
- TEST_CASE (¨Construct from a DataSpan¨)
- TEST_CASE (¨1D image¨)
- TEST_CASE (¨Accessing a const object¨)
- TEST_CASE (¨Indexing of a 2D image¨)
- TEST_CASE (¨Indexing of a 3D image¨)
- TEST_CASE (¨Divide double by int¨)
- TEST_CASE (¨Elementwise multiplication of 3D image¨)
- TEST_CASE (¨Compare two images¨)
- TEST_CASE (¨Size and shape matches¨)
- TEST_CASE (¨Initial value matches for all elements¨)
- TEST_CASE (¨Data layout of 3D image, fast index last¨)
- TEST_CASE (¨Bitwise and on data¨)
- TEST_CASE (¨Elementwise operatios on images¨)

8.26.1 Function Documentation

8.26.1.1 TEST_CASE() [1/14]

```
TEST_CASE (
              ¨1D image¨  )
```

8.26.1.2 TEST_CASE() [2/14]

```
TEST_CASE (
              ¨Accessing a const object¨  )
```

8.26.1.3 TEST_CASE() [3/14]

```
TEST_CASE (
              ¨Bitwise and on data¨  )
```

8.26.1.4 TEST_CASE() [4/14]

```
TEST_CASE (
              ¨Compare two images¨  )
```

8.26.1.5 TEST_CASE() [5/14]

```
TEST_CASE (
             ¨Construct from a DataSpan¨  )
```

8.26.1.6 TEST_CASE() [6/14]

```
TEST_CASE (
             ¨Data layout of 3D image,
             fast index last¨  )
```

8.26.1.7 TEST_CASE() [7/14]

```
TEST_CASE (
             ¨Divide double by int¨  )
```

8.26.1.8 TEST_CASE() [8/14]

```
TEST_CASE (
             ¨Elementwise multiplication of 3D image¨  )
```

8.26.1.9 TEST_CASE() [9/14]

```
TEST_CASE (
             ¨Elementwise operatios on images¨  )
```

8.26.1.10 TEST_CASE() [10/14]

```
TEST_CASE (
             ¨Indexing of a 2D image¨  )
```

8.26.1.11 TEST_CASE() [11/14]

```
TEST_CASE (
             ¨Indexing of a 3D image¨  )
```

8.26.1.12 TEST_CASE() [12/14]

```
TEST_CASE (
             ¨Initial size is zero if no size is specified¨  )
```

8.26.1.13 TEST_CASE() [13/14]

```
TEST_CASE (
             ¨Initial value matches for all elements¨  )
```

8.26.1.14 TEST_CASE() [14/14]

```
TEST_CASE (
            ¨Size and shape matches¨  )
```

8.27 core/test/NDView.test.cpp File Reference

```
#include ¨aare/core/NDView.hpp¨
#include <catch2/catch_test_macros.hpp>
#include <iostream>
#include <vector>
```

Functions

- TEST_CASE (¨Element reference 1D¨)
- TEST_CASE (¨Element reference 2D¨)
- TEST_CASE (¨Element reference 3D¨)
- TEST_CASE (¨Plus and miuns with single value¨)
- TEST_CASE (¨Multiply and divide with single value¨)
- TEST_CASE (¨elementwise assign¨)
- TEST_CASE (¨iterators¨)
- TEST_CASE (¨shape from vector¨)
- TEST_CASE (¨divide with another span¨)
- TEST_CASE (¨Retrieve shape¨)
- TEST_CASE (¨compare two views¨)

8.27.1 Function Documentation

8.27.1.1 TEST_CASE() [1/11]

```
TEST_CASE (
            ¨compare two views¨  )
```

8.27.1.2 TEST_CASE() [2/11]

```
TEST_CASE (
            ¨divide with another span¨  )
```

8.27.1.3 TEST_CASE() [3/11]

```
TEST_CASE (
            ¨Element reference 1D¨  )
```

8.27.1.4 TEST_CASE() [4/11]

```
TEST_CASE (
             "Element reference 2D"  )
```

8.27.1.5 TEST_CASE() [5/11]

```
TEST_CASE (
             "Element reference 3D"  )
```

8.27.1.6 TEST_CASE() [6/11]

```
TEST_CASE (
             "elementwise assign"  )
```

8.27.1.7 TEST_CASE() [7/11]

```
TEST_CASE (
             "iterators"  )
```

8.27.1.8 TEST_CASE() [8/11]

```
TEST_CASE (
             "Multiply and divide with single value"  )
```

8.27.1.9 TEST_CASE() [9/11]

```
TEST_CASE (
             "Plus and miuns with single value"  )
```

8.27.1.10 TEST_CASE() [10/11]

```
TEST_CASE (
             "Retrieve shape"  )
```

8.27.1.11 TEST_CASE() [11/11]

```
TEST_CASE (
             "shape from vector"  )
```

8.28 core/test/ProducerConsumerQueue.test.cpp File Reference

```
#include "aare/core/ProducerConsumerQueue.hpp"
#include <catch2/catch_all.hpp>
```

Functions

- TEST_CASE ("push pop")
- TEST_CASE ("Cannot push to a full queue")
- TEST_CASE ("Cannot pop from an empty queue")

8.28.1 Function Documentation

8.28.1.1 TEST_CASE() [1/3]

```
TEST_CASE (
             "Cannot pop from an empty queue" )
```

8.28.1.2 TEST_CASE() [2/3]

```
TEST_CASE (
             "Cannot push to a full queue" )
```

8.28.1.3 TEST_CASE() [3/3]

```
TEST_CASE (
             "push pop" )
```

8.29 core/test/wrappers.test.cpp File Reference

```
#include <aare/core/Frame.hpp>
#include <aare/core/NDView.hpp>
#include <catch2/catch_test_macros.hpp>
#include <cstdint>
```

Functions

- TEST_CASE ("Frame")
- TEST_CASE ("NDView")
- TEST_CASE ("NDArray")

8.29.1 Function Documentation

8.29.1.1 TEST_CASE() [1/3]

```
TEST_CASE (
             "Frame" )
```

8.29.1.2 TEST_CASE() [2/3]

```
TEST_CASE (
             ¨NDArray¨  )
```

8.29.1.3 TEST_CASE() [3/3]

```
TEST_CASE (
             ¨NDView¨  )
```

8.30 data/jungfrau/read_frame.py File Reference

Namespaces

- namespace read_frame

Variables

- read_frame.header_dt
- int read_frame.rows = 512
- int read_frame.cols = 1024
- int read_frame.frames = 10
- read_frame.data = np.zeros((frames,rows,cols), dtype = np.uint16)
- read_frame.header = np.zeros(frames, dtype = header_dt)
- str read_frame.file_name = 'jungfrau_single_d0_f{}_0.raw'.format(file_id)
- read_frame.f
- read_frame.dtype
- read_frame.count
- read_frame.uint16

