libAHio Reference Manual

0.1

Generated by Doxygen 1.4.4

Fri Apr 7 01:18:08 2006

Contents

Chapter 1

libAHio Module Index

1.1 libAHio Modules

Here is a list of all modules:

Chapter 2

libAHio Directory Hierarchy

2.1 libAHio Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 3

libAHio Namespace Index

3.1 libAHio Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

libAHio Hierarchical Index

4.1 libAHio Class Hierarchy

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

Chapter 5

libAHio Class Index

5.1 libAHio Class List

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

Chapter 6

libAHio File Index

6.1 libAHio File List

Here is a list of all files with brief descriptions:

Chapter 7

libAHio Module Documentation

7.1 File 'Open' Policy classes

Chapter 8

libAHio Directory Documentation

8.1 AH/ Directory Reference

Directories

- directory IO

8.2 AH/IO/Cdda/ Directory Reference

Files

- file Sector.h

8.3 AH/IO/ Directory Reference

Directories

- directory Cdda
- directory Midi
- directory Mpeg
- directory Riff
- directory Smart

Files

- file AudioFile.h
- file CFormat.h
- file CmdOptHandler.h
- file CmdOption.h
- file Convert.h
- file Dir.h
- file Error.h
- file Plot.h
- file Profile.h
- file TimeStamp.h

8.4 AH/IO/Midi/ Directory Reference

Files

- file Device.h
- file RawCmd.h
- file SysExCmd.h

8.5 AH/IO/Mpeg/ Directory Reference

Files

- file FrameHeader.h

8.6 AH/IO/Riff/ Directory Reference

Files

- file Chunk.h
- file Error.h
- file StartChunk.h
- file WaveDataBuf.h
- file WaveFile.h
- file WaveFileCached.h
- file WaveFmt.h
- file WaveFmtChunk.h

8.7 AH/IO/Smart/ Directory Reference

Files

- file Buffer.h
- file Buffer2D.h
- file FilePtr.h

Chapter 9

libAHio Namespace Documentation

9.1 AH Namespace Reference

Classes

- class IOAudioFileOpenPolicy

  generic policy to open any audio file

- class IOAudioFileOpenPolicyFile

  policy to open a known file format.

- class IOAudioFileOpenPolicyMP3

  policy to open an MP3 file by converting it into WAV format

- class IOAudioFileOpenPolicyOGG

  policy to open an OGG/Vorbis file by converting it into WAV format

- class IOAudioFileOpenPolicyMPP

  policy to open an MP+ file by converting it into WAV format

- class IOAudioFileOpenPolicyLPAC

  policy to open an LPAC file by converting it into WAV format

- class IOAudioFileOpener

  opens audio file

- struct IOAudioFileReadPolicy

  empty template policy for reading any SampleType from audio file

- struct IOAudioFileReadPolicy< int32 >

  special policy for reading int32 data from audio file

- struct IOAudioFileReadPolicy< int16 >

  special policy for reading int16 data from audio file

- struct IOAudioFileReadPolicy< float >

  special policy for reading 32bit float data from audio file

- struct IOAudioFileReadPolicy< double >

  special policy for reading 64bit double data from audio file

- class IOAudioFileError

  error class for libsndfile errors

- class IOAudioFileInfo

  utility functions to interpret SF_INFO struct

- class IOAudioFile

  C++ interface to library libsndfile.

- class CddaSector
- class IOCFormat

  C-style printf compatible ostream applicator.

- class IOCmdOptHandler
- class IOCmdOption
- class IOConvert

  class containing converter functions

- class IODir

  directory handler

- class ErrorIO

  represents any Input/Output error

- class IOMidiDevice
- class IOMidiRawCmd
- class IOMidiSysExCmd
- class IOMpegFrameHeader

  class representing a MPEG Header

- class IOPlot

  use gnuplot *to plot data from within this process*

- class IOProfile
- class IORiffChunk

  class representing a RIFF chunk

- class ErrorIORiff

  unspecified Riff IO error

- class ErrorIORiffInvalidFormat
- class ErrorIORiffInvalidWaveFormat
- class IORiffStartChunk
- class IORiffWaveDataBuffer

buffer for RIFF wave data

- class IORiffWaveFile
- class IORiffWaveFileCached
- class IORiffWaveFmt
- class IORiffWaveFmtChunk
- class IOSmartBuffer

  smart buffer for any type `T`

- class IOSmartBuffer2D

  smart 2D buffer for any type `T`

- class IOSmartFilePtr

  smart `stdio` *file pointer*

- class IOTimeStamp

Functions

- ostream & operator<< (ostream &out, const IOCFormat &fmt)

  allow to insert IOCFormat objects directly into an ostream

9.1.1 Function Documentation

9.1.1.1 ostream& AH::operator<< (ostream & *out*, const IOCFormat & *fmt*) [inline]

allow to insert IOCFormat objects directly into an ostream

9.2 std Namespace Reference

Chapter 10

libAHio Class Documentation

10.1 AH::CddaSector Class Reference

`#include <Sector.h>`

Public Member Functions

- CddaSector ()
- CddaSector (double time)
- CddaSector (uint32 bytes)
- CddaSector (const std::string &s)
- CddaSector (uint32 minu, uint32 seco, uint32 sect)
- CddaSector (const CddaSector &cs)
- std::string asString () const
- uint32 asBytes () const
- double asTime () const
- CddaSector & operator= (const CddaSector &cs)
- CddaSector & operator+= (const CddaSector &cs)
- CddaSector & operator-= (const CddaSector &cs)
- CddaSector & operator *= (double k)
- CddaSector & operator/= (double k)

Static Public Attributes

- static const uint32 BYTES_PER_SECOND = 44100 * 4
- static const uint32 BYTES_PER_MINUTE = 44100 * 4 * 60
- static const uint32 BYTES_PER_SECTOR = 2352
- static const uint32 SECTORS_PER_SECOND = 75

Friends

- CddaSector operator+ (const CddaSector &cs1, const CddaSector &cs2)
- CddaSector operator- (const CddaSector &cs1, const CddaSector &cs2)
- CddaSector operator * (const CddaSector &cs, double k)

- CddaSector operator * (double k, const CddaSector &cs)
- CddaSector operator/ (const CddaSector &cs, double k)
- bool operator== (const CddaSector &cs1, const CddaSector &cs2)
- bool operator!= (const CddaSector &cs1, const CddaSector &cs2)
- bool operator< (const CddaSector &cs1, const CddaSector &cs2)
- bool operator> (const CddaSector &cs1, const CddaSector &cs2)

10.1.1 Detailed Description

represents a CDDA sector value

`AH::CddaSector` represents a CDDA sector value.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 AH::CddaSector::CddaSector () `[inline]`

default constructor.

10.1.2.2 AH::CddaSector::CddaSector (double *time*)

convert time value into CddaSector.

10.1.2.3 AH::CddaSector::CddaSector (uint32 *bytes*)

convert byte value into CddaSector.

10.1.2.4 AH::CddaSector::CddaSector (const std::string & *s*)

convert string into CddaSector.

10.1.2.5 AH::CddaSector::CddaSector (uint32 *minu*, uint32 *seco*, uint32 *sect*) `[inline]`

raw constructor.

10.1.2.6 AH::CddaSector::CddaSector (const CddaSector & *cs*) `[inline]`

copy constructor.

10.1.3 Member Function Documentation

10.1.3.1 uint32 AH::CddaSector::asBytes () const

return value as bytes.

10.1.3.2 std::string AH::CddaSector::asString () const

return value as string.

10.1.3.3 double AH::CddaSector::asTime () const

return value as time (seconds).

10.1.3.4 CddaSector& AH::CddaSector::operator *= (double *k*)

overloaded operator * for multiplication of CddaSector with double.

10.1.3.5 CddaSector& AH::CddaSector::operator+= (const CddaSector & *cs*)

overloaded operator += for addition of two CDDA sector values.

10.1.3.6 CddaSector& AH::CddaSector::operator-= (const CddaSector & *cs*)

overloaded operator -= for addition of two CDDA sector values.

10.1.3.7 CddaSector& AH::CddaSector::operator/= (double *k*)

overloaded operator / for division of CddaSector by double.

10.1.3.8 CddaSector& AH::CddaSector::operator= (const CddaSector & *cs*)

overloaded operator =

10.1.4 Friends And Related Function Documentation

10.1.4.1 CddaSector operator * (double *k*, const CddaSector & *cs*) [friend]

overloaded operator * for multiplication of double with CddaSector.

10.1.4.2 CddaSector operator * (const CddaSector & *cs*, double *k*) [friend]

overloaded operator * for multiplication of CddaSector with double.

10.1.4.3 bool operator!= (const CddaSector & *cs1*, const CddaSector & *cs2*) [friend]

overloaded operator != for comparison of two CddaSectors.

10.1.4.4 CddaSector operator+ (const CddaSector & *cs1*, const CddaSector & *cs2*) [friend]

overloaded operator + for addition of two CDDA sector values.

10.1.4.5 CddaSector operator- (const CddaSector & *cs1*, const CddaSector & *cs2*) [friend]

overloaded operator - for subtraction of two CDDA sector values.

10.1.4.6 CddaSector operator/ (const CddaSector & *cs*, double *k*) [friend]

overloaded operator / for division of CddaSector by double.

10.1.4.7 bool operator< (const CddaSector & *cs1*, const CddaSector & *cs2*) [friend]

overloaded operator < for comparison of two CddaSectors.

10.1.4.8 bool operator== (const CddaSector & *cs1*, const CddaSector & *cs2*) [friend]

overloaded operator == for comparison of two CddaSectors.

10.1.4.9 bool operator> (const CddaSector & *cs1*, const CddaSector & *cs2*) [friend]

overloaded operator > for comparison of two CddaSectors.

10.1.5 Member Data Documentation

10.1.5.1 const uint32 AH::CddaSector::BYTES_PER_MINUTE = 44100 * 4 * 60 [static]

10.1.5.2 const uint32 AH::CddaSector::BYTES_PER_SECOND = 44100 * 4 [static]

10.1.5.3 const uint32 AH::CddaSector::BYTES_PER_SECTOR = 2352 [static]

10.1.5.4 const uint32 AH::CddaSector::SECTORS_PER_SECOND = 75 [static]

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

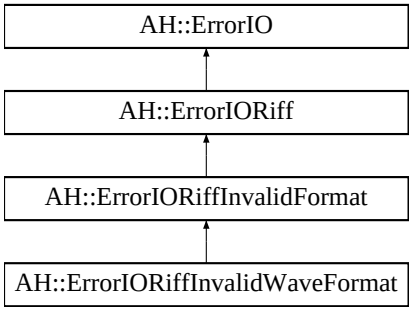
- AH/IO/Cdda/Sector.h

10.2 AH::ErrorIO Class Reference

represents any Input/Output error

`#include <Error.h>`

Inheritance diagram for AH::ErrorIO::



Public Member Functions

- ErrorIO ()
  default constructor
- ErrorIO (const string &m)
  constructor
- ErrorIO (const ErrorIO &src)
  copy constructor
- const ErrorIO & operator= (const ErrorIO &src)
  assignment operator:
- virtual ~ErrorIO () throw ()
  destructor

10.2.1 Detailed Description

represents any Input/Output error

`AH::ErrorIO` represents any Input/Output error.

Please include *AH/.h*.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 AH::ErrorIO::ErrorIO () `[inline]`

default constructor

10.2.2.2 AH::ErrorIO::ErrorIO (const string & *m*) [inline]

constructor

10.2.2.3 AH::ErrorIO::ErrorIO (const ErrorIO & *src*) [inline]

copy constructor

10.2.2.4 virtual AH::ErrorIO::~ErrorIO () throw () [inline, virtual]

destructor

10.2.3 Member Function Documentation

10.2.3.1 const ErrorIO& AH::ErrorIO::operator= (const ErrorIO & *src*) [inline]

assignment operator:

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

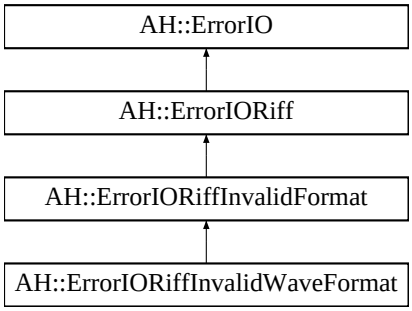
- AH/IO/Error.h

10.3 AH::ErrorIORiff Class Reference

unspecified Riff IO error

`#include <Error.h>`

Inheritance diagram for AH::ErrorIORiff::



Public Member Functions

- ErrorIORiff ()
  default constructor
- ErrorIORiff (const string &m)
  constructor
- ErrorIORiff (const ErrorIORiff &src)
  copy constructor
- const ErrorIORiff & operator= (const ErrorIORiff &src)
  assignment operator:
- virtual ~ErrorIORiff () throw ()
  destructor

10.3.1 Detailed Description

unspecified Riff IO error

10.3.2 Constructor & Destructor Documentation

10.3.2.1 AH::ErrorIORiff::ErrorIORiff () `[inline]`

default constructor

10.3.2.2 AH::ErrorIORiff::ErrorIORiff (const string & *m*) `[inline]`

constructor

10.3.2.3 AH::ErrorIORiff::ErrorIORiff (const ErrorIORiff & *src*) [inline]

copy constructor

10.3.2.4 virtual AH::ErrorIORiff::~ErrorIORiff () throw () [inline, virtual]

destructor

10.3.3 Member Function Documentation

10.3.3.1 const ErrorIORiff& AH::ErrorIORiff::operator= (const ErrorIORiff & *src*) [inline]

assignment operator:

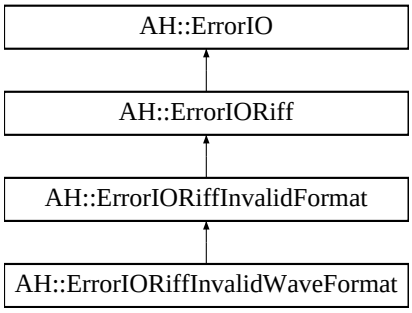
The documentation for this class was generated from the following file:

- AH/IO/Riff/Error.h

10.4 AH::ErrorIORiffInvalidFormat Class Reference

`#include <Error.h>`

Inheritance diagram for AH::ErrorIORiffInvalidFormat::



Public Member Functions

- ErrorIORiffInvalidFormat ()
  default constructor
- ErrorIORiffInvalidFormat (const string &m)
  constructor
- ErrorIORiffInvalidFormat (const ErrorIORiffInvalidFormat &src)
  copy constructor
- const ErrorIORiffInvalidFormat & operator= (const ErrorIORiffInvalidFormat &src)
  assignment operator:
- virtual ~ErrorIORiffInvalidFormat () throw ()
  destructor

