commscan Reference Manual

Generated by Doxygen 1.2.16

Wed Apr 9 23:54:53 2003

Contents

1 commscan Hierarchical Index

1.1 commscan Class Hierarchy

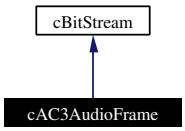
This inheritance list is sorted roughly, but not completely, alphabetically:

2 commscan Class Documentation

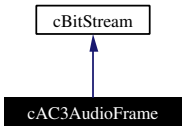
2.1 cAC3AudioFrame Class Reference

`#include <audio.h>`

Inheritance diagram for cAC3AudioFrame:



Collaboration diagram for cAC3AudioFrame:



Public Methods

- cAC3AudioFrame (unsigned char *data, unsigned int length)
- ~cAC3AudioFrame ()
- int FrameSize ()
- int BitRate ()
- int SamplingRate ()
- int ChannelMode ()

2.1.1 Detailed Description

A class representing a compressed AC3 audio frame.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 cAC3AudioFrame::cAC3AudioFrame (unsigned char * *data*, unsigned int *length*)

Constructs the audio frame. Parses the data and fills header structures.

Parameters:

data pointer to frame data

length length of frame data

2.1.2.2 cAC3AudioFrame::~cAC3AudioFrame ()

Destructs the audio frame. Actually does nothing.

2.1.3 Member Function Documentation

2.1.3.1 int cAC3AudioFrame::BitRate ()

Retrieves the bitrate of the audio frame.

Returns:

bitrate of the audio frame in kbits per second

2.1.3.2 int cAC3AudioFrame::ChannelMode ()

Retrieves the channel mode of the audio frame. See constants kA52_... in audio.h for possible modes.

Returns:

channel mode of the audio frame

2.1.3.3 int cAC3AudioFrame::FrameSize ()

Retrieves the size of the audio frame.

Returns:

size of the audio frame in bytes

2.1.3.4 int cAC3AudioFrame::SamplingRate ()

Retrieves sampling rate bit of the audio frame.

Returns:
sampling rate of the audio frame

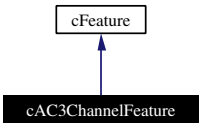
The documentation for this class was generated from the following files:

- audio.h
- audio.cpp

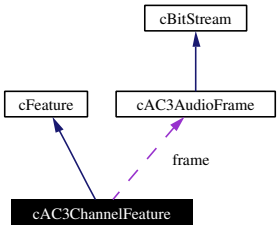
2.2 cAC3ChannelFeature Class Reference

`#include <feature.h>`

Inheritance diagram for cAC3ChannelFeature:



Collaboration diagram for cAC3ChannelFeature:



Public Methods

- cAC3ChannelFeature ()
- virtual ~cAC3ChannelFeature ()
- virtual void Refresh (void *frame)
- virtual char * Dump ()
- int ChannelMode ()

2.2.1 Detailed Description

A class representing detection of changes in AC3 channel modes.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 cAC3ChannelFeature::cAC3ChannelFeature ()

Constructs the feature. Initializes the feature's data.

2.2.2.2 cAC3ChannelFeature::~cAC3ChannelFeature () [virtual]

Destructs the feature. Actually does nothing.

2.2.3 Member Function Documentation

2.2.3.1 int cAC3ChannelFeature::ChannelMode ()

Retrieves the channel mode of the last refreshed frame.

Returns:
channel mode of the last refreshed frame

2.2.3.2 char * cAC3ChannelFeature::Dump () [virtual]

Dumps the feature's data (channel mode) to a string.

Returns:
null-terminated string containing the feature's data.

Reimplemented from cFeature.

2.2.3.3 void cAC3ChannelFeature::Refresh (void * *frame*) [virtual]

Refreshs the feature's data. First the channel mode of the current AC3 audio frame is checked. Then the new channel mode is compared with the old one. If it has changed, the feature's state is set to "changed".

Parameters:
frame AC3 audio frame which channel mode will be checked.

Reimplemented from cFeature.

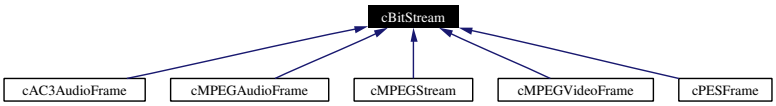
The documentation for this class was generated from the following files:

- feature.h
- feature.cpp

2.3 cBitStream Class Reference

`#include <stream.h>`

Inheritance diagram for cBitStream:



Public Methods

- cBitStream (unsigned char *data, unsigned int length)
- ~cBitStream ()
- unsigned int Length ()
- bool IsOnEnd ()
- void SeekStart ()
- void SeekBytes (unsigned int bytes)
- void SeekBits (unsigned int bits)
- void AlignBits ()
- unsigned int NextBits (unsigned int bits)
- unsigned int GetBits (unsigned int bits)
- unsigned int GetBit ()

2.3.1 Detailed Description

A class representing a data stream with bitwise access.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 cBitStream::cBitStream (unsigned char * *data*, unsigned int *length*)

Constructs the data stream. The caller is responsible for allocating and deallocating the buffer, data is pointing at.

Parameters:

data pointer to the buffer

length length of the buffer

2.3.2.2 cBitStream::~cBitStream ()

Destructs the data stream. Actually does nothing.

2.3.3 Member Function Documentation

2.3.3.1 void cBitStream::AlignBits ()

Moves internal stream pointer to next byte-aligned position.

