psyphibio-proc

Generated by Doxygen 1.8.2

Mon Dec 10 2012 13:05:51

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Adder Class Reference

Public Member Functions

- Adder ()
- int Add (int a, int b)
- int Add (int a)
- void SetDefAddNo (int defAddNo)

Private Attributes

- int _defAddNo

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Adder::Adder ( ) [inline]

4.1.2 Member Function Documentation

4.1.2.1 int Adder::Add (int *a*, int *b*) [inline]

4.1.2.2 int Adder::Add (int *a*) [inline]

4.1.2.3 void Adder::SetDefAddNo (int *defAddNo*) [inline]

4.1.3 Member Data Documentation

4.1.3.1 int Adder::_defAddNo [private]

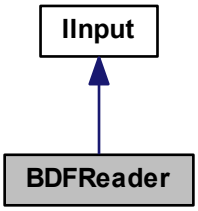
The documentation for this class was generated from the following file:

- source/test_main.cpp

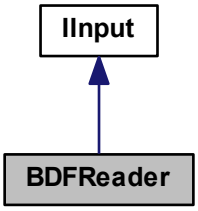
4.2 BDFReader Class Reference

```
#include <BDFReader.h>
```

Inheritance diagram for BDFReader:



Collaboration diagram for BDFReader:



Public Member Functions

- BDFReader ()
- double * getSignal (int &noSamples, int &smpFreq)
- double getNextDataPoint ()
- void Open (string fileName, int channel)
- void Close ()

Private Member Functions

- void PrintHeader (int nChannel)
- void ReadData (int nChannel)

Private Attributes

- struct edf_hdr_struct hdr
- struct tm * datetime
- float samplingFreq
- double * buf
- int channelNo
- int sampleCounter

4.2.1 Constructor & Destructor Documentation

4.2.1.1 BDFReader::BDFReader ( )

4.2.2 Member Function Documentation

4.2.2.1 void BDFReader::Close ( ) [virtual]

Implements IInput.

4.2.2.2 double BDFReader::getNextDataPoint ( ) [virtual]

Implements IInput.

4.2.2.3 double * BDFReader::getSignal (int & *noSamples,* int & *smpFreq*) [virtual]

Implements IInput.

4.2.2.4 void BDFReader::Open (string *fileName,* int *channel*) [virtual]

Implements IInput.

4.2.2.5 void BDFReader::PrintHeader (int *nChannel*) [private]

4.2.2.6 void BDFReader::ReadData (int *nChannel*) [private]

4.2.3 Member Data Documentation

4.2.3.1 double* BDFReader::buf [private]

4.2.3.2 int BDFReader::channelNo [private]

4.2.3.3 struct tm* BDFReader::datetime [private]

4.2.3.4 struct edf_hdr_struct BDFReader::hdr [private]

4.2.3.5 int BDFReader::sampleCounter [private]

4.2.3.6 float BDFReader::samplingFreq [private]

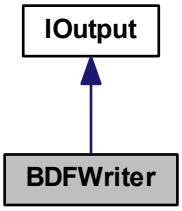
The documentation for this class was generated from the following files:

- include/BDFReader.h
- source/BDFReader.cpp

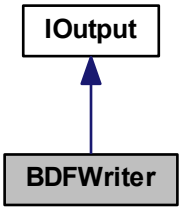
4.3 BDFWriter Class Reference

```
#include <BDFWriter.h>
```

Inheritance diagram for BDFWriter:



Collaboration diagram for BDFWriter:



Public Member Functions

- BDFWriter ()
- void saveSignal (double ∗sig, int noSamp, int smp_freq, string sigLabel, string sigDim)
- void saveNextDataPoint ()
- void DisplayMessage (string message)
- void DisplayHistogram (deque< int > histogram)

Private Attributes

- deque< double ∗ > signals
- deque< int > smpFreqs
- deque< int > nosSamples
- deque< string > sigLabels
- deque< string > sigDim

4.3.1 Constructor & Destructor Documentation

4.3.1.1 BDFWriter::BDFWriter ( )

4.3.2 Member Function Documentation

4.3.2.1 void BDFWriter::DisplayHistogram (deque< int > *histogram*) [virtual]

Implements IOutput.

4.3.2.2 void BDFWriter::DisplayMessage (string *message*) [virtual]

Implements IOutput.

4.3.2.3 void BDFWriter::saveNextDataPoint ( ) [virtual]

Implements IOutput.

4.3.2.4 void BDFWriter::saveSignal (double * *sig*, int *noSamp*, int *smp_freq*, string *sigLabel*, string *sigDim*) [virtual]

Implements IOutput.

4.3.3 Member Data Documentation

4.3.3.1 deque<int> BDFWriter::nosSamples [private]

4.3.3.2 deque<string> BDFWriter::sigDim [private]

4.3.3.3 deque<string> BDFWriter::sigLabels [private]

4.3.3.4 deque<double*> BDFWriter::signals [private]

4.3.3.5 deque<int> BDFWriter::smpFreqs [private]

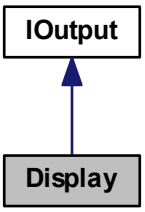
The documentation for this class was generated from the following files:

- include/BDFWriter.h
- source/BDFWriter.cpp

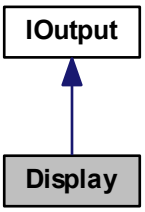
4.4 Display Class Reference

```
#include <Display.h>
```

Inheritance diagram for Display:



Collaboration diagram for Display:



Public Member Functions

- void saveSignal (double ∗sig, int noSamp, int smp_freq, string sigLabel, string sigDim)
- void saveNextDataPoint ()
- void DisplayMessage (string message)
- void DisplayHistogram (deque< int > histogram)

4.4.1 Member Function Documentation

4.4.1.1 void Display::DisplayHistogram (deque< int > *histogram*) [virtual]

Implements IOutput.

4.4.1.2 void Display::DisplayMessage (string *message*) [virtual]

Implements IOutput.

4.4.1.3 void Display::saveNextDataPoint ( ) [virtual]

Implements IOutput.

4.4.1.4 void Display::saveSignal (double * *sig*, int *noSamp*, int *smp_freq*, string *sigLabel*, string *sigDim*) [virtual]

Implements IOutput.