8.31 python/example/read_frame.py File Reference

Namespaces

- namespace example
- namespace example.read_frame

Variables

- example.read_frame.root_dir = Path(os.environ.get(¨PROJECT_ROOT_DIR¨))
- example.read_frame.data_path = str(root_dir / ¨data¨/¨jungfrau_single_master_0.json¨)
- example.read_frame.file = File(data_path)
- example.read_frame.frame = file.get_frame(0)
- example.read_frame.arr = np.array(frame.get_array())

8.32 data/numpy/write_test_files.py File Reference

Namespaces

- namespace write_test_files

Variables

- write_test_files.arr = np.arange(10, dtype = np.int32)
- write_test_files.arr2 = np.zeros((3,2,5), dtype = np.float64)

8.33 data/scripts/read_first_frame_number.py File Reference

Namespaces

- namespace read_first_frame_number

Variables

- read_first_frame_number.header_dt
- read_first_frame_number.frame_number = np.fromfile(f, dtype=header_dt, count=1)[¨Frame Number¨][0]

8.34 data/jungfrau/read_multiport.py File Reference

Namespaces

- namespace read_multiport

Variables

- read_multiport.header_dt
- int read_multiport.frames = 1
- int read_multiport.parts = 2
- int read_multiport.frame_cols = 1024
- int read_multiport.frame_rows = 512
- int read_multiport.part_cols = 1024
- int read_multiport.part_rows = 256
- read_multiport.parts_data = np.zeros((frames,parts,part_rows,part_cols), dtype = np.uint16)
- read_multiport.data = np.zeros((frames,frame_rows,frame_cols), dtype = np.uint16)
- read_multiport.header = np.zeros((frames,parts), dtype = header_dt)
- str read_multiport.file_name = f'jungfrau_double_d{part}_f{frame}_{0}.raw'
- read_multiport.f
- read_multiport.dtype
- read_multiport.count
- read_multiport.uint16
- read_multiport.axis

8.35 data/scripts/read_multiport.py File Reference

Namespaces

- namespace read_multiport

8.36 examples/json_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <iostream>
```

Macros

- #define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"

Functions

- void test (File &f, int frame_number)
- int main ()

8.36.1 Macro Definition Documentation

8.36.1.1 AARE_ROOT_DIR_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

8.36.2 Function Documentation

8.36.2.1 main()

```
int main ( )
```

8.36.2.2 test()

```
void test (
             File & f,
             int frame_number )
```

8.37 examples/logger_example.cpp File Reference

```
#include "aare/utils/logger.hpp"
#include <fstream>
#include <iostream>
```

Functions

- int main ()

8.37.1 Function Documentation

8.37.1.1 main()

```
int main ( )
```

8.38 examples/multiport_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <iostream>
```

Macros

- #define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"

Functions

- void test (File &f, int frame_number)
- int main ()

8.38.1 Macro Definition Documentation

8.38.1.1 AARE_ROOT_DIR_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

8.38.2 Function Documentation

8.38.2.1 main()

```
int main ( )
```

8.38.2.2 test()

```
void test (
             File & f,
             int frame_number )
```

8.39 examples/mythen_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <iostream>
```

Macros

- #define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"

Functions

- void test1 (File &f, int frame_number)
- void test2 (File &f, int frame_number)
- int main ()

8.39.1 Macro Definition Documentation

8.39.1.1 AARE_ROOT_DIR_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

8.39.2 Function Documentation

8.39.2.1 main()

```
int main ( )
```

8.39.2.2 test1()

```
void test1 (
            File & f,
            int frame_number )
```

8.39.2.3 test2()

```
void test2 (
            File & f,
            int frame_number )
```

8.40 examples/numpy_read_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include <iostream>
```

Macros

- #define AARE_ROOT_DIR_VAR ¨PROJECT_ROOT_DIR¨

Functions

- void test (File &f, int frame_number)
- int main ()

8.40.1 Macro Definition Documentation

8.40.1.1 AARE_ROOT_DIR_VAR

```
#define AARE_ROOT_DIR_VAR ¨PROJECT_ROOT_DIR¨
```

8.40.2 Function Documentation

8.40.2.1 main()

```
int main ( )
```

8.40.2.2 test()

```
void test (
            File & f,
            int frame_number )
```

8.41 examples/numpy_write_example.cpp File Reference

```
#include ¨aare/core/Frame.hpp¨
#include ¨aare/file_io/File.hpp¨
#include <iostream>
```

Macros

- #define AARE_ROOT_DIR_VAR ¨PROJECT_ROOT_DIR¨

Functions

- int main ()

8.41.1 Macro Definition Documentation

8.41.1.1 AARE_ROOT_DIR_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

8.41.2 Function Documentation

8.41.2.1 main()

```
int main ( )
```

8.42 examples/raw_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <iostream>
```

Macros

- #define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"

Functions

- void test (File &f, int frame_number)
- int main ()

8.42.1 Macro Definition Documentation

8.42.1.1 AARE_ROOT_DIR_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

8.42.2 Function Documentation

8.42.2.1 main()

```
int main ( )
```

8.42.2.2 test()

```
void test (
            File & f,
            int frame_number )
```

8.43 examples/zmq_receiver_example.cpp File Reference

```
#include "aare/network_io/ZmqSocketReceiver.hpp"
#include "aare/network_io/defs.hpp"
#include <cassert>
#include <fmt/core.h>
#include <string>
```

Functions

- int main ()

8.43.1 Function Documentation

8.43.1.1 main()

```
int main ( )
```

8.44 examples/zmq_restream_example.cpp File Reference

```
#include <chrono>
#include <thread>
#include "aare/file_io/File.hpp"
#include "aare/network_io/ZmqSocketSender.hpp"
#include <boost/program_options.hpp>
```

Functions

- int main (int argc, char ∗∗argv)

8.44.1 Function Documentation

8.44.1.1 main()

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

8.45 examples/zmq_sender_example.cpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocketSender.hpp"
#include "aare/network_io/defs.hpp"
#include "aare/utils/logger.hpp"
#include <ctime>
#include <fmt/core.h>
#include <string>
#include <unistd.h>
```

Functions

- int main ()

8.45.1 Function Documentation

8.45.1.1 main()

```
int main ( )
```

8.46 file_io/include/aare/file_io/File.hpp File Reference

```
#include "aare/file_io/FileInterface.hpp"
```

Data Structures

- class aare::File

  RAII File class for reading and writing image files in various formats wrapper on a FileInterface to abstract the underlying file format.