2.3.3.2 unsigned int cBitStream::GetBit ()

Retrieves next single bit of data. Internal stream pointer is moved.

Returns:
a dword containing the next bit of data

2.3.3.3 unsigned int cBitStream::GetBits (unsigned int *bits*)

Retrieves next bits of data. Internal stream pointer is moved.

Parameters:
bits length of data in bits, a maximum of 32 is allowed.

Returns:
a dword containing the next bits of data

2.3.3.4 bool cBitStream::IsOnEnd ()

Retrieves whether internal stream pointer is at the end of the stream or not.

Returns:
true if internal stream pointer is at the end of the stream, false if not.

2.3.3.5 unsigned int cBitStream::Length ()

Retrieves the length of the stream.

Returns:
length in bytes

2.3.3.6 unsigned int cBitStream::NextBits (unsigned int *bits*)

Retrieves next bits of data. Internal stream pointer is not moved.

Parameters:

bits length of data in bits, a maximum of 32 is allowed.

Returns:

a dword containing the next bits of data

2.3.3.7 void cBitStream::SeekBits (unsigned int *bits*)

Moves internal stream pointer bitwise forward.

Parameters:

bits bits to move in stream

2.3.3.8 void cBitStream::SeekBytes (unsigned int *bytes*)

Moves internal stream pointer bytewise forward.

Parameters:

bytes bytes to move in stream

2.3.3.9 void cBitStream::SeekStart ()

Moves internal stream pointer to the start of the stream.

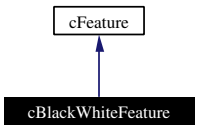
The documentation for this class was generated from the following files:

- stream.h
- stream.cpp

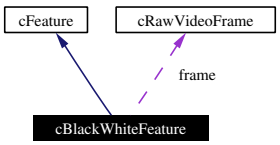
2.4 cBlackWhiteFeature Class Reference

`#include <feature.h>`

Inheritance diagram for cBlackWhiteFeature:



Collaboration diagram for cBlackWhiteFeature:



Public Methods

- cBlackWhiteFeature ()
- virtual ~cBlackWhiteFeature ()
- virtual void Refresh (void *frame)
- virtual char * Dump ()
- int State ()

2.4.1 Detailed Description

A class representing detection of black-and-white programmes.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 cBlackWhiteFeature::cBlackWhiteFeature ()

Constructs the feature. Initializes the feature's data.

2.4.2.2 cBlackWhiteFeature::~cBlackWhiteFeature () `[virtual]`

Destructs the feature. Actually does nothing.

2.4.3 Member Function Documentation

2.4.3.1 char * cBlackWhiteFeature::Dump () [virtual]

Dumps the feature's data (black-and-white state) to a string.

Returns:
null-terminated string containing the feature's data.

Reimplemented from cFeature.

2.4.3.2 void cBlackWhiteFeature::Refresh (void * *frame*) [virtual]

Refreshs the feature's data. First the black-and-white state is checked. Then the new state is compared with the old one. If it has changed, the feature's state is set to "changed".

Parameters:
frame video frame which black-and-white state will be checked

Reimplemented from cFeature.

2.4.3.3 int cBlackWhiteFeature::State ()

Retrieves the black-and-white state of the last refreshed frame.

Returns:
black-and-white state of the last refreshed frame (0 = black-and-white, 1 = colour)

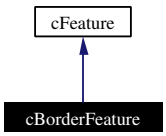
The documentation for this class was generated from the following files:

- feature.h
- feature.cpp

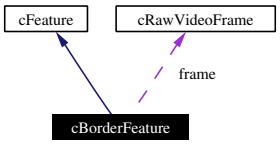
2.5 cBorderFeature Class Reference

`#include <feature.h>`

Inheritance diagram for cBorderFeature:



Collaboration diagram for cBorderFeature:



Public Methods

- cBorderFeature ()
- virtual ~cBorderFeature ()
- virtual void Refresh (void *frame)
- virtual char * Dump ()
- tBorder Border ()
- int GetLuminance ()

2.5.1 Detailed Description

A class representing border detection. It detects the black border a video frame has.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 cBorderFeature::cBorderFeature ()

Constructs the feature. Initializes the feature's data.

2.5.2.2 cBorderFeature::~cBorderFeature () [virtual]

Destructs the feature. Actually does nothing.

2.5.3 Member Function Documentation

2.5.3.1 tBorder cBorderFeature::Border ()

Retrieves the border of the last refreshed video frame.

Returns:
border of the last refreshed video frame.

2.5.3.2 char * cBorderFeature::Dump () [virtual]

Dumps the feature's data (luminance, top, bottom, left, right border, luminance of first and last lines and columns) to a string.

Returns:
null-terminated string containing the feature's data.

Reimplemented from cFeature.

2.5.3.3 int cBorderFeature::GetLuminance ()

Retrieves the luminance of the last refreshed video frame.

Returns:
luminance of the last refreshed video frame.

2.5.3.4 void cBorderFeature::Refresh (void * *frame*) [virtual]

Refreshs the feature's data. First the luminance of the picture is calculated. If it is greater than kMinPictureLuminance, the borders (top, bottom, left, right) will be calculated. Then the new border is compared with the old border. If it has changed, the feature's state is set to "changed".

Parameters:
frame video frame which border will be calculated.

Reimplemented from cFeature.