10.4.1 Constructor & Destructor Documentation

10.4.1.1 AH::ErrorIORiffInvalidFormat::ErrorIORiffInvalidFormat () `[inline]`

default constructor

10.4.1.2 AH::ErrorIORiffInvalidFormat::ErrorIORiffInvalidFormat (const string & *m*) `[inline]`

constructor

10.4.1.3 AH::ErrorIORiffInvalidFormat::ErrorIORiffInvalidFormat (const ErrorIORiffInvalidFormat & *src*) `[inline]`

copy constructor

10.4.1.4 virtual AH::ErrorIORiffInvalidFormat::~ErrorIORiffInvalidFormat () throw () [inline, virtual]

destructor

10.4.2 Member Function Documentation

10.4.2.1 const ErrorIORiffInvalidFormat& AH::ErrorIORiffInvalidFormat::operator= (const ErrorIORiffInvalidFormat & *src*) [inline]

assignment operator:

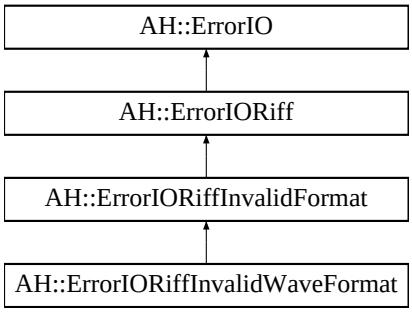
The documentation for this class was generated from the following file:

- AH/IO/Riff/Error.h

10.5 AH::ErrorIORiffInvalidWaveFormat Class Reference

`#include <Error.h>`

Inheritance diagram for AH::ErrorIORiffInvalidWaveFormat::



Public Member Functions

- ErrorIORiffInvalidWaveFormat ()
  default constructor
- ErrorIORiffInvalidWaveFormat (const char *err)
  constructor
- ErrorIORiffInvalidWaveFormat (const ErrorIORiffInvalidWaveFormat &src)
  copy constructor
- const ErrorIORiffInvalidWaveFormat & operator= (const ErrorIORiffInvalidWaveFormat &src)
  assignment operator:
- virtual ~ErrorIORiffInvalidWaveFormat () throw ()
  destructor

10.5.1 Constructor & Destructor Documentation

10.5.1.1 AH::ErrorIORiffInvalidWaveFormat::ErrorIORiffInvalidWaveFormat () `[inline]`

default constructor

10.5.1.2 AH::ErrorIORiffInvalidWaveFormat::ErrorIORiffInvalidWaveFormat (const char * *err*) `[inline]`

constructor

10.5.1.3 AH::ErrorIORiffInvalidWaveFormat::ErrorIORiffInvalidWaveFormat (const ErrorIORiffInvalidWaveFormat & *src*) `[inline]`

copy constructor

10.5.1.4 virtual AH::ErrorIORiffInvalidWaveFormat::~ErrorIORiffInvalidWaveFormat () throw () [inline, virtual]

destructor

10.5.2 Member Function Documentation

10.5.2.1 const ErrorIORiffInvalidWaveFormat& AH::ErrorIORiffInvalidWaveFormat::operator= (const ErrorIORiffInvalidWaveFormat & *src*) [inline]

assignment operator:

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

- AH/IO/Riff/Error.h

10.6 AH::IOAudioFile< SampleType > Class Template Reference

C++ interface to library libsndfile.

`#include <AudioFile.h>`

Public Member Functions

- IOAudioFile (const std::string &name, IOAudioFileOpener &afo, int verbose=0)
  constructor for input audio file including compressed formats
- IOAudioFile (const std::string &name, int verbose=0)
  constructor for input audio file
- virtual ~IOAudioFile ()
  destructor
- int getNumSamples () const
  get number of samples
- int getNumChannels () const
  get number of channels
- int getSampleRate () const
  get sample rate in Hz
- double getTime () const
  get runtime in seconds
- double getTime (uint32 frameNum) const
  get runtime of specified frames in seconds
- int32 readFrame (SampleType *frame)
  read one frame (1 sample per channel) from audio file
- int32 readFrame (std::vector< SampleType > &frame)
  read one frame (1 sample per channel) from audio file
- int getSampleBits () const
  get minimum number of sample bits of the audio file
- std::string getInfo () const
  print information about the audio file

Protected Member Functions

- void attachToHandle ()

Protected Attributes

- const std::string fileName
- int verbose

Classes

- class **SfCache**

10.6.1 Detailed Description

template<typename SampleType> class AH::IOAudioFile< SampleType >

C++ interface to library libsndfile.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 template<typename SampleType> AH::IOAudioFile< SampleType >::IOAudioFile (const std::string & *name*, IOAudioFileOpener & *afo*, int *verbose* = 0) [inline]

constructor for input audio file including compressed formats

Parameters:

name name of output file

afo audio file opener

verbose verbose level

10.6.2.2 template<typename SampleType> AH::IOAudioFile< SampleType >::IOAudioFile (const std::string & *name*, int *verbose* = 0) [inline]

constructor for input audio file

Parameters:

name name of output file

verbose verbose level

10.6.2.3 template<typename SampleType> virtual AH::IOAudioFile< SampleType >::~IOAudioFile () [inline, virtual]

destructor

10.6.3 Member Function Documentation

10.6.3.1 template<typename SampleType> void AH::IOAudioFile< SampleType >::attachToHandle () [inline, protected]

10.6.3.2 template<typename SampleType> std::string AH::IOAudioFile< SampleType >::getInfo () const [inline]

print information about the audio file

10.6.3.3 template<typename SampleType> int AH::IOAudioFile< SampleType >::getNumChannels () const [inline]

get number of channels

10.6.3.4 template<typename SampleType> int AH::IOAudioFile< SampleType >::getNumSamples () const [inline]

get number of samples

10.6.3.5 template<typename SampleType> int AH::IOAudioFile< SampleType >::getSampleBits () const [inline]

get minimum number of sample bits of the audio file

Returns:

positive number: number of integer bits
negative number: number of floating point integer bits
0: number of bits is unknown

10.6.3.6 template<typename SampleType> int AH::IOAudioFile< SampleType >::getSampleRate () const [inline]

get sample rate in Hz

10.6.3.7 template<typename SampleType> double AH::IOAudioFile< SampleType >::getTime (uint32 *frameNum*) const [inline]

get runtime of specified frames in seconds

10.6.3.8 template<typename SampleType> double AH::IOAudioFile< SampleType >::getTime () const [inline]

get runtime in seconds

10.6.3.9 template<typename SampleType> int32 AH::IOAudioFile< SampleType >::readFrame (std::vector< SampleType > & *frame*) [inline]

read one frame (1 sample per channel) from audio file

`readFrame()` reads one sample of each channel into the `frame` vector. The frame vector is not cleared in advance.

Parameters:
frame frame vector

10.6.3.10 template<typename SampleType> int32 AH::IOAudioFile< SampleType >::readFrame (SampleType * *frame*) [inline]

read one frame (1 sample per channel) from audio file

`readFrame()` reads one sample of each channel into the `frame` buffer. Therefore the `frame` buffer must be big enough to contain `getNumChannels()` samples.

Parameters:
frame pointer to frame buffer

10.6.4 Member Data Documentation

10.6.4.1 template<typename SampleType> const std::string AH::IOAudioFile< SampleType >::fileName [protected]

10.6.4.2 template<typename SampleType> int AH::IOAudioFile< SampleType >::verbose [protected]

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

- AH/IO/AudioFile.h

10.7 AH::IOAudioFileError Class Reference

error class for libsndfile errors

`#include <AudioFile.h>`

Public Member Functions

- IOAudioFileError (SNDFILE *sfp)
  constructor: extract error fom libsndfile handle
- IOAudioFileError (int errcode)
  constructor: extract error from error code

10.7.1 Detailed Description

error class for libsndfile errors

10.7.2 Constructor & Destructor Documentation

10.7.2.1 AH::IOAudioFileError::IOAudioFileError (SNDFILE * *sfp*) [inline]

constructor: extract error fom libsndfile handle

10.7.2.2 AH::IOAudioFileError::IOAudioFileError (int *errcode*) [inline]

constructor: extract error from error code

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

- AH/IO/AudioFile.h

10.8 AH::IOAudioFileInfo Class Reference

utility functions to interpret SF_INFO struct

`#include <AudioFile.h>`

Static Public Member Functions

- static std::string getSfInfo (const SF_INFO &info)
  return audio file information from SF_INFO struct as string
- static std::string getSfInfoFormat (int format)
  return format information from SF_INFO struct member 'format' as string
- static int getSampleBits (const SF_INFO &info)
  get minimum number of sample bits of the audio file

10.8.1 Detailed Description

utility functions to interpret SF_INFO struct

10.8.2 Member Function Documentation

10.8.2.1 static int AH::IOAudioFileInfo::getSampleBits (const SF_INFO & *info*) [static]

get minimum number of sample bits of the audio file

Returns:
positive number: number of integer bits
negative number: number of floating point integer bits
0: number of bits is unknown

10.8.2.2 static std::string AH::IOAudioFileInfo::getSfInfo (const SF_INFO & *info*) [static]

return audio file information from SF_INFO struct as string

10.8.2.3 static std::string AH::IOAudioFileInfo::getSfInfoFormat (int *format*) [static]

return format information from SF_INFO struct member 'format' as string

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

- AH/IO/AudioFile.h

10.9 AH::IOAudioFileOpener Class Reference

opens audio file

`#include <AudioFile.h>`

Public Types

- typedef std::vector< IOAudioFileOpenPolicy * > OpenerList

Public Member Functions

- IOAudioFileOpener (OpenerList &userOpeners)

  constructor

- IOAudioFileOpener ()

  default constructor

- FILE * openFile (const std::string &filename, std::string &ConverterName) const

  open file

10.9.1 Detailed Description

opens audio file

IOAudioFileOpener bundles builtin and user defined policies to open audio files.

See also:
IOAudioFileOpenPolicy

10.9.2 Member Typedef Documentation

10.9.2.1 typedef std::vector<IOAudioFileOpenPolicy*> AH::IOAudioFileOpener::OpenerList

10.9.3 Constructor & Destructor Documentation

10.9.3.1 AH::IOAudioFileOpener::IOAudioFileOpener (OpenerList & *userOpeners*) `[inline]`

constructor

Besides those file formats which are supported directly by libsndfile, IOAudioFileOpener is able to handle MP3, OGG/Vorbis, MP3+ and LPAC. The corresponding openers are built in. Use this constructor to insert additional user defined openers.

Parameters:
userOpeners list of user defined openers

10.9.3.2 AH::IOAudioFileOpener::IOAudioFileOpener () [inline]

default constructor

Besides those file formats which are supported directly by libsndfile, IOAudioFileOpener is able to handle MP3, OGG/Vorbis, MP3+ and LPAC. The corresponding openers are built in.

10.9.4 Member Function Documentation

10.9.4.1 FILE* AH::IOAudioFileOpener::openFile (const std::string & *filename*, std::string & *ConverterName*) const [inline]

open file

openFile() uses the first matching opener to open the specified file. As a last resort the normal file opener is used to open the file directly.

Returns:
pointer to open file
NULL in case of error

Parameters:
filename name of file

ConverterName output: set to name of conversion program used to open file

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

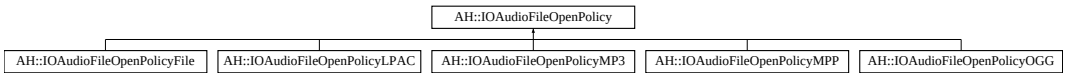
- AH/IO/AudioFile.h

10.10 AH::IOAudioFileOpenPolicy Class Reference

generic policy to open any audio file

`#include <AudioFile.h>`

Inheritance diagram for AH::IOAudioFileOpenPolicy::



Public Member Functions

- IOAudioFileOpenPolicy (const std::string &ConverterName, const std::string &Extension)

  constructor

- virtual ~IOAudioFileOpenPolicy ()

  destructor

- FILE * openFile (const std::string &filename, std::string &ConverterName) const

  open the file

- virtual bool matchesExtension (const std::string &extension) const

  return true if given extension matches any of the extensions of the conversion program

- std::string getConverterName () const

  return name of conversion program

Protected Member Functions

- virtual FILE * openFile (const std::string &filename) const =0

Protected Attributes

- std::string converterName
- std::vector< std::string > extensions

10.10.1 Detailed Description

generic policy to open any audio file

IOAudioFileOpenPolicy uses a conversion program to convert an audio file of a format which cannot be handled by libsndfile, into a format that can be handled (e.g. into uncompressed WAV). Since this is a pure virtual class you must derive a subclass to be able to use it.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 AH::IOAudioFileOpenPolicy::IOAudioFileOpenPolicy (const std::string & *ConverterName*, const std::string & *Extension*) [inline]

constructor

Parameters:

ConverterName conversion program

Extension extension of audio file

10.10.2.2 virtual AH::IOAudioFileOpenPolicy::~IOAudioFileOpenPolicy () [inline, virtual]

destructor

10.10.3 Member Function Documentation

10.10.3.1 std::string AH::IOAudioFileOpenPolicy::getConverterName () const [inline]

return name of conversion program

10.10.3.2 virtual bool AH::IOAudioFileOpenPolicy::matchesExtension (const std::string & *extension*) const [inline, virtual]

return true if given extension matches any of the extensions of the conversion program

matchesExtension() compares the given extension with all supported extensions.

Returns:

true if given extension matches any of the supported extensions
true if any of the supported extensions is empty
false in all other cases

10.10.3.3 virtual FILE* AH::IOAudioFileOpenPolicy::openFile (const std::string & *filename*) const [protected, pure virtual]

Implemented in AH::IOAudioFileOpenPolicyFile, AH::IOAudioFileOpenPolicyMP3, AH::IOAudioFileOpenPolicyOGG, AH::IOAudioFileOpenPolicyMPP, and AH::IOAudioFileOpenPolicyLPAC.

10.10.3.4 FILE* AH::IOAudioFileOpenPolicy::openFile (const std::string & *filename*, std::string & *ConverterName*) const [inline]

open the file

Returns:

pointer to open file
NULL in case of error

Parameters:

filename name of file

ConverterName output: set to name of conversion program used to open file

10.10.4 Member Data Documentation

10.10.4.1 std::string AH::IOAudioFileOpenPolicy::converterName [protected]

10.10.4.2 std::vector<std::string> AH::IOAudioFileOpenPolicy::extensions [protected]

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

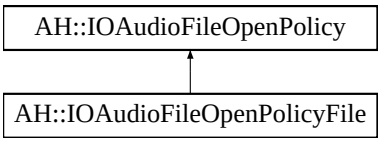
- AH/IO/AudioFile.h

10.11 AH::IOAudioFileOpenPolicyFile Class Reference

policy to open a known file format.