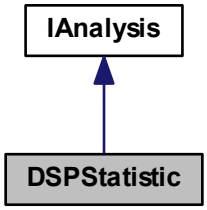
The documentation for this class was generated from the following files:

- include/Display.h
- source/Display.cpp

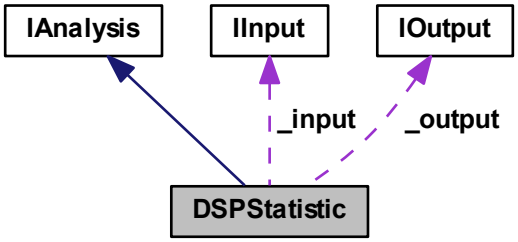
4.5 DSPStatistic Class Reference

`#include <DSPStatistic.h>`

Inheritance diagram for DSPStatistic:



Collaboration diagram for DSPStatistic:



Public Member Functions

- void setInput (IInput *input)
- void setOutput (IOutput *output)
- void Process ()
- DSPStatistic ()
- DSPStatistic (IInput *input, IOutput *output)

Private Member Functions

- void ProcesInput ()
- void ProcesSignal ()
- void CalculateDescriptivesStatic (double ∗sig, int noSamp, double &avg, double &stdev, double &var, double &snr, double &cv)
- void CalculateDescriptivesRunning (double &avg, double &stdev, double &var, double &snr, double &cv)
- deque< int > CalculateDescriptivesStaticHistogram (double ∗sig, int noSamp, double &avg, double &stdev, double &var, double &snr, double &cv)
- void ConvolutionInputSide (double ∗sig, int noSamp, double ∗impulseResponse, int irSize, double ∗∗outputSignal, int &osSize)
- void ConvolutionOutputSide (double ∗sig, int noSamp, double ∗impulseResponse, int irSize, double ∗∗outputSignal, int &osSize)
- void FirstDifference (double ∗sig, int noSamp, double ∗∗outputSignal, int &osSize)
- void RunningSumIntegration (double ∗sig, int noSamp, double ∗∗outputSignal, int &osSize)
- void DiscreteFourierTransformationTimeSide (double ∗timeSignal, int timeSigSize, double ∗∗realX, int &reXSize, double ∗∗imaginaryX, int &imXSize)
- void DiscreteFourierTransformationFreqSide (double ∗timeSignal, int timeSigSize, double ∗∗realX, int &reXSize, double ∗∗imaginaryX, int &imXSize)
- void InverseDiscreteFourierTransformationFreqSide (double ∗realX, int reXSize, double ∗imaginaryX, int imXSize, double ∗∗timeSignal, int &timeSigSize)
- void InverseDiscreteFourierTransformationTimeSide (double ∗realX, int reXSize, double ∗imaginaryX, int imXSize, double ∗∗timeSignal, int &timeSigSize)
- void RectangularToPolarConversion (double ∗realX, int reXSize, double ∗imaginaryX, int imXSize, double ∗∗mag, int &magSize, double ∗∗phase, int &phaseSize)
- void PolarToRectangularConversion (double ∗mag, int magSize, double ∗phase, int phaseSize, double ∗∗realX, int &reXSize, double ∗∗imaginaryX, int &imXSize)
- void PhaseUnwrapping (double ∗phase, int phaseSize, double ∗∗uwPhase, int &uwPhaseSize)

Private Attributes

- IInput ∗ _input
- IOutput ∗ _output
- double ∗ signal
- int noSamples
- int smpFreq

4.5.1 Constructor & Destructor Documentation

4.5.1.1 DSPStatistic::DSPStatistic ( )

4.5.1.2 DSPStatistic::DSPStatistic (IInput ∗ *input,* IOutput ∗ *output*)

4.5.2 Member Function Documentation

4.5.2.1 void DSPStatistic::CalculateDescriptivesRunning (double & *avg,* double & *stdev,* double & *var,* double & *snr,* double & *cv*) [private]

4.5.2.2 void DSPStatistic::CalculateDescriptivesStatic (double ∗ *sig,* int *noSamp,* double & *avg,* double & *stdev,* double & *var,* double & *snr,* double & *cv*) [private]

4.5.2.3 deque< int > DSPStatistic::CalculateDescriptivesStaticHistogram (double ∗ *sig,* int *noSamp,* double & *avg,* double & *stdev,* double & *var,* double & *snr,* double & *cv*) [private]

4.5.2.4 void DSPStatistic::ConvolutionInputSide (double * *sig*, int *noSamp*, double * *impulseResponse*, int *irSize*, double ** *outputSignal*, int & *osSize*) [private]

4.5.2.5 void DSPStatistic::ConvolutionOutputSide (double * *sig*, int *noSamp*, double * *impulseResponse*, int *irSize*, double ** *outputSignal*, int & *osSize*) [private]

4.5.2.6 void DSPStatistic::DiscreteFourierTransformationFreqSide (double * *timeSignal*, int *timeSigSize*, double ** *realX*, int & *reXSize*, double ** *imaginaryX*, int & *imXSize*) [private]

Decompose a time domain signal into sinusoids in frequency domain signals from frequency side. Frequency domain signals, held in *realX* and *imaginaryX*, are calculated from the time domain signal, held in *timeSignal*.

Parameters

in	*timeSignal*	Will hold the time domain signal
in	*timeSigSize*	Will hold the time domain signal size
out	*realX*	Holds the real part of the frequency domain
out	*reXSize*	Holds the real part of the frequency domain size
out	*imaginaryX*	Holds the imaginary part of the frequency domain
out	*imXSize*	Holds the imaginary part of the frequency domain size

Decompose a time domain signal into sinusoids in frequency domain signals from time side. The program loops through each sample in the time domain, calculating the contribution of that point to the frequency domain. The overall frequency domain is found by adding the contributions from the individual time domain points.

Author

Petar Jerčić

Date

09/12/2012 Check correct sizes of the pek anh imk arrays.

Alocate array for real and imaginary parts of frequency domain

Zero out the *realX* and *imaginaryX* so it can be used as accumulators

Decomposition method loops through each sample in the time domain *timeSignal*, calculating the contribution of that point to the frequency domain.

4.5.2.7 void DSPStatistic::DiscreteFourierTransformationTimeSide (double * *timeSignal*, int *timeSigSize*, double ** *realX*, int & *reXSize*, double ** *imaginaryX*, int & *imXSize*) [private]

Decompose a time domain signal into sinusoids in frequency domain signals from time side. Frequency domain signals, held in *realX* and *imaginaryX*, are calculated from the time domain signal, held in *timeSignal*.

Parameters

in	*timeSignal*	Will hold the time domain signal
in	*timeSigSize*	Will hold the time domain signal size
out	*realX*	Holds the real part of the frequency domain
out	*reXSize*	Holds the real part of the frequency domain size
out	*imaginaryX*	Holds the imaginary part of the frequency domain
out	*imXSize*	Holds the imaginary part of the frequency domain size

Decompose a time domain signal into sinusoids in frequency domain signals from time side. Describes how an individual sample in the frequency domain is affected by all of the samples in the time domain. That is, the program calculates each of the values in the frequency domain in succession, not as a group.