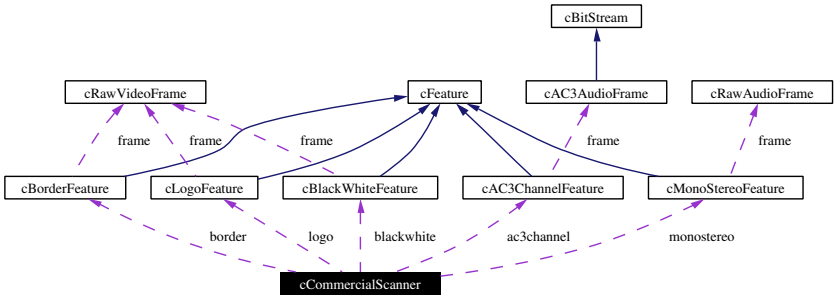
The documentation for this class was generated from the following files:

- feature.h
- feature.cpp

2.6 cCommercialScanner Class Reference

`#include <scanner.h>`

Collaboration diagram for cCommercialScanner:



Public Methods

- cCommercialScanner (const char *fileName)
- ~cCommercialScanner ()
- void Process ()
- void DumpChanges ()
- void SaveChanges (int prefix)

2.6.1 Detailed Description

This class is responsible for reading the recorded data stream using classes from VDR, demultiplexing and decoding it using the frame and stream classes, collecting change data using the feature classes and doing the final analysis for setting the cutting marks.

2.6.2 Constructor & Destructor Documentation

2.6.2.1 cCommercialScanner::cCommercialScanner (const char * *fileName*)

Constructs the scanner. Opens the recording and its index file.

Parameters:

fileName null-terminated string containing the name of the recording

2.6.2.2 cCommercialScanner::~cCommercialScanner ()

Destructs the scanner. Closes the recording and its index file.

2.6.3 Member Function Documentation

2.6.3.1 void cCommercialScanner::DumpChanges ()

Dumps the feature changes to logfile and stdout.

2.6.3.2 void cCommercialScanner::Process ()

Main Method of the scanner. Reads the recording, demultiplexes and decodes it, collects the feature changes and does the final analysis for setting the cutting marks.

2.6.3.3 void cCommercialScanner::SaveChanges (int *prefix*)

Saves the feature changes to files.

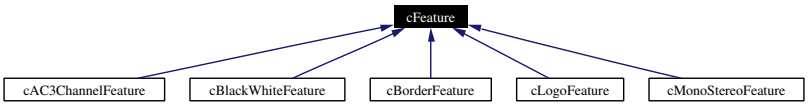
The documentation for this class was generated from the following files:

- scanner.h
- scanner.cpp

2.7 cFeature Class Reference

`#include <feature.h>`

Inheritance diagram for cFeature:



Public Methods

- cFeature ()
- virtual ~cFeature ()
- virtual void Refresh (void *frame)
- virtual char * Dump ()
- virtual bool Changed ()
- virtual int Type ()

2.7.1 Detailed Description

Base class for all features. A feature is a characteristic of the DVB-stream which helps to differentiate between commercial and ordinary programme. This can be characteristics in audio or video signals, as well as in the headers of the various streams.

2.7.2 Constructor & Destructor Documentation

2.7.2.1 cFeature::cFeature ()

Constructs the feature.

2.7.2.2 virtual cFeature::~cFeature () [virtual]

Destructs the feature. Actually does nothing.

2.7.3 Member Function Documentation

2.7.3.1 virtual bool cFeature::Changed () [virtual]

Retrieves whether the feature has changed at the last frame. This method has to be reimplemented by subclasses.

Returns:
true if feature has changed, false if not

2.7.3.2 virtual char* cFeature::Dump () [virtual]

Dumps the feature data to a string. Usefull for logging. This method has to be reimplemented by subclasses.

Returns:
null-terminated string containing the dump

Reimplemented in cBorderFeature, cLogoFeature, cBlackWhiteFeature, cAC3ChannelFeature, and cMonoStereoFeature.

2.7.3.3 virtual void cFeature::Refresh (void * *frame*) [virtual]

Refreshs the feature.

Parameters:
frame the frame, which data the feature is based on

Reimplemented in cBorderFeature, cLogoFeature, cBlackWhiteFeature, cAC3ChannelFeature, and cMonoStereoFeature.

2.7.3.4 virtual int cFeature::Type () [virtual]

Retrieves the feature's type.

Returns:
type of the feature (0 = none (default), 1 = PES, 2 = MPEG audio, 3 = MPEG video, 4 = AC3 audio, 5 = raw audio, 6 = raw video)

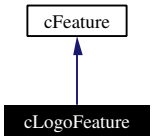
The documentation for this class was generated from the following files:

- feature.h
- feature.cpp

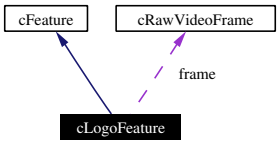
2.8 cLogoFeature Class Reference

`#include <feature.h>`

Inheritance diagram for cLogoFeature:



Collaboration diagram for cLogoFeature:



Public Methods

- cLogoFeature ()
- virtual ~cLogoFeature ()
- virtual void Refresh (void *frame)
- virtual char * Dump ()
- int LogoIndex ()
- tPosition Position ()

2.8.1 Detailed Description

A class representing logo detection. It detects the station logo using a cross-correlation between an edge picture of the current frame and the edge template of the station logo.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 cLogoFeature::cLogoFeature ()

Constructs the feature. Initializes the feature's data. Loads the different logos of different stations.