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.47 File.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/file_io/FileInterface.hpp"

namespace aare {

class File {
  private:
    FileInterface *file_impl;

  public:
    File(std::filesystem::path fname, std::string mode, FileConfig cfg = {});
    void write(Frame &frame);
    Frame read();
    Frame iread(size_t frame_number);
    std::vector<Frame> read(size_t n_frames);
    void read_into(std::byte *image_buf);
    void read_into(std::byte *image_buf, size_t n_frames);
    size_t frame_number(size_t frame_index);
    size_t bytes_per_frame();
    size_t pixels();
    void seek(size_t frame_number);
    size_t tell() const;
    size_t total_frames() const;
    ssize_t rows() const;
    ssize_t cols() const;
    ssize_t bitdepth() const;

    File(File &&other);

    ~File();
};

} // namespace aare
```

8.48 file_io/include/aare/file_io/FileInterface.hpp File Reference

```
#include "aare/core/DType.hpp"
#include "aare/core/Frame.hpp"
#include "aare/core/defs.hpp"
#include <filesystem>
#include <vector>
```

Data Structures

- struct aare::FileConfig

  FileConfig structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.
- class aare::FileInterface

  FileInterface class to define the interface for file operations.

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.49 FileInterface.hpp

Go to the documentation of this file.

```cpp
#pragma once
#include "aare/core/DType.hpp"
#include "aare/core/Frame.hpp"
#include "aare/core/defs.hpp"
#include <filesystem>
#include <vector>

namespace aare {

struct FileConfig {
    aare::DType dtype = aare::DType(typeid(uint16_t));
    uint64_t rows;
    uint64_t cols;
    xy geometry{1, 1};
    bool operator==(const FileConfig &other) const {
        return dtype == other.dtype && rows == other.rows && cols == other.cols && geometry ==
other.geometry;
    }
    bool operator!=(const FileConfig &other) const { return !(*this == other); }
};

class FileInterface {
  public:
    virtual void write(Frame &frame) = 0;

    // virtual void write(std::vector<Frame> &frames) = 0;

    virtual Frame read() = 0;

    virtual std::vector<Frame> read(size_t n_frames) = 0; // Is this the right interface?

    virtual void read_into(std::byte *image_buf) = 0;

    virtual void read_into(std::byte *image_buf, size_t n_frames) = 0;

    virtual size_t frame_number(size_t frame_index) = 0;

    virtual size_t bytes_per_frame() = 0;

    virtual size_t pixels() = 0;

    virtual void seek(size_t frame_number) = 0;

    virtual size_t tell() = 0;

    virtual size_t total_frames() const = 0;
    virtual ssize_t rows() const = 0;
    virtual ssize_t cols() const = 0;
    virtual ssize_t bitdepth() const = 0;

    Frame iread(size_t frame_number) {
        auto old_pos = tell();
        seek(frame_number);
        Frame tmp = read();
        seek(old_pos);
        return tmp;
    };

    std::vector<Frame> iread(size_t frame_number, size_t n_frames) {
        auto old_pos = tell();
        seek(frame_number);
        std::vector<Frame> tmp = read(n_frames);
        seek(old_pos);
        return tmp;
    }

    // function to query the data type of the file
    /*virtual DataType dtype = 0; */

    virtual ~FileInterface(){

    };

  public:
    std::string m_mode;
    std::filesystem::path m_fname;
    std::filesystem::path m_base_path;
    std::string m_base_name, m_ext;
    int m_findex;
    size_t m_total_frames{};
    size_t max_frames_per_file{};
    std::string version;
    DetectorType m_type;
    ssize_t m_rows{};
    ssize_t m_cols{};
    ssize_t m_bitdepth{};
    size_t current_frame{};
};

} // namespace aare
```

8.50 file_io/include/aare/file_io/NumpyFile.hpp File Reference

```
#include "aare/core/DType.hpp"
#include "aare/core/defs.hpp"
#include "aare/file_io/FileInterface.hpp"
#include "aare/file_io/NumpyHelpers.hpp"
#include <filesystem>
#include <iostream>
#include <numeric>
```

Data Structures

- class aare::NumpyFile

    NumpyFile class to read and write numpy files.

Namespaces

- namespace aare

    Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.51 NumpyFile.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/DType.hpp"
#include "aare/core/defs.hpp"
#include "aare/file_io/FileInterface.hpp"
#include "aare/file_io/NumpyHelpers.hpp"
#include <filesystem>
#include <iostream>
#include <numeric>

namespace aare {

class NumpyFile : public FileInterface {

  public:
    NumpyFile(const std::filesystem::path &fname, const std::string &mode = "r", FileConfig cfg = {});

    void write(Frame &frame) override;
    Frame read() override { return get_frame(this->current_frame++); }

    std::vector<Frame> read(size_t n_frames) override;
    void read_into(std::byte *image_buf) override { return get_frame_into(this->current_frame++,
  image_buf); }
    void read_into(std::byte *image_buf, size_t n_frames) override;
    size_t frame_number(size_t frame_index) override { return frame_index; };
    size_t bytes_per_frame() override;
    size_t pixels() override;
    void seek(size_t frame_number) override { this->current_frame = frame_number; }
    size_t tell() override { return this->current_frame; }
    size_t total_frames() const override { return m_header.shape[0]; }
    ssize_t rows() const override { return m_header.shape[1]; }
    ssize_t cols() const override { return m_header.shape[2]; }
    ssize_t bitdepth() const override { return m_header.dtype.bitdepth(); }

    DType dtype() const { return m_header.dtype; }

    std::vector<size_t> shape() const { return m_header.shape; }

    template <typename T, size_t NDim> NDArray<T, NDim> load() {
        NDArray<T, NDim> arr(make_shape<NDim>(m_header.shape));
        fseek(fp, header_size, SEEK_SET);
        fread(arr.data(), sizeof(T), arr.size(), fp);
        return arr;
    }

    ~NumpyFile();

  private:
    FILE *fp = nullptr;
    size_t initial_header_len = 0;
    size_t current_frame{};
    uint32_t header_len{};
    uint8_t header_len_size{};
    size_t header_size{};
    NumpyHeader m_header;
    uint8_t major_ver_{};
    uint8_t minor_ver_{};

    void load_metadata();
    void get_frame_into(size_t, std::byte *);
    Frame get_frame(size_t frame_number);
};

} // namespace aare
```