`#include <AudioFile.h>`

Inheritance diagram for AH::IOAudioFileOpenPolicyFile::



Public Member Functions

- IOAudioFileOpenPolicyFile ()
  default constructor
- virtual FILE * openFile (const std::string &filename) const
  open the file

10.11.1 Detailed Description

policy to open a known file format.

IOAudioFileOpenPolicyFile opens the file directly without conversion.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 AH::IOAudioFileOpenPolicyFile::IOAudioFileOpenPolicyFile () `[inline]`

default constructor

10.11.3 Member Function Documentation

10.11.3.1 virtual FILE* AH::IOAudioFileOpenPolicyFile::openFile (const std::string & *filename*) const `[inline, virtual]`

open the file

Implements AH::IOAudioFileOpenPolicy.

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

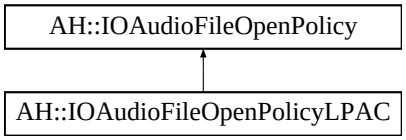
- AH/IO/AudioFile.h

10.12 AH::IOAudioFileOpenPolicyLPAC Class Reference

policy to open an LPAC file by converting it into WAV format

`#include <AudioFile.h>`

Inheritance diagram for AH::IOAudioFileOpenPolicyLPAC::



Public Member Functions

- IOAudioFileOpenPolicyLPAC ()
  default constructor

Protected Member Functions

- virtual FILE * openFile (const std::string &filename) const
  open the file

10.12.1 Detailed Description

policy to open an LPAC file by converting it into WAV format

IOAudioFileOpenPolicyLPAC uses *lpac* to open lossless compressed LPAC files as WAV file.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 AH::IOAudioFileOpenPolicyLPAC::IOAudioFileOpenPolicyLPAC () `[inline]`

default constructor

10.12.3 Member Function Documentation

10.12.3.1 virtual FILE* AH::IOAudioFileOpenPolicyLPAC::openFile (const std::string & *filename*) const `[inline, protected, virtual]`

open the file

Implements AH::IOAudioFileOpenPolicy.

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

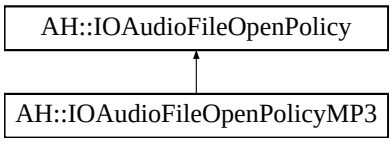
- AH/IO/AudioFile.h

10.13 AH::IOAudioFileOpenPolicyMP3 Class Reference

policy to open an MP3 file by converting it into WAV format

`#include <AudioFile.h>`

Inheritance diagram for AH::IOAudioFileOpenPolicyMP3::



Public Member Functions

- IOAudioFileOpenPolicyMP3 ()
  default constructor

Protected Member Functions

- virtual FILE * openFile (const std::string &filename) const
  open the file

10.13.1 Detailed Description

policy to open an MP3 file by converting it into WAV format

IOAudioFileOpenPolicyMP3 uses *mpg123* to open MP3 files as WAV file.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 AH::IOAudioFileOpenPolicyMP3::IOAudioFileOpenPolicyMP3 () [inline]

default constructor

10.13.3 Member Function Documentation

10.13.3.1 virtual FILE* AH::IOAudioFileOpenPolicyMP3::openFile (const std::string & *filename*) const [inline, protected, virtual]

open the file

Implements AH::IOAudioFileOpenPolicy.

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

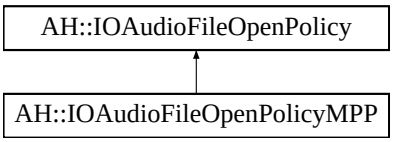
- AH/IO/AudioFile.h

10.14 AH::IOAudioFileOpenPolicyMPP Class Reference

policy to open an MP+ file by converting it into WAV format

`#include <AudioFile.h>`

Inheritance diagram for AH::IOAudioFileOpenPolicyMPP::



Public Member Functions

- IOAudioFileOpenPolicyMPP ()
  default constructor

Protected Member Functions

- virtual FILE * openFile (const std::string &filename) const
  open the file

10.14.1 Detailed Description

policy to open an MP+ file by converting it into WAV format

IOAudioFileOpenPolicyMPP uses *mppdec* to open MP3+ files as WAV file.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 AH::IOAudioFileOpenPolicyMPP::IOAudioFileOpenPolicyMPP () `[inline]`

default constructor

10.14.3 Member Function Documentation

10.14.3.1 virtual FILE* AH::IOAudioFileOpenPolicyMPP::openFile (const std::string & *filename*) const `[inline, protected, virtual]`

open the file

Implements AH::IOAudioFileOpenPolicy.

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

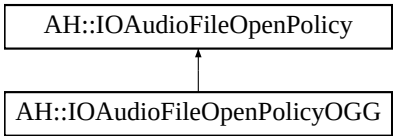
- AH/IO/AudioFile.h

10.15 AH::IOAudioFileOpenPolicyOGG Class Reference

policy to open an OGG/Vorbis file by converting it into WAV format

`#include <AudioFile.h>`

Inheritance diagram for AH::IOAudioFileOpenPolicyOGG::



Public Member Functions

- IOAudioFileOpenPolicyOGG ()
  default constructor

Protected Member Functions

- virtual FILE * openFile (const std::string &filename) const
  open the file

10.15.1 Detailed Description

policy to open an OGG/Vorbis file by converting it into WAV format

IOAudioFileOpenPolicyMP3 uses *oggdec* to open OGG/Vorbis files as WAV file.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 AH::IOAudioFileOpenPolicyOGG::IOAudioFileOpenPolicyOGG () `[inline]`

default constructor

10.15.3 Member Function Documentation

10.15.3.1 virtual FILE* AH::IOAudioFileOpenPolicyOGG::openFile (const std::string & *filename*) const `[inline, protected, virtual]`

open the file

Implements AH::IOAudioFileOpenPolicy.

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

- AH/IO/AudioFile.h

10.16 AH::IOAudioFileReadPolicy< SampleType > Struct Template Reference

empty template policy for reading any SampleType from audio file

`#include <AudioFile.h>`

10.16.1 Detailed Description

template<typename SampleType> struct AH::IOAudioFileReadPolicy< SampleType >

empty template policy for reading any SampleType from audio file

This generic template defines the interface to all usable policies. It is empty to force compilation errors if someone uses this policy with a type which is not supplied as partially specialized template.

This policy supplies the following functions:

- int readBuf(SNDFILE* sfp, SampleType* data, int items)

Andrei Alexandrescu's book "Modern C++ Design" explains this type of template policies in chapter 1 (Policy-Based Class Design).

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

- AH/IO/AudioFile.h

10.17 AH::IOAudioFileReadPolicy< double > Struct Template Reference

special policy for reading 64bit double data from audio file

`#include <AudioFile.h>`

Static Public Member Functions

- static int readBuf (SNDFILE *sfp, double *data, int items)
  read audio data into 64 bit double buffer

10.17.1 Detailed Description

template<> struct AH::IOAudioFileReadPolicy< double >

special policy for reading 64bit double data from audio file

10.17.2 Member Function Documentation

10.17.2.1 static int AH::IOAudioFileReadPolicy< double >::readBuf (SNDFILE * *sfp*, double * *data*, int *items*) [inline, static]

read audio data into 64 bit double buffer

Parameters:

sfp opaque pointer to libsndfile handle

data 64 bit double buffer

items number of 64 bit doubles in 'data' buffer

Returns:

number of 64 bit doubles read from file
0 in case of EOF

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

- AH/IO/AudioFile.h

10.18 AH::IOAudioFileReadPolicy< float > Struct Template Reference

special policy for reading 32bit float data from audio file

`#include <AudioFile.h>`

Static Public Member Functions

- static int readBuf (SNDFILE *sfp, float *data, int items)
  read audio data into 32 bit float buffer

10.18.1 Detailed Description

template<> struct AH::IOAudioFileReadPolicy< float >

special policy for reading 32bit float data from audio file

10.18.2 Member Function Documentation

10.18.2.1 static int AH::IOAudioFileReadPolicy< float >::readBuf (SNDFILE * *sfp*, float * *data*, int *items*) [inline, static]

read audio data into 32 bit float buffer

Parameters:

sfp opaque pointer to libsndfile handle

data 32 bit float buffer

items number of 32 bit floats in 'data' buffer

Returns:

number of 32 bit floats read from file
0 in case of EOF

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

- AH/IO/AudioFile.h

10.19 AH::IOAudioFileReadPolicy< int16 > Struct Template Reference

special policy for reading int16 data from audio file

`#include <AudioFile.h>`

Static Public Member Functions

- static int readBuf (SNDFILE *sfp, int16 *data, int items)
  read audio data into 16 bit integer buffer

10.19.1 Detailed Description

template<> struct AH::IOAudioFileReadPolicy< int16 >

special policy for reading int16 data from audio file

10.19.2 Member Function Documentation

10.19.2.1 static int AH::IOAudioFileReadPolicy< int16 >::readBuf (SNDFILE * *sfp*, int16 * *data*, int *items*) [inline, static]

read audio data into 16 bit integer buffer

Parameters:

sfp opaque pointer to libsndfile handle

data 16 bit integer buffer

items number of 16 bit integers in 'data' buffer

Returns:

number of 16 bit integers read from file
0 in case of EOF

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

- AH/IO/AudioFile.h

10.20 AH::IOAudioFileReadPolicy< int32 > Struct Template Reference

special policy for reading int32 data from audio file

`#include <AudioFile.h>`

Static Public Member Functions

- static int readBuf (SNDFILE *sfp, int32 *data, int items)
  read audio data into 32 bit integer buffer

10.20.1 Detailed Description

template<> struct AH::IOAudioFileReadPolicy< int32 >

special policy for reading int32 data from audio file

10.20.2 Member Function Documentation

10.20.2.1 static int AH::IOAudioFileReadPolicy< int32 >::readBuf (SNDFILE * *sfp*, int32 * *data*, int *items*) [inline, static]

read audio data into 32 bit integer buffer

Parameters:

sfp opaque pointer to libsndfile handle

data 32 bit integer buffer

items number of 32 bit integers in 'data' buffer

Returns:

number of 32 bit integers read from file
0 in case of EOF

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

- AH/IO/AudioFile.h

10.21 AH::IOCFormat Class Reference

C-style printf compatible ostream applicator.

`#include <CFormat.h>`

Public Member Functions

- IOCFormat (const string &format)
  constructor
- IOCFormat (const char *format=0)
  default constructor
- ostream & manip (ostream &out) const
  manipulate ostream

10.21.1 Detailed Description

C-style printf compatible ostream applicator.

`IOCFormat` is a C++ `ostream` manipulator which takes a `printf(3S)` format string as constructor argument and sets the `ostream` flags accordingly when its public member function `manip()` is applied on an `ostream`.

For convenience the `ostream` output operator is overloaded, so just putting an `IOCFormat` object (even an temporary one) into an output stream will be sufficient, see the following example:

```
#include <AH/IO/CFormat.h>

cout << IOCFormat("%5.2f") << 3.141592653 << endl;
```

A local `typedef` makes this even shorter to write and understand:

```
#include <AH/IO/CFormat.h>

typedef IOCFormat fmt;
cout << fmt("%5.2f") << 3.141592653 << endl;
```

If the same format is used often it is recommended to define it once and use it multiply, as then the format conversion is done only once (at construction time):

```
#include <AH/IO/CFormat.h>

IOCFormat fmt("%10.2f");
for (int i = 0; i < 1000; i++)
cout << fmt << (3.141592653 + i) << endl;
```

`IOCFormat` supports the following formats (refer to the `printf(3S)` man pages for explanations):

- flags: -, +, #, 0 (not on strings)

- width: a number specifying the field width
- precision: a decimal point followed by a number specifying the precision
- size: h, l, ll, and L are accepted but have no effect
- type: d, x, X, o, f, e, E, s

Please include {*AH/IO/CFormat*.h}

10.21.2 Constructor & Destructor Documentation

10.21.2.1 AH::IOCFormat::IOCFormat (const string & *format*)

constructor

10.21.2.2 AH::IOCFormat::IOCFormat (const char * *format* = 0)

default constructor

10.21.3 Member Function Documentation

10.21.3.1 ostream& AH::IOCFormat::manip (ostream & *out*) const

manipulate ostream

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

- AH/IO/CFormat.h

10.22 AH::IOCmdOptHandler Class Reference

`#include <CmdOptHandler.h>`

Public Member Functions

- IOCmdOptHandler ()
- IOCmdOptHandler (const std::vector< IOCmdOption * > newOpts)
- ~IOCmdOptHandler ()
- void addOption (IOCmdOption *newOpt)
- void addOptions (const std::vector< IOCmdOption * > newOpts)
- int handleArgs (int argc, char *argv[ ], std::vector< std::string > &remarg, std::vector< std::string > &errwarn)
- std::string getHelpText (const std::string &lineHead)
- std::string getShortOptionText () const

10.22.1 Constructor & Destructor Documentation

10.22.1.1 AH::IOCmdOptHandler::IOCmdOptHandler () `[inline]`

default constructor.

10.22.1.2 AH::IOCmdOptHandler::IOCmdOptHandler (const std::vector< IOCmdOption * > *newOpts*) `[inline]`

constructor.

10.22.1.3 AH::IOCmdOptHandler::~IOCmdOptHandler () `[inline]`

destructor

10.22.2 Member Function Documentation

10.22.2.1 void AH::IOCmdOptHandler::addOption (IOCmdOption * *newOpt*) `[inline]`

add one more option.

10.22.2.2 void AH::IOCmdOptHandler::addOptions (const std::vector< IOCmdOption * > *newOpts*)

add more options.

10.22.2.3 std::string AH::IOCmdOptHandler::getHelpText (const std::string & *lineHead*)

return help text of all options.

10.22.2.4 std::string AH::IOCmdOptHandler::getShortOptionText () const

return short option string, for help text.

10.22.2.5 int AH::IOCmdOptHandler::handleArgs (int *argc*, char * *argv*[ ], std::vector< std::string > & *remarg*, std::vector< std::string > & *errwarn*)

handle options, return number of bad options.