2.8.2.2 cLogoFeature::~cLogoFeature () [virtual]

Destructs the feature.

2.8.3 Member Function Documentation

2.8.3.1 char * cLogoFeature::Dump () [virtual]

Dumps the feature's data (logo name, logo position, luminance of the edge and the frame mask) to a string.

Returns:
null-terminated string containing the feature's data.

Reimplemented from cFeature.

2.8.3.2 int cLogoFeature::LogoIndex ()

Retrieves the logo index of the last refreshed frame.

Returns:
logo index of the last refreshed frame

2.8.3.3 tPosition cLogoFeature::Position ()

Retrieves the logo position of the last refreshed frame.

Returns:
logo position of the last refreshed frame

2.8.3.4 void cLogoFeature::Refresh (void * *frame*) [virtual]

Refreshs the feature's data. The logo is searched and if found new logo index and position will be set. If no logo is found logo index is set to -1. Then the new logo (index and position) is compared with the old one. If it has changed, the feature's state is set to "changed".

Parameters:
frame video frame in which the logo will be searched.

Reimplemented from cFeature.

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

- feature.h
- feature.cpp

2.9 cLogoTemplate Class Reference

`#include <feature.h>`

Public Methods

- cLogoTemplate (char *name, int station, char *edgefile, char *maskfile)
- ~cLogoTemplate ()
- void AddPosition (int x, int y)
- char * Name ()
- int Station ()
- int Width ()
- int Height ()
- int NormalFactor ()
- tPosition Position (int i)
- int PositionCount ()
- unsigned char * Data ()
- unsigned char * Mask ()

2.9.1 Detailed Description

A class representing a station logo. It consists of a name, a station ID, a edge template, a luminance mask and the positions, where the logo can occur.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 cLogoTemplate::cLogoTemplate (char * *name*, int *station*, char * *edgefile*, char * *maskfile*)

Constructs the logo template. Sets the name, station ID. Loads the edge-template and the luminance mask. Calculates the normal factor.

Parameters:

name null-terminated string containing the name of the logo

station station ID of the logo

edgefile null-terminated string containing the filename of the PGM file containing the edge template

maskfile null-terminated string containing the filename of the PGM file containing the luminance mask

2.9.2.2 cLogoTemplate::~cLogoTemplate ()

Destructs the logo template. Actually does nothing.

2.9.3 Member Function Documentation

2.9.3.1 void cLogoTemplate::AddPosition (int *x*, int *y*)

Adds a position where the logo can occur.

Parameters:

x x-coordinate

y y-coordinate

2.9.3.2 unsigned char* cLogoTemplate::Data ()

Retrieves the edge template of the logo.

Returns:

a pointer to the edge template of the logo

2.9.3.3 int cLogoTemplate::Height ()

Retrieves the height of the logo template.

Returns:

height of the logo template

2.9.3.4 unsigned char* cLogoTemplate::Mask ()

Retrieves the luminance mask of the logo.

Returns:

a pointer to the luminance mask of the logo

2.9.3.5 char* cLogoTemplate::Name ()

Retrieves the name of the logo.

Returns:

null-terminated string containing the name of the logo

2.9.3.6 int cLogoTemplate::NormalFactor ()

Retrieves the normal factor of the logo template. The normal factor is the number of white pixels the edge-template has.

Returns:
normal factor of the logo template

2.9.3.7 tPosition cLogoTemplate::Position (int *i*)

Retrieves the possible logo position of the given number.

Returns:
logo position of the given number

2.9.3.8 int cLogoTemplate::PositionCount ()

Retrieves the number of possible positions the logo has.

Returns:
number of possible logo positions

2.9.3.9 int cLogoTemplate::Station ()

Retrieves the station id of the logo.

Returns:
station ID

2.9.3.10 int cLogoTemplate::Width ()

Retrieves the width of the logo template.

Returns:
width of the logo template

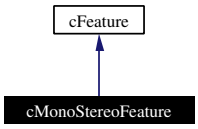
The documentation for this class was generated from the following files:

- feature.h
- feature.cpp

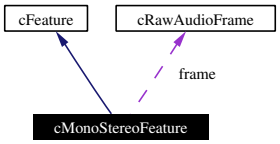
2.10 cMonoStereoFeature Class Reference

`#include <feature.h>`

Inheritance diagram for cMonoStereoFeature:



Collaboration diagram for cMonoStereoFeature:



Public Methods

- cMonoStereoFeature ()
- virtual ~cMonoStereoFeature ()
- virtual void Refresh (void *frame)
- virtual char * Dump ()
- int Channels ()

2.10.1 Detailed Description

A class representing detection of mono programmes, especially mono programmes sended as stereo.

2.10.2 Constructor & Destructor Documentation

2.10.2.1 cMonoStereoFeature::cMonoStereoFeature ()

Constructs the feature. Initializes the feature's data.

2.10.2.2 cMonoStereoFeature::~cMonoStereoFeature () [virtual]

Destructs the feature. Actually does nothing.

2.10.3 Member Function Documentation

2.10.3.1 int cMonoStereoFeature::Channels ()

Retrieves the channels of the last refreshed frame.

Returns:
channels of the last refreshed frame (1 = mono, 2 = stereo)

2.10.3.2 char * cMonoStereoFeature::Dump () [virtual]

Dumps the feature's data (channels) to a string.

Returns:
null-terminated string containing the feature's data.

Reimplemented from cFeature.