8.52 file_io/include/aare/file_io/NumpyHelpers.hpp File Reference

```
#include <algorithm>
#include <array>
#include <filesystem>
#include <fstream>
#include <iostream>
#include <numeric>
#include <sstream>
#include <string>
#include <unordered_map>
#include <vector>
#include "aare/core/DType.hpp"
#include "aare/core/defs.hpp"
```

Data Structures

- struct aare::NumpyHeader

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare::NumpyHelpers

Typedefs

- using aare::shape_t = std::vector< size_t >

Functions

- std::string aare::NumpyHelpers::parse_str (const std::string &in)
- std::string aare::NumpyHelpers::trim (const std::string &str)
- std::vector< std::string > aare::NumpyHelpers::parse_tuple (std::string in)
- bool aare::NumpyHelpers::parse_bool (const std::string &in)
- std::string aare::NumpyHelpers::get_value_from_map (const std::string &mapstr)
- std::unordered_map< std::string, std::string > aare::NumpyHelpers::parse_dict (std::string in, const std↩ ::vector< std::string > &keys)
- template<typename T , size_t N>
  bool aare::NumpyHelpers::in_array (T val, const std::array< T, N > &arr)
- bool aare::NumpyHelpers::is_digits (const std::string &str)
- aare::DType aare::NumpyHelpers::parse_descr (std::string typestring)
- size_t aare::NumpyHelpers::write_header (std::filesystem::path fname, const NumpyHeader &header)
- size_t aare::NumpyHelpers::write_header (std::ostream &out, const NumpyHeader &header)

Variables

- const constexpr std::array< char, 6 > aare::NumpyHelpers::magic_str {'\x93', 'N', 'U', 'M', 'P', 'Y'}
- const uint8_t aare::NumpyHelpers::magic_string_length {6}

8.53 NumpyHelpers.hpp

Go to the documentation of this file.

```

#pragma once
#include <algorithm>
#include <array>
#include <filesystem>
#include <fstream>
#include <iostream>
#include <numeric>
#include <sstream>
#include <string>
#include <unordered_map>
#include <vector>

#include "aare/core/DType.hpp"
#include "aare/core/defs.hpp"

namespace aare {

using shape_t = std::vector<size_t>;

struct NumpyHeader {
    DType dtype{aare::DType::ERROR};
    bool fortran_order{false};
    shape_t shape{};

    std::string to_string() const;
};

namespace NumpyHelpers {

const constexpr std::array<char, 6> magic_str{'\x93', 'N', 'U', 'M', 'P', 'Y'};
const uint8_t magic_string_length{6};

std::string parse_str(const std::string &in);
std::string trim(const std::string &str);

std::vector<std::string> parse_tuple(std::string in);
bool parse_bool(const std::string &in);

std::string get_value_from_map(const std::string &mapstr);

std::unordered_map<std::string, std::string> parse_dict(std::string in, const std::vector<std::string>
    &keys);

template <typename T, size_t N> bool in_array(T val, const std::array<T, N> &arr) {
    return std::find(std::begin(arr), std::end(arr), val) != std::end(arr);
}
bool is_digits(const std::string &str);

aare::DType parse_descr(std::string typestring);
size_t write_header(std::filesystem::path fname, const NumpyHeader &header);
size_t write_header(std::ostream &out, const NumpyHeader &header);

} // namespace NumpyHelpers
} // namespace aare
```

8.54 file_io/include/aare/file_io/RawFile.hpp File Reference

```
#include ¨aare/core/Frame.hpp¨
#include ¨aare/file_io/FileInterface.hpp¨
#include ¨aare/file_io/SubFile.hpp¨
```

Data Structures

- class aare::RawFile

  RawFile class to read .raw and .json files.

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.55 RawFile.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/Frame.hpp"
#include "aare/file_io/FileInterface.hpp"
#include "aare/file_io/SubFile.hpp"

namespace aare {

class RawFile : public FileInterface {
  public:
    RawFile(const std::filesystem::path &fname, const std::string &mode = "r", const FileConfig &cfg =
    {});

    void write(Frame &frame) override { throw std::runtime_error("Not implemented"); };
    Frame read() override { return get_frame(this->current_frame++); };
    std::vector<Frame> read(size_t n_frames) override;
    void read_into(std::byte *image_buf) override { return get_frame_into(this->current_frame++,
    image_buf); };
    void read_into(std::byte *image_buf, size_t n_frames) override;
    size_t frame_number(size_t frame_index) override;

    size_t bytes_per_frame() override { return m_rows * m_cols * m_bitdepth / 8; }

    size_t pixels() override { return m_rows * m_cols; }

    // goto frame number
    void seek(size_t frame_number) override { this->current_frame = frame_number; };

    // return the position of the file pointer (in number of frames)
    size_t tell() override { return this->current_frame; };

    static bool is_master_file(std::filesystem::path fpath);

    inline void set_config(int row, int col) {
        cfg.module_gap_row = row;
        cfg.module_gap_col = col;
    }
    // TODO! Deal with fast quad and missing files

    void find_number_of_subfiles();

    inline std::filesystem::path master_fname();
    inline std::filesystem::path data_fname(int mod_id, int file_id);

    ~RawFile();

    size_t total_frames() const override { return m_total_frames; }
    ssize_t rows() const override { return m_rows; }
    ssize_t cols() const override { return m_cols; }
    ssize_t bitdepth() const override { return m_bitdepth; }

  private:
    void get_frame_into(size_t frame_number, std::byte *image_buf);

    Frame get_frame(size_t frame_number);

    void parse_fname();

    void parse_metadata();

    void parse_raw_metadata();

    void parse_json_metadata();

    void find_geometry();

    sls_detector_header read_header(const std::filesystem::path &fname);

    void open_subfiles();

    size_t n_subfiles;
    size_t n_subfile_parts;
    std::vector<std::vector<SubFile *» subfiles;
    int subfile_rows, subfile_cols;
    xy geometry;
    std::vector<xy> positions;
    RawFileConfig cfg{0, 0};
    TimingMode timing_mode;
    bool quad{false};
};

} // namespace aare
```

8.56 file_io/include/aare/file_io/SubFile.hpp File Reference