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

- AH/IO/CmdOptHandler.h

10.23 AH::IOCmdOption Class Reference

`#include <CmdOption.h>`

Public Types

- enum SecondArg { SecondArgNone, SecondArgAlways, SecondArgOptional }

Public Member Functions

- IOCmdOption (char short_opt, const char *long_opt, SecondArg sec, const std::string &desc)
- virtual ~IOCmdOption ()
- virtual void set ()
- virtual void setSecArg (const std::string &arg)
- int isSet () const
- std::vector< std::string > getSecArg () const
- std::string getLastSecArg () const
- bool isShort () const
- bool isLong () const
- SecondArg getSecArgMode () const
- char getShortOpt () const
- std::string getLongOpt () const
- std::string getDescription () const

Protected Attributes

- char shortOpt
- const char * longOpt
- SecondArg secArg
- std::string description

10.23.1 Detailed Description

`AH::IOCmdOption` ...

Please include *AH/IO/CmdOption.h*.

10.23.2 Member Enumeration Documentation

10.23.2.1 enum AH::IOCmdOption::SecondArg

Enumerator:

SecondArgNone no second argument.

SecondArgAlways needs a second argument.

SecondArgOptional may have a second argument.

10.23.3 Constructor & Destructor Documentation

10.23.3.1 AH::IOCmdOption::IOCmdOption (char *short_opt*, const char * *long_opt*, SecondArg *sec*, const std::string & *desc*) [inline]

constructor.

10.23.3.2 virtual AH::IOCmdOption::~IOCmdOption () [inline, virtual]

destructor

10.23.4 Member Function Documentation

10.23.4.1 std::string AH::IOCmdOption::getDescription () const [inline]

return description.

10.23.4.2 std::string AH::IOCmdOption::getLastSecArg () const [inline]

return last second argument.

10.23.4.3 std::string AH::IOCmdOption::getLongOpt () const [inline]

return long option name.

10.23.4.4 std::vector<std::string> AH::IOCmdOption::getSecArg () const [inline]

return second argument.

10.23.4.5 SecondArg AH::IOCmdOption::getSecArgMode () const [inline]

return second argument.

10.23.4.6 char AH::IOCmdOption::getShortOpt () const [inline]

return short option char.

10.23.4.7 bool AH::IOCmdOption::isLong () const [inline]

return true if it is a long option.

10.23.4.8 int AH::IOCmdOption::isSet () const [inline]

return true if option is set.

10.23.4.9 bool AH::IOCmdOption::isShort () const [inline]

return true if it is a short option.

10.23.4.10 virtual void AH::IOCmdOption::set () [inline, virtual]

set option to set.

10.23.4.11 virtual void AH::IOCmdOption::setSecArg (const std::string & *arg*) [inline, virtual]

set second argument.

10.23.5 Member Data Documentation

10.23.5.1 std::string AH::IOCmdOption::description [protected]

10.23.5.2 const char* AH::IOCmdOption::longOpt [protected]

10.23.5.3 SecondArg AH::IOCmdOption::secArg [protected]

10.23.5.4 char AH::IOCmdOption::shortOpt [protected]

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

- AH/IO/CmdOption.h

10.24 AH::IOConvert Class Reference

class containing converter functions

`#include <Convert.h>`

Static Public Member Functions

- static uint32 BigEndCharsToUint32 (const uint8 data[ ])
  convert big endian data into 32bit unsigned integer
- static void Uint32ToBigEndChars (uint32 val, uint8 data[ ])
  convert 32bit unsigned integer into big endian data
- static uint32 LittleEndCharsToUint32 (const uint8 data[ ])
  convert little endian data into 32bit unsigned integer
- static void Uint32ToLittleEndChars (uint32 val, uint8 data[ ])
  convert 32bit unsigned integer into little endian data
- static int32 BigEndCharsToInt32 (const uint8 data[ ])
  convert big endian data into 32bit signed integer
- static void Int32ToBigEndChars (int32 val, uint8 data[ ])
  convert 32bit signed integer into big endian data
- static int32 LittleEndCharsToInt32 (const uint8 data[ ])
  convert little endian data into 32bit signed integer
- static void Int32ToLittleEndChars (int32 val, uint8 data[ ])
  convert 32bit signed integer into little endian data
- static uint16 BigEndCharsToUint16 (const uint8 data[ ])
  convert big endian data into 16bit unsigned integer
- static void Uint16ToBigEndChars (uint16 val, uint8 data[ ])
  convert 16bit unsigned integer into big endian data
- static uint16 LittleEndCharsToUint16 (const uint8 data[ ])
  convert little endian data into 16bit unsigned integer
- static void Uint16ToLittleEndChars (uint16 val, uint8 data[ ])
  convert 16bit unsigned integer into little endian data
- static int16 BigEndCharsToInt16 (const uint8 data[ ])
  convert big endian data into 16bit signed integer
- static void Int16ToBigEndChars (int16 val, uint8 data[ ])
  convert 16bit signed integer into big endian data

- static int16 LittleEndCharsToInt16 (const uint8 data[ ])

  convert little endian data into 16bit signed integer

- static void Int16ToLittleEndChars (int16 val, uint8 data[ ])

  convert 16bit signed integer into little endian data

- static string timeToString (unsigned int value, const char *unit="s")

  convert time value into readable string

- static string timeToString (double value, const char *unit="s")

  convert time value into readable string

- static string timeToCdString (double value)

  convert time value into readable string in CD format

10.24.1 Detailed Description

class containing converter functions

`IOConvert` contains some usable converter functions.

Please include *AH/IO/Convert.h*

10.24.2 Member Function Documentation

10.24.2.1 static int16 AH::IOConvert::BigEndCharsToInt16 (const uint8 *data*[ ]) [static]

convert big endian data into 16bit signed integer

Use `BigEndCharsToInt16()` to convert big endian data into a 16bit wide unsigned integer.

Please include *AH/IO/Convert.h*

10.24.2.2 static int32 AH::IOConvert::BigEndCharsToInt32 (const uint8 *data*[ ]) [static]

convert big endian data into 32bit signed integer

Use `BigEndCharsToInt32()` to convert big endian data into a 32bit wide unsigned integer.

Please include *AH/IO/Convert.h*

10.24.2.3 static uint16 AH::IOConvert::BigEndCharsToUint16 (const uint8 *data*[ ]) [static]

convert big endian data into 16bit unsigned integer

Use `BigEndCharsToUint16()` to convert big endian data into a 16bit wide unsigned integer.

Please include *AH/IO/Convert.h*

10.24.2.4 static uint32 AH::IOConvert::BigEndCharsToUint32 (const uint8 *data*[ ]) [static]

convert big endian data into 32bit unsigned integer

Use `BigEndCharsToUint32()` to convert big endian data into a 32bit wide unsigned integer.

Please include *AH/IO/Convert.h*

10.24.2.5 static void AH::IOConvert::Int16ToBigEndChars (int16 *val*, uint8 *data*[ ]) [static]

convert 16bit signed integer into big endian data

Use `Int16ToBigEndChars()` to convert a 16bit wide signed integer into big endian data.

Please include *AH/IO/Convert.h*

10.24.2.6 static void AH::IOConvert::Int16ToLittleEndChars (int16 *val*, uint8 *data*[ ]) [static]

convert 16bit signed integer into little endian data

Use `Int16ToLittleEndChars()` to convert a 16bit wide signed integer into little endian data.

Please include *AH/IO/Convert.h*

10.24.2.7 static void AH::IOConvert::Int32ToBigEndChars (int32 *val*, uint8 *data*[ ]) [static]

convert 32bit signed integer into big endian data

Use `Int32ToBigEndChars()` to convert a 32bit wide signed integer into big endian data.

Please include *AH/IO/Convert.h*

10.24.2.8 static void AH::IOConvert::Int32ToLittleEndChars (int32 *val*, uint8 *data*[ ]) [static]

convert 32bit signed integer into little endian data

Use `Int32ToLittleEndChars()` to convert a 32bit wide signed integer into little endian data.

Please include *AH/IO/Convert.h*

10.24.2.9 static int16 AH::IOConvert::LittleEndCharsToInt16 (const uint8 *data*[ ]) [static]

convert little endian data into 16bit signed integer

Use `LittleEndCharsToInt16()` to convert little endian data into a 16bit wide signed integer.

Please include *AH/IO/Convert.h*

10.24.2.10 static int32 AH::IOConvert::LittleEndCharsToInt32 (const uint8 *data*[ ]) [static]

convert little endian data into 32bit signed integer

Use `LittleEndCharsToInt32()` to convert little endian data into a 32bit wide signed integer.

Please include *AH/IO/Convert.h*

10.24.2.11 static uint16 AH::IOConvert::LittleEndCharsToUint16 (const uint8 *data*[ ]) [static]

convert little endian data into 16bit unsigned integer

Use `LittleEndCharsToUint16()` to convert little endian data into a 16bit wide unsigned integer.

Please include *AH/IO/Convert.h*

10.24.2.12 static uint32 AH::IOConvert::LittleEndCharsToUint32 (const uint8 *data*[ ]) [static]

convert little endian data into 32bit unsigned integer

Use `LittleEndCharsToUint32()` to convert little endian data into a 32bit wide unsigned integer.

Please include *AH/IO/Convert.h*

10.24.2.13 static string AH::IOConvert::timeToCdString (double *value*) [static]

convert time value into readable string in CD format

Use `timeToCdString` to convert a time value into a readable string.

Please include *AH/IO/TimeFormat.h*

Parameters:

value time value (floating point)

10.24.2.14 static string AH::IOConvert::timeToString (double *value*, const char * *unit* = "s") [static]

convert time value into readable string

Use `timeToString` to convert a time value into a readable string. Currently only seconds are supported as time unit.

Please include *AH/IO/TimeFormat.h*

Parameters:

value time value (floating point)

unit time unit (ps, ns, us, ms, s, m, d, h, M, Y: default is s)

10.24.2.15 static string AH::IOConvert::timeToString (unsigned int *value*, const char * *unit* = "s") [static]

convert time value into readable string

Use `timeToString` to convert a time value into a readable string. Currently only seconds are supported as time unit.

Please include *AH/IO/TimeFormat.h*

Parameters:

value time value (integer)

unit time unit (ps, ns, us, ms, s, m, d, h, M, Y: default is s)

10.24.2.16 static void AH::IOConvert::Uint16ToBigEndChars (uint16 *val*, uint8 *data*[ ]) [static]

convert 16bit unsigned integer into big endian data

Use `Uint16ToBigEndChars()` to convert a 16bit wide unsigned integer into big endian data.

Please include *AH/IO/Convert.h*

10.24.2.17 static void AH::IOConvert::Uint16ToLittleEndChars (uint16 *val*, uint8 *data*[ ]) [static]

convert 16bit unsigned integer into little endian data

Use `Uint16ToLittleEndChars()` to convert a 16bit wide unsigned integer into little endian data.

Please include *AH/IO/Convert.h*

10.24.2.18 static void AH::IOConvert::Uint32ToBigEndChars (uint32 *val*, uint8 *data*[ ]) [static]

convert 32bit unsigned integer into big endian data

Use `Uint32ToBigEndChars()` to convert a 32bit wide unsigned integer into big endian data.

Please include *AH/IO/Convert.h*

10.24.2.19 static void AH::IOConvert::Uint32ToLittleEndChars (uint32 *val*, uint8 *data*[ ]) [static]

convert 32bit unsigned integer into little endian data

Use `Uint32ToLittleEndChars()` to convert a 32bit wide unsigned integer into little endian data.

Please include *AH/IO/Convert.h*

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

- AH/IO/Convert.h

10.25 AH::IODir Class Reference

directory handler

`#include <Dir.h>`

Public Member Functions

- IODir ()
  constructor
- ~IODir ()
  destructor

Static Public Member Functions

- static int getList (const string &path, vector< string > &list)
  get list of directory entries

10.25.1 Detailed Description

directory handler

10.25.2 Constructor & Destructor Documentation

10.25.2.1 AH::IODir::IODir () [inline]

constructor

10.25.2.2 AH::IODir::~IODir () [inline]

destructor

10.25.3 Member Function Documentation

10.25.3.1 static int AH::IODir::getList (const string & *path*, vector< string > & *list*) [static]

get list of directory entries

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

- AH/IO/Dir.h

10.26 AH::IOMidiDevice Class Reference

`#include <Device.h>`

Public Member Functions

- IOMidiDevice ()
  constructor for MIDI device
- virtual ~IOMidiDevice ()
  destructor
- void openDev ()
  open MIDI device
- void closeDev ()
  close MIDI device
- void execute (IOMidiRawCmd &cmd)
  execute command

10.26.1 Constructor & Destructor Documentation

10.26.1.1 AH::IOMidiDevice::IOMidiDevice ()

constructor for MIDI device

10.26.1.2 virtual AH::IOMidiDevice::~IOMidiDevice () [virtual]

destructor

10.26.2 Member Function Documentation

10.26.2.1 void AH::IOMidiDevice::closeDev ()

close MIDI device

10.26.2.2 void AH::IOMidiDevice::execute (IOMidiRawCmd & *cmd*)

execute command

10.26.2.3 void AH::IOMidiDevice::openDev ()

open MIDI device

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

- AH/IO/Midi/Device.h

10.27 AH::IOMidiRawCmd Class Reference

`#include <RawCmd.h>`

Public Member Functions

- IOMidiRawCmd (const std::vector< uint8 > &sendData, int32 recSize=-1)
  constructor for raw MIDI command
- IOMidiRawCmd (const uint *sendData, int32 sendSize, int32 recSize=-1)
  constructor for raw MIDI command
- IOMidiRawCmd (int32 recSize=-1)
  constructor for empty raw MIDI command
- virtual ~IOMidiRawCmd ()
  destructor
- const std::vector< uint8 > & getSendData () const
  get send data
- uint32 getSendDataSize () const
  get size of send data
- const std::vector< uint8 > & getRecData () const
  get received data
- uint32 getRecDataSize () const
  get size of received data
- int32 getWishRecDataSize () const
  get desired size of receive data
- std::string getSendDataHex () const
  get send data as hex string (for debug)
- std::string getRecDataHex () const
  get received data as hex string (for debug)
- virtual void pushSendByte (uint8 byte)
- virtual void pushRecByte (uint8 byte)

Protected Attributes

- std::vector< uint8 > txData
- std::vector< uint8 > rxData
- int32 rxSize

10.27.1 Constructor & Destructor Documentation

10.27.1.1 AH::IOMidiRawCmd::IOMidiRawCmd (const std::vector< uint8 > & *sendData*, int32 *recSize* = -1) [inline]

constructor for raw MIDI command

10.27.1.2 AH::IOMidiRawCmd::IOMidiRawCmd (const uint * *sendData*, int32 *sendSize*, int32 *recSize* = -1)

constructor for raw MIDI command

10.27.1.3 AH::IOMidiRawCmd::IOMidiRawCmd (int32 *recSize* = -1) [inline]

constructor for empty raw MIDI command

10.27.1.4 virtual AH::IOMidiRawCmd::~IOMidiRawCmd () [inline, virtual]

destructor

10.27.2 Member Function Documentation

10.27.2.1 const std::vector<uint8>& AH::IOMidiRawCmd::getRecData () const [inline]

get received data

10.27.2.2 std::string AH::IOMidiRawCmd::getRecDataHex () const [inline]

get received data as hex string (for debug)