2.10.3.3 void cMonoStereoFeature::Refresh (void * *frame*) [virtual]

Refreshs the feature's data. First the channels of the current audio frame is checked. Then the new channels is compared with the old one. If it has changed, the feature's state is set to "changed".

Parameters:
frame audio frame which channels will be checked.

Reimplemented from cFeature.

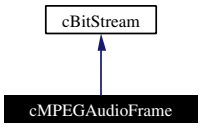
The documentation for this class was generated from the following files:

- feature.h
- feature.cpp

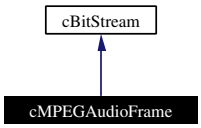
2.11 cMPEGAudioFrame Class Reference

`#include <audio.h>`

Inheritance diagram for cMPEGAudioFrame:



Collaboration diagram for cMPEGAudioFrame:



Public Methods

- cMPEGAudioFrame (unsigned char *data, unsigned int length)
- ~cMPEGAudioFrame ()
- cRawAudioFrame * Decode (AVCodecContext *c)
- int FrameSize ()
- int Version ()
- int Layer ()
- int BitRate ()
- int SamplingRate ()
- int Padding ()
- int ChannelMode ()
- int ModeExtension ()
- int Copyright ()
- int Original ()
- int Emphasis ()

2.11.1 Detailed Description

A class representing a compressed MPEG audio frame.

2.11.2 Constructor & Destructor Documentation

2.11.2.1 cMPEGAudioFrame::cMPEGAudioFrame (unsigned char * *data*, unsigned int *length*)

Constructs the audio frame. Parses the data and fills header structures.

Parameters:

data pointer to frame data

length length of frame data

2.11.2.2 cMPEGAudioFrame::~cMPEGAudioFrame ()

Destructs the audio frame. Actually does nothing.

2.11.3 Member Function Documentation

2.11.3.1 int cMPEGAudioFrame::BitRate ()

Retrieves the bitrate of the audio frame.

Returns:

bitrate of the audio frame in kbits per second

2.11.3.2 int cMPEGAudioFrame::ChannelMode ()

Retrieves the channel mode of the audio frame.

Returns:

channel mode of the audio frame (0 = Stereo, 1 = Joint Stereo, 2 = Dual Channel, 3 = Single Channel(Mono))

2.11.3.3 int cMPEGAudioFrame::Copyright ()

Retrieves the copyright bit of the audio frame.

Returns:

copyright bit of the audio frame

2.11.3.4 cRawAudioFrame * cMPEGAudioFrame::Decode (AVCodecContext * *c*)

Decodes the audio frame using libavcodec.

Returns:
the decompressed audio frame

2.11.3.5 int cMPEGAudioFrame::Emphasis ()

Retrieves the emphasis of the audio frame.

Returns:
emphasis of the audio frame (0 = none, 1 = 50/15ms, 3 = CCIT J.17)

2.11.3.6 int cMPEGAudioFrame::FrameSize ()

Retrieves the size of the audio frame.

Returns:
size of the audio frame in bytes

2.11.3.7 int cMPEGAudioFrame::Layer ()

Retrieves the MPEG Audio Layer of the audio frame.

Returns:
MPEG Audio Layer of the audio frame

2.11.3.8 int cMPEGAudioFrame::ModeExtension ()

Retrieves the channel mode extension of the audio frame.

Returns:
channel mode extension of the audio frame

2.11.3.9 int cMPEGAudioFrame::Original ()

Retrieves the original bit of the audio frame.

Returns:
original bit of the audio frame

2.11.3.10 int cMPEGAudioFrame::Padding ()

Retrieves the padding bit of the audio frame.

Returns:
padding bit of the audio frame

2.11.3.11 int cMPEGAudioFrame::SamplingRate ()

Retrieves the sampling rate of the audio frame.

Returns:
sampling rate of the audio frame

2.11.3.12 int cMPEGAudioFrame::Version ()

Retrieves the MPEG version of the audio frame.

Returns:
MPEG version of the audio frame (0 = MPEG 2.5, 2 = MPEG 2, 3 = MPEG 1)

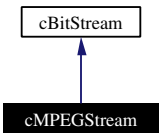
The documentation for this class was generated from the following files:

- audio.h
- audio.cpp

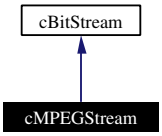
2.12 cMPEGStream Class Reference

`#include <pes.h>`

Inheritance diagram for cMPEGStream:



Collaboration diagram for cMPEGStream:



Public Methods

- cMPEGStream (unsigned char id, unsigned char *data, unsigned int length)
- ~cMPEGStream ()
- void SetPTS (unsigned long long pts)
- unsigned long long GetPTS ()
- unsigned char GetID ()
- unsigned int GetType ()
- int GetFrameCount ()
- cMPEGAudioFrame * GetAudioFrame (int frameNumber)
- cAC3AudioFrame * GetAC3AudioFrame (int frameNumber)
- cMPEGVideoFrame * GetVideoFrame (int frameType)

2.12.1 Detailed Description

A class representing a MPEG stream of one PES frame. This can be audio, video and private (AC3 audio) streams. It is a helper class used by cPESFrame for demuxing.

2.12.2 Constructor & Destructor Documentation

2.12.2.1 cMPEGStream::cMPEGStream (unsigned char *id*, unsigned char * *data*, unsigned int *length*)

Constructs the MPEG stream.

Parameters:

id stream ID

data pointer to the stream data

length length of the stream data

2.12.2.2 cMPEGStream::~cMPEGStream ()

Destructs the MPEG stream.