Author

Petar Jerčić

Date

09/12/2012 Check correct sizes of the pek anh imk arrays.

Alocate array for real and imaginary parts of frequency domain

Zero out the *realX* and *imaginaryX* so it can be used as accumulators

Decomposition method Correlate *timeSignal* with cosine and sine waves

4.5.2.8 void DSPStatistic::FirstDifference (double * *sig,* int *noSamp,* double ** *outputSignal,* int & *osSize*) [private]

4.5.2.9 void DSPStatistic::InverseDiscreteFourierTransformationFreqSide (double * *realX,* int *reXSize,* double * *imaginaryX,* int *imXSize,* double ** *timeSignal,* int & *timeSigSize*) [private]

Synthesis of time domain signal from frequency domain signals from frequency side. Time domain signal, held in *timeSignal*, is calculated from the frequency domain signal, held in *realX* and *imaginaryX*

Parameters

in	*realX*	Holds the real part of the frequency domain
in	*reXSize*	Holds the real part of the frequency domain size
in	*imaginaryX*	Holds the imaginary part of the frequency domain
in	*imXSize*	Holds the imaginary part of the frequency domain size
out	*timeSignal*	Will hold the time domain signal
out	*timeSigSize*	Will hold the time domain signal size

Synthesis of time domain signal from frequency domain signals from frequency side. Each of the scaled sinusoids are generated one at a time and added to an accumulation array, which ends up becoming the time domain signal.

Author

Petar Jerčić

Date

09/12/2012 Check correct sizes of the pek anh imk arrays.

Alocate array for time domain signal (minus two extra data points)

Alocate array for cosine and sine wave amplitudes

Find the cosine and sine wave amplitudes

Cover two special cases

Zero out the *timeSignal* so it can be used as accumulator

Synthesis method Loop through each frequency generating the entire length of the sine and cosine waves, and add them to the accumulator signal *timeSignal*

4.5.2.10 void DSPStatistic::InverseDiscreteFourierTransformationTimeSide (double * *realX,* int *reXSize,* double * *imaginaryX,* int *imXSize,* double ** *timeSignal,* int & *timeSigSize*) [private]

Synthesis of time domain signal from frequency domain signals from time side. Time domain signal, held in *timeSignal*, is calculated from the frequency domain signal, held in *realX* and *imaginaryX*

Parameters

in	*realX*	Holds the real part of the frequency domain
in	*reXSize*	Holds the real part of the frequency domain size
in	*imaginaryX*	Holds the imaginary part of the frequency domain
in	*imXSize*	Holds the imaginary part of the frequency domain size
out	*timeSignal*	Will hold the time domain signal
out	*timeSigSize*	Will hold the time domain signal size

Synthesis of time domain signal from frequency domain signals from time side. Each sample in the time domain signal is calculated one at a time, as the sum of all the corresponding samples in the cosine and sine waves.

Author

Petar Jerčić

Date

09/12/2012 N/A.

Alocate array for time domain signal (minus two extra data points)

Alocate array for cosine and sine wave amplitudes

Find the cosine and sine wave amplitudes

Cover two special cases

Zero out the *timeSignal* so it can be used as accumulator

Synthesis method Loop through each sample in the time domain and sum the corresponding samples from each cosine and sine waves.

4.5.2.11 void DSPStatistic::PhaseUnwrapping (double * *phase,* int *phaseSize,* double ** *uwPhase,* int & *uwPhaseSize*) [private]

Unwrapping the phase It is often easier to understand the phase if it does not have these discontinuities, even if it means that the phase extends above , or below -.

Parameters

in	*phase*	Holds the phase
in	*phaseSize*	Holds the phase size
out	*uwPhase*	Holds the unwrapped phase
out	*uwPhaseSize*	Holds the unwrapped phase size

Unwrapping the phase. It is often easier to understand the phase if it does not have these discontinuities, even if it means that the phase extends above , or below -. a multiple of 2 is added or subtracted from each value of the phase.

Author

Petar Jerčić

Date

09/12/2012 N/A.

Alocate array for magnitude and phase parts of frequency domain

First point of all phase signals is 0

Go through the unwrappnig algorithm

4.5.2.12 void DSPStatistic::PolarToRectangularConversion (double * *mag*, int *magSize*, double * *phase*, int *phaseSize*, double ** *realX*, int & *reXSize*, double ** *imaginaryX*, int & *imXSize*) [private]

Polar To Rectangular Conversion From magnitude and phase to real and imginary

Parameters

in	*mag*	Holds the magnitude
in	*magSize*	Holds the magnitude size
in	*phase*	Holds the phase
in	*phaseSize*	Holds the phase size
out	*realX*	Holds the real part of the frequency domain
out	*reXSize*	Holds the real part of the frequency domain size
out	*imaginaryX*	Holds the imaginary part of the frequency domain
out	*imXSize*	Holds the imaginary part of the frequency domain size

Polar To Rectangular Conversion From magnitude and phase to real and imginary

Author

Petar Jerčić

Date

09/12/2012 N/A.

Alocate array for magnitude and phase parts of frequency domain

Polar To Rectangular Conversion

4.5.2.13 void DSPStatistic::ProcesInput ( ) [private]

4.5.2.14 void DSPStatistic::Process ( ) [virtual]

Implements IAnalysis.

4.5.2.15 void DSPStatistic::ProcesSignal ( ) [private]

4.5.2.16 void DSPStatistic::RectangularToPolarConversion (double * *realX*, int *reXSize*, double * *imaginaryX*, int *imXSize*, double ** *mag*, int & *magSize*, double ** *phase*, int & *phaseSize*) [private]

Rectangular to polar Conversion From real and imginary to magnitude and phase

Parameters

in	*realX*	Holds the real part of the frequency domain
in	*reXSize*	Holds the real part of the frequency domain size
in	*imaginaryX*	Holds the imaginary part of the frequency domain
in	*imXSize*	Holds the imaginary part of the frequency domain size
out	*mag*	Holds the magnitude
out	*magSize*	Holds the magnitude size
out	*phase*	Holds the phase
out	*phaseSize*	Holds the phase size

Rectangular to polar Conversion. From real and imginary to magnitude and phase.

Author

Petar Jerčić

Date

09/12/2012 N/A.

Alocate array for magnitude and phase parts of frequency domain

Rectangular to polar conversion

Prevent divide by zero

Correct arcus tangens ambiguity

4.5.2.17 void DSPStatistic::RunningSumIntegration (double * *sig*, int *noSamp*, double ** *outputSignal*, int & *osSize*) [private]

4.5.2.18 void DSPStatistic::setInput (IInput * *input*) [virtual]

This method will be used to create random noise signal for other components to use.