10.27.2.3 uint32 AH::IOMidiRawCmd::getRecDataSize () const [inline]

get size of received data

10.27.2.4 const std::vector<uint8>& AH::IOMidiRawCmd::getSendData () const [inline]

get send data

10.27.2.5 std::string AH::IOMidiRawCmd::getSendDataHex () const [inline]

get send data as hex string (for debug)

10.27.2.6 uint32 AH::IOMidiRawCmd::getSendDataSize () const [inline]

get size of send data

10.27.2.7 int32 AH::IOMidiRawCmd::getWishRecDataSize () const [inline]

get desired size of receive data

10.27.2.8 virtual void AH::IOMidiRawCmd::pushRecByte (uint8 *byte*) [inline, virtual]

10.27.2.9 virtual void AH::IOMidiRawCmd::pushSendByte (uint8 *byte*) [inline, virtual]

10.27.3 Member Data Documentation

10.27.3.1 std::vector<uint8> AH::IOMidiRawCmd::rxData [protected]

10.27.3.2 int32 AH::IOMidiRawCmd::rxSize [protected]

10.27.3.3 std::vector<uint8> AH::IOMidiRawCmd::txData [protected]

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

- AH/IO/Midi/RawCmd.h

10.28 AH::IOMidiSysExCmd Class Reference

`#include <SysExCmd.h>`

Public Member Functions

- IOMidiSysExCmd (const std::vector< uint8 > &sendData, int32 recSize=-1)
  constructor for System Exclusive MIDI command
- IOMidiSysExCmd (int32 recSize=-1)
  constructor for empty System Exclusive MIDI command
- virtual ~IOMidiSysExCmd ()
  destructor
- virtual void pushSendByte (uint8 byte)
- virtual void pushRecByte (uint8 byte)

10.28.1 Constructor & Destructor Documentation

10.28.1.1 AH::IOMidiSysExCmd::IOMidiSysExCmd (const std::vector< uint8 > & *sendData*, int32 *recSize* = -1)

constructor for System Exclusive MIDI command

10.28.1.2 AH::IOMidiSysExCmd::IOMidiSysExCmd (int32 *recSize* = -1)

constructor for empty System Exclusive MIDI command

10.28.1.3 virtual AH::IOMidiSysExCmd::~IOMidiSysExCmd () [virtual]

destructor

10.28.2 Member Function Documentation

10.28.2.1 virtual void AH::IOMidiSysExCmd::pushRecByte (uint8 *byte*) [inline, virtual]

10.28.2.2 virtual void AH::IOMidiSysExCmd::pushSendByte (uint8 *byte*) [inline, virtual]

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

- AH/IO/Midi/SysExCmd.h

10.29 AH::IOMpegFrameHeader Class Reference

class representing a MPEG Header

`#include <FrameHeader.h>`

Public Member Functions

- IOMpegFrameHeader (const uint8 *frameData)
  constructor
- ~IOMpegFrameHeader ()
  destructor
- bool validHeader () const
  return true if frame is valid
- int getMpegVersion () const
  get version of MPEG: 1=MPEG1, 2=MPEG2, 3=MPEG2.5, -1=invalid
- int getLayer () const
  get layer: 1/2/3=Layer 1/2/3, -1=invalid
- int getBitRate () const
  get bit rate in kBit/sec: 0=variable, -1=invalid
- int getSampleRate () const
  get sample rate in kSamples/sec: -1=invalid
- int getPadding () const
  get padding byte
- bool hasCRC () const
  return true if frame contains CRC checksum
- int getFrameSize () const
  get frame size
- int getCrcSize () const
  get CRC data size
- int getAudioSize () const
  get audio data size
- double getFrameTime () const
  get audio time in ms
- string getInfo () const
  get info string

Static Public Member Functions

- static bool isFirstByte (uint8 byte)
  return true if byte is first byte of a MPEG frame header
- static bool isSecondByte (uint8 byte)
  return true if byte is second byte of a MPEG frame header

Static Public Attributes

- static const int MAX_FRAME_SIZE = (144 ∗ 448) / 8 + 1
  maximum size of buffer for frame

10.29.1 Detailed Description

class representing a MPEG Header

10.29.2 Constructor & Destructor Documentation

10.29.2.1 AH::IOMpegFrameHeader::IOMpegFrameHeader (const uint8 ∗ *frameData*)

constructor

10.29.2.2 AH::IOMpegFrameHeader::∼IOMpegFrameHeader () [inline]

destructor

10.29.3 Member Function Documentation

10.29.3.1 int AH::IOMpegFrameHeader::getAudioSize () const

get audio data size

10.29.3.2 int AH::IOMpegFrameHeader::getBitRate () const

get bit rate in kBit/sec: 0=variable, -1=invalid

10.29.3.3 int AH::IOMpegFrameHeader::getCrcSize () const

get CRC data size

10.29.3.4 int AH::IOMpegFrameHeader::getFrameSize () const

get frame size

10.29.3.5 double AH::IOMpegFrameHeader::getFrameTime () const

get audio time in ms

10.29.3.6 string AH::IOMpegFrameHeader::getInfo () const

get info string

10.29.3.7 int AH::IOMpegFrameHeader::getLayer () const

get layer: 1/2/3=Layer 1/2/3, -1=invalid

10.29.3.8 int AH::IOMpegFrameHeader::getMpegVersion () const

get version of MPEG: 1=MPEG1, 2=MPEG2, 3=MPEG2.5, -1=invalid

10.29.3.9 int AH::IOMpegFrameHeader::getPadding () const

get padding byte

10.29.3.10 int AH::IOMpegFrameHeader::getSampleRate () const

get sample rate in kSamples/sec: -1=invalid

10.29.3.11 bool AH::IOMpegFrameHeader::hasCRC () const

return true if frame contains CRC checksum

10.29.3.12 static bool AH::IOMpegFrameHeader::isFirstByte (uint8 *byte*) [static]

return true if byte is first byte of a MPEG frame header

10.29.3.13 static bool AH::IOMpegFrameHeader::isSecondByte (uint8 *byte*) [static]

return true if byte is second byte of a MPEG frame header

10.29.3.14 bool AH::IOMpegFrameHeader::validHeader () const

return true if frame is valid

10.29.4 Member Data Documentation

10.29.4.1 const int AH::IOMpegFrameHeader::MAX_FRAME_SIZE = (144 * 448) / 8 + 1 [static]

maximum size of buffer for frame

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

- AH/IO/Mpeg/FrameHeader.h

10.30 AH::IOPlot Class Reference

use *gnuplot* to plot data from within this process

`#include <Plot.h>`

Public Member Functions

- IOPlot ()

  constructor: start gnuplot

- IOPlot (const std::string &plotprog)
- ~IOPlot ()

  destructor: exit gnuplot

- void exec (const std::string &cmd)

  send command to gnuplot

- void exec (const std::vector< std::string > &cmds)

  send multiple commands to gnuplot

- void plot (const std::string &file, const std::string &options)

  send PLOT command to gnuplot to plot data from a file

- void plot (const std::string &options)

  send PLOT command to gnuplot

- void plot (const std::vector< std::string > &data, const std::string &options)

  send PLOT command to gnuplot to plot data directly

- FILE * operator() () const

  operator () returns the pipe to gnuplot

- void setXrange (double from, double to)

  set X range (horizontal axis)

- void setYrange (double from, double to)

  set Y range (vertical axis)

10.30.1 Detailed Description

use *gnuplot* to plot data from within this process

`AH::IOPlot` uses *gnuplot* to plot data.

Please include *AH/IO/Plot.h*.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 AH::IOPlot::IOPlot ()

constructor: start gnuplot

10.30.2.2 AH::IOPlot::IOPlot (const std::string & *plotprog*)

10.30.2.3 AH::IOPlot::~IOPlot ()

destructor: exit gnuplot

10.30.3 Member Function Documentation

10.30.3.1 void AH::IOPlot::exec (const std::vector< std::string > & *cmds*)

send multiple commands to gnuplot

10.30.3.2 void AH::IOPlot::exec (const std::string & *cmd*)

send command to gnuplot

10.30.3.3 FILE* AH::IOPlot::operator() () const [inline]

operator () returns the pipe to gnuplot

10.30.3.4 void AH::IOPlot::plot (const std::vector< std::string > & *data*, const std::string & *options*)

send PLOT command to gnuplot to plot data directly

10.30.3.5 void AH::IOPlot::plot (const std::string & *options*)

send PLOT command to gnuplot

10.30.3.6 void AH::IOPlot::plot (const std::string & *file*, const std::string & *options*)

send PLOT command to gnuplot to plot data from a file

10.30.3.7 void AH::IOPlot::setXrange (double *from*, double *to*)

set X range (horizontal axis)

10.30.3.8 void AH::IOPlot::setYrange (double *from*, double *to*)

set Y range (vertical axis)

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

- AH/IO/Plot.h

10.31 AH::IOProfile Class Reference

`#include <Profile.h>`

Public Member Functions

- IOProfile ()
- void catchNow ()
- std::string getLastRunTime () const
- std::string getLastUserRunTime () const
- std::string getLastSystemRunTime () const
- std::string getRunTime () const
- std::string getUserRunTime () const
- std::string getSystemRunTime () const

10.31.1 Detailed Description

profiling utility class.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 AH::IOProfile::IOProfile () [inline]

default constructor.

10.31.3 Member Function Documentation

10.31.3.1 void AH::IOProfile::catchNow () [inline]

catch current time stamp.

10.31.3.2 std::string AH::IOProfile::getLastRunTime () const [inline]

return string containing elapsed time.

10.31.3.3 std::string AH::IOProfile::getLastSystemRunTime () const [inline]

return string containing elapsed system time.

10.31.3.4 std::string AH::IOProfile::getLastUserRunTime () const [inline]

return string containing elapsed user time.

10.31.3.5 std::string AH::IOProfile::getRunTime () const [inline]

return string containing elapsed time.

10.31.3.6 std::string AH::IOProfile::getSystemRunTime () const [inline]

return string containing elapsed system time.

10.31.3.7 std::string AH::IOProfile::getUserRunTime () const [inline]

return string containing elapsed user time.

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

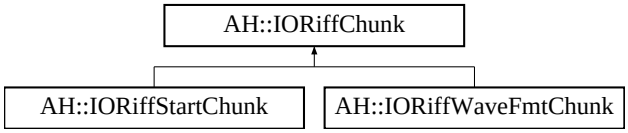
- AH/IO/Profile.h

10.32 AH::IORiffChunk Class Reference

class representing a RIFF chunk

`#include <Chunk.h>`

Inheritance diagram for AH::IORiffChunk::



Public Member Functions

- IORiffChunk (const uint8 *chunkbuf)

  constructor: create IORiffChunk using specified buffer

- IORiffChunk (const char *id, uint8 *data, uint32 datasize)

  constructor: create IORiffChunk with chunk on heap

- virtual ~IORiffChunk ()

  destructor

- virtual const uint8 * nextChunk () const

  get pointer to next chunk:

- virtual const uint8 * chunkBuffer () const

  get pointer to chunk buffer:

- virtual const uint8 * chunkData () const

  get pointer to chunk data:

- const char * getID () const

  return ID of chunk

- uint32 getDataSize () const

  return size of chunk data

- uint32 getChunkSize () const

  return size of complete chunk

- ostream & printInfo (ostream &out, const char *pre=0)

  print information about this chunk

Static Public Attributes

- static const uint32 size = 8

  size of chunk in bytes

Protected Member Functions

- virtual uint8 * chunkDataBuf ()

  get write pointer to chunk data buffer:

Protected Attributes

- uint8 * heapchunk
- const uint8 * chunk

10.32.1 Detailed Description

class representing a RIFF chunk

10.32.2 Constructor & Destructor Documentation

10.32.2.1 AH::IORiffChunk::IORiffChunk (const uint8 * *chunkbuf*)

constructor: create IORiffChunk using specified buffer

10.32.2.2 AH::IORiffChunk::IORiffChunk (const char * *id*, uint8 * *data*, uint32 *datasize*)

constructor: create IORiffChunk with chunk on heap

Parameters:

id chunk identifier

data chunk data

datasize size of chunk data

10.32.2.3 virtual AH::IORiffChunk::~IORiffChunk () `[virtual]`

destructor

10.32.3 Member Function Documentation

10.32.3.1 virtual const uint8* AH::IORiffChunk::chunkBuffer () const `[inline, virtual]`

get pointer to chunk buffer:

10.32.3.2 virtual const uint8* AH::IORiffChunk::chunkData () const [inline, virtual]

get pointer to chunk data:

10.32.3.3 virtual uint8* AH::IORiffChunk::chunkDataBuf () [inline, protected, virtual]

get write pointer to chunk data buffer:

10.32.3.4 uint32 AH::IORiffChunk::getChunkSize () const [inline]

return size of complete chunk

10.32.3.5 uint32 AH::IORiffChunk::getDataSize () const [inline]

return size of chunk data

10.32.3.6 const char* AH::IORiffChunk::getID () const [inline]

return ID of chunk

10.32.3.7 virtual const uint8* AH::IORiffChunk::nextChunk () const [inline, virtual]

get pointer to next chunk:

10.32.3.8 ostream& AH::IORiffChunk::printInfo (ostream & *out*, const char * *pre* = 0)

print information about this chunk

Reimplemented in AH::IORiffStartChunk, and AH::IORiffWaveFmtChunk.

10.32.4 Member Data Documentation

10.32.4.1 const uint8* AH::IORiffChunk::chunk [protected]

10.32.4.2 uint8* AH::IORiffChunk::heapchunk [protected]

10.32.4.3 const uint32 AH::IORiffChunk::size = 8 [static]

size of chunk in bytes

Reimplemented in AH::IORiffStartChunk.