2.12.3 Member Function Documentation

2.12.3.1 cAC3AudioFrame * cMPEGStream::GetAC3AudioFrame (int *frameNumber*)

Retrieves the AC3 audio frame of the given number, according to the internal frame offset table.

Parameters:

frameNumber number of the frame to be retrieved (0 to (frame count - 1)). If no frame of this number exists, NULL will be returned.

Returns:

the retrieved AC3 audio frame, or NULL if an error occured

2.12.3.2 cMPEGAudioFrame * cMPEGStream::GetAudioFrame (int *frameNumber*)

Retrieves the MPEG audio frame of the given number, according to the internal frame offset table.

Parameters:

frameNumber number of the frame to be retrieved (0 to (frame count - 1)). If no frame of this number exists, NULL will be returned.

Returns:

the retrieved MPEG audio frame, or NULL if an error occured

2.12.3.3 int cMPEGStream::GetFrameCount ()

Retrieves the number of frames the stream contains. For video streams number of frames is always 1.

Returns:
number of frames

2.12.3.4 unsigned char cMPEGStream::GetID ()

Retrieves the stream ID.

Returns:
stream ID

2.12.3.5 unsigned long long cMPEGStream::GetPTS ()

Retrieves the PTS of the stream. PTS stands for Presentation Time Stamp.

Returns:
the PTS of the stream

2.12.3.6 unsigned int cMPEGStream::GetType ()

Retrieves the type of the stream.

Returns:
the type of the stream (0 = Unknown, 1 = Audio, 2 = Video, 3 = Private)

2.12.3.7 cMPEGVideoFrame * cMPEGStream::GetVideoFrame (int *frameType*)

Retrieves the MPEG video frame of the stream.

Parameters:
frameType type of the video frame (I, P, B, D)

Returns:
the retrieved MPEG video frame, or NULL if an error occured

2.12.3.8 void cMPEGStream::SetPTS (unsigned long long *pts*)

Sets the PTS of the stream. PTS stands for Presentation Time Stamp.

Parameters:

pts new PTS

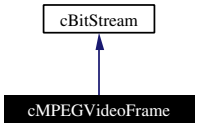
The documentation for this class was generated from the following files:

- pes.h
- pes.cpp

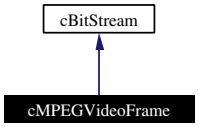
2.13 cMPEGVideoFrame Class Reference

`#include <video.h>`

Inheritance diagram for cMPEGVideoFrame:



Collaboration diagram for cMPEGVideoFrame:



Public Methods

- cMPEGVideoFrame (unsigned char *data, unsigned int length, int frameType)
- ~cMPEGVideoFrame ()
- cRawVideoFrame * Decode ()
- int FrameType ()
- tSequenceHeader * SequenceHeader ()
- tSequenceExtension * SequenceExtension ()
- tSequenceDisplayExtension * SequenceDisplayExtension ()
- tSequenceScalableExtension * SequenceScalableExtension ()
- tGroupOfPicturesHeader * GroupOfPicturesHeader ()
- tPictureHeader * PictureHeader ()
- tPictureCodingExtension * PictureCodingExtension ()
- tQuantMatrixExtension * QuantMatrixExtension ()
- tPictureDisplayExtension * PictureDisplayExtension ()
- tPictureTemporalScalableExtension * PictureTemporalScalableExtension ()
- tPictureSpatialScalableExtension * PictureSpatialScalableExtension ()
- tCopyrightExtension * CopyrightExtension ()

2.13.1 Detailed Description

A class representing a compressed MPEG video frame.

2.13.2 Constructor & Destructor Documentation

2.13.2.1 cMPEGVideoFrame::cMPEGVideoFrame (unsigned char * *data*, unsigned int *length*, int *frameType*)

Constructs the video frame. Parses the data and fills header structures.

Parameters:

data pointer to frame data

length length of frame data

frameType type of frame (I, P, B, D)

2.13.2.2 cMPEGVideoFrame::~cMPEGVideoFrame ()

Destructs the video frame. Actually does nothing.

2.13.3 Member Function Documentation

2.13.3.1 tCopyrightExtension* cMPEGVideoFrame::CopyrightExtension ()

Retrieves the copyright extension data of the video frame.

Returns:

structure of type tCopyrightExtension containing the data

2.13.3.2 cRawVideoFrame * cMPEGVideoFrame::Decode ()

Decodes the video frame using libavcodec. This only works for I-Frames.

Returns:

a pointer to the decompressed video frame

2.13.3.3 int cMPEGVideoFrame::FrameType ()

Retrieves the type of the video frame.

Returns:

frame type (1 = I-Frame, 2 = P-Frame, 3 = B-Frame, 4 = D-Frame)

2.13.3.4 tGroupOfPicturesHeader* cMPEGVideoFrame::GroupOfPicturesHeader ()

Retrieves the group of pictures header data of the video frame.

Returns:
structure of type tGroupOfPicturesHeader containing the data

2.13.3.5 tPictureCodingExtension* cMPEGVideoFrame::PictureCoding-Extension ()

Retrieves the picture coding extension data of the video frame.

Returns:
structure of type tPictureCodingExtension containing the data

2.13.3.6 tPictureDisplayExtension* cMPEGVideoFrame::PictureDisplay-Extension ()

Retrieves the picture display extension data of the video frame.