Author

Petar Jerčić

Parameters

fileName	N/A
channel	N/A

Returns

N/A

See Also

something

Date

08/12/2012

Implements IAnalysis.

4.5.2.19 void DSPStatistic::setOutput (IOutput * *output*) [virtual]

Implements IAnalysis.

4.5.3 Member Data Documentation

4.5.3.1 IInput* DSPStatistic::_input [private]

4.5.3.2 IOutput* DSPStatistic::_output [private]

4.5.3.3 int DSPStatistic::noSamples [private]

4.5.3.4 double* DSPStatistic::signal [private]

4.5.3.5 int DSPStatistic::smpFreq [private]

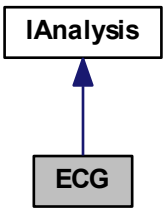
The documentation for this class was generated from the following files:

- include/DSPStatistic.h
- source/DSPStatistic.cpp

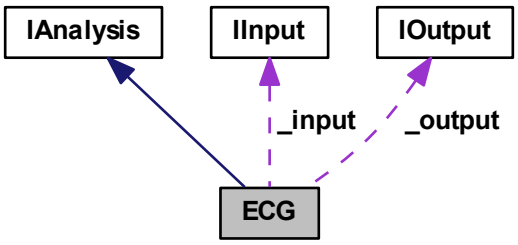
4.6 ECG Class Reference

`#include <ECG.h>`

Inheritance diagram for ECG:



Collaboration diagram for ECG:



Public Member Functions

- void setInput (IInput *input)
- void setOutput (IOutput *output)
- void Process ()

- ECG ()
- ECG (IInput *input, IOutput *output)

Private Member Functions

- void ProcesInput ()
- void ProcesSignal ()

Private Attributes

- IInput * _input
- IOutput * _output

4.6.1 Constructor & Destructor Documentation

4.6.1.1 ECG::ECG ( )

4.6.1.2 ECG::ECG (IInput * *input*, IOutput * *output*)

4.6.2 Member Function Documentation

4.6.2.1 void ECG::ProcesInput ( ) [private]

4.6.2.2 void ECG::Process ( ) [virtual]

Implements IAnalysis.

4.6.2.3 void ECG::ProcesSignal ( ) [private]

4.6.2.4 void ECG::setInput (IInput * *input*) [virtual]

Implements IAnalysis.

4.6.2.5 void ECG::setOutput (IOutput * *output*) [virtual]

Implements IAnalysis.

4.6.3 Member Data Documentation

4.6.3.1 IInput* ECG::_input [private]

4.6.3.2 IOutput* ECG::_output [private]

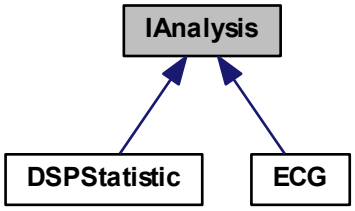
The documentation for this class was generated from the following files:

- include/ECG.h
- source/ECG.cpp

4.7 IAnalysis Class Reference

`#include <IAnalysis.h>`

Inheritance diagram for IAnalysis:



Public Member Functions

- virtual void setInput (IInput *input)=0
- virtual void setOutput (IOutput *input)=0
- virtual void Process ()=0

4.7.1 Member Function Documentation

4.7.1.1 virtual void IAnalysis::Process ( ) [pure virtual]

Implemented in DSPStatistic, and ECG.

4.7.1.2 virtual void IAnalysis::setInput (IInput * *input*) [pure virtual]

Implemented in DSPStatistic, and ECG.

4.7.1.3 virtual void IAnalysis::setOutput (IOutput * *input*) [pure virtual]

Implemented in DSPStatistic, and ECG.

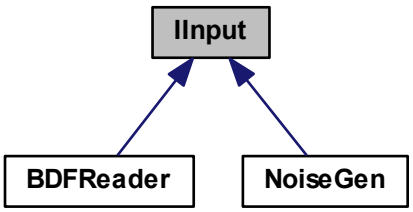
The documentation for this class was generated from the following file:

- include/IAnalysis.h

4.8 IInput Class Reference

`#include <IInput.h>`

Inheritance diagram for IInput:



Public Member Functions

- virtual void Open (string fileName, int channel)=0
- virtual void Close ()=0
- virtual double * getSignal (int &noSamples, int &smpFreq)=0
- virtual double getNextDataPoint ()=0

4.8.1 Member Function Documentation

4.8.1.1 virtual void IInput::Close ( ) [pure virtual]

Implemented in BDFReader, and NoiseGen.

4.8.1.2 virtual double IInput::getNextDataPoint ( ) [pure virtual]

Implemented in BDFReader, and NoiseGen.

4.8.1.3 virtual double* IInput::getSignal (int & *noSamples,* int & *smpFreq*) [pure virtual]

Implemented in BDFReader, and NoiseGen.

4.8.1.4 virtual void IInput::Open (string *fileName,* int *channel*) [pure virtual]

Implemented in BDFReader, and NoiseGen.

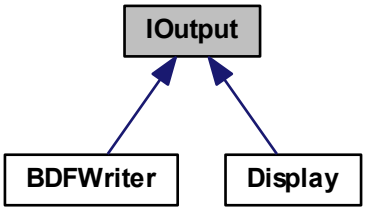
The documentation for this class was generated from the following file:

- include/IInput.h

4.9 IOutput Class Reference

```
#include <IOutput.h>
```

Inheritance diagram for IOutput:



Public Member Functions

- virtual void saveSignal (double ∗sig, int noSamp, int smp_freq, string sigLabel, string sigDim)=0
- virtual void saveNextDataPoint ()=0
- virtual void DisplayMessage (string message)=0
- virtual void DisplayHistogram (deque< int > histogram)=0

4.9.1 Member Function Documentation

4.9.1.1 virtual void IOutput::DisplayHistogram (deque< int > *histogram*) [pure virtual]

Implemented in BDFWriter, and Display.

4.9.1.2 virtual void IOutput::DisplayMessage (string *message*) [pure virtual]

Implemented in BDFWriter, and Display.

4.9.1.3 virtual void IOutput::saveNextDataPoint ( ) [pure virtual]

Implemented in BDFWriter, and Display.

4.9.1.4 virtual void IOutput::saveSignal (double ∗ *sig*, int *noSamp*, int *smp_freq*, string *sigLabel*, string *sigDim*) [pure virtual]

Implemented in BDFWriter, and Display.