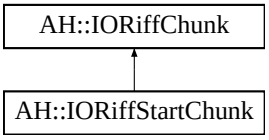
The documentation for this class was generated from the following file:

- AH/IO/Riff/Chunk.h

10.33 AH::IORiffStartChunk Class Reference

`#include <StartChunk.h>`

Inheritance diagram for AH::IORiffStartChunk::



Public Member Functions

- IORiffStartChunk (const unsigned char *chunkbuf)
  constructor: create object using specified buffer
- IORiffStartChunk (uint32 fileSize, const char *type=0)
  constructor: create object with chunk on heap
- virtual ~IORiffStartChunk ()
  destructor
- virtual const unsigned char * nextChunk ()
  get pointer to next chunk:
- const char * getType () const
  get RIFF type
- ostream & printInfo (ostream &out, const char *pre=0)
  print information about this chunk

Static Public Attributes

- static const uint32 size = 12
  size of chunk in bytes

Protected Attributes

- char type [5]

10.33.1 Constructor & Destructor Documentation

10.33.1.1 AH::IORiffStartChunk::IORiffStartChunk (const unsigned char * *chunkbuf*)

constructor: create object using specified buffer

10.33.1.2 AH::IORiffStartChunk::IORiffStartChunk (uint32 *fileSize*, const char * *type* = 0)

constructor: create object with chunk on heap

10.33.1.3 virtual AH::IORiffStartChunk::~IORiffStartChunk () [inline, virtual]

destructor

10.33.2 Member Function Documentation

10.33.2.1 const char* AH::IORiffStartChunk::getType () const [inline]

get RIFF type

10.33.2.2 virtual const unsigned char* AH::IORiffStartChunk::nextChunk () [inline, virtual]

get pointer to next chunk:

10.33.2.3 ostream& AH::IORiffStartChunk::printInfo (ostream & *out*, const char * *pre* = 0)

print information about this chunk

Reimplemented from AH::IORiffChunk.

10.33.3 Member Data Documentation

10.33.3.1 const uint32 AH::IORiffStartChunk::size = 12 [static]

size of chunk in bytes

Reimplemented from AH::IORiffChunk.

10.33.3.2 char AH::IORiffStartChunk::type[5] [protected]

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

- AH/IO/Riff/StartChunk.h

10.34 AH::IORiffWaveDataBuffer Class Reference

buffer for RIFF wave data

`#include <WaveDataBuf.h>`

Public Member Functions

- IORiffWaveDataBuffer (size_t n, size_t bits)
  constructor for buffer of `n` *elements of size* `bits` *bits*
- ~IORiffWaveDataBuffer ()
  destructor
- void resize (size_t n)
  resize buffer for `n` *elements*
- void recreate (size_t n)
  recreate buffer for `n` *elements*
- void clear ()
  empty buffer
- size_t getNum () const
  return number of elements which can be stored in buffer
- size_t getSize () const
  return size of buffer in bytes
- int32 getEntry (size_t idx) const
  return specified entry in buffer
- void * operator() () const
  return pointer to buffer

10.34.1 Detailed Description

buffer for RIFF wave data

`IORiffWaveDataBuffer` represents a buffer for RIFF wave data.

Please include *AH/IO/Riff/WaveDataBuf.h*.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 AH::IORiffWaveDataBuffer::IORiffWaveDataBuffer (size_t *n*, size_t *bits*)

constructor for buffer of `n` elements of size `bits` bits

The constructor creates a new buffer which contains `n` elements where each element occupies `bits` bits.

So far only 8, 16, 24 and 32 bits are supported. For other values an exception is thrown.

10.34.2.2 AH::IORiffWaveDataBuffer::∼IORiffWaveDataBuffer () [inline]

destructor

10.34.3 Member Function Documentation

10.34.3.1 void AH::IORiffWaveDataBuffer::clear () [inline]

empty buffer

10.34.3.2 int32 AH::IORiffWaveDataBuffer::getEntry (size_t *idx*) const

return specified entry in buffer

10.34.3.3 size_t AH::IORiffWaveDataBuffer::getNum () const [inline]

return number of elements which can be stored in buffer

10.34.3.4 size_t AH::IORiffWaveDataBuffer::getSize () const [inline]

return size of buffer in bytes

10.34.3.5 void∗ AH::IORiffWaveDataBuffer::operator() () const [inline]

return pointer to buffer

10.34.3.6 void AH::IORiffWaveDataBuffer::recreate (size_t *n*)

recreate buffer for n elements

`recreate(s)` resizes the buffer to contains `s` elements:

- if `s` is not equal to the current size then the old buffer is destroyed and a new one with the appropriate size is created.
- if `s` is equal to the current size then nothing is done at all.

This means that the buffer grows and shrinks according to the new size `s`. This costs execution time but does not block memory.

10.34.3.7 void AH::IORiffWaveDataBuffer::resize (size_t *n*)

resize buffer for n elements

`resize(s)` resizes the buffer to contains `s` elements:

- if `s` is bigger than the current size then the old buffer is destroyed and a new one with the appropriate size is created.
- if `s` is smaller than the current size then nothing is done at all.

- if `s` is equal to the current size then nothing is done at all.

This means that the buffer does only grow but never shrinks. This speeds up the execution but blocks memory.

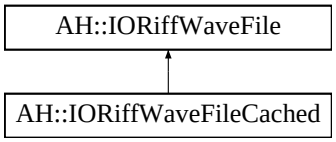
The documentation for this class was generated from the following file:

- AH/IO/Riff/WaveDataBuf.h

10.35 AH::IORiffWaveFile Class Reference

`#include <WaveFile.h>`

Inheritance diagram for AH::IORiffWaveFile::



Public Member Functions

- IORiffWaveFile (const char *name, bool isPipe, int verbose=0)
  constructor for input WAV file
- IORiffWaveFile (const char *name, int verbose=0)
  constructor for input file (WAV, MP3, Ogg, ...)
- IORiffWaveFile (const char *name, uint16 channel_num, uint16 sample_size, uint32 sample_rate, uint32 sample_num, int verbose=0)
  constructor for output WAV file
- virtual ~IORiffWaveFile ()
  destructor
- bool isPipe () const
  return true if wave file is a pipe
- uint32 getSampleNum () const
  get number of samples
- uint16 getChannelNum () const
  get number of channels
- uint32 getSampleRate () const
  get sample rate in Hz
- uint16 getSampleSize () const
  get sample size in bits per sample
- int32 getMaxAmplitude () const
  get maximum amplitude
- void readHeader (const char *name, bool isPipe=false)
  read header of wave file
- int32 readSamples (int32 *data, uint32 num)
  read PCM samples from wave file

- int32 readRaw (unsigned char *data, uint32 size)
  read raw PCM data from wave file
- int32 readBuf (unsigned char *buf, uint32 size)
  read a data buffer from wave file
- void writeHeader (const char *name)
  rewrite header into new wave file
- void writeHeader ()
  write header into current wave file
- int32 writeSamples (const int32 *data, uint32 num)
- int32 writeRaw (const unsigned char *data, uint32 size)
  write raw PCM data into new wave file
- int32 writeBuf (const unsigned char *buf, uint32 size)
- uint32 getDataPos () const
  get current data position
- uint32 getDataPos (double t) const
  get data position from time/sec
- uint32 getDataPos (uint32 sampleNum) const
  get data position from specified sample number
- uint32 setDataPos (double t)
  set file to sample position specified by time/sec
- uint32 setDataPos (uint32 sampleNum)
  set file to specified sample number
- uint32 getFilePos () const
  get current file position
- uint32 setFilePos (uint32 pos)
  set file to specified file position
- uint32 getSampleNum (double t) const
  get number of samples from time/sec
- double getTime (uint32 sampleNum) const
  get time/sec from number of samples
- uint32 getDataBufSize (double t) const
  get size of data for desired time/sec
- uint32 getDataBufSize (uint32 sampleNum) const
  get size of data for desired samples

- int32 getBufSample (unsigned char *buffer, uint32 sampleNum, int channel)

  get specified sample from buffer

- void setBufSample (unsigned char *buffer, uint32 sampleNum, int channel, int32 sampleValue)

  set specified sample in buffer

- void getLimits (double t1, double t2, int16 &lMin, int16 &lMax, int16 &rMin, int16 &rMax, int32 &lMinCnt, int32 &lMaxCnt, int32 &rMinCnt, int32 &rMaxCnt)

  analyze time frame

- virtual std::ostream & printInfo (std::ostream &out)

  print information about the wave file

- virtual std::ostream & printData (std::ostream &out, size_t ptnum, double t1, double t2)

  print data for specific time frame

Protected Attributes

- const char * waveName
- int verbose

10.35.1 Constructor & Destructor Documentation

10.35.1.1 AH::IORiffWaveFile::IORiffWaveFile (const char * *name*, bool *isPipe*, int *verbose* = 0)

constructor for input WAV file

10.35.1.2 AH::IORiffWaveFile::IORiffWaveFile (const char * *name*, int *verbose* = 0)

constructor for input file (WAV, MP3, Ogg, ...)

10.35.1.3 AH::IORiffWaveFile::IORiffWaveFile (const char * *name*, uint16 *channel_num*, uint16 *sample_size*, uint32 *sample_rate*, uint32 *sample_num*, int *verbose* = 0)

constructor for output WAV file

Parameters:

name name of output file

channel_num number of channels

sample_size size of sample in bits

sample_rate sample rate in Hz

sample_num number of samples per channel

verbose verbose level

10.35.1.4 virtual AH::IORiffWaveFile::~IORiffWaveFile () [virtual]

destructor

10.35.2 Member Function Documentation

10.35.2.1 int32 AH::IORiffWaveFile::getBufSample (unsigned char * *buffer*, uint32 *sampleNum*, int *channel*)

get specified sample from buffer

10.35.2.2 uint16 AH::IORiffWaveFile::getChannelNum () const [inline]

get number of channels

10.35.2.3 uint32 AH::IORiffWaveFile::getDataBufSize (uint32 *sampleNum*) const

get size of data for desired samples

10.35.2.4 uint32 AH::IORiffWaveFile::getDataBufSize (double *t*) const

get size of data for desired time/sec

10.35.2.5 uint32 AH::IORiffWaveFile::getDataPos (uint32 *sampleNum*) const

get data position from specified sample number

10.35.2.6 uint32 AH::IORiffWaveFile::getDataPos (double *t*) const

get data position from time/sec

10.35.2.7 uint32 AH::IORiffWaveFile::getDataPos () const

get current data position

10.35.2.8 uint32 AH::IORiffWaveFile::getFilePos () const

get current file position

10.35.2.9 void AH::IORiffWaveFile::getLimits (double *t1*, double *t2*, int16 & *lMin*, int16 & *lMax*, int16 & *rMin*, int16 & *rMax*, int32 & *lMinCnt*, int32 & *lMaxCnt*, int32 & *rMinCnt*, int32 & *rMaxCnt*)

analyze time frame

Parameters:

t1 starting point

t2 end point

10.35.2.10 int32 AH::IORiffWaveFile::getMaxAmplitude () const [inline]

get maximum amplitude

10.35.2.11 uint32 AH::IORiffWaveFile::getSampleNum (double *t*) const

get number of samples from time/sec

10.35.2.12 uint32 AH::IORiffWaveFile::getSampleNum () const [inline]

get number of samples

10.35.2.13 uint32 AH::IORiffWaveFile::getSampleRate () const [inline]

get sample rate in Hz

10.35.2.14 uint16 AH::IORiffWaveFile::getSampleSize () const [inline]

get sample size in bits per sample

10.35.2.15 double AH::IORiffWaveFile::getTime (uint32 *sampleNum*) const

get time/sec from number of samples

10.35.2.16 bool AH::IORiffWaveFile::isPipe () const [inline]

return true if wave file is a pipe

10.35.2.17 virtual std::ostream& AH::IORiffWaveFile::printData (std::ostream & *out*, size_t *ptnum*, double *t1*, double *t2*) [virtual]

print data for specific time frame

Parameters:

out writable stream for data points

ptnum number of data points

t1 starting point

t2 end point

10.35.2.18 virtual std::ostream& AH::IORiffWaveFile::printInfo (std::ostream & *out*) [virtual]

print information about the wave file

10.35.2.19 int32 AH::IORiffWaveFile::readBuf (unsigned char * *buf*, uint32 *size*)

read a data buffer from wave file

10.35.2.20 void AH::IORiffWaveFile::readHeader (const char * *name*, bool *isPipe* = `false`)

read header of wave file

10.35.2.21 int32 AH::IORiffWaveFile::readRaw (unsigned char * *data*, uint32 *size*)

read raw PCM data from wave file

10.35.2.22 int32 AH::IORiffWaveFile::readSamples (int32 * *data*, uint32 *num*)

read PCM samples from wave file

`readSamples()` reads `num` samples into the `data` buffer. If the file contains more than one channel, then the samples are written in this order:

```
sample 1 channel 1
sample 1 channel 2
sample 2 channel 1
sample 2 channel 2
sample 3 channel 1
sample 3 channel 2
```

and so on. Therefore the `data` buffer must have the size `num * getChannelNum()`.

10.35.2.23 void AH::IORiffWaveFile::setBufSample (unsigned char * *buffer*, uint32 *sampleNum*, int *channel*, int32 *sampleValue*)

set specified sample in buffer

10.35.2.24 uint32 AH::IORiffWaveFile::setDataPos (uint32 *sampleNum*)

set file to specified sample number

10.35.2.25 uint32 AH::IORiffWaveFile::setDataPos (double *t*)

set file to sample position specified by time/sec

10.35.2.26 uint32 AH::IORiffWaveFile::setFilePos (uint32 *pos*)

set file to specified file position

10.35.2.27 int32 AH::IORiffWaveFile::writeBuf (const unsigned char * *buf*, uint32 *size*)

10.35.2.28 void AH::IORiffWaveFile::writeHeader ()

write header into current wave file

10.35.2.29 void AH::IORiffWaveFile::writeHeader (const char * *name*)

rewrite header into new wave file

10.35.2.30 int32 AH::IORiffWaveFile::writeRaw (const unsigned char * *data*, uint32 *size*)

write raw PCM data into new wave file

10.35.2.31 int32 AH::IORiffWaveFile::writeSamples (const int32 * *data*, uint32 *num*)

`writeSamples()` writes `num` samples from the `data` buffer. If the file contains more than one channel, then the samples are read from the `data` buffer in this order:

```
sample 1 channel 1
sample 1 channel 2
sample 2 channel 1
sample 2 channel 2
sample 3 channel 1
sample 3 channel 2
```

and so on. Therefore the `data` buffer must have the size `num * getChannelNum()`.