Returns:
structure of type tPictureDisplayExtension containing the data

2.13.3.7 tPictureHeader* cMPEGVideoFrame::PictureHeader ()

Retrieves the picture header data of the video frame.

Returns:
structure of type tPictureHeader containing the data

2.13.3.8 tPictureSpatialScalableExtension* cMPEGVideoFrame::PictureSpatial-ScalableExtension ()

Retrieves the picture spatial scalable extension data of the video frame.

Returns:
structure of type tPictureSpatialScalableExtension containing the data

2.13.3.9 tPictureTemporalScalableExtension* cMPEGVideoFrame::PictureTemporalScalableExtension ()

Retrieves the picture temporal scalable extension data of the video frame.

Returns:
structure of type tPictureTemporalScalableExtension containing the data

2.13.3.10 tQuantMatrixExtension* cMPEGVideoFrame::QuantMatrixExtension ()

Retrieves the quant matrix extension data of the video frame.

Returns:
structure of type tQuantMatrixExtension containing the data

2.13.3.11 tSequenceDisplayExtension* cMPEGVideoFrame::SequenceDisplayExtension ()

Retrieves the sequence display extension data of the video frame.

Returns:
structure of type tSequenceDisplayExtension containing the data

2.13.3.12 tSequenceExtension* cMPEGVideoFrame::SequenceExtension ()

Retrieves the sequence extension data of the video frame.

Returns:
structure of type tSequenceExtension containing the data

2.13.3.13 tSequenceHeader* cMPEGVideoFrame::SequenceHeader ()

Retrieves the sequence header data of the video frame.

Returns:
structure of type tSequenceHeader containing the data

2.13.3.14 tSequenceScalableExtension* cMPEGVideoFrame::SequenceScalableExtension ()

Retrieves the sequence scalable extension data of the video frame.

Returns:

structure of type tSequenceScalableExtension containing the data

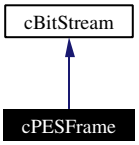
The documentation for this class was generated from the following files:

- video.h
- video.cpp

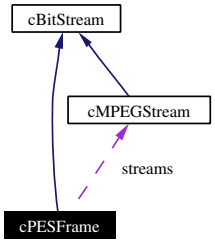
2.14 cPESFrame Class Reference

`#include <pes.h>`

Inheritance diagram for cPESFrame:



Collaboration diagram for cPESFrame:



Public Methods

- cPESFrame (unsigned char *data, unsigned int length, int frameType)
- ~cPESFrame ()
- int GetStreamCount ()
- int GetAudioStreamCount ()
- int GetVideoStreamCount ()
- int GetPrivateStreamCount ()
- unsigned char GetStreamID (int streamNumber)
- int GetAudioFrameCount (int streamNumber)
- int GetAC3AudioFrameCount (int streamNumber)
- cMPEGAudioFrame * GetAudioFrame (int streamNumber, int frameNumber)
- cAC3AudioFrame * GetAC3AudioFrame (int streamNumber, int frameNumber)
- cMPEGVideoFrame * GetVideoFrame (int streamNumber)
- unsigned long long GetAudioPTS (int streamNumber)
- unsigned long long GetVideoPTS (int streamNumber)
- unsigned long long GetPrivatePTS (int streamNumber)

2.14.1 Detailed Description

A class representing a MPEG PES frame. PES stands for Packetized Elementary Stream.

2.14.2 Constructor & Destructor Documentation

2.14.2.1 cPESFrame::cPESFrame (unsigned char * *data*, unsigned int *length*, int *frameType*)

Constructs the PES frame. The data is parsed, the streams are demuxed and corresponding cMPEGStream objects are created.

Parameters:

data pointer to the stream data

length length of the stream data

frameType type of the video frame the PES frame contains

2.14.2.2 cPESFrame::~cPESFrame ()

Destructs the PES frame.

2.14.3 Member Function Documentation

2.14.3.1 cAC3AudioFrame * cPESFrame::GetAC3AudioFrame (int *streamNumber*, int *frameNumber*)

Retrieves the AC3 audio frame of the given number of the given stream.

Parameters:

streamNumber number of the private stream that contains the frame to be retrieved. If no stream of this number exists, NULL will be returned.

frameNumber number of the frame to be retrieved. If no frame of this number exists, NULL will be returned.

Returns:

the retrieved AC3 audio frame, or NULL if an error occured

2.14.3.2 int cPESFrame::GetAC3AudioFrameCount (int *streamNumber*)

Retrieves the number of frames the AC3 audio stream of the given number contains.

Parameters:

streamNumber number of the AC3 audio stream which frame count has to be retrieved. If no stream of this number exists, 0 will be returned.

Returns:

number of frames the AC3 audio stream contains, or 0 if streamNumber is invalid

2.14.3.3 cMPEGAudioFrame * cPESFrame::GetAudioFrame (int *streamNumber*, int *frameNumber*)

Retrieves the MPEG audio frame of the given number of the given stream.

Parameters:

streamNumber number of the audio stream that contains the frame to be retrieved. If no stream of this number exists, NULL will be returned.

frameNumber number of the frame to be retrieved. If no frame of this number exists, NULL will be returned.

Returns:

the retrieved MPEG audio frame, or NULL if an error occured

2.14.3.4 int cPESFrame::GetAudioFrameCount (int *streamNumber*)

Retrieves the number of frames the audio stream of the given number contains.

Parameters:

streamNumber number of the audio stream which frame count has to be retrieved. If no stream of this number exists, 0 will be returned.