```
#include ¨aare/core/defs.hpp¨
#include <cstdint>
#include <filesystem>
#include <map>
#include <variant>
```

Data Structures

- class aare::SubFile

  Class to read a subfile from a RawFile.

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.57 SubFile.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/defs.hpp"
#include <cstdint>
#include <filesystem>
#include <map>
#include <variant>

namespace aare {

class SubFile {
  protected:
    using pfunc = size_t (SubFile::*)(std::byte *);
    pfunc read_impl = nullptr;
    std::map<std::pair<DetectorType, int>, pfunc> read_impl_map = {
        {{DetectorType::Moench, 16}, &SubFile::read_impl_reorder<uint16_t>},
        {{DetectorType::Jungfrau, 16}, &SubFile::read_impl_normal},
        {{DetectorType::ChipTestBoard, 16}, &SubFile::read_impl_normal},
        {{DetectorType::Mythen3, 32}, &SubFile::read_impl_normal},
        {{DetectorType::Eiger, 32}, &SubFile::read_impl_normal},
        {{DetectorType::Eiger, 16}, &SubFile::read_impl_normal}

    };

  public:
    SubFile(std::filesystem::path fname, DetectorType detector, ssize_t rows, ssize_t cols, uint16_t
bitdepth);

    size_t read_impl_normal(std::byte *buffer);

    template <typename DataType> size_t read_impl_flip(std::byte *buffer);

    template <typename DataType> size_t read_impl_reorder(std::byte *buffer);

    size_t get_part(std::byte *buffer, int frame_number);
    size_t frame_number(int frame_index);

    // TODO: define the inlines as variables and assign them in constructor
    inline size_t bytes_per_part() { return (m_bitdepth / 8) * m_rows * m_cols; }
    inline size_t pixels_per_part() { return m_rows * m_cols; }

  protected:
    FILE *fp = nullptr;
    ssize_t m_bitdepth;
    std::filesystem::path m_fname;
    ssize_t m_rows{};
    ssize_t m_cols{};
    ssize_t n_frames{};
    int m_sub_file_index_{};
};

} // namespace aare
```

8.58 file_io/src/File.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/file_io/NumpyFile.hpp"
#include "aare/file_io/RawFile.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/format.h>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.59 file_io/src/NumpyFile.cpp File Reference

```
#include "aare/file_io/NumpyFile.hpp"
#include "aare/utils/logger.hpp"
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.60 file_io/src/NumpyHelpers.cpp File Reference

```
#include "aare/file_io/NumpyHelpers.hpp"
#include <iterator>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare::NumpyHelpers

Functions

- std::unordered_map< std::string, std::string > aare::NumpyHelpers::parse_dict (std::string in, const std↩ ::vector< std::string > &keys)
- aare::DType aare::NumpyHelpers::parse_descr (std::string typestring)
- bool aare::NumpyHelpers::parse_bool (const std::string &in)
- std::string aare::NumpyHelpers::get_value_from_map (const std::string &mapstr)
- bool aare::NumpyHelpers::is_digits (const std::string &str)
- std::vector< std::string > aare::NumpyHelpers::parse_tuple (std::string in)
- std::string aare::NumpyHelpers::trim (const std::string &str)
- std::string aare::NumpyHelpers::parse_str (const std::string &in)
- void aare::NumpyHelpers::write_magic (std::ostream &ostream, int version_major, int version_minor)
- template<typename T >
  std::string aare::NumpyHelpers::write_tuple (const std::vector< T > &v)
- std::string aare::NumpyHelpers::write_boolean (bool b)
- std::string aare::NumpyHelpers::write_header_dict (const std::string &descr, bool fortran_order, const shape_t &shape)
- size_t aare::NumpyHelpers::write_header (std::filesystem::path fname, const NumpyHeader &header)
- size_t aare::NumpyHelpers::write_header (std::ostream &out, const NumpyHeader &header)

8.61 file_io/src/RawFile.cpp File Reference

```
#include "aare/file_io/RawFile.hpp"
#include "aare/core/defs.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/format.h>
#include <nlohmann/json.hpp>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Typedefs

- using json = nlohmann::json

8.61.1 Typedef Documentation

8.61.1.1 json

```
using json = nlohmann::json
```

8.62 file_io/src/SubFile.cpp File Reference

```
#include "aare/file_io/SubFile.hpp"
#include "aare/utils/logger.hpp"
#include <cstring>
#include <fmt/core.h>
#include <iostream>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.63 file_io/test/NumpyFile.test.cpp File Reference

```
#include "aare/file_io/NumpyFile.hpp"
#include "aare/core/NDArray.hpp"
#include <catch2/catch_test_macros.hpp>
#include "test_config.hpp"
```

Functions

- TEST_CASE ("Read a 1D numpy file with int32 data type")
- TEST_CASE ("Read a 3D numpy file with np.double data type")

8.63.1 Function Documentation

8.63.1.1 TEST_CASE() [1/2]

```
TEST_CASE (
            "Read a 1D numpy file with int32 data type" )
```

8.63.1.2 TEST_CASE() [2/2]

```
TEST_CASE (
            "Read a 3D numpy file with np.double data type" )
```

8.64 file_io/test/NumpyHelpers.test.cpp File Reference

```
#include "aare/file_io/NumpyHelpers.hpp"
#include <catch2/catch_test_macros.hpp>
```

Functions

- TEST_CASE ("is_digits with a few standard cases")
- TEST_CASE ("Check for quotes and return stripped string")
- TEST_CASE ("parsing a string without quotes throws")
- TEST_CASE ("trim whitespace")
- TEST_CASE ("parse data type descriptions")
- TEST_CASE ("is element in array")
- TEST_CASE ("Parse numpy dict")

8.64.1 Function Documentation

8.64.1.1 TEST_CASE() [1/7]

```
TEST_CASE (
            "Check for quotes and return stripped string" )
```

8.64.1.2 TEST_CASE() [2/7]

```
TEST_CASE (
            "is element in array" )
```

8.64.1.3 TEST_CASE() [3/7]

```
TEST_CASE (
            ¨is_digits with a few standard cases¨  )
```

8.64.1.4 TEST_CASE() [4/7]

```
TEST_CASE (
            ¨parse data type descriptions¨  )
```

8.64.1.5 TEST_CASE() [5/7]

```
TEST_CASE (
            ¨Parse numpy dict¨  )
```

8.64.1.6 TEST_CASE() [6/7]

```
TEST_CASE (
            ¨parsing a string without quotes throws¨  )
```

8.64.1.7 TEST_CASE() [7/7]

```
TEST_CASE (
            ¨trim whitespace¨  )
```

8.65 file_io/test/RawFile.test.cpp File Reference

```
#include ¨aare/file_io/File.hpp¨
#include ¨aare/utils/logger.hpp¨
#include <catch2/catch_test_macros.hpp>
#include <filesystem>
#include ¨test_config.hpp¨
```