10.35.3 Member Data Documentation

10.35.3.1 int AH::IORiffWaveFile::verbose [protected]

10.35.3.2 const char* AH::IORiffWaveFile::waveName [protected]

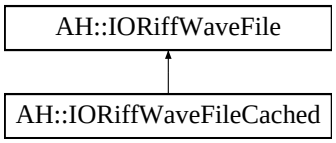
The documentation for this class was generated from the following file:

- AH/IO/Riff/WaveFile.h

10.36 AH::IORiffWaveFileCached Class Reference

`#include <WaveFileCached.h>`

Inheritance diagram for AH::IORiffWaveFileCached::



Public Member Functions

- IORiffWaveFileCached (const char *name, uint32 samSize, bool isPipe, int verbose=0)

  constructor for input WAV file

- IORiffWaveFileCached (const char *name, uint32 samSize, int verbose=0)

  constructor for input file (WAV, MP3, Ogg, ...)

- IORiffWaveFileCached (const char *name, uint16 channel_num, uint16 sample_size, uint32 sample_rate, uint32 sample_num=0, uint32 samSize=44100, int verbose=0)

  constructor for output WAV file

- virtual ~IORiffWaveFileCached ()

  destructor

- int32 readSample (int32 *value, uint32 samNum, uint16 chan)

  read PCM sample from wave file

- void writeSample (int32 value, uint32 samNum, uint16 chan)

10.36.1 Constructor & Destructor Documentation

10.36.1.1 AH::IORiffWaveFileCached::IORiffWaveFileCached (const char * *name*, uint32 *samSize*, bool *isPipe*, int *verbose* = 0)

constructor for input WAV file

Parameters:

name name of output file

samSize size of cache in number of samples

isPipe set to true if we must read from a pipe

verbose verbose level

10.36.1.2 AH::IORiffWaveFileCached::IORiffWaveFileCached (const char * *name*, uint32 *samSize*, int *verbose* = 0)

constructor for input file (WAV, MP3, Ogg, ...)

Parameters:

name name of output file

samSize size of cache in number of samples

verbose verbose level

10.36.1.3 AH::IORiffWaveFileCached::IORiffWaveFileCached (const char * *name*, uint16 *channel_num*, uint16 *sample_size*, uint32 *sample_rate*, uint32 *sample_num* = 0, uint32 *samSize* = 44100, int *verbose* = 0)

constructor for output WAV file

Parameters:

name name of output file

channel_num number of channels

sample_size size of sample in bits

sample_rate sample rate in Hz

sample_num number of samples per channel

samSize size of cache in number of samples

verbose verbose level

10.36.1.4 virtual AH::IORiffWaveFileCached::~IORiffWaveFileCached () [virtual]

destructor

10.36.2 Member Function Documentation

10.36.2.1 int32 AH::IORiffWaveFileCached::readSample (int32 * *value*, uint32 *samNum*, uint16 *chan*)

read PCM sample from wave file

10.36.2.2 void AH::IORiffWaveFileCached::writeSample (int32 *value*, uint32 *samNum*, uint16 *chan*)

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

- AH/IO/Riff/WaveFileCached.h

10.37 AH::IORiffWaveFmt Class Reference

`#include <WaveFmt.h>`

Public Types

- enum FormatCategory {
  UNKNOWN = 0x0000, MS_PCM = 0x0001, ADPCM = 0x0002, IBM_CVSD = 0x0005,
  ALAW = 0x0006, MULAW = 0x0007, OKI_ADPCM = 0x0010, DVI_ADPCM = 0x0011,
  DIGISTD = 0x0015, DIGIFIX = 0x0016, YAMAHA_ADPCM = 0x0020, DSPGROUP_-TRUESPEECH = 0x0022,
  SONARC = 0x0021, IBM_MULAW = 0x0101, IBM_ALAW = 0x0102, IBM_ADPCM = 0x0103,
  CREATIVE_ADPCM = 0x0200 }
  wave format categories

Public Member Functions

- IORiffWaveFmt (const unsigned char *data)
  constructor for input data
- IORiffWaveFmt (uint16 fmt, uint16 chans, uint32 sr, uint16 ss)
  constructor for output data
- virtual ~IORiffWaveFmt ()
  destructor
- uint32 getFormatKey () const
  return format key
- uint32 getChannels () const
  return number of channels
- uint32 getSampleRate () const
  return sample rate in samples per second
- uint32 getDataRate () const
  return data rate in bytes per second
- uint32 getBlockSize () const
  return block size
- uint16 getSampleSize () const
  return sample size in bits per sample
- const uint8 * getDataBuffer () const
  return pointer to data buffer

- uint32 getDataSize () const

  return data buffer size

- virtual std::ostream & printInfo (std::ostream &out, const char *pre=0)

  print information about this chunk

Static Public Member Functions

- static std::string getFormatName (FormatCategory fc)

  return format category as string

Classes

- class **MsPcm**
- class **Other**
- class **Specific**

10.37.1 Member Enumeration Documentation

10.37.1.1 enum AH::IORiffWaveFmt::FormatCategory

wave format categories

Enumerator:

UNKNOWN

MS_PCM

ADPCM

IBM_CVSD

ALAW

MULAW

OKI_ADPCM

DVI_ADPCM

DIGISTD

DIGIFIX

YAMAHA_ADPCM

DSPGROUP_TRUESPEECH

SONARC

IBM_MULAW

IBM_ALAW

IBM_ADPCM

CREATIVE_ADPCM

10.37.2 Constructor & Destructor Documentation

10.37.2.1 AH::IORiffWaveFmt::IORiffWaveFmt (const unsigned char * *data*)

constructor for input data

10.37.2.2 AH::IORiffWaveFmt::IORiffWaveFmt (uint16 *fmt*, uint16 *chans*, uint32 *sr*, uint16 *ss*)

constructor for output data

10.37.2.3 virtual AH::IORiffWaveFmt::~IORiffWaveFmt () [virtual]

destructor

10.37.3 Member Function Documentation

10.37.3.1 uint32 AH::IORiffWaveFmt::getBlockSize () const [inline]

return block size

10.37.3.2 uint32 AH::IORiffWaveFmt::getChannels () const [inline]

return number of channels

10.37.3.3 const uint8* AH::IORiffWaveFmt::getDataBuffer () const [inline]

return pointer to data buffer

10.37.3.4 uint32 AH::IORiffWaveFmt::getDataRate () const [inline]

return data rate in bytes per second

10.37.3.5 uint32 AH::IORiffWaveFmt::getDataSize () const

return data buffer size

10.37.3.6 uint32 AH::IORiffWaveFmt::getFormatKey () const [inline]

return format key

10.37.3.7 static std::string AH::IORiffWaveFmt::getFormatName (FormatCategory *fc*) [static]

return format category as string

10.37.3.8 uint32 AH::IORiffWaveFmt::getSampleRate () const [inline]

return sample rate in samples per second

10.37.3.9 uint16 AH::IORiffWaveFmt::getSampleSize () const

return sample size in bits per sample

10.37.3.10 virtual std::ostream& AH::IORiffWaveFmt::printInfo (std::ostream & *out*, const char * *pre* = 0) [virtual]

print information about this chunk

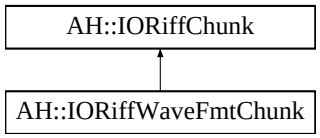
The documentation for this class was generated from the following file:

- AH/IO/Riff/WaveFmt.h

10.38 AH::IORiffWaveFmtChunk Class Reference

`#include <WaveFmtChunk.h>`

Inheritance diagram for AH::IORiffWaveFmtChunk::



Public Member Functions

- IORiffWaveFmtChunk (const unsigned char *data)
  constructor
- virtual ~IORiffWaveFmtChunk ()
  destructor
- virtual const unsigned char * nextChunk ()
  get pointer to next chunk:
- virtual const unsigned char * chunkData ()
  get pointer to chunk data:
- uint32 getFormatKey () const
  return format key
- uint32 getChannels () const
  return number of channels
- uint32 getSampleRate () const
  return sample rate in samples per second
- uint32 getDataRate () const
  return data rate in bytes per second
- uint32 getBlockSize () const
  return block size
- virtual ostream & printInfo (ostream &out, const char *pre=0)
  print information about this chunk

Classes

- class **MsPcm**
- class **Specific**

10.38.1 Constructor & Destructor Documentation

10.38.1.1 AH::IORiffWaveFmtChunk::IORiffWaveFmtChunk (const unsigned char * *data*)

constructor

10.38.1.2 virtual AH::IORiffWaveFmtChunk::~IORiffWaveFmtChunk () [inline, virtual]

destructor

10.38.2 Member Function Documentation

10.38.2.1 virtual const unsigned char* AH::IORiffWaveFmtChunk::chunkData () [inline, virtual]

get pointer to chunk data:

10.38.2.2 uint32 AH::IORiffWaveFmtChunk::getBlockSize () const [inline]

return block size

10.38.2.3 uint32 AH::IORiffWaveFmtChunk::getChannels () const [inline]

return number of channels

10.38.2.4 uint32 AH::IORiffWaveFmtChunk::getDataRate () const [inline]

return data rate in bytes per second

10.38.2.5 uint32 AH::IORiffWaveFmtChunk::getFormatKey () const [inline]

return format key

10.38.2.6 uint32 AH::IORiffWaveFmtChunk::getSampleRate () const [inline]

return sample rate in samples per second

10.38.2.7 virtual const unsigned char* AH::IORiffWaveFmtChunk::nextChunk () [inline, virtual]

get pointer to next chunk:

10.38.2.8 virtual ostream& AH::IORiffWaveFmtChunk::printInfo (ostream & *out*, const char * *pre* = 0) [virtual]

print information about this chunk

Reimplemented from AH::IORiffChunk.

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

- AH/IO/Riff/WaveFmtChunk.h

10.39 AH::IOSmartBuffer< T > Class Template Reference

smart buffer for any type `T`

`#include <Buffer.h>`

Public Member Functions

- IOSmartBuffer (size_t s)
  constructor for buffer of `s` *elements of type* `T`
- IOSmartBuffer ()
  constructor for empty buffer of type `T`
- ~IOSmartBuffer ()
  destructor
- void resize (size_t s)
  resize buffer for `s` *elements of type* `T`
- void recreate (size_t s)
  resize buffer for `s` *elements of type* `T`
- void remove ()
  remove buffer and free its allocated memory
- void clear ()
  clear all buffer entries (set to 0)
- void initAll (const T &t)
  set all buffer entries to specified value
- size_t getNum () const
  return number of types `T` *which can be stored in buffer*
- size_t getSize () const
  return size of buffer in bytes
- T * operator() () const
  return pointer to buffer
- const T & getEntry (size_t idx) const
  return specified entry in buffer (with bound check)
- void setEntry (size_t idx, const T &t) const
  set specified entry in buffer (with bound check)
- const T & operator[ ] (size_t idx) const
  return specified readonly entry in buffer (no bound check)

- T & operator[ ] (size_t idx)

  return specified writable entry in buffer (no bound check)

10.39.1 Detailed Description

template<class T> class AH::IOSmartBuffer< T >

smart buffer for any type `T`

Use `AH::IOSmartBuffer` if you need a resizable buffer or a buffer which gets properly removed if the scope is left, e.g. due to an exception.

Please include *AH/IO/Smart/Buffer.h*.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 template<class T> AH::IOSmartBuffer< T >::IOSmartBuffer (size_t *s*) [inline]

constructor for buffer of `s` elements of type `T`

10.39.2.2 template<class T> AH::IOSmartBuffer< T >::IOSmartBuffer () [inline]

constructor for empty buffer of type `T`

10.39.2.3 template<class T> AH::IOSmartBuffer< T >::~IOSmartBuffer () [inline]

destructor

10.39.3 Member Function Documentation

10.39.3.1 template<class T> void AH::IOSmartBuffer< T >::clear () [inline]

clear all buffer entries (set to 0)

10.39.3.2 template<class T> const T& AH::IOSmartBuffer< T >::getEntry (size_t *idx*) const [inline]

return specified entry in buffer (with bound check)

10.39.3.3 template<class T> size_t AH::IOSmartBuffer< T >::getNum () const [inline]

return number of types `T` which can be stored in buffer

10.39.3.4 template<class T> size_t AH::IOSmartBuffer< T >::getSize () const [inline]

return size of buffer in bytes

10.39.3.5 template<class T> void AH::IOSmartBuffer< T >::initAll (const T & *t*) [inline]

set all buffer entries to specified value

10.39.3.6 template<class T> T* AH::IOSmartBuffer< T >::operator() () const [inline]

return pointer to buffer

10.39.3.7]

template<class T> T& AH::IOSmartBuffer< T >::operator[ ] (size_t *idx*) [inline]

return specified writable entry in buffer (no bound check)

10.39.3.8]

template<class T> const T& AH::IOSmartBuffer< T >::operator[ ] (size_t *idx*) const [inline]

return specified readonly entry in buffer (no bound check)

10.39.3.9 template<class T> void AH::IOSmartBuffer< T >::recreate (size_t *s*) [inline]

resize buffer for `s` elements of type `T`

`recreate(s)` resizes the buffer to contains `s` elements of type `T`:

- if `s` is not equal to the current size then the old buffer is destroyed and a new one with the appropriate size is created.
- if `s` is equal to the current size then nothing is done at all.

This means that the buffer grows and shrinks according to the new size `s`. This costs execution time but does not block memory.

10.39.3.10 template<class T> void AH::IOSmartBuffer< T >::remove () [inline]

remove buffer and free its allocated memory

10.39.3.11 template<class T> void AH::IOSmartBuffer< T >::resize (size_t *s*) [inline]

resize buffer for `s` elements of type `T`

`resize(s)` resizes the buffer to contains `s` elements of type `T`:

- if `s` is bigger than the current size then the old buffer is destroyed and a new one with the appropriate size is created.
- if `s` is smaller than the current size then nothing is done at all.
- if `s` is equal to the current size then nothing is done at all.

This means that the buffer does only grow but never shrinks. This speeds up the execution but blocks memory.