Returns:

number of frames the audio stream contains, or 0 if streamNumber is invalid

2.14.3.5 unsigned long long cPESFrame::GetAudioPTS (int *streamNumber*)

Retrieves the PTS of the given audio stream. PTS stands for Presentation Time Stamp.

Parameters:

streamNumber number of the audio stream that PTS has to be retrieved. If no stream of this number exists, 0 will be returned.

Returns:

the PTS of the audio stream, or 0 if streamNumber is invalid

2.14.3.6 int cPESFrame::GetAudioStreamCount ()

Retrieves the number of audio streams the PES frame contains.

Returns:
number of audio streams

2.14.3.7 unsigned long long cPESFrame::GetPrivatePTS (int *streamNumber*)

Retrieves the PTS of the given private stream. PTS stands for Presentation Time Stamp.

Parameters:
streamNumber number of the private stream that PTS has to be retrieved. If no stream of this number exists, 0 will be returned.

Returns:
the PTS of the private stream, or 0 if streamNumber is invalid

2.14.3.8 int cPESFrame::GetPrivateStreamCount ()

Retrieves the number of private streams the PES frame contains.

Returns:
number of private streams

2.14.3.9 int cPESFrame::GetStreamCount ()

Retrieves the number of streams the PES frame contains. This includes audio, video and private streams.

Returns:
number of streams

2.14.3.10 unsigned char cPESFrame::GetStreamID (int *streamNumber*)

Retrieves the ID of the stream of the given number.

Parameters:
streamNumber number of the stream (0 to (stream count - 1)). If no stream of this number exists, 0 will be returned.

Returns:
the retrieved stream ID, or 0 if streamNumber is invalid

2.14.3.11 cMPEGVideoFrame * cPESFrame::GetVideoFrame (int *streamNumber*)

Retrieves the MPEG video frame of the given stream.

Parameters:

streamNumber number of the video stream that contains the frame to be retrieved. If no stream of this number exists, NULL will be returned.

Returns:

the retrieved MPEG video frame, or NULL if an error occured

2.14.3.12 unsigned long long cPESFrame::GetVideoPTS (int *streamNumber*)

Retrieves the PTS of the given video stream. PTS stands for Presentation Time Stamp.

Parameters:

streamNumber number of the video stream that PTS has to be retrieved. If no stream of this number exists, 0 will be returned.

Returns:

the PTS of the video stream, or 0 if streamNumber is invalid

2.14.3.13 int cPESFrame::GetVideoStreamCount ()

Retrieves the number of video streams the PES frame contains.

Returns:

number of video streams

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

- pes.h
- pes.cpp

2.15 cRawAudioFrame Class Reference

`#include <audio.h>`

Public Methods

- cRawAudioFrame (unsigned char *data, unsigned int length, int channels)
- ~cRawAudioFrame ()
- unsigned char * Data ()
- unsigned int Length ()
- int Channels ()

2.15.1 Detailed Description

A class representing an uncompressed audio frame.

2.15.2 Constructor & Destructor Documentation

2.15.2.1 cRawAudioFrame::cRawAudioFrame (unsigned char * *data*, unsigned int *length*, int *channels*)

Constructs the audio frame.

Parameters:

data pointer to frame data

length length of frame data

channels number of channels of the compressed audio frame

2.15.2.2 cRawAudioFrame::~cRawAudioFrame ()

Destructs the audio frame.

2.15.3 Member Function Documentation

2.15.3.1 int cRawAudioFrame::Channels ()

Retrieves the number of channels of the compressed audio frame.

Returns:

number of channels of the compressed audio frame

2.15.3.2 unsigned char* cRawAudioFrame::Data ()

Retrieves the data of the audio frame.

Returns:
a pointer to the data of the audio frame

2.15.3.3 unsigned int cRawAudioFrame::Length ()

Retrieves the length of the audio frame.

Returns:
length in bytes

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

- audio.h
- audio.cpp

2.16 cRawVideoFrame Class Reference

`#include <video.h>`

Public Methods

- cRawVideoFrame (AVFrame *picture, int width, int height)
- ~cRawVideoFrame ()
- unsigned char * PictureData (int i)
- int LineSize (int i)
- int Width ()
- int Height ()

2.16.1 Detailed Description

A class representing an uncompressed video frame. The data is stored in YUV format.

2.16.2 Constructor & Destructor Documentation

2.16.2.1 cRawVideoFrame::cRawVideoFrame (AVFrame * *picture*, int *width*, int *height*)

Constructs the video frame. Picture data is copied.

Parameters:

picture pointer to picture structure filled by decoder

width width of picture in pixels

height height of picture in pixels

2.16.2.2 cRawVideoFrame::~cRawVideoFrame ()

Destructs the video frame.

2.16.3 Member Function Documentation

2.16.3.1 int cRawVideoFrame::Height ()

Returns the height of the video frame.

Returns:

height of video frame in pixels

2.16.3.2 int cRawVideoFrame::LineSize (int *i*)

Returns the length of a line of component i in memory.

Parameters:

i number of component (0 = Y, 1 = U, 2 = V)

Returns:

length in bytes

2.16.3.3 unsigned char * cRawVideoFrame::PictureData (int *i*)

Retrieves the data of component i.

Parameters:

i number of component (0 = Y, 1 = U, 2 = V)

Returns:

a pointer to the data of component i

2.16.3.4 int cRawVideoFrame::Width ()

Returns the width of the video frame.

Returns:

width of video frame in pixels

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

- video.h
- video.cpp