Functions

- TEST_CASE (¨Read number of frames from a jungfrau raw file¨)
- TEST_CASE (¨Read frame numbers from a jungfrau raw file¨)
- TEST_CASE (¨Read data from a jungfrau 500k single port raw file¨)
- TEST_CASE (¨Read frame numbers from a raw file¨)
- TEST_CASE (¨Compare reading from a numpy file with a raw file¨)

8.65.1 Function Documentation

8.65.1.1 TEST_CASE() [1/5]

```
TEST_CASE (
            ¨Compare reading from a numpy file with a raw file¨ )
```

8.65.1.2 TEST_CASE() [2/5]

```
TEST_CASE (
            ¨Read data from a jungfrau 500k single port raw file¨ )
```

8.65.1.3 TEST_CASE() [3/5]

```
TEST_CASE (
            ¨Read frame numbers from a jungfrau raw file¨ )
```

8.65.1.4 TEST_CASE() [4/5]

```
TEST_CASE (
            ¨Read frame numbers from a raw file¨ )
```

8.65.1.5 TEST_CASE() [5/5]

```
TEST_CASE (
            ¨Read number of frames from a jungfrau raw file¨ )
```

8.66 include/aare/aare.hpp File Reference

8.67 aare.hpp

Go to the documentation of this file.

```
00001 // This is the top level header to include and what most users will use
```

8.68 network_io/include/aare/network_io/ZmqHeader.hpp File Reference

```
#include ¨aare/core/Frame.hpp¨
#include ¨aare/utils/logger.hpp¨
#include ¨simdjson.h¨
#include <array>
#include <cstdint>
#include <map>
#include <string>
```

Data Structures

- struct aare::ZmqHeader

Namespaces

- namespace simdjson
- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.69 ZmqHeader.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/Frame.hpp"
#include "aare/utils/logger.hpp"

#include "simdjson.h"
#include <array>
#include <cstdint>
#include <map>
#include <string>
namespace simdjson {
template <> simdjson_inline simdjson::simdjson_result<std::array<int, 4»
simdjson::ondemand::value::get() noexcept {
    ondemand::array array;
    auto error = get_array().get(array);
    if (error) {
        return error;
    }
    std::array<int, 4> arr;
    int i = 0;
    for (auto v : array) {
        int64_t val;
        error = v.get_int64().get(val);

        if (error) {
            return error;
        }
        arr[i++] = val;
    }
    return arr;
}

template <> simdjson_inline simdjson::simdjson_result<uint32_t> simdjson::ondemand::value::get()
noexcept {
    size_t val;
    auto error = get_uint64().get(val);
    if (error) {
        return error;
    }
    if (val > std::numeric_limits<uint32_t>::max()) {
        return 1;
    }
    return static_cast<uint32_t>(val);
}

template <>
simdjson_inline simdjson::simdjson_result<std::map<std::string, std::string»
simdjson::ondemand::value::get() noexcept {
    std::map<std::string, std::string> map;
    ondemand::object obj;
    auto error = get_object().get(obj);
    if (error) {
        return error;
    }
    for (auto field : obj) {
        simdjson::ondemand::raw_json_string tmp;
        error = field.key().get(tmp);
        if (error) {
            return error;
        }
        error = field.value().get(tmp);
        if (error) {
            return error;
        }
        std::string_view key_view = field.unescaped_key();
        std::string key_str(key_view.data(), key_view.size());
        std::string_view value_view = field.value().get_string();
        map[key_str] = {value_view.data(), value_view.size()};
    }
    return map;
}

} // namespace simdjson

namespace aare {

struct ZmqHeader {
    bool data{true};
    uint32_t jsonversion{0};
    uint32_t dynamicRange{0};
    uint64_t fileIndex{0};
    uint32_t ndetx{0};
    uint32_t ndety{0};
    uint32_t npixelsx{0};
    uint32_t npixelsy{0};
    uint32_t imageSize{0};
    uint64_t acqIndex{0};
    uint64_t frameIndex{0};
    double progress{0};
    std::string fname;
    uint64_t frameNumber{0};
    uint32_t expLength{0};
    uint32_t packetNumber{0};
    uint64_t detSpec1{0};
    uint64_t timestamp{0};
    uint16_t modId{0};
    uint16_t row{0};
    uint16_t column{0};
    uint16_t detSpec2{0};
    uint32_t detSpec3{0};
    uint16_t detSpec4{0};
    uint8_t detType{0};
    uint8_t version{0};
    int flipRows{0};
    uint32_t quad{0};
    bool completeImage{false};
    std::map<std::string, std::string> addJsonHeader;
    std::array<int, 4> rx_roi{};

    std::string to_string() const;
    void from_string(std::string &s);
    // compare operator
    bool operator==(const ZmqHeader &other) const;
};

} // namespace aare
```

8.70 network_io/include/aare/network_io/ZmqSocket.hpp File Reference

```
#include <string>
```

Data Structures

- class aare::ZmqSocket

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.71 ZmqSocket.hpp

Go to the documentation of this file.

```
#pragma once

#include <string>

// Socket to receive data from a ZMQ publisher
// needs to be in sync with the main library (or maybe better use the versioning in the header)

// forward declare zmq_msg_t to avoid including zmq.h in the header
class zmq_msg_t;

namespace aare {

class ZmqSocket {
  protected:
    void *m_context{nullptr};
    void *m_socket{nullptr};
    std::string m_endpoint;
    int m_zmq_hwm{1000};
    int m_timeout_ms{1000};
    size_t m_potential_frame_size{1024 * 1024};
    constexpr static size_t m_max_header_size = 1024;
    char *m_header_buffer = new char[m_max_header_size];

  public:
    ZmqSocket() = default;
    ~ZmqSocket();

    ZmqSocket(const ZmqSocket &) = delete;
    ZmqSocket operator=(const ZmqSocket &) = delete;
    ZmqSocket(ZmqSocket &&) = delete;

    void disconnect();
    void set_zmq_hwm(int hwm);
    void set_timeout_ms(int n);
    void set_potential_frame_size(size_t size);
};

} // namespace aare
```

8.72 network_io/include/aare/network_io/ZmqSocketReceiver.hpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocket.hpp"
#include "aare/network_io/defs.hpp"
#include <cstdint>
#include <string>
```

Data Structures

- class aare::ZmqSocketReceiver

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.73 ZmqSocketReceiver.hpp

Go to the documentation of this file.

```
#pragma once

#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocket.hpp"
#include "aare/network_io/defs.hpp"

#include <cstdint>
#include <string>

// Socket to receive data from a ZMQ publisher
// needs to be in sync with the main library (or maybe better use the versioning in the header)

// forward declare zmq_msg_t to avoid including zmq.h in the header
class zmq_msg_t;

namespace aare {

class ZmqSocketReceiver : public ZmqSocket {
  public:
    ZmqSocketReceiver(const std::string &endpoint);
    void connect();
    std::vector<ZmqFrame> receive_n();

  private:
    int receive_data(std::byte *data, size_t size);
    ZmqFrame receive_zmqframe();
    ZmqHeader receive_header();
};

} // namespace aare
```

8.74 network_io/include/aare/network_io/ZmqSocketSender.hpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocket.hpp"
#include "aare/network_io/defs.hpp"
```

Data Structures

- class aare::ZmqSocketSender

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.75 ZmqSocketSender.hpp

Go to the documentation of this file.