10.39.3.12 template<class T> void AH::IOSmartBuffer< T >::setEntry (size_t *idx*, const T & *t*) const [inline]

set specified entry in buffer (with bound check)

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

- AH/IO/Smart/Buffer.h

10.40 AH::IOSmartBuffer2D< T > Class Template Reference

smart 2D buffer for any type `T`

`#include <Buffer2D.h>`

Public Member Functions

- IOSmartBuffer2D (size_t n, size_t s)
  constructor for buffer of `s` *elements of type* `T`
- IOSmartBuffer2D (size_t n)
  constructor for empty buffer of type `T`
- ~IOSmartBuffer2D ()
  destructor
- void resize (size_t s)
  resize all buffers for `s` *elements of type* `T`
- void recreate (size_t s)
  resize buffer for `s` *elements of type* `T`
- void remove ()
  remove buffer and free its allocated memory
- void clear ()
  clear all buffer entries (set to 0)
- void initAll (const T &t)
  set all buffer entries to specified value
- size_t getNum () const
  return number of types `T` *which can be stored in buffer*
- size_t getSize () const
  return size of buffer in bytes
- T * getBuffer (size_t b)
  return 1D buffer
- T ** getBuffer ()
  return 1D buffer
- T ** operator() ()
  return pointer to buffer
- const T & getEntry (size_t b, size_t idx) const
  return specified entry in buffer (with bound check)

- void setEntry (size_t b, size_t idx, const T &t)
  set specified entry in buffer (with bound check)

10.40.1 Detailed Description

template<class T> class AH::IOSmartBuffer2D< T >

smart 2D buffer for any type `T`

Use `AH::IOSmartBuffer` if you need a resizable buffer or a buffer which gets properly removed if the scope is left, e.g. due to an exception.

Please include *AH/IO/Smart/Buffer.h*.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 template<class T> AH::IOSmartBuffer2D< T >::IOSmartBuffer2D (size_t *n*, size_t *s*) [inline]

constructor for buffer of `s` elements of type `T`

10.40.2.2 template<class T> AH::IOSmartBuffer2D< T >::IOSmartBuffer2D (size_t *n*) [inline]

constructor for empty buffer of type `T`

10.40.2.3 template<class T> AH::IOSmartBuffer2D< T >::~IOSmartBuffer2D () [inline]

destructor

10.40.3 Member Function Documentation

10.40.3.1 template<class T> void AH::IOSmartBuffer2D< T >::clear () [inline]

clear all buffer entries (set to 0)

10.40.3.2 template<class T> T** AH::IOSmartBuffer2D< T >::getBuffer () [inline]

return 1D buffer

10.40.3.3 template<class T> T* AH::IOSmartBuffer2D< T >::getBuffer (size_t *b*) [inline]

return 1D buffer

10.40.3.4 template<class T> const T& AH::IOSmartBuffer2D< T >::getEntry (size_t *b*, size_t *idx*) const [inline]

return specified entry in buffer (with bound check)

10.40.3.5 template<class T> size_t AH::IOSmartBuffer2D< T >::getNum () const [inline]

return number of types `T` which can be stored in buffer

10.40.3.6 template<class T> size_t AH::IOSmartBuffer2D< T >::getSize () const [inline]

return size of buffer in bytes

10.40.3.7 template<class T> void AH::IOSmartBuffer2D< T >::initAll (const T & *t*) [inline]

set all buffer entries to specified value

10.40.3.8 template<class T> T** AH::IOSmartBuffer2D< T >::operator() () [inline]

return pointer to buffer

10.40.3.9 template<class T> void AH::IOSmartBuffer2D< T >::recreate (size_t *s*) [inline]

resize buffer for `s` elements of type `T`

`recreate(s)` resizes the buffer to contains `s` elements of type `T`:

- if `s` is not equal to the current size then the old buffer is destroyed and a new one with the appropriate size is created.
- if `s` is equal to the current size then nothing is done at all.

This means that the buffer grows and shrinks according to the new size `s`. This costs execution time but does not block memory.

10.40.3.10 template<class T> void AH::IOSmartBuffer2D< T >::remove () [inline]

remove buffer and free its allocated memory

10.40.3.11 template<class T> void AH::IOSmartBuffer2D< T >::resize (size_t *s*) [inline]

resize all buffers for `s` elements of type `T`

`resize(s)` resizes all buffers to contains `s` elements of type `T`:

- if `s` is bigger than the current size then the old buffer is destroyed and a new one with the appropriate size is created.
- if `s` is smaller than the current size then nothing is done at all.
- if `s` is equal to the current size then nothing is done at all.

This means that the buffer does only grow but never shrinks. This speeds up the execution but blocks memory.

10.40.3.12 template<class T> void AH::IOSmartBuffer2D< T >::setEntry (size_t *b*, size_t *idx*, const T & *t*) [inline]

set specified entry in buffer (with bound check)

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

- AH/IO/Smart/Buffer2D.h

10.41 AH::IOSmartFilePtr Class Reference

smart `stdio` file pointer

`#include <FilePtr.h>`

Public Member Functions

- IOSmartFilePtr (const string &name, const string &mode)
  constructor for file pointer
- ~IOSmartFilePtr ()
  destructor
- void close ()
  close file
- FILE * operator() ()
  return pointer to file

10.41.1 Detailed Description

smart `stdio` file pointer

Use `AH::IOSmartFilePtr` if you need a `FILE` pointer to a file which is closed properly if the scope is left, e.g. due to an exception.

Please include *AH/IO/Smart/FilePtr.h*.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 AH::IOSmartFilePtr::IOSmartFilePtr (const string & *name*, const string & *mode*) `[inline]`

constructor for file pointer

Parameters:

name name of file

mode file mode: "r" for read, "w" for write

10.41.2.2 AH::IOSmartFilePtr::~IOSmartFilePtr () `[inline]`

destructor

10.41.3 Member Function Documentation

10.41.3.1 void AH::IOSmartFilePtr::close () `[inline]`

close file

10.41.3.2 FILE* AH::IOSmartFilePtr::operator() () [inline]

return pointer to file

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

- AH/IO/Smart/FilePtr.h

10.42 AH::IOTimeStamp Class Reference

`#include <TimeStamp.h>`

Public Member Functions

- IOTimeStamp ()
- IOTimeStamp (const IOTimeStamp &ts)

  copy constructor

- void update ()
- unsigned int getUserClocks () const
- unsigned int getSystemClocks () const
- double getUserTime () const
- double getSystemTime () const
- std::string getUserTimeString () const
- std::string getSystemTimeString () const
- std::string getTimeString () const
- IOTimeStamp operator+ (const IOTimeStamp &ts1)
- IOTimeStamp operator- (const IOTimeStamp &ts1)
- IOTimeStamp & operator= (const IOTimeStamp &ts1)

Friends

- bool operator== (const IOTimeStamp &ts1, const IOTimeStamp &ts2)
- bool operator!= (const IOTimeStamp &ts1, const IOTimeStamp &ts2)

10.42.1 Detailed Description

represents a time stamp (for profiling).

10.42.2 Constructor & Destructor Documentation

10.42.2.1 AH::IOTimeStamp::IOTimeStamp () `[inline]`

default constructor.

10.42.2.2 AH::IOTimeStamp::IOTimeStamp (const IOTimeStamp & *ts*) `[inline]`

copy constructor

10.42.3 Member Function Documentation

10.42.3.1 unsigned int AH::IOTimeStamp::getSystemClocks () const `[inline]`

return system time in clocks.

10.42.3.2 double AH::IOTimeStamp::getSystemTime () const [inline]

return system time in seconds.

10.42.3.3 std::string AH::IOTimeStamp::getSystemTimeString () const [inline]

return system time as string.

10.42.3.4 std::string AH::IOTimeStamp::getTimeString () const [inline]

return user and system time as string.

10.42.3.5 unsigned int AH::IOTimeStamp::getUserClocks () const [inline]

return user time in clocks.

10.42.3.6 double AH::IOTimeStamp::getUserTime () const [inline]

return user time in seconds.

10.42.3.7 std::string AH::IOTimeStamp::getUserTimeString () const [inline]

return user time as string.

10.42.3.8 IOTimeStamp AH::IOTimeStamp::operator+ (const IOTimeStamp & *ts1*)

10.42.3.9 IOTimeStamp AH::IOTimeStamp::operator- (const IOTimeStamp & *ts1*)

10.42.3.10 IOTimeStamp& AH::IOTimeStamp::operator= (const IOTimeStamp & *ts1*)

10.42.3.11 void AH::IOTimeStamp::update () [inline]

update current time stamp.

10.42.4 Friends And Related Function Documentation

10.42.4.1 bool operator!= (const IOTimeStamp & *ts1*, const IOTimeStamp & *ts2*) [friend]

10.42.4.2 bool operator== (const IOTimeStamp & *ts1*, const IOTimeStamp & *ts2*) [friend]

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

- AH/IO/TimeStamp.h

Chapter 11

libAHio File Documentation

11.1 AH/IO/AudioFile.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <AH/Error/Sys.h>
#include <cstdio>
#include <cerrno>
#include <iostream>
#include <string>
#include <vector>
#include <sndfile.h>
```

Namespaces

- namespace AH

Classes

- class AH::IOAudioFileOpenPolicy
  generic policy to open any audio file
- class AH::IOAudioFileOpenPolicyFile
  policy to open a known file format.
- class AH::IOAudioFileOpenPolicyMP3
  policy to open an MP3 file by converting it into WAV format
- class AH::IOAudioFileOpenPolicyOGG
  policy to open an OGG/Vorbis file by converting it into WAV format
- class AH::IOAudioFileOpenPolicyMPP

policy to open an MP+ file by converting it into WAV format

- class AH::IOAudioFileOpenPolicyLPAC
  policy to open an LPAC file by converting it into WAV format
- class AH::IOAudioFileOpener
  opens audio file
- struct AH::IOAudioFileReadPolicy< SampleType >
  empty template policy for reading any SampleType from audio file
- struct AH::IOAudioFileReadPolicy< int32 >
  special policy for reading int32 data from audio file
- struct AH::IOAudioFileReadPolicy< int16 >
  special policy for reading int16 data from audio file
- struct AH::IOAudioFileReadPolicy< float >
  special policy for reading 32bit float data from audio file
- struct AH::IOAudioFileReadPolicy< double >
  special policy for reading 64bit double data from audio file
- class AH::IOAudioFileError
  error class for libsndfile errors
- class AH::IOAudioFileInfo
  utility functions to interpret SF_INFO struct
- class AH::IOAudioFile< SampleType >
  C++ interface to library libsndfile.
- class **AH::IOAudioFile< SampleType >::SfCache**

11.2 AH/IO/Cdda/Sector.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::CddaSector

11.3 AH/IO/CFormat.h File Reference

```
#include <AH/Config/Platform.h>
#include <string>
#include <iostream>
```

Namespaces

- namespace AH
- namespace std

Classes

- class AH::IOCFormat

  C-style printf compatible ostream applicator.

Functions

- ostream & AH::operator<< (ostream &out, const IOCFormat &fmt)

  allow to insert IOCFormat objects directly into an ostream

11.4 AH/IO/CmdOptHandler.h File Reference

```
#include <AH/Config/Platform.h>
#include <string>
#include <vector>
```

Namespaces

- namespace AH

Classes

- class AH::IOCmdOptHandler

11.5 AH/IO/CmdOption.h File Reference

```
#include <AH/Config/Platform.h>
#include <string>
#include <vector>
```

Namespaces

- namespace AH

Classes

- class AH::IOCmdOption

11.6 AH/IO/Convert.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOConvert

  class containing converter functions

11.7 AH/IO/Dir.h File Reference

```
#include <AH/Config/Platform.h>
#include <string>
#include <vector>
```

Namespaces

- namespace AH

Classes

- class AH::IODir
  directory handler

11.8 AH/IO/Error.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Error/Basic.h>
```

Namespaces

- namespace AH

Classes

- class AH::ErrorIO

  represents any Input/Output error

11.9 AH/IO/Riff/Error.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/IO/Error.h>
```

Namespaces

- namespace AH

Classes

- class AH::ErrorIORiff

  unspecified Riff IO error

- class AH::ErrorIORiffInvalidFormat
- class AH::ErrorIORiffInvalidWaveFormat

11.10 AH/IO/Midi/Device.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <cstdio>
```

Namespaces

- namespace AH

Classes

- class AH::IOMidiDevice

11.11 AH/IO/Midi/RawCmd.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <cstdio>
#include <vector>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOMidiRawCmd

11.12 AH/IO/Midi/SysExCmd.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <AH/IO/Midi/RawCmd.h>
```

Namespaces

- namespace AH

Classes

- class AH::IOMidiSysExCmd

11.13 AH/IO/Mpeg/FrameHeader.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOMpegFrameHeader

  class representing a MPEG Header

11.14 AH/IO/Plot.h File Reference

```
#include <AH/Config/Platform.h>
#include <cstdio>
#include <string>
#include <vector>
```

Namespaces

- namespace AH

Classes

- class AH::IOPlot

  use gnuplot *to plot data from within this process*

11.15 AH/IO/Profile.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/IO/TimeStamp.h>
#include <sys/times.h>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOProfile

11.16 AH/IO/Riff/Chunk.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <AH/Error/Sys.h>
#include <cerrno>
#include <string>
#include <iostream>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffChunk

  class representing a RIFF chunk

11.17 AH/IO/Riff/StartChunk.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/IO/Riff/Chunk.h>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffStartChunk

11.18 AH/IO/Riff/WaveDataBuf.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/IO/Riff/Chunk.h>
#include <cstdio>
#include <string>
#include <iostream>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffWaveDataBuffer
  buffer for RIFF wave data

11.19 AH/IO/Riff/WaveFile.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <cstdio>
#include <iostream>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffWaveFile

11.20 AH/IO/Riff/WaveFileCached.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <AH/IO/Riff/WaveFile.h>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffWaveFileCached

11.21 AH/IO/Riff/WaveFmt.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <string>
#include <iostream>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffWaveFmt
- class **AH::IORiffWaveFmt::Specific**
- class **AH::IORiffWaveFmt::MsPcm**
- class **AH::IORiffWaveFmt::Other**

11.22 AH/IO/Riff/WaveFmtChunk.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Config/Types.h>
#include <AH/IO/Riff/Chunk.h>
#include <iostream>
```

Namespaces

- namespace AH

Classes

- class AH::IORiffWaveFmtChunk
- class **AH::IORiffWaveFmtChunk::Specific**
- class **AH::IORiffWaveFmtChunk::MsPcm**

11.23 AH/IO/Smart/Buffer.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Error/Sys.h>
#include <AH/Error/Intern.h>
#include <cerrno>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOSmartBuffer< T >

  smart buffer for any type `T`

11.24 AH/IO/Smart/Buffer2D.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Error/Sys.h>
#include <AH/Error/Intern.h>
#include <cerrno>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOSmartBuffer2D< T >
  smart 2D buffer for any type `T`

11.25 AH/IO/Smart/FilePtr.h File Reference

```
#include <AH/Config/Platform.h>
#include <AH/Error/Sys.h>
#include <cstdio>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOSmartFilePtr

  smart `stdio` *file pointer*

11.26 AH/IO/TimeStamp.h File Reference

```
#include <AH/Config/Platform.h>
#include <sys/times.h>
#include <string>
```

Namespaces

- namespace AH

Classes

- class AH::IOTimeStamp

Index