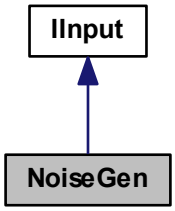
The documentation for this class was generated from the following file:

- include/IOutput.h

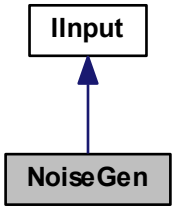
4.10 NoiseGen Class Reference

`#include <NoiseGen.h>`

Inheritance diagram for NoiseGen:



Collaboration diagram for NoiseGen:



Public Member Functions

- NoiseGen ()
- double * getSignal (int &noSamples, int &smpFreq)
- double getNextDataPoint ()
- void Open (string fileName, int channel)
- void Close ()

Private Attributes

- double * rndSignalBuf
- int sampleCounter
- int noRndSamples

4.10.1 Constructor & Destructor Documentation

4.10.1.1 NoiseGen::NoiseGen ( )

4.10.2 Member Function Documentation

4.10.2.1 void NoiseGen::Close ( ) [virtual]

Implements IInput.

4.10.2.2 double NoiseGen::getNextDataPoint ( ) [virtual]

Implements IInput.

4.10.2.3 double * NoiseGen::getSignal (int & *noSamples,* int & *smpFreq*) [virtual]

Implements IInput.

4.10.2.4 void NoiseGen::Open (string *fileName,* int *channel*) [virtual]

Implements IInput.

4.10.3 Member Data Documentation

4.10.3.1 int NoiseGen::noRndSamples [private]

4.10.3.2 double* NoiseGen::rndSignalBuf [private]

4.10.3.3 int NoiseGen::sampleCounter [private]

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

- include/NoiseGen.h
- source/NoiseGen.cpp

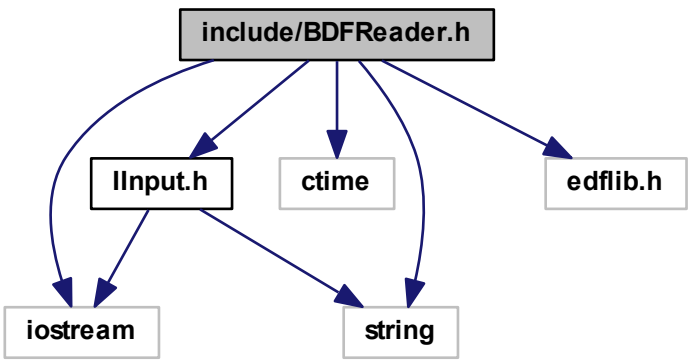
Chapter 5

File Documentation

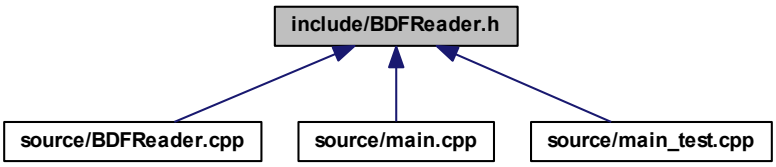
5.1 include/BDFReader.h File Reference

```
#include <iostream>
#include <string>
#include <ctime>
#include "IInput.h"
#include "edflib.h"
```

Include dependency graph for BDFReader.h:



This graph shows which files directly or indirectly include this file:



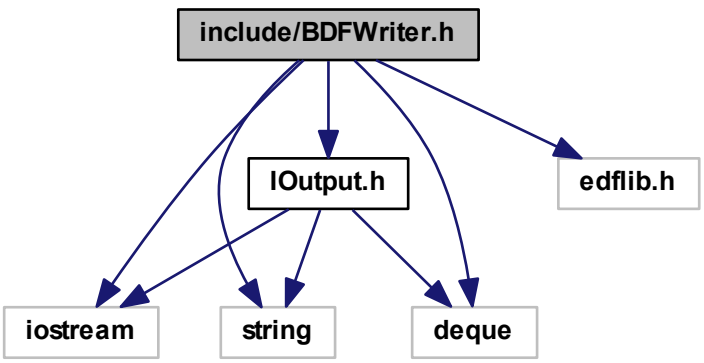
Classes

- class BDFReader

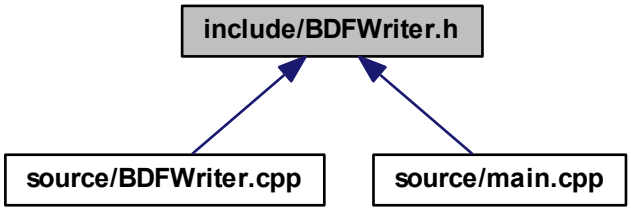
5.2 include/BDFWriter.h File Reference

```
#include <iostream>
#include <string>
#include <deque>
#include "IOutput.h"
#include <edflib.h>
```

Include dependency graph for BDFWriter.h:



This graph shows which files directly or indirectly include this file:



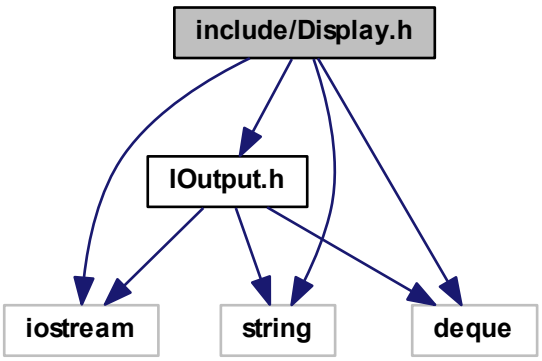
Classes

- class BDFWriter

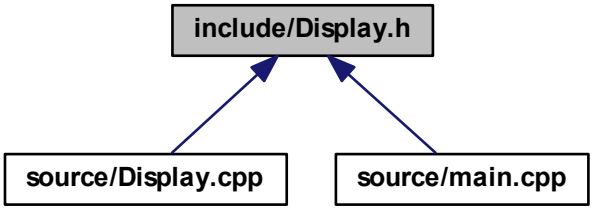
5.3 include/Display.h File Reference

```
#include <iostream>
#include <string>
#include <deque>
#include "IOutput.h"
```

Include dependency graph for Display.h:



This graph shows which files directly or indirectly include this file:



Classes

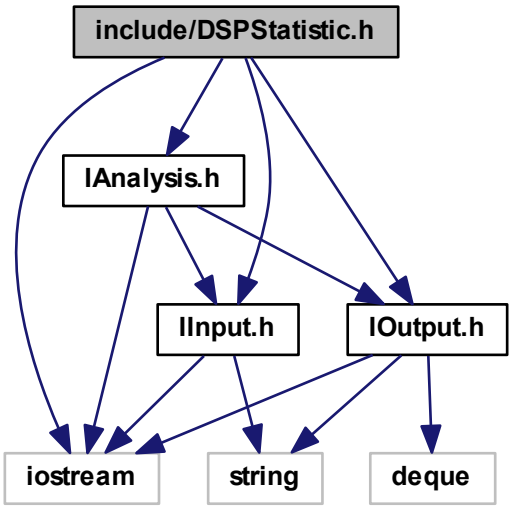
- class Display

5.4 include/DSPStatistic.h File Reference

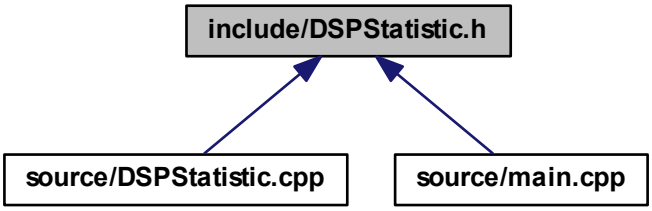
This header file defines an Analysis component using methods of DSP on signals.

```
#include <iostream>
#include "IInput.h"
#include "IOutput.h"
#include "IAnalysis.h"
```

Include dependency graph for DSPStatistic.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DSPStatistic

5.4.1 Detailed Description

This header file defines an Analysis component using methods of DSP on signals.

Author

Petar Jerčić

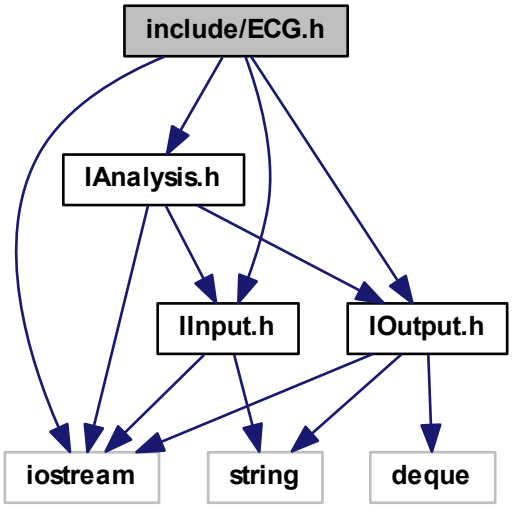
Date

09/12/2012

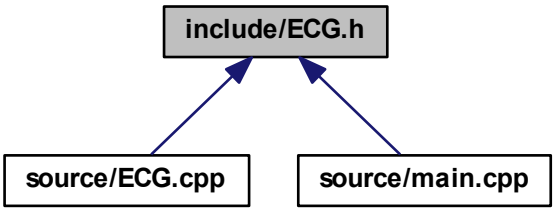
5.5 include/ECG.h File Reference

```
#include <iostream>
#include "IInput.h"
#include "IOutput.h"
#include "IAnalysis.h"
```

Include dependency graph for ECG.h:



This graph shows which files directly or indirectly include this file:



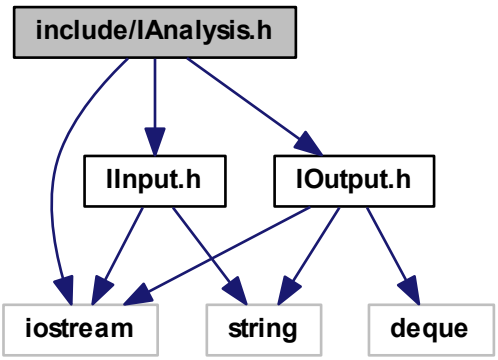
Classes

- class ECG

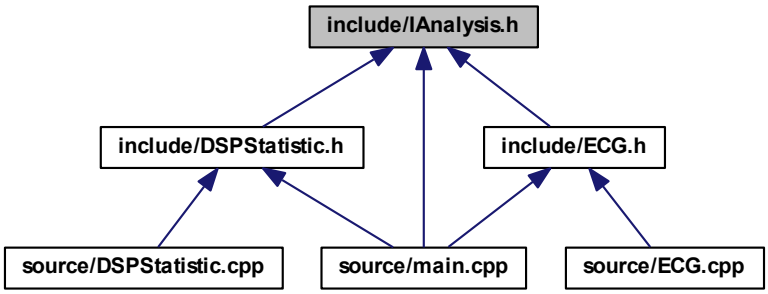
5.6 include/IAnalysis.h File Reference

```
#include <iostream>
#include "IInput.h"
#include "IOutput.h"
```

Include dependency graph for IAnalysis.h:



This graph shows which files directly or indirectly include this file:



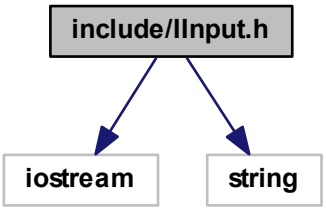
Classes

- class IAnalysis

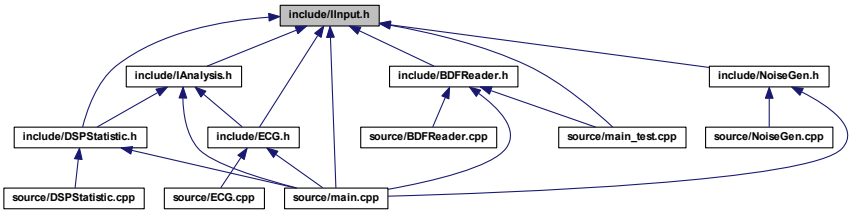
5.7 include/IInput.h File Reference

```
#include <iostream>
#include <string>
```

Include dependency graph for IInput.h:



This graph shows which files directly or indirectly include this file:



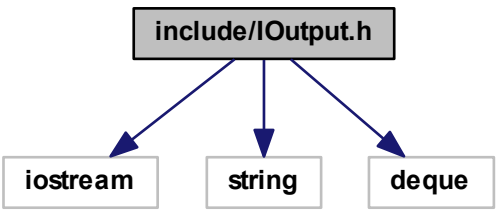
Classes

- class IInput

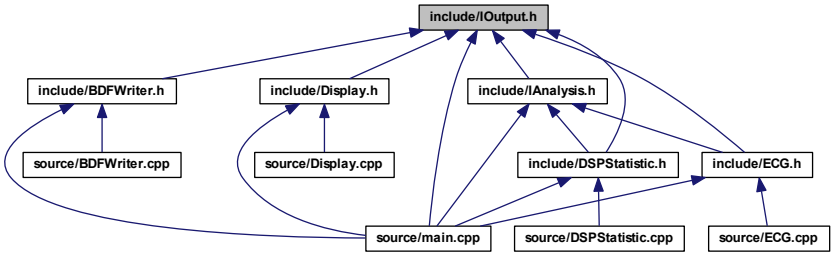
5.8 include/IOutput.h File Reference

```
#include <iostream>
#include <string>
#include <deque>
```