```
#pragma once
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocket.hpp"
#include "aare/network_io/defs.hpp"

namespace aare {
class ZmqSocketSender : public ZmqSocket {
  public:
    ZmqSocketSender(const std::string &endpoint);
    void bind();
    size_t send(const ZmqHeader &header, const std::byte *data, size_t size);
    size_t send(const ZmqFrame &zmq_frame);
    size_t send(const std::vector<ZmqFrame> &zmq_frames);
};
} // namespace aare
```

8.76 network_io/src/ZmqHeader.cpp File Reference

```
#include "aare/network_io/ZmqHeader.hpp"
#include "simdjson.h"
```

Namespaces

- namespace aare

    Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

Functions

- template<typename T >
  void write_digit (std::string &s, const std::string &key, const T &value)

    write a digit to a string takes key and value and outputs->"key": value,
- void write_str (std::string &s, const std::string &key, const std::string &value)
- void write_map (std::string &s, const std::string &key, const std::map< std::string, std::string > &value)
- void write_array (std::string &s, const std::string &key, const std::array< int, 4 > &value)

8.76.1 Function Documentation

8.76.1.1 write_array()

```
void write_array (
            std::string & s,
            const std::string & key,
            const std::array< int, 4 > & value )
```

8.76.1.2 write_digit()

```
template<typename T >
void write_digit (
            std::string & s,
            const std::string & key,
            const T & value )
```

write a digit to a string takes key and value and outputs->"key": value,

Template Parameters

T	type of value (int, uint32_t, ...)

Parameters

s	string to append to
key	key to write
value	value to write

Returns

void

Note

- can't use concepts here because we are using c++17

8.76.1.3 write_map()

```
void write_map (
            std::string & s,
            const std::string & key,
            const std::map< std::string, std::string > & value )
```

8.76.1.4 write_str()

```
void write_str (
            std::string & s,
            const std::string & key,
            const std::string & value )
```

8.77 network_io/src/ZmqSocket.cpp File Reference

```
#include "aare/network_io/ZmqSocket.hpp"
#include <zmq.h>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.78 network_io/src/ZmqSocketReceiver.cpp File Reference

```
#include "aare/network_io/ZmqSocketReceiver.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/core.h>
#include <zmq.h>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.79 network_io/src/ZmqSocketSender.cpp File Reference

```
#include "aare/network_io/ZmqSocketSender.hpp"
#include <cassert>
#include <zmq.h>
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.80 network_io/test/ZmqHeader.test.cpp File Reference

```
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/utils/logger.hpp"
#include <catch2/catch_test_macros.hpp>
```

Functions

- TEST_CASE ("Test ZmqHeader")

8.80.1 Function Documentation

8.80.1.1 TEST_CASE()

```
TEST_CASE (
            "Test ZmqHeader" )
```

8.81 python/aare/File.py File Reference

Data Structures

- class aare.File.File

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare.File

8.82 python/aare/Frame.py File Reference

Data Structures

- class aare.Frame.Frame

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare.Frame

8.83 python/aare/__init__.py File Reference

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.

8.84 python/example/__init__.py File Reference

8.85 python/src/bindings.cpp File Reference

```
#include <cstdint>
#include <filesystem>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
#include <string>
#include ¨aare/FileHandler.hpp¨
#include ¨aare/core/Frame.hpp¨
#include ¨aare/core/defs.hpp¨
```

Functions

- PYBIND11_MODULE (_aare, m)

8.85.1 Function Documentation

8.85.1.1 PYBIND11_MODULE()

```
PYBIND11_MODULE (
            _aare ,
            m )
```

8.86 README.md File Reference

8.87 tests/test.cpp File Reference

```
#include <catch2/catch_test_macros.hpp>
#include <filesystem>
#include <fstream>
#include "test_config.hpp"
```

Functions

- TEST_CASE ("Test suite can find data assets")
- TEST_CASE ("Test suite can open data assets")

8.87.1 Function Documentation

8.87.1.1 TEST_CASE() [1/2]

```
TEST_CASE (
            "Test suite can find data assets" )
```

8.87.1.2 TEST_CASE() [2/2]

```
TEST_CASE (
            "Test suite can open data assets" )
```

8.88 utils/include/aare/utils/logger.hpp File Reference

```
#include <filesystem>
#include <fstream>
#include <iostream>
#include <map>
#include <vector>
```

Data Structures

- class aare::logger::Logger

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare::logger
- namespace aare::logger::internal

Macros

- #define LOCATION std::string(__FILE__) + std::string(":") + std::to_string(__LINE__) + ":" + std::string(__↩
  func__) + ":"

Enumerations

- enum aare::logger::LOGGING_LEVEL { aare::logger::DEBUG = 0 , aare::logger::INFO = 1 , aare::logger::WARNING = 2 , aare::logger::ERROR = 3 }

Functions

- template<typename T >
  std::ostream & operator<< (std::ostream &out, const std::vector< T > &v)
- template<typename T , size_t N>
  std::ostream & operator<< (std::ostream &out, const std::array< T, N > &v)
- template<typename K , typename V >
  std::ostream & operator<< (std::ostream &out, const std::map< K, V > &v)
- template<LOGGING_LEVEL level, typename... Strings>
  void aare::logger::log (const Strings... s)
- template<typename... Strings>
  void aare::logger::debug (const Strings... s)
- template<typename... Strings>
  void aare::logger::info (const Strings... s)
- template<typename... Strings>
  void aare::logger::warn (const Strings... s)
- template<typename... Strings>
  void aare::logger::error (const Strings... s)
- void aare::logger::set_streams (std::streambuf ∗out, std::streambuf ∗err)
- void aare::logger::set_streams (std::streambuf ∗out)
- void aare::logger::set_verbosity (LOGGING_LEVEL level)
- void aare::logger::set_output_file (std::string filename)
- Logger & aare::logger::get_logger_instance ()