Include dependency graph for IOutput.h:



This graph shows which files directly or indirectly include this file:



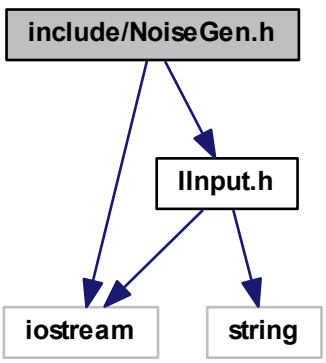
Classes

- class IOutput

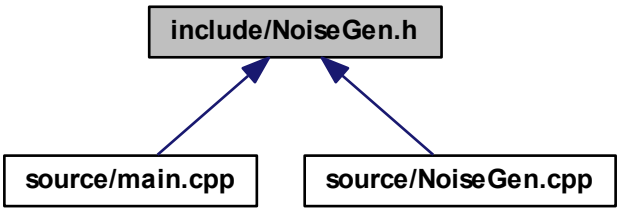
5.9 include/NoiseGen.h File Reference

```
#include <iostream>
#include <IInput.h>
```

Include dependency graph for NoiseGen.h:



This graph shows which files directly or indirectly include this file:



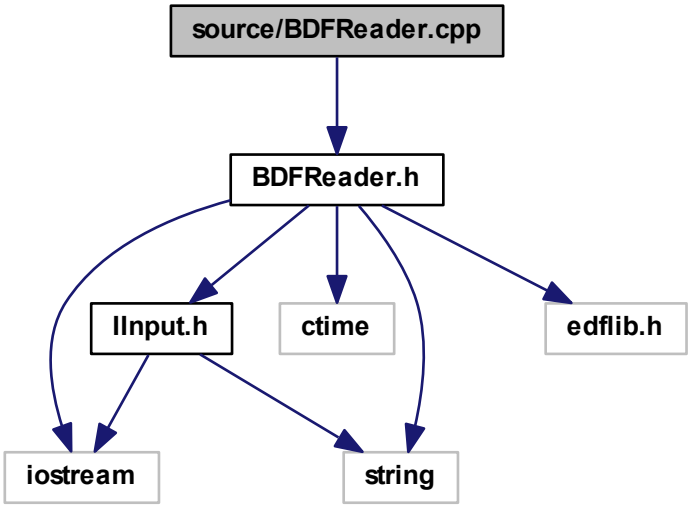
Classes

- class NoiseGen

5.10 source/BDFReader.cpp File Reference

```
#include "BDFReader.h"
```

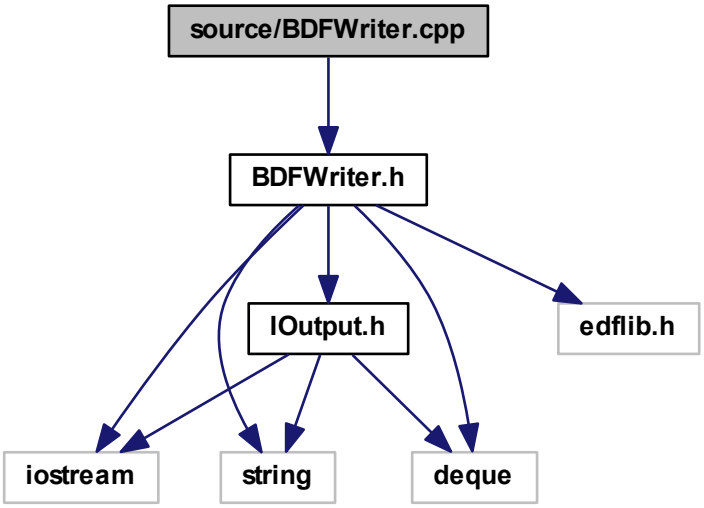
Include dependency graph for BDFReader.cpp:



5.11 source/BDFWriter.cpp File Reference

```
#include <BDFWriter.h>
```

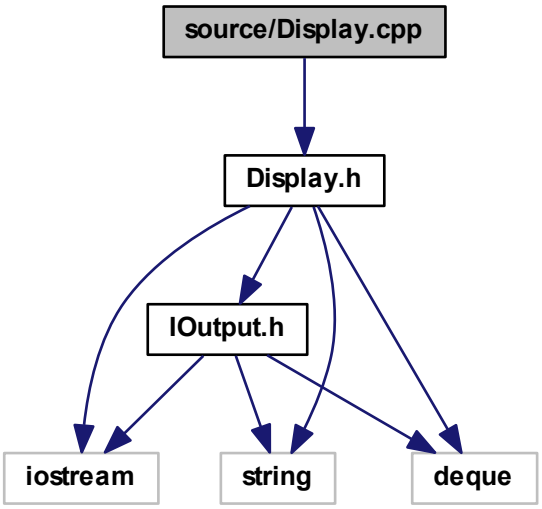
Include dependency graph for BDFWriter.cpp:



5.12 source/Display.cpp File Reference

```
#include "Display.h"
```

Include dependency graph for Display.cpp:

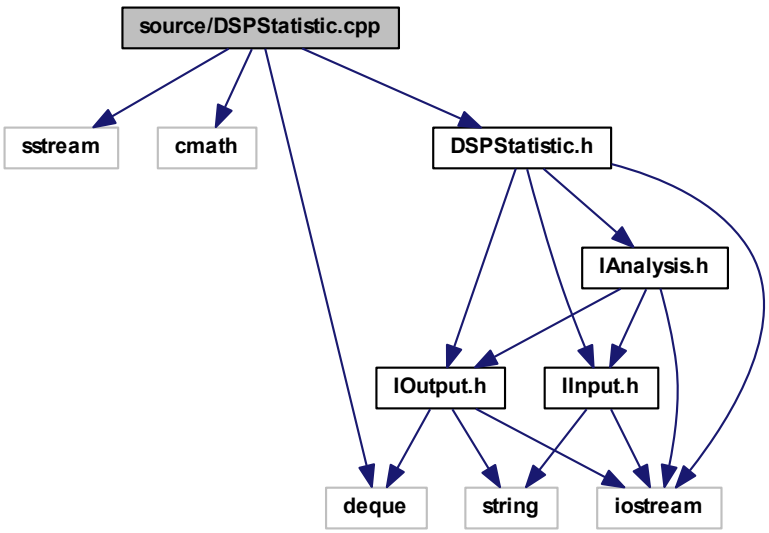


5.13 source/DSPStatistic.cpp File Reference

This source file defines an Analysis component using methods of DSP on signals.

```
#include <sstream>
#include <cmath>
#include <deque>
#include "DSPStatistic.h"
```

Include dependency graph for DSPStatistic.cpp:



5.13.1 Detailed Description

This source file defines an Analysis component using methods of DSP on signals.