Variables

- aare::logger::Logger aare::logger::internal::logger_instance = aare::logger::Logger()

8.88.1 Macro Definition Documentation

8.88.1.1 LOCATION

```
#define LOCATION std::string(__FILE__) + std::string(¨:¨) + std::to_string(__LINE__) + ¨:¨ +
std::string(__func__) + ¨:¨
```

8.88.2 Function Documentation

8.88.2.1 operator<<() [1/3]

```
template<typename T , size_t N>
std::ostream & operator<< (
            std::ostream & out,
            const std::array< T, N > & v )
```

8.88.2.2 operator<<() [2/3]

```
template<typename K , typename V >
std::ostream & operator<< (
            std::ostream & out,
            const std::map< K, V > & v )
```

8.88.2.3 operator<<() [3/3]

```
template<typename T >
std::ostream & operator<< (
            std::ostream & out,
            const std::vector< T > & v )
```

8.89 logger.hpp

Go to the documentation of this file.

```
#pragma once
#include <filesystem>
#include <fstream>
#include <iostream>
#include <map>
#include <vector>

#define LOCATION std::string(__FILE__) + std::string(":") + std::to_string(__LINE__) + ":" +
     std::string(__func__) + ":"

// operator overload to print vectors
// typename T must be printable (i.e. have the « operator)
template <typename T> std::ostream &operator«(std::ostream &out, const std::vector<T> &v) {
    out « "[";
    size_t last = v.size() - 1;
    for (size_t i = 0; i < v.size(); ++i) {
        out « v[i];
        if (i != last)
            out « ", ";
    }
    out « "]";
    return out;
}

```

```
// operator overload for std::array
template <typename T, size_t N> std::ostream &operator«(std::ostream &out, const std::array<T, N> &v)
{
    out « "[";
    size_t last = N - 1;
    for (size_t i = 0; i < N; ++i) {
        out « v[i];
        if (i != last)
            out « ", ";
    }
    out « "]";
    return out;
}
// operator overlaod for std::map
template <typename K, typename V> std::ostream &operator«(std::ostream &out, const std::map<K, V> &v)
{
    out « "{";
    size_t i = 0;
    for (auto &kv : v) {
        out « kv.first « ": " « kv.second « ((++i != v.size()) ? ", " : "");
    }

    out « "}";
    return out;
}

namespace aare {

namespace logger {
enum LOGGING_LEVEL {
    DEBUG = 0,
    INFO = 1,
    WARNING = 2,
    ERROR = 3

};

class Logger {

    std::streambuf *standard_buf = std::cout.rdbuf();
    std::streambuf *error_buf = std::cerr.rdbuf();
    std::ostream *standard_output;
    std::ostream *error_output;
    LOGGING_LEVEL VERBOSITY_LEVEL = LOGGING_LEVEL::INFO;

    std::ofstream out_file;

  public:
    void set_output_file(std::string filename) {
        if (out_file.is_open())
            out_file.close();
        out_file.open(filename);
        set_streams(out_file.rdbuf());
    }
    void set_streams(std::streambuf *out, std::streambuf *err) {
        delete standard_output;
        delete error_output;
        standard_output = new std::ostream(out);
        error_output = new std::ostream(err);
    }
    void set_streams(std::streambuf *out) { set_streams(out, out); }
    void set_verbosity(LOGGING_LEVEL level) { VERBOSITY_LEVEL = level; }
    Logger() {
        standard_output = new std::ostream(standard_buf);
        error_output = new std::ostream(error_buf);
    }

    ~Logger() {
        if (out_file.is_open())
            out_file.close();

        standard_output->flush();
        error_output->flush();
        delete standard_output;
        delete error_output;
    }
    template <LOGGING_LEVEL level, typename... Strings> void log(const Strings... s) {
        if (level >= VERBOSITY_LEVEL)
            log_<level>(s...);
    }
    template <typename... Strings> void debug(const Strings... s) {
log<LOGGING_LEVEL::DEBUG>("[DEBUG]", s...); }
    template <typename... Strings> void info(const Strings... s) { log<LOGGING_LEVEL::INFO>("[INFO]",
s...); }
    template <typename... Strings> void warn(const Strings... s) {
log<LOGGING_LEVEL::WARNING>("[WARN]", s...); }
    template <typename... Strings> void error(const Strings... s) {
```

```
    log<LOGGING_LEVEL::ERROR>("[ERROR]", s...); }

  private:
    template <LOGGING_LEVEL level> void log_() {
        if (level == LOGGING_LEVEL::ERROR) {
            *error_output « std::endl;
        } else {
            *standard_output « std::endl;
        }
    }
    template <LOGGING_LEVEL level, typename First, typename... Strings> void log_(First arg, const
    Strings... s) {
        if (level == LOGGING_LEVEL::ERROR) {
            *error_output « (arg) « ' ';
            error_output->flush();
        } else {
            *standard_output « (arg) « ' ';
            standard_output->flush();
        }
        log_<level>(s...);
    }
};

namespace internal {

extern aare::logger::Logger logger_instance;
} // namespace internal

template <LOGGING_LEVEL level, typename... Strings> void log(const Strings... s) {
    internal::logger_instance.log<level>(s...);
}
template <typename... Strings> void debug(const Strings... s) { internal::logger_instance.debug(s...);
}
template <typename... Strings> void info(const Strings... s) { internal::logger_instance.info(s...); }
template <typename... Strings> void warn(const Strings... s) { internal::logger_instance.warn(s...); }
template <typename... Strings> void error(const Strings... s) { internal::logger_instance.error(s...);
}

extern void set_streams(std::streambuf *out, std::streambuf *err);
extern void set_streams(std::streambuf *out);
extern void set_verbosity(LOGGING_LEVEL level);
extern void set_output_file(std::string filename);
extern Logger &get_logger_instance();

} // namespace logger

} // namespace aare
```

8.90 utils/src/logger.cpp File Reference

```
#include "aare/utils/logger.hpp"
```

Namespaces

- namespace aare

  Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace aare::logger
- namespace aare::logger::internal

Functions

- void aare::logger::set_streams (std::streambuf *out, std::streambuf *err)
- void aare::logger::set_streams (std::streambuf *out)
- void aare::logger::set_verbosity (LOGGING_LEVEL level)
- Logger & aare::logger::get_logger_instance ()
- void aare::logger::set_output_file (std::string filename)

8.91 utils/test/logger.test.cpp File Reference

```
#include ¨aare/utils/logger.hpp¨
#include <catch2/catch_test_macros.hpp>
```

Index