Author

Petar Jerčić

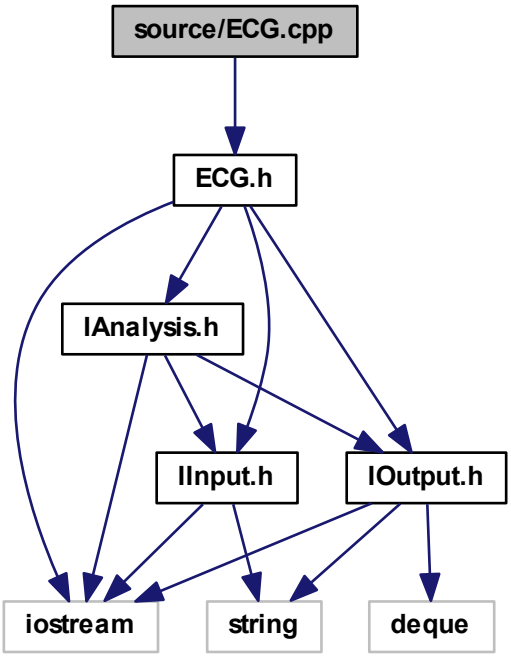
Date

09/12/2012

5.14 source/ECG.cpp File Reference

```
#include "ECG.h"
```

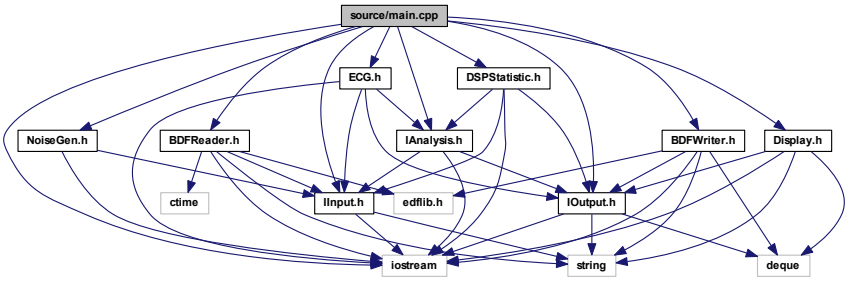
Include dependency graph for ECG.cpp:



5.15 source/main.cpp File Reference

```
#include <iostream>
#include "IInput.h"
#include "IOutput.h"
#include "IAnalysis.h"
#include "BDFReader.h"
#include "NoiseGen.h"
#include "ECG.h"
#include "DSPStatistic.h"
#include "Display.h"
#include <BDFWriter.h>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[])

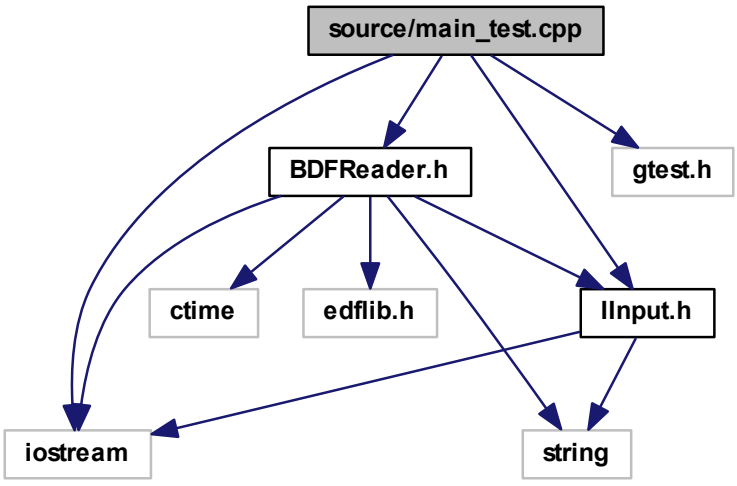
5.15.1 Function Documentation

5.15.1.1 int main (int *argc,* char ∗ *argv[]*)

5.16 source/main_test.cpp File Reference

```
#include <iostream>
#include <gtest.h>
#include <IInput.h>
#include <BDFReader.h>
```

Include dependency graph for main_test.cpp:



Functions

- TEST (BDFReader, ComapareSignalData)

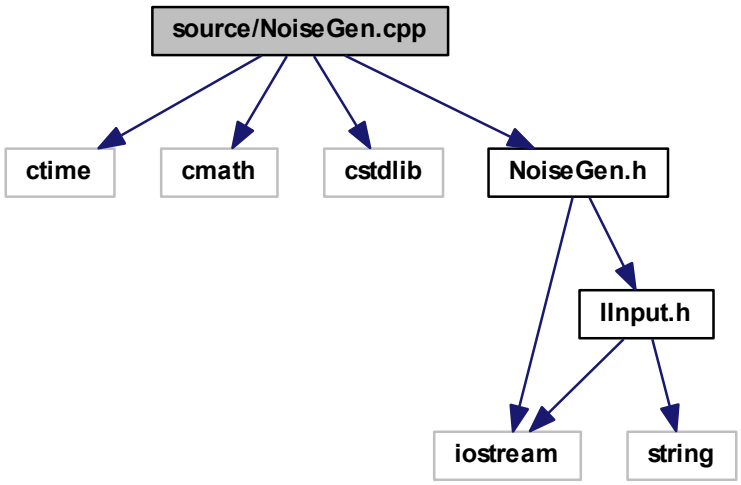
5.16.1 Function Documentation

5.16.1.1 TEST (BDFReader , ComapareSignalData)

5.17 source/NoiseGen.cpp File Reference

```
#include <ctime>
#include <cmath>
#include <cstdlib>
#include <NoiseGen.h>
```

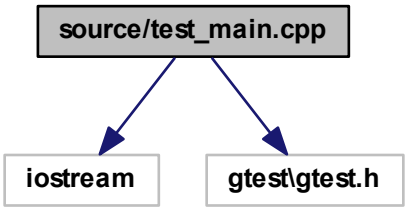
Include dependency graph for NoiseGen.cpp:



5.18 source/test_main.cpp File Reference

```
#include <iostream>
#include <gtest\gtest.h>
```

Include dependency graph for test_main.cpp:



Classes

- class Adder

Functions

- int Add (int a, int b)
- TEST (AddTest, ReturnValueEq)
- TEST (AddTest, ReturnValueDiff)

5.18.1 Function Documentation

5.18.1.1 int Add (int *a,* int *b*)

5.18.1.2 TEST (AddTest , ReturnValueEq)

5.18.1.3 TEST (AddTest , ReturnValueDiff)

Index